UD6: PROGRAMACIÓN DE BASE DE DATOS: PL/SQL (Oracle)

CONCEPTOS INICIALES DE PL/SQL

Bloque

```
[DECLARE
        /* Sección declarativa */
]
BEGIN
        /* Sección ejecutable  */
[EXCEPTION
        */ Sección de excepciones  */
]
END;
```

Identificadores

- Comienzan con una letra + (letras , números, $, _,#)
- Máximo: 30 caracteres.

Comentarios.

--	Comentarios en una sola línea
/* */	Comentarios en varias líneas

Declaración de variables.

<Nombre_Var> <tipo_dato> [CONSTANT] [NOT NULL] [:= / DEFAULT expr];

TIPOS PL/SQL

- **Tipos Derivados:**

  Nombre_Variable NombreTabla.NombreColumna%TYPE;

  - Atributo %TYPE: Permite crear una variable del mismo tipo que una columna de una tabla

  NombreVariable NombreTabla%ROWTYPE;

  - Atributo %ROWTYPE: crea variables que tengan el mismo tipo que una tabla.

- **Entrada por teclado:** &
  V_EMPNO EMPLE.EMP_NO%TYPE:=&EMPLEADO;
  V_SALARIO NUMBER:=&SALARIO;
- Clasificación:
  - **Escalar**: contienen un valor simple
  - **Compuesto**: permiten que se definan y manipulen grupos de valores (colecciones: registros, arrays...)
  - **Referenciado**: llamados *punteros*, designan otros artículos de programa.
  - **LOB**: (CLOB, BLOB, BFILE, NCLOB) contienen valores, llamados *localizadores* que indican la localización de lobs (imágenes,...) que pueden estar almacenados externamente.

Tipos escalares

Numéricos	
NUMBER(P,S)	Numèrico entero o punto flotante.P:precisión(38), S:escala(-84,127)
BINARY_INTEGER	Valores enteros. No almacenar en BD
PLS_INTEGER	Igual que birary, pero genera error en desbordamiento
Carácter	
VARCHAR2(N)	Cadenas de longitud variable (PL:32767, BD:2000,4000(v8))
CHAR(N)	Cadenas de longitud fija (PL:32767,DB:255)
LONG(N)	Cadena de longitud variable(PL:2G,DB:32767)
RAW	
RAW(N)	Almacena datos binarios(PL:32767,BD:255)
LONG RAW(N)	Igual RAW(PL:32767,BD:2G)
DATE	Almacena datos tipo fecha (7bytes)
ROWID	Clave que identifica unívocamente a cada fila, se almacena en hexadecimal: BBBBBBBB.RRRR.FFFF
BOLEAN	TRUE, FALSE, NULL

Tipos compuestos:

- Registros
- Colecciones:
  - VARRAY
  - Tablas Anidadas
  - Tablas Indexadas

OPERADORES

TIPO OPERADOR	
Asignación	**:=**
Lógicos	**AND, OR, NOT**
Concatenación	**||**
Comparación	Igual = distinto != menor que < mayor que >menor o igual <= mayor o igual >= IS NULL BETWEEN LIKE IN
Aritméticos	+, -, *, /, **(potencia)

ESTRUCTURAS DE CONTROL.

IF-THEN-ELSE

```
IF expresión_boleana1 THEN
        secuencia_de_órdenes1;
[ELSIF expresión_boleana2 THEN
        secuencia_de_órdenes2;]
[ELSE
        secuencia_de_órdenes3;]
END IF;
```

SENTENCIA CASE:

```
CASE <expresion>
WHEN <valor_comprobacion1> THEN
   instrucciones1;
WHEN <valor_comprobacion2> THEN
   instrucciones2;
.....
[ELSE
   instrucciones;]
END CASE;
```

```
CASE
WHEN <condición1> THEN
   instrucciones1;
WHEN <condición2> THEN
   instrucciones2;
.....
[ELSE
   instrucciones;]
END CASE;
```

BUCLES SIMPLES

```
LOOP
   instrucciones;
   ...;
   EXIT WHEN condición;
   ...;
   instrucciones;
   ...;
END LOOP;
```

```
LOOP
   instrucciones;
   ...;
   IF condición THEN
         EXIT;
   END IF;
   instrucciones;
   ...;
END LOOP;
```

BUCLES WHILE

```
WHILE condición LOOP
   secuencia_de_órdenes
END LOOP;
```

BUCLES FOR

```
FOR contador_bucle IN [REVERSE]
  límite_inferior..límite_superior LOOP
   secuencia_de_órdenes;
END LOOP;
```

SENTENCIA : SELECT, INSERT, UPDATE,DELETE

SELECT:

```
SELECT <Columna/s> INTO <variable/s>
FROM <Tabla/s>
WHERE  <condición>
....
```

INSERT, UPDATE Y DELETE: Tienen la misma sintaxis que en SQL:

```
DELETE [FROM] nombretabla
        WHERE condición;

INSERT INTO nombretabla [(columna1,columna2,....)]  VALUES (valor [,valor]...);

UPDATE nombretabla
        SET columna1=valor1,..., columnaN=valorN
        WHERE condición;
```

PROCEDIMIENTOS

```
CREATE OR REPLACE PROCEDURE nombre_procedimiento
    [(parametro1 [IN|OUT|IN OUT] <tipo> [{:=|DEFAULT} <valor>,...)]
IS|AS
        declaraciones;
BEGIN
        instrucciones;
[EXCEPTION
        gestión_de_excepciones;]
END [nombre_procedimiento];
```

FUNCIONES

```
CREATE OR REPLACE FUNCTION nombre_función
        [(parametro1 [IN|OUT|IN OUT] <tipo>
                [{:=|DEFAULT} <valor>,...)]
RETURN <tipo_del_valor_devuelto>
IS|AS
        declaraciones;
BEGIN
        órdenes ejecutables;
        RETURN <expr>;
[EXCEPTION
        gestión_de_excepciones;]
END <n_función>;
```


CURSORES

Declarar el cursor

```
CURSOR  nombre_cursor  IS sentencia_SELECT;
```

Apertura del cursor

```
OPEN nombre_cursor;
```

Extracción de los datos del cursor.

```
FETCH  nombre_cursor  INTO  lista_variables;
FETCH  nombre_cursor  INTO  registro_pl/sql;
```

Cierre del cursor.

```
CLOSE  nombre_cursor;
```

Atributos del cursor.

%FOUND	Devuelve TRUE si la última sentencia FETCH devuelve una fila
%NOTFOUND	Devuelve TRUE si la última sentencia FETCH no devuelve ninguna fila
%ISOPEN	Devuelve TRUE si el cursor está abierto
%ROWCOUNT	Devuelve el número de filas devueltas por el cursor hasta el momento

Bucle FOR en un cursor

```
FOR vble_registro IN n_cursor LOOP
        sentencias;
END LOOP;
```

Cursores parametrizados.

```
DECLARE
        CURSOR c_empleados(p_dep  VARCHAR2(7)) IS
                SELECT * FROM empleados
                        WHERE codemp=p_dep;
BEGIN
        OPEN c_empleados('D123');
```

Cursores con SELECT FOR UPDATE.

- Se utiliza para extraer los datos y después modificarlos o borrarlos.
- Pasos:
  1. Declarar el cursor con SELECT ..FOR UPDATE
  2. Órdenes DELETE o UPDATE con la opción WHERE CURRENT OF.

FOR UPDATE.

- Es la última cláusula de la orden SELECT. Sintaxis:

  SELECT ... FROM .... FOR UPDATE [OF referencia_columna][NOWAIT]

- Referencia_columna: una o varias columnas de la tabla para actualizar, si no se especifica ninguna se pueden modificar todas.
- NOWAIT: si los registros a los que se quieren acceder están bloqueados por otro usuario la sentencia SELECT espera hasta que se desbloquean, si no queremos que espere se pondrá la cláusula NOWAIT.

WHERE CURRENT OF.

- Esta cláusula se emplea en las órdenes DELETE o UPDATE cuando el cursor se haya declarado con la opción SELECT .. FOR UPDATE. Se actualizará o borrará la fila que actualmente se haya extraído.
- Sintaxis:

  WHEN CURRENT OF cursor

CURSORES VARIABLES

- Definición :

  TYPE T_CURSOR IS REF CURSOR
  RETURN EMPLE%ROWTYPE;

- Declaración de la variable

  CEMPLE T_CURSOR;

- Asociarlo a una consulta:

  OPEN nombre_variable_cursor **FOR** sentencia_select;
  OPEN CEMPLE FOR SELECT * FROM EMPLE WHERE DEPT_NO=10;

EXCEPCIONES

```
EXCEPTION
WHEN nombre_excepción THEN
        <tratamiento>
WHEN nombre_excepción THEN
        <tratamiento>
....
WHEN OTHERS THEN
        <tratamiento>
        */ Sección de excepciones  */
END;
```

Excepciones internas definidas por ORACLE:

EXCEPCIÓN	DESCRIPCIÓN	SQLCODE
ACCESS_INTO_NULL	El programa intentó asignar valores a los atributos de un objeto no inicializado	-6530
CASE_NOT_FOUND	Ninguna de las opciones existentes en las cláusulas WHEN de un CASE se selecciona, y no hay ninguna cláusula ELSE.	
COLLECTION_IS_NULL	El programa intentó asignar valores a una tabla anidada aún no inicializada	-6531
CURSOR_ALREADY_OPEN	El programa intentó abrir un cursor que ya se encontraba abierto. Recuerde que un cursor de ciclo FOR automáticamente lo abre y ello no se debe especificar con la sentencia OPEN	-6511
DUP_VAL_ON_INDEX	El programa intentó almacenar valores duplicados en una columna que se mantiene con restricción de integridad de un índice único (unique index)	-1
INVALID_CURSOR	El programa intentó efectuar una operación no válida sobre un cursor	-1001
INVALID_NUMBER	En una sentencia SQL, la conversión de una cadena de caracteres hacia un número falla cuando esa cadena no representa un número válido	-1722
LOGIN_DENIED	El programa intentó conectarse a Oracle con un nombre de usuario o password inválido	-1017
NO_DATA_FOUND	Una sentencia SELECT INTO no devolvió valores o el programa referenció un elemento no inicializado en una tabla indexada	+100
NOT_LOGGED_ON	El programa efectuó una llamada a Oracle sin estar conectado	-1012
PROGRAM_ERROR	PL/SQL tiene un problema interno	-6501
ROWTYPE_MISMATCH	Los elementos de una asignación (el valor a asignar y la variable que lo contendrá) tienen tipos incompatibles. También se presenta este error cuando un parámetro pasado a un subprograma no es del tipo esperado	-6504
SELF_IS_NULL	El parámetro SELF (el primero que es pasado a un método MEMBER) es nulo	-30625
STORAGE_ERROR	La memoria se terminó o está corrupta	-6500
SUBSCRIPT_BEYOND_COUNT	El programa está tratando de referenciar un elemento de un array indexado que se encuentra en una posición más grande que el número real de elementos de la colección	-6533
SUBSCRIPT_OUTSIDE_LIMIT	El programa está referenciando un elemento de un array utilizando un número fuera del rango permitido (por ejemplo, el elemento "-1")	-6532
SYS_INVALID_ROWID	La conversión de una cadena de caracteres hacia un tipo *rowid* falló porque la cadena no representa un número	-1410
TIMEOUT_ON_RESOURCE	Se excedió el tiempo máximo de espera por un recurso en Oracle	-51
TOO_MANY_ROWS	Una sentencia SELECT INTO devuelve más de una fila	-1422
VALUE_ERROR	Ocurrió un error aritmético, de conversión o truncamiento. Por ejemplo, sucede cuando se intenta calzar un valor muy grande dentro de una variable más pequeña	-6502
ZERO_DIVIDE	El programa intentó efectuar una división por cero	-1476

- Excepción definidas por el usuario

  DECLARE
  Nombre_excepción **EXCEPTION;**
- Levantar excepción:
  RAISE nombre_excepción;

- Para acceder a los errores de Oracle que no tienen asignada una excepción:
  SQLCODE: código del error
  SQLERRM: mensaje asociado al error
- Errores sin excepción asignada: Asociamos a esa excepción el código de error de ORACLE.
  PRAGMA EXCEPTION_INIT(nombre_excepcion, codigo**);**
- Errores generados por el usuario:
  RAISE_APPLICATION_ERROR (numero_error, mensaje_error**)**
  Rango del nº de error :-20000, -20999

DISPARADORES: TRIGGERS

Formato para la creación de un triggers:

```
CREATE [OR REPLACE] TRIGGER Nombre_del_trigger
        {BEFORE | AFTER |INSTEAD OF}
        {DELETE | INSERT | UPDATE [OF columna1 [,columna2...]]} ...
        ON {tabla |vista}
        [FOR EACH  {ROW |[WHEN (condición)] | STATEMENT}]
/* Comienza el cuerpo del trigger */
[DECLARE
        <declaraciones>
]
BEGIN
        <sentencias>
[EXCEPTION
        <gestión de excepciones>
]
END ;
```

Valores para NEW y OLD.:

- Al utilizar los valores :old (valor anterior) y :new(valor nuevo) debemos tener en cuenta el evento del disparo:
  - **DELETE:** debemos hacer referencia siempre a **:old.nombrecolumna**, ya que :new no existe
  - **INSERT**: debemos hacer referencia siempre a **:new.nombrecolumna**, ya que :old no existe
  - **UPDATE:** se pueden usar las dos

Múltiples eventos de disparo:

INSERTING	Devuelve TRUE si el evento que disparó el trigger fue un comando INSERT
DELETING	Devuelve TRUE si el evento que disparó el trigger fue un comando DELETE
UPDATING	Devuelve TRUE si el evento que disparó el trigger fue un comando UPDATE
UPDATING(columna)	Devuelve TRUE si el evento que disparó el trigger fue un comando UPDATE y la columna especificada ha sido actualizada

Trigger con múltiples eventos de disparo:

```
CREATE or REPLACE TRIGGER nombre_trigger
{BEFORE|AFTER} evento1 OR evento2 OR... ON tabla1
BEGIN
        IF INSERTING THEN
                ...
        ELSIF DELETING THEN
                ...
        ELSIF UPDATING('COLUMNA') THEN
                ...
        END IF;
END;
```

Trigger de Sustitución (Vistas):

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
INSTEAD OF
   {DELETE | INSERT | UPDATE [OF columna1 [,columna2...]]}
   {OR {DELETE|INSERT|UPDATE}...}
ON n_vista
[REFERENCING OLD AS .... , NEW AS ...]
FOR EACH ROW
DECLARE
        ...
BEGIN
        ...
EXCEPTION
        ...
END;
```

PAQUETES

- **Cabecera o Especificación**

```
CREATE [OR REPLACE] PACKAGE nombre_paquete AS
        Declaraciones de tipos, cursores, excepciones,... públicos (accesibles desde el paquete y exterior:n_paquete.n_objeto)
        Especificación de subprogramas (cabeceras=nombre, parámetros y tipo de retorno en funciones)
END [n_paquete];
```

- **Cuerpo del paquete:**

```
CREATE [OR REPLACE] PACKAGE BODY nombre_paquete AS
        declaración de tipos, variables, cursores,... privados
        cuerpo de los subprogramas
BEGIN
        instrucciones iniciales;
END [n_paquete];
```