preocedure - 메서드와 비슷한 기능

프로시저의 장점

1. 하나의 요청으로 여러 SQL문을 실행할 수 있다.(네트워크에 대한 부하를 줄일 수 있다.)
2. 미리 구문 분석 및 내부 중간 코드로 변환을 끝내야 하므로 처리 시간이 줄어든다.
3. 데이터베이스 트리거와 결합하여 복잡한 규칙에 의한 데이터의 참조무결성 유지가 가능하게 된다. 간단히 말하면 응용 프로그램 측 로직을 가지지 않고도 데이터베이스의 데이터 앞뒤가 맞게 될 수 있다.
4. 보수성이 뛰어 나다.

프로시저 목록 확인 방법

show procedure status;

프로시저 확인 방법

show create procedure 프로시저명;

declare - 지역변수를 선언하는 명령어

프로시저 정의 방법

DROP PROCEDURE IF EXISTS search\_user; -> search\_user라는 이름의 프로시저가 있다면 지운다

DELIMITER // -> delimiter 후 기호를 쓰면 해당 기호를 한번 더 만날 때까지 실행하지 않는다

CREATE PROCEDURE search\_user() -> search\_user라는 이름의 프로시저를 생성

BEGIN

SELECT USER,HOST FROM MYSQL.USER; -> 해당 쿼리문을 실행하는 기능

END // -> 앞에서 쓴 //를 만났으므로 실행된다

DELIMITER ;

show procedure status; -> 현재 프로시저를 확인

if 사용법

SET total = in\_mid + in\_end + in\_att + in\_rep; -> 변수 선언

IF total >= 95 AND total <= 100 THEN SET grade = 'A+';

END IF;

IF total >=90 AND total < 95 THEN SET grade = 'A';

END IF;

else if 사용법

SET total = in\_mid + in\_end + in\_att + in\_rep; -> 변수 선언

IF total >= 95 AND total <= 100 THEN

SET grade = 'A+';

ELSE

IF total >= 90 THEN

SET grade = 'A';

END IF;

END IF;

학점을 계산하는 프로시저 생성

DROP PROCEDURE IF EXISTS calculate\_grade;

DELIMITER //

CREATE PROCEDURE calculate\_grade(

IN in\_mid DOUBLE,

IN in\_end DOUBLE,

IN in\_att INT,

IN in\_rep INT,

IN in\_class\_num INT,

IN in\_student\_num INT)

BEGIN

DECLARE total DOUBLE DEFAULT 0;

DECLARE grade VARCHAR(2);

SET total = in\_mid + in\_end + in\_att + in\_rep;

IF total >= 95 AND total <= 100 THEN

SET grade = 'A+';

END IF;

IF total >=90 AND total < 95 THEN

SET grade = 'A';

END IF;

IF total >=85 AND total < 90 THEN

SET grade = 'B+';

END IF;

IF total >=80 AND total < 85 THEN

SET grade = 'B';

END IF;

IF total >=75 AND total < 80 THEN

SET grade = 'C+';

END IF;

IF total >=70 AND total < 75 THEN

SET grade = 'C';

END IF;

IF total >=65 AND total < 70 THEN

SET grade = 'D+';

END IF;

IF total >=60 AND total < 65 THEN

SET grade = 'D';

END IF;

IF total >=0 AND total < 60 THEN

SET grade = 'F';

END IF;

UPDATE

university.course

SET

course\_mid = in\_mid,

course\_end = in\_end,

course\_report = in\_rep,

course\_attendance = in\_att,

course\_total = total,

course\_grade = grade

WHERE course\_student\_num = in\_student\_num

AND

course\_class\_num = in\_class\_num

AND

course\_num >= 1;

END //

DELIMITER ;

프로시져 호출 방법

call 프로시저명(컬럼값);

->call calculate\_grade(35,30,10,10,2,201910101);

예제)out을 이용해 값을 생성하여 리턴

프로시저 생성

drop procedure if exists getNo;

delimiter //

create procedure getNo(

out no varchar(30) -- no라는 변수에 varchar(30)타입의 값을 out한다

) -- in 또는 out에서 선언한 변수는 declare를 통해 다시 지역변수로 선언할 수 없다

begin

set no = '123'; -- no에 '123'을 돌려주는 코드

end //

delimiter ;

프로시저 호출 @는 환경변수로서 어디서든 사용할 수 있다

call getNo(@no); -> 환경변수 no에 getNo프로시저에서 새로 리턴한 no값을 저장한다

select @no; ->select를 통해 @no에 저장된 값을 불러온다

예제) graduation 테이블에서 학생 정보가 주어졌을 때 graduation\_check에 값을 결정하는 프로시저 작성

USE university;

DROP PROCEDURE IF EXISTS update\_graduation;

DELIMITER //

CREATE PROCEDURE update\_graduation(

in in\_student\_num INT,

In in\_requirement\_num Int)

BEGIN

DECLARE s\_english, r\_english INT;

DECLARE s\_project, r\_project VARCHAR(3);

DECLARE s\_paper, r\_paper VARCHAR(3);

DECLARE s\_test, r\_test VARCHAR(3);

DECLARE s\_certificate, r\_certificate VARCHAR(3);

SET s\_english = (SELECT graduation\_english FROM graduation WHERE graduation\_student\_num = in\_student\_num);

SET s\_project = (SELECT graduation\_project FROM graduation WHERE graduation\_student\_num = in\_student\_num);

SET s\_paper = (SELECT graduation\_paper FROM graduation WHERE graduation\_student\_num = in\_student\_num);

SET s\_test = (SELECT graduation\_test FROM graduation WHERE graduation\_student\_num = in\_student\_num);

SET s\_certificate = (SELECT graduation\_certificate FROM graduation WHERE graduation\_student\_num = in\_student\_num);

SET r\_english = (SELECT requirement\_english FROM requirement WHERE requirement\_num = in\_requirement\_num);

SET r\_project = (SELECT requirement\_project FROM requirement WHERE requirement\_num = in\_requirement\_num);

SET r\_paper = (SELECT requirement\_paper FROM requirement WHERE requirement\_num = in\_requirement\_num);

SET r\_test = (SELECT requirement\_test FROM requirement WHERE requirement\_num = in\_requirement\_num);

SET r\_certificate = (SELECT requirement\_certificate FROM requirement WHERE requirement\_num = in\_requirement\_num);

IF r\_english <= s\_english

AND ( r\_paper = 'N' || (r\_paper = 'Y' AND s\_paper='Y') )

AND ( r\_test = 'N' || (r\_test = 'Y' AND s\_test='Y') )

AND ( r\_project = 'N' || (r\_project = 'Y' AND s\_project='Y') )

AND ( r\_certificate = 'N' || (r\_certificate = 'Y' AND s\_certificate='Y') ) THEN

UPDATE graduation

SET graduation\_ok = 'Y'

WHERE graduation\_student\_num = in\_student\_num;

ELSE

UPDATE graduation

SET graduation\_ok = 'N'

WHERE graduation\_student\_num = in\_student\_num;

END IF;

END //

DELIMITER ;

Gym 프로시저 생성

A회원 정보를 검색하는 프로시저로 변환

drop procedure if exists search\_member\_program;

delimiter //

create procedure search\_member\_program( in in\_member\_id varchar(45) )

begin

SELECT member.member\_name as '이름',

course.course\_no as '수강번호',

detail.member\_name as '담당 트레이너 이름',

course\_period as '수강 기간',

program.program\_title as '프로그램명',

program.program\_contents as '프로그램 내용',

program.program\_schedule as '프로그램 일정',

program.program\_total\_limit as '최대 인원',

program.program\_total as '현재 인원’

FROM member

join course

on member.member\_course\_no = course.course\_no

join program

on course.course\_program\_no = program.program\_no

join (select \* from trainer join member on trainer.trainer\_member\_id = member.member\_id) as detail

on program.program\_trainer\_no = detail.trainer\_no

where member.member\_id = in\_member\_id;

end //

delimiter ;

show procedure status;

현재 날짜를 기준으로 yyyymmdd의 형태로 정수를 생성하는 프로시저 생성

drop procedure if exists getNo;

delimiter //

create procedure getNo(

out no varchar(30)

)

begin

declare year varchar(4);

declare month varchar(2);

declare day varchar(2);

set year = year(now());

set month = month(now());

set day = day(now());

if month < 10 then

set month = concat('0', month);

end if;

if day < 10 then

set day = concat('0', day);

end if;

set no = concat(year, month, day);

end //

delimiter ;

트리거와 프로시저의 차이

* 트리거는 이벤트가 발생한 테이블의 다른 이벤트를 처리할 수 없다

ex) A테이블의 insert 트리거를 작성하는데 A테이블을 update하는 코드를 작성하면 에러가 발생

* 프로시저는 call을 통해 호출해야 하지만 트리거는 해당 이벤트가 발생하면 자동으로 호출된다