Tema 4
Introducción a SQL.
MariaDB/MySQL.

Índice

1. Introducción

EL lenguaje *SQL* (*Structured Query Language*) es un lenguaje que permite realizar operaciones, mediante sentencias, sobre una base de datos relacional. Está compuesto por el lenguaje de definición de datos (*DDL*), el lenguaje de manipulación de datos (*DML*) y el lenguaje de control de datos (*DCL*).

2. Sentencias

Todas las sentencias *MariaDB/MySQL* comienzan con una palabra clave que indica la acción a ejecutar sobre la base de datos.

Entre otras podemos encontrar las siguientes sentencias:

SENTENCIA	LENGUAJE	ACCIÓN
Select	DML	Consultar datos de la base de datos
Insert	DML	Insertar datos en la base de datos
Update	DML	Actualizar datos de la base de datos
Delete	DML	Borrar datos de la base de datos
Create	DDL	Crear estructuras en la base de datos
Drop	DDL	Borrar estructuras en la base de datos
Alter	DDL	Modificar estructuras en la base de datos
Grant	DCL	Añadir privilegios a los usuarios de la base de datos
Revoke	DCL	Quitar privilegios a los usuarios de la base de datos

Además, en las sentencias, se pueden encontrar también los siguientes elementos: palabras reservadas, operadores, funciones, nombres de tablas, datos, constantes, ... Todas las sentencias terminan con un ;

En *MariaDB/MySQL* cada base de datos se almacena en un directorio con su nombre dentro del directorio *'data'*. Dentro de este directorio se crea por lo menos un fichero por cada tabla, depende del motor de almacenamiento (que ya veremos, más adelante, lo que es).

Por lo que:

- Dependiendo del sistema operativo los nombres de las bases de datos pueden diferenciar mayúsculas de minúsculas (en *Linux*) y no diferenciar (en *Windows*).
- Los demás elementos de la base de datos no diferencian mayúsculas de minúsculas.

3. Consulta de datos

Las consultas permiten obtener datos de una base de datos mediante la sentencia *SELECT*[1] [2].

Usaremos, para la realización de los ejemplos del tema, la tabla de los mejores jugadores de la liga regular de los *Ángeles Lakers*. Sus atributos son:

CAMPO	DESCRIPCIÓN
Nombre	Nombre del jugador
Partidos	Total de partidos con los *Lakers*
Minutos	Media de minutos jugados por partido
Puntos	Media de puntos logrados por partido
Rebotes	Media de rebotes conseguidos por partido
Tapones	Media de tapones realizados por partido
Asistencias	Media de asistencias por partidos
Valoracion	Valoración del jugador de 1 a 5.

[1] https://mariadb.com/kb/en/select/
[2] https://dev.mysql.com/doc/refman/8.0/en/select.html

La estructura básica de la sentencia *SELECT* es:

```
SELECT [ALL|DISTINCT] {expreColumna1 [, expreColumna2, ..., expreColumnaN ] | * }
FROM {nombreTabla1 [, nombreTabla2, ..., nombreTablaN]}
[WHERE condiciones][
[ORDER BY {columna | expr | posición} [DESC | ASC] [,{columna | expr | posición} [DESC | ASC]]...];
[LIMIT [desplazamiento,] contador]
```

Donde los siguientes elementos, que no deben aparecer en la consulta, significan:

- Todo elemento entre corchetes [] no es obligatorio para la consulta, es opcional.
- En todo elemento entre llaves {} se debe escoger, por lo menos, una de las opciones presentes.
- La barra vertical | hace de separador entre opciones.
- El orden de los elementos es fijo y no se puede alterar.

3.1. Columnas[3]

Una consulta devuelve los datos como una tabla cuyas columnas vienen especificadas por las expresiones *expreColumna* separándose por comas. La lista de columnas se colocan después de Select y, en caso de aparecer, antes del FROM.

Estas expresiones pueden ser:

- Una constante: 4, "Hola", ...
- Una columna de la tabla de consulta: Partidos, minutos, ...
- El resultado de aplicar una o más funciones a un columna/constante: lenght(nombre), ...
- Una expresión aritmética: 5+2, puntos*1.5, ...
- ...

Este es un ejemplo de una consulta donde se obtienen los nombres y la valoración de los jugadores presentes en la tabla jugadores.

```
SELECT nombre, valoracion FROM jugadores;
```

Para obtener todos los datos de la tabla existen dos opciones:

- Introducir los nombres de todas las columnas:

```
SELECT nombre, partidos, minutos, puntos, rebotes, tapones, asistencias, valoracion FROM jugadores;
```

- Usar la expresión * que incluye todas las columnas:

```
SELECT * FROM jugadores;
```

Según los datos de la tabla el resultado de la consulta es:

```
+--------------------+----------+---------+--------+---------+---------+-------------+------------+
| NOMBRE             | Partidos | Minutos | Puntos | Rebotes | Tapones | Asistencias | Valoracion |
+--------------------+----------+---------+--------+---------+---------+-------------+------------+
| Kareem Abdul-Jabbar |    01093 |    34.3 |   22.1 |     9.4 |     2.5 |         3.3 |          5 |
| Magic Johnson      |    00906 |    36.7 |   19.5 |     7.2 |     0.4 |        11.2 |          5 |
| Jerry West         |    00932 |    39.2 |   27.0 |     5.8 |     0.7 |         6.7 |          5 |
| Elgin Baylor       |    00846 |    40.0 |   27.4 |    13.5 |    NULL |         4.3 |          2 |
| Kobe Bryant        |    01103 |    36.4 |   25.3 |     5.3 |     0.5 |         4.7 |          4 |
| James Worthy       |    00926 |    32.4 |   17.6 |     5.1 |     0.7 |         3.0 |          4 |
| Shaquille O'Neal   |    00514 |    37.6 |   27.0 |    11.8 |     2.5 |         3.1 |          4 |
| Byron Scott        |    00846 |    30.2 |   15.1 |     3.0 |     0.3 |         2.8 |          3 |
| Gail Goodrich      |    00687 |    31.7 |   19.0 |     3.0 |     0.2 |         4.2 |          3 |
| Vern Mikkelsen     |    00699 |    32.5 |   14.4 |     9.4 |    NULL |         2.2 |          2 |
+--------------------+----------+---------+--------+---------+---------+-------------+------------+
```

[3] https://mariadb.com/kb/en/identifier-names/

En un *SELECT* existen las clausulas optativas *ALL* y *DISTINCT*:

- *ALL*: recupera todas las filas incluyendo filas repetidas. Esta es la opción por defecto si no se especifican estas clausulas.
- *DISTINCT*: se recuperan todas filas eliminado las repetidas (se unifican todas las filas repetidas en una única fila).

EJEMPLO

```
SELECT ALL valoracion FROM jugadores;
```

VALORACION
5
5
5
2
4
4
4
3
3
2

```
SELECT DISTINCT valoracion FROM jugadores;
```

VALORACION
5
2
4
3

Cuando se visualiza el resultado de una consulta se usan los nombres de las columnas de la tabla de consulta como nombres de las columnas del resultado. Mediante la clausula opcional *AS* se puede crear un alias[4] para visualizar un nombre de columna diferente.

Por ejemplo:

```
SELECT nombre, valoracion AS "Valoración del jugador"  FROM jugadores;
```

Aunque *AS* es opcional se recomienda no omitirlo para ganar en claridad en la sentencia.

3.2. Tablas de datos

Si que quiere consultar los datos contenidos en una o varias tablas de deben indicar, de forma obligatoria, después de la clausula *FROM* separándolas por comas.

Por ejemplo:

```
SELECT nombre, puntos FROM jugadores;
```

La clausula *FROM* solo es obligatoria si los datos a consultar están en tablas, en otro caso se puede omitir:

```
SELECT 'Constante cadena', 5, CURRENT_DATE(), DAYNAME(CURRENT_DATE());
```

Al igual que en el caso de las columnas, mediante la clausula opcional *AS* se puede crear alias a los nombre de las tablas. Aunque su uso se verá en profundidad en temas posteriores mediante el uso de alias se pueden resolver problemas de ambigüedad cuando se realizan consultas combinado dos veces la misma tabla.

```
SELECT  xogadores.nombre, xogadores.puntos , gamers.nombre, gamers.puntos
FROM jugadores AS xogadores, jugadores AS gamers
WHERE xogadores.puntos > gamers.puntos;
```

[4] https://www.techonthenet.com/mariadb/alias.php

3.3. Selección de filas

Por defecto cuando se realiza una consulta sobre una tabla se devuelven todas las filas de la tabla.

La clausula *WHERE*[5] permite filtrar las filas que se obtienen de la consulta quedándonos con aquellas que cumplen un criterio especificado. Las filas que cumplen con una o más condiciones asociadas a la clausula *WHERE* se añaden al resultado las demás no. La condición puede tener cualquier nivel de complejidad.

Por ejemplo:

- Seleccionar el nombre y la valoración de los jugadores con una valoración de 5.

```
SELECT nombre, valoracion FROM jugadores WHERE valoracion=5;
```

- Consultar todos los datos del jugador *Magic Johnson*.

```
SELECT * FROM jugadores WHERE nombre = 'Magic Johnson';
```

- Obtener los jugadores con que han jugado 5 partidos y tienen un 3 de valoración

```
SELECT nombre, partidos FROM jugadores WHERE partidos=5 AND valoracion=3
```

  Cuando en todas las condiciones del *WHERE* se utiliza el mismo operador (en este caso =) se puede utilizar la sintaxis alternativa:

```
SELECT nombre, partidos FROM jugadores WHERE (partidos, valoracion)=(5,3)
```

3.4. Ordenación de los resultados

Cuando se obtienen resultados de una tabla estos se muestran en el orden en que fueron insertados en ella. Mediante la clausula *ORDER BY*[6] se puede especificar un criterio de ordenación diferente.

La sintaxis de *ORDER BY* es:

```
ORDER BY {columna | posición | expr } [ASC | DESC]
```

Donde:

- *columna* | *expr* | *posición*: representa el campo por el que se va a ordenar, puede ser:
  - Una columna de la tabla consultada.
  - Una columna del resultado expresado por su posición después del *SELECT*.
  - El resultado de una expresión.
- *ASC* | *DESC*: indica si se usa un criterio de ordenación ascendente o descendente. Si no se indica el resultado se ordena de forma ascendente.

La siguiente consulta muestra los jugadores ordenados por los puntos marcados de forma descendente:

```
SELECT * FROM jugadores ORDER BY puntos DESC;
```

Es posible concatenar más de un criterio de ordenación. La siguiente consulta ordena los jugadores por valoración de forma ascendente, ordenación por defecto, y luego, dentro de esta ordenación, se ordena por nombre de forma descendente.

```
SELECT * FROM jugadores ORDER BY valoración ASC, nombre DESC;
```

Un ejemplo de la utilización de la posición de una columna y de una expresión es la siguiente:

```
SELECT nombre, valoracion FROM jugadores ORDER BY 2, LENGTH(nombre) DESC;
```

Donde de ordena por la segunda columna de resultado, en nuestro caso valoración, y dentro de esa ordenación ordenar por la longitud del nombre del jugador de forma descendente.

[5] https://www.techonthenet.com/mariadb/where.php
[6] https://mariadb.com/kb/en/order-by/

3.5. Limitación del número de resultados

MariaDB/MySQL dispone de la clausula *LIMIT*[7] para limitar el número máximo de resultados que una consulta puede devolver.

Por ejemplo, si queremos listar los tres primeros jugadores usaríamos:

```
SELECT * FROM jugadores LIMIT 3;
```

Pero además de comenzar desde el principio se le puede indicar el numero de fila, la primera fila es la fila cero, desde la que mostrar los resultados.

```
SELECT * FROM jugadores LIMIT 1, 3;
```

Donde:

- El primer valor (1 en el ejemplo) indica el número de fila desde la que mostrar los resultados. En nuestro ejemplo se muestran los resultados a partir de la fila 2 de la tabla (recordar que la fila 1 es la posición 0).
- El segundo valor (3 en el ejemplo): indica el número de filas a mostrar. En nuestro caso muestra tres filas.

Por lo que el ejemplo anterior mostraría los datos de los jugadores en las posiciones 2, 3 y 4.

Además *LIMIT* se puede combinar con *ORDER BY*:

```
SELECT nombre, valoracion FROM jugadores ORDER BY nombre LIMIT 2
```

3.6. Acceso a las columnas

En *MySQL/MariaDB* se puede hacer referencia a una columna usando cualquiera de las siguientes sintaxis:

REFERENCIA DE COLUMNA	SIGNIFICADO
nombreColumna	La columna *nombreColumna* de cualquiera de las tablas usadas en la consulta que contenga una columna con ese nombre.
nombreTabla.nombreColumna	La columna *nombreColumna* de la tabla *nombreTabla* de la base de datos actual.
nombreBasedatos.nombreTabla. nombreColumna	La columna *nombreColumna* de la tabla *nombreTabla* de la base de datos *nombreBasedatos*.

4. Operadores lógicos[8] [9]

En los sistemas tradicionales los operadores lógicos tienen dos posibles valores: verdadero y falso. Sin embargo debido a que en ciertas ocasiones el valor de un elemento no está definido o no es conocido estos dos valores no son suficientes. Para solucionar este problema se hace uso de la lógica trivaluada en la cual a los valores verdadero y falso se le añade el valor desconocido/nulo (*NULL*).

Internamente *MariaDB/MySQL* almacena el valor verdadero como un 1, el falso como un 0 y el valor desconocido/nulo como *NULL*.

```
SELECT TRUE, FALSE, NULL;
+------+-------+------+
| TRUE | FALSE | NULL |
+------+-------+------+
| 1    | 0     | NULL |
+------+-------+------+
```

[7] https://mariadb.com/kb/en/limit/
[8] https://mariadb.com/kb/en/logical-operators/
[9] https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html

En este apartado se verán los operadores lógicos binarios *Y* y *O* además del operador unario *No*.

Para realizar las tablas de verdad de estos operadores se deberá tener en cuenta el nuevo valor *NULL*.

A	B	A AND B	A OR B
falso	falso	falso	falso
falso	verdadero	falso	verdadero
verdadero	falso	falso	verdadero
verdadero	verdadero	verdadero	verdadero
falso	NULL	falso	NULL
NULL	falso	falso	NULL
verdadero	NULL	NULL	verdadero
NULL	verdadero	NULL	Verdadero
NULL	NULL	NULL	NULL

A	NOT A
falso	Verdadero
verdadero	Falso
NULL	NULL

Donde:

- *Y* → representado por *AND* o por *&&*: su resultado es verdadero cuando los dos operandos son verdaderos. Falso si por lo menos uno de ellos es falso (falso y otro valor cualquiera es falso) y nulo en el resto de los casos.
- *O* → representado por *OR* o por ||: Es falso si los dos operadores son falsos. Verdadero si por lo menos uno de ellos es verdadero. Nulo en otro caso.
- *No* → representado por *NOT* o por !: Si es verdadero el resultado es falso y viceversa. Si su valor es nulo el resultado es nulo.

```
SELECT 1 AND NULL, 0 AND NULL, 1 && 0 AND 1, 1 OR NULL, 0 OR NULL, NOT NULL, !(1 OR 0),  !1 AND 1;
+------------+------------+--------------+-----------+-----------+----------+-----------+----------+
| 1 AND NULL | 0 AND NULL | 1 && 0 AND 1 | 1 OR NULL | 0 OR NULL | NOT NULL | !(1 OR 0) | !1 AND 1 |
+------------+------------+--------------+-----------+-----------+----------+-----------+----------+
|       NULL |          0 |            0 |         1 |      NULL |     NULL |         0 |        0 |
+------------+------------+--------------+-----------+-----------+----------+-----------+----------+
```

Un ejemplo de uso de uso de operadores lógicos es:

```
SELECT * FROM jugadores WHERE valoracion=4 OR partidos>=1000
```

Estos operadores de pueden asociar pero hay que tener en cuenta el orden de prioridad de los operadores lógicos: de más prioritario a menos → *NOT*, *AND* y *OR*.

```
SELECT * FROM jugadores WHERE valoracion=4 OR partidos>=1000 AND Rebotes>10
```

5. Operadores Aritméticos [10] [11]

Son operadores binarios que permiten realizar operaciones aritméticas entre constantes, columnas o el resultado de aplicar una función a una columna. Cualquier operación aritmética con un valor *nulo* da nulo.

Operador	Operación
+	Suma
-	Resta
*	Multiplicación

Operador	Operación
/	División
DIV	División entera
%	Módulo

```
SELECT 4+2, 5/2, 5 DIV 2, 5 % 4, 5 + null, 3*4+5, 3*(4+5);
+-----+--------+---------+-------+----------+-------+---------+
| 4+2 | 5/2    | 5 DIV 2 | 5 % 4 | 5 + null | 3*4+5 | 3*(4+5) |
+-----+--------+---------+-------+----------+-------+---------+
|   6 | 2.5000 |       2 |     1 |     NULL |    17 |      27 |
+-----+--------+---------+-------+----------+-------+---------+
```

[10] https://mariadb.com/kb/en/arithmetic-operators/
[11] https://dev.mysql.com/doc/refman/8.0/en/arithmetic-functions.html

6. Operadores de comparación [12] [13]

Nos permiten comparar dos expresiones. *MariaDB/MySQL* permite realizar comparaciones con expresiones de distinto tipo realizándose conversiones de tipos de forma implícita. Podemos comparar cualquier tipo de dato: números, cadenas, fechas, ...

Tenemos los siguientes operadores:

Operador	Nombre operador	Uso
=	Igualdad	Devuelve verdadero si el resultado de las expresiones es igual. Falso si son diferentes. *Null* si alguno de los operadores es nulo.
<=>	Igualdad con *NULL* seguro	Es idéntico al comparador de igualdad excepto en que los nulos se tratan como un valor más por lo que *Null* = *Null* da verdadero.
!= , <>	Desigualdad	Devuelve verdadero si el resultado de las expresiones es diferente. Falso si son iguales. *Null* si alguno de los operadores en nulo.
<	Menor que	Devuelve verdadero si la primera expresión es menor que la segunda. Falso si es mayor o igual. *Null* si alguno de los operadores es nulo.
<=	Menor o igual	Devuelve verdadero si la primera expresión es menor o igual que la segunda. Falso si es mayor. *Null* si alguno de los operadores es nulo.
>	Mayor	Devuelve verdadero si la primera expresión es mayor que la segunda. Falso si es menor o igual. *Null* si alguno de los operadores es nulo.
>=	Mayor o igual	Devuelve verdadero si la primera expresión es mayor o igual que la segunda. Falso si es menor. *Null* si alguno de los operadores es nulo.

```
SELECT  2 = 1, '3.0' = 3, '.05' = 0.05, NULL = 1, NULL <=> 3, NULL <=> NULL;
+-------+-----------+--------------+----------+------------+---------------+
| 2 = 1 | '3.0' = 3 | '.05' = 0.05 | NULL = 1 | NULL <=> 3 | NULL <=> NULL |
+-------+-----------+--------------+----------+------------+---------------+
|     0 |         1 |            1 |     NULL |          0 |             1 |
+-------+-----------+--------------+----------+------------+---------------+
```

```
SELECT  25 != 21, 'casa' < 'caso', '10' <= '5', 10 <= 5, '2017-11-05' > '2017-11-04'
+----------+-----------------+-------------+---------+----------------------------+
| 25 != 21 | 'casa' < 'caso' | '10' <= '5' | 10 <= 5 | '2017-11-05' > '2017-11-04' |
+----------+-----------------+-------------+---------+----------------------------+
|        1 |               1 |           1 |       0 |                          1 |
+----------+-----------------+-------------+---------+----------------------------+
```

7. Comentarios [14] [15]

En las sentencias se pueden usar tres tipos de comentarios:

- Comentario de línea: desde los símbolos -- hasta el final de la línea

```
SELECT nombre FROM jugadores;  -- Comentario de línea. Después del -- tiene que ponerse un espacio
```

- Comentario de línea: desde el símbolo # hasta el final de la línea.

```
SELECT nombre FROM jugadores;  # Comentario de línea
```

- Comentario multilínea: Desde los símbolos /* hasta los símbolos */. Pueden estar en distintas líneas, pero no es obligatorio.

```
SELECT nombre FROM jugadores; /* Comentario
multilínea */
```

[12] https://mariadb.com/kb/en/comparison-operators/
[13] https://dev.mysql.com/doc/refman/8.0/en/comparison-operators.html
[14] https://mariadb.com/kb/en/comment-syntax/
[15] https://dev.mysql.com/doc/refman/8.0/en/comments.html

8. NULL[16] [17]y NOT NULL[18]

Para comprobar si una columna tiene un valor nulo no se puede utilizar el operador igual ya que, como se ha visto en el punto anterior, siempre dará nulo. Para estos casos se usa el operador *IS NULL*.

Su sintaxis es:

```
<expresión> IS [NOT] NULL
```

Donde:

- Expresión: es cualquier expresión valida en *MariaDB/MySQL*.
- *IS NULL*: es verdadero cuando la expresión toma un valor nulo. Falso en el caso contrario.
- *IS NOT NULL*: es verdadero cuando la expresión toma un valor distinto de nulo. Falso cuando toma un valor nulo.

Ejemplos de consultas con *NULL* y *NOT NULL* son:

- Seleccionar los jugadores que tienen un valor nulo en la columna tapones:

```
SELECT * FROM jugadores WHERE tapones IS NULL;
```

- Seleccionar los jugadores que no tienen un valor nulo en la columna tapones:

```
SELECT * FROM jugadores WHERE tapones IS NOT NULL;
```

9. Operador IN[19] [20]

El operador *IN* se utiliza cuando se quiere comprobar si el valor de una expresión esta en un conjunto de valores.

Imaginemos la siguiente consulta: Consultar los nombres de los jugadores cuya valoración sea 3, 5,7 o 8. Con lo visto hasta ahora tendríamos la consulta siguiente:

```
SELECT * FROM jugadores WHERE valoracion=3 OR valoracion=5 OR valoracion=7 OR valoracion=8;
```

El operador *IN* permite simplificar este tipo de consultas. Su sintaxis es la siguiente:

```
Expresión [NOT] IN (Lista de valores separados por comas)
```

Donde:

- Expresión: es cualquier expresión validada en *MariaDB/MySQL*.
- Lista de valores: es la lista de valores, de cualquier tipo, a comprobar.
- *IN*: si el valor de la expresión (valoración en nuestro caso) es igual a alguno de los valores presentes en la lista (3,5,7,8 en nuestro caso) el resultado el verdadero. El resultado es falso si es distinto a todos los valores.
- *NOT IN*: realiza la operación inversa. Devuelve verdadero si el valor no está en la lista y falso si el elemento está presente en la misma.

El ejemplo anterior con el operador *IN* quedaría:

```
SELECT nombre FROM jugadores WHERE valoracion IN(3,5,7,8);
```

[16] https://mariadb.com/kb/en/is-null/
[17] https://dev.mysql.com/doc/refman/8.0/en/working-with-null.html
[18] https://mariadb.com/kb/en/is-not-null/
[19] https://mariadb.com/kb/en/in/
[20] https://dev.mysql.com/doc/refman/8.0/en/comparison-operators.html#operator_in

Si quememos obtener los jugadores cuyo nombre sean *'Kobe Bryant'*, *'Magic Johnson'* o *'Jerry West'* mediante el operador *IN* la consulta seria:

```
SELECT * FROM jugadores WHERE nombre IN ('Kobe Bryant', 'Magic Johnson' , 'Jerry West');
```

La cual es equivalente a:

```
SELECT * FROM jugadores WHERE nombre='Kobe Bryant' OR nombre='Magic Johnson' OR nombre='Jerry West';
```

10. Operador BETWEEN[21] [22]

El operador *BETWEEN* permite comprobar si un valor está comprendido dentro de un rango indicado por un valor inferior y un valor superior incluyendo estos valores.

Su formato es:

```
Expresión [NOT] BETWEEN valorInferior AND valorSuperior
```

Donde:

- Expresión: es cualquier expresión valida en *MariaDB/MySQL*.
- *BETWEEN*: devuelve verdadero si expresión es mayor o igual que el valor Inferior y es menor o igual que el valor superior.
- *NOT BETWEEN*: es el caso contrario. Devuelve verdadero si expresión no está dentro del rango definido y falso si esta dentro.

Por ejemplo en la consulta siguiente se obtiene el nombre y partidos de los jugadores que hayan jugado entre 850 y 1030 partidos.

```
SELECT nombre, partidos FROM jugadores WHERE partidos BETWEEN 850 AND  1030;
```

Esta consulta es equivalente a:

```
SELECT nombre, partidos FROM jugadores WHERE partidos >= 850 AND partidos <= 1030;
```

11. Operadores para cadenas de caracteres[23] [24]

Hasta ahora para realizar comparaciones con cadenas de caracteres se ha visto el operador igual y el operador *IN* los cuales comprueban si una cadena de caracteres es igual a otra dada.

Existe además el operador *LIKE* cuya sintaxis es la siguiente:

```
<expresión> [NOT] LIKE <patrón>
```

Donde:

- Expresión: es cualquier expresión valida en *MariaDB/MySQL*.
- *LIKE*: permite comparar cadenas bajo un patrón determinado. Por ejemplo cadenas que empiecen con un carácter o cadenas que contengan otra cadena. LIKE devuelve verdadero si la expresión cumple con el patrón especificado y falso si no cumple con el patrón.
- *NOT LIKE*: devuelve verdadero si la expresión no cumple con el patrón especificado y falso cumple con el patrón.
- Patrón: es la secuencia de caracteres a comparar en la que no se distingue mayúsculas de minúsculas. Puede usar los siguientes comodines:
  - %: representa una cadena de cero o mas caracteres.
  - _ : representa un único carácter.

[21] https://mariadb.com/kb/en/between-and/
[22] https://dev.mysql.com/doc/refman/8.0/en/comparison-operators.html#operator_between
[23] https://mariadb.com/kb/en/like/
[24] https://dev.mysql.com/doc/refman/8.0/en/pattern-matching.html

Un ejemplo de consulta con *LIKE* sería el de obtener los nombres de los jugadores en cuyo nombre aparezca una *U*.

```
SELECT nombre FROM jugadores WHERE nombre LIKE "%U%";
```

En la siguiente tabla se muestran ejemplos de patrones para su uso con la cláusula *LIKE*:

Patrón	Descripción
' Jerry West'	La cadena ' Jerry West'. Es idéntico a igual.
'_A%'	Las cadenas que de segundo carácter tengan una A.
'___A_'	Las cadenas de 5 caracteres que en la posición 4 tenga una A.
'A%A'	Las cadenas que comiencen y terminen en A.
'%A%'	Las cadenas que contengan una A no necesariamente al principio o al final.

Aunque no es *SQL* estándar *MariaDB/MySQL* permite usar *LIKE* también con números.

12. Variables[25] [26]

MariaDB/MySQL dispone de la posibilidad de crear variables de usuario y variables globales del servidor.

Las variables de usuario permiten almacenar valores y hacer referencia a ellas en otras consultas con lo que se consigue una transferencia de información entre consultas.

Cada usuario solo tiene solo tiene acceso a las variables de usuario creadas por él en una sesión determinada puesto que al cerrar la conexión todas las variables de sesión se eliminan.

Al igual que con los nombre de columnas los nombres de variables de usuario no son sensibles a mayúsculas.

Para referenciar una variable de usuario se le antepone una @ a su nombre.

Se pueden crear variables de usuario mediante dos sentencias diferentes:

- Mediante sentencia *SET*

```
SET @hola = 'Hola';
SELECT @hola;
```

- Mediante sentencia *SELECT*

```
SELECT @a:=10, @maxPuntos:=puntos FROM jugadores ORDER BY puntos DESC LIMIT 1;
SELECT @a, @maxPuntos, nombre, puntos FROM jugadores WHERE puntos=@maxPuntos;
```

Cualquier variable de usuario a la que no se le ha asignado un valor tendrá un valor nulo.

Por otra parte, las variables globales del servidor representan distintos aspectos de la configuración interna del servidor y ser referencian con doble @@.

Por ejemplo para mostrar la siguiente información del servidor: el directorio en donde se almacenan las bases de datos, idioma de los mensajes del servidor, idioma de los nombres de los días de la semana, la versión del servidor se usará la sentencia:

```
SELECT @@DATADIR, @@LC_MESSAGES, @@LC_TIME_NAMES, @@VERSION;
```

Para establecer un valor se usa *SET*:

```
SET @@LC_TIME_NAMES = "es_ES";
```

[25] https://mariadb.com/kb/en/user-defined-variables/
[26] https://mariadb.com/kb/en/server-system-variables/

Por ejemplo ejecuta las siguientes sentencias y comprueba su resultado.

```
SET @@LC_TIME_NAMES = "en_US";
SELECT @@LC_TIME_NAMES, dayname(current_date());

SET @@LC_TIME_NAMES = "es_ES";
SELECT @@LC_TIME_NAMES, dayname(current_date());
```

Estas modificaciones se pierden, estableciéndose su valor predeterminado, al reiniciarse el servidor.

Para obtener un listado de las variables del sistema se usa el comando *SHOW VARIABLES*. Se puede usar *LIKE* para filtrar filas.

```
SHOW VARIABLES; -- Muestra todas las variables del servidor
SHOW VARIABLES LIKE '%char%'; -- Muestra todas las variables del servidor con la cadena char en su nombre
```

13. Operadores de control de flujo[27] [28]

MariaDB/MySQL dispone de dos operadores para el control de flujo*: IF* y *CASE*.

13.1. IF

El operador *IF* dispone de tres variantes:

- IF (expr1, expr2, expr3)

  Si *expr1* es verdadero (*expr1* no tiene un valor 0, no es falso y no es nulo) entonces *IF* devuelve *expr2*. En otro caso devuelve expr3.

```
SELECT IF(1>2,2,3),IF(1<2,'Si','No'),IF(STRCMP('test','test1'),'No',Si');
+-------------+--------------------+-------------------------------------+
| IF(1>2,2,3) | IF(1<2,'Si','No')  | IF(STRCMP('test','test1'),'No','Si') |
+-------------+--------------------+-------------------------------------+
|           3 | yes                | no                                  |
+-------------+--------------------+-------------------------------------+
```

- IFNULL(expr1, expr2):

  Si *expr1* no es nulo, *IFNULL* devuelve *expr1*. En otro caso devuelve *expr2*.

```
SELECT IFNULL(1,0), IFNULL(NULL,10), IFNULL(1/0,10), IFNULL(1/0,'Si')
+-------------+----------------+----------------+------------------+
| IFNULL(1,0) | IFNULL(NULL,10) | IFNULL(1/0,10) | IFNULL(1/0,'Si') |
+-------------+----------------+----------------+------------------+
|           1 |              10 |        10.0000 | yes              |
+-------------+----------------+----------------+------------------+
```

- NULLIF(expr1, expr2)

  Devuelve nulo si *expr1* = *expr2* es cierto. En otro caso devuelve *expr1*.

```
SELECT NULLIF(1,1), NULLIF(1,2);
+-------------+-------------+
| NULLIF(1,1) | NULLIF(1,2) |
+-------------+-------------+
|        NULL |           1 |
+-------------+-------------+
```

[27] https://mariadb.com/kb/en/control-flow-functions/
[28] https://dev.mysql.com/doc/refman/5.7/en/control-flow-functions.html

13.2. CASE

El operador *CASE* dispone de dos variaciones:

Case busca entre los valores de *WHEN* aquel que coincide con su valor y devuelve el valor indicado por el *THEN* correspondiente. Si no casa con ningún valor devuelve el valor del *ELSE* y si no hay clausula *ELSE* devuelve *NULL*.

```
CASE valor
    WHEN valor1 THEN resultado1
    [WHEN valor2 THEN resultado1]
    ……
    [WHEN valorN THEN resultadoN]
    [ELSE resultado]
END
```

Devuelve el primer resultado en el que se cumpla la condición del clausula *WHEN*. Si no casa con ningún valor devuelve el valor del *ELSE* y si no hay clausula *ELSE* devuelve *NULL*.

```
CASE
    WHEN condición1 THEN resultado1
    [WHEN condición2 THEN resultado2]
    ……
     [WHEN condiciónn THEN resultadon ...]
     [ELSE resultado]
END
```

Veamos uno ejemplos de ambas opciones:

```
SELECT nombre, valoracion,
   CASE valoracion
      WHEN 1 THEN "Normal"
      WHEN 2 THEN "Bueno"
      WHEN 3 THEN "Muy bueno"
      ELSE "Estrella"
   END
FROM jugadores;
```

```
SELECT nombre, partidos,
    CASE
        WHEN partidos>=1000 THEN "Veterano"
        WHEN partidos>700 AND partidos<1000 THEN "Con experiencia"
        ELSE "Novato"
    END
FROM jugadores;
```

14. Resumen de comandos MariaDB/MySQL

Breve reseña de comandos útiles de MariaDB/MySQL:

COMANDO	DESCRIPCIÓN
select version(), current_date;	Muestra la versión y fecha actuales
select user(), connection_id();	Muestra el usuario con el que nos conectamos al servidor *MariaDB/MySQL* y su id de conexión.
show databases;	Muestra las bases de datos que existen en el servidor.
select database ();	Muestra la base de datas actualmente seleccionada
use nombreBD;	Accede a una base de datos
create database nombreBD;	Crea una base de datos
drop database nombreBD;	Elimina una base de datos
show tables [from nombreBD] ;	Muestra las tablas que posee la base de datos seleccionada
describe nombreTabla;	Muestra la estructura de una tabla
show [full] columns from jugadores;	Describe es un sinónimo de este comando. Con la opción full muestra información más detallada.
show create table nombreTabla;	Muestra la instrucción usas para crear la tabla
drop table nombreTabla;	Borra un tabla de la base de datos
select * from nombreTabla;	Mostrar todos los datos de una tabla
show errors;	Muestra errores
show open tables;	Muestra las tablas abiertas en la cache de tablas
show status;	Proporciona información del estado del servidor
show warnings;	Muestra los mensajes de error, advertencia y notas retornadas por el último comando que haya generado algún mensaje, o nada si el último mensaje que haya usado una tabla no ha generado mensajes.
kill thread_id;	Puede matar un flujo con el comando KILL thread_id .
source nombreFichero;	Ejecuta un fichero en modo por lotes
tee nombreFichero;	Guarda los comandos ejecutados y sus resultados en el fichero que se le indique.
notee;	Detiene el registro de operaciones.
help comando;	Muestra información sobre el comando especificados. Por ejemplo help show o help select.
check table nombreTabla;	Consulta el estado de integridad de una tabla