KH정보교육원

# PROCEDURE & FUNCTION

# PROCEDURE & FUNCTION

KH정보교육원

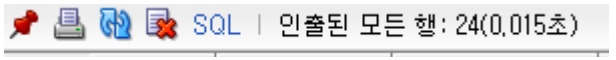
## PROCEDURE

PL/SQL문을 저장하는 객체이다. 필요할 때마다 복잡한 구문을 다시 입력 할 필요 없이 간단하게 호출해서 실행결과를 얻을 수 있다.

```
CREATE TABLE EMP_DUP
AS SELECT * FROM EMPLOYEE;
```

```
Table EMP_DUP이(가) 생성되었습니다.
```

```
SELECT * FROM EMP_DUP;
```



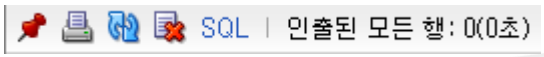

```
CREATE OR REPLACE
PROCEDURE DEL_ALL_EMP
IS
BEGIN
  DELETE FROM EMP_DUP;
  COMMIT;
END;
/
```

```
Procedure DEL_ALL_EMP이(가) 컴파일되었습니다.
```

```
EXEC DEL_ALL_EMP;
```

```
PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
SELECT * FROM EMP_DUP;
```



# PROCEDURE & FUNCTION

KH정보교육원

## PROCEDURE

```
DESC USER_SOURCE;
```

```
이름   널 유형
---- - --------------
NAME   VARCHAR2(30)
TYPE   VARCHAR2(12)
LINE   NUMBER
TEXT   VARCHAR2(4000)
```

```
SELECT * FROM USER_SOURCE;
```

| | NAME | TYPE | LINE | TEXT |
|---|---|---|---|---|
| 1 | DEL_ALL_EMP | PROCEDURE | 1 | PROCEDURE DEL_ALL_EMP |
| 2 | DEL_ALL_EMP | PROCEDURE | 2 | IS |
| 3 | DEL_ALL_EMP | PROCEDURE | 3 | BEGIN |
| 4 | DEL_ALL_EMP | PROCEDURE | 4 | DELETE FROM EMP_DUP; |
| 5 | DEL_ALL_EMP | PROCEDURE | 5 | |
| 6 | DEL_ALL_EMP | PROCEDURE | 6 | COMMIT; |
| 7 | DEL_ALL_EMP | PROCEDURE | 7 | END; |

# PROCEDURE & FUNCTION

KH정보교육원

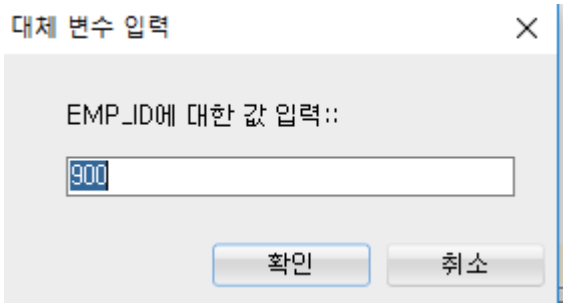
## PROCEDURE

[매개변수 있는 프로시져]

```
CREATE OR REPLACE PROCEDURE DEL_EMP_ID
                                    (V_EMP_ID EMPLOYEE.EMP_ID%TYPE)
IS
BEGIN
  DELETE FROM EMPLOYEE
  WHERE EMP_ID = V_EMP_ID;
  COMMIT;
END;
/
```

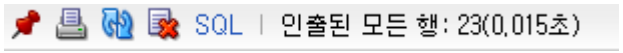
```
Procedure DEL_EMP_ID이(가) 컴파일되었습니다.
```

```
EXECUTE DEL_EMP_ID('&EMP_ID');
```




```
이전:EXECUTE DEL_EMP_ID('&EMP_ID')
신규:EXECUTE DEL_EMP_ID('900')

PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
SELECT * FROM EMPLOYEE;
```



# PROCEDURE & FUNCTION

KH정보교육원

## PROCEDURE

[IN/OUT 매개변수 있는 프로시져]

```
CREATE OR REPLACE PROCEDURE SELECT_EMPID(
  V_EMP_ID IN EMPLOYEE.EMP_NO%TYPE,
  V_EMP_NAME OUT EMPLOYEE.EMP_NAME%TYPE,
  V_SALARY OUT EMPLOYEE.SALARY%TYPE,
  V_BONUS OUT EMPLOYEE.BONUS%TYPE
)
IS
BEGIN
  SELECT EMP_NAME, SALARY, NVL(BONUS, 0)
  INTO V_EMP_NAME, V_SALARY, V_BONUS
  FROM EMPLOYEE
  WHERE EMP_ID = V_EMP_ID;
END;
/
```

```
Procedure SELECT_EMPID이(가) 컴파일되었습니다.
```

```
VARIABLE VAR_EMP_NAME VARCHAR2(30);
VARIABLE VAR_SALARY NUMBER;
VARIABLE VAR_BONUS NUMBER;
```

** 바인드변수 선언

# PROCEDURE & FUNCTION

KH정보교육원

## PROCEDURE

```
EXEC SELECT_EMPID(200, :VAR_EMP_NAME, :VAR_SALARY, :VAR_BONUS);
```

```
PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
PRINT VAR_EMP_NAME;
PRINT VAR_SALARY;
PRINT VAR_BONUS;
```

```
VAR_EMP_NAME
---
선동일
VAR_SALARY
-------
8000000
VAR_BONUS
---
0.3
```

** 프로시져 실행과 동시에 모든 바인딩 변수를 출력하기 위해서는
SET AUTOPRINT ON; 을 실행시킨다.

# PROCEDURE & FUNCTION

KH정보교육원

## FUNCTION

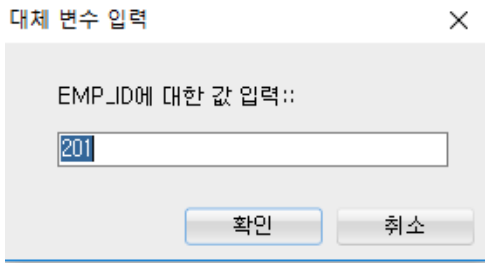
프로시져와 거의 유사한 용도로 사용하지만, 실행 결과를 되돌려 받을 수 있는 점에서 프로시져와 다르다.

```
CREATE OR REPLACE FUNCTION
BONUS_CALC(V_EMPID EMPLOYEE.EMP_ID%TYPE)
RETURN NUMBER
IS
  V_SAL EMPLOYEE.SALARY%TYPE;
  V_BONUS EMPLOYEE.BONUS%TYPE;
  CALC_SAL NUMBER;
BEGIN
  SELECT SALARY, NVL(BONUS, 0)
  INTO V_SAL, V_BONUS
  FROM EMPLOYEE
  WHERE EMP_ID = V_EMPID;

  CALC_SAL := (V_SAL + (V_SAL * V_BONUS)) * 12;
  RETURN CALC_SAL;

END;
/
```

```
Function BONUS_CALC이(가) 컴파일되었습니다.
```

```
VARIABLE VAR_CALC NUMBER;
```

```
EXEC :VAR_CALC :=
        BONUS_CALC('&EMP_ID');
```




```
PL/SQL 프로시저가 성공적으로 완료되었습니다.

CALC_SAL
--------
72000000
```

# PROCEDURE & FUNCTION

KH정보교육원

## FUNCTION

```
SELECT EMP_ID,
       EMP_NAME,
       BONUS_CALC(EMP_ID)
FROM EMPLOYEE;
```

SQL | 인출된 모든 행: 23(0초)

| | EMP_ID | EMP_NAME | BONUS_CALC(EMP_ID) |
|---|---|---|---|
| 1 | 200 | 선동일 | 124800000 |
| 2 | 201 | 송종기 | 72000000 |
| 3 | 202 | 노옹철 | 44400000 |
| 4 | 203 | 송은희 | 33600000 |
| 5 | 204 | 유재식 | 48960000 |
| 6 | 205 | 정중하 | 46800000 |
| 7 | 206 | 박나라 | 21600000 |
| 8 | 207 | 하이유 | 29040000 |
| 9 | 208 | 김해술 | 30000000 |
| 10 | 209 | 심봉선 | 48300000 |
| 11 | 210 | 윤은해 | 24000000 |
| 12 | 211 | 전형돈 | 24000000 |
| 13 | 212 | 장쯔위 | 38250000 |
| 14 | 213 | 하동운 | 30624000 |
| 15 | 214 | 방명수 | 16560000 |
| 16 | 215 | 대북혼 | 45120000 |
| 17 | 216 | 차태연 | 40032000 |
| 18 | 217 | 전지연 | 57096000 |
| 19 | 218 | 이오리 | 34680000 |
| 20 | 219 | 임시환 | 18600000 |
| 21 | 220 | 이중석 | 29880000 |
| 22 | 221 | 유하진 | 29760000 |
| 23 | 222 | 이태림 | 39467088 |

```
SELECT EMP_ID,
       EMP_NAME,
       BONUS_CALC(EMP_ID)
FROM EMPLOYEE
WHERE BONUS_CALC(EMP_ID) >= 30000000
ORDER BY 3 DESC;
```

SQL | 인출된 모든 행: 15(0초)

| | EMP_ID | EMP_NAME | BONUS_CALC(EMP_ID) |
|---|---|---|---|
| 1 | 200 | 선동일 | 124800000 |
| 2 | 201 | 송종기 | 72000000 |
| 3 | 217 | 전지연 | 57096000 |
| 4 | 204 | 유재식 | 48960000 |
| 5 | 209 | 심봉선 | 48300000 |
| 6 | 205 | 정중하 | 46800000 |
| 7 | 215 | 대북혼 | 45120000 |
| 8 | 202 | 노옹철 | 44400000 |
| 9 | 216 | 차태연 | 40032000 |
| 10 | 222 | 이태림 | 39467088 |
| 11 | 212 | 장쯔위 | 38250000 |
| 12 | 218 | 이오리 | 34680000 |
| 13 | 203 | 송은희 | 33600000 |
| 14 | 213 | 하동운 | 30624000 |
| 15 | 208 | 김해술 | 30000000 |

# CURSOR

처리 결과가 여러 개의 행으로 구해지는 SELECT문을 처리하기 위해 처리 결과를 저장해 놓은 객체이다. CURSOR~ OPEN~ FETCH~ CLOSE 단계로 진행된다.

# CURSOR의 상태

| 속 성 | 설 명 |
|---|---|
| %NOTFOUND | 커서 영역의 자료가 모두 FETCH되어 다음 영역이 존재하지 않으면 TRUE |
| %FOUND | 커서 영역에 아직 FETCH되지 않은 자료가 있으면 TRUE |
| %ISOPEN | 커서가 OPEN된 상태이면 TRUE |
| %ROWCOUNT | 커서가 얻어 온 레코드의 갯수 |

KH정보교육원

# CURSOR

```
CREATE OR REPLACE PROCEDURE CURSOR_DEPT
IS
   V_DEPT DEPARTMENT%ROWTYPE;
   CURSOR C1
   IS
   SELECT * FROM DEPARTMENT;
BEGIN
   OPEN C1;
   LOOP
     FETCH C1 INTO V_DEPT.DEPT_ID,
                   V_DEPT.DEPT_TITLE,
                   V_DEPT.LOCATION_ID;

     EXIT WHEN C1%NOTFOUND;

     DBMS_OUTPUT.PUT_LINE('부서코드 : ' || V_DEPT.DEPT_ID
                          || ' , 부서명 : ' || V_DEPT.DEPT_TITLE
                          || ' , 지역 : ' || V_DEPT.LOCATION_ID);
  END LOOP;
  CLOSE C1;
END;
/
```

```
PL/SQL 프로시저가 성공적으로 완료되었습니다.

부서코드 : D1 , 부서명 : 인사관리부 , 지역 : L1
부서코드 : D2 , 부서명 : 회계관리부 , 지역 : L1
부서코드 : D3 , 부서명 : 마케팅부 , 지역 : L1
부서코드 : D4 , 부서명 : 국내영업부 , 지역 : L1
부서코드 : D5 , 부서명 : 해외영업1부 , 지역 : L2
부서코드 : D6 , 부서명 : 해외영업2부 , 지역 : L3
부서코드 : D7 , 부서명 : 해외영업3부 , 지역 : L4
부서코드 : D8 , 부서명 : 기술지원부 , 지역 : L5
부서코드 : D9 , 부서명 : 총무부 , 지역 : L1
```

# PROCEDURE & FUNCTION

KH정보교육원

## CURSOR

```
CREATE OR REPLACE PROCEDURE CURSOR_DEPT
IS
   V_DEPT DEPARTMENT%ROWTYPE;
   CURSOR C1
   IS
   SELECT * FROM DEPARTMENT;
BEGIN
  FOR V_DEPT IN C1 LOOP

    DBMS_OUTPUT.PUT_LINE('부서코드 : ' || V_DEPT.DEPT_ID
                                || ' , 부서명 : ' || V_DEPT.DEPT_TITLE
                                || ' , 지역 : ' || V_DEPT.LOCATION_ID);
  END LOOP;
END;
/
```

```
PL/SQL 프로시저가 성공적으로 완료되었습니다.

부서코드 : D1 , 부서명 : 인사관리부 , 지역 : L1
부서코드 : D2 , 부서명 : 회계관리부 , 지역 : L1
부서코드 : D3 , 부서명 : 마케팅부 , 지역 : L1
부서코드 : D4 , 부서명 : 국내영업부 , 지역 : L1
부서코드 : D5 , 부서명 : 해외영업1부 , 지역 : L2
부서코드 : D6 , 부서명 : 해외영업2부 , 지역 : L3
부서코드 : D7 , 부서명 : 해외영업3부 , 지역 : L4
부서코드 : D8 , 부서명 : 기술지원부 , 지역 : L5
부서코드 : D9 , 부서명 : 총무부 , 지역 : L1
```

** FOR IN LOOP를 이용하면 LOOP 반복시 자동으로 CURSOR를 OPEN하고 행을 인출(FETCH)한다.
   LOOP 종료시 자동으로 CURSOR를 CLOSE한다.

# PROCEDURE & FUNCTION

KH정보교육원

## CURSOR

```
CREATE OR REPLACE PROCEDURE CURSOR_DEPT
IS
   V_DEPT DEPARTMENT%ROWTYPE;
BEGIN
  FOR V_DEPT IN (SELECT * FROM DEPARTMENT) LOOP

    DBMS_OUTPUT.PUT_LINE('부서코드 : ' || V_DEPT.DEPT_ID
                                || ' , 부서명 : ' || V_DEPT.DEPT_TITLE
                                || ' , 지역 : ' || V_DEPT.LOCATION_ID);
  END LOOP;
END;
/
```

** FOR IN 문을 사용하면 커서의 선언도 생략할 수 있다.

```
PL/SQL 프로시저가 성공적으로 완료되었습니다.

부서코드 : D1 , 부서명 : 인사관리부 , 지역 : L1
부서코드 : D2 , 부서명 : 회계관리부 , 지역 : L1
부서코드 : D3 , 부서명 : 마케팅부 , 지역 : L1
부서코드 : D4 , 부서명 : 국내영업부 , 지역 : L1
부서코드 : D5 , 부서명 : 해외영업1부 , 지역 : L2
부서코드 : D6 , 부서명 : 해외영업2부 , 지역 : L3
부서코드 : D7 , 부서명 : 해외영업3부 , 지역 : L4
부서코드 : D8 , 부서명 : 기술지원부 , 지역 : L5
부서코드 : D9 , 부서명 : 총무부 , 지역 : L1
```

# PROCEDURE & FUNCTION

KH정보교육원

## PACKAGE

프로시져와 함수를 보다 효율적으로 관리하기 위해서 패키지 단위로 묶어 관리한다.

```
CREATE OR REPLACE PACKAGE KH_PACK
IS
  PROCEDURE CURSOR_DEPT;
  FUNCTION BONUS_CALC(V_EMPID EMPLOYEE.EMP_ID%TYPE)
    RETURN NUMBER;
END;
/
```

# PROCEDURE & FUNCTION

KH정보교육원

## PACKAGE

```
CREATE OR REPLACE PACKAGE BODY KH_PACK
IS
 PROCEDURE SHOW_EMP
 IS
   V_EMP EMPLOYEE%ROWTYPE;
   CURSOR C1
   IS
   SELECT EMP_ID, EMP_NAME, EMP_NO
   FROM EMPLOYEE;
   BEGIN
     FOR V_EMP IN C1 LOOP
      DBMS_OUTPUT.PUT_LINE('사번 : ' || V_EMP.EMP_ID ||
                           ', 이름 : ' || V_EMP.EMP_NAME ||
                           ', 주민번호 : ' || V_EMP.EMP_NO);
     END LOOP;
   END;
END;
/
```

```
EXEC KH_PACK.CURSOR_DEPT;
```

** 패키지명.프로시져명 혹은 패키지명.함수명 으로 사용한다.