**UT\_5 REALIZACIÓN DE CONSULTAS**

* 1. Introducción
  2. Sentencia SELECT
  3. Operadores
  4. Funciones

**5.1 Introducción**

La primera fase del trabajo con cualquier base de datos comienza con sentencias DDL (Lenguaje de Definición de Datos) que nos permiten definir las estructuras, tablas, donde agrupar la información para más tarde poder recuperarla y trabajar con ella.

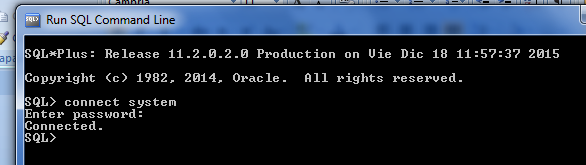
La siguiente fase será manipular los datos, es decir, trabajar con sentencias DML (Lenguaje de Manipulación de Datos) que están orientadas a consultas y a manipulación de los objetos creados. Aquí tenemos las siguientes sentencias: SELECT, INSERT, DELETE y UPDATE, que permiten respectivamente, consultar, insertar, borrar y modificar datos.

Todas las sentencias SQL se pueden ejecutar desde distintos entornos de ejecución:

|  |  |
| --- | --- |
| Desde Inicio->Programas:    Elegir a continuación la opción de Application Express y Comandos SQL. | Desde el entorno de SQL\*Plus:    Ejecutar desde línea de comandos. |

También podrá ejecutarse desde alguna herramienta de acceso a Oracle, SQL Developer o SQLTools.

Si elegimos “línea de comandos”, desde SQL\*Plus, nos aparecerá una pantalla en la que debemos indicar primero que queremos conectarnos, con qué usuario y contraseña. Usaremos la orden **connect.**

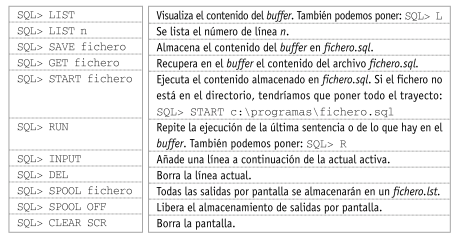
****

Cualquier sentencia SQL que queramos ejecutar en este entorno, se escribirá, se pulsará **Intro** y a continuación comenzará su ejecución.

Para más información, consultar la página:

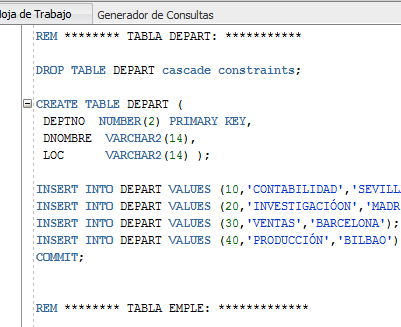
<http://www.infor.uva.es/~jvegas/cursos/bd/sqlplus/sqlplus.html>

Algunos comandos que se pueden utilizar desde el prompt son los siguientes:

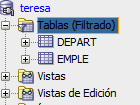


Crearemos dos tablas para realizar las primeras consultas, DEPART y EMPLE que contendrán respectivamente datos de los departementos y de los empleados de una empresa. Se podrán generar ejecutando un fichero script desde la herramienta gráfica o desde el entorno de Sql\*Plus.

Desde SQLDeveloper, copiar el fichero “creaEmpleDepart.sql” en el editor y ejecutarlo como script eligiendo el icono 



Si todo va bien, se generarán dos tablas con datos:



También se puede ejecutar desde el entorno de SQL\*Plus y el resultado sería el mismo:

|  |  |
| --- | --- |
|  | Primero nos conectamos y a continuación ejecutamos la orden:  SQL> @rutaHastaElFichero |

**5.2 Sentencia SELECT**

La sentencia SELECT nos permite recuperar o seleccionar los datos, de una o varias tablas y consta de cuatro partes básicas:

* Cláusula SELECT seguida de la descripción de los que se desea ver, es decir, de los nombres de las columnas que queremos que se muestren separadas por “,”. Este bloque es obligatorio.
* Cláusula FROM seguida del nombre de la/s tabla/s de las que proceden los datos. Es obligatoria.
* Cláusula WHERE seguida de un criterio de selección o condición. Es opcional.
* Cláusula ORDER BY seguida por el criterio de ordenación. Es opcional.

Una primera aproximación a la sintaxis de la sentencia SELECT sería:

**SELECT** [ ALL | DISTINCT ] columna1, columna2, …

**FROM** tabla1, tabla2,

[WHERE condición1, condición2, …]

[ORDER BY criterioOrdenación]

Las cláusulas ALL y DISTINCT son opcionales. Con ALL seleccionaremos todas las filas estén o no repetidas. Es el valor por defecto y no se suele especificar. Con la opción DISTINCT, no se visualizan las filas que estén repetidas.

* Seleccionar los códigos de departamento de la tabla empleados, comprobar la diferencia de utilizar la opción DISTINCT o no.

A la hora de especificar las columnas tendremos en cuenta lo siguiente:

* Delante del nombre de la columna se puede indicar el nombre de la tabla de la que proceden.
* Si queremos incluir todas las columnas en la consulta, podemos indicar “\*” como comodín de columnas. SELECT \* from …
* A cada columna le podemos asociar un alias que será el que aparezca en el resultado y no el nombre de la columna. Se indicará a continuación del nombre de la columna.

### SELECT fnac “Fecha de nacimiento”

### FROM USUARIOS;

* El nombre de las columnas se puede sustituir por constantes, expresiones o funciones SQL.

### SELECT (salario \* 12 ) “Salario anual”

### FROM empleados;

* Hacer una consulta que permita seleccionar todas las columnas de la tabla DEPART.
* Hacer una consulta sobre la tabla DEPART especificando un alias para cada columna.
* Consultar el nombre, salario y fecha de alta en la empresa de todos los empleados.

**Cláusula FROM**

Dentro de la orden SELECT, la cláusula FROM es la que nos permite establecer de dónde se va a obtener la información, de qué tabla procederán las columnas que hemos indicado.

Aquí se definen los nombres de las tablas. Si se necesitan más de una, sus nombres deben aparecer separados por comas.

También se puede añadir el nombre del usuario que es propietario de la tabla con la sintaxis: *USUARIO.TABLA.*

Podemos asociar un alias a las tablas para simplificar las órdenes de la siguiente forma:

### SELECT \* FROM USUARIOS U;

**Cláusula WHERE**

Permite seleccionar un subconjunto de filas de una tabla, solo se devolverán las que cumplan la condición que se especifique.

A continuación de la cláusula WHERE se indicará la condición o criterio de búsqueda que podrá ser más o menos sencillo y para crearlo se pueden conjugar operadores de distinto tipo, funciones o expresiones más o menos complejas.

Ej: si de nuestra tabla USUARIOS queremos el nombre y apellidos de aquellos que son mujeres:

### SELECT nombre, apellidos

### FROM USUARIOS

### WHERE sexo=’M’;

Ejemplos de condiciones:

WHERE NOTA=5

WHERE (NOTA>=10) AND (CURSO=1)

WHER (NOTA IS NULL) OR (UPPER(NOMALUM)= ‘PEDRO’)

* Obtener el nombre y el salario de los empleados que sean ‘ANALISTAS’.
* Obtener el nombre y el oficio de los empleados del departamento 10.

select APELLIDO NOMBRE, OFICIO from EMPLE WHERE DEPTNO=10;

* Obtener el nombre, la fecha de alta y el salario de los empleados cuyo director sea el 7698.

select APELLIDO NOMBRE, FECHAALT, SALARIO from EMPLE WHERE DIR=7698;

* Obtener nombre y oficio de los empleados que cobran más de 1000 euros.

select APELLIDO NOMBRE, OFICIO from EMPLE WHERE SALARIO>1000;

* Obtener el nombre, el salario y la comisión de los empleados que ganan más de 1500 euros teniendo en cuenta su comisión.

select APELLIDO NOMBRE, SALARIO, COMISION from EMPLE WHERE COMISION+SALARIO>1500 OR SALARIO>1500;

* Obtener el nombre, salario y fecha de alta de los empleados que ganan más de 1000 euros y que sean del departamento 20.

select APELLIDO NOMBRE, SALARIO, FECHAALT from EMPLE WHERE SALARIO>1000 AND DEPTNO=20;

* Obtener nombre y fecha de alta de los empleados dados de alta después del ‘01/01/2011’.

select APELLIDO NOMBRE, FECHAALT from EMPLE WHERE FECHAALT>'01/01/2011';

**Cláusula ORDER BY**

Esta cláusula nos permite especificar un criterio de ordenación en nuestra consulta. Se puede ordenar por varias columnas a la vez y además cada una siguiendo determinado orden: ascendente o descendente.

**SELECT** [ ALL | DISTINCT ] columna1, columna2, …

**FROM** tabla1, tabla2,

[WHERE condición1, condición2, …]

[ORDER BY columna1 [ASC | DESC], columna2 [ASC | DESC], ….]

Se utilizarán las palabras reservadas ASC o DESC que indicarán respectivamente, orden ascendente o descendente y si no se especifica nada, la ordenación por defecto será ascendente.

También se puede ordenar utilizando una expresión creada con columnas o con funciones SQL.

En el siguiente ejemplo, se ordena por apellidos y en caso de empate, se ordena por nombre:

### SELECT nombre, apellidos

### FROM USUARIOS

### ORDER BY apellidos, nombre;

Podemos sustituir el nombre del campo por la posición que ocupa en la lista de columnas

En el siguiente ejemplo, se obtiene un listado ordenado por localidad:

### SELECT nombre, apellidos, localidad

### FROM USUARIOS

### ORDER BY 3;

* Obtener un listado de los datos de los empleados ordenado por nombre.

SELECT \*FROM EMPLE ORDER by apellido;

* Obtener un listado con nombre y oficio de los empleados ordenado por salario.

SELECT apellido, oficio FROM EMPLE ORDER by salario;

* Obtener un listado ordenado ascendentemente por nombre y descendente por salario que contenga: nombre, oficio y salario de los empleados.

SELECT apellido, oficio, salario FROM EMPLE ORDER by apellido asc,salario desc;

* Obtener los datos de los empleados cuyo oficio sea ‘vendedor’.

SELECT \*FROM EMPLE Where oficio='VENDEDOR';

* Obtener el nombre, oficio y salario de los empleados que se apelliden ‘ARROYO’.

SELECT apellido, oficio, salario FROM EMPLE Where apellido='ARROYO';

**5.3 Operadores**

Para crear las expresiones que se pueden indicar en la cláusula WHERE necesitamos operadores que son símbolos que permiten realizar operaciones matemáticas, concatenar cadenas o hacer comparaciones por ejemplo.

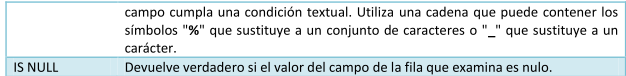
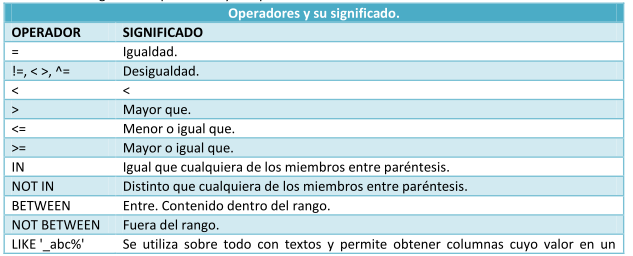
En Oracle se distinguen cuatro tipos de operadores:

* Relacionales o de comparación.
* Aritméticos.
* De comparación.
* Lógicos.

**Operadores de comparación**

También se conocen como relacionales y nos permiten comparar expresiones, si el resultado de la comparación es correcto, la expresión considerada es verdadera, en caso contrario, es falsa.

Los operadores de comparación son los siguientes:



El operador **LIKE** se utilizará de la siguiente manera:

*WHERE columna LIKE ‘caracteres especiales’*

El operador **IS NULL** se utilizará:

*WHERE columna IS NULL / WHERE columna IS NOT NULL*

El operador **IN** permite comprobar si una expresión pertenece o no a un conjunto de valores y su formato es:

*<expresión> [NOT] IN (lista de valores separados por comas)*

El operador **BETWEEN** tiene el siguiente formato:

*<expresión> [NOT] BETWEEN valorInicial AND valorFinal*

* Datos de los empleados cuya comisión es mayor que cero.

select \* from emple where comision > 0;

* Apellido y oficio de los empleados cuya comisión es nula (que no tienen comisión).

select apellido, oficio from emple where comision is null;

* Apellido y oficio de los empleados cuya comisión es no nula.

select apellido, oficio from emple where comision is not null;

* Apellido y sueldo de los empleados que sean analistas, vendedores o empleados simples (utilizar el operador IN).

select apellido, salario from emple where oficio IN ('ANALISTA','VENDEDOR', 'EMPLEADO');

* Apellido y sueldo de los empleados que NO sean analistas, NI vendedores NI empleados simples (utilizar el operador NOT IN).

select apellido, salario from emple where oficio NOT IN('ANALISTA','VENDEDOR', 'EMPLEADO');

* Apellido y oficio de los empleados cuyo salario está entre 1000 y 2000 (operador BETWEEN).

select apellido, oficio from emple where salario between 1000 AND 2000;

* Apellido y oficio de los empleados cuyo salario NO está entre 1000 y 2000 (operador NOT BETWEEN).

select apellido, oficio from emple where salario not between 1000 AND 2000;

* Apellido y oficio de los empleados cuyo apellido empieza por ‘A’.

select apellido, oficio from emple where apellido Like 'A%';

* Apellido y sueldo de los empleados cuyo apellido contiene una ‘A’ al menos.

select apellido, salario from emple where apellido Like '%A%';

* Apellido y fecha de alta de los empleados cuyo apellido termina por ‘Z.

select apellido, FECHAALT from emple where apellido Like '%Z';

* Apellido y departamento de los empleados cuyo apellido empieza por ‘S’ y contiene una ‘L’.

select apellido, DEPTNO from emple where apellido Like 'S%L%';

* Apellido y director de los empleados cuyo apellido tiene tres caracteres.

select apellido, DIR from emple where apellido Like '\_\_\_';

* Apellido y salario de los empleados cuya segunda letra es ‘A’.

select apellido, salario from emple where apellido Like '\_A%';

**Operadores aritméticos y de concatenación**

Los operadores aritméticos permiten realizar cálculos con valores numéricos y son los siguientes:



Permiten obtener salidas en las que una columna es el resultado de un cálculo y no un campo de una tabla. Estos campos calculados se obtienen a través de la sentencia SELECT poniendo a continuación la expresión que queramos. Esta consulta no modifica los valores originales de las columnas de la tabla, sólo muestra una columna nueva con los valores calculados.

Por ejemplo, nombre y salario anual de los empleados (añadirle un alias a la columna)

### SELECT apellido, salario \* 12

### FROM EMPLEADOS;

* Para los empleados del departamento 10 obtener su apellido, su salario actual y el resultado de incrementarle un 5% dicho salario.

select apellido, salario,salario\*1.05 "Salario aumentado" from emple;

Cuando una expresión aritmética se calcula sobre valores NULL, el resultado es NULL.

### SELECT APELLIDO, SALARIO,COMISION, SALARIO + COMISION

### FROM EMPLE;

Para resolver este problema se utiliza la función NVL que permite sustituir el valor NULL por 0.

### SELECT APELLIDO, SALARIO,COMISION, SALARIO + NVL(COMISION,0)

### FROM EMPLE;

SELECT APELLIDO, SALARIO,NVL(COMISION,0), SALARIO FROM EMPLE;

Para concatenar cadenas de caracteres existe el operador de concatenación “||” y Oracle permite convertir automaticamente valores numéricos a cadenas para una concatenación.

Obtener en una sola columna el apellido y el oficio de todos los empleados

### SELECT apellido || ' ' || oficio

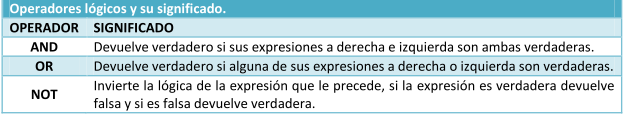
### FROM EMPLE;

select apellido ||' '|| oficio "apellido y oficio" from emple;

**Operadores lógicos**

Permiten evaluar un conjunto de expresiones y comprobar que se cumple una condición de ellas, todas o ninguna. Estos operadores pueden combinarse de forma ilimitada pero teniendo en cuenta el orden de prioridad establecido, si no utilizamos paréntesis, es posible que los resultados no sean los deseados.

Son los siguientes:



* A partir de la tabla EMPLE, obtener el apellido, salario y código de departamente de los empleados cuyo salario sea mayor de 2000 en los departamentos 10 o 20 (Para comprobar la prioridad definir la consulta con paréntesis y sin ellos)

select apellido, salario, deptno from emple where salario >2000 and deptno in (10,20);

En Oracle el orden de precedencia de los operadores es:

1. La multiplicación (\*) y la división (/) se evalúan al mismo nivel.
2. A continuación sumas(+) y restas(-).
3. Concatenación (||).
4. Todas las comparaciones (<, >, …)
5. Operadores IS NULL, IS NOT NULL, LIKE, BETWEEN.
6. NOT.
7. AND.
8. OR.

Para modificar este orden se deben utilizar paréntesis.

**5.4 Funciones**

En todos los Sistemas Gestores de Bases de Datos existen funciones ya creadas que facilitan la creación de consultas más complejas. Las funciones son operaciones que se realizan sobre los datos y que realizan un determinado cálculo. Para ello necesitan unos datos de entrada llamados parámetros o argumentos y en función de éstos, se realizará el cálculo de la función que se esté utilizando. Normalmente los parámetros se especifican entre paréntesis.

Las funciones se pueden incluir en las cláusulas SELECT, WHERE y ORDER BY.

La sintaxis de las funciones es:

*NombreFunción [ (parámetro1, [parámetro2, …)]*

Se pueden anidar funciones dentro de funciones. En Oracle existen funciones para cada tipo de datos:

* Numéricas
* De cadenas de caracteres
* De manejo de fechas
* De conversión
* Otras

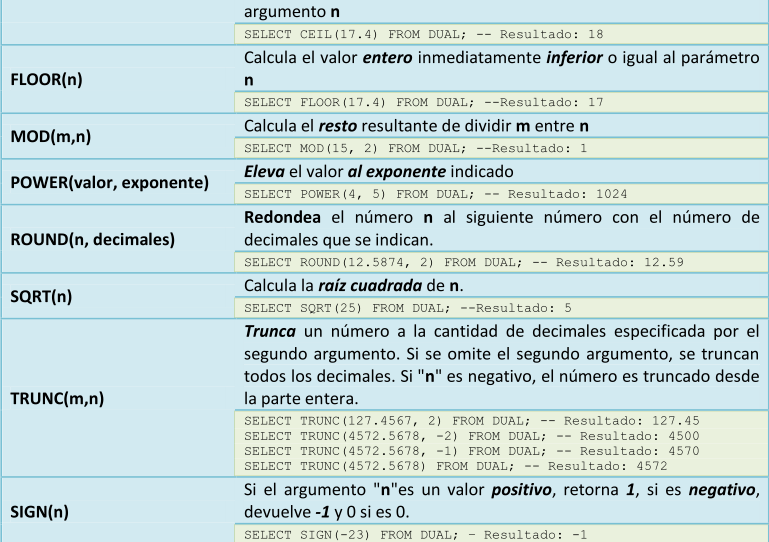
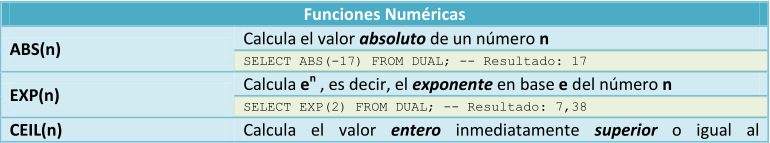
Oracle proporciona una tabla donde se pueden probar todas estas funciones, es la tabla Dual que contiene un solo campo llamado DUMMY y una sola fila.

**Funciones numéricas**

Trabajan con datos de tipo NUMBER e incluye los dígitos de 0 a 9, el punto decimal y el signo menos si es necesario. Estas funciones trabajan con tres clases de números: valores simples, grupos de valores y listas de valores. Algunas modifican los valores sobre los que actúan, otras informan sobre los valores. Se pueden dividir en tres grupos:

* **Funciones de valores simples**

Son funciones que trabajan con números, variables o con una columna de una tabla. Son las siguientes:



* ¿Cuál sería el resultado de ejecutar las siguientes funciones?



* **Funciones de grupos de valores**

También se conocen como funciones de agrupamiento o de agregado. Son funciones que actúan sobre grupos de datos de una columna para obtener un valor. Permiten obtener por ejemplo, la edad media de un grupo de alumnos, la edad del mayor o del más joven, el número de miembros del grupo, etc. Estas funciones ignoran los valores nulos y los cálculos se realizan sin contar con ellos.

La sintaxis de estas órdenes es:

*FUNCION ([ALL | DISTINCT] expresión)*

* Con ALL se tomarán en cuenta todos los valores de la columna. Es el valor por defecto.
* Con DISTINCT se considerarán todas las repeticiones del mismo valor como uno solo.
* El grupo de valores sobre el que actúa la función lo determina el resultado de la expresión que será el nombre de una columna o una expresión basada en una o varias columnas. Por tanto en la expresión nunca podrá aparecer una función de agregado.
* Todas las funciones excepto COUNT ignoran los valores NULL.
* En general, no se pueden mezclar funciones de agregado con nombres de columna simples

Son las siguientes:

* La función **SUM** devuelve la suma de los valores de una expresión y sólo puede utilizarse con valores numéricos. Tiene la siguiente sintaxis:

SUM ([ALL | DISTINCT]) expresión)

### Ejemplo: SELECT SUM (salario) FROM EMPLE;

* La función **COUNT** que permite contar los elementos de un campo (cuenta el número de filas). La sintaxis es:

COUNT ([ALL | DISTINCT]) expresión)

Con expresión se indica el nombre del campo que se desea contar. Puede contar cualquier tipo de datos incluido texto. No se cuentan las filas que tengan valor NULL en el campo a no ser que se especifique COUNT(\*).

* Averiguar cuántas comisiones no nulas hay en la tabla de empleados.
* Contar cuántos departamentos hay en la tabla de empleados.
* Contar cuántos departamentos distintos hay en la tabla de empleados.
* Función **MIN**, devuelve el valor mínimo de una expresión sin considerar los nulos. No se podrá concatenar esta función con otras de grupos de valores.
* Función **MAX**, devuelve el valor máximo de una expresión sin considerar los nulos. Tampoco se puede concatenar con otras funciones de grupos de valores.

Ejemplo:

Obtener el salario mínimo y el máximo de los empleados:

### SELECT min(salario), max(salario) FROM EMPLE;

* Función **AVG**, devuelve el promedio de los valores de un grupo omitiendo los valores nulos. Se aplica a campos numéricos y la expresión asociada puede ser ub nombre de columna o una expresión basada en una columna o varias de una tabla.
* Función **VAR,** devuelve la varianza estadística de todos los valores de la expresión. Se omiten los valores nulos y solo admite datos de tipo numérico.
* Función **STDEV,** devuelve la desviación típica estadística de todos los valores de la expresión. Se omiten los valores nulos y sólo se admite con datos numéricos.

Obtener el salario medio de todos los empleados:

### SELECT avg(salario) FROM EMPLE;

* **Funciones de listas**

Trabajan sobre un grupo de columnas dentro de una misma fila. Comparan los valores para obtener el mayor o el menor. Son las siguientes:

* GREATEST (valor1, valor2, …), que obtiene el mayor valor de la lista.
* LEAST (valor1, valor2, …), que obtiene el menor valor de la lista.

## Ejemplos:

### select greatest(1,2,3), least(1,2,3) from dual;

### select apellido,greatest(salario,comision), least (salario, comision) from emple;

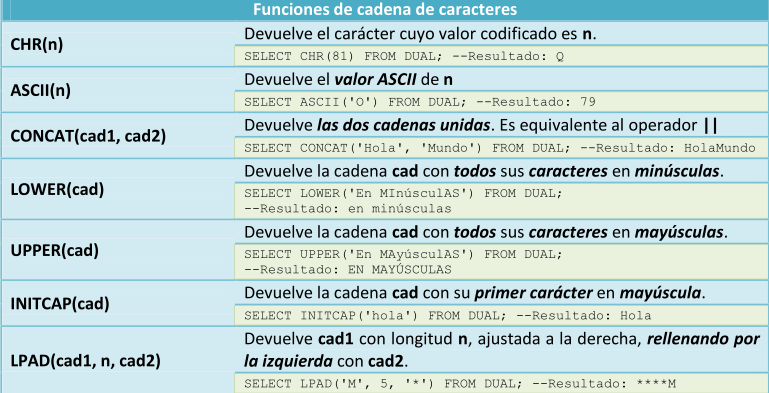
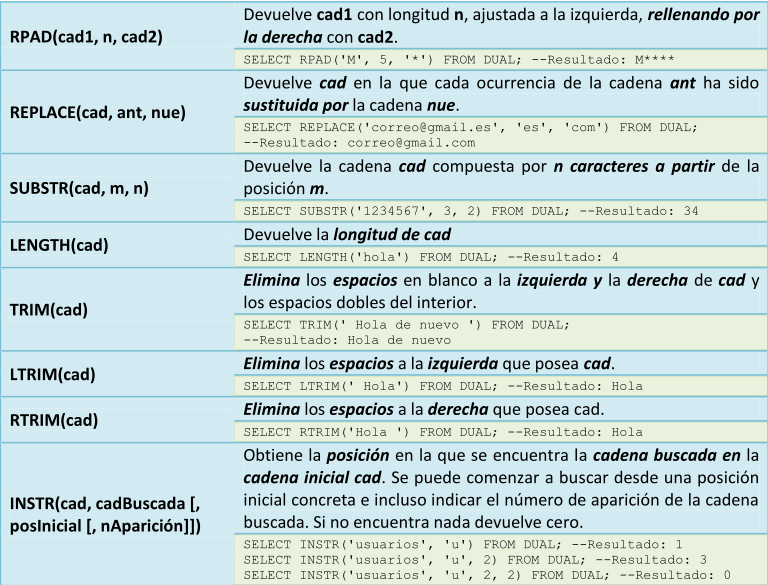
En el segundo caso se puede comprobar que al comparar cualquier valor con null, el resultado es siempre nulo, tanto el más grande como el más pequeño.

También permiten comparar cadenas de caracteres. Ejemplo:

### SELECT GREATEST ('BENITO','JORGE','ANDRES','ISABEL'), LEAST ('BENITO','JORGE','ANDRES','ISABEL') FROM DUAL;

**Funciones de cadenas de caracteres**

Permiten manipular datos de tipo carácter (CHAR) o cadena de caracteres (VARCHAR2). Pueden devolver una cadena o un resultado numérico entero. Las más habituales son:

* Devolver las letras correspondientes a los caracteres ASCII 75 y 365.
* Para cada empleado de la tabla EMPLE obtener la siguiente salida:
  + El apellido es: SANCHEZ
  + El apellido es: ARROYO
  + …
* Visualiza en minúsculas, mayúsculas y con letra capital la cadena: “OraclE y SqL”
* Para cada apellido de los empleados de la tabla EMPLE, obtener una columna con dicho apellido y rellenando por la izquierda hasta longitud 15 con (.) y lo mismo por la derecha con (\*). ( ........SÁNCHEZ SÁNCHEZ\*\*\*\*\*\*\*\*)
* En la cadena ‘BLANCO Y NEGRO’ , sustituir ‘O’ por ‘AS’. Probar a no indicar el tercer parámetro.
* Partiendo de la cadena ‘ABCDEF’, obtener en una columna dos caracteres a partir de la tercera posición, en otra columna otros dos caracteres a partir de la tercera posición empezando por el final de la cadena y en una tercera columna , la cadena a partir de su cuarta posición.
* Visualizar el apellido y su primera letra para los empleados del departamento 10 de la tabla EMPLE. (CEREZO C).
* A partir de la cadena ‘LOS PILARES DE LA TIERRA’, sustituye cada vocal por su minúscula. (buscar ayuda sobre la función TRANSLATE).
* ¿Qué ocurre con la ejecución de la orden:

**SELECT TRANSLATE (‘LOS PILARES DE LA TIERRA’, ‘LAEIOU’,’l’) FROM DUAL;**

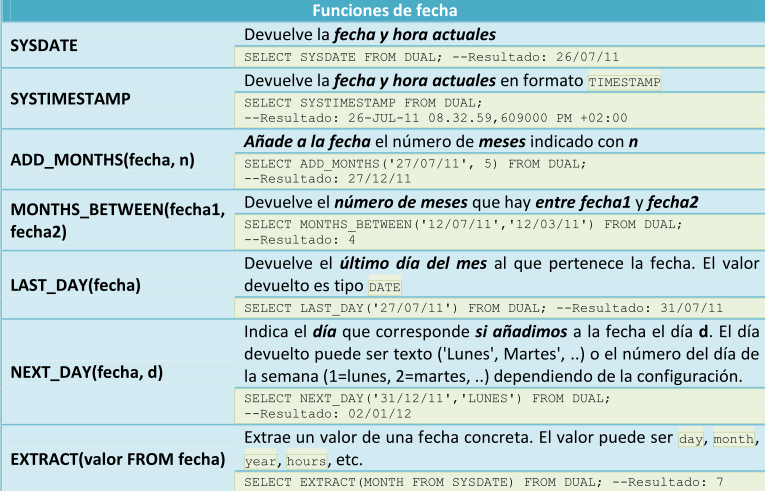
**¿Por qué se obtiene ese resultado?**

* Obtener el valor ASCII de ‘A’.
* A partir de la cadena ‘VUELTA CICLISTA A TALAVERA’, encuentra la primera ocurrencia ‘TA’, la segunda y la primera desde el final.
* A partir de la tabla EMPLE, encontrar la posición de la primera ocurrencia en cada apellido de la letra ‘A’. Visualizar además el apellido.
* Visualizar para cada empleado de la tabla EMPLE, su apellido y la longitud de su apellido.

**FUNCIONES PARA EL MANEJO DE FECHAS**

Oracle dispone de dos tipos de datos para manejar fechas: DATE, que almacena fechas concretas incluyendo a veces la hora y TIMESTAMP, que almacena un instante de tiempo y puede incluir hasta fracciones de segundo.

Las funciones de fechas más habituales son las siguientes:



* Visualizar para los empleados del departamento 10 de la tabla EMPLE, su apellido, su fecha de alta y dicha fecha sumados 6 meses.
* Obtener para los empleados del departamento 10, el último día del mes para su fecha de alta.
* Calcula tu edad como numero entero mediante la función months\_between.
* Con la función next\_day, averigua la fecha del próximo jueves a partir de la fecha actual.
* Extrae de la fecha actual en tres columnas independientes, el año, el mes y el día.
* Obtener para cada empleado el año de su fecha de alta.
* Averiguar en cuántos años distintos se han dado de alta los empleados.

**FUNCIONES DE CONVERSIÓN**

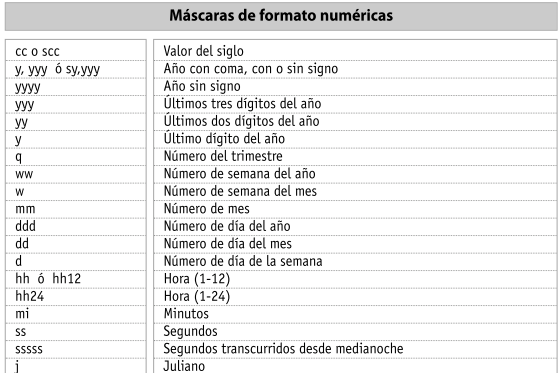
Oracle convierte de manera automática datos para que el resultado de una expresión tenga sentido, por ejemplo, se pasa de texto a número o de número a texto, de texto a fecha y viceversa. Además existen funciones de conversión que permiten realizar conversiones de modo explícito y son las siguientes:

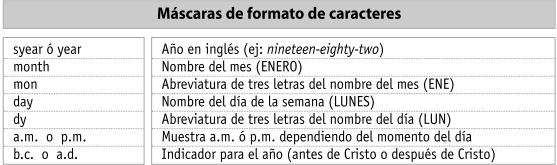
* TO\_NUMBER (cad, formato). Convierte la cadena cad a tipo NUMBER según el formato especificado. Dicha cadena ha de contener números, el carácter decimal o el signo menos a la izquierda. No puede haber espacios entre los números, ni otros caracteres. Ejemplo de TO\_NUMBER:

### SELECT TO\_NUMBER('-12345'), TO\_NUMBER('-12322.22','999999.99') FROM DUAL;

* TO\_CHAR (d, formato), convierte un número o fecha a cadena de caracteres según el formato especificado. Se utiliza fundamentalmente para fechas ya que de número a texto se realiza de forma implícita.

El formato especificado en esta función viene expresado por la siguiente tabla de máscaras de formato:





* A partir de la tabla EMPLE, obtener la fecha de alta formateada para los empleados del departamento 10. La fecha debe aparecer con el nombre del mes con todas sus letras (month, Month o MONTH), el número del dia del mes (dd) y el año (yyyy).
* Visualizar la fecha actual con el formato “1 de Enero de 2016”

Por defecto, el formato para la fecha viene definido por el parámetro NLS\_TERRITORY que especifica el idioma para el formato de la fecha, los separadores de miles, el signo decimal y el símbolo de la moneda. Este parámetro se inicializa al arrancar Oracle. Para el idioma español, el valor de este parámetro es NLS\_TERRITORY=”SPAIN”.

Si queremos consultar los parámetros de inicio de la BD:

### SELECT \* FROM NLS\_DATABASE\_PARAMETERS; //Para la base de datos

### SELECT \* FROM NLS\_INSTANCE\_PARAMETERS; // Para la instancia

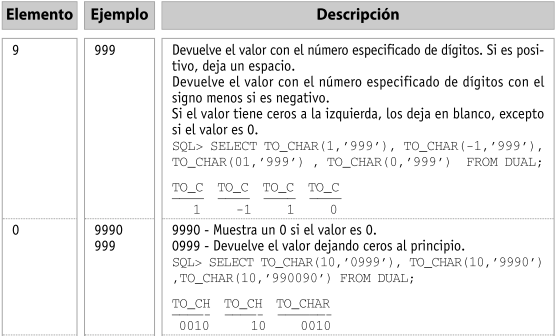
### SELECT \* FROM NLS\_SESSION\_PARAMETERS; // Para la sesión

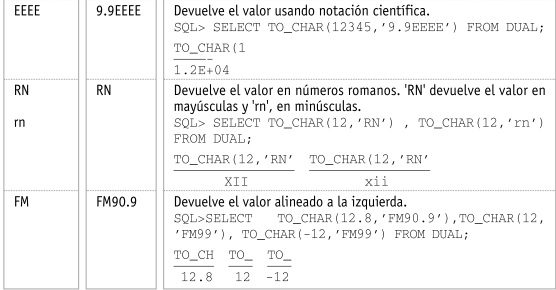
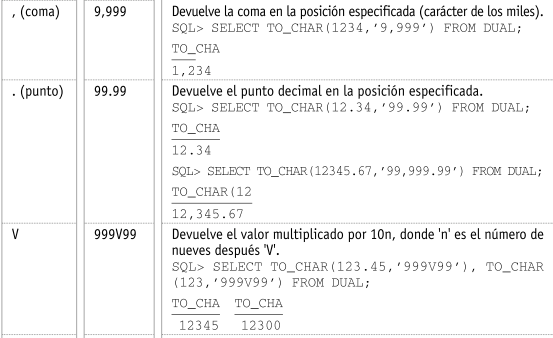
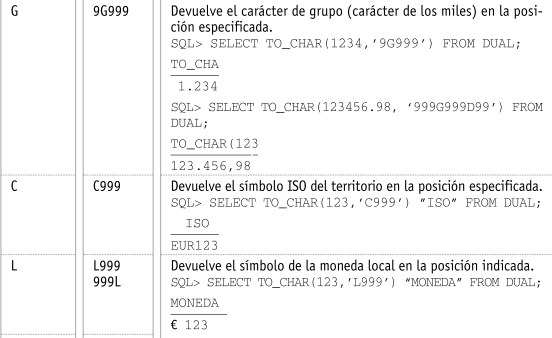
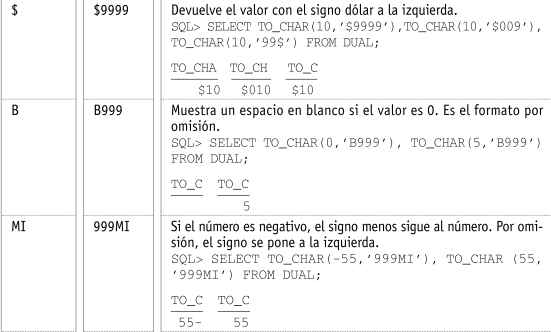
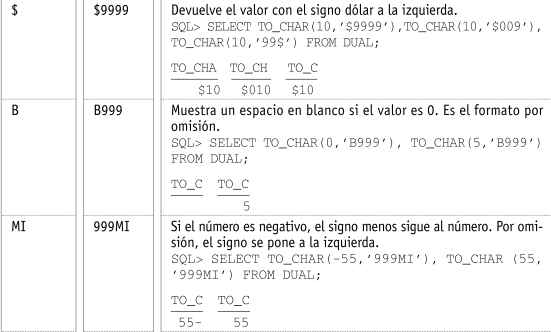
Podemos cambiar el valor por omisión para la fecha usando la orden ALTER SESSION y el parámetro NLS\_DATE\_FORMAT. Por ejemplo, que la fecha aparezca como “dia/ nombre mes/ año” pondremos:

### ALTER SESSION SET NLS\_DATE\_FORMAT=’DD/month/YYYY’;

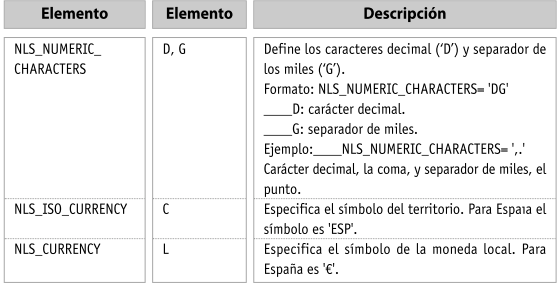
También podemos cambiar el lenguaje para nombrar los meses y los días con el parámetro NLS\_DATE\_LANGUAGE.

Los formatos numéricos que se pueden utilizar con la función TO\_CHAR son los siguientes:





Los caracteres devueltos en algunos de estos formatos se especifican inicializando una serie de parámetros llamados parámetros NLS y son los siguientes:



Para cambiar el valor de estos parámetros se utiliza la orden ALTER SESSION. Por ejemplo, para definir como carácter de miles el (\*) y como carácter decimal (/), se ejecutará la siguiente orden:

### ALTER SESSION SET NLS\_NUMERIC\_CHARACTERS=’/\*’;

### SELECT TO\_CHAR(12345.67 , ‘999G999D999) FROM DUAL;

Los valores para el carácter decimal y de miles permanecerán hasta que se termine la sesión o hasta que de nuevo se ejecute la orden para cambiarlos

### ALTER SESSION SET NLS\_NUMERIC\_CHARACTERS=’,.’;

También podemos cambiar el símbolo de la moneda para que aparezca ‘€’ o ‘Pesetas’ con el parametro NLS\_CURRENCY. Por ejemplo:

### ALTER SESSION SET NLS\_CURRENCY=’PESETAS’;

### SELECT TO\_CHAR(123, ‘ L999’) FROM DUAL;

* TO\_DATE (cad, formato), convierte textos a fechas según el formato especificado. Puede utilizar las mismas máscaras de formato que TO\_CHAR.

Por ejemplo, para convertir una cadena a tipo date:

### SELECT TO\_DATE(‘01012016’) FROM DUAL;

### SELECT TO\_DATE('010116', 'YYMMDD') FROM DUAL;

El año aparecerá con cuatro o dos cifras dependiendo del parámetro NLS\_DATE\_FORMAT.

Ejemplo: obtener el nombre del mes a partir de la cadena ‘01012016’, primero será necesario convertir la cadena a fecha:

### SELECT TO\_CHAR(TO\_DATE('01012016'), 'Month') MES FROM DUAL;

**Otras funciones**

Existen otras funciones que por sus características no pueden ser incluidas en los grupos anteriores y son las siguientes:

* **DECODE (expr1, cond1, valor1 [,cond2 ,valor2, …] default )**. Esta función evalúa una expresión **expr1**, si se cumple la primera condición **cond1** devuelve el valor **valor1,**  en caso contrario se pasa a la siguiente condición y así hasta que se cumpla una de ellas o se devuelve un valor por defecto expresado por **default** si no se cumple ninguna.

Ejemplo: de la tabla EMPLE, seleccionar todas las filas y codificar el OFICIO. Si el oficio es PRESIDENTE, codificar con un 1; si es EMPLEADO, con un 2; en cualquier otro caso, codificar con 5.

### SELECT apellido, oficio, decode(upper(oficio), 'PRESIDENTE', 1, 'EMPLEADO' ,2, 5) FROM emple;

* **VSIZE (expresión).**  Devuelve el número de bytes que ocupa una expresión o nulo si la expresión es nulo.

Ejemplo: calcular el número de bytes que ocupa la columna APELLIDO de la tabla EMPLE.

### SELECT apellido, vsize(apellido) FROM emple;

* **USER**. Devuelve el nombre del usuario actual.
* **UID.** Al crear un usuario, Oracle le asigna un número que le identifica y es único en la base de datos.
* **NVL(valor, expresión).** Si el valor es nulo, devuelve expresión. Hay que tener en cuenta que la expresión y el valor deben ser del mismo tipo
* Visualizar para cada empleado de la tabla EMPLE, su apellido y su comisión teniendo en cuenta que si ésta fuese nula aparecería el texto “sin comisión”.