1. PL/SQL

PL/SQL : Oracle's Procedural Language extension to SQL
오라클에 내장되어 있는 절차형 언어
변수 선언, 조건문, 반복문 등을 지원함

2. PL/SQL의 형식

선언부, 실행부, 예외 처리부로 구성됨

```
declare
--선언부(변수, 상수, CURSOR 등)
begin
--실행부(SQL 명령어, 반복문, 조건문 등)
exception
--예외처리부
end;
```

3. PL/SQL 명령어의 종류

3.1. Anonymous Block(익명 블록)

이름이 없는 블록

3.2. Procedure(프로시저)

DB에 저장되어 반복적으로 사용할 수 있는 블록
매개 변수 입출력 가능

3.3. Function(함수)

저장 프로시저와의 차이점 : 입력 매개변수만 사용 할 수 있고 리턴 타입을 반드시 지정해야 함

4. 저장 프로시저

Stored Procedure(SP, 저장 프로시저)

4.1. 형식

```
CREATE OR REPLACE PROCEDURE 프로시저이름(매개변수)
IS
  변수 선언
BEGIN
  문장
END;
```

4.2. 저장 프로시저 실습 예제

급여 인상 저장 프로시저 실습

```
create or replace procedure sal_p(p_empno number)
is
begin
  update emp
  set sal=sal*1.1
  where empno=p_empno;
end;
/

select * from emp;
execute sal_p(7499);

select * from user_source where name='SAL_P';
```

한줄메모장 저장 프로시저 실습

```
create table memo (
idx number primary key,
writer varchar2(50) not null,
memo varchar2(500) not null,
post_date date default sysdate
);

create sequence memo_seq
start with 1
increment by 1;

insert into memo (idx,writer,memo) values (memo_seq.nextval,
'kim', 'memo');
insert into memo (idx,writer,memo) values (memo_seq.nextval,
'park', 'memo2');

alter table memo add ip varchar2(50);
desc memo;
```

```
create or replace procedure memo_insert_p(p_writer varchar,p
_memo varchar,p_ip varchar)
is
begin
  insert into memo (idx,writer,memo,ip)
  values ( memo_seq.nextval, p_writer, p_memo, p_ip);
end;
/

execute memo_insert_p('김철수', '메모...', '192.168.0.10' );

select * from memo;

select * from user_source where name='MEMO_INSERT_P';
```

5. 함수(Function)

5.1. 형식

```
CREATE OR REPLACE FUNCTION 함수이름(입력매개변수)
RETURN 리턴자료형
IS
  변수 선언
BEGIN
  문장
  return
END;
```

5.2. 저장 프로시저와의 차이점 : return

5.3. 함수 예제

```
create or replace function sal_f(p_empno number)
return number
is
  v_sal number;
begin
  update emp set sal=sal*1.1 where empno=p_empno;

  select sal into v_sal from emp
  where empno=p_empno;
  return v_sal;
end;
/

select sal from emp where empno=7499;

--에러가 발생하므로 함수의 update 명령어를 주석처리한 후 실행합니다.
select sal_f(7499) from dual;
select empno,ename,sal,sal*1.1, sal_f(empno) from emp;

--함수의 update 명령어의 주석을 풀고 다시 실행합니다.
--아래 세줄의 명령어를 한꺼번에 실행합니다.(중간에 주석이 있으면 에러가 발생합니다.)
var salary number;
execute :salary := sal_f(7499);
print salary;
```

6. If 문

6.1. 형식

if 조건 then
elsif 조건 then
else
end if;

6.2. if문 예제

```
create or replace procedure dept_p(p_empno number)
is
  v_deptno number;
begin
  select deptno into v_deptno from emp
  where empno=p_empno;
  dbms_output.put_line('부서코드:'||v_deptno);
  if v_deptno = 10  then
    dbms_output.put_line('교육팀 직원입니다');
  elsif v_deptno = 20 then
    dbms_output.put_line('홍보팀 직원입니다');
  elsif v_deptno = 30 then
    dbms_output.put_line('기획팀 직원입니다');
  else
    dbms_output.put_line('기타부서 직원입니다');
  end if;
end;
/
select * from dept;
set serveroutput on
```

```
select * from emp;
execute dept_p(7782);
execute dept_p(7788);
execute dept_p(7844);
```

7. FOR LOOP 문

for 카운트변수 in [reverse] 시작값 .. 마지막값 loop
  --반복할 문장들
end loop;

```
set serveroutput on
delete from emp where empno <=100;
begin
  for cnt in 1 .. 100 loop
    insert into emp (empno,ename,hiredate) values
      (cnt, 'test'||cnt, sysdate);
  end loop;
  dbms_output.put_line('100개의 레코드가 입력되었습니다.');
end;
/
```

8. Loop 문

EXIT : LOOP 종료

EXIT WHEN : LOOP 종료 조건 설정

```
loop
  --반복할 문장들
  exit [when 조건문]
end loop;
```

```
set serveroutput on
delete from emp where empno <=100;
declare
  cnt number := 1;
begin
  loop
    insert into emp (empno,ename,hiredate) values
      (cnt, 'test'||cnt, sysdate);
    exit when cnt >= 100;
    cnt := cnt+1;
  end loop;
  dbms_output.put_line('100개의 레코드가 입력되었습니다.');
end;
/
```

9. WHILE LOOP

FOR 문과 비슷하며 조건이 TRUE일 경우만 반복되는 LOOP

예제

```
delete from emp where empno<=100;
declare
  cnt number := 1;
begin
  while cnt <=100 loop
      insert into emp(empno, ename , hiredate)
      values (cnt, 'test'||cnt, sysdate);
      cnt := cnt + 1 ;
  end loop ;

  dbms_output.put_line('100개의 레코드가 입력되었습니다.');
end;
/
```

10. 커서(Cursor)

select 명령어의 실행 결과를 하나의 행 단위로 탐색하는 객체

커서 열기

OPEN 커서이름;

커서 페치

커서가 현재 가리키는 레코드를 변수에 저장

FETCH 커서이름 INTO 변수;

커서 닫기

CLOSE 커서이름;

```
create or replace procedure cursor_p(p_deptno number)
is
  cursor cursor_avg is
    select dname,count(empno) cnt, round(avg(sal),1) sal
    from emp e, dept d
    where e.deptno=d.deptno
      and e.deptno=p_deptno
    group by dname;

  dname varchar(50);
  cnt number;
  sal_avg number;
begin
  open cursor_avg;

  fetch cursor_avg into dname, cnt, sal_avg;
```

```
  dbms_output.put_line('부서명:'|| dname);
  dbms_output.put_line('사원수:'|| cnt);
  dbms_output.put_line('평균급여:'|| sal_avg);

  close cursor_avg;
end;
/
execute cursor_p(10);




create or replace procedure cursor2_p
is
  cursor cursor_avg is
    select dname,count(empno) cnt, round(avg(sal),1) sal
    from emp e, dept d
    where e.deptno=d.deptno
    group by dname;
begin
  for row in cursor_avg loop
    dbms_output.put_line('부서명:'|| row.dname);
    dbms_output.put_line('사원수:'|| row.cnt);
    dbms_output.put_line('평균급여:'|| row.sal);
  end loop;
end;
/
execute cursor2_p;
```

11. 트리거

create or replace trigger 트리거이름
before/after
insert/update/delete on 테이블이름
--실행할 명령어들

Trigger(방아쇠) : 연쇄적인 동작을 정의하는 객체
INSERT, UPDATE, DELETE 문이 실행될 때 자동으로 실행되는 기능

Before Trigger : INSERT, UPDATE, DELETE 문이 실행되기 전에 실행
After Trigger : INSERT, UPDATE, DELETE 문이 실행된 후 실행

```
create or replace trigger sum_t
after
  insert or update or delete on emp
declare
  avg_sal number;
begin
  select avg(sal) into avg_sal from emp;
  dbms_output.put_line('급여평균:'|| avg_sal);
end;
/
set serveroutput on;
select avg(sal) from emp;
insert into emp (empno,ename,hiredate,sal) values (1000,'박철수',sysdate,500);
```

```
update emp set sal=600 where empno=1000;
delete from emp where empno=1000;
```