エラー発生時の例外処理（PL/SQL）
例外エラー・コード（PL/SQL）
エラー処理ルーチン（例外ハンドラ）（PL/SQL）

PL/SQL ブロック

```
DECLARE

    ―― 変数の定義 ――

BEGIN

    ―― 処理内容の記述 ――

EXCEPTION

    ―― 例外ハンドラの記述 ――

    WHEN   例外例   ［OR   例外例 ］   THEN
        処理 ;

（ 例 ）
    WHEN   NO_DATA_FOUND   THEN
        DBMS_OUTPUT.PUT_LINE ('レコードが返されません') ;
    WHEN   LOGIN_DENIED   THEN
        DBMS_OUTPUT.PUT_LINE ('不正なログオン行為が発生') ;
    WHEN   OTHERS   THEN
        DBMS_OUTPUT.PUT_LINE ('その他エラーが発生しました') ;

END   ;
```

SQL エラー対処コーディング方法

例外エラー・ルーチンでのエラー・コードとエラー・メッセージの受け取り

関　数　（変数）	機　　能
SQLCODE	エラー・コードが返る
SQLERRM	エラー・メッセージが返る

※ この２つの用意された変数に、エラー内容をシステムでセットして、処理側に知らせてくれる

WHEN 文で判断する条件内容は、次ページ表の例外名を使って行う

```
EXCEPTION
    WHEN  OTHERS  THEN
    DBMS_OUTPUT.PUT_LINE(  'エラーが発生しました。ORA-'  ||
        TO_CHAR(  SQLCODE,  'fm00000'  )  ||  '  '  ||  SQLERRM )  ;
```

出力内容
エラーが発生しました。ORA-00600　　＊＊＊＊メッセージ＊＊＊

【例外エラー・ルーチンでのカーソル使用】
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。

ソースコードは、６ページ後

例外が発生しても、処理を継続させる場合は、BEGIN 節の中に BEGIN 節を入れ子にして記述して、下位の BEGIN 節の中で EXCEPTION 処理を行うと、EXCEPTION 節を処理した後、上位の BEGIN 句の中のステートメントは継続して処理される

```
DECLARE
    emp_record   emp_table%ROWTYPE  ;          --  レコード型変数の定義
BEGIN
    BEGIN
        --  下位の BEGIN 句での処理
       SELECT  enp_name  INTO  emp_record.enp_name  FROM  emp_table
              WHERE  emp_no  = 9999 ;

        --  下位の BEGIN 句での処理（続き）
        /*  この部分は、SELECT で例外エラーが発生した場合には、下位の
        EXCEPTION 節に跳んで、その後の処理は、放棄される  */


    EXCEPTION        --  下位の BEGIN に対する EXCEPTION 節
        /*  ―― 例外ハンドラの記述 ――  */
        WHEN  NO_DATA_FOUND  THEN
        DBMS_OUTPUT.PUT_LINE( 'データがありませんでした'  ) ;
    END  ;

   /*    上位の BEGIN 句での処理  この部分は、下位の EXCEPTION の処理後
          に続けて実行される  */

EXCEPTION   --  上位の BEGIN に対する EXCEPTION 節
    /*  ―― 例外ハンドラの記述 ――  */

END  ;
```

EXCEPTION エラー・ブロックの記述が無かったり、該当するエラーの WHEN 句が無かった場合の対応は、呼出し元へエラーがそのまま戻されます
SQL*PLUS や SQL Developer であれば、エラー・コードやエラー・メッセージが表示されます
しかし、Java や PHP のプログラムでは、呼出したプログラムの呼出し命令（ステートメント）がエラー終了することになります

EXCEPTION エラー・ブロックの記述しエラー対応した後、呼出し元のプログラム（SQL*PLUS、SQL Developer、Java、PHP）にもエラーを返す（伝達する）方法は、RAISE ; ステートメントを記述します

```
EXCEPTION          --  例外ハンドラの記述
    WHEN  NOT_FOUND  THEN
        DBMS_OUTPUT.PUT_LINE ('レコードが無い');
        IF  c_emp%ISOPEN  THEN
            CLOSE  c_emp  ;                 --  ファイルのクローズ
        END  IF  ;
        RAISE  ;                            --  呼出し元へのエラー伝達
END  ;
```

【Oracle エラー一覧】

一般的な例外エラー・コード　一覧

エラー・コード	例　外　名	発　生　原　因
ORA-00001	DUP_VAL_ON_INDEX	一意制約違反の行為
ORA-00051	TIMEOUT_ON_RESOURCE	リソース獲得の待機において、タイムアウト発生
ORA-01001	INVALID_CURSOR	カーソルが無い状態でのカーソル操作
ORA-01012	NOT_LOGGED_ON	Oracle に未接続での操作
ORA-01017	LOGIN_DENIED	不正なユーザーのログオン行為
ORA-01403	NO_DATA_FOUND	返すレコードが無い
ORA-01410	SYS_INVALID_ROWID	誤った ROWID 値の指定
ORA-01422	TOO_MANY_ROWS	Select INTO 文で複数レコードが発生
ORA-01476	ZERO_DIVIDE	ゼロでの割り算実行
ORA-01722	INVALID_NUMER	数字でない文字列の数値変換
ORA-06500	STORAGE_ERROR	メモリ不足 もしくは、メモリ故障の発生
ORA-06501	PROGRAM_ERROR	PL/SQL プロシージャに内部的に問題が発生した
ORA-06502	VALUE_ERROR	算術エラー、変換エラー サイズ制約エラー（桁あふれ）
ORA-06504	ROW_TYPE_MISMATCH	カーソルを使用したレコード呼出し処理において、実レコードと読込用変数でデータ型の不一致が発生した
ORA-06511	CURSOR_ALREADY_OPEN	オープン済カーソルへの重複オープン
ORA-06530	ACCESS_INTO_NULL	プログラムが未初期化のオブジェクトの属性に値を代入しようとした
ORA-06531	COLLECTION_IS_NULL	プログラムが未初期化のネストした表や VARRY の要素に値を代入しようとした
ORA-06532	SUBSCRIPT_OUTSIDE_LIMIT	有効範囲外の値指定
ORA-06533	SUBSCRIPT_BEYOND_COUNT	コレクション中の要素数より大きい添字番号の指定
ORA-06592	CASE_NOT_FOUND	CASE 文のどの WHEN にも一致せず、ELSE 句もない場合
ORA-30625	SELF_IS_NULL	オブジェクト型のインスタンスが初期化されていないのに、メソッドの呼出しを行った

Oracle から例外エラー割り込みは発生するが、
例外名が割当てられていないエラーへの対応方法

PL/SQL ブロック

```
DECLARE
        ―― 変数の定義 ――
    undefined_error   EXCEPTION ;        --   例外名の宣言
    /*    エラー・コードに対する例外名の割当て     */
    PRAGMA EXCEPTION_INIT( undefined_error,   -600 ) ;

BEGIN

    ―― 処理内容の記述 ――

EXCEPTION
    ―― 例外ハンドラの記述 ――
    /*    割当てた例外名に対するエラー・ルーチンの作成     */
    WHEN   undefined_error   THEN
        DBMS_OUTPUT.PUT_LINE ('特殊エラーが発生しました') ;


    WHEN   OTHERS   THEN
        DBMS_OUTPUT.PUT_LINE ('その他エラーが発生しました') ;

END   ;
```

プログラム中で、ユーザーが独自に発生させるエラー割り込みの方法

エラー番号は、-20000～20999 の間を指定のこと

PL/SQL ブロック

```
DECLARE
        ―― 変数の定義 ――
    commission_exception   EXCEPTION ;  --  例外名の宣言
    /*    エラー・コードの宣言と対応する例外名の割当て     */
    PRAGMA EXCEPTION_INIT( commission_exception, -20100 ) ;

BEGIN
    ―― 処理内容の記述 ――
    IF   user_data   < 0   THEN
    /*   ユーザーによるエラー割り込みの発生指示   */
        RAISE   commission_exception   ;
    もしくは、
        RISE_APPLICATION_ERROR ( エラー・コード ,  'エラー・
            メッセージの内容' ) ;
      --  エラー・コードに対応した例外名(WHEN)でのエラーが発生する

EXCEPTION
    ―― 例外ハンドラの記述 ――
    /*    割当てた例外名に対するエラー・ルーチンの作成     */
    WHEN   commission_exception   THEN
        DBMS_OUTPUT.PUT_LINE  ( '特殊エラーが発生しました' ) ;

    WHEN   OTHERS   THEN
        DBMS_OUTPUT.PUT_LINE  ('その他エラーが発生しました') ;

END  ;
```

RISE_APPLICATION_ERROR では、DBMS_STANDARD パッケージが使われるので、DECLARE での「 例外名 EXCEPTION 句 」の記述無しでも、ユーザー定義（例外名の宣言）のエラーをプログラム側からトリガー起動できます

ただし、この場合のエラー処理は、例外名が割当てられていないので、WHEN OTHERS THEN 句で行います

RISE_APPLICATION_ERROR (エラー・コード , 'エラー・メッセージの内容') ;

※ エラー・コード 20000～20999

また、エラー・メッセージの中に、レコードのデータを使うことは、可能です

（例） r_empo.ename || 'エラーが発生したレコードの名前です'

EXCEPTION 節の中でのカーソル使用　例

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。
FETC ループ文でしか記述できない。

```
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
   /*    例外名の宣言     */
   user_define_error   EXCEPTION ;
   /*    エラー・コードの宣言と対応する例外名の割当て     */
   PRAGMA EXCEPTION_INIT( user_define_error, -20100 ) ;
   /*    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
    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     */
        IF   emp_record.user_data   < 0   THEN
        /*   ユーザーによるエラー割り込みの発生指示   */
            RAISE   user_define_error   ;
    END   LOOP   ;

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

EXCEPTION
    /*    —— 例外ハンドラの記述 ——     */
    /*    割当てた例外名に対するエラー・ルーチンの作成     */
    WHEN   user_define_error   THEN
        DBMS_OUTPUT.PUT_LINE ( 'USE_DATA が、負の値でした' ) ;
        IF   c_emp%ISOPEN   THEN
            CLOSE   c_emp   ;                     --  ファイルのクローズ
        END   IF   ;

END   ;
```