SELECT コマンドからのファンクション呼出し方

PL/SQL からのファンクションの呼出し方

PL/SQL からのストアドプロシージャの呼出し方

SQL*PLUS からのストアドプロシージャの呼出し方

SQL*PLUS からのファンクションの呼出し方

SQL*PLUS からの無名 PL/SQL ブロック実行

SQL 文の中でファンクションを直接呼び出す方法

【変数の種類】PL/SQL 内での変数（ストアドプロシージャ、ファンクション内での定義変数）

SQL*PLUS 内での定義変数

・ホスト変数

・バインド変数

ストアドプロシージャの定義方法

```
CREATE OR REPLACE PROCEDURE プロシージャ名（
  i_param  IN  NUMBER ,
  o_param  OUT NUMBER ,                -- 受渡し引数定義
      ・・・・・・・・・・ ）     IS ≒
   変数名 1 データ型 ; ═                -- ローカル変数定義
   変数名 2 データ型 ;
DECLARE
BEGIN
  DECLARE
            ブロック内限定のスコープ変数 定義
  BEGIN
            プログラム部
            RETURN ;
  END ;
END ;
```

ファンクションの定義方法

```
CREATE OR REPLACE FUNCTION  ファンクション名（
  i_param  IN    NUMBER ,
  o_param  OUT   NUMBER ,              -- 受渡し引数定義
      ・・・・・・・・・・ ）
   RETURN NUMBER    IS ≒
   変数名 1 データ型 ; ═                -- ローカル変数定義
   変数名 2 データ型 ;
DECLARE
BEGIN
  DECLARE
            ブロック内限定のスコープ変数 定義
  BEGIN
            プログラム部
            RETURN 戻り値 ;
  END ;
END ;
```

PL/SQL からのファンクションの呼出し方と変数定義

※ パッケージの呼出しには、パッケージ名を修飾してファンクション名を指定する

戻り値変数 : ＝ パッケージ名．ファンクション名（引数）；

```
~~BEGIN~~
    DECLARE
        a   NUMBER  ;
        c   NUMBER  ;
    BEGIN
        a  :=  func_A ( 2, c )  ;        --  ファンクションの呼出しと戻り値のセット
        DBMS_OUTPUT.PUT_LINE ( TO_CHAR ( a ))  ;
    END ;
~~END ;~~
/
```

PL/SQL からのストアドプロシージャの呼出し方と変数定義

※ パッケージの呼出しには、パッケージ名を修飾してプロシージャ名を指定する

~~EXECUTE~~ パッケージ名．プロシージャ名（引数）；

```
~~BEGIN~~
    DECLARE
        a   VARCHAR2 ( 20 )  ;
        c   NUMBER  ;
    BEGIN
        a := 0 ;                                    --  変数への値の代入
        ~~EXECUTE~~  proc_A (  c,  a )  ;       --  プロシージャの呼出し
        DBMS_OUTPUT.PUT_LINE ( a )  ;


    END ;

~~END ;~~
/
```

SELECT 文の中でファンクションを直接呼び出す方法

SQL> SELECT func_A(項目名) FROM テーブル ;

※ このファンクション呼出しの仕方では、引数に出力引数は指定できないので、注意すること

例)

```
SELECT ename , func_A( empno ) FROM emp ;

SELECT func_A( 1 ) FROM DUAL ;
```

使用例)

```
CREATE OR REPLACE FUNCTION Func_A(
  i_param IN NUMBER )
  RETURN VARCHAR2 IS
  NAME VARCHAR2(20) ;
BEGIN
  SELECT ENAME INTO NAME FROM EMP WHERE
EMPNO= i_param ;
  RETURN NAME ;
END ;
/

VARIABLE a VARCHAR2(20);
execute :a := Func_A(1);
print :a

select Func_A(1) FROM DUAL ;
```

```
FUNC_A(1)
----------
愛川こずえ
```

SQL*PLUS からのストアドプロシージャ（PL/SQL）の呼出し方と変数定義

※ パッケージの呼出しには、パッケージ名を修飾してプロシージャ名を指定する
EXECUTE　パッケージ名．プロシージャ名（バインド変数引数）；

```
                /*　ホスト変数の定義　*/
SQL>　VARIABLE　c　NCHAR（10）；
                /*　プロシージャの呼出し バインド変数での引数指定　*/
SQL>　EXECUTE　proc_A（1,　:c）；
                /*　SQL*Plus でのホスト変数値の表示　　*/
SQL>　print　c
```

SQL*PLUS からのストアドプロシージャの呼出しには、ホスト変数の前に「：」を付けて、バインド変数にして引数の指定を行う

SQL*PLUS での変数の種類	説　　明
ホスト変数	SQL*PLUS 内に確保した変数
バインド変数	ストアドプログラムへ渡すための引数の変数指定には、変数にコロン「：」を付ける

※ バインド変数操作は、SQL*PLUS からの呼出しの場合に限り指定が必要です
PL/SQL からの呼出しの場合には、不要です

SQL*PLUS からのファンクション（PL/SQL）の呼出し方
SQL コマンドからのファンクション呼出し方

```
SQL>　SELECT　func_A(1)　FROM　DUAL　；
```

※ このファンクション呼出しの仕方では、引数に出力引数は指定できないので、注意すること

※ パッケージの呼出しには、パッケージ名を修飾してファンクション名を指定する
戻り値変数 ＝ パッケージ名．ファンクション名（引数）；

```
                /*　ホスト変数の定義　*/
SQL>　VARIABLE　a　NUMBER　；
SQL>　VARIABLE　c　NUMBER　；

                /*　ファンクションの呼出しと戻り値のセット　*/
                /*　引数は、バインド変数にして引渡す　　*/
SQL>　EXECUTE　:a　:=　func_A（2,　:c）；

/*　「：」コロンの記述を忘れないように　　*/

/*　SQL*Plus でのホスト変数値の表示　　*/
SQL>　print　:a
```

SQL*PLUS からの無名 PL/SQL ブロックのコーディングと実行

```
SQL>  BEGIN
          DECLARE
             i_param   IN   NUMBER   ;
             o_param   OUT   VARCHAR2(20)  ;              --  変数定義
          BEGIN
           /*    プログラム部    */
              EXECUTE   proc_A ( i_param, o_param )  ;     --プロシージャ呼出し
              DBMS_OUTPUT.PUT_LINE (   o_param   )  ;
          END ;
      END ;

SQL>  ／
      /*    実行させるために、「／」をキー入力する    */
```

SQL*PLUS での変数（バインド変数）を使った Select 文の実行

```
SQL>  VARIABLE   a   NUMBER   ;
SQL>  BEGIN
          :a   :=   12   ;
        END   ;
        /
      PL/SQL プロシージャが正常に完了しました

SQL>  SELECT   user_name   FROM   user_table   WHERE   user_no   =   :a   ;
        /

      user_name
      --------------------
      愛川こずえ
```

使用例）

```
-- プロシージャの作成と使用 --
CREATE OR REPLACE FUNCTION Func_A(
  i_param IN NUMBER ,
  o_param OUT NUMBER )
RETURN NUMBER IS
BEGIN
Select count(*) into o_param from emp;
RETURN 5 ;
END ;
/

show error

VARIABLE a NUMBER ;
VARIABLE c NUMBER ;

EXECUTE :a := func_A ( 2, :c ) ;

print :a

-- プロシージャの作成と使用 --
CREATE OR REPLACE PROCEDURE PROC_A (
  i_param IN NUMBER ,
  o_param OUT NUMBER ) IS
BEGIN
Select count(*) into o_param from emp;
RETURN ;
END ;
/

SHOW ERROR

VARIABLE c NUMBER ;
EXECUTE PROC_A( 2, :c ) ;

PRINT :C
```