

다양한 예제로 쉽게 배우는

# 오라클 SQL 과 PL/SQL

# 15장 PL/SQL 변수

# 15. PL/SQL 변수

## - PL/SQL에서 변수를 사용하는 이유

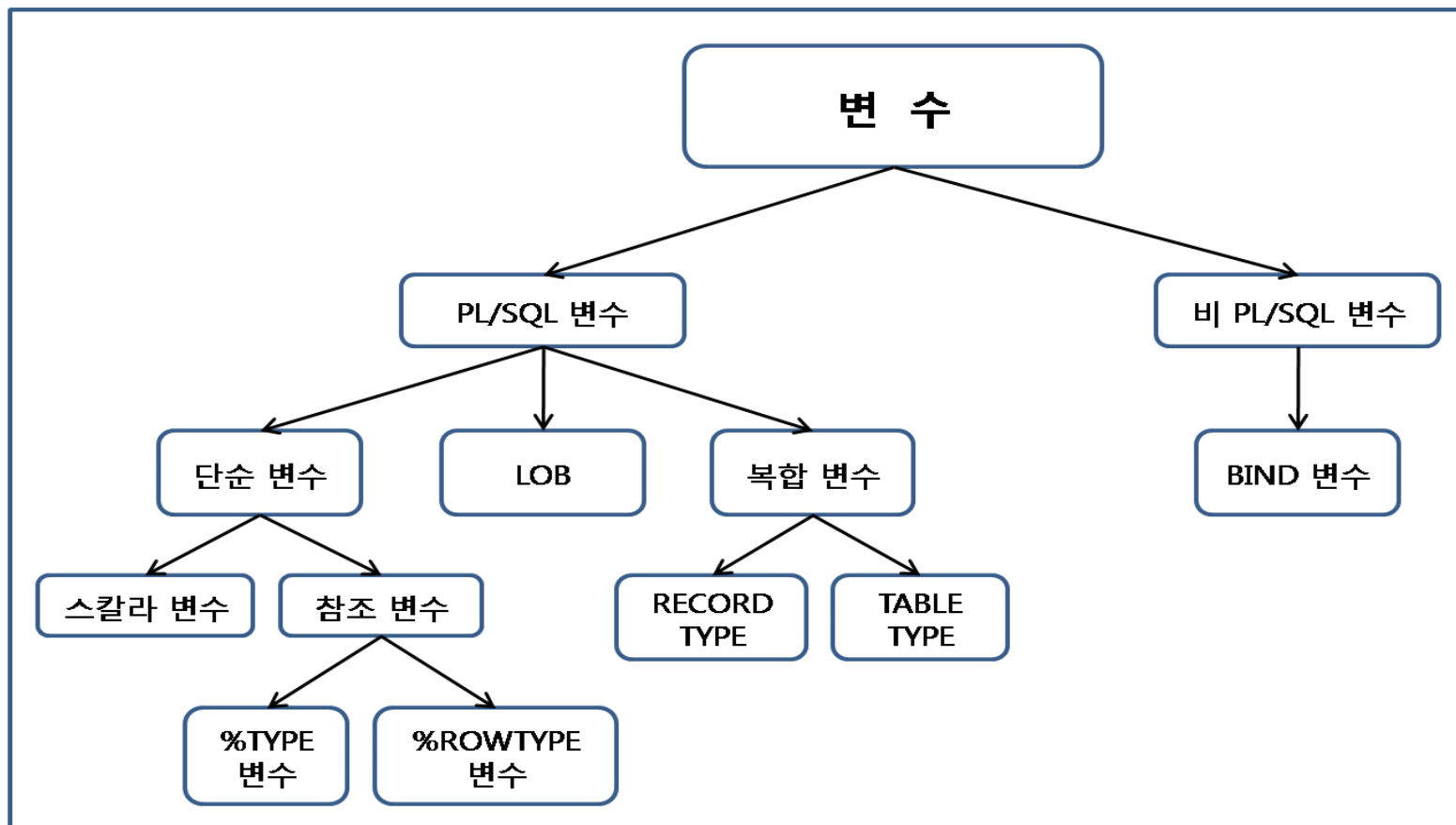
- 변수는 데이터의 임시 저장 영역입니다.
- 저장된 값을 조작하기 위해 사용합니다.
- 저장된 값을 반복해서 재 사용할 수 있습니다.

## - 변수 생성 규칙

- 반드시 문자로 시작해야만 합니다.
- 문자나 숫자, 특수문자를 포함할 수 있습니다.
- 변수명은 30 bytes 이하여야 합니다.
- 예약어를 포함하면 안됩니다.
- 선언부에서 선언되고 원한다면 특정 값으로 초기화도 가능합니다.
- 실행부에서 실행되면서 값이 할당이 됩니다.
- 서브 프로그램의 파라미터로 전달되기도 하며 서브 프로그램의 출력결과를 저장하기도 합니다.

# 15. PL/SQL 변수

## 2. 주요 변수들의 종류



# 15. PL/SQL 변수

## 1) 단순 변수

### - SCALAR 변수 와 Reference 변수

#### (1) SCALAR 변수

문 법:

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

### - 주요 스칼라 변수 선언 예

Vno number(5,3) <- 숫자를 저장하는 변수로 총 5자리이며 소수점 이하 3자리를 의미합니다.

Vname varchar2(10) <- 문자를 저장하는 변수로 총 10 바이트의 길이를 저장할 수 있습니다.

Vday date <- 날짜를 저장하는 변수입니다.

# 15. PL/SQL 변수

## - 주요 SCALAR 변수의 데이터 타입

- **CHAR [(최대길이)]**

이 타입은 고정 길이의 문자를 저장하며 최대 32,767 바이트 값을 저장합니다. 기본 최소값은 1로 설정되어 있습니다.

- **VARCHAR2 (최대길이)**

이 타입은 가변 길이의 문자를 저장하며 최대 32,767 바이트 값을 저장합니다. 기본 값은 없습니다.

- **NUMBER [(전체 자리 수, 소수점 이하 자리 수)]**

이 타입은 전체 자리수와 소수점 이하의 자리 수를 가진 숫자입니다. 전체 자리수의 범위는 1부터 38까지, 소수점 이하 자리수의 범위는 -84 부터 127 까지 입니다.

# 15. PL/SQL 변수

## - 주요 SCALAR 변수의 데이터 타입 - 계속

### • BINARY\_INTEGER

이 타입은 -2,147,483,647 - 2,147,483,647 사이의 정수를 저장하는 타입입니다.

### • PLS\_INTEGER

이 타입은 -2,147,483,647 - 2,147,483,647 사이의 부호 있는 정수에 대한 기본 유형입니다. PLS\_INTEGER 값은 NUMBER 값보다 저장 공간이 적게 필요하고 연산 속도가 더 빠릅니다. Oracle Database 11g에서는 PLS\_INTEGER 및 BINARY\_INTEGER 데이터 유형은 동일합니다. PLS\_INTEGER 및 BINARY\_INTEGER 값의 산술 연산은 NUMBER 값보다 빠릅니다.

### • BOOLEAN

이 타입은 논리적 계산에 사용 가능한 세 가지 값(TRUE, FALSE, NULL)중 하나를 저장하는 기본 유형입니다.

### • BINARY\_FLOAT

이 타입은 IEEE 754 형식의 부동 소수점 수를 나타냅니다. 값을 저장하기 위해 5바이트가 필요합니다.

# 15. PL/SQL 변수

## - 주요 SCALAR 변수의 데이터 타입 - 계속

- **BINARY\_DOUBLE**

이 타입은 IEEE 754 형식의 부동 소수점 수를 나타냅니다.  
값을 저장하기 위해 9바이트가 필요합니다.

- **DATE**

이 타입은 날짜 및 시간에 대한 기본 유형입니다. DATE 값은 자정 이후 경과한 시간을 초 단위로 포함합니다. 날짜의 범위는 4712 B.C. - 9999 A.D 사이입니다.

- **TIMESTAMP**

이 타입은 DATE 데이터 유형을 확장하고 연도, 월, 일, 시, 분, 초 및 소수로 표시되는 초 단위를 저장합니다. 구문은 `TIMESTAMP[(precision)]`이며 여기서 선택적 파라미터인 `precision`은 초 필드의 소수 부분 자릿수를 지정합니다.  
자릿수를 지정하려면 0 - 9 범위의 정수를 사용해야 합니다. 기본값은 6입니다.



# 15. PL/SQL 변수

## - 주요 SCALAR 변수의 데이터 타입 - 계속

### • TIMESTAMP WITH TIME ZONE

이 타입은 TIMESTAMP 데이터 유형을 확장하고 시간대 변위를 포함합니다.

시간대 변위는 로컬시간과 UTC(Coordinated Universal Time—이전의 그리니치 표준시)의 차이(시간과 분)입니다. 구문은 `TIMESTAMP[(precision)] WITH TIME ZONE`이며 여기서 선택적 파라미터 `precision`은 초 필드의 소수 부분 자릿수를 지정합니다.

자릿수를 지정하려면 0- 9 범위의 정수를 사용해야 합니다. 기본값은 6입니다.

### • TIMESTAMP WITH LOCAL TIME ZONE

이 타입은 TIMESTAMP 데이터 유형을 확장하고 시간대 변위를 포함합니다.

시간대 변위는 로컬시간과 UTC(Coordinated Universal Time—이전의 그리니치 표준시)의 차이(시간과 분)입니다. 구문은 `TIMESTAMP[(precision)] WITH LOCAL TIME`이며 여기서 선택적 파라미터 `precision`은 초 필드의 소수 부분 자릿수를 지정합니다.

자릿수를 지정할 때 기호 상수 또는 변수는 사용할 수 없으며 0-9 범위의 정수 리터럴을 사용해야 합니다. 기본값은 6입니다.

이 데이터 유형은 데이터베이스 열에 값을 삽입하면 해당 값이 데이터베이스 시간대로 정규화되고 시간대 변위가 열에 저장되지 않는다는 점에서 `TIMESTAMP WITH TIME ZONE` 과 다릅니다. 값을 검색할 때 Oracle 서버는 로컬 세션 시간대의 값을 반환합니다.

# 15. PL/SQL 변수

## - 주요 SCALAR 변수의 데이터 타입 - 계속

### • INTERVAL YEAR TO MONTH

이 타입은 INTERVAL YEAR TO MONTH 데이터 유형은 연도와 월의 간격을 저장하거나 조작하는 데 사용됩니다. 구문은 INTERVAL YEAR[(precision)] TO MONTH이며 여기서 precision은 연도 필드의 자릿수를 지정합니다. 자릿수를 지정할 때 기호 상수 또는 변수는 사용할 수 없으며 0-4 범위의 정수 리터럴을 사용해야 합니다. 기본값은 2입니다.

### • INTERVAL DAY TO SECOND

이 타입은 일, 시, 분, 초의 간격을 저장하거나 조작하는 데 사용됩니다. 구문은 INTERVAL DAY[(precision1)] TO SECOND[(precision2)]이며 여기서 precision1 및 precision2는 각각 일 필드와 초 필드의 자릿수를 지정합니다. 두 경우 모두 자릿수를 지정할 때 기호 상수 또는 변수는 사용할 수 없으며 0-9 범위의 정수 리터럴을 사용해야 합니다. 기본값은 각각 2와 6입니다.

# 15. PL/SQL 변수

## 2) Reference 변수 (참조 변수)

- Vno emp.empno%TYPE <- emp 테이블의 empno와 동일한 데이터형으로 선언함
- Vname emp.ename%TYPE <- emp 테이블의 ename 과 동일한 데이터형으로 선언함.
- Vrow emp%ROWTYPE <- emp 테이블의 여러 컬럼을 한꺼번에 저장할 변수로 선언함

# 15. PL/SQL 변수

## (1) TYPE 변수를 사용하여 데이터 조회하기

- emp3 테이블에서  
empno가 7900 번인 사원  
의 empno, ename, sal 을  
조회하여 화면에 출력하세  
요.

실습용 테이블 생성

```
SCOTT>CREATE TABLE emp3
2 AS
3 SELECT empno, ename ,
4        sal
5 FROM emp ;
```

```
SCOTT>SET SERVEROUTPUT ON;
SCOTT>DECLARE
```

```
2 vno      emp3.empno%TYPE ;
3 vname    emp3.ename%TYPE ;
4 vsal     emp3.sal%TYPE ;
```

```
5
6 BEGIN
7   SELECT empno, ename, sal
8   INTO vno, vname, vsal
9   FROM emp3
10  WHERE empno=7900 ;
11
12 DBMS_OUTPUT.PUT_LINE(vno||'  '||vname||'  '||vsal) ;
13
14 END ;
15 /
7900  JAMES  950
```

PL/SQL procedure successfully completed.

# 15. PL/SQL 변수

## (2) ROWTYPE 변수를 활용하여 데이터 출력하기

앞의 예 1번에서 출력했던 내용을 ROWTYPE 변수를 사용하여 출력하세요.

```
SCOTT>SET SERVEROUTPUT ON;
SCOTT>DECLARE
2  v row emp3%ROWTYPE ;
3
4 BEGIN
5  SELECT * INTO v_row
6  FROM emp3
7  WHERE empno=7900 ;
8
9  DBMS_OUTPUT.PUT_LINE(v_row.empno||'**'||v_row.ename||'**'||v_row.sal);
10 END ;
11 /
7900**JAMES**950
```

PL/SQL procedure successfully completed.

# 15. PL/SQL 변수

## (3) ROWTYPE 변수를 활용한 데이터의 입력

### 실습용 테이블 생성하기

```
SCOTT>CREATE TABLE row_test
2 ( no NUMBER,
3   name VARCHAR2(10),
4   hdate DATE );
```

Table created.

```
SCOTT>CREATE TABLE row_test2
2 AS SELECT * FROM row_test;
```

Table created.

```
SCOTT>INSERT INTO row_test
2 VALUES (1,'AAA',SYSDATE);
```

1 row created.

```
SCOTT>INSERT INTO row_test
2 VALUES (2,'BBB',SYSDATE);
```

1 row created.

```
SCOTT>INSERT INTO row_test
2 VALUES (3,'CCC',SYSDATE);
```

1 row created.

```
SCOTT>COMMIT ;
```

# 15. PL/SQL 변수

ROWTYPE 변수를  
활용한 데이터 입력

```
SCOTT> DECLARE
  2  v_record row_test%ROWTYPE ;
  3  BEGIN
  4  SELECT * INTO v_record
  5  FROM row_test
  6  WHERE no=1 ;
  7
  8  INSERT INTO row_test2
  9  VALUES v_record ;
 10  END;
 11  /
```

PL/SQL procedure successfully  
completed.

```
SCOTT> SELECT * FROM row_test2;
```

NO	NAME	HDATE
1	AAA	27-MAR-12

# 15. PL/SQL 변수

## (4) ROWTYPE 변수를 활용한 데이터의 변경

```
SCOTT>DECLARE
  2  v_record row_test%ROWTYPE;

3  BEGIN
4    SELECT * INTO v_record
5    FROM row_test
6    WHERE no=1 ;
7
8  v_record.name := 'DDD' ;
9
10   UPDATE row_test2
11   SET row=v_record
12   WHERE no=1;
13 END ;
14 /
```

PL/SQL procedure successfully completed.

```
SCOTT>SELECT * FROM row_test2;
```

NO	NAME	HDATE
1	DDD	27-MAR-12

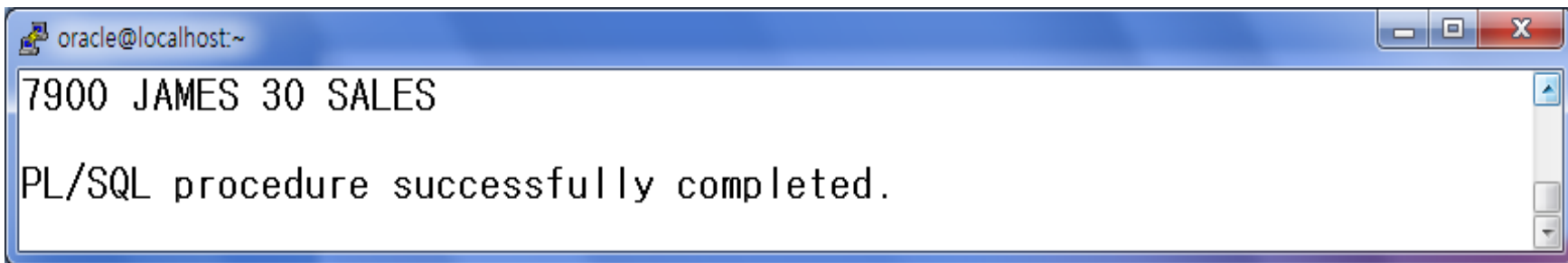
원래 'AAA' 였던 값이  
'DDD' 로 변경되었습니다.



# 15. PL/SQL 변수

## 예제 1.

%TYPE 변수를 사용하여 emp , dept 테이블을 조인하여 empno=7900 인 사람의 정보를 4개의 변수에 넣은 후 empno , ename , deptno, dname 을 아래와 같이 나오도록 출력하세요



```
oracle@localhost:~
7900 JAMES 30 SALES

PL/SQL procedure successfully completed.
```

```
SCOTT>SET SERVEROUTPUT ON ;
SCOTT>DECLARE
  2  v_empno emp.empno%TYPE ;
  3  v_ename emp.ename%TYPE ;
  4  v_deptno dept.deptno%TYPE ;
  5  v_dname  dept.dname%TYPE ;
  6
```

다음 장에 계속...

# 15. PL/SQL 변수

앞장에서 계속.....

```
7 BEGIN
8  SELECT e.empno, e.ename, d.deptno, d.dname
9  INTO v_empno, v_ename, v_deptno, v_dname
10 FROM emp e, dept d
11 WHERE e.empno=7900
12 AND e.deptno=d.deptno ;
13
14 DBMS_OUTPUT.PUT_LINE(v_empno||'  '||v_ename||'  '||v_deptno||'  '||v_dname) ;
15 END ;
16 /
7900  JAMES   30   SALES
```

PL/SQL procedure successfully completed.

# 15. PL/SQL 변수

**예제 2. 사용자로부터 두 개의 숫자를 입력 받아서 합을 구하세요.**

```
SCOTT>SET VERIFY OFF
SCOTT>SET SERVEROUTPUT ON
SCOTT>DECLARE
2  v_no1 NUMBER := &no1 ;
3  v_no2 NUMBER := &no2 ;
4  v_sum NUMBER ;
5
6 BEGIN
7  v_sum := v_no1 + v_no2 ;
8  DBMS_OUTPUT.PUT_LINE('첫번째 수: '||v_no1||', 두번째 수 : '||v_no2||' , 합은 : '||v_sum||' 입니다');
9 END ;
10 /
```

Enter value for no1: 20

Enter value for no2: 40

첫번째 수: 20, 두번째 수 : 40 , 합은 : 60 입니다

PL/SQL procedure successfully completed.

# 15. PL/SQL 변수

## 2) 복합 변수 (조합 변수라고도 합니다)

레코드 Type

profno	name	Birthday
number	Varchar2(10)	date
....	....	....

컬렉션 Type

integer	name
1	Varchar2(10)
2	

# 15. PL/SQL 변수

## (1) PL/SQL RECORD Type 변수

- ① TYPE *type\_name* IS RECORD  
( *field\_declaration* [, *field\_declaration*] ... ) ;
- ② Identifier *type\_name*

```
SQL> set serveroutput on
SQL> declare
2  type emp_record_type is record
3  ( emp_no emp.empno%type,
4    emp_name emp.ename%type,
5    job emp.job%type);
6
7  v_rec1 emp_record_type;
8
9  begin
10  select empno,ename,job
11  into v_rec1
12  from emp
13  where empno=7499;
14
15  dbms_output.put_line('사원번호   사원명   업무');
16  dbms_output.put_line(v_rec1.emp_no||'   ||
17                        v_rec1.emp_name||'   ||
18                        v_rec1.job);
19 end;
20 /
사원번호   사원명   업무
7499      ALLEN   SALESMAN

PL/SQL procedure successfully completed.
```

# 15. PL/SQL 변수

## - Record Type 변수 사용 예 1:

Record type 변수를 활용하여 부서번호가 30번인 부서의 부서번호와 부서명과 지역명을 Record type 변수에 저장한 후 출력하세요. 단 record type 데이터 타입은 dept\_record\_type 로 하겠습니다.

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2  TYPE dept_record_type IS RECORD
  3  ( deptno dept.deptno%TYPE,
  4    dname  dept.dname%TYPE,
  5    loc    dept.loc%TYPE);
  6
  7  v_dept dept_record_type;
  8
  9  BEGIN
 10  SELECT deptno , dname , loc
 11  INTO v_dept
 12  FROM dept
 13  WHERE deptno=30;
```

다음 장에 계속.....

# 15. PL/SQL 변수

앞 장에서 계속.....

```

14
15 DBMS_OUTPUT.PUT_LINE('부서번호    부서명    위치');
16 DBMS_OUTPUT.PUT_LINE(v_dept.deptno||'    ||
17                        v_dept.dname||'    ||
18                        v_dept.loc);
19 END ;
20 /

```

```

부서번호    부서명    위치
30          SALES    CHICAGO
PL/SQL procedure successfully completed.

```

# 15. PL/SQL 변수

## - Record Type 변수 사용 예 2:

emp2 테이블을 사용하여 사용자로부터 사원 번호를 입력 받은 후 아래와 같이 사원번호, 사원이름, 직급, 생일, 연락처, 급여를 출력하세요  
단 직급이 없는 사원은 직급을 사원으로 표시해서 출력하세요

Enter value for empno: 20000102

사원번호: 20000102

사 원 명: 김설악

직    급: 사원

생    일: 22-MAR-83

연 락 처: 031)234-5678

급    여: 30000000



# 15. PL/SQL 변수

```
SCOTT> DECLARE
2  TYPE e2_rec_type IS RECORD (
3      vempno    emp2.empno%TYPE,
4      vname     emp2.name%TYPE,
5      vposition emp2.position%TYPE,
6      vbirth    emp2.birthday%TYPE,
7      vtel      emp2.tel%TYPE,
8      vpay      emp2.pay%TYPE );
9
10 v_e2_record  e2_rec_type ;
11
12 v_empno emp2.empno%TYPE := '&empno';
```

다음 장에 계속....

# 15. PL/SQL 변수

앞 장에서 계속....

```

13 BEGIN
14     SELECT empno,name,NVL(position,'사원'),birthday,tel,pay
15     INTO v_e2_record
16     FROM emp2
17     WHERE empno=v_empno ;
18
19     DBMS_OUTPUT.PUT_LINE('사원번호: '||v_e2_record.vempno);
20     DBMS_OUTPUT.PUT_LINE('사   원   명: '||v_e2_record.vname);
21     DBMS_OUTPUT.PUT_LINE('직       급: '||v_e2_record.vposition);
22     DBMS_OUTPUT.PUT_LINE('생       일: '||v_e2_record.vbirth);
23     DBMS_OUTPUT.PUT_LINE('연   락   처: '||v_e2_record.vtel);
24     DBMS_OUTPUT.PUT_LINE('급       여: '||v_e2_record.vpay);
25 END;
26 /
    
```

# 15. PL/SQL 변수

## (2) PL/SQL Table Type 변수 (컬렉션이라고도 부릅니다)

- \* 연관 배열
- \* 중첩 테이블
- \* VARRAY

### \* 연관 배열(INDEX BY Table)

Key	Name
1	AAA
2	BBB
3	CCC
4	DDD

- **(Unique) Key** 열: 이 열에 들어가는 데이터 유형은 아래 두 가지입니다.

• **숫자일 경우** - BINARY\_INTEGER 또는 PLS\_INTEGER  
이 두 가지 숫자 데이터 유형은 NUMBER보다 적은 저장 영역이 필요하며 해당 데이터 유형에 대한 산술 연산은 NUMBER 산술보다 빠릅니다.

• **문자일 경우** - VARCHAR2 또는 하위 유형 중 하나

- **값(value)** 열: Value열은 실제 값이 들어가는 곳으로 입력되는 데이터의 종류에 따라 스칼라 데이터 유형 또는 레코드 데이터 유형일 수 있습니다. 스칼라 데이터 유형의 열은 해당 하나의 값만 보유할 수 있지만, 레코드 데이터 유형의 열은 해당 여러 값을 보유할 수 있습니다

# 15. PL/SQL 변수

## - 연관 배열의 주요 특징

- 연관 배열은 변수 선언 당시 채워지지 않으며 키나 값을 포함하지 않으므로 선언에서 연관 배열을 초기화할 수 없습니다.
- 연관 배열을 채우려면 명시적 실행 문이 필요합니다.
- 데이터베이스 테이블의 크기와 마찬가지로 연관 배열의 크기에도 제약이 없습니다. 따라서 새 행이 추가됨에 따라 연관 배열이 증가하도록 행 수가 동적으로 늘어날 수 있습니다. 키는 순차적이 아닐 수 있으며 양수 및 음수일 수 있습니다.

# 15. PL/SQL 변수

## - PL/SQL Table (컬렉션 타입) 정의와 선언

- ① TYPE *type\_name* IS TABLE OF  
{column\_type|variable%type|table.column%type} [NOT NULL] table%ROWTYPE  
[INDEX BY BINARY\_INTEGER] ;
- ② Identifier *type\_name*

### ① 정의부분 :

*type\_name* 은 PL/SQL Table 유형의 이름으로 일반적인 프로그래밍 언어에서의 배열과 비슷한 의미입니다. 위 Record Type과 다른 부분은 Record Type은 다른 유형의 데이터 타입을 사용하지만 이 Table Type 형은 동일한 유형의 데이터(또는 데이터 구조)들을 하나의 연속적인 메모리 공간에 확보하기 위해 사용한다는 점입니다.

*INDEX BY* 절은 그 배열내의 요소(element)에 접근하기 위한 첨자(위치) 값으로 사용되며, 범위는 BINARY\_INTEGER 의 범위(-2,147,483,647 - 2,147,483,647 사이의 정수) 에 속합니다.

### ② 선언부분 :

기본적으로 복합형의 데이터는 우선 정의를 하고 해당 정의를 통해 실제 복합 변수를 선언하는 단계로 구성됩니다. 위의 Record Type 형태와 사용방법은 동일하며 이 선언부분에서 실제 복합 변수에 대한 기억공간이 확보가 됩니다. (정의 부분에서는 메모리에 공간이 확보되지는 않으며 단지 복합 데이터 형에 대한 기술이 이루어지는 부분입니다.)

# 15. PL/SQL 변수

## - Table Type 변수 사용 예 1:

Table Type 변수를 사용하여 사원번호가 7499 인 사원의 이름을 조회해서 해당 변수에 저장 한 후 출력해보세요. 단 Table Type 변수 이름은 **tbl\_emp\_name** 으로 하세요.

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2  t_name VARCHAR2(20);
  3
  4  TYPE tbl_emp_name IS TABLE OF
  5  emp.ename%TYPE
  6  INDEX BY BINARY_INTEGER;
  7
  8  v_name tbl_emp_name;
  9
```

```
10 BEGIN
11  SELECT ename INTO t_name
12  FROM emp
13  WHERE empno=7499;
14
15  v_name(0) := t_name;
16  DBMS_OUTPUT.PUT_LINE(v_name(0));
17 END ;
18 /
;
```

# 15. PL/SQL 변수

- Table Type 변수 사용 예 2:

For 반복 문을 사용하여 변수에 여러 건의 데이터를 입력하는 방법입니다

```
SQL> DECLARE
2
3  TYPE e_table_type IS TABLE OF
4    emp.ename%TYPE
5  INDEX BY BINARY_INTEGER ;
6  tab_type e_table_type;
7  a BINARY_INTEGER := 0;
8 BEGIN
9  FOR i IN (SELECT ename FROM emp) LOOP
10    a := a+1;
11    tab_type(a) := i.ename;
12  END LOOP ;
13  FOR j IN 1..a LOOP
14    DBMS_OUTPUT.PUT_LINE (tab_type(j));
15  END LOOP ;
16 END ;
17 /
```

- 출력 결과 화면 -

```
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER
```

# 15. PL/SQL 변수

## 3. 비 PL/SQL 변수

### - 바인드 변수

- 바인드 변수는 호스트 환경에서 생성되어 데이터를 저장하므로 호스트 변수라고도 합니다.
- **VARIABLE** 키워드를 사용하여 생성되며 **SQL** 문과 **PL/SQL** 블록에서 사용됩니다.
- **PL/SQL** 블록이 실행된 후에도 액세스할 수 있습니다.
- 앞에 콜론을 사용하여 참조하며 **PRINT** 명령을 사용하여 값을 출력할 수 있습니다. 단 치환변수와는 구분을 해야 합니다. 치환 변수는 사용자에게 어떤 값을 입력 받아서 치환하며 접두 문자로 & (앰퍼샌트)를 사용합니다.



# 15. PL/SQL 변수

```
SCOTT> BEGIN
2   SELECT (pay*12)+NVL(bonus,0) INTO :v_bind
3   FROM professor
4   WHERE profno=1001 ;
5 END;
6 /
```

PL/SQL procedure successfully completed.

SCOTT>PRINT v\_bind ; -- 바인드 변수에 담긴 값을 출력합니다.

```
V_BIND
-----
6700
```

# 15. PL/SQL 변수

```
SCOTT> SET AUTOPRINT ON ;
```

```
SCOTT> BEGIN
```

```
2   SELECT (pay*12)+NVL(bonus,0) INTO :v_bind
3   FROM professor
4   WHERE profno=1001 ;
5 END;
6 /
```

PL/SQL procedure successfully completed.

V\_BIND

-----  
6700