※ 스토어드 프로시저는 MySQL에서 프로그래밍 기능을 필요할 때 사용하는 데이터베이스 개체

-> SQL 프로그래밍은 기본적으로 스토어드 프로시저 안에 만들어야 함

DELIMITER $$

CREATE PROCEDURE 스토어드\_프로시저\_이름() // 스토어드 프로시저 생성

BEGIN

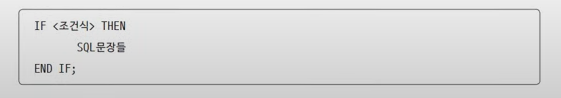
// 이 부분에 원하는 프로그래밍 코딩

END $$ // 스토어드 프로시저 종료

DELIMITER ; // 종료 문자를 다시 세미콜론(;)으로 변경

CALL 스토어드\_프로시저\_이름() // 스토어드 프로시저 실행

# 1. IF문



Ex)

Use market\_db;

DROP PROCEDURE IF EXISTS ifProc1; // 기존에 만든 적이 있다면 삭제

DELIMITER $$

CREATE PROCEDURE ifProc1()

BEGIN

IF 100 = 100 THEN

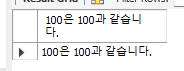
SELECT '100은 100과 같습니다.';

END IF;

END $$

DELIMITER ;

CALL ifProc1();



# 2. IF~ELSE문

Ex)

DROP PROCEDURE IF EXISTS ifProc2;

DELIMITER $$

CREATE PROCEDURE ifProc2()

BEGIN

DECLARE myNum INT; -- myNum 변수선언 // 스토어드 프로시저 밖에서 변수를 선언할 때는 SET @myNum이런 식으로 선언하지만 스토어드 프로시저 안에서 변수를 선언할 때는 DECLARE를 사용해서 선언함

SET myNum = 200; // 변수에 값 대입

IF myNum = 100 THEN

SELECT '100입니다.';

ELSE

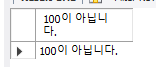
SELECT '100이 아닙니다.';

END IF;

END $$

DELIMITER ;

CALL ifProc2();



Ex2) 에이핑크가 데뷔한지 5년이 지났는지 아닌지에 따라서 문구 다르게 출력하기

DROP PROCEDURE IF EXISTS ifProc3;

DELIMITER $$

CREATE PROCEDURE ifProc3()

BEGIN

DECLARE debutDate DATE; // 데뷔일

DECLARE curDate DATE; // 오늘

DECLARE days INT; // 활동한 일수

SELECT debut\_date INTO debutDate // debut\_date 결과를 debutDate 변수에 대입

FROM market\_db.member

WHERE mem\_id = 'APN';

SET curDATE = CURRENT\_DATE(); // 현재 날짜

SET days = DATEDIFF(curDATE, debutDate); // 날짜의 차이, 일 단위

IF (days/365) >= 5 THEN // 5년이 지났다면

SELECT CONCAT('데뷔한지 ', days, '일이나 지났습니다. 핑순이들 축하합니다!');

ELSE

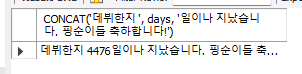
SELECT '데뷔한지 ' + days + '일밖에 안되었네요. 핑순이들 화이팅~' ;

END IF;

END $$

DELIMITER ;

CALL ifProc3();



# 3. case문

- IF문은 참 아니면 거짓 두 가지만 있기 때문에 2중 분기라는 용어를 사용하는데 CASE문은 2가지 이상의 여러 가지 경우일 때 처리가 가능하므로 ‘다중 분기’라고 부른다

Ex) 학점 등급

DROP PROCEDURE IF EXISTS caseProc;

DELIMITER $$

CREATE PROCEDURE caseProc()

BEGIN

DECLARE point INT ; // 학점

DECLARE credit CHAR(1); // 등급

SET point = 88 ;

CASE

WHEN point >= 90 THEN

SET credit = 'A';

WHEN point >= 80 THEN

SET credit = 'B';

WHEN point >= 70 THEN

SET credit = 'C';

WHEN point >= 60 THEN

SET credit = 'D';

ELSE

SET credit = 'F';

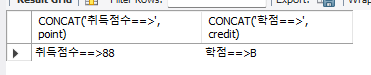
END CASE;

SELECT CONCAT('취득점수==>', point), CONCAT('학점==>', credit);

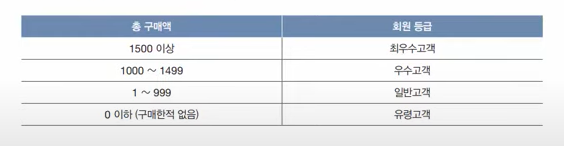
END $$

DELIMITER ;

CALL caseProc();



Ex2) 회원들의 총 구매액을 계산해서 회원의 등급 나누기



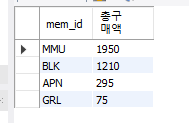
1. 먼저 총 구매액을 구한다

SELECT mem\_id, SUM(price\*amount) "총구매액"

FROM buy

GROUP BY mem\_id

ORDER BY SUM(price\*amount) DESC ;



2. 아이디와 함께 회원 테이블에 있는 이름을 알아야 하기 때문에 JOIN을 사용한다

SELECT B.mem\_id, M.mem\_name, SUM(price\*amount) "총구매액"

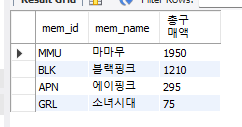
FROM buy B

INNER JOIN member M

ON B.mem\_id = M.mem\_id

GROUP BY B.mem\_id

ORDER BY SUM(price\*amount) DESC ;



3. 유령 고객(한번도 구매하지 않은 고객)의 정보는 나오지 않기 때문에 OUTER JOIN을 사용

SELECT M.mem\_id, M.mem\_name, SUM(price\*amount) "총구매액"

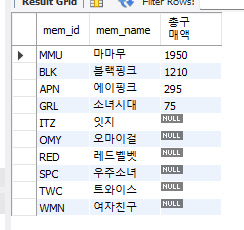
FROM buy B

RIGHT OUTER JOIN member M

ON B.mem\_id = M.mem\_id

GROUP BY M.mem\_id

ORDER BY SUM(price\*amount) DESC ;



4. 이제 CASE문을 섞어서 사용하기

SELECT M.mem\_id, M.mem\_name, SUM(price\*amount) "총구매액",

CASE

WHEN (SUM(price\*amount) >= 1500) THEN '최우수고객'

WHEN (SUM(price\*amount) >= 1000) THEN '우수고객'

WHEN (SUM(price\*amount) >= 1 ) THEN '일반고객'

ELSE '유령고객'

END "회원등급"

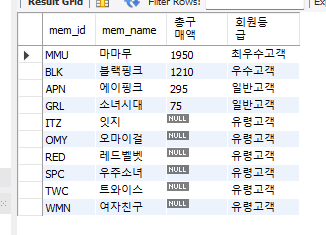
FROM buy B

RIGHT OUTER JOIN member M

ON B.mem\_id = M.mem\_id

GROUP BY M.mem\_id

ORDER BY SUM(price\*amount) DESC ;



# 4. WHILE 반복문

Ex) 1~100까지의 합

DROP PROCEDURE IF EXISTS whileProc;

DELIMITER $$

CREATE PROCEDURE whileProc()

BEGIN

DECLARE i INT; -- 1에서 100까지 증가할 변수

DECLARE hap INT; -- 더한 값을 누적할 변수

SET i = 1;

SET hap = 0;

WHILE (i <= 100) DO

SET hap = hap + i; -- hap의 원래의 값에 i를 더해서 다시 hap에 넣으라는 의미

SET i = i + 1; -- i의 원래의 값에 1을 더해서 다시 i에 넣으라는 의미

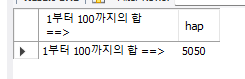
END WHILE;

SELECT '1부터 100까지의 합 ==>', hap;

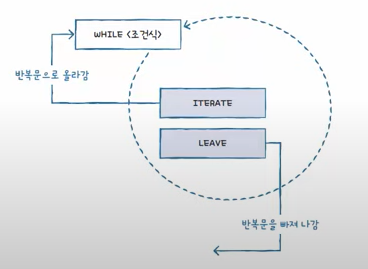
END $$

DELIMITER ;

CALL whileProc();



Ex2)



DROP PROCEDURE IF EXISTS whileProc2;

DELIMITER $$

CREATE PROCEDURE whileProc2()

BEGIN

DECLARE i INT; -- 1에서 100까지 증가할 변수

DECLARE hap INT; -- 더한 값을 누적할 변수

SET i = 1;

SET hap = 0;

myWhile: // 임의로 지어준 이름

WHILE (i <= 100) DO -- While문에 label을 지정

IF (i%4 = 0) THEN

SET i = i + 1;

ITERATE myWhile; -- 지정한 label문으로 가서 계속 진행

END IF;

SET hap = hap + i;

IF (hap > 1000) THEN

LEAVE myWhile; -- 지정한 label문을 떠남. 즉, While 종료.

END IF;

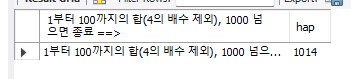
SET i = i + 1;

END WHILE;

SELECT '1부터 100까지의 합(4의 배수 제외), 1000 넘으면 종료 ==>', hap;

END $$

DELIMITER ;

CALL whileProc2();  


# 5. PREPARE와 EXECUTE

- PREPARE는 SQL문을 실행하지는 않고 미리 준비만 해 놓는다

- EXECUTE는 준비한 SQL문을 실행

- 미리 SQL문을 준비한 후에 나중에 실행하는 것을 동적SQL이라고 부름

EX)

CREATE TABLE gate\_table (id INT AUTO\_INCREMENT PRIMARY KEY, entry\_time DATETIME);

SET @curDate = CURRENT\_TIMESTAMP(); -- 현재 날짜와 시간

PREPARE myQuery FROM 'INSERT INTO gate\_table VALUES(NULL, ?)';

EXECUTE myQuery USING @curDate;

DEALLOCATE PREPARE myQuery; // 다 사용하고 나면 닫아준다

SELECT \* FROM gate\_table;

