REPORT

과 목 명 : 데이터베이스기초
담담교수 : 이석균 교수님
소　　속 : 소프트웨어학과
학　　번 : 32151671
이　　름 : 박민혁

단국대학교
Dankook University

과제 7: 저장 프로시저, 커서, 트리거, 보안 및 실기시험 준비

이번 과제는 저장 프로시저, 커서, 그리고 트리거, 보안 등의 개념들 이해하고 사용할 수 있는 지를 확인하는 내용으로 실기시험 형식으로 진행합니다. 과제 6에서 사용한 데이터베이스를 전제로 다음 문제에 답하시오.

1. 학사DB와 유사한 데이터베이스를 생성하는데 데이터베이스 이름은 자신 이름과 학번의 마지막 숫자를 사용한다. 가령 홍길동 학생의 학번이 3219*3이라면 데이터베이스 이름은 '홍길동3'으로 정한다. '홍길동3' 데이터베이스에는 다음의 테이블들이 존재한다.**

홍길동_student3 (id3char(10),name3char(10),major3char(10),gpa3float)
홍길동_course3(cid3char(10),name3char(20),instructor3char(10))
홍길동_course_taken3(sid3char(10),cid3char(10),grade3float, year_takenint)

홍길동_student3의 major3은 전공이름으로 구성되며 student 데이터 중 학번 마지막 수가 3인 경우는 제외한다. 홍길동_course3에서 instructor3는 교수 이름으로 구성된다. 홍길동_course_taken3에서 기본 키는 밑줄 친 부분으로 수정하고 외래키 설정은 학사DB의 course_taken의 정의대로 한다.

(a) 위에서 제시한 대로 테이블들을 정의하시오.

```
DROP DATABASE IF EXISTS 박민혁1;
CREATE DATABASE 박민혁1;
USE 박민혁1;
CREATE TABLE IF NOT EXISTS 박민혁1_student
(         id3 char(10),
     name3 char(10),
     major3 char(10),
     gpa3 float,

     primary key(id3)
);
CREATE TABLE IF NOT EXISTS 박민혁1_course
(         cid3 char(10),
     name3 char(20),
     instructor3 char(10),
     primary key(cid3)
);
CREATE TABLE IF NOT EXISTS 박민혁1_course_taken
(         sid char(10),
```

```
    cid char(10),
    grade3 float,
    year_taken int,

    foreign key(sid) references 박민혁1_student(id3),
    foreign key(cid) references 박민혁1_course(cid3)
);
```

(b) 위에서 요구한 대로 학사DB로 부터 데이터를 입력하시오.

```
    INSERT INTO 박민혁1_student VALUES ('32151671', '박민혁', '소프트웨어학과', '3.7'),
('32153180', '이상민', '소프트웨어학과', '3.7');
    INSERT INTO 박민혁1_course VALUES('01', '데이터베이스기초', '이석균'), ('02', '디자이패턴', '박제호
'), ('03', '컴퓨터그래픽스', '송인식');
    INSERT INTO 박민혁1_course_taken VALUES('32151671', '01', 'A', '2020'), ('32151671', '02',
'A+', '2020'), ('32151671', '03', 'B+', '2020'),
    ('32153180' ,'02', 'A+', '2020'), ('32153180', '03', 'A+', '2020');
```

(c) 다음을 실행하시오.

```
select *
from 홍길동_student3 natural join 홍길동_course_taken3 natural join 홍길동
_course3
order by id3;
```

	name3	id3	major3	gpa3	sid	cid	grade3	year_taken	cid3	instructor3

2. 수강 테이블(홍길동_course_taken3)에 학생의 수강 과목과 성적을 입력하는 저장 프로시저 '홍길동_AddCourseGrade3'를 구현하고 이의 사용 예를 보이시오. 단 이는 매개변수로 학생 이름(pStud_name3), 과목 이름(pCo_name3), 그리고 성적(pGrade3), 그리고 pIsError3라는 int 변수를 입력 받고 프로시저 바디에서 현재의 날짜로부터 year 정보를 입력할 수 있도록 한다. pIsError는 식별할 수 없는 학생 이름이 입력되는 경우는 1을, 식별할 수 없는 과목 이름의 경우는 2를, 이 둘의 경우는 3, 그리고 그 이외의 경우는 4를 반환하도록 한다. 구현에 에러 핸들러는 반드시 포함해야 합니다.

(a) 구현 내용을 보이시오.

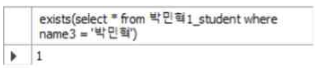
```
use 박민혁1;
drop  procedure if exists AddCourseRecord;
delimiter $$
CREATE procedure AddCourseRecord
(sname char(20),cname char(20),p_grade int, out IsError integer)
begin
    declare vsid char(10);
    declare vcid char(10);
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION  SET IsError=3;

    set IsError = 0;
    select id into vsid  from student
    where name = sname;

    select vsid;
    select id3 into vcid  from course
    where name = cname;

    if vsid is null then
              set IsError = 1;
    elseif vcid is null
          then
              set IsError = 2;
    else
        insert into course_taken(sid,cid,grade,year_taken)
        values ( vsid, vcid, p_grade , year(now()) );
    end if;
end $$
delimiter ;
```

(b) 정상 실행 경우와 각각의 에러가 발생한 경우를 보이시오.

```
call AddCourseRecord('박민혁', '데이터베이스기초', 9, @IsError);
select @IsError;
call AddCourseRecord('박민혁', '디자인패턴', 9, @IsError);
select @IsError;
select s.name3, c.name3, ct.grade3 from (박민혁1_student s join 박민혁1_course_taken ct on
s.id3 = ct.sid ) join 박민혁1_course c on ct.cid = c.cid3 and s.name3 = '박민혁';
call AddCourseRecord('박민혁', '컴퓨터그래픽스', 4, @IsError);
select @IsError;
call AddCourseRecord('이상민', '디자인패턴', 4, @IsError); select @IsError;
```

	@IsError
▶	1

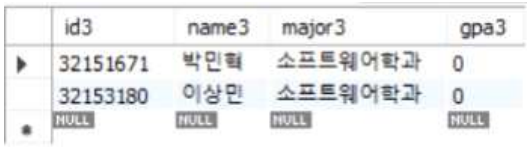
```
select id3  into  @sid from 박민혁1_student where name3 = '박민';
select @sid is null;
select exists(select * from 박민혁1_student where  name3 = '박민혁');
```



	exists(select * from 박민혁1_student where name3 = '박민혁')
▶	1

3. 학생들의 평점(GPA)을 수강 내역으로부터 계산하는 저장 프로시저(홍길동_ComputeGPA3)를 커서를 통해 구현하려고 한다. MySQL에서는 read only 커서만을 제공하므로 갱신은 update문을 사용해야 한다. 구현 내용과 실행 결과를 보이시오.

```
SET SQL_SAFE_UPDATES = 0;
Delimiter //
create procedure ComputeGPA3()
begin
    declare cursor_end bool;
     declare stu_id char(10);
    declare stu_Cursor cursor for select id from student;
     declare continue handler for not found set cursor_end=1;
     open stu_Cursor;
     cursor_loop :Loop
        Fetch stu_cursor into stu_id;
          if cursor_end then leave cursor_loop; end if;
          update 박민혁1_student set gpa3= (select avg(grade3) from 박민혁1_course_taken where
stu_id=sid group by sid);
    end Loop;
     close stu_Cursor;
end //
Delimiter ; update 박민혁1_student set gpa = 0.0;
select * from 박민혁1_student;
```



	id3	name3	major3	gpa3
▶	32151671	박민혁	소프트웨어학과	0
	32153180	이상민	소프트웨어학과	0
*	NULL	NULL	NULL	NULL

4. 수강 내역 테이블 즉 홍길동_course_taken3은 학생들의 수강 과목의 학점들이 포함되는 중요한 정보다. 따라서 수강 내역 테이블에 대한 audit 정보를 담는 홍길동_course_taken_Audit3 테이블을 정의하고 trigger들을 통해 audit 정보를 입력하려고 한다. 홍길동_Course_taken_Audit3 테이블의 필드들은 일련번호, 홍길동_course_taken3의 기본 키, 사용자 정보, 그리고 수정 시간으로 구성된다. 사용자 정보는 user() 함수의 값을 사용하도록 하는데 User()는 현 세션의 사용자를 반환한다.

(a) insert, delete, update시에 사용될 Trigger들의 정의를 보이시오.

```
CREATE TABLE if not exists course_taken_audit (
    no INT AUTO_INCREMENT PRIMARY KEY,
    no_course_taken CHAR(10),
    changedat DATETIME,
    action VARCHAR(50),
    user varchar(50)
);
SET SQL_SAFE_UPDATES = 0;
drop TRIGGER if exists after_course_taken_delete;
DELIMITER $$
CREATE TRIGGER after_course_taken_delete AFTER delete
ON 박민혁1_course_taken     FOR EACH ROW
BEGIN
INSERT INTO course_taken_audit
SET action = 'delete',
     no_course_taken = oldno,
     changedat = NOW(),
           user = user();
END $$
DELIMITER ;
drop TRIGGER if exists after_course_taken_insert;
DELIMITER $$
CREATE TRIGGER after_course_taken_insert AFTER insert
ON 박민혁1_course_taken     FOR EACH ROW
BEGIN
INSERT INTO course_taken_audit
SET action = 'insert',
```

```
no_course_taken=newno ,
     changedat = NOW(),
          user = user();
END $$
DELIMITER ;
drop TRIGGER if exists after_course_taken_update;
DELIMITER $$
CREATE TRIGGER after_course_taken_update AFTER update
ON 박민혁1_course_taken     FOR EACH ROW
BEGIN
INSERT INTO course_taken_audit
SET action = 'update',
     no_course_taken = newno,      -- new 또는 old 요구사항에 맞도록 처리할 것.
          changedat = NOW(),
          user = user();
END $$
DELIMITER ;
```

(b) 위의 trigger들이 정상 작동함을 보이는 예제들을 보이시오.

```
select * from 박민혁1_course_taken where sid = '32151671' and cid = '01';
insert into 박민혁1_course_taken(sid, cid, grade3, year_taken)
values('930405', 'ss312', 4, year(now()));
select * from 박민혁1_course_taken where sid = '930405' and cid = 'ss312';
select * from course_taken_audit;
select * from 박민혁1_course_taken where sid = '930405' and cid = 'ss312';
update 박민혁1_course_taken
set grade3 = 3
where sid = '930405' and cid = 'ss312';
select * from 박민혁1_course_taken where sid = '930405' and cid = 'ss312';
select * from course_taken_audit;
delete from 박민혁1_course_taken
where sid = '930405' and cid = 'ss312';
select * from 박민혁1_course_taken where sid = '930405' and cid = 'ss312';
select * from course_taken_audit;
```