6. GESTIÓN DE EXCEPCIONES.

6.1.- INTRODUCCIÓN

Con las excepciones podemos tratar errores en los mensajes de Oracle. Una excepción es un identificador de PL/SQL que surge durante la ejecución de un bloque y termina con su cuerpo principal de acciones. Un bloque termina cuando PL provoca una excepción.

Existen dos métodos para provocar una excepción:

1. Si se produce un error de Oracle y surge automáticamente la excepción asociada. Por ejemplo, el error ORA-01403 indica que no se ha recuperado ninguna fila y es en este caso cuando PL provoca la excepción NO_DATA_FOUND.

2. Provoca explícitamente una excepción, es decir, que el usuario sea el que la predefina.

Si el bloque no tiene área de excepciones este se interrumpe dando el servidor los mensajes pertinentes que tenga. Si tiene área de excepciones el error se propaga a dicha área, enviando el servidor el mensaje que considere oportuno.

Las excepciones pueden ser:

- **Implícitas**: que son predefinidas y no predefinidas por el servidos Oracle
- **Explicitas**: que son las definidas por el usuario.

Sintaxis:

```
EXCEPTION
WHEN excep1 [ OR excep2 ... ] THEN
        ordenes;
        ...
[WHEN excep3 [ OR excep4 …] THEN
        ordenes;
        … ]
[WHEN OTHERS THEN
        ordenes;
        … ]
```

Donde:

EXCEP: es el nombre de la excepción estándar o definida por el usuario.

OTHERS: es la que irrumpe en cualquier otro caso.

6.2.- EXCEPCIONES INTERNAS DE PL/SQL (PREDEFINIDAS POR EL SERVIDOR)

Este tipo de excepciones se disparan ante un determinado error detectado por PL/SQL. No hay que declararlas. Disponemos de las siguientes:

Excepciones	Descripción
NO_DATA_FOUND	Se produce cuando una orden del tipo SELECT INTO no ha devuelto ningún valor.
ZERO_DIVIDE	Sucede cuando se produce cuando se divide un valor por 0.
INVALID_CURSOR	Se produce cuando se produce una operación ilegal en un cursor.
TOO_MANY_ROWS	Se produce cuando una orden SELECT INTO devuelva más de una fila.
INVALID_NUMBER	Sucede cuando hay un fallo de conversión de una cadena a un número.
DUP_VAL_ON_INDEX	Se produce cuando se intenta introducir un duplicado en una tabla.
LOGIN_DENIED	Sucede cuando se introduce un nombre de usuario o contraseña no valido.
CURSOR_ALREADY_OPEN	Se produce cuando se intenta abrir un cursor que ya está abierto.
STORAGE_ERROR	Se produce cuando hay un fallo en memoria.
VALUE_ERROR	Se produce cuando hay un error de operación aritmética de conversión, de truncamiento o de restricción.
TIMEOUT_ON_RESOURCE	Sucede cuando se acaba el tiempo al coger determinados recursos.
ACCESS_INTO_NULL	Se produce cuando se intenta acceder a los atributos de un objeto no inicializado.
COLLECTION_IS_NULL	Se produce cuando se intenta acceder a elementos de una colección no inicializada.
NOT_LOGGED_ON	Se produce cuando se intenta acceder a la base de datos sin estar conectado a Oracle.
PROGRAM_ERROR	Sucede si hay un problema interno en la ejecución de un programa.
ROWTYPE_MISMATCH	Se produce cuando la variable HOST y la variable PL/SQL pertenecen a tipos incompatibles.
SUBSCRIPT_OUTSIDE_LIMIT	Sucede cuando se intenta acceder a una tabla o a un array con un valor de índice ilegal.

Hacer un bloque PL, utilizando excepciones internas, que introduciendo dos números por teclado divida N1 entre N2.

```
DECLARE
        NUM1 NUMBER:=&n1;
        NUM2 NUMBER:=&n2;
        DIVIDE NUMBER;
BEGIN
        DIVIDE:= NUM1/NUM2;
EXCEPTION
        WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('IMPOSIBLE DIVIDIR POR 0');
END;
/
```

6.3.- EXCEPCIONES DEFINIDAS POR EL USUARIO.

Para poder utilizar o crear una excepción definida a nivel a usuario, lo primero que se debe hacer es declararla en la zona de declaraciones. Siendo su sintaxis en la zona de excepciones idéntica a las anteriores.

Para arrancar una excepción de usuario y pasarle el control a la sección de excepciones utilizaremos la orden RAISE seguida del nombre de la excepción.

Para su utilización hay que dar tres pasos:

1. Se deben declarar en la sección DECLARE de la siguiente forma:

   ```
   nb_excepción EXCEPTION;
   ```

2. Se disparan o levantan en la sección ejecutable del programa con la orden RAISE.

   ```
   RAISE nb_excepción;
   ```

3. Se tratan en la sección EXCEPTION según el formato ya conocido.

   ```
   WHEN nb_excepción THEN tratamiento;
   ```

Ejemplo:

Definir un cursor que seleccione ENAME y COMM, ordenado por comisión de la tabla EMP. Transferir a la tabla otra solo aquellos cuya comisión sea no nula.

```
DECLARE
        CURSOR C1 IS
                SELECT ENAME, COMM
                FROM EMP
                ORDER BY 2;
        V_ENAME EMP.ENAME%TYPE;
        V_COM EMP.COMM%TYPE;
        I BINARY_INTEGER:=14;
        CONT NUMBER:=0;
        ERROR_COMMIS EXCEPTION;
BEGIN
        OPEN C1;
        FETCH C1 INTO V_ENAME, V_COM;
        WHILE (C1%ROWCOUNT <=I) AND (C1%FOUND) LOOP
                IF V_COM IS NULL THEN
                        RAISE ERROR_COMMIS;
                END IF;
                CONT:= CONT+1;
                INSERT INTO OTRA3 VALUES (V_ENAME, V_COM);
                FETCH C1 INTO V_ENAME, V_COM;
        END LOOP;
        CLOSE C1;
EXCEPTION
        WHEN ERROR_COMMIS THEN
        DBMS_OUTPUT.PUT_LINE (CONT || ' REGISTROS INTODUCIDOS');
END;
/
```

6.4.- EXCEPCIONES ASOCIADAS A ERRORES DE ORACLE. (NO PREDEFINIDOS POR EL SERVIDOR)

Son también definidas por el usuario, ya que él es quien decide su asignación y funcionamiento, sin embargo lo que hace es definir una determinada excepción que responda a un tipo de error de Oracle. Para asociar una excepción a alguno de estos errores internos de Oracle que no tienen excepciones predefinidas asociadas. Procederemos de la siguiente forma:

1. Definimos la excepción en la sección declarativa.

```
Nb_excepción EXCEPTION;
```

2. Asociamos esa excepción a un determinado código de error mediante la directiva del compilador PRAGMA EXCEPTION_INIT, según el siguiente formato:

```
PRAGMA EXCEPTION_INIT (nb_excepción, -número_error_oracle);
```

3. Indicamos el tratamiento de la excepción como si se tratase de cualquier otra excepción definida o predefinida.

Ejemplo:

```
DECLARE

        ERROR_EXTERNO EXCEPTION;
        PRAGMA EXCEPTION_INIT (ERROR_EXTERNO, -1547);

BEGIN
        ...

EXCEPTION

        WHEN ERROR_EXTERNO THEN
        DBMS_OUTPUT.PUT_LINE('ERROR, TABLESPACE SIN ESPACIO');

END;
/
```

6.5.- FUNCIONES PARA INTRODUCIR EXCEPCIONES

Cuando se produce una excepción se pretende identificar el código de error asociado o el mensaje mediante dos funciones:

<table>
<tr><td>Funciones</td><td>Descripción</td></tr>
<tr><td>SQLCODE</td><td>Devuelve el número de error de Oracle de las excepciones internas. Se le asigna una variable NUMBER. Los valores DECODE son los siguientes:
<table>
<tr><td>Valor</td><td>Descripción</td></tr>
<tr><td>0</td><td>No se encontró ninguna excepción.</td></tr>
<tr><td>1</td><td>Excepción definida por el usuario.</td></tr>
<tr><td>+100</td><td>Excepción NO_DATA_FOUND.</td></tr>
<tr><td>Nº negativo</td><td>Otro número de error del servidor.</td></tr>
</table></td></tr>
<tr><td>SQLERRM</td><td>Devuelve datos carácter que contienen el mensaje de error asociado.</td></tr>
</table>

Ejemplo 1:

```
DECLARE
...
V_ERROR_CODE NUMBER;
V_ERROR_MENSAJE VARCHAR2(25);

BEGIN
...
EXCEPTION
WHEN OTHERS THEN
```

```
        ROLLBACK;
        V_ERROR_CODE:=SQLCODE;
        V_ERROR_MENSAJE:=SQLERROR;
        ...
        END;
```

Ejemplo 2:

Ejemplo recopilatorio de todo lo visto en la unidad.

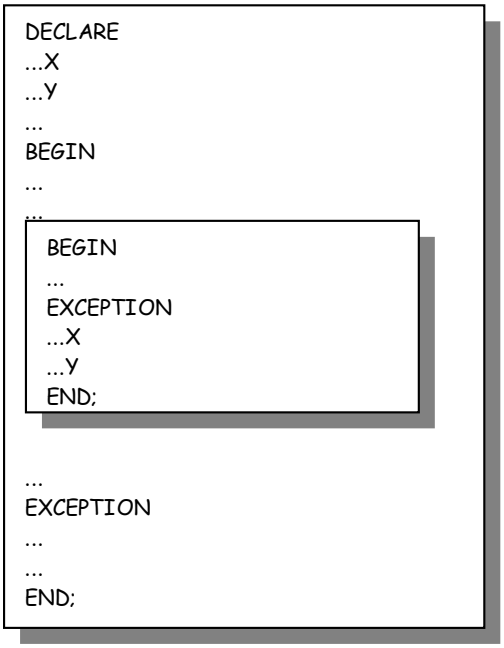
```
 DECLARE
        COD_ERR NUMBER(6);
        VNIF VARCHAR2(10);
        VNOM VARCHAR2(15);
        ERR_BLANCOS EXCEPTION;
        NO_HAY_ESPACIO EXCEPTION;
        PRAGMA EXCEPTION_INIT(NO_HAY_ESPACIO, -1547);
BEGIN
        SELECT COL1, COL2 INTO VNIF, VNOM
        FROM TEMP2;
        IF SUBSTR(VNOM,1,1)<= ' ' THEN
                RAISE ERR_BLANCOS;
        END IF;
        UPDATE CLIENTES SET NOMBRE=VNOM WHERE NIF=VNIF;
EXCEPTION
        WHEN ERR_BLANCOS THEN
                INSERT INTO TEMP2(COL1) VALUES ('ERR BLANCOS');
        WHEN NO_HAY_ESPACIO THEN
                INSERT INTO TEMP2(COL1) VALUES ('ERR TABLESPACE');
        WHEN NO_DATA_FOUND THEN
                INSERT INTO TEMP2(COL1) VALUES ('ERR NO HABIA DATOS');
        WHEN TOO_MANY_ROWS THEN
                INSERT INTO TEMP2(COL1) VALUES ('ERR DEMASIADOS DATOS');
        WHEN OTHERS THEN
                COD_ERR := SQLCODE;
                INSERT INTO TEMP2(COL1) VALUES (COD_ERR);
END;
/
```

6.6.- PROPAGACIÓN DE EXCEPCIONES

Cuando un subprograma gestiona una excepción y este termina normalmente el control se reanuda en el bloque padre después del END del sub-bloque.

Sin embargo, si hubiese una excepción en el hijo ésta se propaga al padre, hasta que encuentre un manejador de la misma.

Las excepciones tratadas en el sub-bloque han de ser declaradas en el bloque principal.



6.7.- RAISE_APPLICATION_ERROR

El paquete DBMS_STANDARD incluye un procedimiento muy útil llamado RAISE_APPLICATIOM_ERROR que sirve para levantar errores, definir y enviar mensajes de error.

Sintaxis:

```
RAISE_APPLICATION_ERROR(-nº_error, mensaje_error);
```

Donde número_error: es un número comprendido entre –20000 y –20999 y mensaje_error es una cadena de hasta 512 bytes.

Ejemplo:

```
PROCEDURE SUBIR_SUELDO
        (NUM_EMPLE NUMBER, INCREMENTO NUMBER)
IS
        SALARIO_ACTUAL NUMBER;
BEGIN
        SELECT SALARIO INTO SALARIO_ACTUAL
        FROM EMPLEADOS
        WHERE EMP_NO = NUM_EMPLE;
        IF SALARIO_ACTUAL IS NULL THEN
                RAISE_APPLICATION_ERROR(-20010, 'SALARIO NULO');
        ELSE
                UPDATE EMPLEADOS
                SET SUELDO=SALARIO_ACTUAL+INCREMENTO
                WHERE EMP_NO = NUM_EMPLE;
        END IF ;
END SUBIR_SUELDO;
```