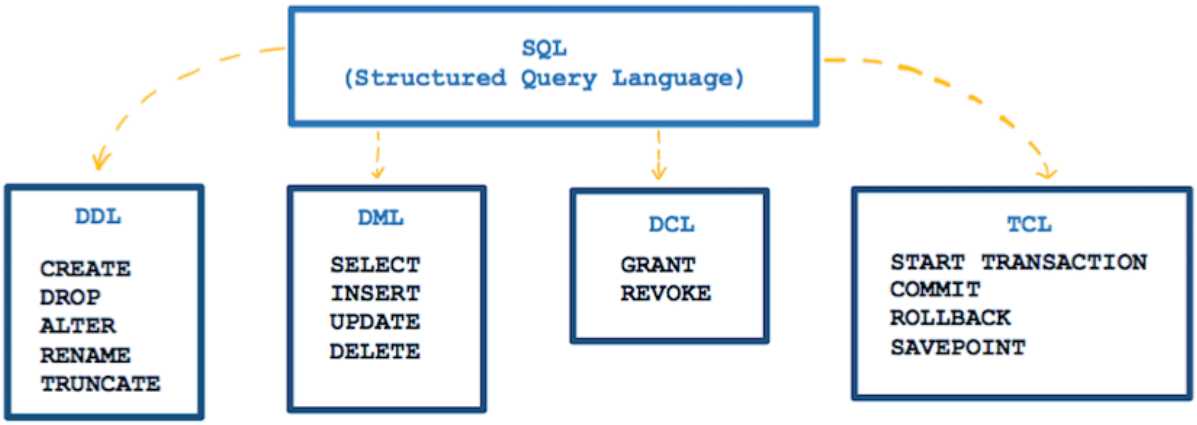
TEMA 05 – CONSULTAS SQL SOBRE UNA TABLA

Realización de consultas SQL



- El DML (Data Manipulation Language) es la parte de SQL dedicada a la manipulación de los datos
- Las sentencias DML son las siguientes:
    - SELECT: realizar consultas y extraer información de la base de datos
    - INSERT: insertar registros en las tablas de la base de datos
    - UPDATE: actualizar los registros de una tabla
    - DELETE: eliminar registros de una tabla
- *Este tema es solo de SELECT

Consultas básicas sobre una tabla

- Sintaxis de la instrucción SELECT

```
SELECT [DISTINCT] select_expr [, select_expr ...]
[FROM table_references]
[WHERE where_condition]
[GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
[HAVING having_condition]
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_COUNT | row_COUNT OFFSET offset}]
```

    - Orden de ejecución:
        - FROM
        - WHERE
        - GROUP BY
        - HAVING
        - SELECT
        - ORDER BY
        - LIMIT
        - ***F**uera de la **W**arner **G**ay **H**ay **SOL***
    - El resultado siempre será una tabla de datos
    - Nos permite almacenar los resultados como una nueva en la base de datos
    - Los resultados se pueden consultar por otras nuevas consultas
- Cláusula SELECT
    - Qué hace
        - Indica cuáles serán las columnas que tendrá la tabla de resultados
        - Opciones que podemos indicar:
            - Nombre de una columna
            - Constante que aparecerá en todas las filas de la tabla resultado
            - Expresión que nos permite calcular nuevos valores
    - Cómo obtener los datos de todas las columnas de una tabla
        - SELECT *
    - Cómo obtener los datos de algunas columnas de una tabla

```
SELECT apellido1, apellido2, nombre
FROM alumno;
```

        - Mostrará las columnas siguiendo el orden que se haya indicado
    - Cómo realizar comentarios en sentencias SQL
        - -- Comentario de una línea (hasta Intro)
        - /* Comentario que puede ser de varias líneas */
        - # Comentario también (hasta Intro)
    - Cómo obtener columnas calculadas

```sql
SELECT id, cantidad_comprada, precio_por_elemento, cantidad_comprada *
    precio_por_elemento
FROM ventas;
```

Y el resultado sería el siguiente:

id	cantidad_comprada	precio_por_elemento	cantidad_comprada * precio_por_elemento
1	2	1.50	3.00
2	5	1.75	8.75
3	7	2.00	14.00
4	9	3.50	31.50
5	6	9.99	59.94

- Cómo realizar alias de columnas con AS
  - Útil cuando estamos calculando columnas

```sql
SELECT id, cantidad_comprada, precio_por_elemento, cantidad_comprada *
    precio_por_elemento AS 'precio total'
FROM ventas;
```

Si el nuevo nombre que estamos creando para el alias contiene espacios en blanco es necesario usar comillas simples.

Al crear este alias obtendremos el siguiente resultado:

id	cantidad_comprada	precio_por_elemento	precio total
1	2	1.50	3.00
2	5	1.75	8.75
3	7	2.00	14.00

- Cómo utilizar funciones de MySQL en la cláusula SELECT
  - Funciones con cadenas

Función	Descripción
CONCAT	Concatena cadenas
CONCAT_WS	Concatena cadenas con un separador
LOWER	Devuelve una cadena en minúscula
UPPER	Devuelve una cadena en mayúscula
SUBSTR	Devuelve una subcadena

  - Funciones matemáticas

Función	Descripción
ABS()	Devuelve el valor absoluto
POW(x,y)	Devuelve el valor de x elevado a y
SQRT()	Devuelve la raíz cuadrada
PI()	Devuelve el valor del número PI
ROUND()	Redondea un valor numérico
TRUNCATE()	Trunca un valor numérico

- Funciones de fecha y hora

Función	Descripción
NOW()	Devuelve la fecha y la hora actual
CURTIME()	Devuelve la hora actual

- Ejemplo de CONCAT:

```
1  SELECT CONCAT(nombre, apellido1, apellido2) AS nombre_completo
2  FROM alumno;
```

nombre_completo
MaríaSánchezPérez
JuanSáezVega

```
1  SELECT CONCAT_WS(' ', nombre, apellido1, apellido2) AS nombre_completo
2  FROM alumno;
```

nombre_completo
María Sánchez Pérez
Juan Sáez Vega

- CONCAT devolverá NULL cuando alguna de las cadenas que está concatenando es igual a NULL
- CONCAT_WS omitirá los NULL y concatenará con el resto de cadenas

- Modificadores ALL, DISTINCT y DISTINCTROW
  - ALL: de deben incluir todas las filas, incluidas las repetidas (por defecto)
  - DISTINCT: elimina las filas repetidas en el resultado de la consulta
  - DISTINCTROW: sinónimo de DISTINCT

- Cláusula ORDER BY

```
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
```

  - Cómo ordenar de forma ascendente: por defecto
    - En nombre es de la A->Z
    - Solo tiene en cuenta la columna que le ponemos

```
SELECT apellido1, apellido2, nombre
FROM alumno
ORDER BY apellido1 ASC;
```

Ramírez	Gea	Pepe
Sáez	Vega	Juan
Sánchez	Pérez	María
Sánchez	Ortega	Lucía

    - Si hay dos con mismo primer apellido, el segundo no lo tiene en cuenta
  - Cómo ordenar de forma descendente

```
SELECT apellido1, apellido2, nombre
FROM alumno
ORDER BY apellido1 DESC;
```

  - Cómo ordenar utilizando múltiples columnas

```
SELECT apellido1, apellido2, nombre
FROM alumno
ORDER BY apellido1, apellido2, nombre;
```

    - También podemos indicar sobre la posición donde aparecen en el SELECT:

```
SELECT apellido1, apellido2, nombre
FROM alumno
ORDER BY 1, 2, 3;
```

- Cláusula LIMIT
  - Permite limitar el número de filas que se incluyen en el resultado

```
[LIMIT {[offset,] row_COUNT | row_COUNT OFFSET offset}]
```

  - Row_COUNT es el número de filas que queremos obtener
  - Offset es el número de filas que nos saltamos antes de empezar a contar
    - La primera que se obtiene es offset + 1
- Cláusula WHERE
  - Qué hace
    - Permite añadir filtros

- Estas condiciones se denominan predicados y el resultado puede ser verdadero, falso o desconocido
    - Desconocido: cuando alguno de los valores utilizados tiene valor NULL
- 5 tipos de condiciones
    - Para comparar valores o expresiones
    - Para comprobar si un valor está dentro de un rango de valores
    - Para comprobar si un valor está dentro de 1 conjunto de valores
    - Para comparar cadenas con patrones
    - Para comprobar si una columna tiene valores NULL
- Los operandos pueden ser nombres de columnas, constantes o expresiones

o Operadores disponibles en MySQL

- Aritméticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo

- Comparación

Operador	Descripción
<	Menor que
<=	Menor o igual
>	Mayor que
>=	Mayor o igual
<>	Distinto
!=	Distinto
=	Igual que

- Lógicos

Operador	Descripción
AND	Y lógica
&&	Y lógica
OR	O lógica
||	O lógica

NOT	Negación lógica
!	Negación lógica

- Comparación se puede usar con fechas:

3. Obtener el nombre y la fecha de nacimiento de todos los alumnos nacieron después del 1 de enero de 1997.

```
SELECT nombre, fecha_nacimiento
FROM alumno
WHERE fecha_nacimiento >= '1997/01/01';
```

- Operador BETWEEN

```
expresión [NOT] BETWEEN cota_inferior AND cota_superior
```

- Para comprobar si un valor está dentro de un rango de valores

```
SELECT *
FROM pedido
WHERE fecha_pedido BETWEEN '2018-01-01' AND '2018-01-31';
```

- Operador IN / NOT IN
  - Permite comprobar si el valor de una determinada columna está incluido en una lista de valores

```
SELECT *
FROM alumno
WHERE apellido1 IN (`Sánchez`, `Martínez`, `Domínguez`);
```

- Operador LIKE

```
columna [NOT] LIKE patrón
```

- Para comparar si una cadena de caracteres coincide con un patrón
  - %: equivale a cualquier conjunto de caracteres
  - _: equivale a cualquier caracter

Devuelva un listado de todos los alumnos que su nombre tenga el carácter a.

```
SELECT *
FROM alumno
WHERE nombre LIKE '%a%';
```

Devuelva un listado de todos los alumnos que tengan un nombre de cuatro caracteres.

```
SELECT *
FROM alumno
WHERE nombre LIKE '____';
```

- Si queremos poner un carácter de escape: \

```sql
SELECT *
FROM producto
WHERE nombre LIKE 'A$%BC%' ESCAPE '$';
```

Por defecto, se utiliza el carácter '' como carácter de escape. De modo que podríamos escribir la consulta de la siguiente manera.

```sql
SELECT *
FROM producto
WHERE nombre LIKE 'A\%BC%';
```

- Operador REGEXP y expresiones regulares
  - Nos permite realizar búsquedas mucho más potentes haciendo uso de expresiones regulares
  - Impacto en el rendimiento, utilizarlas con precaución
- Operadores IS e IS NOT
  - Permiten comprobar si el valor es NULL o no lo es

Ejemplo:

Obtener la lista de alumnos que tienen un valor `NULL` en la columna `teléfono`.

```sql
SELECT *
FROM alumno
WHERE teléfono IS NULL;
```

Ejemplo:

Obtener la lista de alumnos que no tienen un valor `NULL` en la columna `teléfono`.

```sql
SELECT *
FROM alumno
WHERE teléfono IS NOT NULL;
```

- Funciones disponibles en MySQL
  - Funciones con cadenas

Función	Descripción
`CONCAT`	Concatena cadenas
`CONCAT_WS`	Concatena cadenas con un separador
`LOWER`	Devuelve una cadena en minúscula
`UPPER`	Devuelve una cadena en mayúscula
`SUBSTR`	Devuelve una subcadena
`LEFT`	Devuelve los caracteres de una cadena, empezando por la izquierda
`RIGHT`	Devuelve los caracteres de una cadena, empezando por la derecha
`CHAR_LENGTH`	Devuelve el número de caracteres que tiene una cadena
`LENGTH`	Devuelve el número de bytes que ocupa una cadena

REVERSE	Devuelve una cadena invirtiendo el orden de sus caracteres
LTRIM	Elimina los espacios en blanco que existan al inicio de una cadena
RTRIM	Elimina los espacios en blanco que existan al final de una cadena
TRIM	Elimina los espacios en blanco que existan al inicio y al final de una cadena
REPLACE	Permite reemplazar un carácter dentro de una cadena

- Funciones de fecha y hora

Función	Descripción
NOW()	Devuelve la fecha y la hora actual
CURTIME()	Devuelve la hora actual
ADDDATE	Suma un número de días a una fecha y calcula la nueva fecha
DATE_FORMAT	Nos permite formatear fechas
DATEDIFF	Calcula el número de días que hay entre dos fechas
YEAR	Devuelve el año de una fecha
MONTH	Devuelve el mes de una fecha
MONTHNAME	Devuelve el nombre del mes de una fecha
DAY	Devuelve el día de una fecha
DAYNAME	Devuelve el nombre del día de una fecha
HOUR	Devuelve las horas de un valor de tipo DATETIME
MINUTE	Devuelve los minutos de un valor de tipo DATETIME
SECOND	Devuelve los segundos de un valor de tipo DATETIME

Configuración regional en MySQL Server (*locale*)

Importante: Tenga en cuenta que para que los nombres de los meses y las abreviaciones aparezcan en español deberá configurar la variable del sistema `lc_time_names`. Esta variable afecta al resultado de las funciones `DATE_FORMAT`, `DAYNAME` y `MONTHNAME`.

En MySQL las variables se pueden definir como **variables globales** o **variables de sesión**. La diferencia que existe entre ellas es que una variable de sesión pierde su contenido cuando cerramos la sesión con el servidor, mientras que una variable global mantiene su valor hasta que se realiza un reinicio del servicio o se modifica por otro valor. **Las variables globales sólo pueden ser configuradas por usuarios con privilegios de administración.**

Para configurar la variable `lc_time_names` como una **variable global**, con la configuración regional de España tendrá que utilizar la palabra reservada `GLOBAL`, como se indica en el siguiente ejemplo.

```
1  SET GLOBAL lc_time_names = 'es_ES';
```

Para realizar la configuración como una **variable de sesión** tendría que ejecutar:

```
1  SET lc_time_names = 'es_ES';
```

Configuración de la zona horaria en MySQL Server (*time zone*)

Importante: Tenga en cuenta que también será necesario configurar nuestra zona horaria para que las funciones relacionadas con la hora devuelvan los valores de nuestra zona horaria. En este caso tendrá que configurar la variable del sistema `time_zone`, como **variable global** o como **variable de sesión**. A continuación, se muestra un ejemplo de cómo habría que configurar las variables para la zona horaria de Madrid.

```
1  SET GLOBAL time_zone = 'Europe/Madrid';
2  SET time_zone = 'Europe/Madrid';
```

Podemos comprobar el estado de las variables haciendo:

```
1  SELECT @@GLOBAL.time_zone, @@SESSION.time_zone;
```

En la documentación oficial pueden encontrar más información sobre la configuración de la zona horaria en MySQL Server.

Ejemplos:

`NOW()` devuelve la fecha y la hora actual.

```
1  SELECT NOW();
```

`CURTIME()` devuelve la hora actual.

```
1  SELECT CURTIME();
```

- Funciones matemáticas

Función	Descripción
`ABS()`	Devuelve el valor absoluto
`POW(x,y)`	Devuelve el valor de x elevado a y
`SQRT`	Devuelve la raíz cuadrada
`PI()`	Devuelve el valor del número `PI`
`ROUND`	Redondea un valor numérico
`TRUNCATE`	Trunca un valor numérico
`CEIL`	Devuelve el **entero inmediatamente superior o igual**
`FLOOR`	Devuelve el **entero inmediatamente inferior o igual**
`MOD`	Devuelve el resto de una división

Otros

Ejemplo: Resta entre dos valores de tipo `DATE`

En este ejemplo estamos convirtiendo las cadenas a un valor de tipo `DATE` y luego se realiza la resta. En este caso, los datos de tipo `DATE` los convierte a datos de tipo entero pero de una forma más precisa que la conversión anterior, ya que la cadena `'2008-08-31'` la convierte en el número entero `20080831` y la cadena `'2008-09-01'` en el número entero `20080901`, por este motivo el resultado de la resta es `-70`.

```
SELECT CAST('2021-01-31' AS DATE) - CAST('2021-02-01' AS DATE);
 -> -70
```

Ejemplo: Resta entre dos valores de tipo `INT`

```
SELECT 20210131 - 20210201;
 -> -70
```

Ejemplo: Resta entre dos valores de tipo `DATE` utilizando la función `DATEDIFF`

```
SELECT DATEDIFF('2021-01-31', '2021-02-01');
 -> -1
```

```
SELECT fecha_esperada, fecha_entrega,  DATEDIFF(fecha_entrega, fecha_esperada)
FROM pedido;
```

fecha_esperada	fecha_entrega	DATEDIFF(fecha_entrega, fecha_esperada)
2021-01-31	2021-02-01	-1