

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

5
6

7 REM Stored Programs

8 1. 잘 정리된 논리적 코드분할이름 가 .

9 2. 컴파일된 상태에서 데이터베이스에 저장되기 때문에 성능이 향상된다.

10 3. 테이블이름이나 컬럼의 이름을 명시하지 않기 때문에 보안에 도움이 된다.

11 4. 모듈화를 통한 관리 용이

12 5. SQL 문으로 구성된 본문이 있다.

13 6. 본문은 세미콜론 문자로 구분된 여러 SQL문으로 구성된다.

14 7. 종류

15 1) Stored Procedures <-----Java 에서 사용

16 2) Stored Functions

17

18

19 REM Stored PROCEDURE(저장프로시저)

20 1. 목적 : 속도, 보안

21 2. compile 상태로 RDBMS 에 저장

22 3. 나중에 실행될 일련의 명령의 집합

23 4. Syntax

24 DELIMITER //

25 CREATE PROCEDURE procedure_name

26 (

27 [IN | OUT | IN OUT] param_name type

28)

29 BEGIN

30 SQL 문장들

31 END

32 //

33 DELIMITER;

34

35 5. Parameter Mode : 3가지

36 -IN : 입력 매개변수

37 -OUT : 출력 매개변수

38 -IN OUT : 입력, 출력 매개변수

39

40 6. Examples

41

42 delimiter //

43 CREATE PROCEDURE helloworld()

44 BEGIN

45 SELECT 'Hello, World';

46 END

47 //

48 delimiter ;

49

50 CALL helloworld();

```
CREATE PROCEDURE procedure_name(parameter...)
BEGIN
    SQL ...
END

Function

procedure
(OUT Function .)

Function .->
Function 가 .
```

```
IN (Return .)
OUT (Return .)
IN OUT 가 ?
```

```
BEGIN END SQL
가 .
```

```

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

```

```

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, 'marketing', '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,
164             v_deptno);
165     END IF;
166 END
167 //
168 delimiter ;
169
170 CALL sp_emp_input(8000, 'Sujan', 'salesman', 7902, 2000);
171 CALL sp_emp_input(8001, 'Sally', 'clerk', 7566, 3000);
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
190 -- 갱신하고, 업무가 MANAGER, ANALYST 가 아니면 20% 가산하는 SQL문을 작성하시오.
191
192 9. OUT 매개변수
193 delimiter //
194 CREATE PROCEDURE test_proc_out
195 (
196     OUT v_name VARCHAR(30)
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

```

```

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.

```

300
301
302 12. Stored Procedure Deletion
303     DROP PROCEDURE [IF EXISTS] sp_name;
304
305
306 13. Stored Procedure의 특징
307     1)MySQL의 성능을 향상시킨다.
308     2)모듈식 프로그래밍이 가능하다.
309     3)보안을 강화할 수 있다.
310     4)Programing Language에서 Procedure의 이름으로 호출할 수 있다.
311
312
313
314 REM Stored FUNCTION
315 1. MySQL에서 기본적으로 제공하는 함수 이외에 사용자가 필요에 따라 만든 함수
316 2. Procedure와 성격이 매우 비슷하지만, 반환값이 있느냐 없느냐가 가장 큰 차이
317 3. 실행시 반드시 하나의 값을 RETURN하기 위해 사용
318 4. 함수 선언에서 Data Type이 있는 RETURN 절을 추가하고 body에 RETURN문을 포함
319 5. 함수는 IN 파라미터만 사용한다.
320 6. Syntax
321     DELIMITER //
322     CREATE FUNCTION function_name
323     (
324         param_name type
325     )
326     RETURNS type
327     BEGIN
328         SQL 문장들
329         RETURN
330     END
331     //
332     DELIMITER;
333
334     --실행
335     SELECT function_name(argument_list);
336
337
338     delimiter //
339     CREATE FUNCTION chk_sal(v_sal FLOAT)
340     RETURNS FLOAT
341     BEGIN
342         DECLARE t_sal FLOAT;
343         SET t_sal = v_sal * 0.01;
344         RETURN t_sal;
345     END
346     //
347     delimiter ;
348
349     SELECT empno, ename, sal, chk_sal(sal)
350     FROM emp

```

```
351 WHERE deptno = 10;
```

```
352
```

```
353 EMPNO SAL CHK_SAL(SAL)
```

```
354 -----
```

```
355 7782 2450 24.5
```

```
356 7839 5000 50
```

```
357 7934 1300 13
```

```
358
```

```
359
```

```
360 SELECT SUM(sal), chk_sal(SUM(sal))
```

```
361 FROM emp
```

```
362 WHERE deptno = 10;
```

```
363
```

```
364 SUM(SAL) CHK_SAL(SUM(SAL))
```

```
365 -----
```

```
366 8750 87.5
```

```
367
```

```
368
```

```
369 delimiter //
```

```
370 CREATE FUNCTION tax(v_value INT)
```

```
371 RETURNS INT
```

```
372 BEGIN
```

```
373 RETURN v_value * 0.07;
```

```
374 END
```

```
375 //
```

```
376 delimiter ;
```

```
377
```

```
378 SELECT sal, tax(sal)
```

```
379 FROM emp
```

```
380 WHERE empno = 7902;
```

```
381
```

```
382 SAL TAX(SAL)
```

```
383 -----
```

```
384 950 66.5
```

```
385
```

```
386
```

```
387 --사원명으로 검색하여 해당 사원의 직급을 얻어 오는 함수를 fun_sel_empname라는  
이름으로 작성하시오.
```

```
388 delimiter //
```

```
389 CREATE FUNCTION fun_sel_empname(v_ename VARCHAR(10))
```

```
390 RETURNS VARCHAR(10)
```

```
391 BEGIN
```

```
392 DECLARE v_job VARCHAR(9);
```

```
393 SELECT job INTO v_job
```

```
394 FROM emp
```

```
395 WHERE ename = v_ename;
```

```
396
```

```
397 RETURN v_job;
```

```
398 END
```

```
399 //
```

```
400 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 ==> '
431     , v_dname, ', Salary ==> ', v_sal);
```

```
431 END
```

```
432 //
```

```
433 delimiter ;
```

```
434
435 SELECT fun_emp_disp('SCOTT');
```

436 7. 수정

437 -Stored Procedure와 마찬가지로 수정은 할 수 없고 삭제 후 새로 생성해야 한다.

438 8. 삭제

```
439 DROP FUNCTION function_name;
```

440 9. Stored Procedure vs Stored Function

Stored Procedure	vs	Stored Function

파라미터 IN, OUT, INOUT 사용		입력용 파라미터만 사용가능

450 RETURNS 사용불가
451 CALL을 사용하여 호출
452 모든 Statement 사용 가능
453 다양한 목적 사용
454

반드시 RETURN 사용해야
SELECT 문자에서만 사용
SELECT..INTO..사용 가능
계산을 통한 하나의 값 반환시 사용