<< Less 05 : Writing Control Structures >>

주요학습내용:

- IF 문
- CASE : CASE Expression , CASE Statement
- LOOP 문
  - BASIC LOOP : overall conditions 없이 반복적인 작업을 수행
  - FOR LOOP : count(횟수)에 기반한 반복작업을 수행
  - WHILE LOOP : 조건을 만족하는 동안에만 반복적인 작업을 수행

위의 실행문들은 BEGIN .. END; 사이에 기술합니다.

• 본 문서의 실습은 별도의 언급이 없는 한, SQL*Developer를 이용하여
hr 계정으로 데이터베이스에 원격하여 수행합니다.

<< IF 문 >>

(1) 조건을 만족할 때만 실행문들이 실행됩니다.
조건을 만족하지 않으면 실행되는 것이 없습니다.

```
IF
        조건1
                THEN 실행문1;
                     실행문2;
                     ...;
END IF;
```

==> 위와 같이 작성된 IF-문을
"단순 IF 문" 이라고 합니다.

(2) 조건을 만족할 때 THEN 이후에 기술된 실행문들 실행됩니다.
조건을 만족하지 않을 때 ELSE 이후에
작성된 실행문들이 실행됩니다.

```
IF
        조건1
                THEN 실행문1;
                     실행문2;
                     ...
ELSE    실행문11;
        실행문22;
        ...;
END IF;
```

(3) 조건을 여러가지를 적을 수 있습니다.

```
IF
        조건1
                THEN 실행문1;
                     실행문2;
                     ...;
ELSIF   조건2
```

```
                                THEN 실행문11;
                                     실행문22;

                [ELSIF - THEN -;]

                [ELSE   실행문111;
                        실행문222;
                        ...;]
               END IF ;
```

<< 단순 IF-문 예제 >>

```
SQL> SET SERVEROUTPUT ON

SQL> DECLARE
           myage NUMBER := 31;
     BEGIN
           IF myage < 11
            THEN  DBMS_OUTPUT.PUT_LINE(' I am a child ');
           END IF;
     END;
     /

PL/SQL procedure successfully completed.

SQL>
```

<< IF-THEN-ELSE 문 예제 >>

```
SQL> SET SERVEROUTPUT ON
SQL>

SQL> DECLARE
          myage NUMBER := 31;

     BEGIN
          IF myage < 11

             THEN
                DBMS_OUTPUT.PUT_LINE(' I am a child ');

          ELSE
                DBMS_OUTPUT.PUT_LINE(' I am not a child ');

          END IF;
     END;
     /
I am not a child

PL/SQL procedure successfully completed.

SQL>
```

<< IF ELSIF ELSE 예제 >>

```
SQL> SET VERIFY OFF
SQL>
SQL> SET SERVEROUTPUT ON
SQL>
```

```
SQL> DECLARE
        myage NUMBER := &age ;
     BEGIN
        IF myage < 11
                THEN  DBMS_OUTPUT.PUT_LINE(' I am a child ');
        ELSIF myage < 21
                THEN  DBMS_OUTPUT.PUT_LINE(' I am young ');
        ELSIF myage < 30
                THEN  DBMS_OUTPUT.PUT_LINE(' I am in my 20s');
        ELSIF myage < 40
                THEN  DBMS_OUTPUT.PUT_LINE(' I am in my 30s');
        ELSE   DBMS_OUTPUT.PUT_LINE(' I am always young ');
        END IF;
     END;
     /
Enter value for age: 10
I am a child

PL/SQL procedure successfully completed.

SQL>

SQL> /  -- SQL*Plus의 sql-buffer 맨 마지막에 실행한 SQL문장을 하나 저장.
Enter value for age: 20
I am I am young

PL/SQL procedure successfully completed.

SQL> -- SQL*Developer 에서는 / 를 사용할 수 없습니다.
     -- 워크시트에 있는 문장을   [Ctrl, Enter]키를 이용하여
     -- 다시 실행합니다.



SQL> /
Enter value for age: 30
I am in my 30s

PL/SQL procedure successfully completed.



(비교) 아래의 조건범위가 서로 중첩이 되는 경우에는
       처음에 만족하는 하나만 실행됩니다.

SQL> DECLARE
        myage number:=&age;
     BEGIN
         IF myage > 11
                THEN  DBMS_OUTPUT.PUT_LINE(' I am a child ');
         ELSIF myage > 20
                THEN  DBMS_OUTPUT.PUT_LINE(' I am young ');
         ELSIF myage > 30
                THEN  DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
         ELSIF myage > 40
                THEN  DBMS_OUTPUT.PUT_LINE(' I am in my thirties');
         ELSE
                DBMS_OUTPUT.PUT_LINE(' I am always young ');
        END IF;
     END;
     /
Enter value for age: 10
I am always young

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> /
Enter value for age: 50
I am a child

PL/SQL procedure successfully completed.

SQL>
```

<< IF-문 중첩 >>

```
IF condition1
THEN [statement1;]
     IF condition2
                          THEN statement2 ;
    ELSIF condition3      THEN statement3 ;
    ELSE    statement4;
     END IF;
ELSE
     IF condition5
                          THEN statement5 ;
     ELSIF condition6     THEN statement6 ;
     ELSE  statement7;
     END IF;
END IF ;
```

(참고) ELSIF 절(경우1)은 중첩 IF문(경우2)의 코드보다 읽기와
이해하기가 쉽고 논리도 명확하게 알아볼 수 있으므로
가능하면 ELSIF 절을 사용하세요.

(경우1) ELSIF 절이용

조건1을 만족하면 조건2/조건3 검사는 수행하지 않습니다.

```
IF 조건1 THEN 실행문들1;
ELSIF 조건2 THEN 실행문들2;
ELSIF 조건3 THEN 실행문들3;
ELSE  실행문들4;
END IF
```

(경우2) : 반복적인 IF-문 이용

==> IF-문 3개 매번 실행됩니다.

```
IF 조건1 THEN 실행문들1;
END IF ;

IF 조건2 THEN 실행문들2;
END IF ;

IF 조건3 THEN 실행문들3;
END IF ;
```

<< IF-문에서 조건이 NULL을 반환 시에 처리되는 거동 >>

- THEN 이하의 실행문들은 실행되지 않고
  ELSE 이후의 실행문들이 실행됩니다.

```
SQL> DECLARE
        myage NUMBER ;
     BEGIN
          IF myage < 11
                THEN DBMS_OUTPUT.PUT_LINE(' I am a child ');
          ELSE DBMS_OUTPUT.PUT_LINE(' I am not a child ');
          END IF;
     END;
     /
I am not a child

PL/SQL procedure successfully completed.

SQL>
```

(참고) IF-문을 작성하면서 아래와 같이 IF-문을 작성하면
좋지 못한 코드입니다.

아래에서 IF v_five_years THEN ....ELSE .....는
적을 필요가 없습니다.

```
 DECLARE
   v_hire_date  DATE := TO_DATE('2000/12/12','YYYY/MM/DD');
   v_five_years BOOLEAN;

 BEGIN
   IF (MONTHS_BETWEEN(SYSDATE, v_hire_date))/12 > 5
       THEN v_five_years := TRUE ;
            .........;
   ELSE
       v_five_years := FALSE ;
       .........;
   END IF ;

   IF v_five_years     -- 이 IF문이 필요없답니다.
      THEN ....        -- 그 이유를 스스로 생각해보십시오.
   ELSE .....

   ....
```

(참고) 세션의 지역과 언어가 한국어/한국으로 설정되어 있는 경우..

DATE 유형 변수에 날짜를 직접 지정하면서 언어설정을 함수에
사용하려면 NLS_DATE_LANGUAGE 옵션을 사용해서
날짜 형식을 지정하세요.

```
SQL> alter session set nls_language=KOREAN ;
     alter session set nls_territory=KOREA ;   -- RR/MM/DD

     --현재 한글Windows에서 실행중인 SQL*Developer의
     --설정 상태입니다.
```

```
SQL> DECLARE
        v_hire_date1  DATE := TO_DATE('12-DEC-2000','DD-MON-YYYY'
                                    ,'NLS_DATE_LANGUAGE=AMERICAN');


--      v_hire_date  DATE := TO_DATE('12-DEC-2000','DD-MON-YYYY'
--                                  ,'NLS_LANGUAGE=AMERICAN');     --에러

--      v_hire_date2  DATE := TO_DATE('12-DEC-2000','DD-MON-YYYY');--에러.
     BEGIN
          null ;
     end ;
     /
```

<< CASE Type >>

- CASE Expression
  - Simple CASE expression
  - Searched CASE expression
- CASE Statement

<< CASE-표현식(Expressions) >>

[기본 문법]

(1) 단순-CASE-표현식

```
CASE selector
        WHEN expression1 THEN result1
        WHEN expression2 THEN result2
        ...
        WHEN expressionN THEN resultN
[ELSE resultN+1]
END;
```

(2) 검색된(Searched)-CASE-표현식

- 아래에서 search_condition은
  부울값을 반환하는 검색조건입니다.

```
CASE WHEN search_condition1 THEN result1
     WHEN search_condition2 THEN result2
          ...
     WHEN search_conditionN THEN resultN
[ELSE resultN+1]
END;
```

<< CASE-표현식 예제 >>

```
SQL> SET SERVEROUTPUT ON
```

```
     SET VERIFY OFF

SQL> DECLARE
        grade     CHAR(1) := UPPER('&grade');
        appraisal VARCHAR2(20);
     BEGIN
        appraisal := CASE grade
                             WHEN 'A' THEN 'Excellent'
                             WHEN 'B' THEN 'Very Good'
                             WHEN 'C' THEN 'Good'
                     ELSE 'No such grade'
                     END;
        DBMS_OUTPUT.PUT_LINE ('Grade    : '|| grade );
        DBMS_OUTPUT.PUT_LINE ('Appraisal: '|| appraisal);
     END;
     /
Enter value for grade: d
Grade    : D
Appraisal: No such grade

PL/SQL procedure successfully completed.

SQL>
```

<< 검색된 CASE-표현식 예제 >>

```
SQL> SET SERVEROUTPUT ON
     SET VERIFY OFF

     DECLARE
        grade     CHAR(1) := UPPER('&grade');
        appraisal VARCHAR2(20);
     BEGIN
        appraisal := CASE
                         WHEN grade = 'A'        THEN 'Excellent'
                         WHEN grade IN ('B','C') THEN 'Good'
                     ELSE                             'No such grade'
                     END;
        DBMS_OUTPUT.PUT_LINE ('Grade    : '|| grade );
        DBMS_OUTPUT.PUT_LINE ('Appraisal: '|| appraisal);
     END;
     /
Enter value for grade: b
Grade    : B
Appraisal: Good

PL/SQL procedure successfully completed.

SQL>
```

<< CASE-문 >>

- 단순 CASE 문
- 검색된 CASE 문

```
SQL> SET SERVEROUTPUT ON
     SET VERIFY OFF

SQL> DECLARE
        deptid   NUMBER;
```

```
        deptname VARCHAR2(20);
        emps     NUMBER;
        mngid    NUMBER:= &m_id;
     BEGIN
        CASE mngid
          WHEN 108
           THEN SELECT department_id, department_name
                                          INTO deptid, deptname
                FROM   hr.departments
                WHERE  manager_id=108;

                SELECT count(*) INTO emps
                FROM   hr.employees
                WHERE  department_id=deptid;

                DBMS_OUTPUT.PUT_LINE (
                           'You are working in the '|| deptname
                    ||' department.');
                   DBMS_OUTPUT.PUT_LINE ('There are '  || emps
                           ||' employees in this department');
            ELSE
                DBMS_OUTPUT.PUT_LINE ('No Information');
        END CASE;
     END;
     /
Enter value for m_id: 107
No Information

PL/SQL procedure successfully completed.

SQL>


SQL> DECLARE
        deptid   NUMBER;
        deptname VARCHAR2(20);
        emps     NUMBER;
        mngid    NUMBER:= &m_id;
     BEGIN
        CASE
          WHEN mngid=108
           THEN SELECT department_id, department_name
                                          INTO deptid, deptname
                FROM   hr.departments
                WHERE  manager_id=108;

                SELECT count(*) INTO emps
                FROM   hr.employees
                WHERE  department_id=deptid;

                DBMS_OUTPUT.PUT_LINE (
                           'You are working in the '|| deptname
                    ||' department.');
                   DBMS_OUTPUT.PUT_LINE ('There are '  || emps
                           ||' employees in this department');
            ELSE
                DBMS_OUTPUT.PUT_LINE ('No Information');
        END CASE;
     END;
     /
Enter value for m_id: 108
You are working in the Finance department.
There are 6 employees in this department


PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

(주의)

- CASE-문은 END CASE; 로 끝내야 합니다.
- CASE-표현식 END 로 끝내야 합니다.

(참고)

- CASE문은 THEN 및 ELSE 절에
  실행문이 명시됩니다.
- CASE 표현식은 THEN 및 ELSE 절에
  실행문을 명시할 수 없습니다.
  변수에 할당할 값만 명시됩니다.

```
-- 아래의 코드는 위의 문을, CASE문 없이 실행문만 나열하여
-- 실행 시 입력한 사번의 관리자가 관리하는 부서이름과
-- 그 부서의 인원수를 표시하도록 수정한 코드입니다.

SQL> DECLARE
        deptid   NUMBER;
        deptname VARCHAR2(20);
        emps     NUMBER;
        mngid    NUMBER:= &m_id;
     BEGIN
        SELECT department_id, department_name INTO deptid, deptname
        FROM   departments
        WHERE  manager_id=mngid;

        SELECT count(*) INTO emps
        FROM   employees
        WHERE  department_id=deptid;

        DBMS_OUTPUT.PUT_LINE ( 'Department_name:  '|| deptname
                              ||' Employees Count: '|| emps );
    END;
    /
Enter value for m_id: 100
Department_name:  Executive Employees Count: 3

PL/SQL procedure successfully completed.

SQL>
SQL> /
Enter value for m_id: 101
DECLARE
*
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 7


SQL> /
Enter value for m_id: 103
Department_name:  IT Employees Count: 5

PL/SQL procedure successfully completed.

SQL>
```

<< Handling Nulls >>

- NULL을 포함하는 단순 비교 결과는 항상 NULL.
- NULL에 논리 연산자 NOT을 적용하면 결과는 NULL.
- 조건 제어문에서 조건의 결과가 NULL이면
  조건을 만족할 때 명령문 시퀀스가 실행되지 않음.

```
x := 5;
y := NULL;

IF x != y THEN ... -- 조건이 TRUE가 아닌 NULL 널을 반환
                   -- 실행문들이 실행되지 않음.
END IF;


a := NULL;
b := NULL;
IF a = b THEN ...  -- 조건이 TRUE가 아닌 NULL 널을 반환
                   -- 실행문들이 실행되지 않음.
END IF;
```

(참고) Logic Tables

(조건2) AND	(조건1) TRUE	(조건1) FALSE	(조건1) NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

(조건2) OR	(조건1) TRUE	(조건1) FALSE	(조건1) NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

(조건1) NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

[팁] AND 연산자로 2개의 조건을 기술할 때,
FALSE로 결정되는 경우가 많은 조건을 먼저 기술하세요.

OR 연산자로 2개의 조건을 기술할 때,
TRUE로 결정되는 경우가 많은 조건을 먼저 기술하세요.

<< Boolean Conditions >>

```
flag := reorder_flag AND available_flag;
```

REORDER_FLAG	AVAILABLE_FLAG	FLAG
TRUE	TRUE	?
TRUE	FALSE	?
NULL	TRUE	?
NULL	FALSE	?

<< 반복(Iterative) 제어: LOOP-문 >>

- LOOP-문 : 실행문(들)을 반복해서 여러번 실행.

• LOOP-문의 TYPE

(1) 기본 루프(BASIC LOOP)

==> 조건 없이 반복작업을 수행

반드시 EXIT 절을 명시하여
LOOP를 끝낼 조건절을 가져야 함.

(2) WHILE LOOP

==> 조건을 만족하는 동안에만
명령문들을 반복적으로 실행합니다.

(3) FOR LOOP

==> 횟수를 기준으로 작업의 반복을 제어

<< 기본 루프 >>

기본루프에서는 루프를 시작할 때, 이미 EXIT WHEN 조건이
만족되었더라도 해당 명령문을 적어도 한 번 이상은 실행하며
EXIT 문이 없으면 루프가 끝없이 반복실행됩니다.

[문법]

```
LOOP
        statement1;
        ...
        EXIT [WHEN condition]; -- 부울변수 또는 표현식
                                  (TRUE, FALSE, NULL)
END LOOP;
```

<< 기본-LOOP 실습 >>

1> hr.locations2 테이블의 행 갯수를 확인

```
SQL> select count(*) from hr.locations2 ;

COUNT(*)
--------
      23
```

2> hr.locations2.country_id 컬럼값이 CA인 행들을 확인

```
SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

LOCATION_ID CITY           CO
----------- -------------- --
       1800 Toronto        CA
       1900 Whitehorse     CA
```

==> LOCATION_ID 컬럼값을 확인합니다.

3> 위의 정보를 이용하여 아래의 익명블록을 작성해서 실행

```
SQL> DECLARE
        countryid  hr.locations2.country_id%TYPE := 'CA';
        new_city   hr.locations2.city%TYPE       := 'Montreal';
        loc_id     hr.locations2.location_id%TYPE;
        counter    NUMBER(2) := 1;

     BEGIN
                SELECT MAX(location_id) INTO loc_id
                FROM   hr.locations2
                WHERE  country_id = countryid;

                LOOP
                   INSERT INTO locations2(location_id, city, country_id)
                   VALUES((loc_id + counter), new_city, countryid);

                   counter := counter + 1;
                   EXIT;--  WHEN 이하를 빼면, 한번만 실행됩니다.
                END LOOP;
     END;
     /

PL/SQL procedure successfully completed.

SQL>



4> hr.locations2.country_id 컬럼값이 CA인 행들을 확인

SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

        LOCATION_ID CITY            CO
        ----------- --------------- --
               1800 Toronto         CA
               1900 Whitehorse      CA
               1901 Montreal        CA

SQL>

5> 트랜잭션 롤백합니다.

SQL> rollback ;
SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

LOCATION_ID CITY            CO
----------- --------------- --
       1800 Toronto         CA
       1900 Whitehorse      CA

SQL>


--기본 루프문을 이용하여 아래의 3 개 행을  hr.locations2 테이블에
  입력하겠습니다.

               1901 Montreal        CA
               1902 Montreal        CA
               1903 Montreal        CA


-- 입력작업을 수행하는 아래의 익명블록을 작성해서 실행

SQL> DECLARE
```

```
        countryid     hr.locations2.country_id%TYPE := 'CA';
        new_city      hr.locations2.city%TYPE       := 'Montreal';
        loc_id        hr.locations2.location_id%TYPE;
        counter       NUMBER(2) := 1;

     BEGIN
                SELECT MAX(location_id) INTO loc_id
                FROM hr.locations2
                WHERE country_id = countryid;

                LOOP
                   INSERT INTO hr.locations2(location_id, city, country_id)
                   VALUES((loc_id + counter), new_city, countryid);
                   counter := counter + 1;
                   EXIT WHEN counter > 3;
                END LOOP;
    END;
    /

PL/SQL procedure successfully completed.

SQL>



SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

LOCATION_ID CITY            CO
----------- --------------- --
       1800 Toronto         CA
       1900 Whitehorse      CA
       1901 Montreal        CA
       1902 Montreal        CA
       1903 Montreal        CA

SQL>


SQL> rollback ;



-- 위의 정보를 이용하여 아래의 익명블록을 작성해서 실행

SQL> DECLARE
        countryid     hr.locations2.country_id%TYPE := 'CA';
        new_city      hr.locations2.city%TYPE       := 'Montreal';
        loc_id        hr.locations2.location_id%TYPE;
        counter       NUMBER(2) := 1;

     BEGIN
                SELECT MAX(location_id) INTO loc_id
                FROM   hr.locations2
                WHERE  country_id = countryid;

                LOOP
                   INSERT INTO hr.locations2(location_id, city, country_id)
                   VALUES((loc_id + counter), new_city, countryid);
                   counter := counter + 1; -- 주석처리 시 무한 루프
--                 EXIT WHEN counter > 3;  -- 주석처리 Exit 명시가 안됨.
                END LOOP;
        END;
        /
        ERROR at line 1:
        ORA-06502: PL/SQL: numeric or value error: number precision too large
```

```
        ORA-06512: at line 14
```

<< WHILE Loops : 조건이 TRUE인 동안만 실행문들을 반복해서 실행. >>

```
        [문법]

        WHILE condition LOOP  <-- 반복이 시작될 때마다 조건이
                statement1;       평가됩니다.
                statement2;
                . . .
        END LOOP;
        ============================================================
        (참고) 조건에 포함된 변수가 루프 body내에서 변경되지 않으면
               조건이 TRUE로 유지되므로 루프가 종료되지 않습니다.

               조건의 결과가 NULL이면 제어는 루프를 건너뛰어
               다음 명령문으로 전달됩니다.
        ============================================================

SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

LOCATION_ID CITY            CO
----------- --------------- --
       1800 Toronto         CA
       1900 Whitehorse      CA

SQL>

-- 다음의 WHILE LOOP 문이 포함된 PL/SQL 익명블록 코드를 작성 후,
-- 실행시킵니다.

SQL> DECLARE
        countryid     hr.locations2.country_id%TYPE := 'CA';
        new_city      hr.locations2.city%TYPE       := 'Montreal';
        loc_id        hr.locations2.location_id%TYPE;
        counter       NUMBER(2) := 1;

     BEGIN
                SELECT MAX(location_id) INTO loc_id
                FROM   hr.locations2
                WHERE  country_id = countryid;

                WHILE counter <= 3
                LOOP
                        INSERT INTO hr.locations2
                                    (location_id, city, country_id)
                        VALUES((loc_id + counter), new_city, countryid);
                        counter := counter + 1;
--                      exit when counter=2 ;-- 주석처리됨
                END LOOP;
     END;
     /

PL/SQL procedure successfully completed.

SQL>


--실행결과 확인
```

```
SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

LOCATION_ID CITY           CO
----------- -------------- --
       1800 Toronto        CA
       1900 Whitehorse     CA
       1901 Montreal       CA
       1902 Montreal       CA
       1903 Montreal       CA

SQL>


SQL> rollback ;
```

(참고) 조건이 처음부터 거짓인 경우에는 아래처럼 실행이 안됨.

```
SQL> DECLARE
        countryid     hr.locations2.country_id%TYPE := 'CA';
        new_city      hr.locations2.city%TYPE       := 'Montreal';
        loc_id        hr.locations2.location_id%TYPE;
        counter       NUMBER(2) := 4;
     BEGIN
                SELECT MAX(location_id) INTO loc_id
                FROM hr.locations2
                WHERE country_id = countryid;

                WHILE counter <= 3        -- 처음부터 조건이 거짓입니다.
                LOOP
                    INSERT INTO locations2(location_id, city, country_id)
                    VALUES((loc_id + counter), new_city, countryid);

                    counter := counter + 1;
                END LOOP;
     END;
     /

PL/SQL procedure successfully completed.

SQL>
```

==========

<< FOR Loops >>

==========

- 반복횟수를 손쉽게 테스트 가능

[문법]

```
FOR counter IN [REVERSE] lower_bound..upper_bound
   LOOP
        statement1;
        statement2;
        . . .
END LOOP;
```

(작성 예)

```
FOR i IN 3..3       <-- lower_bound와 upper_bound 가 같으면
   LOOP                 한번만 실행됩니다.
        statement1;
END LOOP;
```

```
FOR i IN 1..10      <-- LOOP 내의 실행문들이
   LOOP                 10번 반복 실행됩니다.
        statement1;
END LOOP;
```

(주의) 항상 lower_bound가 upper_bound 보다 작아야 합니다.

- counter는 암시적으로 선언되므로 선언하지 마세요.
- lower_bound..upper_bound 구문이 반드시 필요함.
  : 리터럴, 변수, 또는 표현식이 될 수 있지만
  반드시 정수로 평가되어야 합니다.
- counter는 루프 내에서만 참조할 수 있으며 루프 밖에서는
  정의되지 않습니다.
- counter에 다른 값을 할당할 수 없습니다.

  예, i가 FOR-LOOP문의 counter인 경우,
  i:=i+1; 이러한 실행문이 불가능합니다.

- loop bound는 절대로 NULL 이면 안됨.

<< FOR LOOP 예제 >>

```
SQL> DECLARE
        countryid     hr.locations2.country_id%TYPE := 'CA';
        new_city      hr.locations2.city%TYPE       := 'Montreal';
        loc_id        hr.locations2.location_id%TYPE;
     BEGIN
                SELECT MAX(location_id) INTO loc_id
                FROM hr.locations2
                WHERE country_id = countryid;

--              FOR i IN 1..5 LOOP
                FOR i IN REVERSE 1..5 LOOP

                  INSERT INTO hr.locations2(location_id, city, country_id)
                  VALUES((loc_id + i), new_city, countryid );

--                EXIT WHEN SQL%ROWCOUNT = 1 ;

--                i:=i+1; -- counter를 할당 대상으로 참조하지 말것.

               END LOOP;
     END;
     /

PL/SQL procedure successfully completed.

SQL>
SQL> select location_id, city, country_id
     from hr.locations2
     where  country_id='CA' ;

LOCATION_ID CITY            CO
----------- --------------- --
       1800 Toronto         CA
       1900 Whitehorse      CA
       1905 Montreal        CA
```

```
       1904 Montreal        CA
       1903 Montreal        CA
       1902 Montreal        CA
       1901 Montreal        CA

7 rows selected.

SQL>
```

<< LOOP 문 사용 시 지침 >>

- 루프 내의 명령문을 적어도 한 번은 실행해야 한다면,
  BASIC LOOP 사용

- 조건을 사용하여 해당 조건을 만족하는 동안에만
  실행되길 원하면,
  WHILE LOOP 사용

- 반복횟수를 알고 있다면 FOR LOOP를 사용

<< Nested Loops and Labels >>

- 루프는 여러 레벨로 중첩가능

- 레이블을 이용하여 블록과 루프를 구분

- 레이블을 참조하는 EXIT 문을 사용하여 외부 루프를 종료.

- 루프에 레이블을 지정할 경우 END LOOP 문 다음에
  레이블 이름을
  포함시켜 명확하게 할 수도 있습니다(선택사항)

- Nested LOOP 예제

```
BEGIN
        FOR i IN 1..10
        LOOP
                FOR j IN 1..10
                LOOP
                  null;
                END LOOP;
        END LOOP;
END;
/
```

```
DECLARE
        counter   number(3)  := 0 ;
BEGIN
        <<Outer_loop>>
        LOOP
          counter := counter+1;
        EXIT WHEN counter > 10 ;
              <<Inner_loop>>
               LOOP
                 ...
                 EXIT Outer_loop WHEN total_done = 'YES';
                 -- Leave both loops
```

```
                EXIT Inner_loop WHEN inner_done = 'YES';
                -- Leave inner loop only
                ...
                END LOOP Inner_loop;
        ...
        END LOOP Outer_loop;
END;
/
```