/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

프로시저

- 특정 로직을 처리하는 재사용가능한 서브 프로그램

- 작성후 실행되면 컴파일 되어 객체로 오라클에 저장되며 EXECUTE나 EXEC 명령어로 필요시마다 재사용가능

구문

CREATE [OR REPLACE] PROCEDURE 이름 [(파라미터 선언,..)]

IS --declare대신, 생략불가

[변수 선언]

BEGIN

실행구문

[EXCEPTION

예외처리구문]

END;

- 파라미터 선언

- 개수 제한 없음

- 구문 : 변수명 mode 타입 [:=초기값]

- mode

- IN : 기본. 호출하는 곳으로 부터 값을 전달 받아 프로시저 내에서 사용. 읽기 전용변수.

- OUT : 호출 하는 곳으로 전달할 값을 저장할 변수.

- IN OUT : IN과 OUT의 기능을 모두 하는 변수

- 타입에 size는 지정하지 않는다.

- 초기값이 없는 매개변수는 호출시 반드시 값이 전달되야 한다.

실행

exec 이름[(전달값)]

execute 이름[(전달값)]

프로시저 제거

- DROP PROCEDURE 프로시저이름

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

set serveroutput on;

create or replace procedure message\_sp1

is

--변수선언

v\_message varchar2(100);

begin

--실행구문

v\_message := 'HelloWorld!';

dbms\_output.put\_line(v\_message);

end;

/

--ctrl+enter: 오라클 서버에 저장

--실행

execute message\_sp1;

exec message\_sp1;

create or replace procedure message\_sp2(p\_message in varchar2) --in은 생략가능

is

begin

dbms\_output.put\_line(p\_message);

end;

/

exec message\_sp2('안녕하세요');

exec message\_sp2('Hi');

exec message\_sp2;--error(매개변수가 없음)

create or replace procedure message\_sp2(p\_message in varchar2 := '기본 안녕하세요') --default지정

is

begin

dbms\_output.put\_line(p\_message);

end;

/

exec message\_sp2; --이 경우 default값인 기본 안녕하세요가 출력됨

create or replace procedure message\_sp3(p\_message in varchar2, p\_num pls\_integer)

is

begin

dbms\_output.put\_line(p\_message||p\_num);

end;

/

exec message\_sp3('안녕하세요',20);

create or replace procedure message\_sp4(p\_message in out varchar2) --out: 반환값을 저장할 변수를 받는다.

is

begin

dbms\_output.put\_line(p\_message);

p\_message := 'message\_sp4에서의 메시지';

end;

/

create or replace procedure caller\_sp

is

v\_msg varchar2(100);

begin

--message\_sp4 를 호출 - 메시지를 반환하는 sp

v\_msg := '기본값';

message\_sp4(v\_msg); --message\_sp4에서 v\_msg를 out변수로 받기 때문에 값을 받지 않는다.(in, out일시 받음)

dbms\_output.put\_line(v\_msg);

end;

/

exec caller\_sp;

--매개변수로 부서ID, 부서이름, 위치를 받아 DEPT에 부서를 추가하는 Stored Procedure 작성

create or replace procedure add\_dept(p\_dept\_id dept.dept\_id%type, p\_dept\_name dept.dept\_name%type

,p\_loc dept.loc%type)

is

begin

insert into dept values(p\_dept\_id, p\_dept\_name, p\_loc);

end;

/

exec add\_dept(&id, '&name', '&loc');

--매개변수로 부서ID를 파라미터로 받아 조회하는 Stored Procedure

create or replace procedure find\_dept\_sp(p\_dept\_id dept.dept\_id%type)

is

v\_dept\_id dept.dept\_id%type;

v\_dept\_name dept.dept\_name%type;

v\_loc dept.loc%type;

begin

select dept\_id, dept\_name, loc

into v\_dept\_id, v\_dept\_name, v\_loc

from dept

where dept\_id = p\_dept\_id;

dbms\_output.put\_line(v\_dept\_id||' - '||v\_dept\_name||' - '||v\_loc);

--조회 부서의 loc이 new york이면 seoul로 변경

if v\_loc = 'New York' then

update dept

set loc = 'Seoul'

where dept\_id = p\_dept\_id;

end if;

end;

/

exec find\_dept\_sp(&id);

-- TODO 직원의 ID를 파라미터로 받아서 직원의 이름(emp.emp\_name), 급여(emp.salary), 업무\_ID (emp.job\_id),

--입사일(emp.hire\_date)를 출력하는 Stored Procedure 를 구현

create or replace procedure TODO\_01\_sp(p\_emp\_id emp.emp\_id%type) --이름 대소문자 상관 X

is

v\_emp\_name emp.emp\_name%type;

v\_salary emp.salary%type;

v\_job\_id emp.job\_id%type;

v\_hire\_date emp.hire\_date%type;

begin

select emp\_name, salary, job\_id, hire\_date

into v\_emp\_name, v\_salary, v\_job\_id, v\_hire\_date

from emp

where emp\_id = p\_emp\_id;

dbms\_output.put\_line(v\_emp\_name||' - '||v\_salary||' - '||v\_job\_id||' - '||v\_hire\_date);

end;

/

EXEC TODO\_01\_sp(100);

EXEC TODO\_01\_sp(110);

-- TODO 업무\_ID(job.job\_id)를 파라미터로 받아서 업무명(job.job\_title)과

--업무최대/최소 급여(job.max\_salary, min\_salary)를 출력하는 Stored Procedure 를 구현

create or replace procedure TODO\_02\_sp(p\_job\_id job.job\_id%type)

is

v\_job\_title job.job\_title%type;

v\_max\_salary job.max\_salary%type;

v\_min\_salary job.min\_salary%type;

begin

select job\_title, max\_salary, min\_salary

into v\_job\_title,v\_max\_salary,v\_min\_salary

from job

where job\_id = p\_job\_id;

dbms\_output.put\_line(v\_job\_title||' - '||v\_max\_salary||' - '||v\_min\_salary);

end;

/

EXEC TODO\_02\_sp('FI\_ACCOUNT');

EXEC TODO\_02\_sp('IT\_PROG');

-- TODO: 직원\_ID를 파라미터로 받아서 직원의 이름(emp.emp\_name), 업무 ID(emp.job\_id), 업무명(job.job\_title) 출력하는 Stored Procedure 를 구현

create or replace procedure todo\_03\_sp(p\_emp\_id emp.emp\_id%type)

is

v\_emp\_name emp.emp\_name%type;

v\_job\_id emp.job\_id%type;

v\_job\_title job.job\_title%type;

begin

select e.emp\_name, e.job\_id, j.job\_title

into v\_emp\_name, v\_job\_id, v\_job\_title

from emp e left join job j on e.job\_id = j.job\_id

where e.emp\_id = p\_emp\_id;

dbms\_output.put\_line(v\_emp\_name||' - '||v\_job\_id||' - '||v\_job\_title);

end;

/

EXEC TODO\_03\_sp(110);

EXEC TODO\_03\_sp(200);

-- TODO 직원의 ID를 파라미터로 받아서 직원의 이름(emp.emp\_name), 급여(emp.salary),

--소속부서이름(dept.dept\_name), 부서위치(dept.loc)를 출력하는 Stored Procedure 를 구현

create or replace procedure todo\_04\_sp(p\_emp\_id emp.emp\_id%type)

is

v\_emp\_name emp.emp\_name%type;

v\_salary emp.salary%type;

v\_dept\_name dept.dept\_name%type;

v\_loc dept.loc%type;

begin

select e.emp\_name, e.salary, d.dept\_name, d.loc

into v\_emp\_name, v\_salary, v\_dept\_name, v\_loc

from emp e left join dept d on e.dept\_id = d.dept\_id

where e.emp\_id = p\_emp\_id;

dbms\_output.put\_line(v\_emp\_name||' - '||v\_salary||' - '||v\_dept\_name||' - '||v\_loc);

end;

/

EXEC TODO\_04\_sp(100);

EXEC TODO\_04\_sp(120);

-- TODO 직원의 ID를 파라미터로 받아서 그 직원의 이름(emp.emp\_name), 급여(emp.salary),

--업무의 최대급여(job.max\_salary), 업무의 최소급여(job.min\_salary), 업무의 급여등급(salary\_grade.grade)를 출력하는 store procedure를 구현

create or replace procedure todo\_05\_sp(p\_emp\_id emp.emp\_id%type)

is

v\_emp\_name emp.emp\_name%type;

v\_salary emp.salary%type;

v\_max\_salary job.max\_salary%type;

v\_min\_salary job.min\_salary%type;

v\_grade salary\_grade.grade%type;

begin

select e.emp\_name, e.salary, j.max\_salary, j.min\_salary, s.grade

into v\_emp\_name, v\_salary, v\_max\_salary, v\_min\_salary, v\_grade

from emp e left join job j on e.job\_id = j.job\_id

left join salary\_grade s on e.salary between s.low\_sal and s.high\_sal

where e.emp\_id = p\_emp\_id;

dbms\_output.put\_line(v\_emp\_name||' - '||v\_salary||' - '||v\_max\_salary||' - '||v\_min\_salary||' - '||v\_grade);

end;

/

EXEC TODO\_05\_sp(100);

EXEC TODO\_05\_sp(120);

-- TODO: 직원의 ID(emp.emp\_id)를 파라미터로 받아서 급여(salary)를 조회한 뒤 급여와 급여등급을 출력하는 Stored Procedure 를 구현.

-- 업무급여 등급 기준: 급여가 $5000 미만 이면 LOW, %5000 ~ $10000 이면 MIDDLE, $10000 초과면 HIGH를 출력

create or replace procedure todo\_06\_sp(p\_emp\_id emp.emp\_id%type)

is

v\_salary emp.salary%type;

begin

select salary

into v\_salary

from emp

where emp\_id = p\_emp\_id;

case when v\_salary < 5000 then

dbms\_output.put\_line('LOW');

when v\_salary <10000 then

dbms\_output.put\_line('MIDDLE');

else

dbms\_output.put\_line('HIGH');

end case;

/\*

if v\_salary < 5000 then

dbms\_output.put\_line('LOW');

elsif v\_salary <10000 then

dbms\_output.put\_line('MIDDLE');

else

dbms\_output.put\_line('HIGH');

end if;

\*/

end;

/

exec TODO\_06\_sp(100);

exec TODO\_06\_sp(130);