

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

## 6차시: PL/SQL Module(Named Block = Stored Block)

### I. 시작

#### [학습목표]

PL/SQL은 Anonymous Block 과 Stored(Named) Block 2가지 유형의 Block이 있습니다. 6차시에는 Stored Block을 활용하여 사용자 정의 Function , Procedure, Package, Trigger를 학습합니다.

#### [학습목차]

6-1 Block의 유형

6-2 Parameter 유형

6-3 FunctionW

### 6-1 Block의 유형

이전 차시에 학습 했던 내용을 복습 해보죠!

PL/SQL BLOCK은 2가지 유형이 있습니다.

(1) Anonymous Block   (2) Named Block (= Stored Block)

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

## 6-1-1 Anonymous Block

사전적인 의미대로 해석을 하면 이름이 없는 무명의 Block 입니다.

Anonymous Block의 주요한 특징은

- ① 저장위치 : Client Program 내에 저장되거나 독립적인 SQL Script 형태로 저장

```
for (;;) /* Loop infinitely */
{
    printf("\n** Name of employee? (<return> to end) ");
    gets(empname.arr); /* Get the name */
    if (strlen(empname.arr) == 0) /* No name entered, */
    {
        EXEC SQL COMMIT WORK RELEASE; /* so log off ORACLE */
        exit(0); /* and exit program */
    }
    empname.len = strlen(empname.arr);

    /* ----- */
    /* ----- Begin the PL/SQL block ----- */
    /* ----- */
    EXEC SQL EXECUTE

    BEGIN
        SELECT job, hiredate, sal, deptno
            INTO :jobtype, :hired, :salary, :dept FROM emp
            WHERE ename = UPPER(:empname);

        /* Get number of people whose length of *
         * service is longer */
        SELECT count(*) INTO :worked_longer FROM emp
            WHERE hiredate < :hired;

        /* Get number of people with a higher salary */
        SELECT count(*) INTO :higher_sal FROM emp
            WHERE sal > :salary;

        /* Get number of people in the same department */
        SELECT count(*) INTO :total_in_dept FROM emp
            WHERE deptno = :dept;
    END;

    END-EXEC;
```

<참고> Pro\*C 에서 Anonymous Block을 사용하는 예제

Pro\*C 는 PreCompiler C 의 약자로 Oracle社 에서 제공하는 개발언어로  
ANSI C 언어에 Oracle社의 Library 와 PreCompiler를 추가  
가볍고 이식성 좋은 C 언어의 장점과 데이터 처리에 강점을 가진 언어인  
SQL,PL/SQL을 결합한 형태.

Host Language( ) + Embedded Language( )

- ② 실행방식 : 실행시점에 해당 Block이 DBMS에게 전달되어 실행

- ③ 용 도 : 단위 Application 내에서만 고유하게 사용하는 Module 작성시 사용  
Anonymous Block이 저장되고 실행되는 방식을 보면 각 단위  
Application내에 저장되어 있다가 실행시점에 DBMS로 전달.

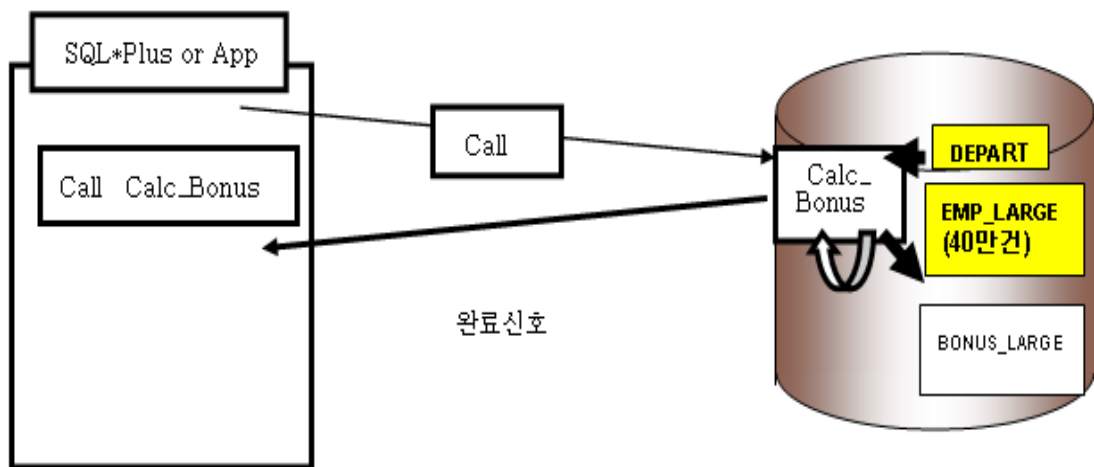
# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

공용 Library 처럼 다수의 Application에서 재사용(REUSE)하는  
목적이 아닌 단위 Application이 독립적으로 실행하는 Module 작성을  
위해 사용.

## 6-1-2 Named Block

사전적인 의미대로 해석을 하면 이름이 있는 Block

- ① 저장위치 : DBMS내에 Data Dictionary에 저장
- ② 실행방식 : 실행시점에 해당 Block을 호출(Call)하여 실행



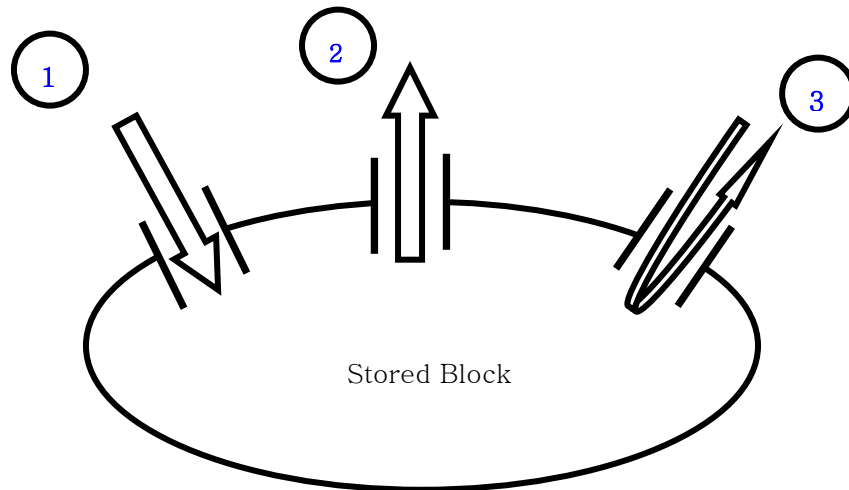
<참고> DBMS 서버내에 calc\_bonus 라는 이름(Named)의 Module이 저장되어 있는  
경우 Client 프로그램 내에서는 해당 Module을 Call(호출)을 하면 DBMS 서버내에서  
calc\_bonus Module이 처리된 후 종료신호를 보내는 과정받는 과정

- ③ 용   도 : 공용 Library Module  
개개의 Application내에 저장되는 것이 아니라 중앙의 DBMS내에 저장되어  
1개의 Module을 다수의 Application이 호출하여 재사용하는 방식

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

## 6-2 Parameter 유형

Parameter의 3가지 유형(Mode)



### ① IN

Stored Block 외부에서 있는 값을 Stored Block 에게 전달하는 Parameter Mode  
(예: 함수 수행시의 입력값 )

### ② OUT

Stored Block 내부에서 있는 값을 Stored Block 외부에 전달하는 Parameter Mode  
(예: 프로시저 수행 결과를 저장하여 전달하기 위한 용도)

### ③ IN OUT

IN 와 Out 2가지 역할 수행

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

## 6-3 Function

6\_function\_1.sql

```
CREATE OR REPLACE FUNCTION CALC_BONUS(P_SALARY IN NUMBER,P_DEPTNO IN NUMBER)
RETURN NUMBER
IS
    V_BONUS_RATE          NUMBER          := 0;
    V_BONUS                NUMBER(7,2)    := 0;
BEGIN
    -- 근무 부서별로 Bonus 차등 지급
    IF    P_DEPTNO = 10      THEN
        V_BONUS_RATE := 0.1;
    ELSIF P_DEPTNO = 20      THEN
        V_BONUS_RATE := 0.2;
    ELSIF P_DEPTNO = 30      THEN
        V_BONUS_RATE := 0.3;
    ELSE
        V_BONUS_RATE := 0.05;
    END IF;
    V_BONUS := ROUND(NVL(P_SALARY,0) * C_BONUS_RATE,2);
    RETURN  V_BONUS;
END CALC_BONUS;
/
```

- ① CREATE OR REPLACE 의 의미는 동일한 OBJECT가 없으면 새로 생성하고 동일한 OBJECT가 있으면 수정하라는 의미 입니다. **OR REPLACE**는 선택적(Optional)구문 이지만 CREATE OR REPLACE로 사용 하는 것이 개발 및 유지보수 진행중에 편리한 이유는 개발시 공통 Module생성은 1회에 완료되는것이 아니라 빈번한 수정 사항이발생 합니다. 그런 상황에서 OR REPLACE 구문은 편리한 역할을 합니다.

CALC\_BONUS는 **Stored Block(=Stored Block)의 이름**입니다.

해당 이름으로 DBMS 서버내에 저장 되고 이름을 통해 사용됩니다.

P\_SALARY, P\_DEPTNO는 함수 내부로 전달(IN)되는 PARAMETER 입니다.

P\_SALARY IN NUMBER

P\_SALARY 매개변수 이름(PARAMETER NAME)

IN 매개변수 모드(PARAMETER MODE)

NUMBER 매개변수 데이터타입(PARAMETER DATA TYPE)

3가지 PARAMETER MODE중 IN MODE가 DEFAULT 입니다.

P\_SALARY NUMBER 라고 표기해도 동일한 의미를 가지게 됩니다.

PARAMETER DATA TYPE시 주의 사항은 DATA LENGTH(길이)를 표현하지 않고

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

유형만 표현하는 부분입니다. P\_SALARY IN NUMBER(6) 은 틀린 표현입니다.  
처음 접하시는 경우 실수 하기 쉬운 부분입니다.

RETURN 은 함수가 리턴하는 결과값의 데이터 유형을 표시하는 부분입니다.  
위의 함수는 보너스값을 계산하여 리턴해주는 함수 이므로 NUMBER  
TYPE이 사용 됩니다.

IS 는 Anonymous Block의 DECLARE 구문과 동일한 의미 입니다.

IS ~ BEGIN 사이가 선언부 입니다. IS 대신 AS가 사용될수 있습니다.

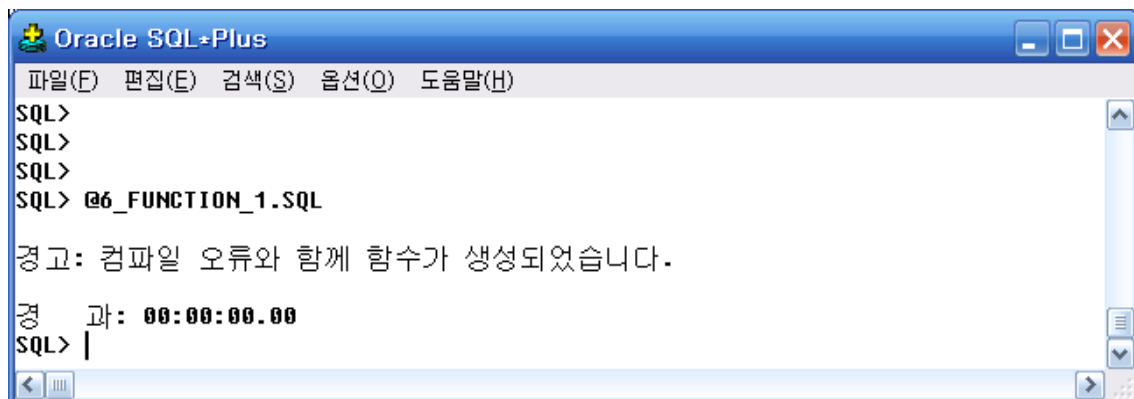
## ② Stored block 디버깅

C\_BONUS\_RATE 라는 변수는 정의되지 않은 변수 입니다.

구문상의 오류를 일으켜 함수 생성시 COMPILE TIME Error 가 발생 합니다.

함수 생성시 디버깅 방법 및 오류 수정하는 방법을 보기 위한 부분 입니다.

C\_BONUS\_RATE ➔ V\_BONUS\_RATE로 수정해야 함수가 생성 됩니다.



<참고> Stored Block 생성시 발생 에러는 USER\_ERRORS라는 데이터 디렉터리에 저장

1) SELECT \* FROM USER\_ERRORS;

2) SHOW ERROR

개발하는 과정에서 ERROR가 발생하면 디버깅을 하기 위해 SELECT를 사용해서  
에러를 조회하는것이 불편해서 SHOW ERROR 명령어를 지원.

<참고> 약간의 주의가 필요한 부분인데.. 에러의 위치는 개발 소스상의 위치가

아니라 SQL\*PLUS의 SQL BUFFER상의 위치 입니다. 6\_FUNCTION\_1.SQL  
소스에서 22 라인이 아닙니다.

## 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

L이라는 명령어는 LIST 의 약자로 SQL\*PLUS 명령어 입니다.

SQL BUFFER의 내용을 LIST 하라는 의미 입니다.

SQL BUFFER상의 22 LINE에서 ERROR가 발생 했습니다. 아래는

소스상에는 29 LINE에 해당하죠! 간단하지만 정확한 구분을 해두어야 복잡한

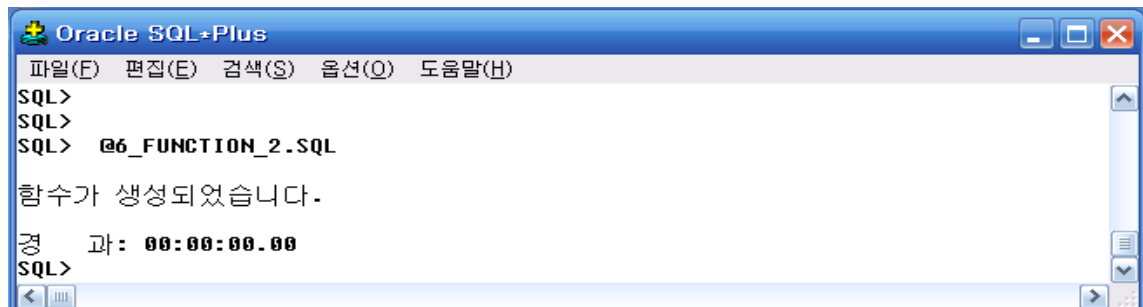
PL/SQL 개발시 혼동이 생기지 않습니다.

### 실습 2단계

6\_FUNCTION\_2.SQL은 6\_FUNCTION\_1.SQL 에서 에러가 발생하는

부분인 C\_BONUS\_RATE ==> V\_BONUS\_RATE 로 수정한후 함수를 생성 하십시오

## 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

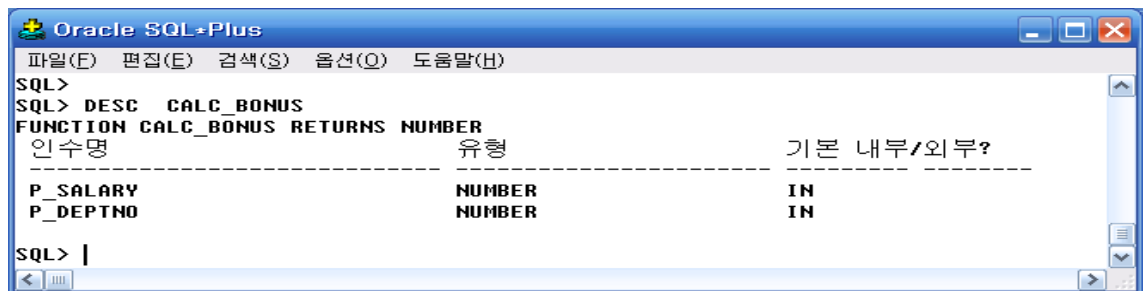


<참고> 함수가 정상적으로 생성 되었습니다.

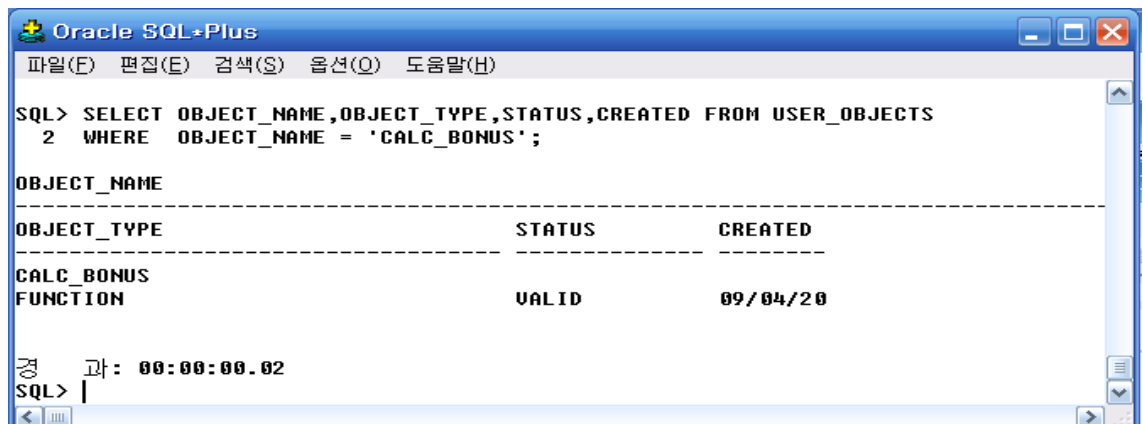
Stored Block은 DBMS 서버내에 데이터 디렉터리에 저장됩니다.

자! 생성된 함수를 확인해봅시다.

### 1) DESC CALC\_BONUS



### 2) 데이터 디렉터리에 생성된 함수 확인 하기



개발 진행중 또는 운영중에 FUNCTION 또는 PROCEDURE들이 정상적인 실행이 안되는 경우 USER\_OBJECTS의 STATUS를 제일 먼저 확인합니다.

### 3) 데이터 디렉터리에 저장된 FUNCTION Text SOURCE 확인

SELECT NAME,LINE,TEXT FROM USER\_SOURCE WHERE NAME = 'calc\_bonus'



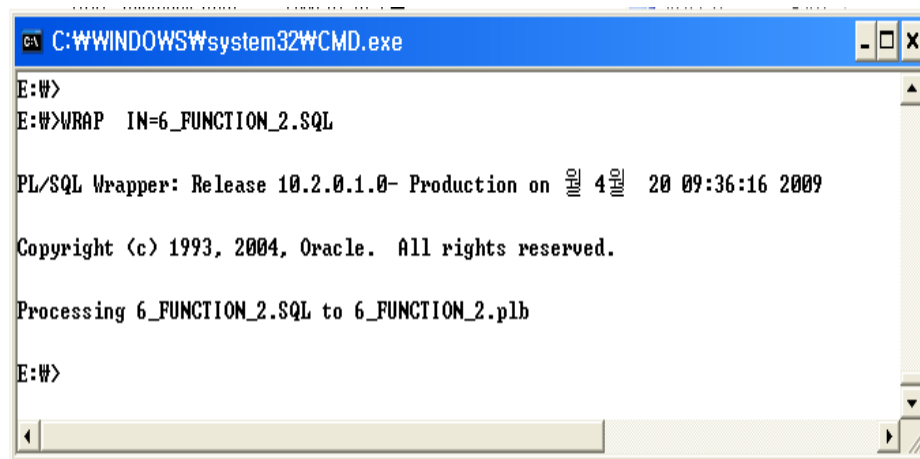
## 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

<참고> SELECT 를 통해 데이터 디렉터리에서 저장된 FUNCTION의 TEXT SOURCE를 확인할 수 있습니다. 일반 SI 개발이라면 유지 보수를 위해 TEXT SOURCE 상태로 배포가 되지만 지속적인 판매가 필요한 제품이라면 TEXT SOURCE 상태로 배포할 경우 보안이 필요한 내부 기술이 유출될 가능성.

#### 4) Binary SOURCE 로 Function 생성

Oracle社에서는 WRAP 이라는 유틸리티를 제공 한다 .WRAP은 TEXT SOURCE를 BINARY SOURCE로 변환 한다. 변환된 BINARY SOURCE로 Function을 생성하게 되면 위의 문제를 해결할수 있다.

#### <4-1 WRAP을 통해 Binary Source로 변환하는 방법 >



- 입력 파일명을 지정하면 출력 파일명은 자동으로 ~.PLB로

#### <4-2 변환된 Binary Source를 에디터상에서 확인>

```
CREATE OR REPLACE FUNCTION CALC_BONUS wrapped
3000000
34e
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
abcd
8
20f 140
FDaUsb4hNP0dv1bCT59s93Zw8v4wg/D3Lp7WfC9gmC7cfwxIcVEJnpQf+dvfcID09FCN/p3p
+Ormae3vtMz*Ml13IEr5vxXLMrxjfrHkw/qxhhya8v56mSxsClgV7xEgkQ/h8J+o2nhHMY
yUsl+UZS4Bk-RF6akOJbp4qVt4WH5szMQ/nuoF2WN3fd3SzVt2D4MsvaZtKlBfztga/
yYapVQZHyaWfS38/Zxx/fifr3QUElS35h0vPduUrdq6KG8rOUxx37Md7VHXVFmt7zwe6ejZWb
yX1oAbpL#QXkg315TJFD#0xyllM=
```

### < 4-3 Binary Source로 함수 생성 >

@6\_FUNCTION 2.PLB

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

< 4-4 데이터 덱서너리에서 Binary Source 확인>

```
SELECT LINE,TEXT FROM USER_SOURCE WHERE NAME = 'CALC_BONUS';
```

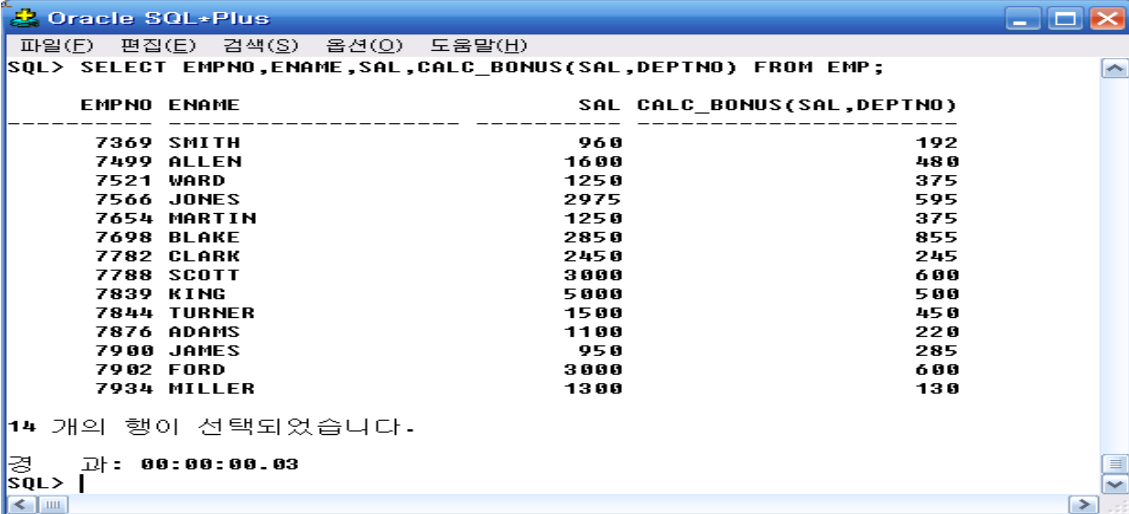
<WRAP 사용시 주의 >

- 1, 개발이 진행중인 상태에서는 Text Source가 효율적,  
개발이 완료 되어 최종 배포 상태에서 WRAP을 하여 배포,
- 2, Binary Source 상태에서 배포가 되므로 원본 소스 관리에 주의 필요  
Text Source는 원본 소스 손상시 데이터 덱서너리에 저장되어 있어 언제라도 추출이 가능,
- 3, 상용화되는 특정 Application에만 한정적으로 사용합니다.(예:오라클 DBMS 제공 내부 패키지 등) 일반응용 Application에 사용시에는 자칫하면 유지보수시 문제 초래

## 실습 3단계

FUNCTION CALC\_BONUS를 SELECT와 함께 실행 시켜봅시다. !

아래의 결과는 사용자가 정의한 함수를 SQL문과 함께 사용하는 예 입니다.



EMPNO	ENAME	SAL	CALC_BONUS(SAL,DEPTNO)
7369	SMITH	960	192
7499	ALLEN	1600	480
7521	WARD	1250	375
7566	JONES	2975	595
7654	MARTIN	1250	375
7698	BLAKE	2850	855
7782	CLARK	2450	245
7788	SCOTT	3000	600
7839	KING	5000	500
7844	TURNER	1500	450
7876	ADAMS	1100	220
7900	JAMES	950	285
7902	FORD	3000	600
7934	MILLER	1300	130

14 개의 행이 선택되었습니다.

경 과: 00:00:00.03

SQL> |

<참고> 위의 예제는 FUNCTION 학습을 위한 예제 입니다. 예제를 위한 예제 입니다.

- (1) 보너스 계산을 하는 방법이 부서번호를 기준으로 고정적으로 부여 되었죠!  
현실에서는 이렇게 지급하지 않습니다.

- (2) 사용자 정의 함수의 남용에 주의를 해야 합니다.

현업에서 성능 문제가 생기는 원인을 분석해보면 사용자 정의 함수의 남용으로 인해 발생하는 경우를 간혹 발견 하게 됩니다.

## 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

위의 예제에서 EMP TABLE은 14개의 데이터를 가지고 있습니다.

CALC\_BONUS 함수는 14번의 실행을 하게 됩니다.

만약 조회하는 대상 데이터가 140만건이라면 함수 역시 140만번 실행됩니다.

1회 실행이 0.001 초 가 걸린다면  $0.001 * 140\text{만번} \rightarrow 1400\text{초}$ , 함수 실행에만 1400초가 걸리게 됩니다. 이점을 꼭 기억해두셔야 합니다.

배보다 배꼽이 더 커질수 있습니다. ㅎㅎ SQL 실행보다 SQL에서 호출한 사용자정의 함수가 시간이 더 걸리게 되는...

현업에서 보면 종종 많은 개발자분들이 CODE 값을 Parameter로 해서 CODE명을 리턴하는 사용자 정의 함수를 만들어서 사용합니다.

사용자 정의 함수가 효율적인지 Join인 효율적인지에 대한 검토후 사용해야 합니다.

(3) 사용자 정의 함수는 **공용 LIBRARY** 의 개념 입니다.

개발자 개개인이 임의대로 만들어서 사용하는 것이 아니라.

분석/설계 단계에서 부터 공통 Module에 대한 Application 분석/설계에 고려되어 작성 해야 합니다. 무분별한 개개인의 함수 생성은 개발 효율성 및 유지보수에 비효율적인 결과를 가져 오게 됩니다.

### <참고>

SQL문에서 사용자가 생성한 Stored Block을 호출할 시 주의사항 입니다.

- ① PROCEDURE는 호출될 수 없다
- ② SINGLE-ROW FUNCTION 역할을 한다
- ③ PARAMETER MODE는 IN MODE 가 사용된다.
- ④ FUNCTION안에서 사용된 DATA TYPE은 SQL DATATYPE 만 사용되어야 한다
- ⑤ RETURN DATA TYPE은 SQL DATA TYPE만 사용되어야 한다.

### < 정리>

#### PARAMETER MODE

IN	OUT	IN OUT
DEFAULT Mode	명시적으로 Mode를 표기	명시적으로 Mode를 표기
SUBPROGRAM 안으로 값을 전달할 때 사용	SUBPROGRAM 밖으로 값을 전달할 때 사용	IN 과 OUT의 2가지 기능
상수처럼 참조용으로 사용된다 (AS A CONSTANT)	값이 할당되어 있지 않는 변수 처럼 사용된다, (UNINITIALIZED VARIABLE)	값이 할당되어 있는 변수 처럼 사용된다, (INITIALIZED VARIABLE)

# 데이터베이스 프로그래밍-PL/SQL Named Block (6차시)

- ① Parameter Mode를 명시적으로 표기 하지 않을 경우 Default 로 IN Mode로 사용되면 OUT, IN OUT을 사용하기 위해서는 명시적으로 표기를 해야 한다.
- ② IN 내부로 값을 전달할 때, OUT 외부로 값을 내보낼 때, IN OUT은 2가지 기능  
일반적으로 FUNCTION에서는 IN Mode를 주로 사용.  
일반적으로 PROCEDURE/PACKAGE에서는 IN,OUT,IN OUT을 필요시에 사용
- ③ IN Mode로 지정된 Parameter는 상수 처럼 사용  
즉 참조용으로만 사용되며 변수 처럼 해당 내용의 값을 수정 하거나  
다른 값으로 변경하여 저장 할 수는 없다.  
OUT, IN OUT Mode Parameter는 변수 처럼 참조 및 변경이 가능 하다