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

변수 사용

- 변수 선언

- DECLARE 절에서 한다.

- 구문: 변수명 [CONSTANT] 데이터타입 [NOT NULL] [ := 기본값]

- CONSTANT : 값을 변경할 수 없는 변수. 선언시 반드시 초기화 해야 한다.

- 데이터타입: 오라클 데이터타입+PL/SQL 데이터타입 지정한다.

- NOT NULL: 변수가 NULL을 가져선 안됨을 제약조건으로 지정. 선언시 반드시 초기화 되야함.

- 선언시 [DEFAULT 값] 으로 초기값 선언 가능

- 대입연산자

- 변수 := 값

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

declare

--변수명(identifier) datatype

v\_message1 varchar2(100); --null로 초기화

v\_message2 varchar2(100) not null := '기본메시지'; --not null은 반드시 초기화를 해주어야한다.

c\_num constant pls\_integer := 20; --정수형 상수

--c\_num constant pls\_integer default 20;

--not null변수와 constant(상수)는 선언시 초기화 해야 한다.

begin

v\_message1 := 'hello world';

v\_message2 := '&msg'; --치환연산자

--c\_num := 5000; --상수에는 값을 재할당 할 수 없다.

dbms\_output.put\_line(v\_message1);

dbms\_output.put\_line(v\_message2);

dbms\_output.put\_line(c\_num);

end;

/

-- TODO: DECLARE 에서 본인 정보를 저장할 변수들(이름, 나이, 주소 등등)을 선언하고 실행 블록(BEGIN)에서 값을 대입한 뒤 출력하는 코드 작성.

declare

v\_name nvarchar2(50);

v\_age pls\_integer; --number

v\_address nvarchar2(100);

begin

v\_name := '김선경';

v\_age := 24;

v\_address := '경기도 용인시 수지구';

dbms\_output.put\_line('이름: '||v\_name);

dbms\_output.put\_line('나이: '||v\_age);

dbms\_output.put\_line('주소: '||v\_address);

end;

/

-- TODO: emp 테이블의 이름(emp\_name), 급여(salary), 커미션비율(comm\_pct), 입사일(hire\_date) 를 값을 저장할 수 있는 변수를 선언하고

-- &변수를 이용해 값을 입력 받아 출력하는 코드를 작성.

declare

v\_emp\_name varchar2(50);

v\_salary number(7,2);

v\_comm\_pct number(2,2);

v\_hire\_date date;

begin

v\_emp\_name := '&name';

v\_salary := &salary;

v\_comm\_pct := &comm\_pct;

v\_hire\_date := '&hire\_date';

dbms\_output.put\_line('이름: '||v\_emp\_name);

dbms\_output.put\_line('급여: '||v\_salary);

dbms\_output.put\_line('comm\_pct: '||v\_comm\_pct);

dbms\_output.put\_line('입사일: '||v\_hire\_date);

end;

/

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

- %TYPE 속성

- 컬럼의 데이터타입을 이용해 변수의 타입 선언된

- 구문: 테이블명.컬럼%TYPE

ex) v\_emp\_id emp.emp\_id%TYPE

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

-- dept 테이블의 컬럼들의 데이터 타입을 이용해 변수 선언

declare

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

v\_dept\_name dept.dept\_name%type :='없음';

v\_loc dept.loc%type;

begin

v\_dept\_id := 2000;

v\_loc := '서울';

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

end;

/

-- TODO: job 테이블의 컬럼들의 타입을 이용해 v\_job\_id, v\_job\_title, v\_max\_salary, v\_min\_salary 를 생성하고 & 변수를 이용해 값을 입력 받아 출력하는 코드 작성

desc job;

declare

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

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

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

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

begin

v\_job\_id := '&job\_id';

v\_job\_title := '&job\_title';

v\_max\_salary := &max\_salary;

v\_min\_salary := &min\_salary;

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

end;

/

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

바인드 변수, 호스트 변수

※ 치환변수 &연산자와 다른 점은 db의 입장에서 &name은 입력한 값을 대체하여 받지만

바인드 변수는 :name 그대로 보낸다. 변수를 초기화하는 과정을 툴이 해준다.

보통 쿼리문에서 사용

-선언

variable 변수명 타입;

var 변수명 타입;

var name varchar2(100);

-사용

:변수명

:name

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

var e\_id number;

exec :e\_id :=200;

select \* from emp

where emp\_id = :e\_id

and emp\_name = :e\_name;

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

PL/SQL에서 시퀀스(Sequence) 사용

- select 구문없이 sequence이름.nextval 로 바로호출 가능.

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

create sequence t\_seq;

declare

v\_num number;

begin

v\_num := t\_seq.nextval;

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

dbms\_output.put\_line(t\_seq.currval);

end;

/

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

프로시저에서 함수 사용

- 문자/숫자/날짜/변환 함수는 처리결과가 1개일 때 실행 구문에서 단독으로 사용할 수 있다.

- 집계함수(max, min등)와 DECODE() 함수는 SQL문에서만 사용할 수 있다.

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

begin

dbms\_output.put\_line(length('aaaaa')); --별도의 select문 없이 바로 함수를 사용 할 수 있다.

dbms\_output.put\_line(round(4.232323,2));

end;

/

-- TODO: 다음을 순서대로 출력하는 코드를 작성

/\*

"서울시 강남구 역삼동" 문자열의 길이를 출력

"Hello World" 를 대문자로 출력

100.23456 를 소숫점 첫번째 이하에서 반올림 해서 출력

SYSDATE에서 시:분:초 를 출력\*/

begin

dbms\_output.put\_line(length('서울시 강남구 역삼동'));

dbms\_output.put\_line(upper('Hello World'));

dbms\_output.put\_line(round(100.23456,1));

dbms\_output.put\_line(to\_char(sysdate,'hh24:mi:ss'));

end;

/

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

중첩 구문

- PL/SQL 실행블록내에 PL/SQL 구문을 중첩해서 작성할 수 있다.

- 안쪽 실행블록에서는 바깥쪽 실행구문에서 선언한 변수를 사용할 수 있지만 반대는 안된다.

- 예

declare

begin

declare

begin

end

end

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

declare

v\_outer varchar2(100) := 'v\_outer';

begin

declare

v\_inner varchar2(100) := 'v\_inner';

begin

dbms\_output.put\_line('INNER: '||v\_outer); --outer의 변수를 inner에서 사용가능.

dbms\_output.put\_line('INNER: '||v\_inner);

end;

dbms\_output.put\_line('OUTER: '||v\_outer);

--dbms\_output.put\_line('OUTER: '||v\_inner); --error: inner에서 선언된 변수는 outer에서 사용못함.

end;

/

begin <<outer\_p>>

declare

v\_outer varchar2(100) := 'v\_outer';

begin

declare

v\_inner varchar2(100) := 'v\_inner';

v\_outer number := 30;

begin

dbms\_output.put\_line('INNER: '||v\_outer); --inner의 v\_outer가 호출됨

dbms\_output.put\_line('INNER: '||outer\_p.v\_outer); --outer의 v\_outer가 호출됨

dbms\_output.put\_line('INNER: '||v\_inner);

end;

dbms\_output.put\_line('OUTER: '||v\_outer);

end;

end;

/

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

DML 구문

- insert/delete/update

- SQL은 동일

- 처리후 commit

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

drop table dept\_copy;

create table dept\_copy

as

select \* from dept;

select \* from dept\_copy order by dept\_id desc;

--insert

declare

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

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

v\_loc dept.loc%type;

begin

v\_dept\_id := :id;

v\_dept\_name := :name;

v\_loc := :loc;

--insert into dept\_copy values(2000, '기획부', '인천');

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

v\_dept\_name := '연구소'; --값을 바꿈

v\_loc := '부산'; --값을 바꿈

insert into dept\_copy values(v\_dept\_id+1, v\_dept\_name, v\_loc);

insert into dept\_copy values(v\_dept\_id+2, v\_dept\_name, v\_loc);

commit;

end;

/

rollback;

--delete

begin

delete from dept\_copy

where loc = '서울';

commit;

end;

/

select \* from dept\_copy order by dept\_id desc;

--update

declare

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

v\_loc dept.loc%type := :loc;

begin

update dept\_copy

set loc = v\_loc

where dept\_id = v\_dept\_id;

commit;

end;

/

select \* from emp\_copy;

drop table emp\_copy;

create table emp\_copy

as

select emp\_id, emp\_name, salary, comm\_pct from emp;

desc emp\_copy;

-- TODO : emp 테이블에 새로운 세개행을 추가하는 구문을 작성.

declare

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

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

v\_emp\_salary emp.salary%type;

v\_comm\_pct emp.comm\_pct%type;

begin

v\_emp\_id := :id;

v\_emp\_name := :name;

v\_emp\_salary := :salary;

v\_comm\_pct := :commpct;

insert into emp\_copy values(v\_emp\_id, v\_emp\_name, v\_emp\_salary, v\_comm\_pct);

insert into emp\_copy values(v\_emp\_id+1, v\_emp\_name, v\_emp\_salary, v\_comm\_pct);

insert into emp\_copy values(v\_emp\_id+2, v\_emp\_name, v\_emp\_salary, v\_comm\_pct);

commit;

end;

/

-- TODO : 위에서 추가한 세개 행의 salary와 comm\_pct 의 값을 세배 올리는 구문을 작성

select \* from emp\_copy;

begin

update emp\_copy

set salary = salary \*3, comm\_pct = comm\_pct\*3

where emp\_id in (2000,2001,2002);

end;

/

rollback;

-- TODO : 위에서 추가한 세개 행을 삭제하는 구문을 작성

begin

delete from emp\_copy where emp\_id in (2000,2001,2002);

commit;

end;

/