3. SQL/DQL. Consultas simples

El DQL (Data Query Language), lenguaje de consultas de datos, agrupa las instrucciones `SELECT` en diferentes niveles. En este apartado vamos a comenzar por las más simples. Serán consultas que eligen qué campos de la tabla vamos a mostrar, aunque podemos abreviar si los queremos todos utilizando el comodín del asterisco *.

Fig. 6.3. Orden en que se han de escribir las cláusulas dentro de la orden `SELECT`, que hace la función de verbo Seleccionar.

A. SELECT

La instrucción `SELECT` nos permite seleccionar los campos que queremos ver de la tabla que indiquemos en la cláusula `FROM`. Si solo ponemos * entre estas, seleccionará todos los campos. Y si además queremos filtrar el resultado por algún valor de algún campo concreto lo podremos hacer a través de la cláusula `WHERE`; esta última no es obligatoria ponerla, en cuyo caso se mostrarán todos los valores de la tabla seleccionada. Vamos a ver toda su sintaxis.

Descripción

La instrucción **`SELECT`** se utiliza para **seleccionar (consultar)** datos de la/s tabla/s de nuestra base de datos. Obligatoriamente tenemos que poner las sentencias `SELECT` y `FROM`. El resto de sentencias son opcionales, todas las que aparecen entre [ ].

Sintaxis básica

```
SELECT [ {ALL | DISTINCT | DISTINCTROW }  ]
    <nombre_campo> [ , <nombre_campo> [ , ... ] ]
FROM <nombre_tabla>  [ , {<nombre_tabla>} [ , ...] ]
[WHERE <condición> [ {AND | OR} <condición> [ , ...] ]
[ORDER BY <condición> [ {AND | OR} <condición> [ , ...] ] [DESC]
[LIMIT [<posición_inicial>], <número_registros>]
```

Ejemplos

```
SELECT * FROM Director ; -- Nos mostrará todos los campos de la
tabla director sin filtros
SELECT nombreDirector, edadDirector
FROM Director
WHERE salarioDirector > 10000 AND nacionalidadDirector = 1; -- Solo
mostrará algunos
```

Lista el nombre y la edad de todos los directores de cine que cumplen la condición, es decir, cuyo salario es superior a 10.000 €, o $, dependiendo de la moneda definida en el SGBDR, y además sea de la nacionalidad 1, que es la de Estados Unidos en la tabla PAÍS.

Tabla 6.15. Sintaxis de la sentencia `SELECT`.

Hay que comentar que la opción opcional `LIMIT` <posición_inicial>], <número_registros>, es muy interesante para hacer aplicaciones de escritorio o web. Nos permite ir recuperando los registros de la consulta en bloques con el número de registros indicados. Si se omite la posición inicial empezará por el registro inicial o cero, pero en las siguientes muestras de datos ya no podremos omitir el valor de la posición inicial. De este modo, podemos realizar consultas con muchos registros (cientos o miles o millones) e ir recuperando a intervalos (del tamaño que nosotros queramos) toda la información.

B. Tipos de operadores de SQL

En las sentencias SQL vamos a poder crear expresiones complejas que iremos creando, y para ello necesitamos ciertos operadores que nos den la potencia requerida para evaluarlas adecuadamente. Vamos a ver los diferentes tipos y los símbolos o palabras reservadas que vamos a poder utilizar.

B1. Operadores aritméticos de SQL

Estos operadores aritméticos solo trabajan con números enteros. Supongamos que vamos a utilizar la variable operador1, que tiene 23, y la variable operador2, que tiene 11; entonces, para cada operador obtendremos los siguientes resultados:

Operador	Descripción	Ejemplo
Sumar + *(add)*	Suma los valores a ambos lados del operador.	**operador1** + operador2 = 34
Restar – *(subtract)*	Resta el valor de la derecha del valor de la izquierda.	**operador1** - operador2 = 12
Multiplicar * *(multiply)*	Multiplicar los valores a ambos lados del operador.	**operador1** * operador2 = 253
Dividir / *(divide)*	Dividir el operando de la izquierda por el operando de la derecha.	**operador1** / operador2 = 2
% *(módulo)*	Dividir el operando de la izquierda por el operando de la derecha, devolviendo el resto.	**operador1** % operador2 = 1

Tabla 6.16. Operadores aritméticos de SQL.

B2. Operadores comparación o búsqueda de SQL

Cuando necesitemos comparar los operadores para saber si el resultado es verdadero o falso para hacer otras cosas vamos a seguir utilizando op1=23 y op2=11.

Operador	Descripción	Ejemplo
Menor < *(minor)*	Comprueba si el valor del operando izquierdo es menor que el valor del operando derecho.	(op1 < op2) es falso
Menor o igual <= *(less than or equal to)*	Comprueba si el valor del operando izquierdo es menor o igual que el valor del operando derecho.	(op1 <= op2) es falso
Mayor > *(minor)*	Comprueba si el valor del operando izquierdo es mayor que el valor del operando derecho; si es así, la condición se convierte en verdadera.	(op1 > op2) es cierto
Mayor o igual >= *(greater than or equal to)*	Comprueba si el valor del operando izquierdo es mayor o igual que el valor del operando derecho.	(op1 >= op2) es cierto
igual = *(equal to)*	Comprueba si los valores de ambos operadores son iguales o no.	(op1 = op2) es falso
Distinto <> *(distinct)*	Comprueba si los valores de ambos operadores no son iguales.	(op1 <> op2) es cierto

Tabla 6.17. Operadores comparativos de SQL.

Comentar que también existen estos operadores combinando el de negación: !=, !< y !>.

Los siguientes operadores, no tan conocidos, también son de comparación.

Operador	Descripción
`BETWEEN`	Sirve para buscar valores que están dentro de un rango de valores.
`EXISTS`	Sirve para buscar la existencia de una fila en tablas que cumpla la condición.
`UNIQUE`	Sirve para buscar si no existen duplicados en cada fila de una tabla para ver si cumple la restricción de unicidad.
`ALL`	Sirve para comparar un valor con todos los valores de otro conjunto.
`ANY`	Sirve para comparar un valor con cualquier otro según la condición.
`IN`	Sirve para comparar un valor con una lista de valores dados.
`IS NULL`	Sirve para comparar un valor con un valor `NULL`.
`LIKE`	Sirve para comparar un valor con valores similares con operadores comodín.

Tabla 6.18. Operadores literales de búsqueda o comparación de SQL.

B3. Operadores comodín de SQL

Cuando queremos buscar datos filtrándolos por la cláusula `WHERE` dentro de un rango de valores alfanuméricos podemos utilizar estos operadores comodín % o _.

Operador	Descripción
Símbolo del porcentaje % *(percent symbol)*	Puede coincidir con uno, ninguno o varios caracteres.
Símbolo del guion bajo _ *(underscore)*	Solo puede coincidir con un carácter.

Tabla 6.19. Operadores comodín de SQL.

Hay que comentar que el SGBDR Microsoft Access utiliza el * en lugar del % y el ? en lugar del _, como sucede también en consolas o terminales CLI como el CMD de MS-DOS o el Shell bash de GNU/Linux. Hay que recordar que en SQL el * sirve para seleccionar todos los campos de la/s tabla/s sobre las que vayamos a hacer la consulta.

Algunos ejemplos buscando el valor VALENCIA o derivados utilizando los comodines:

```
SELECT * FROM Hotel WHERE provinciaHotel LIKE 'VAL%';
SELECT * FROM Hotel WHERE provinciaHotel LIKE '%ALENC%';
SELECT * FROM Hotel WHERE provinciaHotel LIKE 'VALENCI_';
SELECT * FROM Hotel WHERE provinciaHotel LIKE '_ALENCIA';
SELECT * FROM Hotel WHERE provinciaHotel LIKE '_ALENCI_';
```

B4. Operadores lógicos de SQL

En SQL disponemos de algunos operadores lógicos más aparte de los tradicionales `AND`, `OR` y `NOT`. En la siguiente tabla os explicamos para qué sirven.

Operador	Descripción
AND	Sirve para poder poner múltiples condiciones en la cláusula `WHERE`. Da como resultado true (verdadero) si TODAS las condiciones son verdaderas
NOT	Sirve para invertir el valor del operador lógico con el que se aplique.
OR	Sirve para poder poner múltiples condiciones en la cláusula `WHERE`. Da como resultado true (verdadero) si CUALQUIERA de las condiciones es verdadera.

Tabla 6.20. Operadores lógicos de SQL.

B5. Operadores de fechas de SQL

En SQL es interesante utilizar funciones específicas de fechas para realizar operaciones con ellas. Vamos a ver algunas de ellas.

Operador lógico	Descripción
NOW()	Sirve para la fecha de hoy en formato '2021-07-31 12:21'.
DATEDIFF()	Sirve para poder averiguar la diferencia entre dos fechas.
DATE_FORMAT()	Sirve para formatear la fecha como nos pueda interesar.

Tabla 6.21. Operadores de fechas de SQL.

Como siempre, cada fabricante de SGBDR, en su implementación ampliada de su SQL no estándar, dispone de sus propias funciones, en este caso de fechas; para verlas todas consultar esta web.