과제 7: 저장 프로시저, 커서, 트리거, 보안 및 실기시험 준비

1분반 32193430 이재원

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 (sid3 char(10), sname3 char(10), major3 char(10), gpa3 float)
홍길동_course3(cid3 char(10), cname3 char(20), instructor3 char(10))
홍길동_course_taken3(sid3 char(10), cid3 char(10), grade3 float, year_taken3 int)

홍길동_student3의 major3은 **전공 이름**으로 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 이재원0;
create database 이재원0;
use 이재원0;


drop table if exists 이재원_student0;
create table 이재원_student0(
sid0 char(10) primary key check((sid0) % 10 <> 3),
-- sid0가 마지막 수가 3인 것을 입력하면 오류발생!, 도메인 제약조건
sname0 char(10) default null,
major0 char(10) default null,
gpa0 float
);
drop table if exists 이재원_course0;
create table 이재원_course0(
cid0 char(10) primary key,
cname0 char(20)
```

```
check(cname0 in ('황형태','조성제','조경산','이장택','이석균','이강섭','유해영','우진운', null)),
-- 교수 이름이 아닌 것을 입력하면 오류 발생!
instructor0 char(10)
);

drop table if exists 이재원_course_taken0;
create table 이재원_course_taken0(
sid0 char(10),
cid0 char(10),
grade0 float default null,
year_taken0 int default null,
primary key(sid0, cid0),
CONSTRAINT fk_CourseTaken_Student foreign key(sid0) references 이재원_student0(sid0),
CONSTRAINT fk_CourseTaken_Course foreign key(cid0) references 이재원_course0(cid0)
);
show tables;
```

Tables_in_이재원0
이재원_course0
이재원_course_taken0
이재원_student0

(b) 위에서 요구한 대로 학사DB로 부터 데이터를 입력하시오.

```
-- student table에 데이터 입력
insert into 이재원_student0 (sid0, sname0, major0, gpa0)
select s.id, s.name, d.name, s.gpa
from 학사DB.student s, 학사db.department d
where (s.id) % 10 <> 3 and s.major = d.id; -- id의 끝이 3인 경우는 제외

select * from 이재원_student0;
```

sid0	sname0	major0	gpa0
930405	이재원	전산전공	3.29
950564	허영만	전산전공	1.8
960157	이동주	전산전공	1.5
980115	이미숙	통계전공	3.6
NULL	NULL	NULL	NULL

-- course table에 데이터 입력

```
insert into 이재원_course0(cid0, cname0, instructor0)
select c.id, c.name, i.name
from 학사db.course c, 학사db.instructor i where c.instructor = i.pid;
```

```
select * from 이재원_course0;
```

cid0	cname0	instructor0
cs111	기초전산	이석균
cs221	자료구조론	우진운
cs311	컴퓨터 구조론	조경산
cs312	알고리즘	우진운
cs321	프로그래밍언어론	이석균
cs322	운영체제	조성제
cs411	데이타베이스	이석균
cs413	컴퓨터네트워크	조경산
cs421	소프트웨어 공학	유해영

이재원_course0 23

-- 이재원_course_taken0에 데이터 입력!

```
insert into 이재원_course_taken0(sid0, cid0, grade0, year_taken0)
select ct.sid, ct.cid, ct.grade, ct.year_taken
from 학사db.course_taken ct
where ct.sid in (select sid0 from 이재원_student0 )
and -- 이재원_student0과 이재원_course_taken0의 외래키 제약 조건을 지키기 위해!
ct.cid in (select cid0 from 이재원_course0);
select * from 이재원_course_taken0;
```

sid0	cid0	grade0	year_taken0
930405	cs111	2	1993
930405	cs221	3	1996
930405	cs311	3	1997
930405	cs321	4	1997
930405	cs411	4	1998
930405	ie_이재원	4	1996
950564	cs111	2	1995
950564	cs311	2	1998
950564	cs411	2	1999

이재원_course_taken0 100

(c) 다음을 실행하시오.

```
select *
```

from 홍길동_student3 natural join 홍길동_course_taken3 natural join 홍길동_course3
order by sid3;

cid0	sid0	sname0	major0	gpa0	grade0	year_taken0	cname0	instructor0
cs111	930405	이재원	전산전공	3.29	2	1993	기초전산	이석균
cs221	930405	이재원	전산전공	3.29	3	1996	자료구조론	우진운
cs311	930405	이재원	전산전공	3.29	3	1997	컴퓨터 구조론	조경산
cs321	930405	이재원	전산전공	3.29	4	1997	프로그래밍언어론	이석균
cs411	930405	이재원	전산전공	3.29	4	1998	데이타베이스	이석균
ie_이재원	930405	이재원	전산전공	3.29	4	1996	수치해석	유해영
cs111	950564	허영만	전산전공	1.8	2	1995	기초전산	이석균
cs311	950564	허영만	전산전공	1.8	2	1998	컴퓨터 구조론	조경산
cs411	950564	허영만	전산전공	1.8	2	1999	데이타베이스	이석균

Result 101 ×

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.

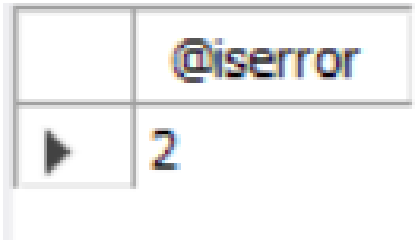
```
drop  procedure 이재원_AddCourseGrade0;

delimiter $$
create procedure 이재원_AddCourseGrade0(
in pStud_name0 char(10), in pCo_name0 char(10), in pGrade0 float, out pIsError int)
begin
Declare sid char(10);
Declare cid char(10);
Declare exit handler for sqlexception
Begin
show errors;
set pIsError = 4;
if(sid is null and cid is null) then set pIsError = 3;
elseif(sid is null) then set pIsError = 1;
elseif(cid is null) then set pIsError = 2;
end if;
End;

set sid = (select sid0 from 이재원_student0 where sname0 = pStud_name0);
set cid = (select cid0 from 이재원_course0 where cname0 = pCo_name0);
select sid, cid;
```

```
-- insert문에서 오류가 발생할 경우, error handler 실행!
insert into 이재원_course_taken0
values(sid, cid, pGrade0, year(now()));
end $$
delimiter ;
```

(b) 정상 실행 경우와 각각의 에러가 발생한 경우를 보이시오.

```
select * from 이재원_course_taken0;
-- 에러가 발생한 경우 1 sid0가 null 나머지는 올바른 값
call 이재원_AddCourseGrade0(null, '기초전산', 4.5, @iserror);
select @iserror;
```


```
-- 에러가 발생한 경우 2 cid0가 null 나머지는 올바른 값
call 이재원_AddCourseGrade0('이재원', null, 4.5, @iserror);
select @iserror;
```




```
-- 에러가 발생한 경우 3 sid0, cid0가 null
call 이재원_AddCourseGrade0(null, null, 4.5, @iserror);
select @iserror;
```


```
-- 에러가 발생한 경우 4 나머지 에러
call 이재원_AddCourseGrade0('이재원', '기초전산', 4.5, @iserror);
call 이재원_AddCourseGrade0('이재원', '기초전산', 4.5, @iserror);
select @iserror;
```

	@iserror
▶	4

-- 정상적인 실행 모두 다 올바른 값

```
delete from 이재원_course_taken0 where sid0 = '930405' and cid0 = 'cs111';
call 이재원_AddCourseGrade0('이재원', '기초전산', 4.5, @iserror);
select @iserror;
select * from 이재원_course_taken0;
```

	sid0	cid0	grade0	year_taken0
▶	930405	cs111	4.5	2021
	930405	cs221	3	1996
	930405	cs311	3	1997
	930405	cs321	4	1997
	930405	cs411	4	1998
	930405	ie_이재원	4	1996
	950564	cs111	2	1995
	950564	cs311	2	1998
	950564	cs411	2	1999

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

```
drop procedure 이재원_ComputeGPA0;
delimiter //
create procedure 이재원_ComputeGPA0()
begin
declare sid char(10);
declare gpa decimal(5,2); -- gpa 선언
declare v_finished int default 0;
declare GpaCursor cursor for
select sid0, round(avg(grade0),2) from 이재원_course_taken0 group by sid0;

declare continue handler for not found
set v_finished = 1;
```

```
open GpaCursor;
fetch GpaCursor into sid, gpa;
while(v_finished = 0) do
update 이재원_student0 -- 계산한 내용을 student table에 반영!
set gpa0 = gpa
where sid0 = sid;
select sid, gpa;
fetch GpaCursor into sid, gpa;
end while;
close GpaCursor;
end //
delimiter ;

call 이재원_ComputeGPA0;
select * from 이재원_student0;
```

	sid0	sname0	major0	gpa0
▶	930405	이재원	전산전공	3.29
	950564	허영만	전산전공	1.8
	960157	이동주	전산전공	1.5
	980115	이미숙	통계전공	3.6
*	NULL	NULL	NULL	NULL

call 이재원_ComputeGPA0; 이전

	sid0	sname0	major0	gpa0
▶	930405	이재원	전산전공	3.33
	950564	허영만	전산전공	2
	960157	이동주	전산전공	1.5
	980115	이미숙	통계전공	3.5
*	NULL	NULL	NULL	NULL

이재원 student0 18 ×

call 이재원_ComputeGPA0; 이후

	sid	gpa
▶	930405	3.33

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, 실행 연산, 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.

```
-- 1. insert trigger 정의
drop trigger if exists before_course_taken_insert;
delimiter //
create trigger before_course_taken_insert
before insert on 이재원_course_taken0 for each row
begin
insert into 이재원_course_taken_Audit0(sid, cid, userinfo, action, changetime)
values(new.sid0, new.cid0, user(), 'insert', now());
end //
delimiter ;

-- 2. delete trigger 정의
delimiter $$
create trigger before_course_taken_delete
before delete on 이재원_course_taken0 for each row
begin
insert into 이재원_course_taken_Audit0(sid, cid, userinfo, action, changetime)
values(old.sid0, old.cid0, user(), 'delete', now());
end $$
delimiter ;

-- 3. update trigger 정의
delimiter $$
create trigger before_course_taken_update
```

```
before update on 이재원_course_taken0 for each row
begin
insert into 이재원_course_taken_Audit0
set sid = old.sid0, cid = old.cid0, userinfo = user(),
action = 'update', changetime = now();
end $$
delimiter ;
```

(b) 위의 trigger들이 정상 작동함을 보이는 예제들을 보이시오.

```
select * from 이재원_course_taken0;
delete from 이재원_course_taken0 where sid0 = '930405' and cid0 = 'ie_이재원';
insert into 이재원_course_taken0 values('930405', 'ie_이재원', '4', '1996');
update 이재원_course_taken0
set grade0 = 4
where sid0 = '930405' and cid0 = 'ie_이재원' ;
select * from 이재원_course_taken_Audit0;
```

id	sid	cid	userinfo	action	changetime
1	930405	ie_이재원	root@localhost	delete	2021-06-04 16:08:02
2	930405	ie_이재원	root@localhost	insert	2021-06-04 16:08:06
3	930405	ie_이재원	root@localhost	update	2021-06-04 16:08:10
NULL	NULL	NULL	NULL	NULL	NULL

5. 장학생(홍길동_ScholarshipStudent3) 테이블을 view로 정의하려고 한다.

(a) 이를 정의하고 실행 결과를 보이시오.

```
drop view 이재원_ScholarshipStudent0;
create view 이재원_ScholarshipStudent0(id, name, gpa) as
select sid0, sname0, gpa0
from 이재원_student0
where gpa0 >= 3
-- gpa가 3이상인 경우 장학생으로 선발!
with check option;
select * from 이재원_ScholarshipStudent0;
```

id	name	gpa
930405	이재원	3.33
980115	이미숙	3.5

(b) MySQL의 Check Option의 의미를 설명하고 이를 장학생 뷰에 적용했을 때 어떤 변화를 보이는 지를 보이시오.

```
/*
view를 정의할 때 with Check Option을 적용하면
where절의 조건을 만족하는 경우에만 view에 대한 insert, delete, update가 가능하다.
*/
delete from 이재원_ScholarshipStudent0 where id = 32193430;
insert into 이재원_ScholarshipStudent0 values
(32193430,'이재원', '4.5'); -- 이런 경우는 가능!
select * from 이재원_ScholarshipStudent0;
insert into 이재원_ScholarshipStudent0 values
(32193430,'이재일', '2'); -- 오류!

update 이재원_ScholarshipStudent0
set gpa = '2.5'
where name = '이미숙'; -- 오류!
```

	id	name	gpa
▶	32193430	이재원	4.5
	930405	이재원	3.33
	980115	이미숙	3.5

Error Code: 1369. CHECK OPTION failed '이재원0.이재원_scholarshipstudent0'

6. 학사DB의 사용을 위한 사용자 계정으로 student, professor, academicManager를 생성한다. academicManager는 학사DB 전체에 대한 모든 접근 권한과 이에 권한 부여 능력을 갖는다. 한편, student는 모든 테이블들에 대한 select 권한만을, professor는 모든 테이블들에 대한 select 권한, 그리고 course_taken에 대해 insert, update 권한만을 갖도록 한다. Root 사용자 세션에서 실행했을 때, 위의 작업이 가능한 script 파일을 생성하고 show grants for 문을 통해 각 사용자의 권한을 확인하도록 한다. 스크립트 파일의 내용과 실행화면을 출력하여 제출하시오.

```
-- 유저 생성
create user student@localhost identified by "student32193430";
create user professor@localhost identified by "professor32193430";
create user academicManager@localhost identified by "academicManager32193430";
-- 1) academicManager에 학사DB 전체에 대한 모든 접근 권한과  권한 부여 능력을 부여
grant all
on 학사DB.*
```

```
to academicManager@localhost
with grant option;

-- 2) student에 모든 테이블들에 대한 select 권한을 부여
grant select
on 학사DB.*
to student@localhost;

-- 3) professor는 모든 테이블들에 대한 select 권한, course_taken에 대해 insert, update 권한을 부여
grant select
on 학사DB.*
to professor@localhost;

grant insert, update
on 학사DB.course_taken
to professor@localhost;

-- 권한을 확인
show grants for academicManager@localhost;
show grants for student@localhost;
show grants for professor@localhost;
```