34. PL/SQL : Oracle's Procedual Language extension to SQL

#SQL의 장점과 단점

<장점>

(1) 사용자가 이해하기 쉬운 단어로 구성
(2) 쉽게 배울 수 있다.
(3) 복잡한 로직을 간단하게 작성할 수 있다.
(4) ANSI에 의해 문법이 표준화되어 있다.

<단점>

(1) 반복처리를 할 수 없다.(Loop)
(2) 비교처리를 할 수 없다.(IF)
(3) Error 처리를 할 수 없다.(예외처리)
(4) SQL문을 캡슐화 할 수 없다.
(5) 변수선언을 할 수 없다.
(6) 실행할 때 마다 분석작업 후 실행
(7) Network Traffic을 유발한다.

이러한 단점을 극복하기 위해 PL/SQL을 사용
PL/SQL은 SQL로 얻을 수 없는
절차적 언어의 기능을 가지고 있다.

PL/SQL 사용 이유

1. 반복 처리를 할 수 있다.(Loop)
2. 비교 처리를 할 수 있다.(IF)
3. Error 처리를 할 수 있다.(예외처리)
4. SQL문을 캡슐화 할 수 있다. (데이터의 보안 및 무결성)
5. 변수 선언을 할 수 있다.
6. 실행할 때 마다 분석된 결과를 실행 만 하기 때문에 성능이 빠르다.
7. Network Traffic이 감소된다. 여러 sql문장을 block으로 묶고
   한번에 블럭 전부를 서버로 전송하기 때문에 통신량을 줄일 수 있다.

#PL/SQL의 처리

PL/SQL으로 작성된 블럭을 오러클 서버로 보내면
그 안에 있는 PL/SQL 엔진이 sql문과 non-sql문을 구분하여
non-sql문은 PL/SQL 엔진 내의 Proceduer statement executor가 수행하고,
sql문은 SQL statement executor가 처리하게 된다.

#PL/SQL Block의 구조

DECLARE
　　실행부에서 참조할 모든 변수,상수,
BEGIN
　　SQL, PL/SQL 문장
EXCEPTION
　　에러 발생시 수행해야 할 문장들 기술
END;

지켜야 할 사항
1) PL/SQL 블럭내에서는 한 문장이 종료할 때 마다 세미콜론(;)을 기술한다.
2) end뒤에 세미콜론(;)을 사용하여 하나의 블럭이 끝났다는 것을 명시한다.

#PL/SQL의 블럭 유형
1) Anonymous Block (익명 블럭)

```
[DECLARE]
BEGIN
        실행문장;
        실행문장;
        ....
[EXCEPTION]
END;
```

- 실행하기 위해 프로그램 안에서 선언되고 실행을 위해 PL/SQL엔진으로 전달된다. 선행 컴파일러 프로그램과 Sql plus 또는 서버 관리자에서 익명의 블럭을 내장 할 수 있다.

2) Subprogram (Procedure, function)

Anonymous Block	Procedure	Function

DECLARE
선언문;
.....
BEGIN
실행문장;
실행문장;
....
[EXCEPTION]
END;
	CREATE PROCEDUER name IS
	BEGIN
	실행문장;
	실행문장;
	....
	[EXCEPTION]
	END;
		CREATE FUNCTION name
		RETURN datatype IS
		BEGIN
		실행문장;
		실행문장;
		....
		RETURN value;
		[EXCEPTION]
		END;

- Subprogram은 매개변수를 사용할 수 있고 호출할 수 있는 PL/SQL블럭이다.
  Procedure 또는 Function으로 선언될 수 있는데,
  어떤 작업을 수행하기 위해 Procedure를 사용하고
  값을 계산하기 위해 Function을 사용한다.

실습1] Anonymous Block으로 작성

```
SET SERVEROUTPUT ON
BEGIN
        DBMS_OUTPUT.PUT_LINE('Hello World!');
END;
```

실습2] 변수를 선언하여 데이터를 대입후 출력

```
SET SERVEROUTPUT ON
DECLARE
        I_MSG VARCHAR2(100);
BEGIN
        I_MSG := 'HELLO WORLD';
        DBMS_OUTPUT.PUT_LINE(I_MSG);
END;
```

실습3]

```
SET VERIFY OFF

VARIABLE HELLO VARCHAR2(30)  -- 변수 선언
ACCEPT P_NUM PROMPT '숫자를 입력하세요=> ' -- 데이타입력을 받아 P_NUM에 대입

DECLARE
  V_NUM NUMBER := &P_NUM;
BEGIN
  IF MOD(V_NUM,2) = 0 THEN
     :HELLO := 'EVEN';
  ELSE
     :HELLO :='ODD';
  END IF;
  DBMS_OUTPUT.PUT_LINE(:HELLO);
END;
```

PRINT HELLO로 확인 가능

변수

일단 선언되면 다른 선언적 문장을 포함한 다른 문장에서 간단하게 그것을
반복적으로 사용할 수 있다. %type %ROWTYPE 을 사용하여 변수를 선언하면
테이블의 구조가 변경되더라도 애플리케이션에서는 실행시간에 테이블을 참조하여
변수가 정의되므로 데이터의 독립성, 유지비용 절감을 제공한다.

#PL/SQL에서의 변수 처리

1) 선언 섹션 내에서 변수를 선언하고 초기화한다.
2) 실행 섹션에서 변수에 대한 새 값을 할당한다.
3) 매개변수를 통해 PL/SQL블럭으로 값을 전달한다.
4) 출력변수를 통해 결과를 본다.

#변수 선언 구문

식별자 [CONSTANT] 자료형 [NOT NULL] [:= | DEFAULT 표현식]

[예1]

```
DECLARE
        v_hieredate     date;
        v_deptno        number(2) not null  := 10;
        v_loc           varchar2(13) := 'atlanta';
        v_com           constant number := 1400;--상수숫자선언(변할 수 없다)
        v_flag    BOOLEAN   NOT NULL := TRUE;--NOT NULL값 TRUE로 초기화
```

규칙

1) not null로 지정된 변수의 경우 초기화를 한다.
2) 지정연산자(:=)를 사용하거나 default를 사용하여 초기화할 수 있다.
3) 한 라인에 하나의 식별자만 선언한다.
4) 상수선언에서 constant는 자료형 지정자보다 먼저 기술돼야 한다.
5) 하나의 블럭에서 동일명의 변수를 선언할 수 없다.
6) 블럭이 다르면 동일 이름을 선언할 수 있다.
7) 변수명을 테이블 컬럼명과 동일하게 선택하면 안된다.

[예2]

```
DECLARE
        v_hiredate      date;
        v_ename         varchar2(10);
BEGIN
        v_hiredate := '16-04-29';   -- to_date함수 사용가능
        v_ename := 'THOMAS';
```

%TYPE 데이터형

- %TYPE 데이터형은 기술한 데이터베이스 테이블의 컬럼 데이터 타입을 모
를 경우 사용할 수 있고,
- 또. 코딩이후 데이터베이스 컬럼의 데이터 타입이 변경될 경우 다시 수정할 필요가 없다.
- 이미 선언된 다른 변수나 데이터베이스 컬럼의 데이터 타입을 이용하여 선언합니다.
- 데이터 베이스 테이블과 컬럼 그리고 이미 선언한 변수명이 %TYPE앞에 올수 있습니다.

[예제]

```
        v_empno   emp.empno%TYPE := 7900 ;
        v_ename   emp.ename%TYPE;
```

[실습]

```
create or replace procedure Emp_Info (p_empno in emp.empno%type)
is
        v_empno emp.empno%type;
        v_ename emp.ename%type;
        v_sal emp.sal%type;
begin
        dbms_output.enable;
        select empno, ename, sal into v_empno, v_ename, v_sal from emp
        where empno=p_empno;
        dbms_output.put_line('사원번호:'||v_empno);
        dbms_output.put_line('사원이름:'||v_ename);
        dbms_output.put_line('사원급여:'||v_sal);
end;
```

[실행]

```
SET SERVEROUTPUT ON;
EXECUTE Emp_Info(7369);
```

[결과]

```
사원번호 : 7369
사원이름 : SMITH
사원급여 : 880
```

%ROWTYPE

하나 이상의 데이터값을 갖는 데이터 타입으로 배열과 비슷한 역할을 하고 재사용이 가능합니다.

%ROWTYPE데이터 형과, PL/SQL테이블과 레코드가 복합 데이터 타입에 속합니다.

- 테이블이나 뷰 내부의 컬럼 데이터형,크기,속성등을 그대로 사용할수 있다.
- %ROWTYPE 앞에 오는 것은 데이터 베이스 테이블 이름이다.
- 지정된 테이블의 구조와 동일한 구조를 갖는 변수를 선언할수 있다.
- 데이터베이스 컬럼들의 수나 DATATYPE을 알지 못할 때 편리.
- 테이블의 데이터 컬럼의 DATATYPE이 변경될 경우 프로그램을 재수정할 필요가 없다

[실습]

```
CREATE OR REPLACE PROCEDURE RTEST (p_empno in emp.empno%TYPE)
is
        v_emp emp%rowtype;
begin
        dbms_output.enable;
        select empno, ename, hiredate into v_emp.empno, v_emp.ename,
                                v_emp.hiredate FROM emp
        WHERE empno=p_empno;
        DBMS_OUTPUT.PUT_LINE('사번: '||v_emp.empno);
        DBMS_OUTPUT.PUT_LINE('이름: '||v_emp.ename);
        DBMS_OUTPUT.PUT_LINE('입사일: '||v_emp.hiredate);
END;
```

[실행]

```
SET SERVEROUTPUT ON;
EXECUTE RTEST(7900);
```

[결과]

```
사번:7900
이름:JAMES
입사일:81/12/03
```

[소스 확인] 프로시저 소스 확인

```
SELECT text FROM user_source;
```

[1] PL/SQL RECORD TYPE

[실습]

```
SET VERIFY OFF
SET SERVEROUTPUT ON
ACCEPT P_ENAME PROMPT '조회할 사원의 이름을 입력하세요=>'

-----------
DECLARE
        -- TYPE 정의
        TYPE emp_record_type IS RECORD(
                V_EMPNO emp.empno%TYPE,
                 V_ENAME emp.ename%TYPE,
                 V_JOB emp.job%type,
                 V_MGR emp.mgr%type,
                 V_HIREDATE emp.hiredate%type,
                 V_SAL emp.sal%type,
                 v_comm emp.comm%type,
                 v_deptno emp.deptno%type
                 );
        emp_record emp_record_type;
        v_ename emp.ename%type := '&p_ename';
BEGIN
        SELECT *
           INTO emp_record
           FROM emp
           WHERE ename=UPPER(v_ename);

dbms_output.put_line('사번 : '||TO_CHAR(emp_record.v_empno));
dbms_output.put_line('이름 : '||emp_record.v_ename);
dbms_output.put_line('업무 : '||emp_record.v_job);
dbms_output.put_line('급여 : '||LTRIM(TO_CHAR(emp_record.v_sal,'$999,999.00')));

EXCEPTION
        WHEN NO_DATA_FOUND THEN
                dbms_output.put_line('&p_ename'||'의 데이터는 없어요');
         WHEN TOO_MANY_ROWS THEN
                dbms_output.put_line('&p_ename'||'의 데이터가 2건 이상이에요');
```

```
        WHEN OTHERS THEN
            dbms_output.put_line('기타 에러입니다');
END;
```

NON-PL/SQL변수
- 바인드 변수: 호스트 환경에서 선언된 변수
  하나 이상의 PL/SQL프로그램 내부나 외부에
  전달하기 위해 사용한다.

- 바인드 변수 선언 구문
  VAR[IABLE] [variable [NUMBER| CHAR(n)| VARCHAR2(n)]]

---[실습]--------------------
```
variable rvar number

declare
begin
:rvar :=100;
--바인드 변수를 참조하기 위해서는 바인드 변수에
--콜론(:)을 참조 접두어로 기술한다.
end;
/
```
[실행]
```
print rvar
```

프로시저 파라미터 종류
1) IN PARAMETER
2) OUT PARAMETER
3) IN OUT PARAMETER

1) IN PARAMETER : 호출자에 의해 프로시저에 전달되는
   파라미터. 파라미터의 디폴트 값으로
   파라미터 앞에 아무것도 표시하지
   않으면 IN 파라미터다.
   프로시저에서 IN 파라미터의 값을
   변경할 수 없다.

```
------------------------------
create or replace procedure myadd
(p_param in number default 10)
is
begin
insert into test(A) values(p_param);
end;
/
--------------------------------
실행
exec myadd(50);
exec myadd(p_param=> 60);
```

2) OUT PARAMETER : 프로시저가 사용자에게 넘겨주는 값
프로시저에서 값을 변경할 수 있다.
디폴트 값을 지정할 수 없다.

```
---------------------------------
create or replace procedure empFind
(pempno in emp.empno%type,
 oename out emp.ename%type)
 is
 begin
 select ename into oename
 from emp where empno=pempno;
 end;
 /
---------------------------------
** out 파라미터가 있는 프로시저를 실행하려면
    변수를 선언한후 실행해야 함.
   variable fname varchar2(20);
   exec empFind(7788,:fname);
   print fname;
```

3) IN OUT PARAMETER :프로시저가 읽고 쓰는 작업을
동시에 할 수 있는 파라미터

//Insert문 실습//

```
create or replace procedure empInsert
(
v_eno emp.empno%type,
v_ename emp.ename%type,
v_deptno emp.deptno%type)
is
begin
DBMS_OUTPUT.ENABLE;
insert into emp(empno,ename,hiredate,deptno)
values(v_eno,v_ename,sysdate,v_deptno);

DBMS_OUTPUT.PUT_LINE('사번: '||v_eno);
DBMS_OUTPUT.PUT_LINE('이름: '||v_ename);
DBMS_OUTPUT.PUT_LINE('부서: '||v_deptno);
DBMS_OUTPUT.PUT_LINE('위의 데이터 입력 성공');
end;
/
-----------------------------------
SET SERVEROUTPUT ON;
EXECUTE(1000,'홍길동',20);
*자동 커밋이 안된다. 하고나서 커밋해주던지 아님 아래처럼...
////////////////////////////////////////////////


///UPDATE문실습//////////////////////////////
create or replace procedure empUpdate
(v_eno in emp.empno%type,    --수정할 사원 사번
v_rate in number)                    --급여 인상률
is
 -- 수정 데이터를 확인하기 위한 변수 선언
v_emp emp%rowtype;
begin
dbms_output.enable;
update emp
set sal=sal+(sal*(v_rate/100))
where empno=v_eno;
dbms_output.put_line('데이터 수정 성공');
```

```
-- 수정된 데이터 확인하기 위해 검색
select empno,ename,sal
into v_emp.empno,v_emp.ename,v_emp.sal
from emp
where empno=v_eno;

dbms_output.put_line('==결과======');
dbms_output.put_line('사번: '||v_emp.empno);
dbms_output.put_line('이름: '||v_emp.ename);
dbms_output.put_line('급여: '||v_emp.sal);
end;
/
----------------------------
SET SERVEROUTPUT ON
EXECUTE EMPUPDATE(7900,20);
EXECUTE EMPUPDATE(7900,-10);
//////////////////////////////////////////


create or replace procedure myinsert
(v_name in salary.name%type,
v_pay in salary.pay%type)
is
begin
insert into salary values(v_name, v_pay);
commit;
end myinsert;
/
create or replace procedure youinsert
(vname in test.name%type)
is
begin
insert into test values(scott.test_id.nextval,
vname, sysdate);
commit;
end youinsert;
```

```
/
///DELETE문/////////////////////////////////////////
CREATE OR REPLACE PROCEDURE Del_Emp
    ( p_empno IN  emp.empno%TYPE )
        IS
        BEGIN
        -- 삭제 쿼리
        DELETE
        FROM emp
        WHERE empno = p_empno ;
       DBMS_OUTPUT.ENABLE;
       DBMS_OUTPUT.PUT_LINE(p_empno+'번 사원정보 삭제 성공');
       END;
/
execute Del_Emp(7788);
/////////////////////////////////////////////////////

# PL/SQL의 제어문
[1] IF문

       IF 조건 THEN
              실행문
       ELSIF 조건 THEN
              실행문

       ELSE
              실행문
       END IF;

       **주의] ELSIF문이 ELSEIF가 아니라 ELSIF이란 점에 주의하자.**

-------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE DEPT_SEARCH
(p_empno in emp.empno%type)
IS
v_deptno emp.deptno%type;
v_ename emp.ename%type;
```

```
v_msg varchar2(20);
BEGIN
    select deptno,ename into v_deptno,v_ename
    from emp
    where empno=p_empno;
IF v_deptno=10 THEN
        v_msg :='회계부서';
ELSIF v_deptno =20 THEN
        v_msg :='연구부서';
ELSIF v_deptno =30 THEN
        v_msg :='영업부서';
ELSIF v_deptno =40 THEN
        v_msg :='운영부서';
ELSE
        v_msg :='부서가 없어요';
END IF;
        DBMS_OUTPUT.PUT_LINE(p_empno||'번: 'v_ename||'님은 '||v_msg );
END;
/

set serveroutput on
execute dept_search(7499);
```

---

[2] FOR LOOP문

```
        FOR index IN [reverse] 시작값..end값 LOOP
        문장1
        문장2
        ....
        END LOOP;
```

-index는 자동선언되는 binary_integer 형 변수이고. 1씩 증가한다

- reverse 옵션이 사용될 경우 index 는
  upper_bound에서
  lower_bound로 1씩 감소한다.
- IN 다음에는 coursor나 select 문이 올수 있다.

```
--실습-----------------------------------
SET SERVEROUTPUT ON

declare
  --배열의 데이터 타입과 배열명
  type ename_table is table of emp.ename%type index by binary_integer;
  type sal_table is table of emp.sal%type index by binary_integer;
  -- 배열형 변수 선언
  ename_tab ename_table;
  sal_tab sal_table;
 --정수형변수
  i INTEGER:=0;
BEGIN
    FOR e IN (SELECT ENAME,SAL FROM EMP WHERE DEPTNO=10)LOOP
        i :=i+1;
        ename_tab(i) :=e.ename;
        sal_tab(i) := e.sal;
    END LOOP;

    FOR k IN 1..i LOOP
        DBMS_OUTPUT.PUT_LINE('사원명: '||ename_tab(k));
        DBMS_OUTPUT.PUT_LINE('급 여 : '||sal_tab(k));
    END LOOP;
END;
-------------------------------------------

[3]루프문

LOOP
        실행문장;
EXIT [WHEN 조건문]
END LOOP;

LOOP 문 예제---------------------------------

SET SERVEROUTPUT ON ;
```

```
-- (DBMS_OUTPUT.PUT_LINE을 출력하기 위해 사용)

DECLARE
      v_cnt number(3) := 100;
BEGIN
    DBMS_OUTPUT.ENABLE ;
    LOOP
       INSERT INTO emp(empno, ename , hiredate)
       VALUES(v_cnt, 'test'||to_char(v_cnt),   sysdate);

       v_cnt := v_cnt+1;
    EXIT WHEN v_cnt > 105; --조건에 참이면 loop 종료
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('데이터 입력 완료');
    DBMS_OUTPUT.PUT_LINE(v_cnt-100 || '개의 데이터가 입력되었습니다');

 END;
         /


데이터 입력 완료
6개의 데이터가 입력되었습니다

PL/SQL 처리가 정상적으로 완료되었습니다.

-----------------------------------------------------------------
[4] WHILE LOOP 문

WHILE LOOP문은 FOR 문과 비슷하며 조건이
TRUE일 경우만 반복되는 LOOP문 입니다.

예제 ---------------------------------

DECLARE
  cnt number(3):=1;
BEGIN
```

```
  WHILE cnt <10 LOOP
    INSERT INTO EMP(EMPNO,ENAME,HIREDATE)
    VALUES(CNT,'TEST1'||TO_CHAR(CNT),SYSDATE);
  EXIT WHEN CNT=5;--cnt가 5이면 종료
    CNT:=CNT+1;
  END LOOP;
END;
/
----------------------------------------
```

cnt가 10이면 반복 While Loop를 탈출-CNT가 5가되면 루프문 이탈

EXIT WHEN조건 => 조건이 만족할 때 반복 loop를 탈출합니다. .
