<< Less 02 : PL/SQL 변수(VARIABLES) 선언(DECLARE) >>

중요 학습 내용:

- 변수 선언방법 및 초기화 방법
- 변수(Variable) 타입
- tablename.columnname%TYPE
- SQL*Developer/SQL*Plus에서 bind 변수 선언 및 사용방법
- SQL*Developer/SQL*Plus에서 bind 변수 출력
  (SQL*Developer/SQL*Plus의 print 명령어)
- SQL*Developer/SQL*Plus 치환변수

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

<< 변수 용도 >>

• Temporary storage of data

• Manipulation of stored values

• Reusability

<< PL/SQL 코드에서 변수(VARIABLES)를 사용하는 방법 >>

• 변수는

  - declarative section(DECLARE 부분)에서 선언 및 초기화해야 함

  - executable section(BEGIN~ END ;)에서 사용되고
    또 새로운 값으로 변경(assigned)됩니다.

<< 식별자(Identifiers)- 변수(VARIABLES)의 이름 >>

- 변수 선언 시에 설정된 이름을 식별자라고 합니다.

<< PL/SQL 변수 선언 및 초기화 지침 관련 실습 >>

• 변수 이름은 다음의 규칙을 지켜야 합니다.
  - 숫자가 아닌 영문자로 시작해야합니다.
  - 영문자(letters) 또는 숫자(numbers)만 포함될 수 있습니다.
  - $ _ # 의 특수문자만 포함될 수 있습니다.
  - 30 Bytes를 넘을 수 없습니다.
  - 예약어는 사용될 수 없습니다.

• 변수의 이름 지정규칙(naming conventions)를 지켜야 합니다.
  - 예를들면, 개발초기에 여러 개발자들이 모여서
    "우리가 PL/SQL 프로그램을 작성할 때,
    코드 안에 변수들 이름은 v_로 시작합시다" 라는 규칙 처럼
    자체적으로 정한 규칙을 "변수이름 지정 규칙"이라고 합니다.

• 변수를 상수로 지정하고 싶을 때는 CONSTANT 키워드를 명시하고

반드시 초기값을 지정하여 변수를 상수로 지정합니다.

- NOT NULL이 정의되어 있으면 반드시 초기값을 지정함
- 초기값을 지정하지 않으면 변수 식별자는 NULL 값을 가짐.
- 일반적으로 한 줄에 한 개의 변수 식별자를 선언하는 것이 권장.

- 변수 선언 시에 초기값 지정은, 할당 연산자(:=)나
  DEFAULT 키워드를 이용하여 정의함.

```
v_myname1 VARCHAR2(20):='John';
v_myname2 VARCHAR2(20) DEFAULT 'John';
```

- 테이블명이나 테이블의 컬럼명을 변수나 상수의 이름(식별자)으로
  사용하지 말것.

<< PL/SQL 코드에서 변수를 선언하고 초기화하는 방법 >>

[변수 선언 및 초기화 문법]

```
identifier [CONSTANT] datatype [NOT NULL][DEFAULT expr];
identifier [CONSTANT] datatype [NOT NULL][:= expr];
```

[설명 예제]

```
DECLARE
-- 1. NULL 로 초기화
    v_emp_hiredate        DATE ;

-- 2. 지정된 값으로 초기화
--    << 초기화방법 >>
--       (1) 할당연산자(:=)를 사용
--       (2) DEFAULT 키워드를 사용

    v_location   VARCHAR2(13)          := 'Atlanta';
    v_addr       varchar2(30)           default 'Seoul' ;
    v_emp_deptno NUMBER(2)    NOT NULL := 10;


--(주의) NOT NULL 규칙을 변수에 지정할 수 있고,
--       NOT NULL 규칙이 사용된 경우에는
--       반드시 특정값으로 초기화시켜야만 합니다


-- 3. CONSTANT 키워드를 사용하면 상수를 선언
--    (주의: 상수는 선언 시에 반드시 특정 값을
--           지정해서 초기화시켜야만 합니다)

    v_comm      CONSTANT NUMBER            := 1400;

--     상수는 BEGIN..END; 사이에서
--     새로운 값으로 변경이 불가합니다.
```

<< 변수 선언 및 초기화 실습 예제 >>

1> SQL*Developer를 이용하여 hr 계정으로 데이터베이스에 원격접속합니다.

2> SQL*Developer 에서
   DBMS_OUTPUT.PUT_LINE 프로시저의 실행 결과를 표시하기 위하여
   출력 옵션 설정

```
SQL> SET SERVEROUTPUT ON;
SQL>
```

3> 다음의 익명블록을 작성하여 실행합니다.

```
SQL>
DECLARE
        v_myname VARCHAR2(20) ;
BEGIN
        DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
        v_myname := 'John';
        DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
END;
/
My name is:
My name is: John

PL/SQL procedure successfully completed.

SQL>
```

4> 다음의 익명블록을 작성하여 실행합니다.

```
SQL>
SET SERVEROUTPUT ON

SQL>
DECLARE
        v_myname VARCHAR2(20) := 'John';
BEGIN
        DBMS_OUTPUT.PUT_LINE('My name's: '||v_myname); --오류발생
        v_myname := 'Steven';
        DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
END;
/
ERROR:
ORA-01756: quoted string not properly terminated


SQL> -- 'My name's: ' 리터럴 문자열 내에 작은 따옴표 때문에
     -- 오류가 발생되었습니다.
```

5> 다음의 익명블록을 작성하여 실행합니다.

```
SQL>
DECLARE
        myname VARCHAR2(20) := 'John';
BEGIN
        DBMS_OUTPUT.PUT_LINE('My name''s: '||myname);
        myname := 'Steven';
        DBMS_OUTPUT.PUT_LINE('My name is: '||myname);
END;
/
My name's: John
My name is: Steven

PL/SQL procedure successfully completed.
```

```
SQL> -- 앞에서 오류난 'My name's: ' 리터럴 문자열 내에
     -- 작은 따옴표를 'My name''s: '두 번 명시했습니다.

SQL> --아래 코드는 작은 따옴표 두 번 대신 오라클의  q 연산자를
     --사용하여 오류를 방지했습니다

SQL>
DECLARE
        myname VARCHAR2(20) := 'John';
BEGIN
        DBMS_OUTPUT.PUT_LINE(q'[My name's:/// ]'||myname); --  에러
        myname := 'Steven';
        DBMS_OUTPUT.PUT_LINE('My name is: '||myname);
END;
/
My name's:/// John
My name is: Steven

PL/SQL procedure successfully completed.
```

(참고) q 연산자 (< > { } [ ] ( ))

(참고) Delimiters : 연산자, 구분자등의 기호를 의미합니다.

6> 다음의 익명블록을 작성하여 실행합니다.

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL>
DECLARE
        v_event VARCHAR2(15) ;
BEGIN
        v_event := 'Father''s day'; -- 문자열내에 포함된 작은따옴표(') 출력: 두번적어주세요
        DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :'||v_event);
        v_event := q'[Mother's day]'; -- 위의 작은따옴표 ' 두 번 적는 것을
                                      -- 연산자(q)로 구현
        DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is  :'||v_event);
END;
/

3rd Sunday in June is :Father's day
2nd Sunday in May is  :Mother's day

PL/SQL procedure successfully completed.

SQL>
```

<< 변수의 유형 >>

1) PL/SQL 변수: PL/SQL 코드 안에서 선언 및 초기화 되어 사용되는
변수의 유형에는 다음의 것들이 있습니다.

- Scalar 유형: 하나의 값을 할당받는 변수
- Composite: 하나 이상의 값을 할당받는 변수
- Reference: 포인터 역할을 하는 변수
- Large object (LOB): 길이가 아주 긴 값을 할당하는 변수

2) Non-PL/SQL 변수
   : PL/SQL 프로그램이 호출된 환경(예, SQL*Developer/SQL*Plus)에서
     선언 및 초기화되는 변수

        • Bind-변수(Host-변수)
        • 치환 변수

<< PL/SQL 변수 선언 및 초기화 지침 관련 실습 >>

7> 다음의 익명블록을 작성하여 실행합니다.

```
SQL>
SET SERVEROUTPUT ON

SQL>
DECLARE

        v_employee_id NUMBER(6);

BEGIN

        SELECT employee_id INTO v_employee_id
        FROM hr.employees
        WHERE last_name = 'Kochhar';

        DBMS_OUTPUT.PUT_LINE(v_employee_id);
END;
/
101

PL/SQL procedure successfully completed.

SQL>
```

<< 대표적인 스칼라 데이터 유형 >>

   단순 데이터형으로 하나의 데이터 값만 가지는
   데이터 유형으로 대표적으로 다음의 유형이 해당됩니다.
   - CHAR
   - VARCHAR2
   - NUMBER
   - PLS_INTEGER, BINARY_INTEGER
   - BOOLEAN

<< 기본 스칼라 데이터 유형(Base Scalar Data Types) >>

   CHAR(maximum_length)
   : 최대 32767 바이트 길이의 고정길이 문자를 표현

   VARCHAR2(maximum_length)
   : 최대 32767 바이트 길이의 가변길이 문자를 표현

   NUMBER(precision, scale)
   : 22 byte 사용하여 실수(정수 포함)를 표현

   PLS_INTEGER 및 BINARY_INTEGER
   : 4바이트를 이용하여 정수를 표현

   BOOLEAN
   : TRUE, FALSE, NULL 값을 가짐

```
--(참고) 오라클 데이터베이스의 테이블의 컬럼에 지정하는
--       VARCHAR2, CHAR 데이터유형은 최대길이가
--       각각 4000바이트, 2000바이트입니다.
--
--       PL/SQL 코드의 변수에 지정하는 VARCHAR2, CHAR 데이터유형은
--       최대길이는 32767 바이트입니다.


--(참고) 오라클 데이터베이스에서 정수값을 처리할 때는
--       NUMBER 보다는 PLS_INTEGER 또는 BINARY_INTEGER
--       두 데이터유형으로 처리하는 것을 권장합니다.
--
--       오라클 11g 버전부터는 PLS_INTEGER 와 BINARY_INTEGER가
--       동일합니다.
```

<< 스칼라 데이터 유형을 사용하는 변수 선언 방법 >>

```
        DECLARE
           c_tax_rate     CONSTANT NUMBER(3,2) := 8.25;
           count_loop     BINARY_INTEGER   := 0;
           dept_total_sal NUMBER(9,2)      := 0;
           orderdate      DATE             := SYSDATE + 7;
           valid          BOOLEAN NOT NULL := TRUE;
           emp_job        VARCHAR2(9);
           v_tel          VARCHAR2(9);
           v_jid          emp_job%TYPE;
           emp_lname      hr.employees.last_name%TYPE;
           ...
```

<< %TYPE 속성(Attribute) >>

- "먼저 선언된 다른 변수"나
  "데이터베이스 테이블의 컬럼"의 데이터 타입을 그대로 사용하고자
  할 때, 코드에 명시합니다.

- %TYPE Attribute 사용시의 장점
  : 코딩 이후 데이터베이스 컬럼의 데이터 타입의 속성이 변경될 경우
    다시 수정할 필요가 없음.

예를 들면,

```
DECLARE
   emp_lname1  varchar2(30)
   emp_lname2  hr.employees.last_name%TYPE NOT NULL := 'No Name';

  -- hr.employees.last_name 컬럼에
  -- NOT NULL 제약조건이 적용되어 있다면
  -- emp_lname2 변수에도 NOT NULL 을 명시해 줍니다.


   balance     NUMBER(7,2);
   min_balance balance%TYPE := 1000;
```

9> 다음의 익명블록을 작성하여 실행합니다.
   오류가 발생됩니다.

```
SQL>
DECLARE
   v_name varchar2(3);
BEGIN
   SELECT first_name into v_name
   FROM hr.employees
```

```
   WHERE employee_id = 100;

   DBMS_OUTPUT.PUT_LINE(
      'The First Name of the Employee is '||v_name );
END ;
/
DECLARE
*
ERROR at line 1:
ORA-06502: PL/SQL: numeric or value error: character string buffer too small
ORA-06512: at line 4


SQL>  -- 테이블의 컬럼으로부터 가지고 온 값의 길이가
      -- f_name 변수의 길이보다 길기 때문에 오류가 발생됩니다.



--아래의 코드는 위의 코드에서  f_name 변수의 데이터유형을
--%TYPE을 이용하여 수정한 코드입니다.

SQL>
DECLARE
   v_name hr.employees.first_name%TYPE ;   --수정됨
BEGIN
   SELECT first_name into v_name
   FROM hr.employees
   WHERE employee_id = 100;

   DBMS_OUTPUT.PUT_LINE(
     'The First Name of the Employee is '||v_name );
END ;
/
The First Name of the Employee is Steven

PL/SQL procedure successfully completed.

SQL>




<< Boolean 데이터 유형 변수 선언 실습 >>

SQL>
DECLARE
   v_flag1 BOOLEAN := FALSE;
   v_flag2 BOOLEAN := NULL;
   v_emp_sal1 PLS_INTEGER := 50000;
   v_emp_sal2 PLS_INTEGER := 60000;
BEGIN
   v_flag1 := TRUE;
   v_flag2 := v_emp_sal1 < v_emp_sal2;  -- v_flag2:=TRUE
--     DBMS_OUTPUT.PUT_LINE(v_flag1);
--     DBMS_OUTPUT.PUT_LINE(v_flag2);
END;
/

PL/SQL 프로시저가 성공적으로 완료되었습니다.

--위의 코드에서 주석기호(--)를 지우고 실행하면,
--Boolean 변수는 할당된 값(TRUE, FALSE, NULL)을
--DBMS_OUTPUT.PUT_LINE(flag1); 를 이용해서 표시할 수 없기때문에
--오류가 발생됩니다.


--아래코드는 flag2 변수값을 v_flag1 문자유형 변수를 통해서
--해당 값을 표시하도록 수정된 코드입니다.
```

```
SQL>
DECLARE
   v_flag1 BOOLEAN := FALSE;
   v_flag2 BOOLEAN := NULL;
   v_flag11  VARCHAR2(5) ;
   v_emp_sal1 PLS_INTEGER := 50000;
   v_emp_sal2 PLS_INTEGER := 60000;
BEGIN
   v_flag1 := TRUE;
   v_flag2 := v_emp_sal1 < v_emp_sal2;  -- v_flag2:=TRUE
   IF v_flag1 THEN v_flag11 :='TRUE';
   ELSE v_flag11 :='FALSE';
   END IF;
   DBMS_OUTPUT.PUT_LINE(v_flag11);
--     DBMS_OUTPUT.PUT_LINE(v_flag2);
END;
/
TRUE


PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

<< BINARY_FLOAT 및 BINARY_DOUBLE 데이터 유형 사용 실습 >>

8> 다음의 익명블록을 작성하여 실행합니다.

```
SQL>
SET SERVEROUTPUT ON

SQL>
DECLARE
   bf_var BINARY_FLOAT;
   bd_var BINARY_DOUBLE;
BEGIN
   bf_var := 270/35f;
   bd_var := 140d;
       DBMS_OUTPUT.PUT_LINE('bf: '|| bf_var);
       DBMS_OUTPUT.PUT_LINE('bd: '|| bd_var);
END;
/
bf: 7.71428585E+000
bd: 1.4E+002

PL/SQL procedure successfully completed.

SQL>
```

<< Bind-변수와 SQL*Developer VARIABLE 명령어 >>

- BIND-변수의 선언 및 초기화가
  Environment(PL/SQL block module이 호출 및 실행되는 환경)
  에서 일어납니다.

- PL/SQL block에서는 환경에서 선언된 BIND-변수를
  호출 해서 사용만 합니다.

```
--SQL*Developer에서 바인드-변수 선언
```

```
SQL> VARIABLE emp_salary NUMBER   -- 숫자 유형의 BIND-변수를 선언 시에
SQL>                                 길이를 명시할 수 없습니다.

SQL> VARIABLE emp_salary NUMBER(2)
Usage: VAR[IABLE] [ <variable> [ NUMBER | CHAR | CHAR (n [CHAR|BYTE]) |
                   VARCHAR2 (n [CHAR|BYTE]) | NCHAR | NCHAR (n) |
                   NVARCHAR2 (n) | CLOB | NCLOB | BLOB | BFILE
                   REFCURSOR | BINARY_FLOAT | BINARY_DOUBLE ] ]
SQL>
        ==> 숫자 유형의 BIND-변수를 선언 시에 길이를 명시하면
            위와 같이 사용법을 표시합니다(에러).


SQL> VARIABLE emp_lname VARCHAR2(30)
SQL>

        ==> 문자 형식 CHAR, VARCHAR2 의 경우에는 최대 허용 길이를
            명시할 수 있습니다.


SQL> VARIABLE emp_lname VARCHAR2(4001)
SP2-0676: Bind variable length cannot exceed 4000
SQL>

        ==> BIND-변수의 경우에, VARCHAR2 데이터 유형은 최대길이가
            4000 BYTE 입니다.


        (참고) PL/SQL 코드 내에서의 VARCAHR2의 최대허용 길이 범위는
               1 부터 32767 BYTE 입니다.



<< SQL*Plus PRINT 명령어 >>

        - BIND-변수에 할당된 값을 표시해 줍니다.

-- BIND-변수 선언

SQL> VARIABLE emp_salary NUMBER
SQL>


SQL>
BEGIN
   SELECT salary INTO :emp_salary
   FROM employees
   WHERE employee_id = 100;
END;
/
PL/SQL procedure successfully completed.

SQL>


-- BIND-변수에 할당된 값을 확인

SQL> PRINT emp_salary

EMP_SALARY
----------
     24000

SQL>
```

```
SQL> SELECT first_name, last_name , salary
     FROM  employees
     WHERE salary=:emp_salary;

FIRST_NAME      LAST_NAME           SALARY
--------------- --------------- ----------
Steven          King                 24000

SQL>

-- 아래처럼 SQL*Plus의 AUTOPRINT 옵션을 ON으로 설정하면
   PRINT 명령어를 사용하지 않더라도 BIND-변수에 할당된 값을
   자동으로 표시해 줍니다.

SQL> SET AUTOPRINT ON
SQL>

-- 아래에서 VARIABLE emp_salary NUMBER 명령어를
-- 다시 실행하면, 위에서 사용하여 할당된 값이 삭제됩니다.

SQL> VARIABLE emp_salary NUMBER

SQL> print emp_salary

EMP_SALARY
----------




-- BIND-변수 선언

SQL> VARIABLE emp_salary NUMBER
SQL>

-- 선언된 BIND-변수에 값을 할당합니다.

SQL> execute :emp_salary := 1000

PL/SQL procedure successfully completed.


EMP_SALARY
----------
      1000

SQL> -- SET AUTOPRINT ON 설정때문에 실행과 동시에
     -- BIND-변수에 할당값이 자동으로 표시됩니다.




<< 치환변수 사용 예제 >>

SQL> VARIABLE emp_salary NUMBER  --바인드-변수 선언
SQL>
SQL> SET AUTOPRINT ON  --바인드-변수에 할당값을 자동으로 표시
SQL>
SQL>
DECLARE
    v_empno NUMBER(6):=&s_empno;
BEGIN
    SELECT salary INTO :emp_salary
    FROM employees
    WHERE employee_id = v_empno;
END;
/
```

```
Enter value for s_empno: 100
old   2:        v_empno NUMBER(6):=&s_empno;
new   2:        v_empno NUMBER(6):=100;

PL/SQL procedure successfully completed.


EMP_SALARY
----------
     24000

SQL>

SQL> ;
  1  DECLARE
  2     v_empno NUMBER(6):=&s_empno;
  3       BEGIN
  4     SELECT salary INTO :emp_salary
  5     FROM employees
  6     WHERE employee_id = v_empno;
  7*      END;
SQL>
SQL> /
Enter value for s_empno: 900
old   2:        v_empno NUMBER(6):=&s_empno;
new   2:        v_empno NUMBER(6):=900;
DECLARE
*
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 4



EMP_SALARY
----------
     24000

SQL>
```

==> 이전 실행(employee_id=100)에서 할당된 값이 표시됩니다.

==========

<< SQL*Plus ACCEPT 명령어 >>

--DEFINE 명령어로 치환변수 선언하는 대신
ACCEPT 명령어는 치환변수 입력값을 물어오는 질문을 설정할 수
있습니다. 해당 치환변수에 입력된 값은 세션에 설정됩니다.

```
SQL> ACCEPT s_empno PROMPT '사번을 입력하시오: '
사번을 입력하시오: 100
SQL>
SQL> define s_empno -- 치환변수 값 확인
DEFINE S_EMPNO        = "100" (CHAR)
SQL>
```

-- ACCEPT 명령어 사용 실습

1> 메모장을 실행합니다. vi accept_test.sql

2> 아래의 === 사이의 내용을 복사 및 붙여넣거나 타이핑합니다.

3> 파일이름을 accept_test.txt 로 지정하고 c:₩ 디렉토리에
   저장합니다.

```
================================================================================
SET VERIFY OFF
VARIABLE emp_salary NUMBER
ACCEPT s_empno PROMPT '사번을 입력하세요: '
SET AUTOPRINT ON
DECLARE
 v_empno NUMBER(6):= &s_empno;
BEGIN
 SELECT salary INTO :emp_salary
 FROM hr.employees
 WHERE employee_id = v_empno;
END;
/
--print emp_salary
undefine s_empno
================================================================================
```

4> 오라클 서버가 실행 중인 윈도우즈 시스템에서
   CMD를 실행합니다.

5> 실행된 CMD에서 sqlplus를 이용하여 HR 계정으로 접속합니다.

```
C:₩Users₩oracle>sqlplus hr/oracle4U
```

6> 다음을 실행합니다.

```
SQL> @c:₩accept_test.txt
사번을 입력하세요: 100

PL/SQL 처리가 정상적으로 완료되었습니다.


EMP_SALARY
----------
     24000

SQL>
```

<< DEFINE 명령어를 사용한 치환변수 선언 >>

```
SQL> SET VERIFY OFF
SQL>
SQL> DEFINE lname= Urman
SQL>
SQL> SET SERVEROUT ON
SQL>
SQL>
DECLARE
   fname VARCHAR2(25) ;
BEGIN
   SELECT first_name INTO fname
   FROM hr.employees
   WHERE last_name= '&lname';
   DBMS_OUTPUT.PUT_LINE(fname) ;
END;
/
Jose Manuel

PL/SQL procedure successfully completed.
```

```
SQL>


-- 접속 종료

SQL> exit
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options 에
서 분리되었습니다.

C:₩Users₩oracle>
```