# 【カーソル FETCH 文を使用した~~複数~~レコード処理】
# SELECT によるテーブルの読込み　カーソルの使用(PL/SQL)

レコード Select のための制御手順

１．カーソルの宣言（ DECLARE　CURSOR ）
　　↓
２．カーソルのオープン（ OPEN ）
　　↓
３．カーソルを使ったフェッチ（ FETCH ）・・・・・必要件数ループ
　　↓
４．カーソルのクローズ（ CLOSE ）

手順．１
カーソルの宣言（ DECLARE　CURSOR ）
　　CURSOR　カーソル名　[ (パラメータ１　[　,　パラメータ２　・・・　]　) ]
　　[　RETURN　戻り値のデータ型　]
　　IS　Select・・・・・文　;

手順．２
カーソルのオープン（ OPEN ）
　　OPEN　カーソル名　;

手順．３
カーソルを使ったフェッチ（ FETCH ）
　　FETCH　カーソル名　INTO　レコード型変数名　;

手順．４
カーソルのクローズ（ CLOSE ）
　　CLOSE　カーソル名　;

※ FETCH に伴う操作結果の判断 および、戻り属性の使用

| | |
|---|---|
| カーソル名%FOUND | フェッチの成功判断 |
| カーソル名%NOTFOUND | フェッチの失敗判断 |
| カーソル名%ISOPEN | カーソルのオープン済かの判断 |
| カーソル名%ROWCOUNT | FETCH の実施回数のカウント<br>（１回ごとの FETCH 操作で加算される）<br>オープン時に件数がセットされる訳ではない |

（ 使用例 ）

```
DECLARE
    p_dept   emp_table.dept_no%TYPE  ;          --  WHERE 文中の値指定用変数定義
    emp_record   emp_table%ROWTYPE  ;           --  レコード型変数の定義

   /*    カーソルの定義      */
    CURSOR   c_emp   IS
    SELECT   *   FROM   emp_table   WHERE   dept_no   =   p_dept   ;

BEGIN
    p_dept   :=   10   ;
    OPEN   c_emp   ;                                --  カーソル・オープン
    FETCH   c_emp   INTO   emp_record   ;           --  フェッチ

    IF   カーソル名%FOUND   THEN                    --  フェッチの成功判断
       DBMS_OUTPUT.PUT_LINE(' /*    フェッチ成功時の処理    */'  );
    END IF;

    IF   カーソル名%NOTFOUND      THEN          --  フェッチの失敗判断
       DBMS_OUTPUT.PUT_LINE(' /*    フェッチ失敗時の処理    */'  );
    END IF;

    IF   NOT   カーソル名%ISOPEN   THEN          --  カーソルのオープン済かの判断
       DBMS_OUTPUT.PUT_LINE(' /*    カーソルが OPEN していなかった時の処理
       */'  );
    END IF;

    IF   カーソル名%ROWCOUNT   >   1000   THEN  --  FETCH 実施回数のカウント
       DBMS_OUTPUT.PUT_LINE(' /*    FETCH が、1000 回を超えた時の処理
       */'  );
    END IF;

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

END   ;
/
```

参考)

%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     ;
```

【FETCH とループ文を使用した複数レコード処理】
【カーソル FOR ループ文を使用した複数レコード処理】
SELECT による処理における対象データの全件処理ループ（PL/SQL）

FETCH とループ文（ 例 1）

```
DECLARE
    emp_record  emp_table%ROWTYPE  ;          --  レコード型変数の定義
   /*    カーソルの定義      */
    CURSOR  c_emp  IS
    SELECT  *  FROM  emp_table ;

BEGIN
    OPEN  c_emp  ;                             --  カーソル・オープン
    LOOP
        FETCH  c_emp  INTO  emp_record  ;      --  フェッチ
        EXIT  WHEN  c_emp%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  c_emp  ;                            --  カーソル・クローズ

END  ;
/
```

参考)

%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     ;
```

## カーソル FOR ループ文（ 例 ２）

```
DECLARE
    emp_record  emp_table%ROWTYPE ;          --  レコード型変数の定義
   /*    カーソルの定義      */
    CURSOR  c_emp  IS
    SELECT  *  FROM  emp_table  ;

BEGIN
    OPEN  c_emp ;                            --  カーソル・オープン
    FOR  r_emp  IN  c_emp  LOOP
     /*  カーソル FOR ループ文を使うと、FOR 以下の文字がレコード型（%
         ROWTYPE）の変数定義として宣言されたことになり、変数として使える */
     /*  r_emp がレコード型変数として自動定義され、レコードの値がループごとに
         セットされる */

    LOOP
        FETCH  c_emp  INTO  emp_record ;     -- フェッチ
        EXIT  WHEN  c_emp%NOTFOUND ;     -- フェッチの終了判断
        /*   データが存在した時の処理   */

        /*  読込んだレコードの値の使い方  */
              -- r_emp.  列名
        DBMS_OUTPUT.PUT_LINE( ' 番 号 : ' || r_emp.empno || ' 名 前 : ' ||
                                        r_emp.ename );

    END  LOOP  ;
    CLOSE  c_emp ;                           --  カーソル・クローズ
EXCEPTION
    WHEN  エラー条件  THEN
       --    エラー処理
END  ;
```

【注意（デメリット）】

FOR IN LOOP 文を使用した場合には、この例外処理部分（EXCEPTION 句）の中では、このカーソル変数 r_emp は、変数のスコープ（有効範囲）外になり使用できないことになる。

FETCH  c_emp  INTO  emp_record の場合は、外のに DECLARE で定義されているので変数のスコープ（有効範囲）内であり使用することができる。

## カーソル FOR ループ文

（ 例 ３ ） 複数テーブルの読込み多重ループ

```
DECLARE
    p_dept   emp_table. dept_no%TYPE  ;          --  WHERE 文中の値指定用変数定義
   /*    カーソルの定義      */
    CURSOR   c_dept   IS
    SELECT   *   FROM   dept_table   ORDER   BY   dept_no   ;
    CURSOR   c_emp   IS
    SELECT   *   FROM   emp_table   WHERE   dept_no   =   p_dept   ;

BEGIN
    FOR   r_dept   IN   c_dept   LOOP
     /*   r_dept がレコード型変数として自動定義され、レコードの値がループごとに
          セットされる  */
          p_dept   :=   r_dept.dept_no   ;    --   WHERE 文中の検索条件値を変数にセット

          FOR   r_emp   IN   c_emp   LOOP
                /*   r_emp がレコード型変数として自動定義され、レコードの値がルー
                     プごとにセットされる  */
               /*    フェッチしたレコードに対する処理     */
          END   LOOP   ;
    END   LOOP   ;
END   ;
```