Tipos de Datos Compuestos

TIPOS DE DATOS COMPUESTOS

- Son estructuras de datos formadas de otras más simples
- También son conocidos como registros y “collections”
- Tipos:
  - REGISTROS PL/SQL
  - Collections PL/SQL: Varray, tabla anidada y tablas indexadas
- Contienen componentes internos
- Son reutilizables

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos

TIPOS DE DATOS COMPUESTOS

- **REGISTRO**: grupo de elementos de datos almacenados en campos, cada uno con su propio nombre y tipo de datos, pero todos relacionados.

  Ej: datos acerca de un empleado: nombre, sueldo, fecha_contratación.

Nombre	Sueldo	Fecha_contratacion

- **COLLECTIONS (arrays)**: estructuras de datos compuestos por lista de elementos. Se utilizan para guardar datos en forma de múltiples filas similar a las tablas de BD. Pueden ser:

  Varrays, tablas anidadas, tablas indexadas

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

REGISTROS

- Esta formado por uno o más componentes llamados campos de cualquier tipo de datos escalar, REGISTRO o TABLA PL/SQL; y no tiene por que coincidir con el registro de una tabla de la BD
- No es lo mismo que las filas de una tabla de la BD, aunque pueden coincidir
- Tratan una colección de campos como una unidad lógica, como un todo
- Son adecuados para recuperar una fila de datos de una tabla o una sentencia sobre varias tablas para su posterior procesamiento.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

REGISTROS

- Para declarar un tipo registro:
  - Puede tener tantos campos como sea necesario
  - Se puede asignar valores iniciales y pueden definirse como NOT NULL. Si no se inicializan su valor será NULL
  - Se pueden definir un registro (RECORD) en la parte declarativa de un bloque, subprograma o paquete.
  - Un registro puede formar parte de otro, formando lo que se denomina registro anidado. Ej. Un registro fecha dentro de otro registro
  - Un registro puede ser una variable cursor en una sentencia SELECT

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

REGISTROS: CREACIÓN

- Para crear un registro, se realiza en dos pasos:

  - **PASO 1:** Primero se define el tipo de dato REGISTRO del usuario

```
TYPE def_tipo_registro  IS  RECORD
                (nom_campo1 { tipo_dato}  [ [NOT NULL] {:= | DEFAULT} expr],
                 nom_campo2 { tipo_dato}  [ [NOT NULL] {:= | DEFAULT} expr],
                 nom_campo3 { tipo_dato}  [ [NOT NULL] {:= | DEFAULT} expr],
                 .....
                );
```

  - **PASO 2**: Se define una variable de es nuevo tipo de dato

```
v_nueva                                 def_tipo_registro;
```

NOTA: Es una declaración, por lo tanto se define en el DECLARE o en el IS/AS de los PROCEDURE y FUNCTION

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

REGISTROS: CREACION

```
DECLARE
....
  TYPE def_tipo_emple_reg        IS   RECORD (
            nombre    varchar2(10),
            oficio      emple.oficio%TYPE,
            sal          number(8,2) not null :=0
  );
.....
  var_emple_reg       def_tipo_emple_reg;
```

Defino un nuevo tipo de datos

Defino una variable del nuevo tipo de dato definido en el TYPE

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

REGISTROS: ACCESO A LOS CAMPOS

Para hacer referencia a un campo:

nom_registro . nom_campo

Ejemplo:

```
emple_reg.nombre:='Roberto';
dmbs_output.put_line('El nombre del empleado es: ' || emple_reg.nombre);
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

REGISTROS: ATRIBUTO %ROWTYPE

- Declara una variable de acuerdo con una colección de columnas de una vista o tabla de base de datos
- Los campos del registro toman sus nombres y tipos de datos de las columnas de la vista o la tabla
- Es útil para recuperar una fila de la base de datos.

Ejemplo:

```
DECLARE
    emple_reg    emple%ROWTYPE;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

EJERCICIOS

- **Crear un procedimiento que permita visualizar "TODOS" los datos de un empleado a partir del emp_no que introducirá el usuario en el programa principal. Definir en el procedimiento una variable tipo registro llamada reg_emple utilizando el ROWTYPE. La salida será**

  Numero empleado :
  Nombre :
  Oficio :
  Jefe :
  Salario:
  Comision :
  Fecha Alta:
  Departamento :

- **Modificar el ejercicio anterior para que en Jefe, y Departamento aparezcan el nombre del Jefe y el Nombre del departamento.**

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

EJERCICIOS

- **Crear** un procedimiento basado en el ejercicio 1 para que una vez obtenido los datos del empleado, se inserte en la tabla EMPLE un nuevo registro donde:
  - Numero empleado: será el siguiente al mayor de la tabla
  - El nombre, código departamento, oficio y jefe serán los mismos que los de el empleado de la consulta
  - El salario será el salario del empleado del que se ha realizado la consulta +1 euro
  - No tendrá comisión.
  - Fecha de alta será la fecha actual

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

EJERCICIOS

- Modificar o realizar otro procedimiento de forma que introducido el emp_no en el programa principal cargue en el registro definido por el usuario, llamado *mi_tipo_registro* , todos los campos de la tabla emple, y el nombre del departamento; utilizando para ellos como nombre de los campos del registro

  Los campos de mi_tipo_registro serán :

  num_emp
  apel
  ofi
  jefe
  alta
  salario
  comision
  departamento
  nom_departamento

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

EJERCICIOS

- Realizar un procedimiento que a partir de un número de empleado introducido por teclado visualice el emp_no, apellido, oficio y nombre del departamento del empleado. Utilizar un registro definido por el usuario que contenga estos cuatro campos.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

EJERCICIOS

- Crear una tabla llamada T_DATOS con la siguiente información:
  - Emp_no
  - Nombre
  - Sueldo

- Crear un bloque PL/SQL que una vez introducido un número de empleado, localice dicho empleado en la tabla EMPLE e inserte en la tabla T_DATOS la información correspondiente al emp_no, apellido, sueldo. Utilizar un registro

NOTA: NO IMPORTA QUE SE INSERTEN VALORES REPETIDOS, para evitar esto se debería definir clave primaria y utilizar las EXCEPTION que se verán más adelante**.**

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Registros

EJERCICIOS

```
accept cod prompt 'Introduce el codigo de empleado:';
DECLARE
      -- defino mis tipos de datos especiales
      TYPE def_t_datos IS RECORD(
                  emp_no            emple.emp_no%type,
                  apellido          emple.apellido%type ,
                  sueldo            number(11)
   );
   --defino las variables con sus tipos de datos
   reg_t_datos                      def_t_datos;
begin
    select emple.emp_no,emple.apellido,(emple.salario+nvl(emple.comision,0)) into
                  reg_t_datos.emp_no, reg_t_datos.apellido, reg_t_datos.sueldo from emple   where
   emple.emp_no=&cod;
 insert into t_datos values (reg_t_datos.emp_no, reg_t_datos.apellido, reg_t_datos.sueldo);
    commit;
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS

- Están compuestas por lista de elementos.
- Se usan para guardar datos en formato de múltiples filas similar a las BD
- Pueden ser:
  - Varrays
  - Tablas anidadas
  - Tablas indexadas
- Todas ellas son listas de una dimensión, aunque sus elementos pueden ser compuestos

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

VARRAYS

- Son equivalentes a los arrays (tablas de una dimensión) de los lenguajes tradicionales
- Se puede acceder a ellas tanto desde SQL como desde PL/SQL
- Tienen un índice secuencial que permite el acceso a sus elementos y siempre comienza en 1
- En el momento de su creación se le debe de asignar una longitud fija

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

VARRAYS. CREACIÓN

- Se declaran en la zona declarativa del bloque (DECLARE) o subprograma (AS)
- El proceso se realiza en dos fases:
  - Declaración de un tipo de datos VARRAY (TYPE……IS VARRAY ….)
  - Declarar una variable de ese tipo de dato
  - Inicializar cargando valores
- No se puede inicializar una tabla en su declaración

SINTAXIS:

-- definimos mi tipo dato varray

```
TYPE nom_tipo_varray IS VARRAY(num_elementos) OF tipo_datos  [NOT NULL];
```

-- declaramos una variable de ese nuevo tipo de dato tabla e inicializamos

```
nom_variable_varray      nom_tipo_varray;
nom_variable_array       nom_tipo_varray := nom_tipo_array( valor1, valor2,…)
```

también podemos declararla en la parte declarativa e inicializar en el cuerpo

```
nom_variable_array  := nom_tipo_array( valor1, valor2,…)
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

VARRAYS. CREACIÓN

-- también podemos declarar una variable VARRAY e inicializarla como una lista vacía

```
nom_variable_array      nom_tipo_varray := nom_tipo_array( );
```

para asignar un valor, será necesario entonces, añadir un elemento usando el método EXTEND y a continuación asignarle un valor

```
nom_variable_varray.EXTEND;                 --añado una fila a la lista
nom_variable_varray( posicion ) := valor;   --asigno un valor a la fila añadida
```

NOTA: no se puede añadir más filas (EXTEND) que las definidas en la dimensión del VARRAY

-- para acceder a un elemento

```
nom_variable_varray( posicion )
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

VARRAYS. CREACIÓN

- Ejemplo:

```
DECLARE
    TYPE t_meses IS VARRAY (12) OF  varchar2(10);   -- defino nuevo tipo de datos
    v_tabla_meses       t_meses;                    -- declaro la variable

BEGIN
   -- inicializo la variable tabla
  v_tabla_meses := t_meses('ENERO', 'FEBRERO' ,……,'DICIEMBRE');

   for i IN 1..12 LOOP
            DMBS_OUTPUT.PUT_LINE( TO_CHAR(i) || ' – ' || v_tabla_meses(i));
   END LOOP;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

VARRAYS. CREACIÓN

- Ejemplo:

```
DECLARE
     TYPE tipo_tabla IS VARRAY(10) OF  number;   -- defino nuevo tipo de datos

   -- declaro e inicializo la  variable con nulos
   v_tabla       tipo_tabla:=tipo_tabla(null,null,null,null,null,null,null,null,null,null);

BEGIN
  for i IN 1..10 loop
                v_tabla(i):=i;
   end loop;

  for i IN 1..10 LOOP
                DMBS_OUTPUT.PUT_LINE( 'v_tabla[' || i || ']= ' || v_tabla(i));
   END LOOP;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

VARRAYS. CREACIÓN

- Ejemplo OTRA SOLUCIÓN, definiendo una **lista vacía**

```
DECLARE
    TYPE tipo_tabla IS VARRAY(10) OF  number;   -- defino nuevo tipo de datos
    v_tabla       tipo_tabla:=tipo_tabla();     -- declaro la  variable e inicializo la tabla sin elementos
                                                -- como una lista vacia

BEGIN
  for i IN 1..10 loop
                v_tabla.extend;     -- añado una fila nueva a la variable v_tabla. No puedo añadir mas filas
                                    -- que las especificadas en el tamaño del array, en este caso 10
                v_tabla(i):=i;       -- asigno un valor a esa fila
   end loop;

  for i IN 1..10 LOOP
                DMBS_OUTPUT.PUT_LINE( 'v_tabla[' || i || ']= ' || v_tabla(i));
   END LOOP;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS ANIDADAS

- Son muy similares a las VARRAYS
- Comparten muchas características estructurales y funcionales (lista de elementos, índice para acceso, posibilidad de uso en SQL y PL/SQL)
- Se diferencias de las VARRAYS en que las tablas anidadas **NO** tienen una longitud fija

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS ANIDADAS. CREACIÓN

- Se declaran en la zona declarativa del bloque (DECLARE) o subprograma (AS)

- El proceso se realiza en dos fases:
  - Declaración de un tipo de datos TABLE (TYPE ….IS TABLE ….)
  - Declarar una variable de ese tipo de dato
  - Inicializar cargando valores

SINTAXIS:

-- definimos el mi tipo dato table

```
TYPE    mitipo_tabla_anidada  IS  TABLE   OF tipo_datos  [NOT NULL];
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS ANIDADAS. CREACIÓN

--declaramos y si queremos inicializamos una variable de ese nuevo tipo de dato tabla

```
var_tablaanidada   mitipo_tabla_anidada;  --solo declara no inicializa
var_tablaanidada   mitipo_tabla_anidada:= mitipo_tabla_anidada( ); --inicializa lista vacía
```

--para añadir una nueva fila se utiliza el método EXTEND aplicándolo a la tabla y a continuación se introduce el valor en la nueva posición de la tabla

```
var_tablaanidada .EXTEND;
var_tablaanidada (pos) := valor;
```

-- para acceder a un elemento

```
var_tablaanidada( posicion )
```

Otra forma de inicializar, una vez declarada:

--declaramos una variable de ese nuevo tipo de dato tabla

```
var_tablaanidada   mitipo_tabla_anidada ;
```

--inicializamos la tabla

```
var_tablaanidada := mitipo_tabla_anidada( valor1,….,valor n);
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS ANIDADAS. CREACIÓN

- Ejemplo:

```
DECLARE
        TYPE t_meses    IS TABLE  OF  varchar2(10); --defino el tipo de dato
        v_tabla_meses    t_meses ;    --solo declaro la variable
BEGIN
  v_tabla_meses := t_meses('ENERO', 'FEBRERO' ,……,'OCTUBRE');
  v_tabla_meses.EXTEND;
  v_tabla_meses(11):='NOVIEMBRE';
  v_tabla_meses.EXTEND;
  v_tabla_meses(12):='DICIEMBRE';
  for i IN 1..12 LOOP
        DBMS_OUTPUT.PUT_LINE( TO_CHAR(i) || ' – ' || v_tabla_meses(i));
  END LOOP;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS INDEXADAS

- Son similares a las anteriores
- No pueden usarse en tablas de las bases de datos
- No pueden manipularse con comando SQL, solo con PL/SQL
- No tienen una longitud determinada
- No requieren inicialización
- Todos los elementos se crean dinámicamente
- ***El índice NO es secuencial***, pues utiliza un índice que no tiene por que tomar valores consecutivos
- Están formadas por dos componentes:
  - **TIPO DE DATOS DE CLAVE PRIMARIA** BINARY_INTEGER, que indexa la tabla
  - **COLUMNA DE TIPOS DE DATOS ESCALARES O DE REGISTRO**, que almacena los elementos de la tabla
- Para recorrer la tabla no conviene utilizar un bucle FOR pues no se garantiza que todos los elementos de la tabla estén almacenados de forma secuencial por el índice.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS INDEXADAS. CREACIÓN

- Se declaran en la zona declarativa del bloque, subprograma.
- El proceso se realiza en dos fases:
  - Declaración de un tipo de datos TABLA (TYPE ...IS TABLE OF…INDEX BY…)
  - Declarar una variable de ese tipo de dato
- No requieren inicialización

SINTAXIS:

-- definir el tipo dato tabla anidada

```
TYPE mitipo_tabla IS TABLE OF   { tipo_datos_tabla}  [NOT NULL]
     INDEX BY [ BINARY_INTEGER  |  PLS_INTEGER  | VARCHAR2 (long) ];
```

--declarar de una variable de ese nuevo tipo de dato tabla

```
          var_nom_tabla   mitipo_tabla;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS INDEXADAS. CREACIÓN

-- Para hacer referencia a los elementos

var_nom_tabla (indice)

Nota: en las tablas indexadas, las variables de este tipo de tabla no requieren inicialización y tampoco reservar memoria para nuevos elementos, basta con introducir un valor indicando el índice del elemento donde se quiere introducir

Para recorrer una tabla indexada es mas interesante hacerlo con un while que con un for, pues con este último se hace un recorrido secuencial del índice

```
pos := variabletabla.FIRST;
While pos IS NOT NULL loop
   .....
   pos:=variabletabla.next(pos)
end loop;
```

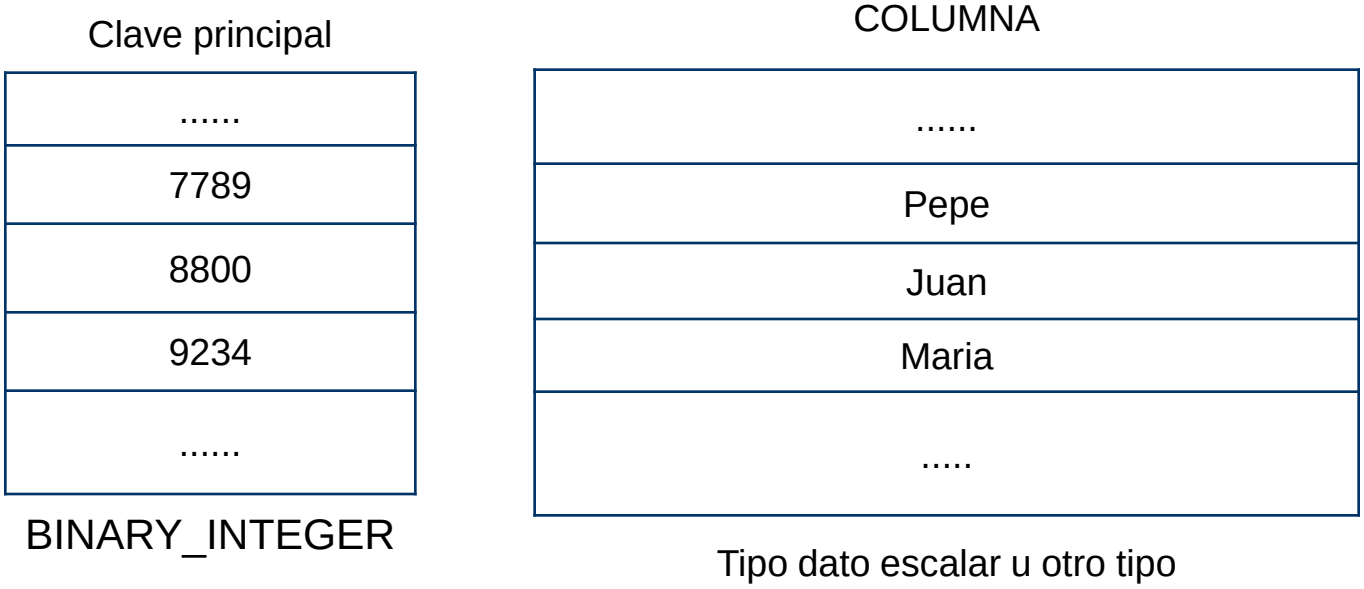
SI los elementos de la tabla son de tipo registro, recordar utilizar la notación:

var_nom_tabla (indice). Campo

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

ESTRUCTURA



Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS INDEXADAS

Ejemplo

```
declare
   type t_tabla_emple IS TABLE OF emple%ROWTYPE  index by binary_integer;
   tab_emple   t_tabla_emple;
….
BEGIN
   …
   select * into tab_emple(7900) where emp_no = 7900;
   …
   dbms_ouput.put_line(tab_emple(7900).apellido);
   tab_emple(7900).salario := 900;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS INDEXADAS

Ejemplo

```
declare
   type t_tabla_emple IS TABLE OF emple%ROWTYPE  index by binary_integer;
   tab_emple   t_tabla_emple;
   CURSOR c_emple IS SELECT * from emple;
BEGIN
   for v_reg IN c_emple loop
                tab_emple(v_reg.emp_no) :=v_reg;
   end loop;
   …;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

METODOS PARA COLLECTIONS

- Son funciones que facilitan la utilización de las tablas.
- Sixtansis:

Nom_variable_tabla.nombre_método [(parámetros)]

METODO	DESCRIPCION
EXISTS(n)	Retorna TRUE si en la posición “n” de la tabla existe un elemento
COUNT	Retorna el número de elementos que contiene la tabla
FIRST LAST	Retorna el primer y último índice numérico en una tabla y NULL si está vacía
PRIOR(n)	Retorna el índice numérico que precede al índice “n”

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

METODOS PARA COLLECTIONS

METODO	DESCRIPCION
NEXT(n)	Retorna el índice numérico que sigue al índice “n”
EXTEND(n,i) (no válido para indexadas)	Incrementa el tamaño de la tabla. EXTEND: añade un elemento nulo a la tabla EXTEND(n): añade “n” filas de nulos EXTEND(n,i): añade “n” filas del elemento i
TRIM	Elimina el elemento del final de la tabla
DELETE (no valido para varray)	DELETE elimina todos los elementos de una tabla DELETE(n) elimina el elemento “n” de la tabla DELETE(m,n) elimina los elementos entre “m” y “n”

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

METODOS PARA COLLECTIONS

Ejemplos de la utilidad de los métodos:

- PRIOR y NEXT se pueden utilizar para recorrer una tabla (en cualquiera de los dos sentidos, aunque hay que tener cuidado ya que el PRIOR del primer elemento es NULL igual que el NEXT al último elemento.
- Igual para FIRST y LAST

```
.....
i:=variable_tabla.FIRST;
WHILE i IS NOT NULL LOOP
        ....
        i:=variable_tabla.NEXT(i);
END LOOP;
.....
```

```
FOR i IN variable_tabla.FIRST..variable_tabla.LAST LOOP
.....
END LOOP;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

TABLAS INDEXADAS

Ejemplo

```
declare
    type t_tabla_emple IS TABLE OF emple%ROWTYPE  index by binary_integer;
    tab_emple   t_tabla_emple;
   CURSOR c_emple IS SELECT * from emple;
   pos number;
BEGIN
    for v_reg IN c_emple loop
                    tab_emple(v_reg.emp_no) :=v_reg;
   end loop;

   pos := tab_emple.first;
   while pos is not null loop
     dbms_output.put_line (tab_emple(pos).emp_no || tab_emple(pos).apellido|| ……);
     pos:=tab_emple.next(pos);
   end loop ;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS

- Realizar la definición de una variable tipo tabla cuyo contenido números enteros
- Realizar un bloque PL/SQL que permita cargar los diez primeros números en la tabla. Visualizar después la tabla.
- Modificar el programa anterior para que visualice cuantos elementos tiene la tabla, cual es el primero y cual es el último.
- Modificar el bloque anterior para que la tabla se cargue con los 10 primeros números pares.
- Realizar un bloque PL/SQL que permita cargar los diez primeros números en la tabla. Visualizar después la tabla; a continuación eliminar de la tabla desde el elemento 4º al 6º y visualizar otra vez toda la tabla. Recordar que si se intenta visualizar un elemento de la tabla que no existe se produce un error.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS

- Realizar un bloque PL/SQL que permita cargar los diez primeros números en la tabla. Visualizar después la tabla

```
DECLARE
        TYPE t_numeros  IS TABLE  OF  number;
        v_tabla_num    t_numeros :=t_numeros();
BEGIN
   for i IN 1..10 LOOP
     v_tabla_num.EXTEND;
     v_tabla_num(i):=i;
   END LOOP;

   for i IN 1..v_tabla_num.last LOOP
        dbms_output.put_line('tabla ['||i || '] = ' || v_tabla_num(i));
   END LOOP;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS

- Realizar la definición de una variable tipo tabla indexada cuyo contenido será el nombre de los departamentos de la tabla DEPART

- Realizar un bloque PL/SQL que cargue en la tabla anterior los nombres de los departamentos de la tabla DEPART y una vez cargada la tabla la visualice.

  NOTA: Lo primero que se deberá de conocer en el bloque será el número de departamentos distintos que existen en la tabla.

- Modificar el programa anterior para que muestre, utilizando los **MÉTODOS PARA TABLAS**:
  - Número de elementos de la tabla
  - Cual es el primer elemento y cual es el último

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS. Solución

- Realizar un bloque PL/SQL que cargue en la tabla anterior los nombres de los departamentos de la tabla DEPART y una vez cargada la tabla la visualice.

  NOTA: Lo primero que se deberá de conocer en el bloque será el número de departamentos distintos que existen en la tabla.

- Modificar el programa anterior para que muestre, utilizando los **MÉTODOS PARA TABLAS**:
  - Número de elementos de la tabla
  - Cual es el primer elemento y cual es el último

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS. Solución (I)

```
DECLARE
   TYPE def_tabla_depart IS TABLE OF  depart.DNOMBRE%type
                                 INDEX BY BINARY_INTEGER;
    v_tabla_depart                 def_tabla_depart;
   total_depart                    number;

BEGIN
    select count(*) into total_depart from depart;
    --cargar tabla
    for i in 1..total_depart loop
                select dnombre into v_tabla_depart(i) from depart   where dept_no=10*i;
    end loop;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS. Solución (I)

```
--visualizar tabla
    for i in 1..total_depart loop  -- tambien for i IN v_tabla_depart.first..v_tabla_depart.last loop
        if v_tabla_depart.exists(i) then
                    dbms_output.put_line('departamento '||10*i||' => '||v_tabla_depart(i));
                  else
                    dbms_output.put_line('departamento '||10*i||' => vacia');
        end if;
    end loop;

    dbms_output.put_line('El número de elementos de la tabla es: '|| v_tabla_depart.count);
   dbms_output.put_line('El primer elemento de la tabla es: '||
                                                          v_tabla_depart(v_tabla_depart.first));
    dbms_output.put_line('El último de elemento de la tabla es: '||
                                                           v_tabla_depart(v_tabla_depart.last));
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS. Solución (II)

```
DECLARE
   TYPE def_tabla_depart IS TABLE OF  depart.DNOMBRE%type
                                 INDEX BY BINARY_INTEGER;
    v_tabla_depart              def_tabla_depart;
   total_depart                 number;
   i                            number;
BEGIN
    select count(*) into total_depart from depart;
    --cargar tabla
    for i in 1..total_depart loop
                select dnombre into v_tabla_depart(i) from depart   where dept_no=10*i;
    end loop;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

EJERCICIOS. Solución (parte II)

```
--visualizar tabla
 i:=v_tabla_depart.first;
 While i IS NOT NULL LOOP
    if v_tabla_depart.exists(i) then
              dbms_output.put_line('departamento '||10*i||' => '||v_tabla_depart(i));
            else
              dbms_output.put_line('departamento '||10*i||' => vacia');
    end if;
    i:=v_tabla_depart.next(i);
  end loop;
  dbms_output.put_line('El número de elementos de la tabla es: '|| v_tabla_depart.count);
  dbms_output.put_line('El primer elemento de la tabla es: '||
                                                  v_tabla_depart(v_tabla_depart.first));
  dbms_output.put_line('El último de elemento de la tabla es: '||
                                                   v_tabla_depart(v_tabla_depart.last));
END;
```

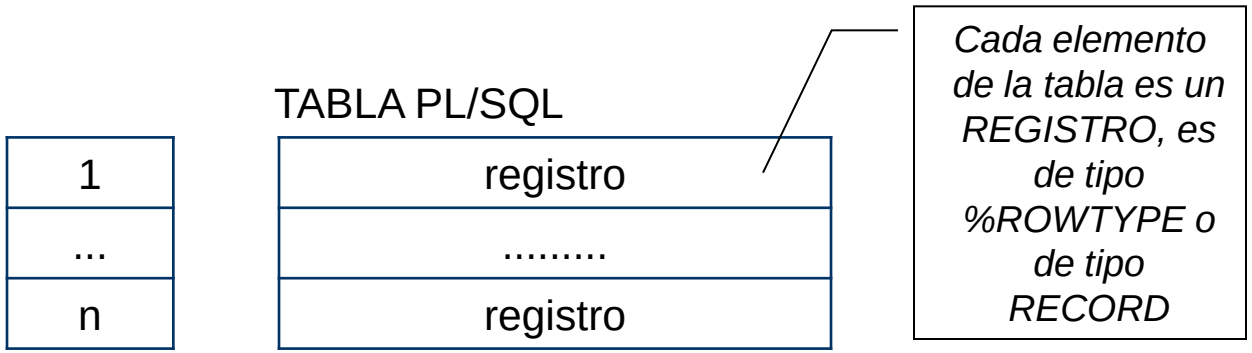
Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS PL/SQL DE REGISTROS

- Son tablas donde cada uno de los elementos que forman la tabla está formada por datos de tipo %ROWTYPE o tipo de tipo RECORD

- Permite aumentar la funcionalidad de las tablas de PL/SQL



Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS PL/SQL DE REGISTROS

```
DECLARE

- defino el nuevo tipo de dato tabla
TYPE def_tipo_tabla_depart IS TABLE OF depart%ROWTYPE
            INDEX BY BINARY_INTEGER;

- - defino la variable de ese tipo de dato tabla
  v_tabla_depart          def_tipo_tabla_depart;
```

Para acceder a un campo concreto de cada uno de los elementos de la tabla:

v_tabla_depart(posicion).campo;

Ejemplo: v_tabla_depart(5).salario

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS. EJERCICIOS

- Definir una tabla para almacenar las filas de la tabla de la base de datos DEPART

- Realizar un bloque PL/SQL que cargue en la tabla PL/SQL todas las filas de la tabla de la base de datos DEPART. Una vez cargada en memoria visualizar la tabla.
  - NOTAS de ayuda para solucionar el ejercicio:
    - La tabla se cargará fila a fila
    - ¿Como harías para saber cuantos elementos va a tener la tabla?
    - Sabiendo que el dept_no= 10,20,30,40,..., ¿cómo obtendrías todos los datos del departamento 10? ¿y los del 20? Y los del 30?....
    - ¿cómo almacerías los datos del dept_no 10 en la 1ª fila de la tabla?
    - ¿cómo almacerías los datos del dept_no 20 en la 2ª fila de la tabla?
    - ¿cómo almacerías los datos del dept_no 30 en la 3ª fila de la tabla?

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS. EJERCICIOS

- Diseñar un bloque que cargue en una tabla (anidada o indexada como se desee) el numero de departamento, nombre de departamento y los empleados que tiene dicho departamento. Una vez cargada, visualizar toda la tabla y mostrar el primero y el ultimo elemento de la tabla

  NOTAS:

  - Crear un cursor con la consulta
  - Crear un registro del mismo tipo que los campos de la consulta
  - Crear una tabla indexada que contendrá el registro

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS.EJERCICIOS. Solución (1/3)

- Realizar un bloque PL/SQL que cargue en la tabla PL/SQL todas las filas de la tabla de la base de datos DEPART. Una vez cargada en memoria visualizar la tabla.
  - NOTAS de ayuda para solucionar el ejercicio:
    - La tabla se cargará fila a fila
    - ¿Como harías para saber cuantos elementos va a tener la tabla?
    - Sabiendo que el dept_no= 10,20,30,40,..., ¿cómo obtendrías todos los datos del departamento 10? ¿y los del 20? Y los del 30?....
    - ¿cómo almacerías los datos del dept_no 10 en la 1ª fila de la tabla?
    - ¿cómo almacerías los datos del dept_no 20 en la 2ª fila de la tabla?
    - ¿cómo almacerías los datos del dept_no 30 en la 3ª fila de la tabla?

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS.EJERCICIOS. Solución (2/3)

```
DECLARE
      TYPE def_tipo_tabla_depart IS TABLE OF depart%ROWTYPE
                                                INDEX BY BINARY_INTEGER;

--Defino las variables
    v_tabla_depart                          def_tipo_tabla_depart;
    num_elem                                number;

BEGIN
     -- comprueb num de filas de la tabla depart
     SELECT COUNT(*) INTO num_elem FROM depart;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS.EJERCICIOS. Solución (3/3)

```
--cargo en cada fila de la tabla la correspondiente fila de la tabla DEPART
FOR i IN 1..num_elem LOOP
   SELECT dept_no,dnombre,loc INTO
            v_tabla_depart(i).dept_no,v_tabla_depart(i).dnombre,
              v_tabla_depart(i).loc  FROM DEPART WHERE  dept_no=i*10;
END LOOP;
--visualizo la tabla
FOR i IN 1..num_elem LOOP
    if v_tabla_depart.exists(i) then
          DBMS_OUTPUT.PUT_LINE(v_tabla_depart(i).dept_no||'.....'||
                v_tabla_depart(i).dnombre||'.....'||v_tabla_depart(i).loc );
       else
          dbms_output.put_line('posicion '||i||' de la tabla => vacia');
     end if;
END LOOP;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS. EJERCICIOS (1/3)

- Diseñar un bloque que cargue en una tabla (anidada o indexada como se desee) el numero de departamento, nombre de departamento y los empleados que tiene dicho departamento. Una vez cargada, visualizar toda la tabla y mostrar el primero y el ultimo elemento de la tabla

  NOTAS:

  - Crear un cursor con la consulta
  - Crear un registro del mismo tipo que los campos de la consulta
  - Crear una tabla indexada que contendrá el registro

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS. EJERCICIOS (2/3)

```
DECLARE
    -- defino un cursor con la consulta
    CURSOR cursor_depart IS SELECT depart.dept_no, dnombre, count(emp_no) numemple from
            depart,emple where depart.dept_no=emple.dept_no group by depart.dept_no, dnombre;
    -- defino un registro compatible con el cursor
    -- no incluyo el numero de departamento pues en este caso voy a utilizar una tabla indexada y dicho
    -- campo será utilizado como índice de la tabla
    TYPE tipo_reg IS RECORD
                                (nombre_depart      depart.dnombre%type,
                                 numemple           number);

    --defino una tabla indexada cuyas filas serán del registro definido anteriormente,
    --y donde yo utilizaré el numero de departamento para acceder a los elementos de dicha tabla
    TYPE tipo_tabla_depart IS TABLE of tipo_reg INDEX BY BINARY_INTEGER;

    --defino una variable tabla del tipo tabla definido anteriormente
    var_tabla tipo_tabla_depart;

    pos number:= 0;
```

Tipos de Datos Compuestos: Collections

COLLECTIONS DE REGISTROS. EJERCICIOS (3/3)

```
BEGIN
     --ejecuto el cursor con un bucle FOR ...IN CURSOR y según recorro las filas del cursor voy
    cargando la tabla
     for var_cur IN cursor_depart LOOP
                        var_tabla(var_cur.dept_no).nombre_depart := var_cur.dnombre;
                        var_tabla(var_cur.dept_no).numemple := var_cur.numemple;
     end loop;

     pos := var_tabla.first;
     while var_tabla.exists(pos) loop
                        dbms_output.put_line( ' Departamento Numero : '||pos||
                                              '          Departamanto : '|| var_tabla(pos).nombre_depart||
                                              '          Empleado  :'||var_tabla(pos).numemple);
                        pos:=var_tabla.next(pos);
     end loop;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES

- ¿Qué es?:
  - Una excepción es un identificador PL/SQL que surge durante la ejecución
- ¿Cómo surge?
  - Se produce un error Oracle
  - También puede ser provocado explícitamente por el programador
- ¿Cómo se gestiona?
  - Interrumpiéndolo con un manejador
  - Propagándola al entorno de llamadas

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

GESTIÓN EXCEPCIONES

- Existen dos formas de gestionar una excepción:
  - **Interrumpir la Excepción**: el procesamiento se ramifica al correspondiente manejador de excepciones de la sección de excepciones del propio bloque. El PL/SQL termina satisfactoriamente.
  - **Propagar la excepción**: Si la excepción surge en la sección ejecutable (BEGIN....END) del bloque y no hay ningún manejador de excepciones correspondiente, el bloque PL/SQL propaga la excepción al correspondiente bloque padre de forma sucesiva hasta encontrar un manejador.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

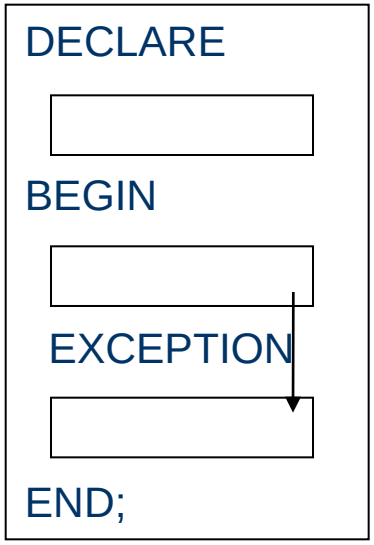
Excepciones

GESTIÓN EXCEPCIONES

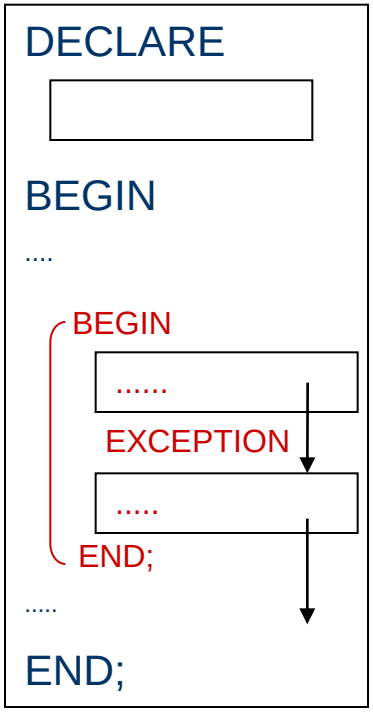
INTERRUMPIR LA EXCEPCION

SURGE LA EXCEPCION

SE INTERRUMPE LA EXCEPCION y se gestiona



PROPAGAR LA EXCEPCION



SURGE LA EXCEPCION

Se ejecuta la excepción y no se interrumpe la excepción, propagándose al entorno de llamada

SE PROPAGA AL ENTORNO DE LLAMADAS

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

TIPOS DE EXCEPCIONES

- Provocadas implícitamente:
  - Predefinidas por el Servidor Oracle
  - No predefinidas por el Servidor Oracle
- Provocadas explícitamente: definidas por el usuario
  (por ejemplo update no produce excepción si no ha podido realizar la actualización, lo que hace es devolver un valor con el número de filas afectadas, con lo cual una posible gestión sería con una excepción forzada por el usuario)

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

TIPOS DE EXCEPCIONES

TIPOS DE EXCEPCIONES		
Provocadas Implícitamente por el servidor Oracle	**Predefinidas por el Servidor Oracle**	Las más comunes. No se declaran y el servidor Oracle las soporta automáticamente
	No predefinidas por el Servidor Oracle	Errores estándar reconocidos por el servidor Oracle. Se declaran en el apartado de declaraciones y el servidor las reporta automáticamente
Provocadas explícitamente por el programador	Forzadas por el programador	Se declara en la sección de declaraciones y se generan de forma explícita, por programa

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES

Sintaxis:

```
BEGIN
.....
.....
EXCEPTION
   WHEN <nom_excepcion1> THEN
                        instrucciones;
   WHEN <nom_excepcion2> THEN
                        instucciones;
   ....
   [WHEN OTHERS THEN
                        instrucciones;]
END <nom_programa>
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES PREDEFINIDAS EN ORACLE

- Se disparan automáticamente al producirse determinados errores de Oracle

Nombre Excepcion	Número Error Oracle	Descripción
NO_DATA_FOUND	ORA-01403	La SELECT no devolvió filas de datos
TOO_MANY_ROWS	ORA-01422	SELECT devolvió más de una fila
INVALID_CURSOR	ORA-01001	Se produjo una operación de cursor ilegal
ZERO_DIVIDE	ORA-01476	Se intentó dividir entre 0
DUP_VAL_ON_INDEX	ORA-00001	Se intentó insertar un valor duplicado
INVALID_NUMBER	ORA-01722	Fallo de conversión de una cadena de caracteres a números
LOGIN_DENIED	ORA-01017	Conexión con un usuario y/o contraseña no válidos

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES PREDEFINIDAS EN ORACLE

Nombre Excepcion	Número Error Oracle	Descripción
ACCESS_INTO_NULL	ORA-09530	Intento de asignar valores a los atributos de un objeto no inicializado
COLLECTION_IS_NULL	ORA-06531	Intento de aplicar a una tabla o array no inicializada cualquier método que no sea EXISTS
CURSOR_ALREADY_OPEN	ORA-06511	Intento de abrir un cursor ya abierto
NOT_LOGGED_ON	ORA-01012	PL/SQL hace una llamada a una BD sin estar conectado
PROGRAM_ERROR	ORA-06501	PL/SQL tiene un problema interno
TIMEOUT_ON_RESOURCE	ORA-00051	Ocurre un Time Out cuando Oracle está esperando un recurso
VALUE_ERROR	ORA-06502	Error aritmético, conversión, truncado o tamaño
SUBSCRIPT_BEYOND_COUNT	ORA-006533	Acceso a un elemento de una tabla array usando un índice mayor que el número de elementos que tiene

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES PREDEFINIDAS EN ORACLE

```
BEGIN
    .....
    .....
EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('La Select no ha devuelto ninguna fila);
    WHEN TOO_MANY_ROWS THEN
      DBMS_OUTPUT.PUT_LINE('La Select ha devuelto mas de 1 fila);
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error no catalogado');
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES PREDEFINIDAS EN ORACLE

- Diseñar un bloque PL/SQL con un *cursor implícito* que devuelva el nombre del empleado cuyo oficio es ANALISTA, utilizar la correspondiente excepción para el caso de que el cursor devuelva más de una fila, de forma que visualice el mensaje **“ERROR, la consulta ha devuelto más de una fila”.** Comprobar utilizando la excepción y sin utilizarla

- Diseñar un bloque PL/SQL con un cursor implícito que permita consultar el nombre y salario de un empleado a partir del numero de empleado que se introducirá por teclado. Utilizar la excepción correspondiente para el caso en que no exista el correspondiente numero de empleado, en cuyo caso deberá de mostrarse el mensaje “ERROR, no existe el numero de empleado .....”

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES PREDEFINIDAS EN ORACLE

- Diseñar un bloque PL/SQL con un *cursor implícito* que devuelva el nombre del empleado cuyo oficio es ANALISTA, utilizar la correspondiente excepción para el caso de que el cursor devuelva más de una fila, de forma que visualice el mensaje **“ERROR, la consulta ha devuelto más de una fila”.**

```
declare
      v_nombre     emple.apellido%type;
begin
      select apellido into v_nombre from emple where oficio=‘ANALISTA’;
      dbms_output.put_line('el nombre es '||v_nombre);
exception
    when TOO_MANY_ROWS then
         dbms_output.put_line('ERROR, la sentencia select ha devuelto mas de una fila para
     este departamento');
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES PREDEFINIDAS EN ORACLE

- Diseñar un bloque PL/SQL con un cursor implícito que permita consultar el nombre y salario de un empleado a partir del numero de empleado que se introducirá por teclado. Utilizar la excepción correspondiente para el caso en que no exista el correspondiente numero de empleado, en cuyo caso deberá de mostrarse el mensaje "**ERROR, no existe el numero de empleado .....**"

```
accept emp prompt 'Número del empleado';
declare
    v_salario        emple.salario%type;
     v_nom           emple.nombre%type;
begin

    dbms_output.put_line('Código del empleado '||&emp);
      select salario,apellido into v_salario,v_nom from emple where emp_no=&emp;
      dbms_output.put_line('Nombre: '||v_nom||' Su Salario : '||v_salario);
exception
      when no_data_found then
          dbms_output.put_line('ERROR, no existen el empleado con código '||&emp);
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES NO PREDEFINIDAS EN ORACLE

Para interrumpir un error no predefinido en Oracle:

- NO PREDEFINIDAS EN EL SERVIDOR ORACLE: Utilizar el manejador ***OTHERS***: Son errores reconocido por el servidor Oracle pero no predefinidos, y permitirá asociar las acciones a un determinado error de Oracle no predefinido
- ERRORES DE FUNCIONAMIENTO DE PROGRAMA: Diseña el propio programador sus ***interrupciones asociadas al propio funcionamiento del programa PL/SQL del usuario***, por ejemplo: importe erróneo, venta errónea,… (*no es capaz el Oracle de detectarlas*)

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

OTRAS EXCEPCIONES. NO PREDEFINIDAS. OTHERS

Son excepciones producidas por el servidor Oracle que no tienen asociadas ningún nombre, pero si un número y un breve mensaje de error, y a las cuales el usuario puede asignar un nombre para procesarlas después como cualquier excepción predefinida. Por defecto estos errores transfieren el control a la sección EXCEPTION, y se tratarán en la cláusula OTHERS

- SQLCODE: devuelve el código del error
- SQLERRM: devuelve el mensaje de error

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

OTRAS EXCEPCIONES. NO PREDEFINIDAS. OTHERS

```
…
EXCEPTION
    ....
    WHEN OTHER THEN
      DMBS_OUTPUT.PUT_LINE( ‘Error: ‘ || SQLCODE || ’ ‘ || SQLERRM);
    ....
END;
```

Una vez conocido el código de error y el mensaje de error el usuario puede asignarle un nombre y utilizarla como cualquier excepción predefinida, para ello será necesario utilizar la sentencia PRAGMA EXCEPTION_INIT

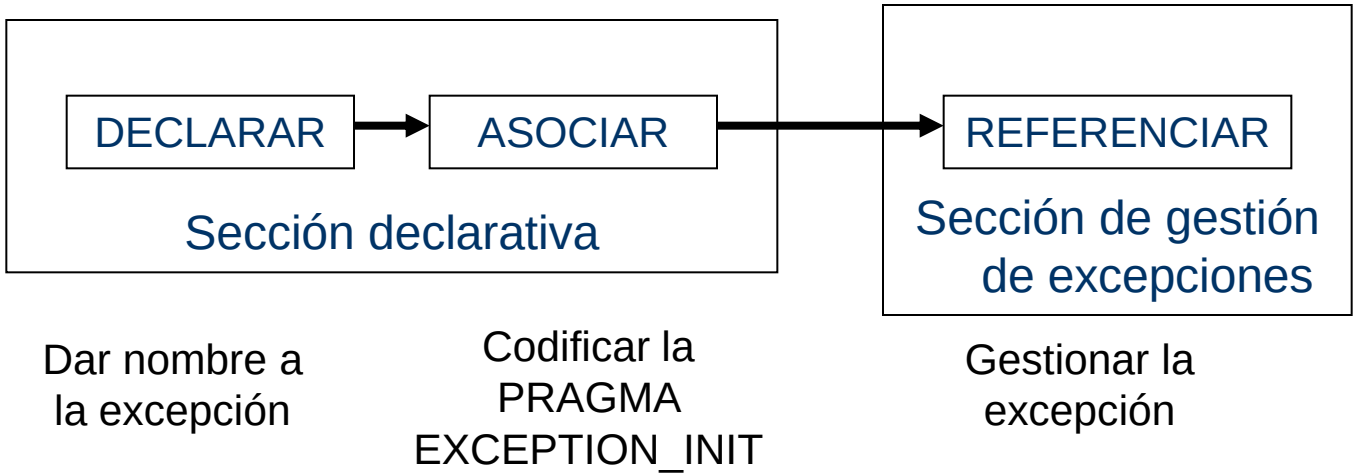
Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

OTRAS EXCEPCIONES. NO PREDEFINIDAS. OTHERS

Conocido el código del error, se puede ASIGNAR UN NOMBRE a esa excepción:

- crear un nombre de error (dar un nombre y asignarle tipo EXCEPTION)
- asociar ese error al código de error de Oracle (PRAGMA)
- Gestionarlo en el apartado EXCEPTION cuando se levante de forma automática al detectarlo el servidor de Oracle



Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

OTRAS EXCEPCIONES. OTHERS

```
DECLARE
    mi_excepcion                              EXCEPTION;      - - declaro una nueva exception
    PRAGMA EXCEPTION_INIT (mi_excepcion, -2292);   -- se asocia a un error de ORACLE
BEGIN
    ....
    .... - - no es necesario levantar la excepción (pues es un error de los detectables por el servidor de
        Oracle), cuando se produzca el error en ORACLE salta a la parte excepción asociada a ese codigo
    ...
EXCEPTION
    .....
    WHEN mi_excepcion THEN    -- automáticamente se ejecuta la exception cuando el
                instrucción 1;     -- servidor de Oracle detecte el error -2292
                instrucción 2;
                .....
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

OTRAS EXCEPCIONES. OTHERS (1/2)

```
DECLARE
    cod_err             number(6);
    vnif                varchar2(10);
    vnom                varchar2(15);
    no_hay_espacio      EXCEPTION; --para error del servidor de oracle no declarados

    PRAGMA EXCEPTION_INIT  (no_hay_espacio, -1547); -- se asocia a un error de ORACLE

BEGIN
  …. (continua en la pagina siguiente)
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

OTRAS EXCEPCIONES. OTHERS (2/2)

```
….. (continuación de la página anterior)
  EXCEPTION
    WHEN no_hay_espacio THEN
                insert into tabla(col1) values (‘ERR TABLESPACE’);
    WHEN NO_DATA_FOUND THEN
                insert into tabla(col1) values (‘ERR no habia datos’);
    WHEN TOO_MANY_ROWS THEN
                insert into tabla(col1) values (‘ERR demasiados datos’);
    WHEN OTHERS THEN
                cod_err :=SQLCODE;
                insert into tabla(col1) values (cod_err);
END;
```

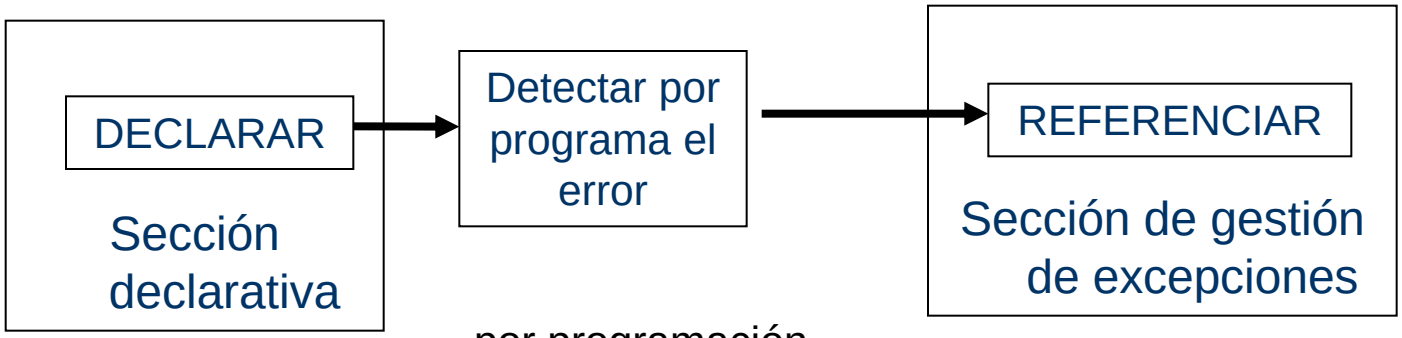
Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES DEFINIDAS POR EL USUARIO

Para interrumpir un error NO PREDEFINIDO EN EL SERVIDOR ORACLE:

- Utilizar el manejador OTHERS (vistas anteriormente)
- Declarar y definir el usuario una actuación para dicho error



Nombrar la excepción

por programación detectar el problema y levantar la excepcion

Gestionar la excepción

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES DEFINIDAS POR EL USUARIO

ERRORES DE FUNCIONAMIENTO DE PROGRAMA: Son errores propios del programa del usuario y que el usuario tratará de una forma concreta. Para su utilización hay que dar 3 pasos:

1. Declarar la excepción en la sección DECLARE
   <nom_excepcion_usr> EXCEPTION;

2. Cuando en la sección ejecutable se detecte el error se levantará la excepción con la orden RAISE
   RAISE nom_excepción_usr;

3. Se tratará en la sección EXCEPTION como una excepción más
   WHEN nom_excepcion_usr THEN
   .....

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES DEFINIDAS POR EL USUARIO

```
DECLARE
        ....
        importe_errorneo      EXCEPTION;
        ...
BEGIN
        ....
        IF precio NOT BETWEEN precio_min AND precio_max THEN
                RAISE importe_erroneo;  - - fuerzo la excepción por programa
        END_IF;
        ....
EXCEPTION
        ....
        WHEN importe_erroneo THEN
                DBMS_OUTPUT.PUT_LINE('Importe erróneo. Venta  cancelada');
        ....
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Ejemplo: EXCEPCIONES NO PREDEFINIDAS (1/3)

```
DECLARE
    cod_err          number(6);
    vnif             varchar2(10);
    vnom             varchar2(15);
    err_blancos      EXCEPTION; -- para error de programa de usuario (se verá despues)
    no_hay_espacio   EXCEPTION;  -- para error del servidor de oracle no declarados

    PRAGMA EXCEPTION_INIT  (no_hay_espacio, -1547); -- se asocia a un error de ORACLE
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES NO PREDEFINIDAS (2/3)

```
BEGIN
    SELECT col1, col2 INTO vnif, vnom FROM TABLA;
    if SUBSTR(vnom,1,1)<= ‘ ‘ then
            RAISE     err_blancos; -- error de prog de usuario, tengo que forzar la excepción
    end if;
    UPDATE clientes SET nombre=vnom where nif=vnif;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

EXCEPCIONES NO PREDEFINIDAS (3/3)

```
EXCEPTION
    WHEN err_blancos THEN
                insert into tabla(col1) values ('ERR blancos');
    WHEN no_hay_espacio THEN
                insert into tabla(col1) values ('ERR TABLESPACE');
    WHEN NO_DATA_FOUND THEN
                insert into tabla(col1) values ('ERR no habia datos');
    WHEN TOO_MANY_ROWS THEN
                insert into tabla(col1) values ('ERR demasiados datos');
    WHEN OTHERS THEN
                cod_err :=SQLCODE;
                insert into tabla(col1) values (cod_err);
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

PROPAGACIÓN DE LAS EXCEPCIONES

- Cuando un SUBbloque gestiona una excepción, éste termina, y el control se reanuda en el bloque padre que lo contiene, en la sentencia siguiente al END del subbloque.

- Aprovechando esta circunstancia, cuando un bloque genera una excepción, si no tiene un manejador para esa excepción, ésta se *propaga* a los bloques padres sucesivos hasta que encuentra el correspondiente manejador o hasta que resulta una excepción no gestionada en el entorno host. En el caso de ser gestionada se devuelve el control bloque padre donde fue gestionada.

- Una ventaja de este comportamiento es que es posible incluir sentencias que necesitan su propia y exclusiva gestión en su propio bloque, y dejar las más generales al bloque padre.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

PROPAGACIÓN DE LAS EXCEPCIONES

```
DECLARE
      ....
      err_no_filas       exception;
      err_integridad     exception;
      PRAGMA EXCEPTION_INIT (err_integridad,-2292);
BEGIN
      FOR v_reg IN cursor_emp LOOP
         BEGIN
            SELECT .....
            UPDATE.....
           IF SQL%NOTFOUND THEN
                        RAISE err_no_filas;
            END IF;
          EXCEPTION
           WHEN err_integridad          THEN .....
           WHEN err_no_filas            THEN....
         END;
      END LOOP;
EXCEPTION
      WHEN NO_DATA_FOUND THEN....
      WHEN TOO_MANY_ROWS THEN...
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

ÁMBITO DE LAS EXCEPCIONES

Reglas a tener en cuenta con las excepciones, en el diseño de aplicaciones:

- Cuando se levanta una excepción se busca la excepción en el bloque actual. Si no está definida en ella, se propaga al bloque que llamó al actual a la sección de EXCEPTION y, 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 de quien lo disparó.

- Si la excepción se levanta en la sección declarativa, automáticamente devuelve el control al bloque que llamó al actual

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

ÁMBITO DE LAS EXCEPCIONES

- Se puede levantar una excepción en la sección EXCEPTION de forma voluntaria o por un error que se produzca al tratar la excepción, pasando el control al bloque que llamó al actual, para ello usar RAISE al final de la correspondiente excepción.

- Las excepciones declaradas en un bloque son locales al bloque y no son conocidas por bloques de nivel superior

- Las variables locales, globales y los atributos de un cursor se pueden utilizar en la EXCEPCION, pero si la excepción se ha disparado con un bucle FOR CURSOR no se podrá acceder a los atributos del cursor, ya que ORACLE cierra el cursor antes de disparar la excepción.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

RAISE_APPLICATION_ERROR

- Permite comunicar de forma interactiva mensajes de error definidos por el propio usuario desde subprogramas almacenados, devolviendo un código de error y un mensaje de error no estándar.

RAISE_APPLICATION_ERROR (err_num, mensaje_error);

Donde:

err_num : es un número especificado entre -20000 y -20999

mensaje_error: mensaje de la excepción definida por el usuario

Ejemplo:

```
raise_application_error(-20001,'Error en dato, salario nulo')
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

RAISE_APPLICATION_ERROR

```
BEGIN
    ....
    select salario INTO v_salario FROM emple WHERE emp_no=num_emp;
    if v_salario IS NULL THEN
                RAISE_APPLICATION_ERROR (-20002,' Error, salario nulo');
        else
                .....
    end if;
    ....
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Ejercicios

- Escribir un bloque PL/SQL que imprima el nombre del empleado que gana 100€ más o menos del valor de un salario introducido por teclado.
  - En caso de encontrar dicho empleado visualizar el nombre.
  - (ERROR) Si no hay ningún empleado dentro de ese rango de sueldo, mostrar un mensaje al usuario indicando cuál es el caso, utilizando una excepción para este caso
  - (ERROR) Si hay uno o más empleados dentro de este rango, el mensaje debería indicar cuántos hay en ese rango de sueldo
  - (ERROR) Gestionar cualquier otra excepción con el manejador de excepciones adecuado, el mensaje debería de indicar que se ha producido otro error.

- Diseñar una función llamada FUNC_NUM_EMP que permita devolver el emp_no de un determinado empleado a partir de su apellido, el departamento y el oficio que tiene. Controlar las posibles excepciones.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Ejercicios

```
accept p_sal prompt 'Introduce el salario';
Declare
        v_sal                   emple.salario%type;
        sal_max                 emple.salario%type;
        sal_min                 emple.salario%type;
        v_nombre                emple.apellido%type;
        v_total                 number;
begin
        sal_max:= &p_sal+100;
        sal_min:= &p_sal-100;
     select apellido,salario into v_nombre,v_sal from emple where salario between sal_min and sal_max;
        dbms_output.put_line('Nombre : '||v_nombre|| '     Salario: '||v_sal);
exception
        when no_data_found then
            dbms_output.put_line('ERROR, No existe ningún empleado con salario en este rango ('||sal_min||','||sal_max||')');
        when too_many_rows then
            select count(*) into v_total from emple where salario between sal_min and sal_max;
            dbms_output.put_line('ERROR, más de un empleado con salario en este rango ('||sal_min||','||sal_max||')');
        when others then
            dbms_output.put_line('se ha producido otro error: '||SQLCODE);
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Ejercicios

```
accept p_sal prompt 'Introduce el salario';
declare
      v_sal                          emple.salario%type;
      sal_max                        emple.salario%type;
      sal_min                        emple.salario%type;
      total_emp                      number(7);
      err_cero_emp                   EXCEPTION;
      err_mas_de_un_emp              EXCEPTION;
begin

      sal_max:= &p_sal+100;
      sal_min:= &p_sal-100;
    select COUNT(apellido)  into total_emp from emple
                                 where salario between sal_min and sal_max;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Ejercicios

```
        if total_emp = 0 then
                        RAISE err_cero_emp;
            elsif  total_emp >1 then
                    RAISE err_mas_de_un_emp;
                else
                    dbms_output.put_line('Existe solo un empleado con salario entre(‘
                                        ||sal_min||','||sal_max||')');
                    select apellido into v_nom fom emple where salario between sal_min and sal_max;
                    dbms_output.put_line(‘El empleado es : || v_nom);
        end if;
exception
  when err_cero_emp then
    dbms_output.put_line('ERROR, No existe ningún empleado con salario entre ('||sal_min||','||sal_max||')');
  when err_mas_De_un_emp then
     dbms_output.put_line('ERROR, más de un empleado con salario entre ('||sal_min||','||sal_max||')  Total
    empelados:‘|| total_emp);
  when others then
        dbms_output.put_line('se ha producido otro error: ');
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Ejercicios

- Crear un procedimiento, NEW_EMP, para insertar un nuevo empleado dentro de la tabla EMPLE. El procedimiento deberá:
  - El procedimiento recibirá como datos el apellido,oficio y el departamento
  - Deberá llamar a una **función VALID_DEPTNO** que devolverá TRUE o FALSE según exista o no el departamento, finalizando el programa en caso de no existir.
  - Como DIR, se asignará el DIRECTOR del correspondiente departamento, y en caso de existir departamento y no tener DIRECTOR se asignará como DIR el emp_no del presidente de la empresa.
  - Para el salario, se utilizará una **función CALCULO_SALARIO**. Se asignará el menor salario de ese oficio en dicho departamento, en caso de no existir este oficio en ese departamento se asignará el menor salario de los empleados de ese oficio.
  - Como comisión será 0 y fecha de alta la actual.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

Funcion VALID_DEPTNO

```
create or replace function valid_deptno(dep IN DEPART.dept_no%type)
    RETURN boolean
IS
  v_dep  depart.DEPT_NO%type:=0;
begin
     select dept_no into v_dep from DEPART where dept_no=dep;
     return (TRUE);
exception
    when no_data_found then
        return (FALSE);
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

FUNCION CALCULO_SALARIO

```
create or replace function calculo_salario(dep IN DEPART.dept_no%type,ofi IN emple.oficio%type)
RETURN emple.salario%type
IS
  v_sal   emple.salario%type;
  err_sal exception;
begin
    select min(salario) into v_sal from emple where dept_no=dep and oficio=ofi;
    if v_sal is null then
                    RAISE err_sal;
      else
                   return v_sal;
    end if;
exception
     when err_sal then
                  dbms_output.put_line('Para ese oficio y departamento no existen empleados');
                  dbms_output.put_line('el salario será el menor de los salarios de todos los empleados
de ese oficio');
                  select min(salario) into v_sal from emple where oficio=ofi;
                  return v_sal;
end calculo  salario;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

PROCEDIMIENTO NEW_EMP

```
create or replace procedure NEW_EMP(dep emple.dept_no%type,ape emple.apellido%type, ofi emple.oficio%type)
IS
  new_emp_no              emple.emp_no%type;
  new_dir                 emple.dir%type;
  new_sal                 emple.salario%type;
  new_com                 emple.comision%type:=0;
  err_sal                 EXCEPTION;
begin
  --comprueba si existe departamento mediante la funcion VALID_DEPTNO
  if valid_deptno(dep) then
        --calcular el siguiente emp_no
        select max(emp_no) into new_emp_no from emple;
        new_emp_no:=new_emp_no+1;

         --calculo del DIR del departamento
           begin
                 select dir into new_dir from emple where dept_no=dep and oficio='DIRECTOR';
            exception
                 when no_data_found then
                    dbms_output.put_line('Para ese oficio y departamento no existen DIRECTOR');
                    dbms_output.put_line('el DIRECTOR será el PRESIDENTE de la empresa');
                   select emp_no into new_dir from emple where oficio='PRESIDENTE';
            end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Excepciones

PROCEDIMIENTO NEW_EMP

```
            --calculo del salario mediante la funcion CALCULO_SALARIO
            new_sal:=calculo_salario(dep,ofi);

            dbms_output.put_line(' ');
            dbms_output.put_line('Insertando nuevo empleado en la tabla EMPLE');
            dbms_output.put_line(new_emp_no||' '|| ape||' '|| ofi||' '||new_dir||' '||new_sal||'
                                    '||new_com||' '||dep);
            insert into emple (emp_no,apellido,oficio,dir,salario,comision,dept_no,fecha_alt)
                    values (new_emp_no, ape, ofi,new_dir, new_sal,new_com,dep,sysdate);
            commit;
      else
            dbms_output.put_line('ERROR, el dep '||dep||' no existe');
  end if;
end;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES

- Agrupan de forma lógica conceptos PL/SQL relacionados: tipos, items, subprogramas,….
- Están formados por:
  - ESPECIFICACIÓN: contiene declaraciones públicas de subprogramas, tipos, constantes, variables, cursores, excepciones,...
  - CUERPO: contiene los detalles de implementación y declaraciones privadas accesible solo desde los objetos del paquete
- No pueden ser llamados, parametrizados o anidados
- Permiten leer múltiples objetos en memoria de una sola vez.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

CREACION DE PAQUETES

- Especificación

```
CREATE OR REPLACE PACKAGE nom_paquete
AS
    <declaraciones de tipos, constantes, variables, cursores, excepciones,...>
    <especificación de subprogramas>
END nom_paquete;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

CREACION DE PAQUETES

- Cuerpo

```
CREATE OR REPLACE PACKAGE BODY nom_paquete
AS
    <declaraciones de tipos, constantes, variables, cursores, excepciones,...>
    <especificación de subprogramas>
BEGIN
    ....
    instrucciones;
    ....
END nom_paquete;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES. ejemplo

```
/* cabecera o especificación del paquete */
CREATE OR REPLACE PACKAGE paq_prueba
AS
    TYPE t_reg_emple IS RECORD
      (num_emp emple.emp_no%type,
       apellido              emple.apellido%type,
       oficio                emple.oficio%type,
       salario               emple.salario%type,
       departamento          emple.dept_no%type);

    PROCEDURE ver_por_num (v_empno emple.emp_no%type);

    PROCEDURE ver_por_ape(v_ape emple.apellido%type);

    FUNCTION DATOS (v_empno emple.emp_no%type) RETURN t_reg_emple;

END paq_prueba ;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES. ejemplo

```
/* cuerpo del paquete */
CREATE OR REPLACE PACKAGE BODY paq_prueba
AS
    vg_emple t_reg_emple;

    PROCEDURE ver_por_num (v_empno emple.emp_no%type)
    is
    begin
       select emp_no, apellido, oficio, salario, dept_no into vg_emple from emple where emp_no=v_empno;
       ver_emple();
    END ver_por_num;

PROCEDURE ver_por_ape(v_ape emple.apellido%type)
    is
    begin
       select emp_no, apellido, oficio, salario, dept_no into vg_emple from emple where apellido=v_ape;
       ver_emple();
    end ver_por_ape;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES. ejemplo

```
FUNCTION DATOS (v_empno emple.emp_no%type) RETURN t_reg_emple
    is
    begin
      select emp_no, apellido, oficio, salario, dept_no into vg_emple
                                  from emple where emp_no=v_empno;
      return vg_emple;
    end datos;

     PROCEDURE ver_emple
    is
      BEGIN
          DBMS_OUTPUT.PUT_LINE (vg_emple.num_emp|| ' '||vg_emple.apellido|| '
      '||vg_emple.oficio||' '  ||vg_emple.salario|| ' '||vg_emple.departamento);
      END ver_emple;
END paq_prueba ;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES: UTILIZACIÓN DE LOS PAQUETES

- Todos los objetos declarados en el paquete pueden ser utilizados por los demás objetos del mismo

- Para llamar a un subprograma u objeto de un paquete

```
EXECUTE nom_paquete.nom_subprograma(parametros);
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES: Ejercicios

- Crear un paquete llamado MIPACK que tenga los siguientes procedimientos y funciones:
  1. Procedimiento para que a partir de un emp_no, busque y visualice todos los datos del empleado (BUSQ_EMP)
  2. Una función que a partir del dept_no devuelva el nombre del departamento (BUSQ_NOM_DEP)
  3. Una función que a partir de un emp_no calcule su sueldo (CALC_SUELDO)
  4. Un procedimiento que visualice todos los empleados de un departamento (BUSQ_EMP_DEP)

  Recordar que primero hay que definir la cabecera y luego el cuerpo del paquete.

NOTA: Realizar el proceso en un principio solo para el punto 1 y cuando funcione ir ampliándolo sucesivamente con el punto 2, punto 3,...

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES: Ejercicios

1. Procedimiento para que a partir de un emp_no, busque y visualice todos los datos del empleado (BUSQ_EMP)

```
CREATE OR REPLACE PACKAGE mipack AS
    /*Procedimiento para que a partir de un emp_no, busque y visualice todos los datos del
    empleado (BUSQ_EMP)*/
    procedure busq_emp(numemp emple.emp_no%type);
END mipack;
==========================================
CREATE OR REPLACE PACKAGE BODY mipack AS

    /*Procedimiento para que a partir de un emp_no, busque y visualice todos los datos del
    empleado (BUSQ_EMP)*/
    procedure busq_emp(numemp emple.emp_no%type)
    IS
     v_reg emple%rowtype;
    BEGIN
            select * into v_reg from emple where emp_no=numemp;
            dbms_output.put_line(v_reg.emp_no||' '||v_reg.apellido||' '||v_reg.oficio||' '||v_reg.dir||'
    '||v_reg.salario||' '||v_reg.comision||' '||v_reg.dept_no||' ');
    END busq_emp;
END mipack;
```

PAQUETES

PAQUETES: Ejercicios

2. Una función que a partir del dept_no devuelva el nombre del departamento (BUSQ_NOM_DEP)

```
CREATE OR REPLACE PACKAGE mipack AS

    /*Procedimiento para que a partir de un emp_no, busque y visualice todos los
    datos del empleado (BUSQ_EMP)*/
    procedure busq_emp(numemp emple.emp_no%type);


    /*Una función que a partir del dept_no devuelva el nombre del departamento
    (BUSQ_NOM_DEP)*/
    procedure busq_nom_dep(numdept depart.dept_no%type);

End mipack;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

PAQUETES: Ejercicios

3. Una función que a partir del dept_no devuelva el nombre del departamento (BUSQ_NOM_DEP)

```
CREATE OR REPLACE PACKAGE BODY mipack AS

    /*Procedimiento para que a partir de un emp_no, busque y visualice todos los
    datos del empleado (BUSQ_EMP)*/
    .....

    /*Una función que a partir del dept_no devuelva el nombre del departamento
    (BUSQ_NOM_DEP)*/
    procedure busq_nom_dep(numdept depart.dept_no%type)
    IS
     nom_dept depart.dnombre%type;
    BEGIN
            select dnombre into nom_dept from depart where dept_no=numdept;
            dbms_output.put_line('El departamento '||numdept||' es '||nom_dept);
    END busq_nom_dep;

End mipack;
```


Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

EJERCICIOS

- Añadir MIPACK los siguientes procedimiento:
  - Función llamada ULT_EMPNO que permita calcular el último emp_no que existe en la tabla EMPLE
  - Función que permita comprobar si existe un nombre de departamento en DEPART
  - Procedimiento para insertar un departamento. Se pasarán como parámetros el número de departamento a insertar, el nombre del departamento y la localidad. Habrá que validar que no exista dicho departamento (ni el número ni el nombre) utilizando para ello las funciones existentes en MIPACK
  - Procedimiento para borrar un departamento. Se pasarán los mismos parámetros que en el apartado anterior. Deberá de verificarse que dicho departamento existe en DEPART

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

PAQUETES

- Diseñar un paquete llamado **GEST_DEPART** para gestionar los departamentos y deberá de incluir, al menos, los siguientes subprogramas:
  - **Insertar_nuevo_depart**: permite insertar un departamento nuevo. El procedimiento recibe el nombre y la localidad del nuevo departamento. Creará el nuevo departamento comprobando que el nombre no se duplica y le asignará como número de departamento la decena siguiente al último número de departamento utilizado.
  - **Borrar_depart**: permite borrar un departamento. El procedimiento recibirá dos números de departamento, de los cuales el primero corresponde al departamento que queremos borrar y el segundo al departamento al que pasarán los empleados del departamento que se va a eliminar. El procedimiento se encargará de realizar los cambios oportunos en los número de departamento de los empleados correspondientes.
  - **Modif_loc_depart**: modifica la localidad del departamento. El procedimiento recibirá el número del departamento que se modifica y la nueva localidad y realizará el cambio solicitado.
  - **Visualizar_datos_depart**: visualizará los datos de un departamento cuyo número se pasará en la llamada. Deberá de comprobar que exista el departamento utilizando una función específica para ello. En caso de existir además de los datos relativos al departamento, se visualizará el número de empleados que pertenecen actualmente al departamento.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

Ejercicios

EJERCICIOS

- Incorporar al paquete MIPACK los siguientes procedimientos y funciones:
  - Incluir dentro del paquete todos los procedimientos y funciones utilizados en el ejercicio de la página 87 incluidos el procedimiento NEW_EMP que permitía insertar un nuevo empleado, así como el propio procedimiento NEW_EMP
- Crear un bloque PL/SQL que permita dar de alta un nuevo empleado utilizando para ello el procedimiento NEW_EMP del paquete MIPACK
- Diseñar un bloque PL/SQL que permita visualizar todos los empleados que existen en el departamento donde se dio de alta el empleado del ejercicio anterior, utilizando para ello el procedimiento que existe en el paquete MIPACK.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS o DISPARADORES

- Se ejecutan o disparan automáticamente cuando se producen ciertos eventos o sucesos que afectan a la tabla (inserción, borrado o modificación de filas) o al sistema.
- Prevenir transacciones erróneas
- Implementar reglas administrativas complejas
- Generar automáticamente valores derivados
- Auditar actualizaciones e, incluso, enviar alertas
- Gestionar réplicas remotas de la tabla

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS DE LA BD: Creación

Antes de codificar un trigger, es necesario decidir sobre:

Parte	Descripción	Valores Posibles
Momento	Cuando se ejecuta el trigger con relación al evento	BEFORE AFTER INSTEAD OF (ejecuta el cuerpo del trigger en lugar de la sentencia. Para vistas que no pueden ser modificadas de otra forma)
Evento	Qué manipulación de datos sobre la tabla causa la ejecución del trigger	INSERT UPDATE DELETE
TABLA	Tabla sobre la que actúa el trigger	Nombre de la tabla
TIPO TRIGGER	Número de veces que se ejecuta el cuerpo del trigger	Sobre SENTENCIA (solo actúa una vez, por defecto) Sobre FILA (una vez para cada registro afectado por el evento
WHEN condición	Cuando se cumpla la condición indicada	
CUERPO DEL TRIGGER	Bloque PL/SQL	

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

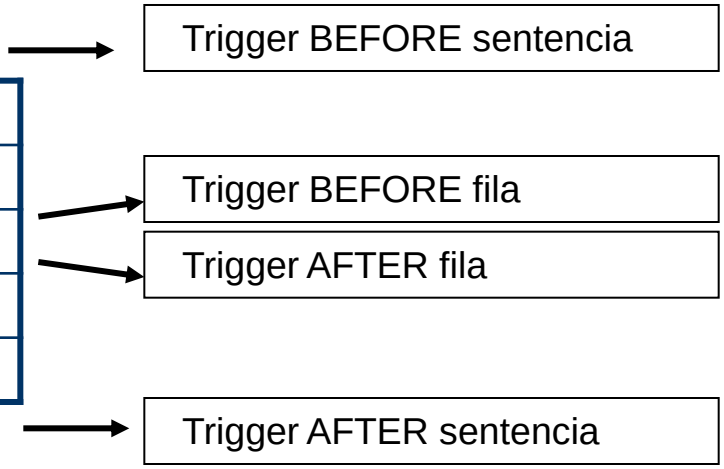
TRIGGERS DE LA BD

Secuencia de disparo de un trigger en una tabla cuando se manipula una sola fila:

INSERT INTO DEPT(deptno,dname,loc) values (50,'EDUCACION','NEW YORK');

TABLA DEPT

DEPTNO	DNAME	LOC
10	CONTABILIDAD	NEW YORK
20	INVESTIGACION	DALLAS
30	VENTAS	CHICAGO
40	OPERACIONES	BOSTON



Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

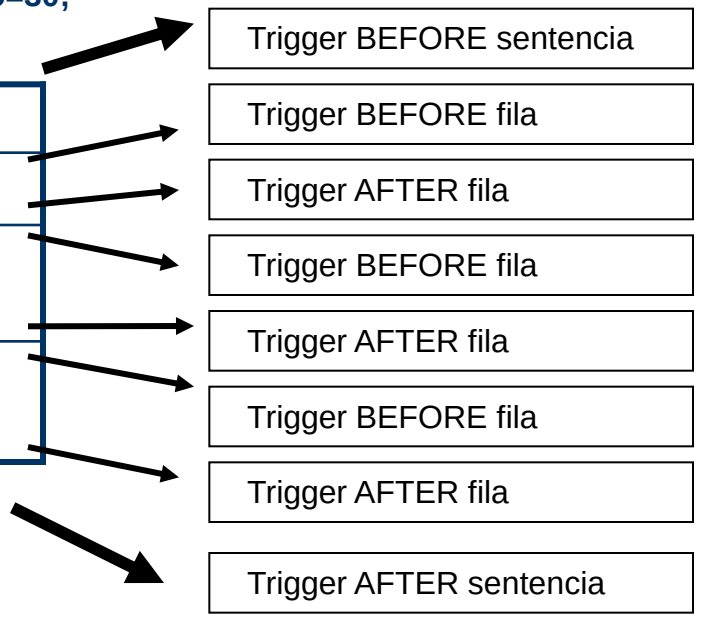
TRIGGERS DE LA BD

Secuencia de disparo de un trigger en una tabla cuando se manipula varias filas:

UPDATE EMP SET SAL=SAL*1.1 WHERE deptno=30;

TABLA EMP

EMPNO	ENAME	DEPTNO
7839	KING	30
7698	BLAKE	30
7788	SMITH	30



Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE SENTENCIA

CREATE [OR REPLACE] TRIGGER *nom_trigger*
{ BEFORE | AFTER} { DELETE | INSERT | UPDATE [OF <*list_columnas*>}
[OR { BEFORE | AFTER} { DELETE | INSERT | UPDATE [OF <*list_columnas*>}]
ON *nom_tabla*
[FOR EACH {STATEMENT [WHEN (*condicion*)] }]
/* inicio bloque PL/SQL*/
[DECLARE
<*declaraciones*>]
BEGIN
<*acciones*>
[EXCEPTION
<*gestión de excepciones*>]
END;

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE SENTENCIA

```
CREATE OR REPLACE TRIGGER secure_emp
    BEFORE INSERT ON emple
BEGIN
   IF (TO_CHAR(sysdate,'DY') in ('SAT','SUN'))
     OR (TO_CHAR(sysdate,'HH24') not between '08' AND '18'
     THEN RAISE_APPLICATION_ERROR (-20500,' Solo se puede insertar en la tabla
                                                    EMP en horario laboral');
   END IF;
END;
```

Si intento hacer un INSERT INTO EMP en horario no laboral se dispara el trigger

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS: Uso de predicados condicionales

- Permite combinar varios eventos de trigger en uno solo aprovechando los predicados condicionales INSERTING, UPDATING y DELETING dentro del cuerpo del trigger.

```
CREATE OR REPLACE TRIGGER nom_trigger
     BEFORE INSERT OR UPDATE OR DELETE ON tabla
BEGIN
    IF INSERTING THEN
                    .....
    ELSIF DELETING THEN
                    .....
    ELSIF UPDATING ('nom_col') || UPDATING THEN
                    .......
    END IF;
    .....
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS: Ejercicios

- Diseñar un disparador que salte, e interrumpa la ejecución, en el caso de intentar modificar un número de empleado (**emp_no**) , asignando como numero de error de Oracle con el valor -20001 y como texto a visualizar 'NO SE PERMITE MODIFICAR EL empno', utilizando RAISE_APPLICATION_ERROR (num_error,' texto')
- Realizar una prueba intentando modificar el emp_no del empleado 7934 y actualizándolo por 7777 y comprobar la tabla
- Realizar una prueba intentando modificar el emp_no de todos los empleados para que su valor sea 9999 y comprobar la tabla
- Realizar una prueba intentando modificar el apellido de todos los empleados y asignarle 'PEREZ' (**NO hacer commit**) y comprobar la tabla

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS: Ejercicios

- Diseñar un disparador que salte en el caso de intentar modificar un número de empleado, asignando como numero de error de Oracle con el valor -20001 y como texto a visualizar 'NO SE PERMITE MODIFICAR EL empno', utilizando RAISE_APPLICATION_ERROR (num_error,' texto')

```
CREATE OR REPLACE TRIGGER modif_empno
BEFORE UPDATE OF emp_no ON emple
BEGIN
    RAISE_APPLICATION_ERROR (-20001,' ERROR, no es posible modificar el emp_no en
    EMPLE');
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE REGISTRO o FILA

Se dispara cada vez que se efectúa un evento (insert, delete or update) sobre una fila. Por lo tanto se ejecutará tantas veces como filas se ven afectadas por la sentencia

```
CREATE [OR REPLACE] TRIGGER nom_trigger
    { BEFORE | AFTER} { DELETE | INSERT | UPDATE [OF <list_columnas>}
       [OR { BEFORE | AFTER} { DELETE | INSERT | UPDATE [OF <list_columnas>}]
ON nom_tabla
[REFERENCING OLD AS old | NEW AS new]
[ FOR EACH {ROW [WHEN (condicion) ] } ]
/* inicio bloque PL/SQL*/
[ DECLARE
    <declaraciones>]
BEGIN
    <acciones>
[ EXCEPTION
    <gestión de excepciones> ]
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE REGISTRO O FILA

```
CREATE OR REPLACE TRIGGER audit_subida_salario
    AFTER UPDATE OF salario  ON emple
 FOR EACH ROW

BEGIN                          /* inicio bloque PL/SQL*/
   insert INTO auditaremple
        values ('SUBIDA DE SALARIO DEL EMPLEADO ' ||:old.emp_no
                ||  ' DE: ' ||:old.salario || ' A UN SALARIO DE ' || :new.salario);
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE REGISTRO O FILA

```
CREATE OR REPLACE TRIGGER restringir_salario
     BEFORE INSERT OR  UPDATE OF salario ON emple
 FOR EACH ROW

BEGIN   /* inicio bloque PL/SQL*/
    IF NOT (:new.oficio IN ('MANAGER','PRESIDENT'))
                    AND :new.salario >5000
      THEN
                    RAISE_APPLICATION_ERROR (-20202,'El empleado no puede
                                                ganar esa cantidad por el oficio que
    desempeña');
    END IF;
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE REGISTRO O FILA: OLD y NEW

- Un disparador a nivel de fila se ejecuta por cada fila en la que se produce el suceso
- Los triggers a nivel de fila tienen asociados unos *registros especiales*: el registro **:old** y el registro **:new**
- :old y :new son registros que nos permiten acceder a los datos de la fila actual de un disparador de registro o fila, y solo se pueden utilizar si el trigger es a nivel de fila, es decir si incluye **FOR EACH ROW**
- Si se va hacer referencia a los valores new y old, en la restricción del trigger (WHEN), lo haremos sin poner los dos puntos.

Formato:

:new.campo_regtabla ➔hace referencia al nuevo valor que tiene el campo de la fila que se está tratando

:old.campo_regtabla ➔ hace referencia al viejo valor que tenía el campo de la fila que se está tratando

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE REGISTRO O FILA: OLD y NEW

```
CREATE OR REPLACE TRIGGER audit_emp
                    AFTER DELETE OR INSERT OR UPDATE ON emple
FOR EACH ROW
BEGIN
    INSERT INTO audit_emp_tabla (emp_no, fecha, nom, sal_viejo, sal_nuevo)
            VALUES (emp_no, sysdate, :old.apellido, :old.salario, :new.salario);
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS A NIVEL DE REGISTRO O FILA: OLD / NEW

Valores de los registros :old y :new : solo existen si el trigger es a nivel de fila (**FOR EACH ROW**)

SUCESO	:old	:new
INSERT	NULL	Nuevos valores
UPDATE	Valores actuales	Nuevos valores
DELETE	Valores actuales	NULL

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS INSTEAD OF o DE SUSTITUCIÓN

- Se utilizan para actualizar datos sobre los que se realiza una sentencia DML asociada a una **vista NO modificable**.
- Oracle Server ejecuta el trigger en lugar de ejecutar la sentencia, actuando directamente sobre las tablas implicadas, debiendo por lo tanto realizar la correspondientes actualizaciones en el cuerpo del trigger.
- El trigger se disparará cuando se intente realizar una actualización sobre la vista.
- Ejemplo: Una actualización sobre una vista puede implicar INSERT en una tabla y UPDATE sobre otra. Deberemos diseñar un trigger INSTEAD OF que contenga este INSERT y el UPDATE, de forma que cuando ejecutemos un INSERT sobre la vista se disparará automáticamente el trigger que contiene estas dos operaciones.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS INSTED OF: Actúan sobre vistas

```
CREATE OR REPLACE TRIGGER nom_trigger
     INSTEAD OF
     {  DELETE | INSERT | UPDATE [OF <list_columnas>}
ON nom_vista
[REFERENCING OLD AS old | NEW AS new]
[ FOR EACH {ROW [WHEN (condicion) ] } ]
/* inicio bloque PL/SQL*/
[ DECLARE
     <declaraciones>]
BEGIN
     <acciones>
[ EXCEPTION
     <gestión de excepciones> ]
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

- Construir un disparador que permita auditar las operaciones de inserción o borrados de datos que se realicen en la tabla EMPLE, según las siguientes especificaciones:
  - Crear una tabla AUDITAREMPLE, con los campos abajo indicados
  - Cuando se produzca cualquier manipulación se insertará una fila en la tabla anterior con la siguiente información:
    - Fecha actual
    - Numero de empleado sobre el que se actúa
    - Apellido del empleado sobre el que se actúa
    - La operación de actualización realizada: INSERCION o BORRADO

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

```
CREATE TABLE auditaremple (
fecha date; emp_no number; apellido varchar2(25), accion varchar2(50));

CREATE OR REPLACE TRIGGER auditar_act_emp
    BEFORE INSERT OR DELETE  ON EMPLE
    FOR EACH ROW
  BEGIN
    IF DELETING THEN
      INSERT INTO AUDITAREMPLE
                    VALUES( sysdate,|:OLD.EMP_NO, :OLD.APELLIDO ,'BORRAR ');
    ELSIF INSERTING THEN
       INSERT INTO AUDITAREMPLE
                    VALUES( sysdate,| :NEW.EMP_NO, :NEW.APELLIDO ,‘INSERTAR ');
    END IF;
  END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

- Escribir un trigger de base de datos que permita auditar las modificaciones en la tabla empleados insertado en la tabla *AUDITAREMPLE* los siguientes datos:
  - Fecha
  - Número de empleado
  - Apellido
  - La operación de actualización: *MODIFICACIÓN en campo XXXXXX*

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

```
CREATE OR REPLACE TRIGGER audit_modif
      BEFORE UPDATE  ON EMPLE
      FOR EACH ROW
DECLARE
      CAD    VARCHAR2(50) :=‘ ‘;
BEGIN
      IF UPDATING ('EMP_NO') THEN
            CAD:=CAD || ‘EMPNO ’;
      ENDIF;
      IF UPDATING ('APELLIDO') THEN
            CAD:= CAD || ‘APELLIDO’;
      END IF;
      IF UPDATING ('OFICIO') THEN
            CAD:= CAD || ‘OFICIO’
      END IF;
      IF UPDATING ('DIR') THEN
            CAD:= CAD || ‘DIRECTOR’;
      END IF;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

```
    IF UPDATING ('FECHA_ALT') THEN
            CAD:= CAD || 'FECHA_ALTA';
    END IF;
    IF UPDATING ('SALARIO') THEN
            CAD:= CAD || 'SALARIO';
    END IF;
    IF UPDATING ('COMISION') THEN
              CAD:= CAD || 'COMISION';
    END IF;
    IF UPDATING ('DEPT_NO') THEN
             CAD:= CAD || 'DEPARTAMENTO';
    END IF;

     --creo la cadena MODIFICACION en campo + campo_modificado
    CAD:='MODIFICIACION en campo  '||CAD;
    INSERT INTO AUDITAREMPLE
                        VALUES( sysdate, :OLD.EMP_NO, :OLD.APELLIDO ,CAD);
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

- Crear un disparador que haga fallar cualquier operación de modificación de apellido o del número de un empleado o que suponga una subida de sueldo superior al 10 por 100, utilizar el RAISE_APPLICATION_ERROR

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

- Crear un disparador que haga fallar cualquier operación de modificación de apellido o del número de un empleado o que suponga una subida de sueldo superior al 10 por 100, utilizar el RAISE_APPLICATION_ERROR

```
CREATE OR REPLACE TRIGGER fallo_modif
    BEFORE UPDATE OF apellido, emp_no, salario
    ON emple
    FOR EACH ROW
BEGIN
    IF UPDATING('emp_no') OR UPDATING('apellido')
OR (UPDATING ('salario')  AND
 :new.salario>:old.salario*1.1)
THEN
       RAISE_APPLICATION_ERROR
(-20002,'Err. Modificacion no permitida, intento de actualizar emp_no, apellido o
    incremento de salario mayor al 10%');
    END IF;
END;
```

EJERCICIOS

- Diseñar los siguientes triggers

  - Diseñar una tabla llamada AUDITAR_DEPART con los siguientes campos:
    - FECHA, CAMPO, OPERACIÓN, VALOR_VIEJO, VALOR_NUEVO
  - Diseñar un trigger que permita auditar cualquier operación de inserción en la tabla DEPART, siempre y cuando esta se lleve a buen fin. En esta caso se deberá de insertar en la tabla AUDITAR_DEPART la fecha, el nombre de campo DEPT_NO, el texto 'INSERTADO', en valor nuevo deberá de aparecer el valor del DEPT_NO que se ha insertado.
  - Crear un nuevo trigger que permita auditar cualquier operación de borrado en la tabla DEPART, siempre y cuando este se lleve a buen fin. En este caso se deberá insertar en la tabla la fecha, el nombre de campo DEPT_NO, el texto 'BORRADO', en valor viejo deberá de aparecer el valor del DEPT_NO que se ha borrado.
  - Crear un nuevo trigger que permita auditar cualquier operación de modificación en la localidad. En este caso se deberá de deberá insertar en la tabla la fecha, el nombre de campo LOCALIDAD, el texto 'MODIFICADA', en valor viejo deberá de aparecer el valor antiguo de la localidad y en valor nuevo el valor nuevo.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

EJERCICIOS

- Diseñar un trigger que permita auditar todas las operaciones de insercion y borrado en la tabla DEPART, en una tabla llamada AUDITARDEPART. En esta tabla se deberá de almacenar el número de departamento, nombre, operación realizada (ALTA o BAJA) y fecha en la que se realizó la operación

- Diseñar un trigger que no permita modificar el número de departamento de la tabla DEPART, ni su nombre

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS DE SISTEMA

- Se disparan cuando ocurre un evento de sistema (arranque o parada de la BD, conexión o desconexión de un usuario, creación modificación de un objeto,.....)

```
CREATE OR REPLACE TRIGGER nom_trigger
    {  BEFORE | AFTER }   { lista eventos definición}  | { lista eventos del sistema }
ON {DATABASE | SCHEMA }  [ WHEN (condicion)  ]

/* inicio bloque PL/SQL*/
[ DECLARE
    <declaraciones>]
BEGIN
    <acciones>
[ EXCEPTION
    <gestión de excepciones> ]
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS DE SISTEMA

- < lista eventos definición > puede incluir uno o más eventos DDL separados por OR
- < lista eventos sistema > puede incluir uno o más eventos del sistema separados por OR
- ON SCHEMA se disparará siempre que ocurra el evento en el esquema indicado por el trigger
- ON DATABASE se disparará siempre el evento.

Al asociar un trigger a un evento de sistema es necesario indicar el momento del disparo, aunque algunos eventos solo se pueden disparar ANTES de producirse (ej. Cerrar conexión,…) y otros solo DESPUES (ej. arrancar BD,…)

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS DE SISTEMA

STARTUP	AFTER	Se dispara después de arrancar la instancia
SHUTDOWN	BEFORE	Antes de apagar la instancia
LOGON	AFTER	Después de que el usr se conecte a la BD
LOGOFF	BEFORE	Antes de que el usr se desconecte de la BD
SERVERERROR	AFTER	Cuando ocurre un error de servidor
CREATE	AFTER | BEFORE	Antes o después de crear un objeto en el esquema
DROP	AFTER | BEFORE	Antes o después de borrar un objeto en el esquema
ALTER	AFTER | BEFORE	Antes o después de modificar un objeto en el esquema
TRUNCATE	AFTER | BEFORE	Antes o después de ejecutar un TRUNCATE
GRANT	AFTER | BEFORE	Antes o después de ejecutar un GRANT
REVOKE	AFTER | BEFORE	Antes o después de ejecutar un REVOKE
Otros comandos	AFTER | BEFORE	RENAME, AUDIT, ANALYZE,…

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

TRIGGERS DE SISTEMA

EJEMPLO

```
CREATE OR REPLACE TRIGGER ctrl_conexiones
    AFTER LOGON ON DATABASE
BEGIN
    insert into control_conexiones (usuario, momento, evento)
      values (ORA_LOGIN_USER, sysdate, ORA_SYSEVENT);
END;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

GESTIÓN DE TRIGGERS

- Cuando se crea un trigger por primera vez, éste se activa automáticamente
- Oracle Server comprueba las restricciones de integridad de los triggers activados.
- ACTIVAR o DESACTIVAR UN TRIGGER DE LA BD

ALTER TRIGGER nom_trigger DISABED | ENABLE

- ACTIVAR o DESACTIVAR TODOS LOS TRIGGERS DE UNA TABLA

ALTER TABLE nom_table DISABED | ENABLE ALL TRIGGERS

- RECOMPILAR UN TRIGGER

ALTER TRIGGER nom_trigger COMPILE

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio

TRIGGERS

ELIMINACIÓN DE TRIGGERS

- Para borrar un trigger de la B.D.

```
DROP TRIGGER nom_trigger
```

VISTAS CON INFORMACIÓN SOBRE TRIGGERS

- Para mostrar los triggers existentes y su estado:

```
SELECT * FROM USERS_TRIGGERS
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Sotrondio