CONTROL DE ERRORES. EXCEPCIONES

- En PL/SQl se denomina excepción a cualquier identificador que nos permita manejra un error que se produzca durante la ejecución del programa, las excepciones pueden ser definidas internamente por el sistema o definidas por el usuario.

- El tratamiento de excepciones de PL/SQL clarifica el código del programa al separarlo del resto del proceso, definiendo las acciones a realizar en caso de producirse algún error.

- **Cuando se produce un error**, se activa una excepción y la ejecución normal del programa se detiene y **finaliza el bloque donde se produzca el error**, pasándose el control a la sección de excepciones del bloque.

- En la parte de excepciones del programa existirán distintas rutinas, cada una dedicada a procesar una o varias excepciones.

- Cada rutina comienza con una cláusula WHEN donde se especifica la excepción o excepciones a las que se aplicará dicha rutina y **después de ejecutarse la rutina correspondiente a un error, el proceso continúa con la siguiente sentencia del bloque contenedor**.

- Si queremos que una misma rutina trate varias excepciones, detrás de la palabra WHEN pondremos los nombres de esas excepciones, separados por el operador relacional OR.

Sintaxis:

```
EXCEPTION
WHEN exception1 [OR exception2...]THEN

        Instrucciones;
WHEN  exception2 [OR exception2...]THEN
        Instrucciones2;
WHEN OTHERS
        Instrucciones3;
```

- La palabra EXCEPTION es obligatoria.
- Las excepciones no pueden aparecer en asignaciones de variables ni en sentencias SQL
- Como mucho podremos tener una claúsula OTHERS y en ella se filtran todos los errores que no tengan su correspondiente manejador
- Podríamos tener en una zona de excepciones como único manejador la clausula OTHERS de tal modo que cualquier error en ejecución se controla dentro de esta zona.
- Un error producido en la ejecución de la sección de excepciones no puede ser tratado en ella. En este caso, se termina la ejecución del bloque con un error que se intentará tratar en algunos de los bloques externos.
- En la declaración de variables pueden producirse excepciones al realizar inicializaciones incorrectas. Las excepciones producidas en la parte de declaraciones no pueden ser tratadas en el bloque, dando lugar a la finalización de la ejecución del bloque con un error no tratado.

Ejemplo:

```
DECLARE
maximo CONSTANT NUMBER(4) := 10000;
```

→ Finalización del programa con error.

Ejemplo:

```
DECLARE
  comunidad NUMBER(7);

BEGIN

  BEGIN
    SELECT pts_comunidad INTO comunidad
      FROM vecinos
      WHERE apellido1 LIKE 'SALVADOR';

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      SELECT pts_comunidad INTO comunidad
        FROM vecinos
        WHERE apellido1 LIKE 'CASTRILLO';

  END;

  IF comunidad < 7000 THEN
    comunidad := comunidad + 1000;
  END IF;

  INSERT INTO vecinos VALUES (8, 'A', 'ROBERTO', 'LOSADA',
    '22-JUN-99', comunidad, comunidad/166.386);

  COMMIT;

END;
/
```

- Si en la parte de excepciones de un bloque no existe rutina para el tratamiento de un determinado error, o si no existe sección de excepciones, dicho error pasa sin tratar al bloque contenedor, que intentará tratarlo en su sección de excepciones. De esta forma, un error no tratado va pasando de bloque en bloque hasta que es tratado o hasta que termina el programa con error.

- Existen dos tipos de excepciones:

  Excepciones predefinidas.- Son activadas por el Servidor Oracle de forma automática cuando se produce el error correspondiente.

  Excepciones definidas por el usuario.- Deben de activarse explícitamente con la sentencia RAISE.

  Excepciones predefinidas

  CURSOR_ALREADY_OPEN.- Se intenta abrir un cursor ya abierto.

  DUP_VAL_ON_INDEX.- Intento de insertar o actualizar, violando la condición de unicidad de índice.

  INVALID_CURSOR.- Operación ilegal sobre un cursor.

  INVALID_NUMBER.- En una sentencia SQL se intentó realizar una conversión de cadena de caracteres a número y la cadena contenía caracteres no numéricos.

  LOGIN_DENIED.- Se intentó iniciar una sesión de base de datos con nombre de usuario o password no válidos.

  NO_DATA_FOUND.- Una sentencia SELECT ... INTO no devuelve ninguna fila.

  NOT_LOGGED_ON.- PL/SQL realiza una llamada a ORACLE y no existe conexión.

  PROGRAM_ERROR.- Error interno de PL/SQL.

  STORAGE_ERROR.- Se produce un error de memoria.
  TIMEOUT_ON_RESOURCE.- Se termina el tiempo de espera por un recurso.

  TOO_MANY_ROWS.- Una sentencia SELECT ... INTO devuelve más de una fila.

  VALUE_ERROR .- Se produjo un error aritmético o de conversión, un truncamiento o la violación de una restricción.

  ZERO_DIVIDE.- Se intentó dividir por cero.

  OTHERS.- En esta rutina se tratará cualquier excepción para la que no exista tratamiento específico.

  Hay alguna más pero estas son las más utilizadas y tenemos que tener en cuenta que no es necesario declararlas en la sección DECLARE.

Excepciones definidas por el usuario

El programador puede definir sus propias excepciones, las cuales deben ser declaradas, llamadas y activadas. Los formatos a utilizar son:

```
DECLARE
  ...
  nom_excep_usu EXCEPTION; → Declaración de excepción
  ...
BEGIN
  ...
  RAISE nom_excep_usu; → Llamada a la excepción
  ...
EXCEPTION
  ...
  WHEN nom_excep_usu THEN → Activar sentencias ejecutar
  sentencias;
  ...
END;
/
```

Pasos a realizar:

Se requieren tres pasos para trabajar con excepciones de usuario:

Definición: se realiza en la zona de DECLARE con el siguiente formato:
nombre_excepción EXCEPTION

Disparar o levantar la excepción mediante la orden raise: RAISE ;

Tratar la excepción en el apartado EXCEPTION: WHEN THEN ;

Ejemplo: De la tabla VECINOS incrementar en 1000 pesetas su columna de PTS_COMUNIDAD y la cantidad correspondiente a la columna de valor en euros, salvo para las filas de fecha llegada nula, que se modificará únicamente la columna FECHA_LLEGADA con el valor de la fecha de sistema.

```
DECLARE
  fecha_nula EXCEPTION;
  fecha_no_nula EXCEPTION;
  CURSOR c_vecinos IS
    SELECT * FROM vecinos FOR UPDATE;
BEGIN
  FOR i IN c_vecinos LOOP
    BEGIN
      IF i.fecha_llegada IS NULL THEN
        RAISE fecha_nula;
      END IF;
      RAISE fecha_no_nula;
```

```
    EXCEPTION
      WHEN fecha_nula THEN
        UPDATE vecinos SET fecha_llegada = SYSDATE
          WHERE CURRENT OF c_vecinos;
      WHEN fecha_no_nula THEN
        i.pts_comunidad := i.pts_comunidad + 1000;
        UPDATE vecinos SET
          pts_comunidad = i.pts_comunidad,
          e_comunidad = i.pts_comunidad / 166.386
          WHERE CURRENT OF c_vecinos;
    END;
  END LOOP;
  COMMIT;
END;
/
```

Otras excepciones

Existen otros errores internos de Oracle que no tienen asignada una excepción, sino un código de error y un mensaje, a los que se accede mediante funciones SQLCODE y SQLERRM.

Cuando se produce un error de estos se trasfiere directamente el control a la sección EXCEPTION donde se tratara el error en la clausula WHEN OTHERS de la siguiente forma:

WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Error'||SQLCODE||SQLERRM.)

RAISE_APPLICATION_ERROR

En el paquete DBMS_STANDARD se incluye un procedimiento llamado RAISE_APPLICATION_ERROR que nos sirve para levantar errores y definir mensajes de error. Su formato es el siguiente:
RAISE_APPLICATION_ERROR(numero_error,mensaje_error);

Es importante saber que el numero de error esta comprendido entre -20000 y -20999 y el mensaje es una cadena de caracteres de hasta 512 bytes.
Este procedimiento crea una excepción que solo puede ser tratada en WHEN OTHERS.

Ejemplo:

```
CREATE or REPLACE PROCEDURE subir_horas (emple NUMBER, horas_subir NUMBER)
IS
  horas_actuales NUMBER;
BEGIN
  Select horas into horas_actuales from empleados where id_empleado=emple;
  if horas_actuales is NULL then
    RAISE_APPLICATION_ERROR(-20010,'No tiene horas');
  else
```

```
                update empleados set horas=horas_actuales + horas_subir where
id_empleado=emple;
   end if;
End subir_horas;
```