```
REM Author : Henry
REM Date : 2024.06.12
REM Objective :
REM Environment : Ubuntu Server 22.04 LTS, MySQL Workbench 8.0 CE, MySQL Community Server 8.0.37-0ubuntu0.22.04.3 (ubuntu)


REM Stored Programs
1. 잘 정리된 논리적 코드분할이름
2. 컴파일된 상태에서 데이터베이스에 저장되기 때문에 성능이 향상된다.
3. 테이블이름이나 컬럼의 이름을 명시하지 않기 때문에 보안에 도움이 된다.
4. 모듈화를 통한 관리 용이
5. SQL 문으로 구성된 본문이 있다.
6. 본문은 세미콜론 문자로 구분된 여러 SQL문으로 구성된다.
7. 종류
   1)Stored Procedures   <-----Java 에서 사용
   2)Stored Functions


REM Stored PROCEDURE(저장프로시저)
1. 목적 : 속도, 보안
2. compile 상태로 RDBMS 에 저장
3. 나중에 실행될 일련의 명령의 집합
4. Syntax
   DELIMITER //
   CREATE PROCEDURE procedure_name
   (
      [IN | OUT | IN OUT]  param_name  type
   )
   BEGIN
      SQL 문장들
   END
   //
   DELIMITER;

5. Parameter Mode  : 3가지
   -IN : 입력 매개변수
   -OUT : 출력 매개변수
   -IN OUT : 입력, 출력 매개변수

6. Examples

   delimiter //
   CREATE PROCEDURE helloworld()
   BEGIN
      SELECT 'Hello, World';
   END
   //
   delimiter ;

   CALL helloworld();
```

이미 컴파일 단계에서 에러 여부를 판단했기 때문에
이후에는 검증을 하지 않아 속도가 빠르다.

```
CREATE PROCEDURE procedure_name(parameter...)
BEGIN
   SQL 구문 ...
END
와 같은 구조로 Function과의 큰 차이점은 리턴이 없다는 것이다.

procedure는 주로 코드를 묶어 실행할 때 사용
(OUT으로 출력을 보내 Function처럼 사용할 수 는 있다.)

반대로 Function은 리턴이 있어야 한다. -> 프로시저만 배워도
Function 구성 가능.
```

```
IN은 입력 매개변수( 기본값이다.)
OUT은 출력 매개변수 (Return이 없는 대신 존재)
INOUT은 입력으로 사용되며 출력으로 나가는? 같은 변수로 입, 출력 받을 때 사용
```

```
BEGIN과 END 사이의 SQL은
오류없이 수행 가능한 구문이어야 한다.
```

```
51
52
53      delimiter //
54      CREATE PROCEDURE test_proc()
55      BEGIN
56          SET @v_name = '백두산';
57          SELECT CONCAT('My name is ', @v_name);
58      END
59      //
60      delimiter ;
61
62      CALL test_proc();
63
64
65      -- emp 테이블이 모든 데이터를 삭제하는 Stored Procedure 를 작성하시오.
66      delimiter //
67      CREATE PROCEDURE del_all()
68          BEGIN
69      DELETE FROM emp_copy;
70      END
71      //
72      delimiter ;
73
74      CALL del_all();
75
76
77  7. 확인하기
78      DESC INFORMATION_SCHEMA.ROUTINES;
79
80      SELECT * FROM INFORMATION_SCHEMA.ROUTINES
81      WHERE specific_name = 'helloworld';
82
83
84  8. IN 매개변수
85
86      delimiter //
87      CREATE PROCEDURE test_proc_in(IN p_name VARCHAR(30))
88      BEGIN
89          SELECT CONCAT('My name is ', p_name);
90      END
91      //
92      delimiter ;
93
94      CALL test_proc_in('백두산');
95
96
97      --사원번호와 봉급을 입력받아 업데이트하는 Procedure를 완성하시오.
98
99      delimiter //
100     CREATE PROCEDURE emp_sal_update(v_empno SMALLINT, v_sal FLOAT)
101     BEGIN
```

IN은 기본 값이므로 생략 가능.

```
102         UPDATE emp SET sal = v_sal
103         WHERE empno = v_empno;
104         COMMIT;
105     END
106     //
107     delimiter ;
108
109     CALL emp_sal_update(7369, 1000);
110
111
112     --사번을 받아 삭제하는 프로시저
113     delimiter //
114     CREATE PROCEDURE emp_del(v_empno SMALLINT)
115     BEGIN
116         DELETE FROM emp
117         WHERE empno = v_empno;
118         COMMIT;
119     END
120     //
121     delimiter ;
122     CALL emp_del(7900);
123
124
125     --부서번호, 부서이름, 지역을 받아 삽입하는 프로시저
126     delimiter //
127     CREATE PROCEDURE sp_insert_dept(v_deptno TINYINT, v_dname VARCHAR(14), v_loc
        VARCHAR(13))
128     BEGIN
129         INSERT INTO DEPT
130         VALUES (v_deptno, UPPER(v_dname), UPPER(v_loc));
131         COMMIT;
132     END
133     //
134     delimiter ;
135
136     CALL sp_insert_dept(50, 'marketting', 'Seoul')
137
138
139     --emp table에서 새로운 사원의 정보를 이름, 업무, 매니저, 급여를 입력받아 등록하는
        emp_input 프로시저를 생성하라. 단, 부서번호는 매니저의 부서 번호와 동일하게 하고
        보너스는 SALESMAN은 0을 그 외는 NULL을 입력하라.
140
141     delimiter //
142     CREATE PROCEDURE sp_emp_input
143     (
144         v_empno  SMALLINT,
145         v_ename  VARCHAR(10),
146         v_job    VARCHAR(9),
147         v_mgr    SMALLINT,
148         v_sal    FLOAT
149     )
```

```
150     BEGIN
151         DECLARE v_deptno TINYINT;
152 
153         SELECT deptno
154         INTO   v_deptno
155         FROM emp
156         WHERE empno = v_mgr;
157 
158         IF UPPER(v_job) = 'SALESMAN' THEN
159             INSERT INTO emp
160             VALUES(v_empno, v_ename, UPPER(v_job), v_mgr, CURDATE(), v_sal, 0, v_deptno);
161         ELSE
162             INSERT INTO emp
163             VALUES(v_empno, v_ename, UPPER(v_job), v_mgr, CURDATE(), v_sal, NULL,
                v_deptno);
164         END IF;
165     END
166     //
167     delimiter ;
168 
169     CALL sp_emp_input(8000, 'Sujan', 'salesman', 7902, 2000);
170     CALL sp_emp_input(8001, 'Sally', 'clerk', 7566, 3000);
171 
172 
173     -- 우편번호 검색하기
174     delimiter //
175     CREATE PROCEDURE sp_zipcode
176     (
177         IN v_dong VARCHAR(100)
178     )
179     BEGIN
180         SELECT zipcode, sido, gugun, dong, bunji
181         FROM  zipcode
182         WHERE dong LIKE CONCAT('%', v_dong, '%');
183     END //
184     delimiter ;
185 
186     CALL sp_zipcode('역삼');
187 
188 
189     --이름을 입력받아서 그 사람의 업무가 MANAGER, ANALYST 이면 급여가 50% 가산하여
        갱신하고, 업무가 MANAGER, ANALYST 가 아니면 20% 가산하는 SQL문을 작성하시오.
190 
191 
192 9. OUT 매개변수
193     delimiter //
194     CREATE PROCEDURE test_proc_out
195     (
196         OUT v_name VARCHAR(30)   OUT은 항상 명시해야 한다.
197     )
198     BEGIN
```

```
199         DECLARE p_name VARCHAR(30);
200         SET p_name = 'My name is 한라산';
201         SELECT p_name INTO v_name;
202     END
203     //
204     delimiter ;
205
206     CALL test_proc_out(@t_name);     --binding 변수필요
207     SELECT @t_name;
208
209
210     --주어진 두개의 수 중 작은 수 구하기
211     delimiter //
212     CREATE PROCEDURE findMin
213     (
214         IN x INT,   IN y INT,  OUT z INT
215     )
216     BEGIN
217         DECLARE v_min INT;
218
219         IF x < y THEN
220             SET v_min = x;
221         ELSE
222             SET v_min = y;
223         END IF;
224
225         SELECT v_min INTO z;
226     END
227     //
228     delimiter ;
229
230     CALL findMin(23, 45, @t_min);
231     SELECT CONCAT('Minimum of (23,45) ==> ', @t_min);
232
233
234     --사번을 받아 사원이름과 봉급 검색
235     delimiter //
236     CREATE PROCEDURE sp_emp_select
237     (
238         IN v_empno SMALLINT,
239         OUT v_ename VARCHAR(10),
240         OUT v_sal FLOAT
241     )
242     BEGIN
243         SELECT ename, sal INTO v_ename, v_sal
244         FROM emp
245         WHERE empno = v_empno;
246     END
247     //
248     delimiter ;
249
```

OUT 변수는 프로시저 내에서만 동작하는 지역변수이기 때문에 외부에서 사용하기 위해서는 "Binding 변수에 값을 넣고" 그 변수를 사용해야 한다.

```
250     CALL sp_emp_select(7788, @t_ename, @t_sal);
251     SELECT @t_ename, @t_sal;
252
253
254     --이름을 입력받아서 그 사원의 정보 중 부서명과 급여를 검색하는 프로시저를 완성하시오.
255     delimiter //
256     CREATE PROCEDURE emp_dept_sal_select
257     (
258         IN v_ename VARCHAR(10),
259         OUT v_dname  VARCHAR(14),
260         OUT v_sal  FLOAT
261     )
262     BEGIN
263         DECLARE v_deptno TINYINT;
264
265         SELECT deptno, sal INTO v_deptno, v_sal
266         FROM   emp
267         WHERE ename = v_ename;
268
269         SELECT dname INTO v_dname
270         FROM   dept
271         WHERE  deptno = v_deptno;
272     END
273     //
274     delimiter ;
275
276     CALL emp_dept_sal_select('SMITH', @t_dname, @t_sal);
277     SELECT CONCAT('SMITH의 부서명 ==> ', @t_dname, ', 봉급 ==> ', @t_sal);
278
279
280 10. IN OUT 파라미터
281     delimiter //
282     CREATE PROCEDURE test_proc_inout(INOUT v_name VARCHAR(30))
283     BEGIN
284         DECLARE v_str VARCHAR(30);
285         SET v_str = CONCAT('My name is ', v_name);
286
287         SELECT v_str INTO v_name;
288     END
289     //
290     delimiter ;
291
292     SET @t_name = '북한산';
293     CALL test_proc_inout(@t_name);
294     SELECT @t_name;
295
296
297 11. Stored Procedure ALTER
298     -MySQL에서는 Stored Procedure 의 파라미터나 body를 수정할 수 있는 ALTER Procedure
        는 지원하지 않는다.
299     -수정이 필요하면 프로시저 삭제 후 새로 생성해야 한다.
```

값이 초기화된 Binding 변수를 프로시저에 입력하면
INOUT은 Binding 변수의 값을 바꿔 저장한다.

12. Stored **Procedure** Deletion
**DROP PROCEDURE [IF EXISTS]** sp_name;

13. Stored **Procedure**의 특징
1)MySQL의 성능을 향상시킨다.
2)모듈식 프로그래밍이 가능하다.
3)보안을 강화할 수 있다.
4)Programing **Language**에서 **Procedure**의 이름으로 호출할 수 있다.

REM Stored **FUNCTION** 프로시저와는 Return 절이 있냐 없느냐로 구분
1. MySQL에서 기본적으로 제공하는 함수 이외에 사용자가 필요에 따라 만든 함수
2. **Procedure**와 성격이 매우 비슷하지만, 반환값이 있느냐 없느냐가 가장 큰 차이
3. 실행시 반드시 하나의 값을 **RETURN**하기 위해 사용
4. 함수 선언에서 **Data Type**이 있는 **RETURN** 절을 추가하고 **body**에 **RETURN**문을 포함
5. 함수는 **IN** 파라미터만 사용한다.
6. Syntax

```
DELIMITER //
CREATE FUNCTION function_name
(
    param_name  type
)
RETURNS type
BEGIN
    SQL 문장들
    RETURN
END
//
DELIMITER;

--실행
SELECT function_name(argument_list);


delimiter //
CREATE FUNCTION chk_sal(v_sal FLOAT)
RETURNS FLOAT
BEGIN
    DECLARE t_sal FLOAT;
    SET t_sal = v_sal * 0.01;
    RETURN t_sal;
END
//
delimiter ;

SELECT empno, ename, sal, chk_sal(sal)
FROM emp
```

```
WHERE deptno = 10;

     EMPNO        SAL CHK_SAL(SAL)
---------- ---------- ------------
      7782       2450         24.5
      7839       5000           50
      7934       1300           13


SELECT SUM(sal), chk_sal(SUM(sal))
FROM emp
WHERE deptno = 10;

  SUM(SAL) CHK_SAL(SUM(SAL))
---------- -----------------
      8750              87.5


delimiter //
CREATE FUNCTION tax(v_value INT)
RETURNS INT
BEGIN
    RETURN v_value * 0.07;
END
//
delimiter ;

SELECT  sal, tax(sal)
FROM  emp
WHERE empno = 7902;

       SAL   TAX(SAL)
---------- ----------
       950       66.5


--사원명으로 검색하여 해당 사원의 직급을 얻어 오는 함수를 fun_sel_empname라는 이름으로 작성하시오.
delimiter //
CREATE FUNCTION fun_sel_empname(v_ename VARCHAR(10))
RETURNS VARCHAR(10)
BEGIN
    DECLARE v_job VARCHAR(9);
    SELECT job INTO v_job
    FROM emp
    WHERE ename = v_ename;

    RETURN v_job;
END
//
delimiter ;
```

```
401
402     SELECT fun_sel_empname('SCOTT');
403
404     --emp table에서 이름을 입력받아 부서번호, 부서명, 급여를 검색하는 함수(fun_emp_disp)을
        작성하시오. 단 부서번호를 RETURN에 사용하시오.
405         delimiter //
406         CREATE FUNCTION fun_emp_disp(v_ename VARCHAR(10))
407         RETURNS TINYINT
408         BEGIN
409             SELECT depno, dname, sal
410             FROM emp NATURAL JOIN dept
411             WHERE ename = v_ename;
412
413             RETURN deptno;
414         END
415         //
416         delimiter ;      ==> Error
417
418         delimiter //
419         CREATE FUNCTION fun_emp_disp(v_ename VARCHAR(10))
420         RETURNS VARCHAR(100)
421         BEGIN
422             DECLARE v_deptno TINYINT;
423             DECLARE v_dname  VARCHAR(14);
424             DECLARE v_sal FLOAT;
425
426             SELECT deptno, dname, sal INTO v_deptno, v_dname, v_sal
427             FROM emp NATURAL JOIN dept
428             WHERE ename = v_ename;
429
430             RETURN CONCAT('Department Number ==> ', v_deptno, ', Department Name ==> '
                , v_dname, ', Salary ==> ', v_sal);
431         END
432         //
433         delimiter ;
434
435         SELECT fun_emp_disp('SCOTT');
436
437
438 7. 수정
439     -Stored Procedure와 마찬가지로 수정은 할 수 없고 삭제 후 새로 생성해야 한다.
440
441 8. 삭제
442    DROP FUNCTION function_name;
443
444
445 9. Stored Procedure vs Stored Function
446
447     Stored Procedure                vs          Stored Function
448 ----------------------------------------------------------------------
449     파라미터 IN, OUT, INOUT 사용                  입력용 파라미터만 사용가능
```

| | | |
|---|---|---|
| **RETURNS** 사용불가 | | 반드시 **RETURN** 사용해**야** |
| **CALL**을 사용하여 호출 | | **SELECT** 문자에서만 사용 |
| 모든 **Statement** 사용 가능 | | **SELECT..INTO..**사용 가능 |
| 다양한 목적 사용 | | 계산을 통한 하나의 값 반환시 사용 |

**일반적으로 값의 연산은**
**백엔드(자바, 파이썬)에서 처리하지**
**DB에서는 잘 처리하지않음.**