EXCEPCIONES

Las **excepciones** sirven para tratar errores en tiempo de ejecución, así como errores y situaciones definidas por el usuario.

Cuando se produce un error PL/SQL levanta una excepción y pasa el control a la sección EXCEPTION, donde buscará un manejador WHEN para la excepción o uno genérico (WHEN OTHERS) y dará por finalizada la ejecución del bloque actual.

Para controlar posibles errores en otros lenguajes que no disponen de gestión de excepciones, se debe controlar después de cada orden cada una de las posibles condiciones de error.

El formato de la sección EXCEPTION es:

```
...
EXCEPTION
    WHEN <NombredeExcepción1> THEN
      <instrucciones1>;
    WHEN <NombredeExcepción2> THEN
      <instrucciones2>;
    ...
    [WHEN OTHERS THEN
       <instrucciones>;]
END <nombre de programa>;
```

A continuación, estudiaremos los tres tipos de excepciones disponibles.

A. Excepciones internas predefinidas

Están predefinidas por Oracle. Se disparan automáticamente al producirse determinados errores. En la tabla adjunta se incluyen las excepciones más frecuentes con los códigos de error correspondientes:

Código error Oracle	Valor de SQL CODE	Excepción	Se disparan cuando...
ORA-06530	-6530	ACCESS_INTO_NULL	Se intenta acceder a los atributos de un objeto no inicializado.
ORA-06531	-6531	COLLECTION_IS_NULL	Se intenta acceder a elementos de una colección que no ha sido inicializada.
ORA-06511	-6511	CURSOR_ALREADY_OPEN	Intentamos abrir un cursor que ya se encuentra abierto.
ORA-00001	-1	DUP_VAL_ON_INDEX	Se intenta almacenar un valor que crearía duplicados en la clave primaria o en una columna con la restricción UNIQUE.
ORA-01001	-1001	INVALID_CURSOR	Se intenta realizar una operación no permitida sobre un cursor (por ejemplo, cerrar un cursor que no estaba abierto).
ORA-01722	-1722	INVALID_NUMBER	Fallo al intentar convertir una cadena a un valor numérico.
ORA-01017	-1017	LOGIN_DENIED	Se intenta conectar a ORACLE con un usuario o una clave no válidos.
ORA-01012	-1012	NOT_LOGGED_ON	Se intenta acceder a la base de datos sin estar conectado a Oracle.
ORA-01403	+100	NO_DATA_FOUND	Una sentencia SELECT ... INTO ... no devuelve ninguna fila.
ORA-06501	-6501	PROGRAM_ERROR	Hay un problema interno en la ejecución del programa.
ORA-06504	-6504	ROWTYPE_MISMATCH	La variable del cursor del HOST y la variable del cursor PL/SQL pertenecen a tipos incompatibles.
ORA-06533	-6533	SUBSCRIPT_OUTSIDE_LIMIT	Se intenta acceder a una tabla anidada o a un *array* con un valor de índice ilegal (por ejemplo, negativo).
ORA-06500	-6500	STORAGE_ERROR	El bloque PL/SQL se ejecuta fuera de memoria (o hay algún otro error de memoria).
ORA-00051	-51	TIMEOUT_ON_RESOURCE	Se excede el tiempo de espera para un recurso.
ORA-01422	-1422	TOO_MANY_ROWS	Una sentencia SELECT ... INTO ... devuelve más de una fila.
ORA-06502	-6502	VALUE_ERROR	Un error de tipo aritmético, de conversión, de truncamiento, etcétera.
ORA-01476	-1476	ZERO_DIVIDE	Se intenta la división entre cero.

Tabla 10.2. *Excepciones más frecuentes en Oracle con los códigos de error.*

No hay que declararlas en la sección DECLARE. Únicamente debemos incluir los manejadores WHEN con el tratamiento para cada excepción y/o un manejador genérico WHEN OTHERS que capturará cualquier otra excepción.

```
DECLARE
    ...
BEGIN
    ...
EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

```
    BDMS_OUTPUT.PUT_LINE ('ERROR datos no encontrados');
  WHEN TOO_MANY_ROWS THEN
    BDMS_OUTPUT.PUT_LINE ('ERROR demasiadas filas');
  WHEN OTHERS THEN
    BDMS_OUTPUT.PUT_LINE ('ERROR');
END;
```

B. Excepciones definidas por el usuario

Las excepciones definidas por el usuario se usan para tratar condiciones de error definidas por el programador.

Para su utilización hay que seguir tres pasos:

1. Se declaran en la sección DECLARE de la forma siguiente:

   ```
   <nombreexcepción> EXCEPTION;
   ```

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

   ```
   RAISE <nombreexcepción>;
   ```

   También pueden levantarse excepciones en la sección EXCEPTION, pero no es habitual.

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

   ```
   WHEN <nombreexcepción> THEN <tratamiento>;
   ```

```
DECLARE
    ...
    importe_erroneo EXCEPTION;
    ...
BEGIN
    ...
    IF precio NOT BETWEEN precio_min AND precio_maximo
THEN
        RAISE importe_erroneo;
    END IF;
    ...
EXCEPTION
    ...
    WHEN importe_erroneo THEN
        DBMS_OUTPUT.PUT_LINE('Importe erróneo.
                              Venta cancelada.');
    ...
END;
```

La instrucción RAISE se puede usar varias veces en el mismo bloque con la misma o con distintas excepciones, pero sólo puede haber un manejador WHEN para cada excepción.

```
DECLARE
    ...
    venta_erronea EXCEPTION;
    ...
    importe_erroneo EXCEPTION;
    ...
BEGIN
    ...
    RAISE venta_erronea;
    ...
    RAISE importe_erroneo;
    ...
    RAISE venta_erronea;
    ...
EXCEPTION
    ...
    WHEN importe_erroneo THEN
      ...;
    WHEN venta_erronea THEN
      ...;
END;
```

Caso práctico

5 **El siguiente ejemplo recibe un número de empleado y una cantidad que se incrementará al salario del empleado correspondiente. Utilizaremos dos excepciones, una definida por el usuario `salario_nulo` y la otra predefinida `NO_DATA_FOUND`.**

```
CREATE OR REPLACE
PROCEDURE subir_salario(
   num_empleado INTEGER,
   incremento REAL)
IS
   salario_actual REAL;
   salario_nulo EXCEPTION;

BEGIN
   SELECT salario INTO salario_actual FROM emple
      WHERE emp_no = num_empleado;

   IF salario_actual IS NULL THEN
      RAISE salario_nulo;  -- levanta salario_nulo
   END IF;

   UPDATE emple SET salario = salario + incremento
          WHERE emp_no = num_empleado;
```

(Continúa)

(Continuación)

```
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE(num_empleado||'*Err.No encontrado');
      WHEN salario_nulo THEN
       DBMS_OUTPUT.PUT_LINE(num_empleado||'*Err. Salario nulo');
    END subir_salario;
```

C. Otras excepciones

Existen otros errores internos de Oracle que no tienen asignada una excepción. No obstante, generan (como cualquier otro error de Oracle) un código de error y un mensaje de error, a los que se accede mediante las funciones SQLCODE y SQLERRM.

Cuando se produce uno de estos errores también se transfiere el control a la sección EXCEPTION, donde se tratará el error en la cláusula WHEN OTHERS:

```
EXCEPTION
  ...
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error:'||SQLCODE||SQLERRM);
   ...
END;
```

En este ejemplo, se muestra al usuario el texto 'ERROR' con el *código de error* y el *mensaje de error* utilizando las funciones correspondientes.

Estas dos funciones, SQLCODE y SQLERRM, no son accesibles desde órdenes SQL*Plus, pero se pueden pasar a este entorno a través de soluciones como la siguiente:

```
...
  WHEN OTHERS THEN
    :cod_err := SQLCODE;
    :msg_err := SQLERRM;
    ROLLBACK;
    EXIT;
END;
```

Podemos asociar una excepción de usuario a alguno de estos errores internos que no tienen excepciones predefinidas asociadas. Para ello procederemos así:

1. Definimos la excepción en la sección de declaraciones como si fuese una excepción definida por el usuario:

   <nombreexcepción> EXCEPTION;

En realidad, las excepciones predefinidas estudiadas anteriormente no son sino errores de Oracle asociados con excepciones (de manera similar a éstas) en el paquete STANDARD.

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

```
PRAGMA EXCEPTION_INIT(<nombre_excepción>, <número_de_error_
Oracle>);
```

3. Indicamos el tratamiento que recibirá la excepción en la sección EXCEPTION como si se tratase de cualquier otra excepción definida por el usuario o predefinida.

```
DECLARE
    ...
    err_externo EXCEPTION;  -- Se define la excepción
    ...
    PRAGMA EXCEPTION_INIT(err_externo, -1547);/* Se asocia
                con el número de error de Oracle -1547 */
    ...
BEGIN
    ...
    /* no hay que levantar la excepción, pues cuando
       se produzca el error Oracle lo hará */
    ...
EXCEPTION
    ...
    WHEN err_externo THEN    -- Se trata como cualquier otra
         <tratamiento>;
END;
```

Caso práctico

6 El siguiente ejemplo ilustra lo estudiado hasta ahora respecto a la gestión de excepciones. Crearemos un bloque donde se define la excepción `err_blancos` asociada con un error definido por el programador y la excepción `no_hay_espacio` asociándola con el error número -1547 de Oracle.

```
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;
```

(Continúa)

(Continuación)

```
        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 ('ERdemasiados datos');
       WHEN OTHERS THEN
          cod_err := SQLCODE;
          INSERT INTO temp2(col1) VALUES (cod_err);
    END;
```

Cabe subrayar respecto a las excepciones que aparecen:

- *error_blancos* hay que declararla y levantarla.
- *no_hay_espacio* hay que declararla y asociarla con el error de Oracle, pero no hay que levantarla.
- *NO_DATA_FOUND* y *TOO_MANY_ROWS* no hay que declararlas ni asociarlas. Tampoco hay que levantarlas.
- El manejador WHEN OTHERS cazará cualquier otra excepción e insertará el código de error de Oracle en la tabla *temp2* que suponemos creada.

E. Propagación y ámbito de las excepciones

La gestión de excepciones en PL/SQL tiene unas reglas que se deben considerar en el diseño de aplicaciones:

- Cuando se levanta una excepción en la sección ejecutable, el programa bifurca a la sección EXCEPTION del bloque actual. Si no está definida en ella, la excepción se propaga a la sección EXCEPTION del bloque que llamó al actual; así, hasta encontrar tratamiento para la excepción o devolver el control al programa Host.
- Una vez tratada la excepción en un bloque, se devuelve el control al bloque que llamó al que trató la excepción, con independencia del que la disparó.
- Si, después de tratar una excepción, queremos volver a la línea siguiente a la que se produjo, no podemos hacerlo directamente, pero sí es posible diseñar el programa para que funcione así. Esto se consigue encerrando el comando o comandos que pueden levantar la excepción en un bloque junto el tratamiento para la excepción:

```
SELECT INTO ... -- > Puede levantar NO_DATA_FOUND
...
```

Si queremos que dos o más excepciones ejecuten la misma secuencia de instrucciones, podremos indicarlo en la cláusula WHEN indicando las excepciones unidas por el operador OR:

```
WHEN exc1 OR exc2 OR
exc3... THEN...
```

No obstante, en la lista no podrá aparecer WHEN OTHERS.

Para que el control del programa no salga del bloque actual cuando se produzca una excepción, podemos encerrar el comando que puede levantar la excepción en un bloque y tratar la excepción en ese bloque:

```
...
BEGIN
  SELECT INTO ...      -- > Puede levantar NO_DATA_FOUND
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    ...;               -- > tratamiento para la excepción
END;
...
```

- La cláusula WHEN OTHERS tratará cualquier excepción que no aparezca en las cláusulas WHEN anteriores, con independencia del tipo de excepción. De este modo, se evita que la excepción se propague a los bloques de nivel superior.

  En el siguiente ejemplo, si se levanta la excepción *ex1*, se tratará en el mismo bloque, pero el control pasará después al bloque <<exterior>> en la línea siguiente a la llamada. Si se hubiese levantado NO_DATA_FOUND, al no encontrar tratamiento, la excepción se propaga al bloque <<exterior>>, donde será tratada por WHEN OTHERS (suponiendo que no exista un tratamiento específico), devolviendo el control al bloque o la herramienta que llamó a <<exterior>>:

```
<<exterior>>
DECLARE
  ...
BEGIN
  ...
    <<interior>>
    DECLARE
        ...
        ex1 EXCEPTION;
    BEGIN
        ...
        RAISE ex1;
        ...
         SELECT col1, col2 INTO ...; --> Levanta
 NO_DATA_FOUND
        ...
    EXCEPTION
        ...
        WHEN ex1 THEN  --> Tratamiento para ex1
            ROLLBACK; --> Después pasará el control
 a (1)
        ...
    END;
  /*(1)*/... ;
  ...
EXCEPTION
  ...
  WHEN OTHERS THEN...
END;
```

- Si la excepción se levanta en la sección declarativa (por un fallo al inicializar una variable, por ejemplo), automáticamente se propagará al bloque que llamó al actual, sin comprobar si existe tratamiento para la excepción en el mismo bloque que se levantó.

Si la excepción se levanta en la sección declarativa o en la sección EXCEPTION, automáticamente se propagará al bloque que llamó al actual.

- También se puede levantar una excepción en la sección EXCEPTION de forma voluntaria o por un error que se produzca al tratar una excepción. En este caso, también se propagará de forma automática al bloque que llamó al actual, sin comprobar si existe tratamiento para la excepción en el mismo bloque que se levantó. La excepción original (la que se estaba tratando cuando se produjo la nueva excepción) se perderá, ya que solamente puede estar activa una excepción.

```
EXCEPTION
   WHEN INVALID_NUMBER THEN
      INSERT INTO ...-- podría levantar DUP_VAL_ON_INDEX
   WHEN DUP_VAL_ON_INDEX THEN -- no atrapará la exception
```

Algunos tipos de excepciones se han de tratar con mucha precaución, ya que se puede caer en un bucle infinito (por ejemplo, NOT_LOGGED_ON).

- En ocasiones, puede resultar conveniente volver a levantar la misma excepción después de tratarla para que se propague al bloque de nivel superior. Esto puede hacerse indicando al final de los comandos de manejo de la excepción el comando **RAISE** sin parámetros:

```
...
WHEN TOO_MANY_ROWS THEN
     ...;    -- intrucciones de manejo del error
     RAISE;  -- levanta de nuevo la excepción y la pro-
paga
...
```

- Las excepciones declaradas en un bloque son locales al bloque, no son conocidas en bloques de nivel superior. No se puede declarar dos veces la misma excepción en el mismo bloque, pero sí en distintos bloques. En este caso, la excepción del subbloque prevalecerá sobre la del bloque, aunque esta última se puede levantar desde el subbloque utilizando la notación de punto (RAISE *nombrebloque.nombreexcepción*).

Si una excepción de usuario no encuentra tratamiento en el bloque en el que ha sido declarada se propagará a bloques de nivel superior; pero éstos sólo pueden cazarlas mediante WHEN OTHERS pues la excepción es desconocida fuera de su ámbito.

- Las variables locales, las globales, los atributos de un cursor, y otros objetos del programa se pueden referenciar desde la sección EXCEPTION según las reglas de ámbito que rigen para los objetos del bloque. Pero si la excepción se ha disparado dentro de un bucle cursor FOR...LOOP, no se podrá acceder a los atributos del cursor, ya que Oracle cierra este cursor antes de disparar la excepción.

- En la sección EXCEPTION no se puede usar < GOTO etiqueta > para salir de esta sección o entrar en otra cláusula WHERE.

- Cuando no se encuentra tratamiento para una excepción en ninguno de los bloques o programas, Oracle da por finalizado con fallos el programa y ejecutará automáticamente un ROLLBACK deshaciendo todos los cambios pendientes de confirmación realizados por el programa.

F. Utilización de RAISE_APPLICATION_ERROR

RAISE_APPLICATION_ERROR tiene un tercer parámetro opcional cuyos detalles de uso quedan fuera de nuestros objetivos.

En el paquete DBMS_STANDARD se incluye un procedimiento muy útil llamado RAISE_APPLICATION_ERROR que sirve para levantar errores y definir y enviar mensajes de error. Su formato es el siguiente:

RAISE_APPLICATION_ERROR(*número_de_error, mensaje_de_error*);

Donde *número_de_error* es un número comprendido entre -20000 y -20999, y *mensaje_de_error* es una cadena de hasta 512 bytes.

Cuando un subprograma hace esta llamada, se levanta la excepción y produce la salida del programa. Esta excepción solamente puede ser manejada con WHEN OTHERS.

Caso práctico

7 **El siguiente ejemplo muestra el funcionamiento de *RAISE_APPLICATION_ ERROR* en un procedimiento de funcionalidad similar al estudiado en el caso práctico 7 (*subir_salario*).**

```
CREATE OR REPLACE
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;
   ENDIF
END subir_sueldo;
```