02. PL/SQL 기본문법

목차

1. 변수

▣ 변수

- 3,4세대 프로그래밍 언어들 처럼 PL/SQL block내에서 사용하는 변수/상수/사용자 정의 예외/커서 선언
- 선언부(Declare section)에서 변수를 정의 하고 초기화
- 다양한 데이터 타입을 변수로 정의 ex) number, varchar2, date, clob, blob

```
SET  SERVEROUTPUT ON
DECLARE
        V_XX          NUMBER(4) := 10;                  -- 변수선언(0) AND 초기값 할당(O)
        V_YY          NUMBER(4);                        -- 변수선언(0) AND 초기값 할당(X)
        V_TAX_RATE    CONSTANT NUMBER(5,3) := 0.054;    -- 제약(Constraint)사항 : 상수
        V_SAL         NUMBER(4) NOT NULL    := 9999;    -- 제약(Constraint)사항 : NOT NULL
BEGIN
        DBMS_OUTPUT.PUT_LINE('V_XX =>'||V_XX);
        DBMS_OUTPUT.PUT_LINE('V_YY =>'||V_YY);
        DBMS_OUTPUT.PUT_LINE('V_YY =>'||NVL(V_YY,-99));                    -- 함수 사용 가능
        DBMS_OUTPUT.PUT_LINE('V_ZZ + V_YY =>'||TO_CHAR(V_XX+V_YY));        -- NULL 연산
        DBMS_OUTPUT.PUT_LINE('V_TAX_RATE =>'||V_TAX_RATE);

        --V_TAX_RATE := 0.055;                                             -- 상수 변경시
        --V_SAL := V_XX + V_YY;                                            -- NOT NULL
END;
/
```

2. 연산자

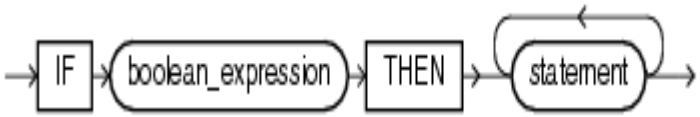
연산자 구분	내용
산술연산자	+ , / , * ,-,**
비교연산자	= , != , <> ,~=, <,>, <=,>=
논리연산자	AND,OR,NOT
SQL 연산자	LIKE, BETWEEN,IN, IS NULL

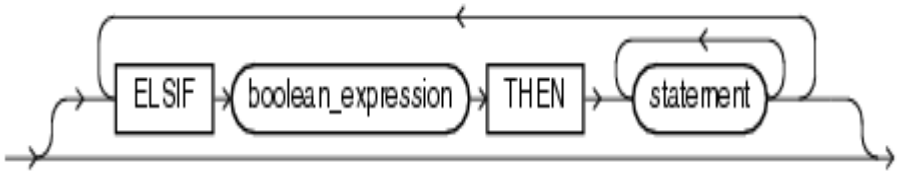
대입연산자	:=
비교연산자	=

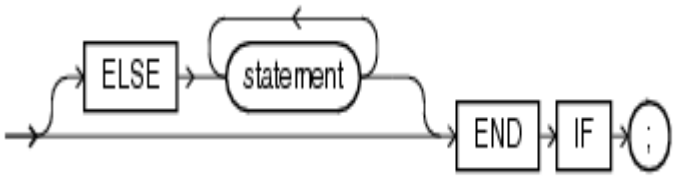
```
DECLARE
        V_XX                    NUMBER(4) := 10;
        V_YY                    BOOLEAN;
        V_TAX_RATE    CONSTANT NUMBER(5,3) := 0.054;
        V_SAL                   NUMBER(4) := 9999;
BEGIN
  IF V_YY IS NULL THEN                                          // SQL 연산자
        DBMS_OUTPUT.PUT_LINE('SQL operator');
  END IF;

  V_YY := ((V_XX - 1) ** 3 > 500) AND (V_SAL > 7000) ;          // 산술 연산자 & 비교 연산자 & 논리 연산자
  DBMS_OUTPUT.PUT_LINE( case when V_YY then 'TRUE' else 'FALSE' end);   // case & decode

  IF (V_SAL * V_TAX_RATE) BETWEEN 500 AND 1000 THEN             // SQL 연산자
        DBMS_OUTPUT.PUT_LINE('High pay');
  END IF;
END;
/
```

3. 조건식

if statement ::=







3. 조건식

```
DECLARE
        V_EMPNO         NUMBER(4)     := 8888;          -- 변수선언 및 초기화
        V_DEPTNO    NUMBER(2);
        V_ENAME         VARCHAR2(10) := 'XMAN';         -- 변수선언 및 초기화
        V_JOB           VARCHAR2(9);
        V_SAL           NUMBER(7,2);
BEGIN
        V_DEPTNO := 20;                                 -- 변수에 값 대입
        IF V_JOB IS NULL THEN
                V_JOB := '신입';
        END IF;

        IF V_JOB = '신입'    THEN                       -- TRUE, FALSE, NULL
                V_SAL := 2000;
        ELSIF V_JOB IN ('MANAGER','ANALYST') THEN       -- ELSEIF(X)
                V_SAL := 3500;
        ELSE
                V_SAL := 2500;
        END IF;

        INSERT INTO EMP(DEPTNO,EMPNO,ENAME,SAL,JOB)
                VALUES(V_DEPTNO,V_EMPNO,V_ENAME,V_SAL,V_JOB);
        COMMIT;
END;
/
SELECT * FROM EMP WHERE EMPNO = 8888;
```

4. 반복문

PL/SQL 반복문의 3가지 유형

▣ 기본 Loop

```
Ex)  LOOP
       ~
     END LOOP;
```

▣ FOR Loop

```
Ex)  FOR  ~
     LOOP
     ~
     END LOOP;
```

▣ WHILE Loop

```
Ex)  WHILE  ~
     LOOP
     ~
     END LOOP;
```

4. 반복문

▣ 기본 Loop

```
SET SERVEROUTPUT ON

DECLARE
        LOOP_INDEX          NUMBER(4)  :=  1;
        MAX_LOOP_INDEX      NUMBER(4)  :=  30;
BEGIN
        LOOP
                DBMS_OUTPUT.PUT_LINE('LOOP COUNT => '||TO_CHAR(LOOP_INDEX));

                LOOP_INDEX  :=  LOOP_INDEX + 1;

                EXIT  WHEN  MAX_LOOP_INDEX  < LOOP_INDEX ;

        END  LOOP;
END;
/
```

① LOOP로 시작하고 END LOOP로 종료 , END LOOP 다음에 세미콜론(;)을 문장 종결자로 사용

PL/SQL 문법은 다양한 변형이 가능한 C 언어 문법과 달리 문법 자체가 규칙적으로 정형화
예) IF ~ END IF;
LOOP ~ END LOOP;

4. 반복문

▣ 기본 Loop

② LOOP 구문 사용시 명시적으로 LOOP를 종료하는 EXIT절 사용

< 참고 >

LOOP는 2가지 유형의 LOOP 종료 방식을 사용 할수 있습니다.
① EXIT를 사용하는 방법 ② IF 문을 사용하여 조건 CHECK후 EXIT를 하는방법

```
LOOP
   statement1;
   statement2;
   EXIT  [ WHEN  conditon ];
END  LOOP;
```

```
LOOP
   statement1;
   statement2;
   IF  condition  THEN
      EXIT;
   END IF;
END  LOOP;
```

4. 반복문

▣ FOR Loop

FOR LOOP는 기본 LOOP 반복문에 LOOP COUNTER 기능이 추가된 형태

LOOP COUNTER는 것은 FOR 문장에서 **LOOP INDEX**(LOOP 첨자)를 제어

```
FOR loop_index in [REVERSE] lower_bound .. upper_bound
LOOP
    statement1;
    statement2;
END LOOP;
```

* **loop_index**의 특징
  - block내에서 **정수 데이터 타입(INTEGER TYPE)으로 암시적, 자동으로(implicit)으로 선언**
  - **증가,감소 폭은 1 , 증가폭 변경 불가 (참조만 가능)**

    ex) V_tmp := loop_index; (O)

    loop_index := loop_index + 1; (X)

* **lower_bound .. upper_bound : loop_index 가 실행되는 시작(하한값) 및 종료(상한값) 범위**

lower_bound ~ 하한값(시작)

upper_bound ~ 상한값(종료)

* REVERSE ~ UPPER BOUND에서 LOWER BOUND로 1씩 감소

4. 반복문

```
SET SERVEROUTPUT ON

DECLARE
    LOOP_INDEX              NUMBER(4) :=  1;
    MAX_LOOP_INDEX          NUMBER(4) := 30;
BEGIN
       FOR   LOOP_INDEX  IN  1.. MAX_LOOP_INDEX
    -- FOR   LOOP_INDEX  IN  30..1
    -- FOR   LOOP_INDEX  IN  REVERSE 1..30
    -- FOR   LOOP_INDEX  IN  REVERSE 30..1
    LOOP
            DBMS_OUTPUT.PUT_LINE('COUNT :'||TO_CHAR(LOOP_INDEX));
    END LOOP;
END;
/
```

4. 반복문

▣ WHILE Loop
기본적인 LOOP 반복문에 BOOLEAN CHECK 기능 추가
FOR LOOP는 일정한 횟수만큼 LOOP 반복
WHILE LOOP는 조건식이 참(TRUE)일 동안 반복.

조건식(condition)이 TRUE인 경우만 실행되고 FALSE 나 NULL인 경우 실행되지 않는다

```
WHILE  condition
LOOP
    statement1;
    statement2;
END  LOOP;
```

```
DECLARE
       V_INDEX                    NUMBER(3) := 0;          -- loop 횟수 체크
    --V_INDEX                     NUMBER(3);               -- NULL
    --V_INDEX                     NUMBER(3) := 30;         -- loop 횟수 체크
BEGIN
              WHILE(V_INDEX  >= 0 )
              LOOP
                        DBMS_OUTPUT.PUT_LINE(' While loop ['||TO_CHAR(V_INDEX)||']');
                        V_INDEX := V_INDEX - 1;
              END LOOP;
END;
/
```