——— 動的 SQL 文の実行 ———　　(PL/SQL)

【FETCH とループ文を使用した複数レコード処理】

Where の条件内容を、対象とする変数を変更して SQL 文を作成し実行する
プロシージャの作成方法

プロシージャの作成において、動的な SQL 文の処理を行う場合のカーソル宣言は、**REF** CURSOR 句を使う

プロシージャの作成（１レコード処理、複数件処理）

```
CREATE OR REPLACE PROCEDURE get_count (
    str_where IN VARCHAR2 ,
    Ncount OUT NUMBER
) IS
    sql_statement VARCHAR ( 350 ) ;
    emp_record emp_table%ROWTYPE ;    -- レコード型変数の定義
    TYPE select_cursor_type IS REF CURSOR ; -- タイプ定義（カーソル）
    cursor1 select_cursor_type ;              -- 定義タイプに対する変数定義

BEGIN
    /* ここで、動的 SQL 文を作成する */
    sql_statement := ' SELECT count(*) from emp_table ' || str_where ;
    DBMS_OUTPUT.PUT_LINE ( sql_statement ) ;
    /* SQL 文の処理 ⇐ ここで、カーソル動作を確定させる          */
    OPEN cursor1 FOR sql_statement ;
    FETCH cursor1 INTO Ncount ;
                -- １回のフェッチのみ
    DBMS_OUTPUT.PUT_LINE ('cnt= ' || TO_CHAR ( Ncount ) ) ;
    CLOSE cursor1 ;

    /* ここで、動的 SQL 文を作成する */
    sql_statement := ' SELECT * from emp_table ' || str_where ;
    DBMS_OUTPUT.PUT_LINE (' - - - - - - - - - -') ;
    DBMS_OUTPUT.PUT_LINE ( sql_statement ) ;

    /* SQL 文の処理 ⇐ ここで、カーソル動作を確定させる          */
    OPEN cursor1 FOR sql_statement ;
    LOOP
        FETCH cursor1 INTO emp_record ;      -- フェッチ
        EXIT WHEN cursor1%NOTFOUND ;      -- フェッチの終了判断
        /*  読込んだデータは、「emp_record. 列名」の中にセットされている  */
        DBMS_OUTPUT.PUT_LINE(' 番号 : ' || emp_record.empno || ' 名前 : '
                || emp_record.ename ) ;
    END LOOP ;
    CLOSE cursor1 ;

END ;

/
```

プロシージャの呼出し

```
                    /*   SQL*Plus からプロシージャの呼出し                */


SQL>   SET   SERVEROUTPUT   ON;

                    /*   ホスト変数の定義   */
SQL>   VARIABLE   Istr_where   VARCHAR2 ( 300 ) ;
SQL>   VARIABLE   Ocount   NUMBER ;

        /*   引数は、バインド変数にして引渡す          */
SQL>   EXECUTE   :Istr_where   :=   ' WHERE   empno > 0 ' ;

SQL>   EXECUTE   get_count (   :Istr_where ,   :Ocount   ) ;
```

（実行結果）

```
        SELECT   count(*)   from   emp_table   WHERE   empno > 0
        cnt= 4
        -   -   -   -     -   -   -   -     -
        SELECT   *   from   emp_table   WHERE   empno > 0
        番号：1 名前：KOZUE
        番号：2 名前：IKURA
        番号：3 名前：MINKA
        番号：4 名前：HATSUNEPL/SQL プロシージャが正常に完了しました。

SQL> print   Ocount
        OCOUNT
        ----------
                4
```

【SQL 文（条件）を変更しての再度のプロシージャの呼出し】

```
SQL>   EXECUTE   get_count (' WHERE   ename = "KOZUE"   ' ,   :Ocount   ) ;
       /*   「"」は、シングルクォーテーション２個   ~~ダブルクォーテーション~~     */
```

（ 補足 ）シングルクォーテーション「'」を文字列データとして扱ってほしい場合には、
シングルクォーテーション「'」を２個並べて、連結させる。

```
                例）wk_string :=   ''   ||   '愛川 こずえ'   ||   ''   ;
                    wk_string :=   '''愛川 こずえ'''   ;
```

（実行結果）

SELECT count(*) from emp_table WHERE ename = 'KOZUE'
cnt= 1
PL/SQL プロシージャが正常に完了しました。
- - - - - - - - -
SELECT * from emp_table WHERE ename = 'KOZUE'
番号：1 名前：KOZUE

SQL> print Ocount
OCOUNT

1

参考)

複数個の列へテーブルのレコード型である ROWTYPE を使用しないでの対応

%ROWTYPE を使用しなかったときのレコード変数の定義方法

```
TYPE   emp_record_type   IS   RECORD   (
                                          empno   NUMBER   ,
                                          ename   VARCHAR2(14) ,
                                          deptno   NUMBER
                                        )   ;

emp_record          emp_record_type     ;
```

もしくは、

テーブルのレコード型である IS　RECORD 句を使用しない場合

列の値を保存するための**配列変数**を、列の個数分用意する

```
TYPE   wk_event_type      IS TABLE OF
                          V$ACTIVE_SESSION_HISTORY.event%TYPE ;
wk_event                  wk_event_type ;

TYPE   wk_event_count_type        IS TABLE OF   NUMBER   ;
wk_event_count            wk_event_count_type ;

TYPE   wk_sample_time_type        IS TABLE OF   VARCHAR2( 32 ) ;
wk_sample_time            wk_sample_time_type ;
```

用意した複数個の配列変数を、BULK　COLLECT　INTO 句の後に、並べる

```
EXECUTE   IMMEDIATE   wk_sql   BULK   COLLECT   INTO
                          wk_sample_time ,   wk_event ,   wk_event_count    ;
```

【 例２　全件ループ処理 】

```
DECLARE
    where_statement   VARCHAR2(1024)   ;              --  WHERE 文の条件内容
    emp_record      emp_table%ROWTYPE   ;              --  レコード型変数の定義
   sql_statement   VARCHAR2(1024)   ;
   /*    カーソルの定義      */
    TYPE   select_cursor_type   IS   REF   CURSOR   ;     --  タイプ定義（カーソル）
    cursor1   select_cursor_type   ;                    --  定義タイプに対する変数定義

BEGIN
    where_statement   :=   'WHERE   dept_no   >   0   ';

    /*   動的 SQL 文の組み立て   */
   sql_statement   :=   'SELECT   *   FROM   emp_table'   ||   where_statement;

    /*   動的 SQL 文に対するカーソルのオープン操作   */
    OPEN   cursor1   FOR   sql_statement   ;

    LOOP
        FETCH   cursor1   INTO   emp_record   ;   --   フェッチ
        EXIT   WHEN   cursor1%NOTFOUND   ;   --   フェッチの終了判断
        /*    データが存在した時の処理     */
            /*    読込んだデータは、「emp_record.  列名」の中にセットされている     */
        DBMS_OUTPUT.PUT_LINE( ' 番号 : ' || emp_record.empno || ' 名前 : ' ||
                                        emp_record.ename );

    END   LOOP   ;

    CLOSE   cursor1   ;                                 --  カーソル・クローズ

END   ;
/
```

（ 補足 ）シングルクォーテーション「'」を文字列データとして扱ってほしい場合には、シングルクォーテーション「'」を２個並べて、連結させる。

```
        例）wk_string := ''   ||   '愛川  こずえ'   ||   ''   ;
            wk_string := '''愛川  こずえ'''   ;
```

――― 動的 SQL 文の実行 ―――　　(PL/SQL)

【パラメータ付きカーソル】

カーソル宣言時にパラメータ付で指定して、実行時に SQL 文を変更して実行するプロシージャの作成方法

【 構 文 】

CURSOR　カーソル名 (パラメータ名　データ型 ［ , パラメータ名　データ型］)
IS　SELECT 文・・・・・・ ;
レコード型変数名　テーブル名%LOWTYPE ;

使用例)

```
DECLARE
  CURSOR  cur_emp_table ( p_empno  NUMBER  DEFAULT  1 )
  IS  SELECT  *  FROM  emp_table  WHERE  empno  =  p_empno ;
  r_emp  emp_table%ROWTYPE ;

BEGIN
  OPEN    cur_emp_table( 3 ) ;
  FETCH  cur_emp_table  INTO  r_emp  ;
           --  1 回のフェッチのみ
  DBMS_OUTPUT.PUT_LINE ('番号 : ' || TO_CHAR (r_emp.empno ) ||
                          '     名前 : ' || r_emp.ename  );
  CLOSE  cur_emp_table  ;

  OPEN    cur_emp_table( ) ;            --  パラメータ値省略での呼出し
  FETCH  cur_emp_table  INTO  r_emp  ;
           --  1 回のフェッチのみ
  DBMS_OUTPUT.PUT_LINE ('番号 : ' || TO_CHAR (r_emp.empno ) ||
                          '     名前 : ' || r_emp.ename  );
  CLOSE  cur_emp_table  ;

END  ;

/
      /*      /は、プロシージャの実行のため      */
```