Advanced Product Service

Oracle Database 11g : PL/SQL Fundamentals

Creation Date	**:**	**2013년 12월 09일**
Last Updated	**:**	**2013년 12월 11일**
Version	**:**	**1.0**

Chong Ha, Ryu

chongha.ryu@gmail.com
http://oukr.tistory.com

1. PL/SQL 소개

```
SQL> CREATE TABLE emp_sum
       AS SELECT deptno, SUM(sal) AS sum_sal
          FROM emp
          GROUP BY deptno ;
```

※ SQL 문으로 작업

```
SQL> SELECT * FROM emp_sum ;
    DEPTNO     SUM_SAL
---------- ----------
        30        9400
        20       10875
        10        8750

SQL> SELECT empno, ename, sal, deptno
       FROM emp
       WHERE empno = 7788 ;
     EMPNO ENAME                SAL     DEPTNO
---------- ---------- ---------- ----------
      7788 SCOTT              3000         20
SQL> DELETE emp
       WHERE empno = 7788 ;

SQL> UPDATE emp_sum
       SET sum_sal = sum_sal - 3000
       WHERE deptno = 20 ;

SQL> SELECT * FROM emp_sum ;
    DEPTNO     SUM_SAL
---------- ----------
        30        9400
        20        7875
        10        8750

SQL> ROLLBACK ;
```

※ PL/SQL 문으로 작업

```
SQL> SET SERVEROUTPUT ON
DECLARE
   emp_rec     emp%ROWTYPE ;
   sum_rec     emp_sum%ROWTYPE ;
BEGIN
   SELECT * INTO emp_rec
   FROM emp
   WHERE empno = 7788 ;

   DELETE emp
   WHERE empno = 7788 ;

   UPDATE emp_sum
   SET sum_sal = sum_sal - emp_rec.sal
   WHERE deptno = emp_rec.deptno ;

   SELECT * INTO sum_rec
   FROM emp_sum
   WHERE deptno = emp_rec.deptno ;

   DBMS_OUTPUT.PUT_LINE ( 'SUM Salary : ' || sum_rec.sum_sal ) ;

END ;
/
```

SUM Salary : 7875

```
PL/SQL procedure successfully completed.

SQL> ROLLBACK ;
```

※ Procedure 생성

```
CREATE OR REPLACE PROCEDURE delete_emp
( p_empno     NUMBER)   AS
    emp_rec     emp%ROWTYPE ;
    sum_rec     emp_sum%ROWTYPE ;
BEGIN
  SELECT * INTO emp_rec
  FROM emp
  WHERE empno = p_empno ;

  DELETE emp
  WHERE empno = p_empno ;

  UPDATE emp_sum
  SET sum_sal = sum_sal - emp_rec.sal
  WHERE deptno = emp_rec.deptno ;

  SELECT * INTO sum_rec
  FROM emp_sum
  WHERE deptno = emp_rec.deptno ;

  DBMS_OUTPUT.PUT_LINE ( 'SUM Salary : ' || sum_rec.sum_sal ) ;

END ;
/

Procedure created.

SQL> SET SERVEROUTPUT ON
SQL> EXECUTE delete_emp (7788)

SUM Salary : 7875
PL/SQL procedure successfully completed.

SQL> ROLLBACK ;
```

2. 실행문 작성 (변수 정의)

※ 변수 선언

```
SQL> SET SERVEROUTPUT ON
DECLARE
   v_hiredate       DATE ;
   v_deptno         NUMBER(2) NOT NULL    := 10 ;
   v_location       VARCHAR2(13)          := 'Atlanta';
   c_comm           CONSTANT NUMBER       := 1400 ;
BEGIN
   DBMS_OUTPUT.PUT_LINE ( v_hiredate ) ;
   DBMS_OUTPUT.PUT_LINE ( v_deptno ) ;
   DBMS_OUTPUT.PUT_LINE ( v_location ) ;
   DBMS_OUTPUT.PUT_LINE ( c_comm ) ;
END ;
/
10
Atlanta
1400
PL/SQL procedure successfully completed.
```

※ PL/SQL 에서 Date type의 주의 사항

```
SQL> ALTER SESSION SET nls_date_format = 'DD-MON-RR' ;
SQL> SET SERVEROUTPUT ON
DECLARE
   v_hiredate       DATE   :=   '09-DEC-13' ;
BEGIN
   DBMS_OUTPUT.PUT_LINE ( v_hiredate ) ;
END ;
/
09-DEC-13
PL/SQL procedure successfully completed.
```

```
SQL> ALTER SESSION SET nls_date_format = 'RR/MM/DD' ;
SQL> SET SERVEROUTPUT ON
DECLARE
  v_hiredate      DATE   :=   '09-DEC-13' ;
BEGIN
  DBMS_OUTPUT.PUT_LINE ( v_hiredate ) ;
END ;
/
```

09/12/13 << 올바른 값인가?

```
PL/SQL procedure successfully completed.

SQL> ALTER SESSION SET nls_date_format = 'RR/MM/DD' ;
SQL> SET SERVEROUTPUT ON
DECLARE
  v_hiredate      DATE   :=   TO_DATE('09-DEC-13','DD-MON-RR') ;
BEGIN
  DBMS_OUTPUT.PUT_LINE ( v_hiredate ) ;
END ;
/
```

13/12/09

```
PL/SQL procedure successfully completed.

SQL> ALTER SESSION SET nls_date_format = 'DD-MON-RR' ;
```

※ BINARY_FLOAT, BINARY_DOUBLE type 확인

```
SQL> SET SERVEROUTPUT ON
DECLARE
 bf_var            binary_float ;
 bd_var            binary_double ;

BEGIN
  bf_var := 270/35 ;
  bd_var := 140/0.35 ;
  DBMS_OUTPUT.PUT_LINE ('bf:   ' || bf_var );
  DBMS_OUTPUT.PUT_LINE ('bd:   ' || bd_var );
END ;
/
```

bf: 7.71428585E+000

bd: 4.0E+002

```
PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON
DECLARE
 bf_var            NUMBER ;
 bd_var            NUMBER ;

BEGIN
  bf_var := 270/35 ;
  bd_var := 140/0.35 ;
  DBMS_OUTPUT.PUT_LINE ('bf:   ' || bf_var );
  DBMS_OUTPUT.PUT_LINE ('bd:   ' || bd_var );
END ;
/
```

bf: 7.71428571428571428571428571428571428571

bd: 400

PL/SQL procedure successfully completed.

※ %TYPE 사용

```
SQL> SET SERVEROUTPUT ON
DECLARE
   v_sal           emp.sal%TYPE ;
BEGIN
   SELECT sal INTO v_sal
   FROM emp
   WHERE empno = 7788   ;

   DBMS_OUTPUT.PUT_LINE ( v_sal ) ;
END ;
/
```

3000

PL/SQL procedure successfully completed.

※ Bind Variable 사용

```
SQL> VARIABLE   b_sal   NUMBER
SQL> PRINT b_sal
      B_SAL
----------

BEGIN
    SELECT sal INTO :b_sal
    FROM emp
    WHERE empno = 7788   ;
END ;
/
PL/SQL procedure successfully completed.

SQL> PRINT b_sal
      B_SAL
----------
       3000
```

3. 실행문 작성

※ PL/SQL 에서의 함수 사용

```
SQL> SET SERVEROUTPUT ON
DECLARE
   v_desc_size              INTEGER(5);
   v_prod_description       VARCHAR2(70) := 'You can use this product with your radios for higher frequency';
BEGIN
   v_desc_size := LENGTH(v_prod_description) ;
   DBMS_OUTPUT.PUT_LINE (v_desc_size) ;
END ;
/
62
PL/SQL procedure successfully completed.

DECLARE
   v_desc_size              INTEGER(5);
   v_prod_description       VARCHAR2(70) := 'You can use this product with your radios for higher frequency';
BEGIN
   v_desc_size := MAX(v_prod_description) ;
   DBMS_OUTPUT.PUT_LINE (v_desc_size) ;
END ;
/
ERROR at line 5:
ORA-06550: line 5, column 18:
PLS-00204: function or pseudo-column 'MAX' may be used inside a SQL statement only
ORA-06550: line 5, column 3:
PL/SQL: Statement ignored

DECLARE
   v_sum           NUMBER ;
BEGIN
   SELECT SUM(sal) INTO v_sum
   FROM emp ;
   DBMS_OUTPUT.PUT_LINE (v_sum) ;
END ;
/
29025
PL/SQL procedure successfully completed.
```

※ PL/SQL 에서 SEQUENCE 사용

```
SQL> CREATE SEQUENCE empno_seq START WITH 1000 ;
SQL> SET SERVEROUTPUT ON
DECLARE
  v_num          NUMBER := empno_seq.NEXTVAL ;
BEGIN
  DBMS_OUTPUT.PUT_LINE ( v_num ) ;
END ;
/
```

1000

```
PL/SQL procedure successfully completed.

DECLARE
  v_num          NUMBER ;
BEGIN
  SELECT empno_seq.NEXTVAL INTO v_num
  FROM dual ;
  DBMS_OUTPUT.PUT_LINE ( v_num ) ;
END ;
/
```

1001

```
PL/SQL procedure successfully completed.
```

※ 중첩 블록에서 변수의 범위

```
SQL> SET SERVEROUTPUT ON
DECLARE
  v_outer          VARCHAR2(100) := 'Outer Variable' ;
  BEGIN
   DBMS_OUTPUT.PUT_LINE ('Outer : ' || v_outer ) ;
      DECLARE
        v_inner    VARCHAR2(100) := 'Inner Variable' ;
      BEGIN
        DBMS_OUTPUT.PUT_LINE ('Outer : ' || v_outer ) ;
        DBMS_OUTPUT.PUT_LINE ('Inner : ' || v_inner) ;
      END ;
  DBMS_OUTPUT.PUT_LINE ('Outer : ' || v_outer ) ;
END ;
/
```

Outer : Outer Variable

Outer : Outer Variable

Inner : Inner Variable

Outer : Outer Variable

```
PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON
DECLARE
  v_outer          VARCHAR2(100) := 'Outer Variable' ;
  BEGIN
   DECLARE
        v_inner    VARCHAR2(100) := 'Inner Variable' ;
      BEGIN
        DBMS_OUTPUT.PUT_LINE ('Outer : ' || v_outer ) ;
        DBMS_OUTPUT.PUT_LINE ('Inner : ' || v_inner) ;
      END ;
  DBMS_OUTPUT.PUT_LINE ('Outer : ' || v_outer ) ;
  DBMS_OUTPUT.PUT_LINE ('Inner : ' || v_inner) ;
END ;
/
ERROR at line 11:
ORA-06550: line 11, column 39:
PLS-00201: identifier 'V_INNER' must be declared
ORA-06550: line 11, column 3:
PL/SQL: Statement ignored

BEGIN <<outer>>
  DECLARE
    v_father_name        VARCHAR2(20):='Patrick';
    v_date_of_birth      DATE := TO_DATE('20-APR-1972','DD-MON-YYYY') ;
  BEGIN
    DECLARE
        v_child_name     VARCHAR2(20):='Mike';
        v_date_of_birth  DATE := TO_DATE('12-DEC-2002','DD-MON-YYYY') ;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
        DBMS_OUTPUT.PUT_LINE('Date of Birth: '||outer.v_date_of_birth);
        DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
        DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
    END;
  END;
END outer;
/
Father's Name: Patrick
Date of Birth: 20-APR-72
Child's Name: Mike
Date of Birth: 12-DEC-02
PL/SQL procedure successfully completed.
```

4. PL/SQL 프로그램에서 SQL 문과 상호 작용

※ DML(INSERT, UPDATE, DELETE) 문

```
BEGIN
  INSERT INTO emp(empno, ename, sal, deptno)
  VALUES (1234, 'RYU', 3000, 20) ;
END ;
/
PL/SQL procedure successfully completed.

BEGIN
  UPDATE emp
  SET sal = 4000
  WHERE empno = 1234 ;
END ;
/
PL/SQL procedure successfully completed.

BEGIN
  DELETE emp
  WHERE empno = 1234 ;
END ;
/
PL/SQL procedure successfully completed.

BEGIN
  UPDATE emp
  SET sal = 4000
  WHERE empno = 7788 ;

  UPDATE emp
  SET sal = 3500
  WHERE empno = 7566 ;
END ;
/
PL/SQL procedure successfully completed.
```

```
SQL> SELECT empno, sal FROM emp
       WHERE empno IN (7788, 7566) ;
     EMPNO        SAL
---------- ----------
      7566       3500
      7788       4000

SQL> ROLLBACK ;

SQL> SELECT empno, sal FROM emp
       WHERE empno IN (7788, 7566) ;
     EMPNO        SAL
---------- ----------
      7566       2975
      7788       3000

SQL> SET SERVEROUTPUT ON
BEGIN
  UPDATE emp
  SET sal = 4000
  WHERE empno = 7788 ;
  DBMS_OUTPUT.PUT_LINE ( SQL%ROWCOUNT || ' rows updated') ;

  DELETE emp
  WHERE deptno = 10 ;
  DBMS_OUTPUT.PUT_LINE ( SQL%ROWCOUNT || ' rows deleted') ;
END ;
/
1 rows updated
3 rows deleted
PL/SQL procedure successfully completed.

SQL> ROLLBACK ;
```

※ SELECT 문

```
SQL> SET SERVEROUTPUT ON
DECLARE
    v_ename       VARCHAR2(10) ;
    v_sal         emp.sal%TYPE ;
BEGIN
    SELECT ename, sal INTO v_ename, v_sal
    FROM emp
    WHERE empno = 7788 ;

    DBMS_OUTPUT.PUT_LINE ( v_ename || ' : ' || v_sal ) ;
END ;
/
```

SCOTT : 3000

```
PL/SQL procedure successfully completed.

SQL> SET SERVEROUTPUT ON
DECLARE
    v_ename       VARCHAR2(10) ;
    v_sal         emp.sal%TYPE ;
BEGIN
    SELECT ename, sal INTO v_ename, v_sal
    FROM emp
    WHERE deptno = 10 ;

    DBMS_OUTPUT.PUT_LINE ( v_ename || ' : ' || v_sal ) ;
END ;
/
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 5
```

```
SQL> SET SERVEROUTPUT ON
DECLARE
    v_ename        VARCHAR2(10) ;
    v_sal          emp.sal%TYPE ;
BEGIN
    SELECT ename, sal INTO v_ename, v_sal
    FROM emp
    WHERE empno = 1111 ;

    DBMS_OUTPUT.PUT_LINE ( v_ename || ' : ' || v_sal ) ;
END ;
/
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 5
```

※ DDL, DCL 문

```
BEGIN
   DROP TABLE emp ;
END ;
/
ERROR at line 2:
ORA-06550: line 2, column 3:
PLS-00103: Encountered the symbol "DROP" when expecting one of the following:
( begin case declare exit for goto if loop mod null pragma raise return select update while with <an
identifier> <a double-quoted delimited-identifier> <a bind variable> <<continue close current delete
fetch lock insert open rollback
savepoint set sql execute commit forall merge pipe purge
```

5. 제어 구조 작성

※ IF 문

```
SQL> SET SERVEROUTPUT ON
DECLARE
  v_myage        NUMBER := 10 ;
BEGIN
  IF v_myage < 11 THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  END IF;
END;
/
```

I am a child

PL/SQL procedure successfully completed.

```
DECLARE
  v_myage        NUMBER := 31 ;
BEGIN
  IF v_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.

```
DECLARE
  v_myage        NUMBER ;
BEGIN
  IF v_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.

```
DECLARE
  v_myage        NUMBER := 31 ;
BEGIN
  IF v_myage < 11 THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
    ELSIF v_myage < 20 THEN
      DBMS_OUTPUT.PUT_LINE(' I am young ');
    ELSIF v_myage < 30 THEN
      DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
    ELSIF v_myage < 40 THEN
      DBMS_OUTPUT.PUT_LINE(' I am in my thirties');
  ELSE
    DBMS_OUTPUT.PUT_LINE(' I am always young ');
  END IF;
END;
/
```

I am in my thirties

```
PL/SQL procedure successfully completed.
```

※ CASE 표현식

```
DECLARE
  v_grade        CHAR(1) := UPPER('&grade') ;
  v_appraisal    VARCHAR2(20) ;
BEGIN
  v_appraisal := CASE v_grade      WHEN 'A' THEN 'Excellent'
                                   WHEN 'B' THEN 'Very Good'
                                   WHEN 'C' THEN 'Good'
                 ELSE 'No such grade'
                 END;
  DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade || ' Appraisal ' || v_appraisal);
END;
/
```

Enter value for grade: B

```
old   2:   v_grade      CHAR(1) := UPPER('&grade') ;
new   2:   v_grade      CHAR(1) := UPPER('B') ;
```

Grade: B Appraisal Very Good

```
PL/SQL procedure successfully completed.
```

※ CASE 문

```
DECLARE
  v_sum          NUMBER ;
  v_deptno       NUMBER := &deptid ;
BEGIN
  CASE v_deptno
     WHEN 10 THEN
        SELECT SUM(sal) INTO v_sum
        FROM emp
        WHERE deptno = 10 ;
     WHEN 20 THEN
        SELECT SUM(sal) INTO v_sum
        FROM emp
        WHERE deptno = 20 ;
     WHEN 30 THEN
        SELECT SUM(sal) INTO v_sum
        FROM emp
        WHERE deptno = 30 ;
     ELSE
        SELECT SUM(sal) INTO v_sum FROM emp ;
     END CASE ;
  DBMS_OUTPUT.PUT_LINE ( v_sum ) ;
END ;
/
```

Enter value for deptid: 30

```
old   3:   v_deptno     NUMBER := &deptid ;
new   3:   v_deptno     NUMBER := 30 ;
```

9400

```
PL/SQL procedure successfully completed.
```

※ Loop 문

```
SQL> SET SERVEROUTPUT ON
DECLARE
  v_count          NUMBER(2) := 1 ;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE ('count: '||to_char(v_count)) ;
    v_count := v_count + 1 ;
    EXIT WHEN v_count = 4 ;
  END LOOP ;
END;
/
```

count: 1

count: 2

count: 3

```
PL/SQL procedure successfully completed.

DECLARE
      v_count     NUMBER(2) := 1 ;
BEGIN
  WHILE   v_count <= 3   LOOP
    DBMS_OUTPUT.PUT_LINE ('count: '||to_char(v_count)) ;
    v_count :=   v_count + 1 ;
  END   LOOP ;
END ;
/
```

count: 1

count: 2

count: 3

```
PL/SQL procedure successfully completed.

BEGIN
  FOR i IN 1..3 LOOP
    DBMS_OUTPUT.PUT_LINE ('count: '||to_char(i)) ;
  END LOOP ;
END ;
/
```

count: 1

count: 2

count: 3

```
PL/SQL procedure successfully completed.
```

```
BEGIN
  FOR i IN REVERSE 1..3 LOOP
    DBMS_OUTPUT.PUT_LINE ('count: '||to_char(i)) ;
  END LOOP ;
END ;
/
```

count: 3

count: 2

count: 1

PL/SQL procedure successfully completed.

※ Nested Loops

```
DECLARE
  x       NUMBER := 3 ;
  y       NUMBER ;
BEGIN
  <<OUTER_LOOP>>
  LOOP
    y := 1 ;
    EXIT WHEN x > 5 ;
    <<INNER_LOOP>>
    LOOP
      DBMS_OUTPUT.PUT_LINE ( x || ' * ' || y || ' = ' || x * y ) ;
    --   EXIT OUTER_LOOP WHEN x*y > 15 ;
      y := y + 1 ;
      EXIT WHEN y > 5 ;
    END LOOP INNER_LOOP ;
    x := x + 1 ;
  END LOOP OUTER_LOOP ;
END ;
/
```

3 * 1 = 3

3 * 2 = 6

3 * 3 = 9

3 * 4 = 12

3 * 5 = 15

4 * 1 = 4

4 * 2 = 8

4 * 3 = 12

4 * 4 = 16

4 * 5 = 20

5 * 1 = 5

5 * 2 = 10

5 * 3 = 15

5 * 4 = 20

5 * 5 = 25

PL/SQL procedure successfully completed.

```
DECLARE
  x       NUMBER := 3 ;
  y       NUMBER ;
BEGIN
  <<OUTER_LOOP>>
  LOOP
    y := 1 ;
    EXIT WHEN x > 5 ;
    <<INNER_LOOP>>
    LOOP
      DBMS_OUTPUT.PUT_LINE ( x || ' * ' || y || ' = ' || x * y ) ;
      EXIT OUTER_LOOP WHEN x*y > 15 ;
      y := y + 1 ;
      EXIT WHEN y > 5 ;
    END LOOP INNER_LOOP ;
    x := x + 1 ;
  END LOOP OUTER_LOOP ;
END ;
/
```

3 * 1 = 3

3 * 2 = 6

3 * 3 = 9

3 * 4 = 12

3 * 5 = 15

4 * 1 = 4

4 * 2 = 8

4 * 3 = 12

4 * 4 = 16

PL/SQL procedure successfully completed.

※ CONTINUE 문

```
SQL> SET SERVEROUTPUT ON
DECLARE
  v_total          SIMPLE_INTEGER := 0;
BEGIN
  FOR i IN 1..5 LOOP
    v_total := v_total + i;
    DBMS_OUTPUT.PUT_LINE ('Total is: '|| v_total) ;

    CONTINUE WHEN i > 3 ;
    v_total := v_total + i;
    DBMS_OUTPUT.PUT_LINE ('Out of Loop Total is: '|| v_total);
  END LOOP;
END;
/
```

Total is: 1 **<= 0 + 1 (i)**
Out of Loop Total is: 2 **<= 1 + 1 (i)**
Total is: 4 **<= 2 + 2 (i)**
Out of Loop Total is: 6 **<= 4 + 2 (i)**
Total is: 9 **<= 6 + 3 (i)**
Out of Loop Total is: 12 **<= 9 + 3 (i)**
Total is: 16 **<= 12 + 4 (i)**
Total is: 21 **<= 16 + 5 (i)**

```
PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON
DECLARE
  v_total          NUMBER := 0;
BEGIN
  <<BeforeTopLoop>>
  FOR i IN 1..5 LOOP
    v_total := v_total + 1;
    DBMS_OUTPUT.PUT_LINE ('Outer Total is: ' || v_total) ;

    FOR j IN 1..5 LOOP
      CONTINUE BeforeTopLoop WHEN i + j > 5 ;
      v_total := v_total + 1;
      DBMS_OUTPUT.PUT_LINE ('Inner Total is: ' || v_total) ;
    END LOOP;
  END LOOP;
END ;
/
```

Outer Total is: 1 <= 0 + 1 (i=1)
Inner Total is: 2 <= 1 + 1 (i=1 , j=1)
Inner Total is: 3 <= 2 + 1 (i=1 , j=2)
Inner Total is: 4 <= 3 + 1 (i=1 , j=3)
Inner Total is: 5 <= 4 + 1 (i=1 , j=4)
Outer Total is: 6 <= 5 + 1 (i=2)
Inner Total is: 7 <= 6 + 1 (i=2 , j=1)
Inner Total is: 8 <= 7 + 1 (i=2 , j=2)
Inner Total is: 9 <= 8 + 1 (i=2 , j=3)
Outer Total is: 10 <= 9 + 1 (i=3)
Inner Total is: 11 <= 10 + 1 (i=3 , j=1)
Inner Total is: 12 <= 11 + 1 (i=3 , j=2)
Outer Total is: 13 <= 12 + 1 (i=4)
Inner Total is: 14 <= 13 + 1 (i=4 , j=1)
Outer Total is: 15 <= 14 + 1 (i=5)

```
PL/SQL procedure successfully completed.
```

6. 조합 데이터 유형

※ PL/SQL Record

```
DECLARE
   TYPE emp_rec_typ IS RECORD
   ( ename          VARCHAR2(10),
     sal            emp.sal%TYPE,
     job            emp.job%TYPE := 'NONE' ) ;

   emp_rec          EMP_REC_TYP ;
BEGIN
   SELECT ename, sal, job INTO emp_rec
   FROM emp
   WHERE empno = 7788 ;
END ;
/
PL/SQL procedure successfully completed.
```

※ %ROWTYPE 사용

```
DECLARE
   emp_rec          emp%ROWTYPE ;
BEGIN
   SELECT * INTO emp_rec
   FROM emp
   WHERE empno = 7788 ;
END ;
/
PL/SQL procedure successfully completed.
```

※ Record Type 사용

```
SQL> CREATE TABLE copy_emp
        AS
        SELECT * FROM emp
        WHERE deptno = 10 ;

DECLARE
  emp_rec        emp%ROWTYPE ;
BEGIN
  SELECT * INTO emp_rec
  FROM emp
  WHERE empno = 7788 ;

  INSERT INTO copy_emp
  VALUES emp_rec ;

  SELECT * INTO emp_rec
  FROM emp
  WHERE empno = 7782 ;

  emp_rec.sal        := emp_rec.sal * 1.2 ;
  emp_rec.hiredate := SYSDATE ;

  UPDATE copy_emp
  SET ROW = emp_rec
  WHERE empno = 7782 ;
END ;
/
PL/SQL procedure successfully completed.

SQL> SELECT empno, ename, sal, hiredate, deptno
        FROM copy_emp ;
     EMPNO ENAME             SAL HIREDATE       DEPTNO
---------- ---------- ---------- --------- ----------
      7782 CLARK            2940 10-DEC-13         10
      7839 KING             5000 17-NOV-81         10
      7934 MILLER           1300 23-JAN-82         10
      7788 SCOTT            3000 19-APR-87         20

SQL> ROLLBACK ;
```

※ PL/SQL Collection (INDEX BY Table) 사용

```
SQL> SET SERVEROUTPUT ON
DECLARE
   TYPE tab_typ IS TABLE OF VARCHAR2(10)
   INDEX BY PLS_INTEGER ;

   tab     tab_typ ;
BEGIN
   tab(100) := 'AAA' ;
   tab(10)   := 'BBB' ;
   tab(50)   := 'CCC' ;
   tab(35)   := 'DDD' ;

   FOR i IN 1..tab.last LOOP
      IF tab.exists(i) THEN
         DBMS_OUTPUT.PUT_LINE ( i || ' : ' || tab(i) ) ;
      END IF ;
   END LOOP ;

END ;
/
```

10 : BBB

35 : DDD

50 : CCC

100 : AAA

PL/SQL procedure successfully completed.

```
DECLARE
  TYPE ename_tab_typ IS TABLE OF emp.ename%TYPE
  INDEX BY PLS_INTEGER ;

  ename_tab   ename_tab_typ ;
BEGIN
  SELECT ename BULK COLLECT INTO ename_tab
  FROM emp
  WHERE deptno = 10 ;

  FOR i IN ename_tab.first .. ename_tab.last LOOP
    IF ename_tab.exists(i) THEN
      DBMS_OUTPUT.PUT_LINE ( i || ' : ' || ename_tab(i) ) ;
    END IF ;
  END LOOP ;
END ;
/
```

1 : CLARK

2 : KING

3 : MILLER

```
PL/SQL procedure successfully completed.
```

※ PL/SQL Collection (Nested Table) 사용

```
SQL> SET SERVEROUTPUT ON
DECLARE
  TYPE tab_typ IS TABLE OF VARCHAR2(10) ;

  tab    tab_typ := tab_typ ('AAA','BBB','CCC') ;
BEGIN
  FOR i IN tab.first .. tab.last LOOP
    IF tab.exists(i) THEN
      DBMS_OUTPUT.PUT_LINE ( i || ' : ' || tab(i) ) ;
    END IF ;
  END LOOP ;

END ;
/
```

1 : AAA

2 : BBB

3 : CCC

```
PL/SQL procedure successfully completed.
```

※ PL/SQL Collection (VARRAY) 사용

```
DECLARE
   TYPE tab_typ IS VARRAY(3) OF VARCHAR2(10) ;

   tab     tab_typ := tab_typ ('AAA','BBB','CCC') ;
BEGIN
   FOR i IN tab.first .. tab.last LOOP
      IF tab.exists(i) THEN
         DBMS_OUTPUT.PUT_LINE ( i || ' : ' || tab(i) ) ;
      END IF ;
   END LOOP ;

END ;
/
```

1 : AAA

2 : BBB

3 : CCC

PL/SQL procedure successfully completed.

※ PL/SQL Collection (INDEX BY Record Table) 사용

```
DECLARE
   TYPE emp_tab_typ IS TABLE OF emp%ROWTYPE
   INDEX BY PLS_INTEGER ;

   emp_tab    emp_tab_typ ;
BEGIN
   SELECT * BULK COLLECT INTO emp_tab
   FROM emp
   WHERE deptno = 10 ;

   FOR i IN emp_tab.first .. emp_tab.last LOOP
      IF emp_tab.exists(i) THEN
         DBMS_OUTPUT.PUT_LINE ( i || ' : ' || emp_tab(i).ename || ' ' || emp_tab(i).sal ) ;
      END IF ;
   END LOOP ;
END ;
/
```

1 : CLARK 2450

2 : KING 5000

3 : MILLER 1300

PL/SQL procedure successfully completed.

7. 명시적 커서 사용

```
SQL> SET SERVEROUTPUT ON
DECLARE
  emp_rec        emp%ROWTYPE ;
BEGIN
  SELECT * INTO emp_rec
  FROM emp
  WHERE deptno = 10 ;

  DBMS_OUTPUT.PUT_LINE ( emp_rec.empno || ' ' || emp_rec.ename ) ;
END ;
/
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 4

DECLARE
  CURSOR emp_cur IS
    SELECT * FROM emp WHERE deptno = 10 ;
  emp_rec        emp%ROWTYPE ;
BEGIN
  OPEN emp_cur ;

  FETCH emp_cur INTO emp_rec ;
  DBMS_OUTPUT.PUT_LINE ( emp_rec.empno || ' ' || emp_rec.ename ) ;

  FETCH emp_cur INTO emp_rec ;
  DBMS_OUTPUT.PUT_LINE ( emp_rec.empno || ' ' || emp_rec.ename ) ;

  CLOSE emp_cur ;
END ;
/
7782 CLARK
7839 KING
PL/SQL procedure successfully completed.
```

```
DECLARE
  CURSOR emp_cur IS
    SELECT * FROM emp WHERE deptno = 10 ;
  emp_rec        emp%ROWTYPE ;
BEGIN
  OPEN emp_cur ;
  LOOP
    FETCH emp_cur INTO emp_rec ;
    EXIT WHEN emp_cur%NOTFOUND ;
    DBMS_OUTPUT.PUT_LINE ( emp_rec.empno || ' ' || emp_rec.ename ) ;
  END LOOP ;
  CLOSE emp_cur ;
END ;
/
```

7782 CLARK

7839 KING

7934 MILLER

```
PL/SQL procedure successfully completed.
```

```
DECLARE
  CURSOR emp_cur IS
    SELECT * FROM emp WHERE deptno = 10 ;
BEGIN
  FOR emp_rec IN emp_cur LOOP
    DBMS_OUTPUT.PUT_LINE ( emp_rec.empno || ' ' || emp_rec.ename ) ;
  END LOOP ;
END ;
/
```

7782 CLARK

7839 KING

7934 MILLER

```
PL/SQL procedure successfully completed.
```

```
BEGIN
  FOR emp_rec IN ( SELECT * FROM emp WHERE deptno = 10 ) LOOP
    DBMS_OUTPUT.PUT_LINE ( emp_rec.empno || ' ' || emp_rec.ename ) ;
  END LOOP ;
END ;
/
```

7782 CLARK

7839 KING

7934 MILLER

```
PL/SQL procedure successfully completed.
```

```
DECLARE
  CURSOR emp_cur ( p_deptno     NUMBER ) IS
    SELECT * FROM emp WHERE deptno = p_deptno ;
BEGIN
  FOR emp_rec IN emp_cur (10) LOOP
    DBMS_OUTPUT.PUT_LINE ( emp_rec.deptno || ' : ' || emp_rec.empno || ' ' || emp_rec.ename ) ;
  END LOOP ;

  FOR emp_rec IN emp_cur (20) LOOP
    DBMS_OUTPUT.PUT_LINE ( emp_rec.deptno || ' : ' || emp_rec.empno || ' ' || emp_rec.ename ) ;
  END LOOP ;
END ;
/
```

10 : 7782 CLARK
10 : 7839 KING
10 : 7934 MILLER
20 : 7369 SMITH
20 : 7566 JONES
20 : 7788 SCOTT
20 : 7876 ADAMS
20 : 7902 FORD

```
PL/SQL procedure successfully completed.
```

※ WHERE CURRENT OF 절 사용

```
SQL> CONN system/oracle
SQL> GRANT DBA TO ora1 ;
SQL> CONN ora1/oracle

SQL> SELECT session_id, owner, name, mode_held, blocking_others
       FROM dba_dml_locks ;
no rows selected

SQL> SELECT empno, ename, sal, deptno
       FROM emp
       WHERE deptno = 10 ;
...

SQL> SELECT session_id, owner, name, mode_held, blocking_others
       FROM dba_dml_locks ;
no rows selected
```

```
SQL> SELECT empno, ename, sal, deptno FROM emp
       WHERE deptno = 10 FOR UPDATE ;
...

SQL> SELECT session_id, owner, name, mode_held, blocking_others
       FROM dba_dml_locks ;
SESSION_ID OWNER           NAME            MODE_HELD     BLOCKING_OTHERS
---------- --------------- --------------- ------------- --------------------
        20 ORA1            EMP             Row-X (SX)    Not Blocking
SQL> ROLLBACK ;

SQL> SELECT empno, ename, sal FROM emp
       WHERE deptno = 10 ;
     EMPNO ENAME                SAL
---------- ---------- ----------
      7782 CLARK              2450
      7839 KING               5000
      7934 MILLER             1300

DECLARE
  CURSOR emp_cur IS
    SELECT * FROM emp
    WHERE deptno = 10 FOR UPDATE ;
BEGIN
  FOR emp_rec IN emp_cur LOOP
    IF emp_rec.sal < 2000 THEN
      UPDATE emp
      SET sal = sal * 1.1
      WHERE CURRENT OF emp_cur ;       /*   WHERE empno = emp_rec.empno */
    END IF ;
  END LOOP ;
END ;
/
PL/SQL procedure successfully completed.

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE deptno = 10 ;
     EMPNO ENAME                SAL
---------- ---------- ----------
      7782 CLARK              2450
      7839 KING               5000
      7934 MILLER             1430

SQL> ROLLBACK ;
```

8. 예외 처리

※ 예외 처리

```
SQL> ALTER TABLE emp
       ADD CONSTRAINT emp_ck CHECK ( sal > 0 ) ;

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE deptno = 10 ;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7782 CLARK            2450
      7839 KING             5000
      7934 MILLER           1300

BEGIN
  UPDATE emp
  SET sal = 3000
  WHERE empno = 7782 ;

  UPDATE emp
  SET sal = 0
  WHERE empno = 7934 ;
END ;
/
ERROR at line 1:
ORA-02290: check constraint (ORA1.EMP_CK) violated
ORA-06512: at line 6

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE deptno = 10 ;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7782 CLARK            2450
      7839 KING             5000
      7934 MILLER           1300
```

```
SQL> SET SERVEROUTPUT ON
BEGIN
  UPDATE emp
  SET sal = 3000
  WHERE empno = 7782 ;

  UPDATE emp
  SET sal = 0
  WHERE empno = 7934 ;

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ( SQLERRM ) ;
END ;
/
```

ORA-02290: check constraint (ORA1.EMP_CK) violated

```
PL/SQL procedure successfully completed.

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE deptno = 10 ;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7782 CLARK            3000
      7839 KING             5000
      7934 MILLER           1300

SQL> ROLLBACK ;

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE deptno = 10 ;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7782 CLARK            2450
      7839 KING             5000
      7934 MILLER           1300
```

※ 미리 정의된 예외 처리

```
SQL> SET SERVEROUTPUT ON
DECLARE
  emp_rec        emp%ROWTYPE ;
BEGIN
  SELECT * INTO emp_rec
  FROM emp
  WHERE ename = UPPER('&name') ;

  DBMS_OUTPUT.PUT_LINE ( emp_rec.sal ) ;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('NO_DATA_FOUND') ;

  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE ('TOO_MANY_ROWS') ;

  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('OTHERS') ;
END ;
/
Enter value for name: RYU
old   6:   WHERE ename = UPPER('&name') ;
new   6:   WHERE ename = UPPER('RYU') ;
NO_DATA_FOUND
PL/SQL procedure successfully completed.
```

※ 미리 정의되지 않은 예외 처리

```
SQL> SET SERVEROUTPUT ON
DECLARE
  emp_rec        emp%ROWTYPE ;
  e_ck           EXCEPTION ;
  PRAGMA EXCEPTION_INIT (e_ck , -2290) ;
BEGIN
  SELECT * INTO emp_rec
  FROM emp
  WHERE ename = UPPER('&name') ;

  IF emp_rec.sal < 2000 THEN
   UPDATE emp
   SET sal = &salary
   WHERE empno = emp_rec.empno ;
  END IF ;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('NO DATA') ;
  WHEN E_CK THEN
    DBMS_OUTPUT.PUT_LINE ('Invalid Salary') ;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE (SQLERRM) ;
END ;
/
Enter value for name: smith
old   8:   WHERE ename = UPPER('&name') ;
new   8:   WHERE ename = UPPER('smith') ;
Enter value for salary: 0
old  12:    SET sal = &salary
new  12:    SET sal = 0
Invalid Salary
PL/SQL procedure successfully completed.
```

※ 사용자 정의 예외 처리

```
DECLARE
   v_deptno                NUMBER := 50 ;
   v_name                  VARCHAR2(20) := 'Testing' ;
   e_invalid_department    EXCEPTION;
BEGIN
   UPDATE dept
   SET dname = v_name
   WHERE deptno = v_deptno ;

   IF SQL%NOTFOUND THEN
      RAISE e_invalid_department ;
   END IF;

   COMMIT;

EXCEPTION
   WHEN e_invalid_department THEN
      DBMS_OUTPUT.PUT_LINE('No such department id.');
END;
/
```

No such department id.

```
PL/SQL procedure successfully completed.
```

※ RASE_APPLICATION_ERROR 사용

```
BEGIN
   UPDATE dept
   SET dname = 'Testing'
   WHERE deptno = 50 ;

   IF SQL%NOTFOUND THEN
      RAISE_APPLICATION_ERROR ( -20001, 'No such department id.' ) ;
   END IF;
END;
/
ERROR at line 1:
```

ORA-20001: No such department id.

```
ORA-06512: at line 7
```

```
BEGIN
  UPDATE dept
  SET dname = 'Testing'
  WHERE deptno = 50 ;

  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR ( -20001, 'No such department id.' ) ;
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM) ;
END;
/
```

ORA-20001: No such department id.

```
PL/SQL procedure successfully completed.

DECLARE
  emp_rec        emp%ROWTYPE ;
BEGIN
  SELECT * INTO emp_rec
  FROM emp
  WHERE deptno = 10 ;
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR ( -20001, 'Too Many Rows', TRUE ) ;
END ;
/
ERROR at line 1:
```

ORA-20001: Too Many Rows

```
ORA-06512: at line 9
```

ORA-01422: exact fetch returns more than requested number of rows

※ 예외 전달

```
BEGIN
  UPDATE emp
  SET sal = 7777
  WHERE empno = 7788 ;

  BEGIN
    UPDATE emp
    SET sal = 9999
    WHERE empno = 7566 ;

    UPDATE emp
    SET sal = 0
    WHERE empno = 7839 ;

    UPDATE emp
    SET sal = 9999
    WHERE empno = 7499 ;
  END ;

  UPDATE emp
  SET sal = 7777
  WHERE empno = 7369 ;
END ;
/
ERROR at line 1:
ORA-02290: check constraint (ORA1.EMP_CK) violated
ORA-06512: at line 11

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE empno IN (7788, 7566, 7839, 7369, 7499) ;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7566 JONES            2975
      7788 SCOTT            3000
      7839 KING             5000
```

```
SQL> SET SERVEROUTPUT ON
BEGIN
  UPDATE emp
  SET sal = 7777
  WHERE empno = 7788 ;

  BEGIN
    UPDATE emp
    SET sal = 9999
    WHERE empno = 7566 ;

    UPDATE emp
    SET sal = 0
    WHERE empno = 7839 ;

    UPDATE emp
    SET sal = 9999
    WHERE empno = 7499 ;

  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE (SQLERRM) ;
  END ;

  UPDATE emp
  SET sal = 7777
  WHERE empno = 7369 ;
END ;
/
ORA-02290: check constraint (ORA1.EMP_CK) violated
PL/SQL procedure successfully completed.

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE empno IN (7788, 7566, 7839, 7369,7499) ;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH            7777
      7499 ALLEN            1600
      7566 JONES            9999
      7788 SCOTT            7777
      7839 KING             5000

SQL> ROLLBACK ;
```

```
SQL> SET SERVEROUTPUT ON
BEGIN
  UPDATE emp
  SET sal = 7777
  WHERE empno = 7788 ;

  BEGIN
    UPDATE emp
    SET sal = 9999
    WHERE empno = 7566 ;

    UPDATE emp
    SET sal = 0
    WHERE empno = 7839 ;

    UPDATE emp
    SET sal = 9999
    WHERE empno = 7499 ;
  END ;

  UPDATE emp
  SET sal = 7777
  WHERE empno = 7369 ;

EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE (SQLERRM) ;
END ;
/
ORA-02290: check constraint (ORA1.EMP_CK) violated
PL/SQL procedure successfully completed.

SQL> SELECT empno, ename, sal
       FROM emp
       WHERE empno IN (7788, 7566, 7839, 7369,7499) ;
EMPNO ENAME             SAL
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7566 JONES            9999
      7788 SCOTT            7777
      7839 KING             5000

SQL> ROLLBACK ;
```