Índice

1. UTILIZACIÓN DE DML DENTRO DE PL/SQL

En PL/SQL podemos utilizar las sentencias **SELECT, INSERT, UPDATE, y DELETE**. Estas sentencias tienen las mismas restricciones y características que cuando se usaban en SQL.

Todas las operaciones del DML se refieren a una tabla. La referencia a la tabla, de forma general, es:

[propietario.] tabla [@ **dblink**].

Donde propietario identifica al dueño de la tabla y dblink indica que la tabla esta en una base de datos remota.

Las referencias a las tablas pueden ser complicadas, sobre todo si se utilizan los apuntes y los links a las bases de datos remotas. Por ello es conveniente la utilización de sinónimos.

CREATE SYNONYM *sinonimo_nombre* **FOR** *referencia*.

Además de la utilización de las sentencias de SQL también se permiten la utilización de las funciones de SQL como UPPER, COUNT.... etc.

2. FUNCIONES

2.1 FUNCIONES DE CARACTERES

•**CHR** *chr(x)*.- Devuelve el carácter equivalente al valor puesto entre paréntesis.

•**CONCAT** *concat(cadena1, cadena2)*.- Concatena la cadena2 a continuación de la cadena1.

•**INICAP** *initcap(cadena*).- Pone todas las primeras letras de cada palabra a mayúsculas.

•**LOWER** *lower(cadena)*.- Pone en minúsculas.

•**LPAD** *lpad(cadena1, x[cadena2])* .- Añade espacios ala izquierda; 'x' es el numero de caracteres totales de la cadena, si se le añade la cadena2 rellena los espacios con esa cadena.

•**LTRIM** *ltrim(cadena1,cadena2)*.- Devuelve la cadena1 con los caracteres de la izquierda que están en la cadena2 borrados.

•**NLS_INITCAP** *nls_initcap(cadena1 [nls, parámetros])*.- Su resultado es igual al de INITCAP pero se le pueden añadir parámetros para la realización de la función.

•**NLS_LOWER** *nls_lower(cadena1,[nls, parametros])*.- Como LOWER pero con parámetros.

•**REPLACE** *replace(cadena1, search_str [,remplace_str]*).- Devuelve una cadena en la que reemplaza cada concurrencia de la cadena SEARCH_STR, en la cadena1 , con el contenido de la cadena REPLACE_STR.

•**RPAD** *rpad(cadena1, x ,[cadena2])* .- Idéntica a RPAD pero a la derecha.

•**RTRIM** *rtrim(cadena1 [,cadena2])* .- Idéntica que la lTRIM pero a la derecha.

•**SOUNDEX** *soundex(cadena)*.- Retorna la representación fonética de la cadena.

•**SUBSTR** *substr (cadena1, a [,b])*.- Retorna una poción de la cadena1, empezando en la posición 'a' y con 'b' caracteres de largo.

•**SUBSTRB** *substrb (cadena1,a [,b])* .- Igual que SUBSTR pero los valores de 'a' y de 'b' son de tipo byte.

•**TRANSLATE** *translate(cadena1, from_str, to_str)*.- Devuelve una cadena en la que por cada concurrencia de caracteres en la Cadena1 con la FROM_STR, se coloca el carácter de TO_STR.

•**UPPER** *upper(cadena)* .- Pone todos los caracteres de la cadena a mayúsculas.

2.2. FUNCIONES DE CARACTERES QUE DEVUELVEN NÚMEROS

•**ASCII (cadena)** .- Devuelve el valor del carácter.

•**INSTR (cadena1, cadena2 [,a][,b])**.- Da la posición de la *cadena2* en la *cadena1*, empezando en 'a' .

•**INSTRB (cadena1, cadena2 [,a][,b])**.- Igual que el anterior pero la posición va expresada en bytes.

•**LENGTH (cadena)** .- Da la longitud de la cadena.

•**LENGTHB (cadena)** .- Igual que el anterior pero el valor en bytes.

2.3. FUNCIONES NUMÉRICAS

•**ABS (X)**.- Retorna el valor absoluto de la 'x'.

•**CEIL (x)** .- Redondea hacia arriba.

•**COS (X)**.- Retorna el coseno de x.

•**EXP (x)** .- Eleva 'x' a 'e'.

•**FLOOR (x)**.- Redondea hacia abajo.

•**LN (X)**.- Devuelve el logaritmo natural de x. (x>0).

•**LOG (X,Y)** .- Logaritmo en base x de y.

•**MOD (X,Y)** .- Retorna el modulo de la división x/y.

•**POWER (X,Y)**.- Eleva x a la potencia y.

•**ROUND (X [,Y])**.- Redondea 'x' a 'y' espacios a la derecha de la coma decimal.

•**SING (X)**.- Si el valor de 'x' es menor que cero retorna -1,si es igual retorna 0, si es mayor 1.

•**SIN (X)** .- Seno de 'x'.

•**SINH (X)** .- Seno hiperbólico de 'x'.

•**SQRT (X)**.- Raíz cuadrada 'X'.

•**TAN (X)**.- Tangente de 'x'.

•**TANH (X)**.- Tangente hiperbólico de 'x'.

•**TRUNC (X[,Y])** .- Trunca 'x'.

2.4 FUNCIONES DE FECHA

• **ADD_MONTHS (d, x)** .- Añade a la fecha 'd' el un numero 'x' de meses.

• **LAST_DAY (d)**.- Devuelve el último día del mes de la fecha 'd'.

• **MONTHS_BETWEEN (fecha1, fecha2)**.- Indica el número de meses entre dos fechas.

• **NEW_TIME (d t , zona1, zona2)**.- Dice la fecha y la hora de la zona1, en la zona2.

TABLAS DE FORMATOS DE ZONA

AST	TIEMPO ESTENDAR ATLANTICO
ADT	" DE DÍA "
BST	" ESTANDAR BERING
BDT	" DE DÍA "
CST	" ESTANDAR CENTRAL
CDT	" DE DÍA "
EST	" ESTANDAR ESTE
EDT	" DE DÍA "
GMT	" ESTANDAR GREENWICH
HST	" ESTANDAR ALASKA - HAWAII
HDT	" DE DÍA "
MST	" ESTANDAR MOUNTAIN
MDT	" DE DÍA "
NST	" ESTANDAR NEWFOUNDLAND
PST	" ESTANDAR PACIFICO
PDT	" DE DÍA "
YST	" ESTANDAR YUKON
YDT	" DE DÍA "

• **NEXT_DAY (d, cadena)**.- Retorna el día siguiente a la fecha 'd', indicando el día de la semana (cadena).

• **ROUND (d [,format])**.- Redondea la fecha 'd' al formato indicado en format.

• **SYSDATE**.- Fecha actual del sistema.

2.5. FUNCIONES DE CONVERSIÓN

- **CONVERT (cadena, dest_set [,source_set])**.- Convierte el tipo de caracteres de la cadena (source_set) al tipo de caracteres de dest_set. Identificadores de juego de caracteres:

Us7ascii	ASCII 7-bit.
We8dec	DEC Europa occidental 8-bit.
We8hp	HP Europa occidental laserjet 8-bit.
F7dec	DEC francés 7-bit.
We8ebcdic500	IBM Europa occidental code page 500.
We8pc850	IBM Europa. Este juego de caracteres es usado por la mayoria de PC's . También lo utiliza Oracle cuando trabaja con PC's.

- **TO_CHAR (etiquetas)**.- Convierte la etiquetas (tipo de dato mlslabel) a varchar2.

- **TO_LABEL (cadena)**.- Convierte la cadena a etiqueta (o tipo de dato mlslabel).

- **TO_MULTI_BYTE (cadena)**.- Devuelve una cadena de caracteres, remplazando todos los bytes por su equivalente en caracteres multibyte.

- **TO_SINGLE_BYTE (cadena)**.- La inversa a la anterior.

2.6. FUNCIONES DE GRUPO

- **AVG ([DISTINT | ALL] col)**.- Da la media de la columna.

- **COUNT (*|[DISTINCT| ALL] col)**.- Cuenta el numero de filas de la consulta.

- **GLB ([DISTINCT | ALL] etiqueta)**.- Devuelve la etiqueta (de salto) mayor de entre las mas bajas.

- **LUB([DISTINCT| ALL] etiqueta)**.- Devuelve la etiqueta (de salto) menor de entre las mas altas.

- **MAX ([DISTINCT | ALL] col)**.- El máximo valor de la lista.

- **MIN ([DISTINCT | ALL] col)**.- El mínimo valor de la lista.

- **STDDEV ([DISTINCT | ALL] col)**.- Obtiene la desviación típica de del objeto seleccionado.

- **SUM ([DISTINCT | ALL] col)**.- Obtiene la suma de la lista .

- **VARIANCE ([DISTINCT | ALL] col)**.- Obtiene la varianza del elemento seleccionado.

2.7. OTRAS FUNCIONES

• **DECODE (Base_expr, compare1, valor1, compare2, valor2, ... por defecto.)**.- Realiza una comparación y si toma el mismo valor que alguno de los compare(1...n) coloca el valor de su valor(1...n) correspondiente.

• **DUMP (expr [, numero_f [, start_pos] [,longitud]])**.- devuelve un varchar2 que contiene la información sobre la representación de expr. Numero_f especifica la base de los valoresretornados de acuerdo a la siguiente tabla:

numero_f	formato de salida
8	Notación octal.
10	" decimal.
16	" hexadecimal.
17	caracteres simples.

Start_pos y longitud indican el principio y la longitud en bytes.

• **GREATEST (EXPR1 [,EXPR2]...)**.- Devuelve la expresión mayor de las suministradas.

• **LEAST (EXPR1 [,EXPR2]...)**.- Devuelve la expresión menor.

• **NVL (EXPR1, EXPR2)**.- Si la expresión (espr1) analizada tiene el valor a null se muestra el valor expr2.

• **UID**.- Devuelve un entero que identifica al usuario.

• **USER**.- Devuelve un varchar2 con el nombre del usuario.

• **USERENV (opción)**.- Devuelve un varchar2 que contiene la información sobre la sesión actual. Opciones:

'isdba'	retorna si es una sesión del administrador.
'language'	idioma de la sesión.
'terminal'	sistema operativo de la terminal.

• **VSIZE (value)**.- Devuelve el número de bytes, el tamaño de la columna.

3. PSEUDO COLUMNAS

- **LEVEL** .- Se usan solo dentro del select, cuando se implementa un árbol jerárquico, devolviendo el nivel actual dentro del árbol.

- **ROWID**.- Se usa para seleccionar filas en una consulta. Devuelve el numero identificador de cada línea.

- **ROWNUM** .- Devuelve el número de líneas en una consulta.

4. CONTROL DE TRANSACCIONES

Una transacción es una serie de órdenes SQL que se completan o fallan como una unidad. Las transacciones son un componente estándar de las bases de datos y sirven para evitar la inconsistencia de los datos. Una transacción comienza con la primera orden SQL emitida después de terminar la transacción anterior, o con la primera orden SQL después de conectarse con la base de datos. La transacción termina con la orden COMMIT o ROLLBACK.

Cuando se da una orden COMMIT a la base de datos, termina la transacción y:

⇨ Se hace permanente todo el trabajo realizado por la transacción

⇨ Otras sesiones pueden ver los cambios realizados por la transacción

⇨ Se liberan todos los bloques establecidos por la transacción

La sintaxis de la orden COMMIT es:

```
COMMIT [WORK];
```

La palabra clave opcional WORK tiene como único objeto facilitar la comprensión del propósito de esta orden. Hasta que la transacción se confirme mediante COMMIT, solo la sesión que esté ejecutando la transacción podrá ver los cambios hechos por ella

Cuando de da a la base de datos una orden ROLLBACK, la transacción termina y:

⇨ Todo el trabajo hecho por la transacción se deshace, como si no se hubieran emitido las órdenes correspondientes.

⇨ Se liberan todos los bloques establecidos por la transacción

La sintaxis de la orden ROLLBACK es:

```
ROLLBACK [WORK];
```

Donde la palabra clave WORK es opcional y sólo tiene por objeto hacer más claro el cometido de esa orden. Se suele utilizar una orden ROLLBACK explícita cuando se detecta un error en el programa que impide continuar con el trabajo. Si una sesión se desconecta de la base de datos sin terminar la transacción actual con COMMIT o ROLLBACK, la transacción es automáticamente cancelada por la base de datos.

NOTA IMPORTANTE

Cuando atacamos a la base de datos y queremos guardar información de una consulta en una variable lo hacemos de la siguiente forma:

```
SELECT nombre_campo INTO variable
FROM nombre_tabla
WHERE condición
```

El valor del campo se guarda en la variable y se pone una condición porque si no se pusiera, como se recorre toda la tabla, siempre se guardaría el valor de la última fila. La variable ha de estar previamente definida en la sección declare. En temas posteriores veremos cómo se guardan en una variable los valores de todas las filas de una tabla.

Ejemplo: Guardar en una variable que se denomine salario, el salario que cobra un empleado introducido por teclado y después visualizar el valor de la variable salario.

```
declare
        empleado employee.employee_id%type;
        salario employee.salary%type;
        begin
        empleado:='&empleado';
        select salary into salario
        from employee
        where employee_id=empleado;
        dbms_output.put_line('el empleado '||empleado|| ' tiene un salario de '
        ||salario||' euros');
end;
/
```

Comprobamos que con la siguiente sentencia select sale lo mismo

```
Select employee_id, salary
From employee
Where employee_id=empleado_introducido;
```

Se pueden utilizar tantas variables como campos tengamos en el select. Ahora bien, solo podemos extraer una fila de la consulta.

El extraer más de una fila de la consulta y guardarlo en variables se verá en temas posteriores.

5. EJERCICIOS PROPUESTOS

1º.- Realizar un bloque Pl/Sql en el que introducimos una letra por teclado y visualizamos cuál es su número ASCII.

2º.- Realizar un bloque Pl/Sql en el que introducimos el valor de los catetos de un triángulo y visualizamos el valor de su hipotenusa.

Nota $h = \sqrt{C^2 + c^2}$

3º.- Realizar un bloque Pl/Sql que nos permita visualizar el abecedario de forma que cada letra aparezca en una línea.

4º.- Realizar un bloque Pl/Sql en el que introducimos una palabra por teclado y visualizamos cada una de sus letras en una línea.

5º.- Realizar un bloque Pl/Sql en el que introducimos una palabra por teclado y visualizamos si esta es palíndroma. Una palabra es palíndroma si se lee igual de izquierda a derecha que de derecha a izquierda.

6º.- Realizar un bloque Pl/Sql en el que introducimos tres números por teclado y visualizamos el mayor de los números introducidos.

7º.- Realizar un bloque Pl/Sql en el que introducimos el numero de un empleado e incrementamos el salario de dicho empleado en función del número de empleados de los que es jefe.

- Si no es jefe de ningún empleado será 30 euros.
- Si es jefe de un empleado la subida será 50 euros.
- Si es jefe de 2 empleados la subida será 70 euros.
- Si es jefe de más de 2 empleados la subida será 100 euros.
- Además, si el empleado es el presidente se incrementará el salario en 30 euros.

8º.- Realizar un bloque Pl/Sql en el que introducimos el número de un empleado y modificamos el salario o la comisión de dicho empleado en función de las siguientes premisas:

- Si el empleado pertenece al departamento 13 su comisión será la comisión media del departamento al que pertenece.
- Si el empleado pertenece al departamento 14 su comisión será la comisión mínima del departamento al que pertenece.
- Si el empleado pertenece al departamento 23 su comisión será la comisión máxima del departamento al que pertenece.
- Si el empleado pertenece al departamento 10 su salario será el salario mínimo del departamento al que pertenece.
- Si el empleado pertenece al departamento 12 su salario será el salario máximo del departamento al que pertenece
- Si el empleado pertenece a cualquier otro departamento su salario será el salario medio del departamento al que pertenece.

9º.- Crear una vista denominada salarios_caros, con los mismos campos que la tabla empleados, pero en castellano, en donde vamos a guardar todos los empleados que tienen salarios superiores a 2500.

10º.- Realizar un bloque Pl/Sql en el que insertemos un empleado a través de la vista cuyo código de empleado sea el código del ultimo empleado más uno, de la tabla empleados. Con arreglo a las siguientes premisas:

- Si el empleado es del departamento 12 su salario es la media del salario de los empleados del departamento que pertenecen a la vista.
- Si el empleado es del departamento 20 su salario es el mínimo del salario de los empleados del departamento que pertenecen a la vista.
- Si el empleado es del departamento 30 su salario es el máximo del salario de los empleados del departamento que pertenecen a la vista.

11º.- Realizar un bloque Pl/Sql en el que introducimos el número de un empleado y actualizamos la última línea de pedido, del cliente que ha realizado el pedido de mayor cuantía. De forma que el campo Quantity pasa a valer 10, 20, 30, 40, 50, 60, 70,80, 90 ó 100 según sea la última línea 1, 2, 3, 4, 5, 6, 7, 8, 9 ó 10.