1. **PACKAGE ?**

   1) **PACKAGE** 의 사전적인 의미는 꾸러미이다.

   2) 관련 있는 프로시저와 함수를 효율적으로 관리하기 위하여 패키지 단위로 배포할 때 유용하게 사용된다.

   3) 특정 처리를 위해 관련된 PL/SQL 블록들이 논리적으로 하나의 그룹을 이루는 특수한 형태

   4) 예를 들어, 사원관리 업무 중에 입사관리, 연봉관리, 상여금관리, 근태관리, 퇴사자 관리등의 업무가 많이 있을 경우 사원관리 라는 패키지를 생성하고,그 세부에 각각작업을 수행하는 함수나 프로시저를 생성해서 관리하면 훨씬 업무가 간결하고 편해지는 효과를 볼 수 있을 것이다.

   5) 패키지는 선언부(Specification)와 몸체부(**body**)로 구성된다.

   6) 패키지 선언부의 역할은 해당 패키지에 사용될 함수나 프로시저, 변수 등에 대한 정의를 선언하는 부분이다.

   7) 패키지 몸체부에서는 선언부에서 선언된 함수나 프로지서등이 실제 구현되는부분이다.

   8) 하지만 패키지 선언부에서 선언되지 않더라도 패키지 몸체부에서 사용될 수는 있지만 별로 권장사항은 아니다.

   9) 생성된 패키지의 구성 요소(멤버)에 접근할 때에는 패키지명을 접두어로 사용하면 된다.

   10)만약 생성된 패키지의 선언부가 변경되었다면 무조건 패키지 몸체부는 다시 재 생성해야 하며, 패키지를 참조(호출)하는 서브 프로그램들도 재 번역(recompile)해야 한다.

   11)반대로 패키지 몸체부만 변경되는 경우라면 패키지 선언부와 다른 관련 서브프로그램에 영향을 주지 않고 몸체부만 재 생성하면 된다.

   12) 몸체부에 정의한 프로시저나 함수는 이제까지 학습한 저장 프로시저와 저장 함수와 동일한 문법 구조를 갖는다.

2. Syntax

```
--명세부(선언부, Specification)
CREATE [OR REPLACE] PACKAGE package_name
IS | AS
   PUBLIC TYPE AND item declarations
   subprogram specifications
END;
/
```

   - **OR REPLACE :** 생성하고자 하는 패키지가 기존에 동일명으로 존재할 경우, 기존의 내용을현재의 내용으로 수정하는 옵션. 이 옵션은 해당 패키지를 삭제한 후 재 생성한다.

   - package_name **:** 생성하고자 하는 패키지명으로, 스키마 내에서는 유일한 이름이어야 한다. 패키지 선언부와 패키지 몸체부의 패키지명은 동일해야 한다.

   - **PUBLIC TYPE AND** item declarations **:** 변수, 상수, 명시적 커서, 사용자 정의 예외, **PRAGMA** 등을 선언한다. 이들은 모두 **PUBLIC**이란 특징을 가진다.

   - subprogram specifications **:** PL/**SQL** 서브 프로그램을 선언하는 부분. 선언할 때에는 형식 매개변수를 포함한 헤더만을 기술한다.

```
   PROCEDURE procedure_name1;
   PROCEDURE procedure_name2;
   FUNCTION funtion_name1;

--몸체부
CREATE [OR REPLACE] PACKAGE BODY package_name
IS | AS
   PUBLIC TYPE AND item declarations
   subprogram specifications
END;
/
```

   - subprogram specifications **:** 이 부분이 실제 작동할 서브프로그램(프로시저,함수 등)을 기록하는 부분이다. 단, 주의할 사항은 서브 프로그램의 순서이다. 기본적으로 참조되는 변수이든 서브프로그램이든 참조하는 서브프로그램보다는 먼저 정의돼야한다.

3. 패키지 실행

```
   - 패키지는 여러 환경에서 호출되어 실행될 수 있지만 생성된 패키지 오브젝트에 대한 실행 권한을
   가진 사용자만이 패키지를 호출하여 실행할 수 있다.
   EXECUTE [package_name].[procedure_name]

4. 패키지 삭제
   -패키지를 삭제할 때에는 패키지 선언부와 패키지 몸체부를 모두 삭제할 수도 있고 패키지 몸체부만
   삭제할 수도 있다.
   DROP PACKAGE package_name;
   DROP PACKAGE BODY package_name;

--------------------------------------------------------------------------------
   SET SERVEROUTPUT ON
   CREATE PACKAGE package_emp_sal
   AS
      PROCEDURE find_sal(v_empno IN emp.empno%type);
   END;
   /

   CREATE OR REPLACE PACKAGE BODY package_emp_sal
   AS
      PROCEDURE find_sal(v_empno IN emp.empno%TYPE)
      IS
         v_sal emp.sal%TYPE;
      BEGIN
         SELECT sal INTO v_sal
         FROM emp
         WHERE empno = v_empno;
         DBMS_OUTPUT.PUT_LINE('Salary: '|| v_sal);
      END;
   END;
   /

   DECLARE
      code customers.id%type := &cc_id;
   BEGIN
      cust_sal.find_sal(code);
   END;
   /

   SQL> @demo.sql
   Enter a Employee No. : 7788
   old   2:    t_empno  emp.empno%type := &p_empno;
   new   2:    t_empno  emp.empno%type := 7788;
   Salary: 3300

   PL/SQL procedure successfully completed.
-------------------------------------------------------------
   SET SERVEROUTPUT ON
   CREATE OR REPLACE PACKAGE package_emp
   AS
      -- Adds Employee
      PROCEDURE sp_emp_add
      (
         v_empno  IN  emp.empno%TYPE,
         v_ename  IN  emp.ename%TYPE,
         v_sal       IN  emp.sal%TYPE,
```

```
97          v_job      IN  emp.job%TYPE,
98          v_deptno IN  emp.deptno%TYPE
99       );
100
101      -- Removes Employee
102      PROCEDURE sp_emp_del(v_empno   IN  emp.empno%TYPE);
103
104      --Lists all Employee
105      PROCEDURE sp_emp_list;
106
107   END;
108   /
109
110   CREATE OR REPLACE PACKAGE BODY package_emp
111   AS
112      PROCEDURE sp_emp_add
113      (
114         v_empno IN  emp.empno%TYPE,
115         v_ename IN  emp.ename%TYPE,
116         v_sal IN  emp.sal%TYPE,
117         v_job IN  emp.job%TYPE,
118         v_deptno IN  emp.deptno%TYPE
119      )
120      IS
121      BEGIN
122         INSERT INTO emp(empno, ename, sal, job, hiredate, deptno)
123         VALUES (v_empno, v_ename, v_sal, v_job, SYSDATE, v_deptno);
124      END;
125
126      PROCEDURE sp_emp_del(v_empno   IN  emp.empno%TYPE)
127      IS
128      BEGIN
129         DELETE FROM emp
130         WHERE empno = v_empno;
131      END;
132
133      PROCEDURE sp_emp_list IS
134         CURSOR cursor_emp IS
135            SELECT empno, ename, deptno
136            FROM emp;
137         emp_record  emp%ROWTYPE;
138      BEGIN
139         FOR emp_record IN cursor_emp LOOP
140            DBMS_OUTPUT.PUT_LINE(emp_record.empno || ' ' || emp_record.
               ename || ' ' || emp_record.deptno);
141         END LOOP;
142      END;
143
144   END;
145   /
146
147   DECLARE
148   BEGIN
149      package_emp.sp_emp_add(7777, 'Sally', 1000, 'DEVELOPER', 10);
150      package_emp.sp_emp_add(8888, 'Michael', 1500, 'DESIGNER',20);
151      package_emp.sp_emp_list;
```

```
    package_emp.sp_emp_del(7788);
    package_emp.sp_emp_list;
END;
/

-------------------------------------------------------------
SET SERVEROUTPUT ON
CREATE OR REPLACE PACKAGE pack_sample
IS
   FUNCTION calc_bonus(v_empno IN emp.empno%TYPE)
      RETURN NUMBER;
   PROCEDURE cursor_sample;
END;
/

CREATE OR REPLACE PACKAGE BODY pack_sample
IS
   FUNCTION calc_bonus(v_empno IN emp.empno%TYPE)
      RETURN NUMBER
IS
   v_sal  NUMBER(7,2);
BEGIN
   SELECT sal INTO v_sal
   FROM emp
   WHERE empno = v_empno;

   RETURN v_sal * 200;
END;

PROCEDURE cursor_sample
IS
   v_dept_record dept%ROWTYPE;
   CURSOR c1
   IS
   SELECT * FROM dept;
BEGIN
   DBMS_OUTPUT.PUT_LINE('부서번호 | 부서명 | 위치');
   DBMS_OUTPUT.PUT_LINE('---------------------------');
   FOR v_dept_record IN c1 LOOP
      EXIT WHEN c1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_dept_record.deptno || ' | ' || v_dept_record.
      dname || ' | ' || v_dept_record.loc);
   END LOOP;
END;

END;
/

SQL> @demo.sql

Package created.


Package body created.

SQL> VAR answer NUMBER
```

```
SQL> EXECUTE :answer := pack_sample.calc_bonus(7788)

PL/SQL procedure successfully completed.

SQL> PRINT answer

    ANSWER
----------
    600000

SQL> EXEC pack_sample.cursor_sample
부서번호  |  부서명  |  위치
-----------------------------
10 | ACCOUNTING | NEW YORK
20 | RESEARCH | DALLAS
30 | SALES | CHICAGO
40 | OPERATIONS | BOSTON

PL/SQL procedure successfully completed.

-------------------------------------------------------------

SET SERVEROUTPUT ON
CREATE OR REPLACE PACKAGE emp_total
IS
   PROCEDURE emp_sum;
   PROCEDURE emp_avg;
END;
/

CREATE OR REPLACE PACKAGE BODY emp_total
IS
PROCEDURE emp_sum
IS
   CURSOR emp_total_sum  IS
      SELECT COUNT(*), SUM(sal)
      FROM emp;
   total_num NUMBER;
   total_sum NUMBER;
BEGIN
   OPEN emp_total_sum;
   FETCH emp_total_sum INTO total_num, total_sum;
   DBMS_OUTPUT.PUT_LINE('총 인원수 : ' || total_num || ', 급여합계 : ' ||
   total_sum);
   CLOSE emp_total_sum;
END;

PROCEDURE emp_avg
IS
   CURSOR emp_total_avg IS
      SELECT COUNT(*), AVG(sal)
      FROM emp;
   total_num NUMBER;
   total_avg NUMBER;
BEGIN
      OPEN emp_total_avg;
```

```
        FETCH emp_total_avg INTO total_num, total_avg;
        DBMS_OUTPUT.PUT_LINE('총인원수 : ' || total_num || ', 급여평균 : ' ||
        total_avg);
        CLOSE emp_total_avg;
    END;

    END;
    /

    SQL> EXEC emp_total.emp_sum;
    총 인원수 : 14, 급여합계 : 28141.5

    PL/SQL procedure successfully completed.

    SQL> exec emp_total.emp_avg;
    총인원수 : 14, 급여평균 : 2010.107142857142857142857142857142857143

    PL/SQL procedure successfully completed.

5. PACKAGE 조회하기
    --선언부 조회하기
    SELECT text FROM user_source
    WHERE TYPE = 'PACKAGE';

    --몸체부 조회하기
    SELECT text FROM user_source
    WHERE TYPE LIKE 'PACKAGE BODY';

-----------------------------------------------------------------
    SET SERVEROUTPUT ON
    CREATE OR REPLACE PACKAGE emp_comm
    IS
       g_comm    NUMBER := 10;
       PROCEDURE reset_comm(v_comm IN NUMBER);
    END;
    /

    CREATE OR REPLACE PACKAGE BODY emp_comm
    IS
    FUNCTION validate_comm(v_comm IN  NUMBER)
          RETURN BOOLEAN
    IS
       v_max_comm NUMBER;
    BEGIN
       SELECT MAX(comm) INTO v_max_comm
       FROM emp;
       IF v_comm > v_max_comm  THEN
          RETURN FALSE;
       ELSE
          RETURN TRUE;
       END IF;
    END;

    PROCEDURE reset_comm
    (v_comm IN NUMBER)
    IS
```

```
317       v_valid  BOOLEAN;
318     BEGIN
319           v_valid := validate_comm(v_comm);
320           IF v_valid = TRUE THEN
321              g_comm := v_comm;
322              DBMS_OUTPUT.PUT_LINE(g_comm);
323           ELSE
324              RAISE_APPLICATION_ERROR(-20210, 'Invalid Commission');
325           END IF;
326     END;
327
328     END;
329     /
330
331     SQL> EXEC emp_comm.reset_comm(100)
332     100
333
334     PL/SQL procedure successfully completed.
335
336     SQL> EXEC emp_comm.reset_comm(1500)
337     BEGIN emp_comm.reset_comm(1500); END;
338
339     *
340     ERROR at line 1:
341     ORA-20210: Invalid Commission
342     ORA-06512: at "SCOTT.EMP_COMM", line 27
343     ORA-06512: at line 1
344
345   ----------------------------------------------------------------------
346   REM Overloading
347   1. What ?
348      -하나의 PACKAGE 내에서 동일한 이름의 프로시저를 여러 개 만들 수 있는 기능.
349   2. 조건
350      1)반드시 매개변수의 갯수가 달라야 한다.
351      2)매개변수의 갯수가 같은 경우에는 변수의 데이터 타입이 달라야 한다.
352      3)매개변수의 갯수가 같은 경우에는 변수의 순서가 달라야 한다.
353      4)같은 이름을 가진 프로시저, 함수의 갯수에는 제한이 없다.
354
355     CREATE OR REPLACE PACKAGE emp_comm  IS
356        g_comm  NUMBER := 10;
357        PROCEDURE reset_comm(v_comm IN NUMBER);
358     END;
359     /
360
361     CREATE OR REPLACE PACKAGE BODY emp_comm IS
362     FUNCTION validate_comm(v_comm IN NUMBER)
363        RETURN BOOLEAN
364     IS
365        v_max_comm NUMBER;
366     BEGIN
367        SELECT MAX(comm) INTO v_max_comm
368        FROM emp;
369        IF v_comm > v_max_comm THEN
370           RETURN FALSE;
371        ELSE
372           RETURN TRUE;
```

```
373       END IF;
374    END;
375
376    PROCEDURE reset_comm(v_comm IN NUMBER)
377    IS
378       v_valid   BOOLEAN;
379    BEGIN
380       v_valid := validate_comm(v_comm);
381       IF v_valid = TRUE THEN
382          DBMS_OUTPUT.PUT_LINE(g_comm);
383       ELSE
384          RAISE_APPLICATION_ERROR(-20210, 'Invalid Commission');
385       END IF;
386    END;
387
388    PROCEDURE reset_comm(v_comm IN NUMBER, v_sal IN VARCHAR2)
389    IS
390    BEGIN
391       RAISE_APPLICATION_ERROR(-20210, 'Invalid Commission');
392    END;
393
394    END;
395    /
396
397    SQL> @demo.sql
398
399    Package created.
400
401
402    Package body created.
403
404
405 REM Forward Declaration
406    - 어떤 프로시저나 함수를 호출할 때 해당 프로시저보다 먼저 정의되어야 한다.
407 1. 전위적 프로시저와 함수
408    1)패키지의 특징은 관련된 많은 프로시저와 함수를 하나의 패키지로 모아서 생성할 수 있는 것이다.
409    2)또한, 프로시저와 함수들은 서로 호출되고, 호출하는 관계를 가지고 있다.
410    3)이런 경우, 어떤 프로시저 내에서 다른 프로시저를 호출할 때, 호출되는 프로시저는 호출하는
       프로시저보다 패키지 내에서 먼저 정의되어야 한다.
411    4)만약, 호출되는 프로시저보다 호출하는 프로시저의 위치가 선행된다면 실행 시 에러가 발생하게 된다.
412    5)이런 원칙을 전위적 선언(Forward Declaration)이라고 한다.
413
414 2. 위의 예에서는 PACKAGE 내의 두번째 프로시저인 reset_comm이 먼저 정의된
       validate_comm함수를 호출하기 때문에, 즉 아래에서 위의 함수를 호출하기 때문에 에러가 발생하지
       않는다. 하지만, 그 순서가 바뀌면 에러가 난다.
415
416    CREATE OR REPLACE PACKAGE emp_comm  IS
417       g_comm  NUMBER := 10;
418       PROCEDURE reset_comm(v_comm IN NUMBER);
419
420    END;
421    /
422
423    CREATE OR REPLACE PACKAGE BODY emp_comm IS
424
425    PROCEDURE reset_comm(v_comm IN NUMBER) --두번째 프로시저가 앞으로 옴
```

```
   IS
      v_valid   BOOLEAN;
   BEGIN
      v_valid := validate_comm(v_comm);
      IF v_valid = TRUE THEN
         DBMS_OUTPUT.PUT_LINE(g_comm);
      ELSE
         RAISE_APPLICATION_ERROR(-20210, 'Invalid Commission');
      END IF;
   END;

   --호출당하는 함수가 아래에 위치함.
   FUNCTION validate_comm(v_comm IN NUMBER)
      RETURN BOOLEAN
   IS
      v_max_comm NUMBER;
   BEGIN
      SELECT MAX(comm) INTO v_max_comm
      FROM emp;
      IF v_comm > v_max_comm THEN
         RETURN FALSE;
      ELSE
         RETURN TRUE;
      END IF;
   END;

   END;
   /

   SQL> @demo.sql

   Package created.

   Warning: Package Body created with compilation errors.

   SQL>show error
   SQL> show error
   Errors for PACKAGE BODY EMP_COMM:

   LINE/COL ERROR
   -------- -----------------------------------------------------------------
   7/3      PL/SQL: Statement ignored
   7/14     PLS-00313: 'VALIDATE_COMM' not declared in this scope


REM ONE-TIME ONLY 프로시저
   - 패키지가 사용자 세션에서 처음으로 호출될 때 one-TIME ONLY 프로시저가 한 번 실행된다.
1. What?
   1)PACKAGE 내에서 정의된 프로시저 또는 함수가 사용자에 의해 호출될 때 최초 반드시 한 번
   실행되는 프로시저이다.
   2)주로 package가 실행될 때 기본적으로 처리해야 할 로직이나 변수들의 초기화 값을 설정해야
   하는 경우 사용되는 기능이다.
   3)one-TIME ONLY 프로시저는 패키지 몸체부의 가장 마지막 부분에 BEGIN 절과 함께
   정의하면 된다.

   CREATE OR REPLACE PACKAGE emp_comm  IS
```

```
479         g_comm  NUMBER := 10;
480         FUNCTION validate_comm(v_comm IN NUMBER)
481            RETURN BOOLEAN;
482         PROCEDURE reset_comm(v_comm IN NUMBER);
483      END;
484      /
485
486      CREATE OR REPLACE PACKAGE BODY emp_comm IS
487      FUNCTION validate_comm(v_comm IN NUMBER)
488         RETURN BOOLEAN
489      IS
490         v_max_comm NUMBER;
491      BEGIN
492         SELECT MAX(comm) INTO v_max_comm
493         FROM emp;
494         IF v_comm > v_max_comm THEN
495            RETURN FALSE;
496         ELSE
497            RETURN TRUE;
498         END IF;
499      END;
500
501      PROCEDURE reset_comm(v_comm IN NUMBER)
502      IS
503         v_valid   BOOLEAN;
504      BEGIN
505         v_valid := validate_comm(v_comm);
506         IF v_valid = TRUE THEN
507            DBMS_OUTPUT.PUT_LINE(g_comm);
508         ELSE
509            RAISE_APPLICATION_ERROR(-20210, 'Invalid Commission');
510         END IF;
511      END;
512
513      --패키지가 실행될 때 가장 먼저 실행되는 ONE-TIME ONLY 프로시저로서 패키지내에서 사용될
         변수를 초기화하는 경우 또는 반드시 먼저 실행되어야 할 SQL문이 있는 경우에 사용된다.
514      BEGIN
515         SELECT AVG(sal) INTO g_comm
516         FROM emp;
517         DBMS_OUTPUT.PUT_LINE(g_comm);
518      END;
519      /
520
521   REM 패키지 함수
522      -호출되는 패키지 내의 테이블과 변수에 대한 읽기/쓰기를 제한할 때 사용되는 함수.
523   1. 이 함수는 패키지 내의 프로시저 또는 함수 내에서 실행되는 DML문에 의해 테이블이 변경되거나,
      변수 값이 변경되는 행위를 가능하게 할 것인지를 제한할 수 있는 기능을 갖고 있다.
524   2. 만약 어떤 프로시저나 함수는 순수하게 SELECT문만을 실행해야 한다면 READ(읽기작업)만
      가능한 상태로 환경을 설정할 수 있다.
525   3. 대부분의 프로시저, 함수는 기본적으로 생성하면 READ, WRITE가 가능하다.
526   4. 환경설정에 사용되는 상태 값
527      1)WNDS(WRITE No DATABASE State) : 테이블에 대한 DML 문 수행이 안된다.
528      2)WNPS(WRITE NO PACKAGE State) : 패키지 내의 지역변수 값을 변경할 수 없다.
529      3)RNDS(READ NO DATABASE State) : 테이블에 대한 SELECT문 수행이 안된다.
530      4)RNPS(READ NO PACKAGE State) : 패키지 내의 전역변수 값을 참조할 수 없다.
531
```

```
CREATE OR REPLACE PACKAGE emp_comm  IS
   g_comm  NUMBER := 10;
   FUNCTION validate_comm(v_comm IN NUMBER)
      RETURN BOOLEAN;
   PROCEDURE reset_comm(v_comm IN NUMBER);

   PRAGMA RESTRICT_REFERENCES (reset_comm, WNDS, RNDS);
   --reset_comm 프로시저는 테이블을 변경할 수 없으며, 패키지 내의 테이블을 질의할 수 없다.
END;
/

CREATE OR REPLACE PACKAGE BODY emp_comm IS
FUNCTION validate_comm(v_comm IN NUMBER)
   RETURN BOOLEAN
IS
   v_max_comm NUMBER;
BEGIN
   SELECT MAX(comm) INTO v_max_comm
   FROM emp;
   IF v_comm > v_max_comm THEN
      RETURN FALSE;
   ELSE
      RETURN TRUE;
   END IF;
END;

PROCEDURE reset_comm(v_comm IN NUMBER)
IS
   v_valid   BOOLEAN;
BEGIN
   v_valid := validate_comm(v_comm);
   IF v_valid = TRUE THEN
      DBMS_OUTPUT.PUT_LINE(g_comm);
   ELSE
      RAISE_APPLICATION_ERROR(-20210, 'Invalid Commission');
   END IF;
END;

END;
/

@demo.SQL

Warning : PACKAGE BODY created WITH compilation errors.

SQL> show error
Errors for PACKAGE BODY EMP_COMM:

LINE/COL ERROR
-------- -----------------------------------------------------------------
16/2     PLS-00452: Subprogram 'RESET_COMM' violates its associated pragma


REM 출력을 위한 DBMS_OUTPUT PACKAGE
   SQL> conn  sys as sysdba

```

```
SQL> DESC DBA_objects
SQL> SELECT object_name FROM dba_objects
  2  WHERE  object_type='PACKAGE' AND
  3  object_name LIKE  'DBMS_%'
  4  ORDER  BY  object_name;
```

결과 중에 DBMS_OUTPUT이 있다.

1)DBMS_OUTPUT패키지는 PL/SQL에서입력을 받거나 어떤 처리에 따라 결과를 화면에 출력할 때 사용되는 프로시저나 함수를 제공한다.

```
SQL> desc DBMS_OUTPUT
```

2)다음은 DBMS_OUTPUT 패키지의 여러 프로시저 중 자주 사용되는 것

DISABLE :화면에 문자열을 출력하는 모드를 해제한다.

ENABLE : 화면에 문자열을 출력하는 모드를 설정한다.

GET_LINE / GET_LINES : 현재 라인에서 입력한 값을 읽어 간다.

NEW_LINE : GET_LINE에서 읽혀진 행의 다음 라인을 읽는다.

PUT / PUT_LINE : 주어진 데이터를 화면에 출력한다.