Certified Tech Developer

The Ultimate Degree

Introducción a la informática

BASE DE DATOS

MYSQL WORKBENCH

Cheaty Sheet

COMENTAR

/* Esto es un comentario */

SELECT

Se empiezan siempre las consultas con esa palabra. Se complementa con FROM, ya que expresamos qué es lo que queremos seleccionar y de cuál tabla.

```SQL
SELECT nombre_columna, nombre_columna, ...
FROM nombre_tabla;
```

WHERE

Se utiliza para expresar condiciones y así filtrar los resultados.

```SQL
SELECT nombre_columna_1, nombre_columna_2, ...
FROM nombre_tabla
WHERE condicion;
```

ORDER BY

Permite ordenar los resultados según el valor de la columna especificada.

```SQL
SELECT nombre_columna1, nombre_columna2
FROM tabla
WHERE condicion
ORDER BY nombre_columna1;
```

BETWEEN

Nos permite obtener valores dentro de un rango. Incluye los extremos. Funciona con números, fechas y textos.

```SQL
SELECT nombre, edad
FROM alumnos
WHERE edad BETWEEN 6 AND 12;
```

LIKE

Cuando hacemos un filtro con WHERE, podemos especificar un patrón de búsqueda que nos permita especificar algo concreto que queremos encontrar en los registros.

COMODIN %

Es un sustituto que representa cero, uno o varios caracteres.

COMODIN_

Es un sustituto de un solo carácter.

```SQL
SELECT nombre
FROM usuarios
WHERE edad LIKE '_a%';
```

LIMIT

Su funcionalidad es limitar la cantidad de filas devueltas en la consulta.

```SQL
SELECT *
FROM peliculas
WHERE premios > 4
LIMIT 10;
```

OFFSET

Nos permite especificar desde qué fila comenzar la recuperación de datos solicitados. Recordar que, si pongo OFFSET 20, va a devolver a partir del 21.

```
SELECT id, nombre, apellido
FROM alumnos
LIMIT 20
OFFSET 20;
```

ALIAS

Se usan para dar un nombre temporal y más amigable a tablas, columnas y funciones. Para emplearlos utilizamos **AS.**

```SQL
SELECT nombre_columna1 AS alias_nombre_columna1
FROM nombre_tabla;
```

FUNCIONES DE AGREGACIÓN

COUNT

Devuelve la cantidad de filas o registros que cumplen con el criterio.

```sql
SELECT COUNT(*) FROM movies;
```

AVG

Nos da el promedio de una columna de valores.

```sql
SELECT AVG(rating) FROM movies;
```

SUM

Devuelve la suma de una columna con **valores numéricos.**

```sql
SELECT SUM(length) FROM movies;
```

MAX

Devuelve el valor máximo de una columna con **valores numéricos.**

```sql
SELECT MAX(length) FROM movies;
```

MIN

Devuelve el valor mínimo.

```sql
SELECT MIN(rating) FROM movies;
```

GROUP BY

Se usa para agrupar registros de la tabla resultante de una consulta.

```sql
SELECT columna_1
FROM nombre_tabla
WHERE condition
GROUP BY columna_1;
```

HAVING

Funciona como WHERE, a diferencia que HAVING se va a poder usar en conjunto con funciones de agregación (también funciona con alias).

```sql
SELECT columna_1
FROM nombre_tabla
WHERE condition
GROUP BY columna_1
HAVING condition_Group
ORDER BY columna_1;
```

JOINS

Permite hacer consultas a varias tablas y unir estos resultados. **INNER JOIN** hace una cruza entre dos tablas, pero si hay alguna fila que tiene datos NULL no la muestra en el resultado. Se necesita aclarar dónde esta el cruce (PK & FK), y la palabra **ON** nos permite indicar el filtro en donde se realiza el cruce.

```sql
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes
INNER JOIN ventas
ON clientes.id = ventas.cliente_id
```

DISCTINCT

Nos devuelve valores únicos de una columna. A medida que agregamos columnas dentro del **SELECT,** se van agregando más datos, ya que busca devolvernos el dato único de la combinación. Ej: si abajo agregáramos la columna película, los nombres de los actores que hayan participado en más de una, se repetirían.

```sql
SELECT DISTINCT actors.first_name, actors.last_name
FROM actors
INNER JOIN actor_movie ON actors.id =
actor.movie.actor_id
INNER JOIN movies ON movies.id = actor_movie.movie_id
WHERE movies.title LIKE '%Harry Potter%';
```

FUNCIONES DE ALTERACIÓN.

Permiten alterar como se presentan los datos solicitados en la query, no modifica la base de datos.

CONCAT

Nos permite concatenar dos o más expresiones.

```sql
SELECT CONCAT('Hola ', 'a ', 'todos.');
```

COALESCE

Lo usamos para obtener la primera expresión not null de cada registro, aclarando entre paréntesis las columnas a tener en cuenta.

```sql
SELECT id, nombre, COALESCE(celular, casa, trabajo) AS telefono
FROM clientes;
```

DATEDIFF

Nos devuelve la diferencia entre dos fechas, tomando como gradualidad el intervalo especificado.

```sql
SELECT DATEDIFF(hour, '2017/08/25 07:00', '2017/08/25 12:45');
```

TIMESTAMPDIFF

Es más exacto. Los intervalos se expresan igual que DATEDIFF. Si queremos consultar la diferencia con el momento actual usamos **NOW().**

EXTACT

Nos permite extraer partes de la fecha. Luego del paréntesis debemos especificar cuál parte es la que nos interesa extraer.

```sql
SELECT EXTRACT(SECOND FROM '2014-02-13 08:44:21');
```

REPLACE

Lo usamos para reemplazar una secuencia de caracteres por otro string. Se escribe entre paréntesis :

1 – Dónde buscar: string, columna...

2- Qué quiero cambiar.

3- Por qué lo quiero reemplazar.

```SQL
SELECT REPLACE('abc abc', 'a', 'B');
> Bbc Bbc
```

```SQL
SELECT REPLACE('abc abc', 'A', 'B');
> abc abc
```

DATE FORMAT

Nos permite cambiar el formato de salida de una fecha.

```SQL
SELECT DATE_FORMAT('2017-06-15', '%Y');
> '2017'
```

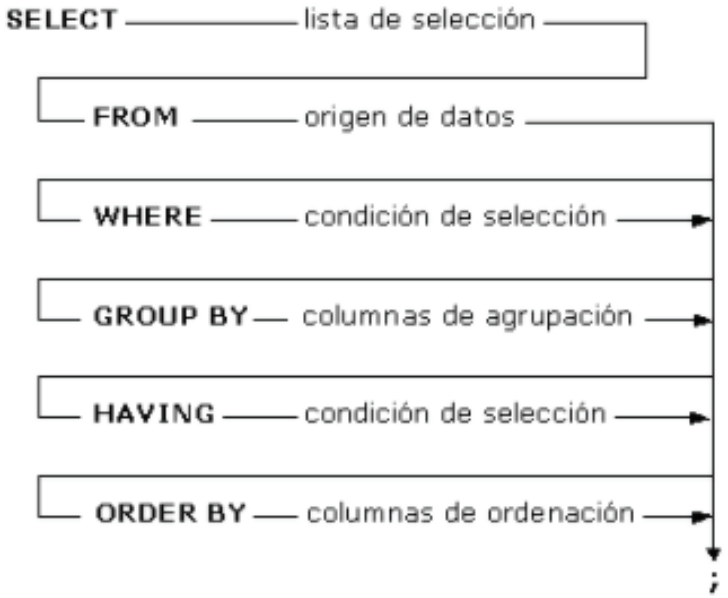
```SQL
SELECT DATE_FORMAT('2017-06-15', '%W %M %e %Y');
> 'Thursday June 15 2017'
```

CASE

Lo utilizamos para evaluar condiciones y devolver la primera que se cumpla.

```SQL
SELECT id, title, rating
  CASE
    WHEN rating < 4 THEN 'Mala'
    WHEN rating < 6 THEN 'Regular'
    WHEN rating < 8 THEN 'Buena'
    WHEN rating < 9.5 THEN 'Muy buena'
    ELSE 'Excelente'
  END AS rating_categories
FROM movies
ORDER BY rating
```

Estructura de una QUERY



BUENA SUERTE A TODOS!