Introducción a PL/SQL. Parte II.

Ayuda para prácticas. 2º cuatrimestre.

1.- Tratamiento de errores. Excepciones

PL/SQL implementa los mecanismos de tratamiento de errores mediante excepciones y gestores de excepciones. Podemos asociar las excepciones a los errores de Oracle o a errores definidos por el usuario. Cuando se produce un error, se genera una excepción. Cuando esto sucede, el control pasa al gestor de excepciones, que es una sección independiente del programa.

Declaración de Excepciones.

Las excepciones se declaran en la sección declarativa de un bloque, se generan en la sección ejecutable y se tratan en la sección de excepciones. Existen dos tipos: *las definidas por el usuario y las predefinidas.*

Excepciones definidas por el usuario:

Se declaran en la sección declarativa de un bloque PL/SQL. Tienen un tipo asociado (EXCEPTION) y un ámbito. Ejemplo:

```
DECLARE
 e_DemasiadosEstudiantes EXCEPTION;
```

El ámbito de una excepción es igual que el de cualquier otra variable o cursor en la misma sección declarativa.

Excepciones predefinidas:

Excepción	Descripción
TOO_MANY_ROWS	Hay más de una fila que corresponde a una orden SELECT..INTO
VALUE_ERROR	Error de truncamiento , aritmético o de conversión
ZERO_DIVIDE	División por cero
COLLECTION_IS_NULL	Se ha intentado aplicar métodos de conversión distintos de EXISTS a una tabla con valor NULL
CURSOR_ALREADY_OPEN	Se ha intentado abrir un cursos que ya estaba abierto
DUP_VAL_ON_INDEX	Violación de una restricción de unicidad.
INVALID_CURSOR	Operación ilegal con un cursor.
INVALID_NUMBER	Falló la conversión a un número.
LOGIN_DENIED	Nombre de usuario o contraseña no valido.
NO_DATA_FOUND	No se ha encontrado ningún dato.
NOT_LOGGED_ON	No existe conexión con Oracle.
PROGRAM_ERROR	Error interno PL/SQL
ACCESS_INTO_NULL	Se ha intentado asignar valores a los atributos de un Objeto que tiene el valor NULL
ROWTYPE_MISMATCH	Una variable de cursor del host y una variable de cursor de PL/SQL tienen tipos de filas incompatibles
SELF_IS_NULL	El programa ha intentado llamar a un método MEMBER con el primer parámetro null
STORAGE_ERROR	Error interno PL/SQL se queda sin memoria
SUBSCRIPT_BEYOND_COUNT	Una referencia a una tabla anidada o índice de varray mayor que el número de elementos de la colección

SUBSCRIPT_OUTSIDE_LIMIT	Una referencia a una tabla anidada o índice de varray fuera del rango declarado
SYS_INVALID_ROWID	Fallo al convertir un string de caracteres a un tipo ROWID

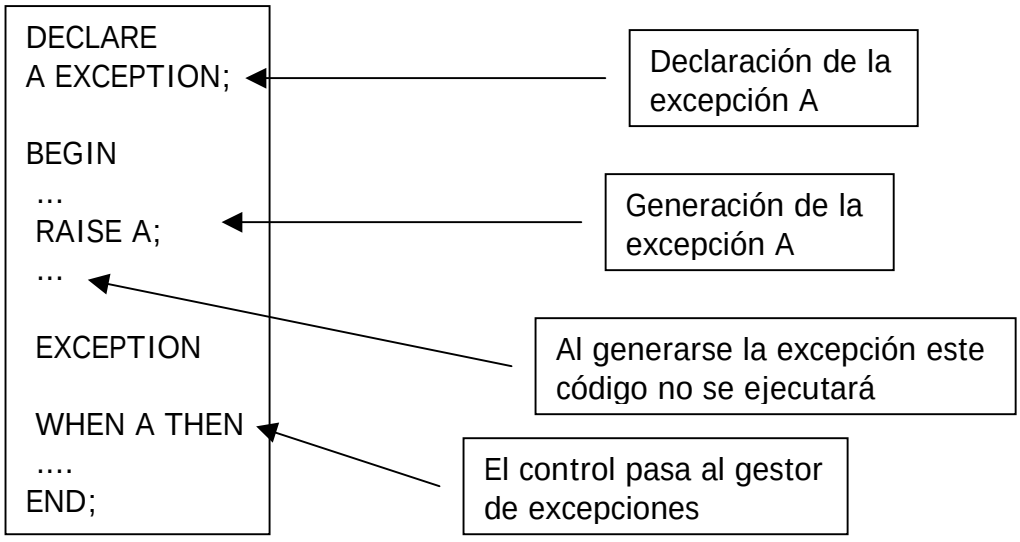
Generación de excepciones:

Cuando se produce un error asociado con una excepción, dicha excepción es generada. Las excepciones definidas por el usuario se generan explícitamente mediante la orden RAISE.

Ejemplo:

```
DECLARE
  e_MuchosEstudiantes EXCEPTION;  -- Declaración de la excepción
  v_CurrentStudents NUMBER(3);  -- Número de estudiantes en HIS-101
  v_MaxStudents NUMBER(3);      -- Máximo número de estudiantes permitido en  HIS-101
BEGIN
  /* Haya el número actual de estudiantes y el máximo permitido. */
  SELECT current_students, max_students
    INTO v_CurrentStudents, v_MaxStudents
    FROM classes
    WHERE department = 'HIS' AND course = 101;
  /* Comprueba el número de estudiantes en ese curso */
  IF v_CurrentStudents > v_MaxStudents THEN
    /*Demasiados estudiantes—lanza la excepción. */
    RAISE e_MuchosEstudiantes;
  END IF;
EXCEPTION
  WHEN e_Muchos estudiantes THEN
    /* Gestor de excepciones, captura la excepción y ejecuta el código asociado */
    INSERT INTO log_table (info) VALUES ('History 101 has ' || v_CurrentStudents ||
      'students: max allowed is ' || v_MaxStudents);
WHEN OTHERS THEN
    /* Este gestor se ejecutará para cualquier otro error */
    INSERT INTO log_table (info) VALUES ('Another error occurred');
END;
```

Si no hay sección de excepciones en el bloque la excepción se propaga al bloque de nivel superior:



Gestión de excepciones:

La sección de excepciones está compuesta por *gestores* para las distintas excepciones. Un gestor de excepciones contiene el código que se ejecutará cuando ocurra el error asociado con la excepción y ésta sea generada.

Cada gestor de excepciones está formado por la cláusula WHEN la excepción a capturar (pueden agruparse varias mediante un OR) y las ordenes que se ejecutarán cuando la excepción sea capturada.

El gestor OTHERS capturará cualquier excepción generada. Debe siempre ser el último gestor de un bloque. Es una buena práctica de programación el definir un gestor OTHERS en el nnivel superior del programa para asegurarse que ningún error queda sin capturar. Para saber que error provoco la excepción dentro del gestor OTHERS podemos usar las funciones SQLCODE y SQLERM para obtener el código del error y el mensaje asociado.

2.- Procedimientos y funciones

Los procedimientos y funciones en PL/SQL son muy similares a los de cualquier lenguaje de programación.

Ejemplo procedimiento que inserta un estudiante en la base de datos:

```
CREATE OR REPLACE PROCEDURE AgregaEstud (
  p_FirstName  students.first_name%TYPE,
  p_LastName   students.last_name%TYPE,
  p_Major      students.major%TYPE) AS
BEGIN
  -- Inserta una nueva fila en la tabla students
  INSERT INTO students (ID, first_name, last_name,
                  major, current_credits)
    VALUES (student_sequence.nextval, p_FirstName, p_LastName,
          p_Major, 0);

  COMMIT;
END AgregaStud;
/
```

Una vez creado este procedimiento podemos llamarlo desde otro bloque PL/SQL:

```
BEGIN
  AgregaStud('David', 'López', 'Musica');
END;
```

Notas:

- Cuando se crea un procedimiento con la orden CREATE OR REPLACE PROCEDURE, éste se compila en primer lugar y queda almacenado en la Base de datos de forma compilada. El código compilado puede ser posteriormente utilizado por cualquier bloque PL/SQL
- A un procedimiento pueden pasársele parámetros en la llamada.
- Un procedimiento es un bloque PL/SQL, con una sección declarativa, una sección ejecutable y una sección de manejo de excepciones. al igual que en los bloques anónimos solo la sección ejecutable es obligatoria.

- Para modificar unprocedimiento crado debemos reemplazarlo por el nuevo volviendo a compilarlo añadiendo las palabras clave OR REPLACE. Podemos eliminar un procedimientro mediante la orden DROP PROCEDURE.

La sintaxis para la orden CREATE OR REPLACE PROCEDURE es:

CREATE [OR REPLACE] PROCEDURE *nombre_procedimiento*
[(*argumento*) [{IN | OUT | IN OUT}] *tipo*,
...
[(*argumento*) [{IN | OUT | IN OUT}] *tipo*)] {IS|AS}

cuerpo procedimiento

Parámetros:

Los parámetros formales de un procedimiento pueden ser de tres modos: IN, OUT o IN OUT. Si no se especifica el modo de un parámetro toma, por defecto, el modo IN.

Modo	Descripción
IN	El valor del parámetro real se pasa al procedimiento cuando éste es llamado. Dentro del procedimiento el parámetro formal se considera como de ***sólo lectura***, y no puede ser modificado. Cuando termina el procedimiento, y se devuelve el control al entorno que realizó la llamada el parámetro real no ha sufrido cambios
OUT	Se ignora cualquier valor que tenga el parámetro real cuando se llama al procedimiento. Dentro del procedimiento, el parámetro formal se considera como de ***sólo escritura***; no puede ser leído, sino que tan sólo pueden asignársele valores. Cuando termina el procedimiento y se devuelve el control, el contenido del parámetro formal se asigna al parámetro real.
IN OUT	Es la convinación de los dos anteriores, el valor del parámetro real se pasa al procedimiento, dentro del procedimiento el parámetro formal puede ser leído y modificado al devolver el control los contenidos del parámetro formal se asignan al parámetro real.

El mecanismo de paso de parámetros no sólo pasa los valores, sino también las restricciones que afectan a las variables. En una declaración de procedimeinto es ilegar restringir un parámetro CHAR o VARCHAR2 con una determinada longitud o un valor NUMBER con una determinada precisión o escala, el tamaño o las restricciones vendrán determinado por los parámetros reales. La única forma de definir restricciones en un parámetro formal es usar %TYPE.

Los argumentos reales se asocian con los formales por posición, aunque PL/SQL también admite una asociación nominal que asocia explicitamente los valores con los parámetros.. Ejemplo llamada

nombre_procedimiento(param1=> 'Carlos', param2 => 'López');

Los parámetros pueden inicializarse a un valor predeterminado, en ese caso no es necesario pasarlo al realizar la llamada, lo mejor en este caso es hacer una asociación nominal.

Funciones:

Las funciones comparten todo lo señalado para los procedmientos, pero además devuelve un valor por lo que una llamada a una función debe realizarse dentro de una expresión.

Sintaxis de las funciones:

```
CREATE [OR REPLACE] FUNCTION nombre_función
[(argumento [IN| OUT | IN OUT)] tipo,
....
argumento [IN| OUT | IN OUT)] tipo)]
RETURN tipo_retorno {IS|AS}

cuerpo_función
```

El **tipo_retorno** es el tipo de valor que devuelve la función.

La orden RETURN

Dentro del cuerpo de la función la orden RETURN se explea paea devolver el control, y un valor, al entorno que realizó la llamada. La sintaxis de la orden RETURN es:

```
RETURN expresión
```

Donde expresión es el valor que la función evuelve, el cual se convierte al tipo especificado en la la cláusula RETURN, si es que no era de ese tipo.

Puede haber más de una orden RETURN dentro de una función pero sólo se ejecutará una, aquella que la lógica del programa encuentre en primer lugar.