EXCEPCIONES

1. TIPOS DE EXCEPCIONES
2. PROPAGACIÓN DE EXCEPCIONES

Una excepción permite tratar errores en tiempo de ejecución o situaciones definidas por el usuario. Al producirse un error, PL/SQL levanta una excepción y pasa el control a la correspondiente sección de excepciones, aquí intentará localizar su correspondiente manejador (WHEN) o uno genérico(WHEN OTHERS), seguirá las instrucciones que encuentre y dará por terminado el bloque donde se había producido.

El formato de la zona de excepciones es:

```
EXCEPTION
…
WHEN excepcion1 THEN
        Instrucciones1;
WHEN excepcion2 THEN
        Instrucciones2;
….
[WHEN OTHERS THEN
        Instrucciones;]
END; -- Fin del programa donde está incluida esta sección
```

1. TIPOS DE EXCEPCIONES

En PL existen tres tipos de excepciones:

- Excepciones predefinidas correspondientes a errores internos
- Excepciones definidas por el usuario
- Otras excepciones internas no definidas por el sistema

Excepciones internas predefinidas

Se disparan automáticamente al producirse determinados errores y las más habituales son las siguientes:

Excepcion	Nº error	Descripción
NO_DATA_FOUND	ORA-01403	SELECT NO DEVUELVE DATOS
TOO_MANY_ROWS	ORA-01422	SELECT DEVUELVE MÁS DE 1 FILA
INVALID_CURSOR	ORA-01001	OPERACIÓN ILEGAL DE CURSOR
ZERO_DIVIDE	ORA-01476	DIVIDIR POR 0
DUP_VAL_ON_INDEX	ORA-01017	INSERTAR UN REGISTRO DUPLICADO

No es necesario declararlas en la sección DECLARE, simplemente tendremos un manejador WHEN para ellas en la zona de excepciones o pueden ser capturadas por el manejador genérico WHEN OTHERS.

Si se utiliza el manejador WHEN OTHERS, éste debe ser siempre el último gestor de un bloque. Para saber el error que provocó la excepción dentro del gestor de excepciones OTHERS podemos usar las funciones SQLCODE y SQLERRM para obtener el código del error y el mensaje asociado

Ejemplos de uso:

```
EXCEPTION
WHEN NO_DATA_FOUND THEN
        MOSTRAR('ERROR, DATOS NO ENCONTRADOS');
WHEN TOO_MANY_ROWS THEN
                MOSTRAR('ERROR, DEMASIADAS FILAS');
WHEN OTHERS THEN
        MOSTRAR('SE HA PRODUCIDO UN ERROR');
```

Ejemplo de excepción de sistema:

```
create or replace procedure localidad                                   (iddep in number) is
  loc                                                                          varchar2(60);
begin
  select loc                          into loc                  from DEPART
   where deptno = iddep;
  dbms_output.put_line(loc);
end;
```

Ejecutamos el procedimiento desde una consola SQL para obtener la descripción del metal con identificador 2:

```
begin
   localidad(10);
end;
```

Nos devolverá la localidad donde se encuentra el departamento con código 10.

Si lo ejecutamos con un departamento cuyo código no existe, ejemplo, localidad(1) se producirá un error nos devolverá un mensaje:

```
ORA-01403: no data found
ORA-06512: at "FINAL.LOCALIDAD", line 4
ORA-06512: at line 2
01403. 00000 -  "no data found"
```

Observamos que el log de error viene toda la traza, para ello repasamos el log de abajo a arriba, para ir obteniendo los saltos desde donde se han hechos las llamadas hasta dar con el error.

ORA-06512: at line 2 -> hace referencia al bloque de código anónimo que hemos escrito en la consola SQL, nos está indicando que el error se ha producido en la línea 2, es decir: localidad (1);

ORA-06512: at "FINAL.LOCALIDAD", line 4 -> nos informa que en un segundo nivel, y en este caso último, el error se ha producido dentro del procedimiento LOCALIDAD del esquema FINAL, concretamente en la línea 4, que es donde está la instrucción SELECT.

ORA-01403: no data found -> por último se indica el tipo de error propiamente dicho, en este caso se trata de un error "no data found". Está indicando que la consulta de la línea 4 del procedimiento LOCALIDAD no ha encontrado el registro y ha provocado la excepción.

Para tratar el error, se añade en el procedimiento una zona de excepciones con su correspondiente manejador WHEN:

```
create or replace procedure localidad(iddep in number) is
  loca varchar2(60);
begin
  select loc into loca from DEPART
   where deptno = iddep;
  dbms_output.put_line(loca);
exception
        when no_data_found then
                dbms_output.put_line('No existe ese departamento');
end;
```

***Actividad 1**. Realiza un bloque que pruebe las excepciones predefinidas más habituales indicadas en la página 1.*

Excepciones definidas por el usuario

Se utilizan para tratar condiciones de error definidas por el programador. Hay que declararlas en la zona DECLARE, se levantan dentro de la sección ejecutable con la instrucción RAISE y se tratan en la zona EXCEPTION con su correspondiente WHEN.

Ejemplo:

```
create or replace function raiz_cuadrada(p_valor in number) return number is
begin
  return sqrt(p_valor);
end;
begin
  dbms_output.put_line('Resultado: '||raiz_cuadrada(9));
end;
```

Se mostrará: Resultado: 3

Si ejecutamos:

```
begin
  dbms_output.put_line('Resultado: '||raiz_cuadrada(-9));
end;
```

Se mostrará:

```
ORA-06502: PL/SQL: numeric or value error
ORA-06512: at "FINAL.RAIZ_CUADRADA", line 3
ORA-06512: at line 2
```

Vamos a definir una excepción que llamaremos NUMERO_NEGATIVO, para que si el número que le pasamos a la función es negativo forzar que salte hasta el apartado de excepciones, evitando así que se intente calcular la raíz de un numero negativo. En tal caso la función retornará un valor nulo.

```
create or replace function raiz_cuadrada(p_valor in number) return number is
  NUMERO_NEGATIVO exception;
begin
  if p_valor < 0 then
    raise NUMERO_NEGATIVO;
  end if;

  return sqrt(p_valor);

exception
  when NUMERO_NEGATIVO then
    return null;
end;
```

Con esta solución ya no se produce error, simplemente se devuelve nulo o indeterminado.

La instrucción RAISE puede aparecer varias veces en el mismo bloque pero sólo existirá un manejador WHEN para ella en la zona de excepciones.

***Actividad 2**: diseñar un procedimiento que reciba un número de empleado y la cantidad que se incrementará al salario de dicho empleado. Utilizar dos excepciones, una definida por el usuario llamada salarioNulo que saltará cuando el salario del empleado sea nulo y la otra predefinida, NO_DATA_FOUND*

Otras excepciones

PRAGMA EXCEPTION_INIT, son excepciones internas de Oracle que no tienen nombre. Se les asocia un código y un mensaje dentro de la zona de declaraciones y se levantará automáticamente cuando se produzca la situación que queremos controlar. Partimos de las tablas EMPLE y DEPART, si queremos borrar un departamento y éste tiene asociados empleados nos dará el error:

```
delete from depart where deptno=10;
```

```
ORA-02292: integrity constraint (FINAL.EMPLE_FK1) violated - child record found
```

Podemos recoger ese código y darle nuestro propio mensaje de la siguiente forma:

```
DECLARE
  -- declaración y asociación de la excepción
        depNoVacio EXCEPTION;
        PRAGMA EXCEPTION_INIT (depNoVacio, -2292); -- le asociamos el código de error que nos dio en el anterior mensaje
        v_deptno dept.deptno%type:=10;
BEGIN
        DELETE FROM dept WHERE deptno = v_deptno;
         COMMIT;
EXCEPTION
        WHEN depNoVacio THEN
            dbms_output.put_line('no se puede borrar depto , existen empleados');
END;
```

RAISE_APPLICATION_ERROR, es un procedimiento dentro del paquete DBMS_STANDARD que sirve para definir excepciones de usuario que no están declaradas en la zona de declaraciones, permiten levantar un error y definir su código y su mensaje de error. Su formato es:
RAISE_APPLICATION_ERROR (número de error, mensaje de error), donde el número de error debe estar siempre comprendido entre -20000 y -209999 y el mensaje de error será una cadena.

Sólo se puede utilizar dentro de un subprograma almacenado y puede ser utilizado tanto en la sección ejecutable como en la de excepciones. Por ejemplo, dentro de la zona de ejecución:

```
create or replace procedure borrard(vde number)
is

BEGIN
        DELETE FROM depart WHERE deptno = vde;
        if sql%notfound then
                raise_application_error(-20001,' no existe el depto');
         end if;
END;
Para ejecutarlo:

        exec borrard(11);
```

SQLCODE Y SQLERRM, son dos pseudovariables que nos permiten obtener información relativa al error que se ha producido, respectivamente el número o código y el mensaje. Normalmente se utilizan en el manejador WHEN OTHERS.

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

2. PROPAGACIÓN DE EXCEPCIONES

Para gestionar las excepciones hay que tener en cuenta las siguientes reglas:

- Cuando se levanta una excepción en la sección ejecutable, se intenta tratar en la zona EXCEPTION del bloque actual. Si aquí no está tratada, se propaga a la zona EXCEPTION del bloque que llamó al actual y así sucesivamente hasta encontrar el correspondiente manejador o devolver el error correspondiente.
- Una vez tratada la excepción en un bloque, se devuelve el control al bloque que llamó al que trató la excepción.
- Si una vez tratada la excepción queremos volver a la línea siguiente a la que se produjo, no se puede hacer directamente. Se puede conseguir encerrando la orden que puede levantar la excepción en un subbloque junto al tratamiento de la excepción

Ejemplo: si tenemos un bloque que contiene la excepción SELECT … INTO.. , esta puede levantar NO_DATA_FOUND y nunca volverá a la instrucción siguiente:

```
…
SELECT … INTO … -> levantará NO_DATA_FOUND
Siguiente instrucción – que nunca se ejecutará
…
```

Solución:

```
BEGIN – Empieza el bloque principal
        …
        BEGIN --Subbloque
                SELECT … INTO … -> levantará NO_DATA_FOUND
        EXCEPTION
                WHEN NO_DATA_FOUND THEN
                        ….. – aquí se trata la excepción
        END; -- fin del subbloque
        Siguiente instrucción -- la ejecución del programa pasaría a esta instrucción
END; --fin del bloque principal
```

- La cláusula WHEN OTHERS tratará cualquier excepción que no aparezca en los manejadores anteriores, de esta forma se evita que 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 padre en la línea siguiente a la llamada. Si se hubiese levantado NO_DATA_FOUND, al no encontrar tratamiento, se propagará a la zona de excepciones del bloque padre y se tratará con WHEN OTHERS, devolviendo el control al bloque que hubiese llamado al padre.

```
DECLARE –BLOQUE PADRE
        …
BEGIN
        …
        DECLARE – BLOQUE HIJO
                …
                EX1 EXCEPTION;
        BEGIN
                Raise EX1;
                SELECT … INTO … ->  QUE LEVANTA EXCEPCION NO_DATA_FOUND
        EXCEPTION
                WHEN  EXE1 THEN …
        END;
…
WHEN OTHERS THEN
        …
END;
```

- Si la excepción se levanta en la zona declarativa, automáticamente se propagará al bloque que llamó al actual y no se comprueba si existe tratamiento para la excepción en el mismo bloque.

- En la zona EXCEPTION también se puede levantar una excepción. Entonces se propagará al bloque que llamó al actual, sin comprobar si existe tratamiento en el mismo bloque.

- A veces, 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 del tratamiento la orden RAISE sin parámetros.

- Las excepciones declaradas en un bloque son locales al bloque y no son conocidas en bloques de nivel superior. En el mismo bloque no se puede declarar dos veces la misma excepción, pero sí en distintos bloques.

- Si una excepción de usuario no encuentra tratamiento en el bloque en el que se declara, se propaga al bloque de nivel superior y sólo podrá ser capturada por el manejador WHEN OTHERS.

- Cuando no se encuentra tratamiento para una excepción en ninguno de los bloques, ORACLE da por finalizado el programa con fallos y ejecutará automáticamente un ROLLBAK deshaciendo todos los cambios pendientes.

***Actividad 3**. Escribir un procedimiento que reciba todos los datos de un nuevo empleado y procese su inserción en la tabla EMPLE. Para ellos se gestionarán los posibles errores que puedan producirse:*

- *Que no exista el departamento donde queremos insertarlo,*
- *Que no exista el director como empleado de la empresa,*
- *Qué el código del nuevo empleado ya exista en la tabla,*
- *Que el salario sea nulo, utilizad RAISE_APPLICTION_ERROR,*
- *Otros errores de Oracle que puedan producirse indicando código y mensaje del error.*