

**MP0484.**  
**Bases de datos**

**UF3. Formulación de consultas básicas**

**3.2. Operadores y  
Funciones**

# Índice

---

☰	Objetivos	3
☰	Operadores	4
☰	Operadores aritméticos o numéricos	5
☰	Operadores de comparación	6
☰	Operadores de caracteres	7
☰	Operadores lógicos	8
☰	Comprobaciones con conjunto de valores	9
☰	IS NULL	11
☰	Prioridades	12
☰	Las funciones	13
☰	Funciones aritméticas	14
☰	Funciones de cadenas de caracteres	16
☰	Funciones de fechas	17
☰	Funciones de conversión	18
☰	Ejercicio resuelto	19

# Objetivos

---

Dentro de las expresiones que podemos utilizar en SQL disponemos de **funciones y operadores** que pueden actuar sobre valores o columnas para producir nuevos resultados

## Objetivos:

- Conocer los principales operadores que se pueden utilizar en la base de datos
- Conocer las principales funciones con las que podemos trabajar

# Operadores

---

Los operadores permiten construir expresiones al **relacionar variables, constantes u otras expresiones.**

Cada operador efectúa una **operación dentro de un ámbito** como la lógica, la matemática o la concatenación.

- Operadores aritméticos
- Operadores fechas
- Operadores caracteres
- Operadores lógicos
- Operadores de comparación

# Operadores aritméticos o numéricos

Son los operadores válidos para números usuales en todos los lenguajes.

## Operadores Aritméticos

- + Suma
- - Resta
- \* Producto
- / División
- % Resto
- \*\* ó ^ Esponenciación
- = Igualdad

Ejemplo:

```
SELECT nombre, apellidos, (sueldo * 1.05) AS aumento FROM directores;
```

Esta sentencia devuelve una consulta con una **columna nueva calculada** llamada aumento con el valor del sueldo de los directores aumentado en un 5% .

```
SELECT nombre, articulo, (precio + precio * 0.21) AS precioiva  
FROM articulos;
```

Si tuviéramos una tabla con los precios de los artículos, esta sentencia devuelve una consulta con una **columna calculada llamada precioiva** con el precio aumentado por el 21 % de IVA.

# Operadores de comparación

Permiten **comparar** los elementos de una columna con una expresión o dos expresiones.

Al construir expresiones con estos operadores, los dos **operando**s deben ser del mismo tipo, ya sean booleanos, números, cadenas o fechas.

**Operadores de comparación:**

- = Permite seleccionar los registros que contenga un **valor concreto de una columna**. El operando es un valor.
- != Permite seleccionar todos los registros que sean **diferentes al valor especificado** en el operando.
- <> Sinónimo del anterior.
- > Permite seleccionar los registros con valores en la columna **mayores al especificado** en el operando.
- < Permite seleccionar los registros con valores en la columna **menores al especificado** en el operando.
- >= Permite seleccionar los registros con valores en la columna **mayores o iguales al especificado** en el operando.
- <= Permite seleccionar los registros con valores en la columna **menores o iguales al especificado** en el operando.

1

Ejemplo:

```
SELECT * FROM clientes WHERE nombre = 'Antonio';
```

Busca en la tabla clientes los que tengan como valor 'Antonio' en la columna nombre.

2

Ejemplo:

```
SELECT * FROM directores where sueldo > 1200;
```

Busca en la tabla de directores los que tengan sueldo mayor de 1200 Euros.

# Operadores de caracteres

Para comparar columnas de caracteres, hasta ahora hemos utilizado el operador de comparación =. Pero este operador no nos sirve si queremos hacer consultas de este tipo:

- Obtener los datos de los directores cuyo apellido empiece por una « P» u « obtener los nombres de los clientes que incluyan la palabra Pérez» .

Para ello disponemos del **operador LIKE** que permite utilizar los siguientes caracteres especiales en las cadenas de comparación:

%

Seleccionar cualquier cadena de 0  
o más caracteres y para  
seleccionar cualquier carácter.

\_

Comodín para seleccionar  
cualquier carácter. Un único  
cualquier carácter. Un único  
caracter.

## Ejemplos:

```
SELECT nombre from clientes where nombre LIKE 'MART%';
```

Selecciona todos los nombres de la tabla clientes que comienzan por MART

```
SELECT * FROM Clientes WHERE Nombre LIKE '%A%';
```

Selecciona los clientes que tengan un A en su nombre.

```
SELECT * FROM Clientes WHERE Nombre LIKE 'JUAN____';
```

Selecciona los clientes cuyo nombre empieza por JUAN y tiene 3 caracteres más

# Operadores lógicos

Con los operadores lógicos dos o más **condiciones** pueden ser combinadas para formar **expresiones más complejas** con distintos criterios. Cuando existen dos o más condiciones deberán estar unidas por AND u OR.

Operando **AND** operando

Permite seleccionar los registros que cumplan las expresiones de los **dos operandos**. Los operandos son expresiones de comparación.

Operando **OR** operando

Permite seleccionar los registros que cumplan con al menos **una de las expresiones** de los dos operandos. Los operandos son expresiones de **comparación**.

**NOT** operando

Permite seleccionar los registros que **no cumplen** con la **expresión** del operando.

Por ejemplo:

```
SELECT nombre FROM clientes WHERE (nombre LIKE 'P%') OR (nombre LIKE '_____');
```

Selecciona los clientes cuyo nombre comienza por P o tiene 6 caracteres

```
SELECT * FROM clientes WHERE (fecha_de_alta >'01/01/2017') AND (nsucursal =1001);
```

Selecciona los clientes que tengan fecha de alta mayor de '01/01/2017' y sean de la sucursal 1001.

# Comprobaciones con conjunto de valores

En la sentencia **WHERE** se pueden incluir **operadores** que permiten comparar una columna o una expresión con una lista de valores

1

**Between.**

Permite consultar los registros con los valores de la columna que se encuentra entre dos límites.

**Sintaxis:**

```
[...] WHERE columna BETWEEN limite1 AND limite2.
```

**Ejemplo:**

```
SELECT * FROM directores WHERE sueldo BETWEEN 1300 AND 1500;
```

Selecciona los directores cuyo sueldo esté entre 1300 y 1500 Euros

2

**IN.**

Permite filtrar los registros con un valor de la columna que coincide con los valores de una lista.

**Sintaxis:**

```
[...] WHERE columna IN ({valor1[, ...],valorN})
```

**Ejemplo:**

```
SELECT * FROM clientes WHERE nsucursal IN (1001,1002);
```

Busca los clientes que sean de la sucursal 1001 ó 1002

# IS NULL

Se dice que una **columna de una fila es NULL si está completamente vacía.**

Para comprobar si el valor de una columna es nulo empleamos la expresión:

```
columna IS NULL
```

Si queremos saber si el valor de una columna no es nulo, usamos la expresión:

```
columna IS NOT NULL
```

Cuando comparamos con valores nulos o no nulos no podemos utilizar los operadores de igualdad, mayor o menor.

**Ejemplo:**

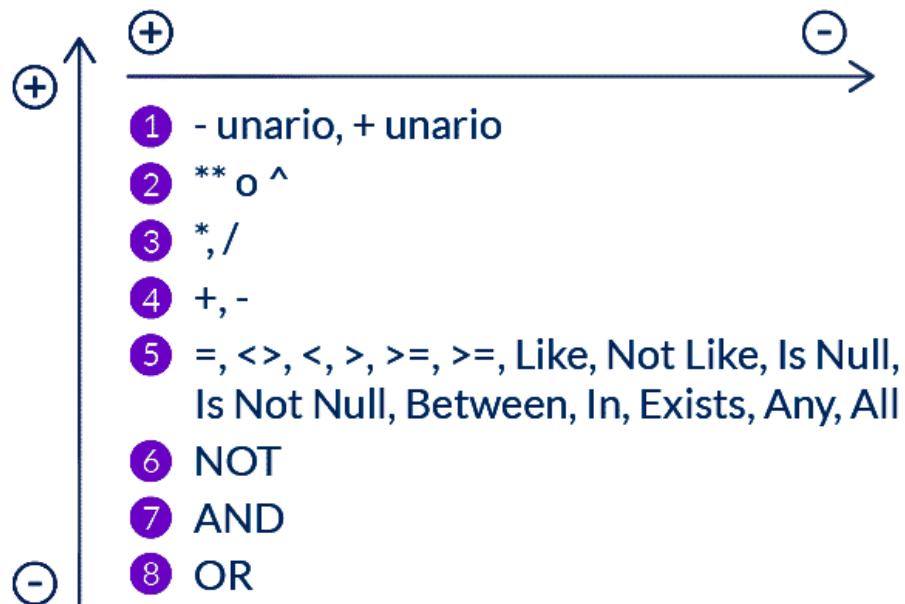
Selecciona a los clientes que no tengan fecha\_de\_alta;

```
SELECT * FROM Clientes WHERE fecha_de_alta IS NULL;
```

# Prioridades

Cuando trabajamos con operadores y **no usamos paréntesis** es importante conocer su prioridad de aplicación, ya que el resultado puede cambiar si no tenemos en cuenta dicha prioridad.

Las prioridades de los operadores **decrecen** de izquierda a derecha y de arriba a abajo. En este listado se muestran las prioridades:



# Las funciones

**Las funciones se usan dentro de expresiones y actúan con los valores de las columnas, variables o constantes.**

Generalmente producen dos tipos diferentes de resultados:

- Unas que producen una modificación de la información original (por ejemplo, poner en minúscula una cadena que está en mayúscula).
- El resultado de otras indica alguna cosa sobre la información (por ejemplo, el número de caracteres que tiene una cadena).

Se pueden utilizar en: cláusulas SELECT, cláusulas WHERE y cláusulas ORDER BY. Es posible el anidamiento de funciones. Existen cinco tipos de funciones:

- 1 Aritméticas
- 2 Cadena de caracteres
- 3 Manejo de Fechas
- 4 Conversión
- 5 Otras funciones



Las funciones son muy dependientes del Gestor de Base de datos con el que estemos trabajando, por lo que es importante recurrir a la documentación asociada para conocer qué tipo de funciones están permitidas. En este apartado se van a listar las funciones principales utilizadas en ORACLE.

# Funciones aritméticas

Las funciones aritméticas trabajan con datos de tipo numérico NUMBER.

Este tipo incluye los dígitos de 0 a 9, el punto decimal y el signo menos, si es necesario. Los literales numéricos no se encierran entre comillas. **Ejemplo:** -123.32.

Estas funciones trabajan con dos clases de números:

- Valores simples.
- Grupos de valores.

**Algunas modifican los valores sobre los que actúan, otras informan de algo sobre los valores.**

Podemos dividir las funciones aritméticas en dos grupos:

- Funciones de valores simples.
- Funciones de grupos de valores. ( Las veremos en la siguiente unidad )

## Funciones de valores simples

Las funciones de valores simples son funciones sencillas que trabajan con valores simples. Un valor simple es: un número (como 6522,90), una variable o una columna de una tabla.

Para probar algunas de estas funciones puedes usar la tabla DUAL. Es una tabla especial de una sola columna presente de manera predeterminada. La tabla tiene una sola columna VARCHAR2(1) llamada DUMMY.

Ejemplo:

```
SELECT 1+1 FROM DUAL;
```

Las funciones de valores simples se muestran en la siguiente tabla:

Función	Propósito
<b>ABS (n)</b>	Nos devuelve el valor absoluto de n
<b>CEIL (n)</b>	Nos devuelve el valor entero igual o inmediatamente superior a n
<b>FLOOR (n)</b>	Nos devuelve el valor entero igual o inmediatamente inferior a n
<b>MOD (m,n)</b>	Nos devuelve el resto de la división de m entre n
<b>POWER (m, exponente)</b>	Calcula la potencia de m elevado a exponente
<b>SIGN (valor)</b>	Nos devuelve el signo de valor
<b>NVL (valor, expresión)</b>	Función que nos sustituye valor por expresión siempre que valor sea NULL
<b>ROUND (número [,m])</b>	Nos redondea numero a m decimales
<b>SQRT (n)</b>	Nos devuelve la raíz cuadrada de n
<b>TRUNC (número [,m])</b>	Trunca los números para que tengan m decimales.

Por ejemplo para obtener el valor absoluto del SALARIO - 1500 de todas los directores.

```
SELECT nombre,apellidos,ABS(sueldo-1800) FROM directores;
```

# Funciones de cadenas de caracteres

Las **funciones de cadenas de caracteres** trabajan con datos de tipo CHAR o VARCHAR2. Estos datos incluyen cualquier carácter alfanumérico: letras, números y caracteres especiales. Los literales se deben encerrar entre comillas simples.

Ejemplo de una cadena de caracteres: 'Instituto'.

Las funciones de cadenas permiten manipular cadenas de letras u otros caracteres. Estas funciones pueden calcular el número de caracteres de una cadena, convertir una cadena a mayúsculas o a minúsculas, suprimir o añadir caracteres a la izquierda o a la derecha, etcétera.

Función	Propósito
<b>CHR (n)</b>	Nos devuelve el carácter cuyo valor en binario es n
<b>CONCAT (cad1, cad2)</b>	Nos devuelve cad1 concatenada con cad2
<b>UPPER (cad)</b>	Convierte cad a mayúsculas
<b>LOWER (cad)</b>	Convierte cad a minúsculas
<b>LPAD (cad1,n[,cad2])</b>	Con esta función añadimos caracteres a cad1 por la izquierda hasta una longitud máxima dada por n
<b>INITCAP (cad)</b>	Convierte la primera letra de cad a mayúscula
<b>LTRIM (cad [,set])</b>	Elimina un conjunto de caracteres a la izquierda de cad, siendo set el conjunto de caracteres a eliminar
<b>RPAD (cad1, n[,cad2])</b>	Con esta función añadimos caracteres de la misma forma que con la función LPAD pero esta vez los añadimos a la derecha
<b>RTRIM (cad[,set])</b>	Hace lo mismo que LTRIM pero por la derecha
<b>REPLACE (cad,cadena_buscada [,cadena_sustitucion])</b>	Sustituye un conjunto de caracteres de 0 o más caracteres, devuelve cad con cada ocurrencia de cadena_buscada sustituida por cadena_sustitucion
<b>SUBSTR (cad, m[,n])</b>	Devuelve la subcadena de cad que abarca desde m hasta el numero de caracteres dados por n.
<b>TRANSLATE (cad1,cad2,cad3)</b>	Convierte caracteres de una cadena en caracteres diferentes. Devuelve cad1 con los caracteres encontrados en cad2 y sustituidos por los caracteres de cad3

Ejemplo:

En este ejemplo se muestra una columna nombre con el texto 'El nombre del cliente es .... ' y cada uno de los nombres de los clientes que tenemos almacenado en la tabla clientes.

```
SELECT CONCAT ('El nombre del cliente es ', nombre) as nombre from clientes;
```

# Funciones de fechas

En SQL disponemos del tipo de dato Fecha (DATE) y posee una interesante utilidad para formatear las fechas de cualquier manera que necesitemos.

Tiene un formato por omisión: 'DD/MM/YY', pero con la función TO\_CHAR es posible mostrar las fechas de cualquier modo. Los literales de fecha deben encerrarse entre comillas simples.

- Ejemplo: '18/11/05'.

El tipo de datos DATE se suele almacenar en la mayoría de los SGBD en un formato especial que incluye Siglo/Año/Mes/Día/Hora/Minutos/Segundos. Las funciones para el manejo de fechas se exponen en la siguiente tabla:

Función	Propósito
<b>SYSDATE</b>	Devuelve la fecha del sistema
<b>ADD_MONTHS (fecha, n)</b>	Devuelve la fecha incrementada en n meses
<b>LAST_DAY (fecha)</b>	Devuelve la fecha del último día del mes que contiene fecha
<b>MONTHS_BETWEEN (fecha1, fecha2)</b>	Devuelve la diferencia en meses entre la fecha1 y la fecha2
<b>NEXT_DAY (fecha, cad)</b>	Devuelve la fecha del primer día de la semana indicado por cad después de la fecha indicada por fecha.

Ejemplo:

Vamos a sumar 2 meses a la fecha de alta de los clientes

```
SELECT fecha_de_alta, add_months(fecha_de_alta,2) FROM clientes;
```

# Funciones de conversión

La mayoría de las funciones que hemos visto hasta ahora son funciones de transformación, esto es, cambian los objetos.

Hay otras funciones que cambian los objetos de una manera especial, pues transforman un tipo de dato en otro. Las funciones de conversión elementales se muestran en la siguiente tabla:

Función	Propósito
<b>TO_CHAR (fecha,'formato')</b>	Esta función nos convierte una fecha de tipo date a una fecha de tipo varchar2
<b>TO_CHAR (numero,'formato')</b>	Nos convierte un dato de tipo number a un tipo varchar2
<b>TO_DATE (cad,'formato')</b>	Convierte un tipo de dato varchar2 o char a un valor de tipo date con el formato especificado
<b>TO_NUMBER (cadena,['formato'])</b>	Convierte una cadena a tipo de dato number, y si se le pasa, con el formato especificado

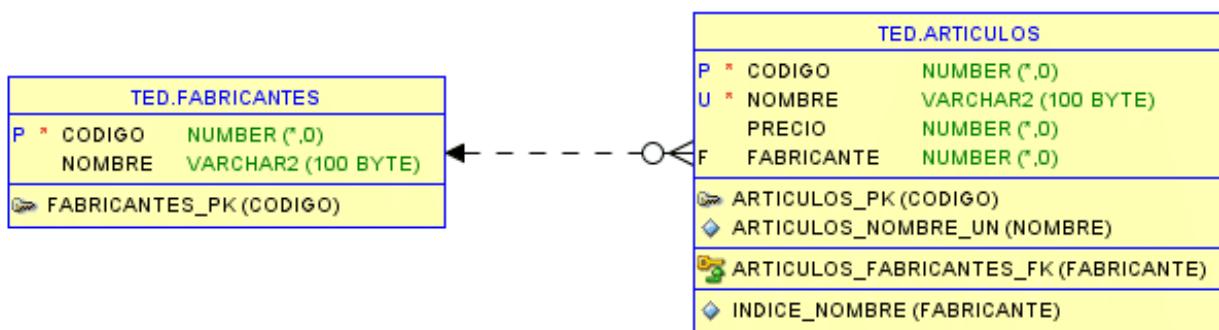
Ejemplo:

Mostrar el año en el que se han dado de alta los clientes.

```
SELECT TO_CHAR(fecha_de_alta,'yyyy') FROM clientes;
```

# Ejercicio resuelto

Se pretende llevar el control de los artículos de una tienda almacenando dichos artículos y sus fabricantes en una base de datos. Para ello se pide a partir del siguiente modelo relacional las siguientes tareas:



1

Crear las tablas con estas características:

- La tabla fabricantes tiene como PRIMARY KEY su código.
- La tabla artículos tiene como PRIMARY KEY su código.
- La FOREIGN KEY de la tabla artículos es el campo fabricante que está relacionado con el código de la tabla fabricantes.
- Cuando se elimine un fabricante, se debe de colocar en el artículo que tenga ese fabricante el valor de NULL.
- No permitir ni nulos ni repetidos en el nombre del artículo
- Crear un índice en el código del fabricante de la tabla artículos
- Poner por defecto 100 al campo precio de la tabla artículos
- No permitir meter precios menores de 50.

2

Insertar valores tanto en la tabla de Artículos como en la de Fabricantes. ¿Cuál de esas tablas habrá que rellenar primero para que no tengamos problemas con la integridad referencial?

3

Realizar las siguientes consultas:

- Obtener los nombres de los productos de la tienda
- Obtener los nombres y los precios de los productos de la tienda
- Obtener el nombre de los productos cuyo precio sea mayor a 200€
- Obtener todos los datos de los artículos cuyo precio esté entre los 60€ y los 120€
- Obtener el nombre y el precio en dólares, teniendo en cuenta que 1 EUR = 1,19489 dólares
- Obtener el nombre y precio de los artículos cuyo precio sea mayor o igual a 180€ y ordenarlos descendente por precio, y luego ascendentemente por nombre
- Cambiar el nombre del artículo 1002 a Acer Aspire Switch 5
- Aplicar un descuento del 10% (multiplicar el precio por 0,9) a todos los productos
- Aplicar un descuento de 10€ a todos los productos cuyo precio sea mayor o igual a 120€

## Creación de las tablas

```
-- TABLA FABRICANTES
CREATE TABLE fabricantes
(
    codigo INTEGER PRIMARY KEY,
    nombre VARCHAR2(100)
);

-- TABLA ARTÍCULOS
CREATE TABLE articulos
(
    codigo INTEGER PRIMARY KEY,
    nombre VARCHAR2(100) UNIQUE NOT NULL,
    precio INTEGER DEFAULT 100,
    fabricante INTEGER,
    CHECK (precio >=50),
    CONSTRAINT arti_fabric_fk FOREIGN KEY (fabricante)
        REFERENCES fabricantes(codigo)
        ON DELETE SET NULL
);
CREATE INDEX indice_nombre ON articulos(fabricante);
```

## Insertar valores

```
/* Primero hay que introducir los fabricantes para que
los artículos se puedan asociar */

--- DATOS FABRICANTES ---
INSERT INTO fabricantes (codigo, nombre) VALUES (1, 'Acer');
INSERT INTO fabricantes (codigo, nombre) VALUES (2, 'Brother');
INSERT INTO fabricantes (codigo, nombre) VALUES (3, 'Dell');
```

```
INSERT INTO fabricantes ( codigo, nombre ) VALUES (4, 'Epson');
```

```
INSERT INTO fabricantes ( codigo, nombre ) VALUES (5, 'Toshiba');
```

-- DATOS DE ARTICULOS

```
INSERT INTO ARTICULOS(codigo,nombre,precio,fabricante)  
values (1001,'Portatil Acer Aspire Intel Celeron',243,1);
```

```
INSERT INTO ARTICULOS(codigo,nombre,precio,fabricante)  
values (1002,'Acer Aspire Switch 3',499,1);
```

```
INSERT INTO ARTICULOS(codigo,nombre,precio,fabricante)  
values (1003,'Impresora WF-2750DWF Multifunción',70,4);
```

```
INSERT INTO ARTICULOS(codigo,nombre,precio,fabricante)  
values (1004,'Altavoces Skydusk',52,3);
```

```
INSERT INTO ARTICULOS(codigo,nombre,precio,fabricante)  
values (1005,'Teclado Trust Elight',59,5);
```

### Obtener los nombres de los productos de la tienda

```
SELECT Nombre FROM articulos;
```

### Obtener los nombres y los precios de los productos de la tienda

```
SELECT Nombre, Precio FROM articulos;
```

### Obtener el nombre de los productos cuyo precio sea mayor a 200€

```
SELECT Nombre FROM ARTICULOS WHERE Precio > 200;
```

### Obtener todos los datos de los artículos cuyo precio esté entre los 60€ y los 120€

-- Con AND

```
SELECT * FROM ARTICULOS  
WHERE Precio >= 60 AND Precio <= 120;
```

-- Con BETWEEN

```
SELECT * FROM ARTICULOS  
WHERE Precio BETWEEN 60 AND 120;
```

Obtener el nombre y el precio en dólares, teniendo en cuenta que 1 EUR = 1,19489 dólares

– Sin AS

```
SELECT Nombre, Precio * 1.19489 FROM ARTICULOS;
```

– Con AS

```
SELECT Nombre, Precio * 1.19489 AS PrecioDolares FROM ARTICULOS;
```

Obtener el nombre y precio de los artículos cuyo precio sea mayor o igual a 180€ y ordenarlos descendente por precio, y luego ascendentemente por nombre

```
SELECT Nombre, Precio FROM ARTICULOS WHERE Precio >= 180  
ORDER BY Precio DESC, Nombre;
```

Cambiar el nombre del artículo 1002 a Acer Aspire Switch 5

```
UPDATE ARTICULOS SET Nombre = 'Acer Aspire Switch 5'  
WHERE Código = 1002;
```

Aplicar un descuento del 10% a todos los productos

```
UPDATE ARTICULOS  
SET Precio = Precio * 0.9;
```

Aplicar un descuento de 10€ a todos los productos cuyo precio sea mayor o igual a 120€

```
UPDATE ARTICULOS SET Precio = Precio - 10 WHERE Precio >= 120;
```



**PROEDUCA**