

Sistemas Gestores de Bases de Datos

Tema 8: Fundamentos de PL/SQL



IES Gonzalo Nazareno
CONSEJERÍA DE EDUCACIÓN

Raúl Ruiz Padilla

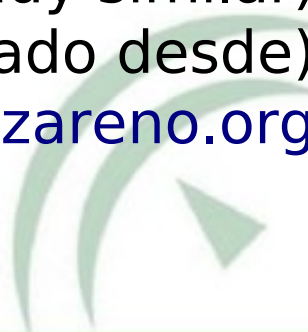
[rruizp@gmail.com](mailto:r Ruizp@gmail.com)

Febrero 2011

© Raúl Ruiz Padilla, Febrero de 2011
Basado en un trabajo previo de Laura Mateos Párraga y otros

Algunos derechos reservados.
Este artículo se distribuye bajo la licencia
"Reconocimiento-CompartirIgual 3.0 España" de Creative
Commons, disponible en
<http://creativecommons.org/licenses/by-sa/3.0/es/deed.es>

Este documento (o uno muy similar)
esta disponible en (o enlazado desde)
<http://informatica.gonzalonazareno.org>




Índice del Tema

- Características de PL/SQL
- Estructura de un bloque PL/SQL
- Operaciones básicas en PL/SQL
- Estructuras de control en PL/SQL
- Subprogramas: Funciones y Procedimientos
- Cursores
- Excepciones



Características de PL/SQL (I)

- Es una extensión de SQL con características típicas de los lenguajes de programación.
 - Procedural language / structured query language.
 - Las sentencias SQL de consulta y manipulación de datos pueden ser incluidas en unidades procedurales de código, pero no pueden usarse instrucciones DDL ni DCL.
 - Se ejecuta en el lado del servidor y los procedimientos y funciones se almacenan en la BD.
 - No tiene instrucciones de entrada por teclado o salida por pantalla.
- 

Características de PL/SQL (II)

- Incluye los tipos de datos y operadores de SQL.
- Los programas se pueden compilar desde SQL*Plus (comando /) o usar SQL Developer u otros IDEs.
- Los comentarios comienzan por - - o se colocan entre /* y */.
- Trae unas librerías con funciones predefinidas, se llaman paquetes.
- Para ejecutar los procedimientos almacenados desde SQL*Plus se usa el comando exec.



Estructura de un bloque PL/SQL

DECLARE **opcional**

variables, cursores, excepciones definidas por el usuario

BEGIN **obligatorio**

sentencias SQL

sentencias de control PL/SQL

EXCEPTION **opcional**

acciones a realizar cuando se producen errores

END; **obligatorio**



Estructura de un bloque PL/SQL

Ejemplo

DECLARE

```
v_usuario      VARCHAR2(10);  
v_fecha        DATE;
```

BEGIN

```
SELECT user_id, fecha  
INTO v_usuario, v_fecha  
FROM tabla;
```

EXCEPTION

```
WHEN nombre_excepcion then  
.....;
```

END;




Declaración de Variables en PL/SQL

Sintaxis

Identificador [CONSTANT] tipo_dato [NOT NULL] [:= | DEFAULT expresion];

Ejemplos

v_fecha	DATE;
v_deptno	NUMBER(2) NOT NULL := 10;
v_localidad	VARCHAR2(13) := 'ATLANTA';
v_comision	CONSTANT NUMBER := 1400;



Atributos %TYPE y %ROWTYPE

- **%TYPE** sirve para declarar una variable basada en:
 - Otras previamente declaradas.
 - Una columna de la BD.
- Preceder %TYPE por:
 - La tabla y la columna de la BD.
 - El nombre de la variable definida con anterioridad
- Usar **%ROWTYPE** para tipos de datos compuestos (filas completas, registros...).



Asignación de valores a variables

Sintaxis: **identificador := expresión;**

Ejemplos: **v_fecha := '31-OCT-2003';**
 v_apellido := 'López';

Desde consulta:

```
SELECT sal * 0.10 INTO v_comision  
FROM emp  
WHERE empno = 7082;
```



Dbms_output.put_line

Método de un paquete predefinido de ORACLE que sirve para que el servidor muestre información por pantalla, se usa fundamentalmente en la fase de depuración de los procedimientos.

begin

 dbms_output.put_line ('Mi 1º bloque PL/SQL
 diseñado por '||user||' el día '||sysdate);

end;



Sentencias SQL en PL/SQL

Para recuperar datos de la BD con SELECT.

Sintaxis:

```
SELECT lista_columnas  
INTO {variable_nombre[, .....]...  
      | nombre_registro}  
FROM nombre_tabla  
WHERE condicion;
```



Inserción de datos en PL/SQL

Ej.: *Añadir información sobre un nuevo empleado en la tabla emp.*

BEGIN

```
INSERT INTO emp(empno, ename, job, deptno)
VALUES(empno_sequence.nextval, 'HARDING',
'CLERK', 10);
```

END;



Actualización de datos en PL/SQL

Aumenta el salario de todos los empleados de la tabla emp que son analistas.

DECLARE

v_incre_sal emp.sal%TYPE := 2000;

BEGIN

UPDATE emp

SET sal = sal + v_incre_sal

WHERE job = 'ANALYST';

.....

END;



Borrado de datos en PL/SQL

Suprimir los empleados que trabajan en el DPTO 10.

DECLARE

v_deptno dept.deptno%TYPE;

BEGIN

DELETE FROM emp

WHERE deptno = v_deptno;

END;



Estructuras de Control en PL/SQL

Sentencias IF

Sintaxis:

```
IF condicion THEN  
    instrucciones;  
[ELSEIF condicion THEN  
    instrucciones;  
[ELSE  
    instrucciones;  
END IF;
```



Estructuras de Control en PL/SQL

Sentencias CASE

Similar al switch de C, evalúa cada condición hasta encontrar alguna que se cumpla. La sintaxis es:

```
CASE [expresion]
WHEN [condicion1|valor1] THEN
    bloque_instrucciones_1
WHEN [condicion2|valor2] THEN
    bloque_instrucciones_2
....
ELSE
    bloque_instrucciones_por_defecto
END CASE;
```



Estructuras de Control en PL/SQL

Sentencias CASE. Ejemplos

```
CASE
WHEN v_sal<1000 THEN
    v_sal:=v_sal+100;
WHEN v_sal>2000 THEN
    v_sal:=v_sal-100;
END CASE;
```

```
CASE tipo
WHEN 1 THEN
    procesar_pedido_local;
WHEN 2 THEN
    procesar_pedido_domicilio;
WHEN 3 THEN
    procesar_pedido_extranjero;
ELSE
    procesar_pedido_normal;
END CASE;
```




Condiciones booleanas

*¿Cuál es el valor de **v_result** en cada caso?*

v_result := v_vari1 AND v_vari2;

v_result	v_vari1	v_vari2
TRUE	TRUE	TRUE
FALSE	FALSE	FALSE
NULL	TRUE	NULL
FALSE	FALSE	NULL

Estructuras repetitivas en PL/SQL

- Los bucles repiten una sentencia o un grupo de sentencias varias veces.
 - Hay 3 tipos de bucles:
 - Bucle básico. **LOOP**. Acciones repetitivas sin condiciones globales. Como si no existiera.
 - Bucle **FOR**. Acciones repetitivas basándose en un contador. Número conocido de vueltas.
 - Bucle **WHILE**. Basándose en una condición.
- 

Estructuras repetitivas en PL/SQL

Bucle WHILE

```
WHILE condición LOOP  
    instrucciones;  
    .....  
END LOOP;
```

Se utiliza este bucle para repetir sentencias mientras una condición sea cierta.



Estructuras repetitivas en PL/SQL

Bucle WHILE. Ejemplo.

DECLARE

v_contador binary_integer := 0;

BEGIN

WHILE v_contador <= 10 LOOP

INSERT INTO prueba (id, contador)

VALUES (v_id, v_contador);

v_contador := v_contador + 1;

END LOOP;

COMMIT;

END;



Estructuras repetitivas en PL/SQL

Bucle FOR

```
FOR indice IN [REVERSE] valor_inicial .. valor_final LOOP
    instrucciones;
    .....
END LOOP;
```

- Usar bucle FOR para n° fijo de repeticiones.
- El índice se declara implícitamente.



Estructuras repetitivas en PL/SQL

Bucle FOR

- El índice fuera del bucle no está definido, por lo tanto no se puede referenciar.
- Usa una expresión para hacer referencia al valor actual de un índice.
- No hagas uso de un índice como objetivo de una asignación.



Estructuras repetitivas en PL/SQL

Bucle FOR. Ejemplo

DECLARE

valor_inicial

NUMBER := 1;

valor_final

NUMBER := 100;

FOR i **IN** valor_inicial .. valor_final **LOOP**

.....

END LOOP;



Estructuras repetitivas en PL/SQL

Bucle FOR. Ejemplo

Inserta las 10 primeras filas del pedido 601.

DECLARE

v_id prueba.id%TYPE := 601;

BEGIN

.....

FOR i IN 1 .. 10 LOOP

 INSERT INTO prueba (id, contador)

 VALUES (v_id, i);

END LOOP;

.....

END;



Programación modular en PL/SQL

En PL/SQL se pueden escribir cuatro tipos de bloques de código:

- **Bloques anónimos:** No se almacenan en la BD. Se ejecutan tras escribirlos.
- **Procedimientos:** Se almacenan en el DD y son invocados. Pueden recibir y devolver múltiples parámetros.
- **Funciones:** Se almacenan en el DD y son invocadas. Pueden recibir parámetros y devuelven un valor “en su nombre”.
- **Triggers:** Se almacenan en el DD y se ejecutan automáticamente cuando ocurre algún evento.

Procedimientos en PL/SQL

CREATE OR REPLACE PROCEDURE [esquema].nomproc
(nombre-parámetro {IN, OUT, IN OUT} tipo de dato, ..)
{IS, AS}

Declaración de variables;
Declaración de constantes;
Declaración de cursores;

BEGIN

Cuerpo del subprograma PL/SQL;

EXCEPTION

Bloque de excepciones PL/SQL;

END;



Funciones en SQL

```
CREATE OR REPLACE FUNCTION [esquema].nombre-funcion  
    (nombre-parámetro tipo-de-dato, ..)  
    RETURN tipo-de-dato  
{IS, AS}
```

Declaración de variables;
Declaración de constantes;
Declaración de cursores;

```
BEGIN  
    Cuerpo del subprograma PL/SQL;
```

```
EXCEPTION  
    Bloque de excepciones PL/SQL;
```

```
END;
```



Aclaraciones sintaxis procedimientos y funciones

Nombre-parámetro: es el nombre que nosotros queramos dar al parámetro. Podemos utilizar múltiples parámetros. En caso de no necesitarlos podemos omitir los paréntesis.

IN: especifica que el parámetro es de entrada y que por tanto dicho parámetro tiene que tener un valor en el momento de llamar a la función o procedimiento. Si no se especifica nada, los parámetros son por defecto de tipo entrada.

OUT: especifica que se trata de un parámetro de salida. Son parámetros cuyo valor es devuelto después de la ejecución el procedimiento al bloque PL/SQL que lo llamó. Las funciones PLSQL no admiten parámetros de salida.

IN OUT: Son parámetros de entrada y salida a la vez.



Ejemplo de procedimiento PL/SQL

```
CREATE OR REPLACE PROCEDURE procedimiento1
    (a IN NUMBER, b IN OUT NUMBER)
IS
    vmax NUMBER;
BEGIN
    SELECT salario, maximo INTO b, vmax
    FROM empleados
    WHERE empleado_id=a;
    IF b < vmax THEN
        b:=b+100;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        b:= -1;
        RETURN;
END;
```



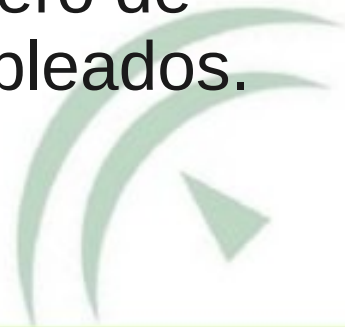
Ejemplo de llamada a un procedimiento PL/SQL

```
DECLARE
    vsalario NUMBER;
BEGIN
    procedimiento1 (3213, vsalario);
    dbms_output.put_line ('El salario del empleado 3213
es ', vsalario);
END;
```



Actividades

- Realiza un procedimiento *mostrar_defts* que reciba un nombre de usuario y muestre el nombre de su tablespace por defecto.
- Realiza un procedimiento que reciba un número de departamento y muestre su nombre y localidad.
- Realiza una función *devolver_sal* que reciba un número de empleado y devuelva su salario.
- Realiza un procedimiento que reciba un número de departamento y muestre una lista de sus empleados.
¿Que problema tienes?




Cursores

Tipos

Los **cursores** son áreas de memoria que almacenan datos extraídos de la base de datos. Hay dos tipos:

Implícitos: Declarados implícitamente para todas las sentencias del DML y SELECT de PL/SQL, consultas que devuelven una sola fila.

Explícitos: Declarados y nombrados por el programador. Manipulados por sentencias específicas en las instrucciones ejecutables del bloque. Se usan para consultas que devuelven más de 1 fila.



Cursores Implícitos

a) **SELECT** de una sola fila.

Si no devuelve ninguna fila o devuelve más de una se origina un error.

Lo controlaremos en la zona **EXCEPTION**, mediante los manejadores :

NO_DATA_FOUND Y **TOO_MANY_ROWS**.

b) **Sentencias** del **DML**, de más de una fila.



Cursores Explícitos

Declaración de Cursor

Sintaxis:

```
CURSOR      nombre_cursor      IS  
Sentencia select ;      /* sin INTO */
```

Ejemplo:

```
DECLARE  
    v_empno    emp.empno%TYPE;  
    v_ename    emp.empno%TYPE;  
  
    CURSOR emp_cursor IS  
    SELECT empno, ename  
    FROM emp  
    WHERE deptno = 30;
```

.....



Cursores Explícitos

Apertura del cursor

- Al abrir el cursor se ejecuta realmente la consulta.
- Si la consulta no devuelve ninguna fila no se producirá ninguna EXCEPTION.
- Hay que utilizar los atributos del cursor para comprobar los resultados obtenidos tras una recuperación.

OPEN nombre_cursor;



Cursores Explícitos

Recuperar datos del cursor

FETCH nombre_cursor **INTO** [var1,var2,...] | nom_registro;

- Recuperar la fila actual e introducir los valores en las variables de salida.
- Incluir el mismo nº de variables.
- Relacionar posicionalmente variables y columnas.
- Comprobar si el cursor tiene filas



Cursores Explícitos

Cierre del Cursor

Sintaxis:

CLOSE nombre_cursor

- Desactiva el cursor y libera la memoria reservada.
- Cerrar cursor tras el procesamiento de las filas.
- Volver a abrir el cursor si fuese necesario. Hay un máximo número de cursores que pueden estar abiertos a la vez en la BD.
- No se pueden recuperar datos del cursor una vez cerrado.



Cursores Explícitos

Ejemplo

```
DECLARE
    CURSOR emp_cursor IS
        SELECT empno, ename FROM emp;

BEGIN
    OPEN emp_cursor;
    FETCH emp_cursor INTO v_empno, v_ename;
    WHILE emp_cursor%FOUND LOOP
        // Procesar información
        FETCH emp_cursor INTO v_empno, v_ename;
    END LOOP;
    CLOSE emp_cursor;
END;
```



Cursores explícitos

Atributos

%ISOPEN	booleano	TRUE si está abierto
%NOTFOUND	booleano	TRUE si la recuperación más reciente no devuelve fila
%FOUND	booleano	TRUE si la recuperación más reciente devuelve fila
%ROWCOUNT	número	nº de filas devueltas hasta ese momento



Cursores explícitos

Apertura. Comprobación

```
BEGIN
```

```
IF NOT cursor_emp%ISOPEN THEN
```

```
    DBMS_OUTPUT.PUT_LINE ('El cursor está  
                           cerrado. Vamos a abrirlo');
```

```
    OPEN cursor_emp;
```

```
END IF;
```



Cursores explícitos

Cierre del cursor

```
CLOSE cursor_emp;  
END;
```



Cursores explícitos

Cursor con variable registro


```
DECLARE
    CURSOR emp_cursor IS
        SELECT ename, sal FROM emp WHERE deptno = 10;
    v_reg_cursor      emp_cursor%ROWTYPE;
BEGIN
    OPEN emp_cursor;    // Abrir cursor
    FETCH emp_cursor INTO v_reg_cursor; // Leer primera fila

    WHILE emp_cursor%FOUND LOOP // mientras haya filas

        DBMS_OUTPUT.PUT_LINE(v_reg_cursor.ename||' * '||
                               v_reg_cursor.sal); // procesar

        FETCH emp_cursor INTO v_reg_cursor; // leer siguiente

    END LOOP;
    CLOSE emp_cursor; // cerrar cursor
    // presentar resultados finales (si procede)
END;
```



Cursores explícitos

Bucles FOR de cursor

```
FOR nombre_registro IN nombre_cursor LOOP  
    instrucción_1;  
    instrucción_2;  
    ....  
END LOOP;
```

- Apertura, recuperación y cierre implícitos.
- La variable registro se declara implícitamente



Cursores explícitos

Bucle FOR de cursor.Ejemplo

```
DECLARE
    CURSOR emp_cursor IS
        SELECT ename, sal, TRUNC((SYSDATE – hiredate)/365) as antigüedad
        FROM emp
        WHERE deptno = 10;
BEGIN
    FOR v_reg_cursor IN emp_cursor LOOP
        DBMS_OUTPUT.PUT_LINE(v_reg_cursor.ename || ' * ' ||
                               v_reg_cursor.sal || ' * ' ||
                               v_reg_cursor.antigüedad);
    END LOOP;
END;
```



Excepciones en PL/SQL

- ¿**Qué es una excepción?**
 - Identificador PL/SQL que surge durante la ejecución provocado por un error.
- ¿**Cómo surge?**
 - Se produce un error Oracle.
 - Es provocado explícitamente por el programador.
- ¿**Como se gestiona?**
 - Tratándola con un manejador en el propio bloque.
 - Propagándola al proceso padre.



Excepciones en PL/SQL

Gestión de excepciones en PL/SQL

Captura de una excepción:

Cuando se produce una excepción en la sección ejecutable, esta se ramifica a la zona de EXCEPTION, donde se puede capturar y tratar programando su correspondiente manejador.

Propagación de una excepción:

Cuando se produce una excepción en un bloque y no hay el correspondiente manejador para ella, el bloque termina con un error. Podremos gestionarla en el entorno de llamada de dicha bloque.



Excepciones en PL/SQL

Tipos de excepciones.

- Del servidor Oracle. Provocadas implícitamente.
 - Con nombre. Ej: NO_DATA_FOUND
 - Sin nombre. ORA-12560
- Definidas por el usuario. Provocadas explícitamente
 - Con nombre. (se declaran previamente)
 - Sin nombre. (RAISE_APPLICATION_ERROR)



Excepciones en PL/SQL

Manejadores de excepciones. Sintaxis.

EXCEPTION

WHEN exception1 [**OR** exception2...] **THEN**

instruccion1;

instruccion2;

[.....]

WHEN OTHERS THEN

instruccion3;

instruccion4;]



Excepciones en PL/SQL

Reglas para interrumpir excepciones

- EXCEPTION, inicia la sección de gestión de excepciones.
- Se permiten varios manejadores de excepciones.
- Solo se procesa un manejador de excepciones antes de salir del bloque.
- WHEN OTHERS es la última cláusula



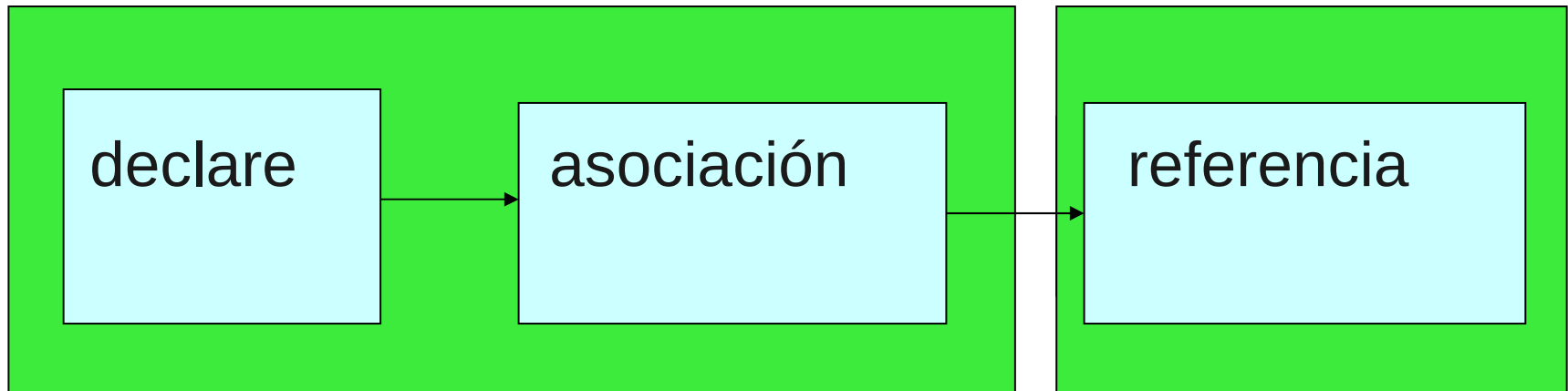
Excepciones en PL/SQL

Predefinidas por Oracle y con nombre. Ejemplos.

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

Excepciones en PL/SQL

Predefinidas de ORACLE, sin nombre. Método 1



sección declarativa

declarar
excepción

codificar PRAGMA
EXCEPTION_INIT

sec. gest. excepciones

gestionar la excepción

Excepciones en PL/SQL

Predefinidas de ORACLE, sin nombre. Método 1.Ejemplo

DECLARE

// declaración y asociación de la excepción

e_restri_integri_emp EXCEPTION;

PRAGMA EXCEPTION_INIT(e_restri_integri_emp, -2292);

v_deptno dept.deptno%type:=10;

BEGIN

DELETE FROM dept WHERE deptno = v_deptno;

COMMIT;

EXCEPTION

WHEN e_restri_integri_emp THEN

 dbms_output.put_line('no se puede borrar depto,
 existen empleados');

END;



Excepciones en PL/SQL

Predefinidas de ORACLE, sin nombre. Método 2


Cuando se produce una excepción sin nombre, hay dos pseudovariantes que almacenan información del error:

SQLCODE: N° del código del error.

SQLERRM: Mensaje asociado al n° del error.

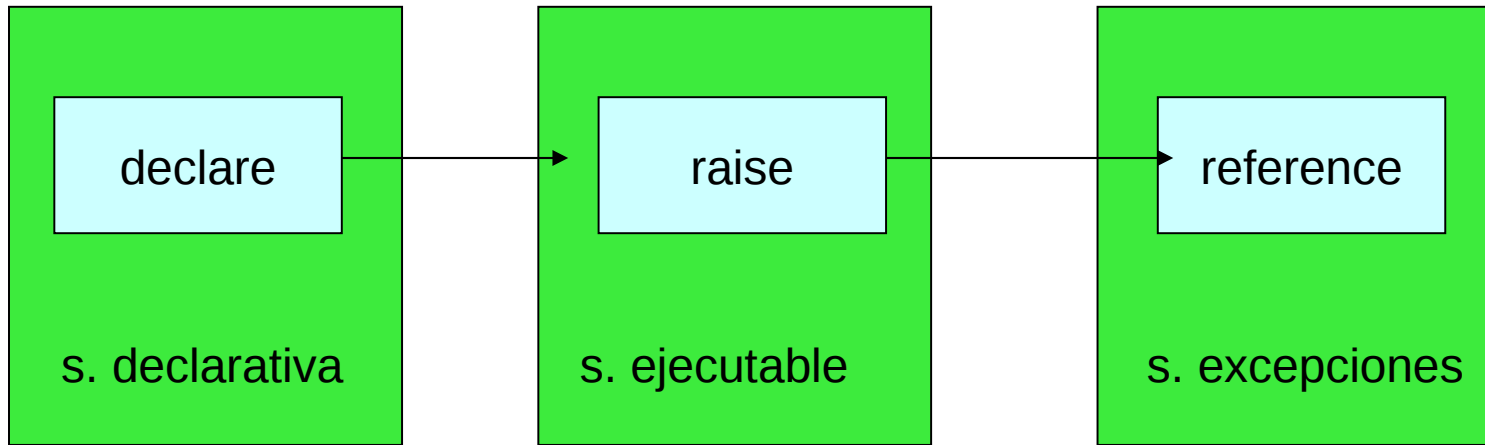
Así, se pueden gestionar en el manejador WHEN OTHERS:

```
WHEN OTHERS THEN
  IF SQLCODE= - 2292 THEN
    dbms_output.put_line ('Error de integridad referencial');
  END IF;
```



Excepciones en PL/SQL

Definidas por el usuario, con nombre



Se especifica la
excepción

Se provoca
explícitamente
la excepción
con RAISE

Se gestiona la
excepción



Excepciones en PL/SQL

Definidas por el usuario, con nombre. Ejemplo

DECLARE

 e_invalid_product EXCEPTION;

BEGIN

 UPDATE product

 SET descript = p_descripcion_product

 WHERE prodid = p_product_number;

 IF SQL%NOTFOUND THEN

 RAISE e_invalid_product;

 END IF;

 COMMIT;

EXCEPTION

 WHEN e_invalid_product THEN

 DBMS_OUTPUT.PUT_LINE('nº del producto invalidado');

END;



Excepciones en PL/SQL. Definidas por el usuario, sin nombre RAISE_APPLICATION_ERROR

- Procedimiento que permite emitir mensajes de error definidos por el usuario desde subprogramas almacenados.
- Solo se puede llamar desde un subprograma almacenado.
- Se usa en la sección ejecutable o en la de excepciones.
- Devuelve condiciones de error al usuario de forma consistente con otros errores del servidor Oracle.

Sintaxis:

`RAISE_APPLICATION_ERROR(num_error, mensaje);`
donde num_error entre -20000 y -20999

Excepciones en PL/SQL Definidas por el usuario, sin nombre RAISE_APPLICATION_ERROR. Ejemplo.

.....

EXCEPTION

```
WHEN NO_DATA_FOUND THEN
```

```
    RAISE_APPLICATION_ERROR (-20201, 'manager, no es un  
    empleo válido,');
```

.....

```
DELETE FROM emp
```

```
WHERE mgr = v_mgr;
```

```
IF SQL%NOTFOUND THEN
```

```
    RAISE_APPLICATION_ERROR(-20202, 'este no es un jefe válido');  
END IF;
```

.....

