



SQL



Explicaciones con gatitos

GUÍA



Intro

Hola, soy gatito Sqly, junto con otros amigos vamos a enseñarte SQL, esperamos todo esto te ayude a entender las bases



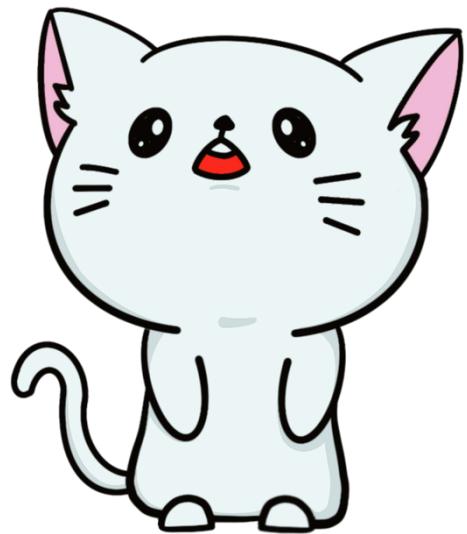
Antes de iniciar



Antes que nada, decirte que son explicaciones, SQL es algo complejo, pero lo haremos de la manera más simple posible.



General



Antes de irnos a explicar algunas cuestiones de SQL, primero decirte que SQL es un lenguaje de consulta estructurado que nos permite manipular bases de datos tipo relacional.

Las BD relacionales son como si tuviéramos una cajita y queremos almacenar y dar acceso a puntos de datos relacionados entre sí.



Puntos importantes



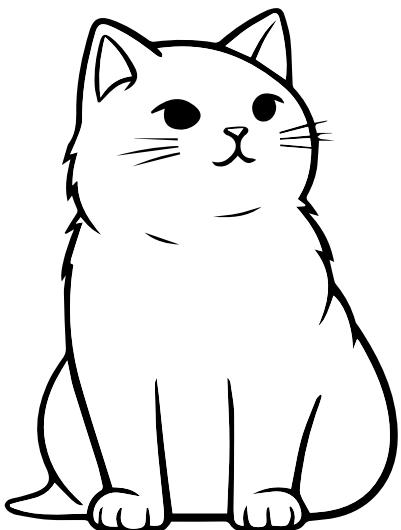
SQL nos permite recuperar información, podemos insertar, actualizar, borrar o crear tablas, procedimientos, etc.

SQL nos permite visualizar datos, proteger nuestros datos de fallos.

Puedes entender una base de datos como si tuvieras muchas tablas (colecciones) y quieras que tu información esté estructurada.



Consulta



Una consulta nos permite extraer o manipular información de una BD y nos permite especificar cómo se verá nuestra tabla. Podríamos decir que una consulta se compone de sentencias.



Comandos SQL



Sé que mi amigo te pudo confundir un poco con lo de las consultas, pero eso lo revisaremos más adelante, por eso, te vamos a explicar algunos comandos SQL y para que se utilizan y algunos ejemplos. Eso te ayudará a entender un poco las bases.



INSERT



Imagina que estás en el espacio y dices:
"Necesitamos primero crear información
en nuestro espacio e insertar las filas de
nuestras cajas" ,en este caso queremos
crear filas en una tabla de nuestra BD
Entonces cuando utilizas **INSERT**
Por lo tanto es como crear filas donde
registrarás toda la información que
lanzará tu nave

 SQL ▾



```
1 INSERT INTO gatitos (nombre, color_pelaje) VALUES ('Garfield', 'naranja')
2 INSERT INTO gatitos VALUES ('Sqly', 'gris', 2)
3
```



SELECT



Imagina que ya creaste tu espacio y dices:
“Necesitamos seleccionar la información de nuestro espacio”.
Entonces cuando utilizas **SELECT**



SQL ▾



```
1 SELECT * FROM universe |
```



UPDATE



Imagina que estás en el espacio y dices:
“Necesitamos actualizar la edad de unos
nuestros gatitos para saber si vamos a
dejarlo que vaya al espacio” .
Entonces cuando utilizas **UPDATE**, es
como si quisieras actualizar a un gatito
de una de tus tablas de tu BD



SQL ▾



```
1 UPDATE gatitos
2 SET edad = 19
3 WHERE nombre = 'Squely';
4
-
```



DELETE



Imagina que estás en el espacio y ahora necesitas eliminar a uno de los gatitos porque es muy pequeño, no cumple la condición de gatitos => 18 años.

Entonces **DELETE** va a eliminar a nuestro gatito Mittens de la fila de gatitos.



SQL ▾



```
1 DELETE FROM gatitos
2 WHERE nombre = 'Mittens';
3
```



CRUD



Te preguntarás porqué vimos INSERT, SELECT, UPDATE, DELETE.

Bueno es simple, es importante que conozcas sobre CRUD.

Create

Read

Update

Delete

Es muy útil saber que create nos permite crear nuevos datos, read leer los datos existentes, update actualizarlos, y delete eliminarlos.



TIPOS DE SGBD

Tenemos diversas opciones como MySQL, Oracle, y Microsoft Access.



ORACLE



Tipos de comandos



Existen los comandos DDL, DQL, DML, DCL, TCL, te los voy a explicar de la manera más simple.

DDL: Se conocen como Data Definition Language, son los que definen la estructura de nuestra BD. Algunos comandos son: Create, Alter, Drop, Truncate, Rename, Comment, etc.

DQL: Se conocen como Data Query Language que nos permiten consultar dentro de nuestra estructura de BD. Un ejemplo es SELECT



Tipos de comandos 2



DML: Data Manipulation Language nos permite la edición de datos en nuestra BD. Podemos encontrar Insert, Delete, Update, etc.

DCL: Son los que nos permite la gestión de permisos de acceso a nuestra BD. Podemos encontrar "Grant (Privilegios)" o "Revoke" (Quitar acceso).

TCL: Nos permite la transacción en nuestra BD.

Por ejemplo commit nos permite efectuarla, y Rollback revertir



Consultas



Existen diversas consultas que podemos aplicar como: All/Distinct, AS, FROM, WHERE, LIKE, BETWEEN, IN, AND, NOT, ORDER BY, ASC/DESC, etc.

Pero eso lo vamos a explicar más adelante.

Esperamos que esta introducción te haya quedado clara.



Empecemos

Vamos a explicarte más a detalle
SELECT, FROM, WHERE Y AND



SELECT/FROM



SELECT es una sentencia básica que nos permite obtener datos de nuestra BD. Cualquier base de datos contienen datos estructurados en tablas que contienen columnas y filas. Entonces SELECT permite elegir las columnas de nuestra tabla o filtrar filas. Te darémos un ejemplo con el nombre de nuestros amigos.



Ejemplo S & F



Imagina que estamos en el espacio, y queremos seleccionar todo lo que están en él, así que utilizamos * para elegir y universe para indicar desde donde vamos a sacar la información



Otro Ejemplo



Imagina que estamos en el espacio, y queremos seleccionar a un gatito llamado “Mittens” entonces colocamos a nuestro amigo y diremos que será desde el espacio para encontrarlo

```
SQLite

1 SELECT "Mittens" FROM universe;
```



WHERE



En WHERE utilizamos tal cláusula para filtrar datos en una consulta, donde especificamos las condiciones que se deben cumplir para que la fila se incluya en el resultado. Te daré un ejemplo simple



SQL ▾



```
1 SELECT *
2 FROM Universe
3 WHERE gatitos <= 55 |
```



Explicación



Te lo explicaré, digamos que seleccionamos todo el espacio, pero queremos saber si hay en una parte de ese espacio una cantidad menor o igual a 55 gatitos, de ser así se van a mostar los datos sobre qué parte del universo tiene esos 55 gatitos



SQL ▾



```
1 SELECT *
2 FROM Universe
3 WHERE gatitos <= 55 |
```



AND



Cuando utilizamos AND queremos filtrar 2 resultados con 2 condiciones. AND te mostrará los datos si se cumplen exactamente las 2

```
SQLite

1 SELECT * FROM universe;
2 WHERE nombre = 'Mittens'
3 AND apellido = 'Fugaz'
```



Explicación



Imagina que seleccionamos todo el universo, queremos que encuentre a Mittens a través de su nombre y apellido, así que si está su nombre y apellido tal cual, lo veremos en la tabla de nuestra BD

SQLite

```
1 SELECT * FROM universe;  
2 WHERE nombre = 'Mittens'  
3 AND apellido = 'Fugaz'
```



OrderBy



Imagina que seleccionamos todo el universo, queremos que nuestros gatitos se ordenen el número de gatitos que existen, de mayor a menor, así que te daré un ejemplo

```
SQLite

1 SELECT *
2 FROM universe
3 WHERE gatitos <= 55
4 ORDER BY gatitos DESC;
5
```



Explicación



Imagina que estas en el universo, y lo seleccionas por completo, especificas que busque menos o igual a 55 gatitos, ya que quieres saber la cantidad que hay, entonces le dices que ordene los datos de manera descendente, de mayor a menor

SQLite

```
1 SELECT *
2 FROM universe
3 WHERE gatitos <= 55
4 ORDER BY gatitos DESC;
5
```

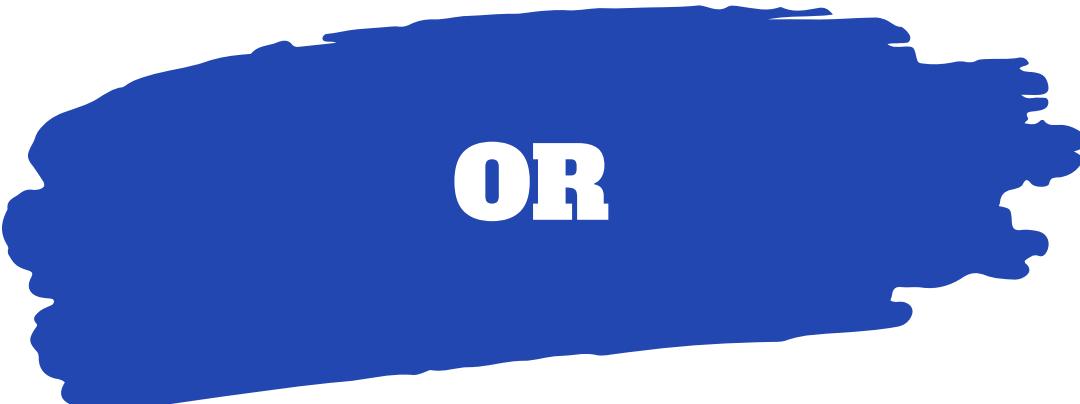


Ascendente y descendente



Te preguntarás que ascendente y descendente, verás ascendente no es necesario especificarlo para ordenar tus tablas SQL ya que por defecto se suele ordenar de menor a mayor (si son números) y descendente se especifica con DESC para ordenar de mayor a menor (si son números). El orden depende del tipo de datos que utilizas. Puedes ordenar números, nombres, etc.





OR



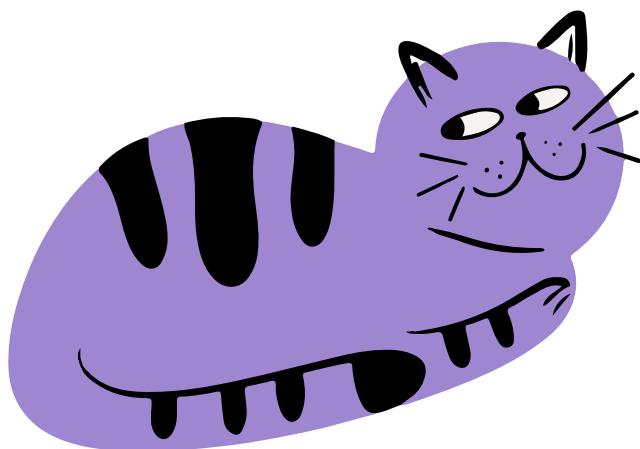
Cuando utilizamos OR estamos especificando que queremos que una de las condiciones que cumplan. Puede cumplirse la primera o segunda condición.

SQLite

```
1 SELECT * FROM gatitos WHERE color = 'gris' OR color = 'blanco';
2
3
```



Explicación



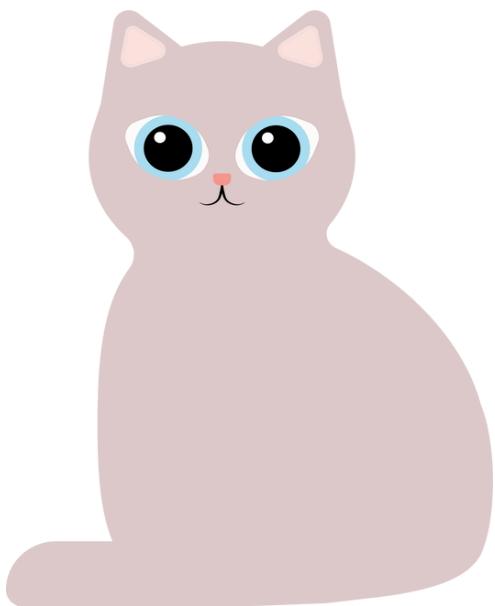
Cuando utilizamos `OR` queremos que una de las condiciones se cumplan en el siguiente ejemplo puedes ver que queremos obtener el dato de color de nuestros amigos gatitos, si una de las condiciones se cumple podemos obtener el dato

SQLite

```
1 SELECT * FROM gatitos WHERE color = 'gris' OR color = 'blanco';
2
3
```



2 columnas x tabla



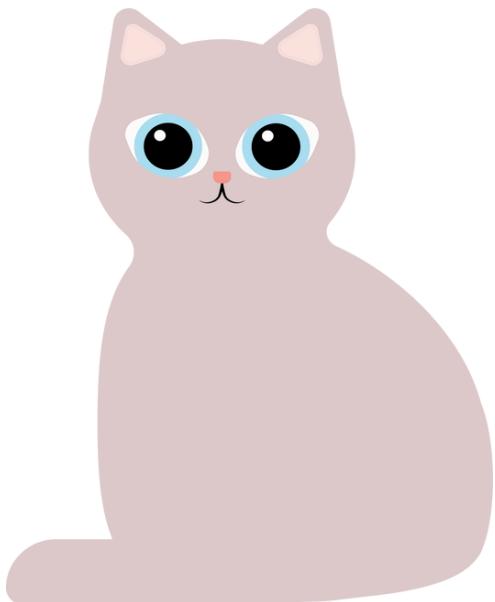
Hemos visto el uso de * donde elegimos todo el universo, pero también tenemos la opción de elegir dos universos (columnas) de una tabla, así que te mostraré

SQLite

```
1 SELECT gatitos_name,  
2      gatitos_lastname  
3 FROM universe;  
4  
5
```



Ejemplo



Como puedes ver en el ejemplo anterior, queremos seleccionar el nombre y apellido de los gatitos, aquí estamos especificando que queremos obtener dos datos de nuestro universo (tabla)

SQLite

```
1 SELECT gatitos_name,  
2      gatitos_lastname  
3 FROM universe;  
4  
5
```



NOT



El NOT invierte el valor de cualquier expresión tipo booleana. Digamos que es como negar una expresión.

```
< Input
SELECT name, age
FROM gatitos
WHERE NOT age <13
```



Ejemplo



Como puedes ver en el ejemplo anterior, estamos seleccionando el nombre y la edad de nuestros gatitos desde nuestra tabla gatitos. Luego, utilizamos la cláusula WHERE para aplicar un filtro a los resultados. Dentro de esta cláusula, usamos NOT para negar la condición que sigue. En este caso, la condición es age < 13, lo que significa que estamos buscando gatitos cuya edad no sea menor que 13 años.

Input

```
SELECT name, age
FROM gatitos
WHERE NOT age <13
```



NOT part 2



También puedes utilizar WHERE NOT así como AND NOT, imagina que te dicen que debes hacer la cultura así para que no lances al espacio a esos gatitos, así que te va un ejemplo:

```
< Input
SELECT name, age,color
FROM gatitos
WHERE NOT age <13
AND NOT color = 'pink';
```



Ejemplo 2

Como puedes ver en el ejemplo anterior, estamos seleccionando el nombre, edad y color de nuestros gatitos de nuestra tabla gatitos. Utilizamos la cláusula WHERE y usamos NOT para negar 2 condiciones. La primera es que solo seleccionamos gatitos cuya edad no sea mayor que 13 y la segunda que el color de los gatitos no sea rosa.



< Input

```
SELECT name, age,color
FROM gatitos
WHERE NOT age <13
AND NOT color = 'pink';
```



General



Hoy queremos explicarte un ejemplo, con lo que hemos visto como WHERE, NOT, ORDER BY, y DESC, para que te quede más claro y podamos aplicar todo lo visto en un ejemplo, antes de ver más sobre SQL

Input

```
SELECT nombre, edad, color
FROM gatitos
WHERE edad > 5
AND NOT (color = 'blanco' OR color = 'negro')
ORDER BY edad DESC;
```



Explicación



En el ejemplo anterior seleccionamos el nombre, edad, color de nuestros amigos gatitos de nuestra tabla gatitos. Utilizamos la cláusula WHERE donde encontramos la condición de que la edad tengan más de 5 años de edad. Sin embargo utilizamos AND NOT para excluir a los gatitos que tengan color blanco y negro, y con ORDER BY , estamos ordenando los resultados de forma descendente, donde ordenamos de mayor a menor edad

Input

```
SELECT nombre, edad, color
FROM gatitos
WHERE edad > 5
AND NOT (color = 'blanco' OR color = 'negro')
ORDER BY edad DESC;
```



OPERADORES LÓGICOS con SUBCONSULTAS



Existen operadores lógicos más complejos que te permiten subconsultas, como ALL, ANY/SOME, BETWEEN, EXISTS, IN, LIKE, que iremos revisando y explicandote



ALL



Empezaremos con la subconsulta ALL, como ves en el ejemplo utilizamos tal subconsulta y te la voy a explicar a continuación

SQLite

```
1 SELECT name, age
2 FROM gatitos
3 WHERE age > ALL (SELECT edad FROM edades_gatitos);
4
```



ALL



Puedes usar ALL para hacer esto en SQL. Funciona comparando la edad de cada gatito con todas las edades en la tabla edades_gatitos. Si la edad de un gatito es mayor que todas las edades en la tabla edades_gatitos, se selecciona ese gatito.

SQLite

```
1 SELECT name, age
2 FROM gatitos
3 WHERE age > ALL (SELECT edad FROM edades_gatitos);
4
```



¿Qué sigue?



Vamos a ver más subconsultas y ejemplos, también empezaremos a dejarte ejercicios para que practiques y puedas aprender en la práctica.

Y luego estaremos diseñando unas tablas para que te diviertas haciendo consultas con gatitos.

Por ahora a seguir aprendiendo



Antes de finalizar

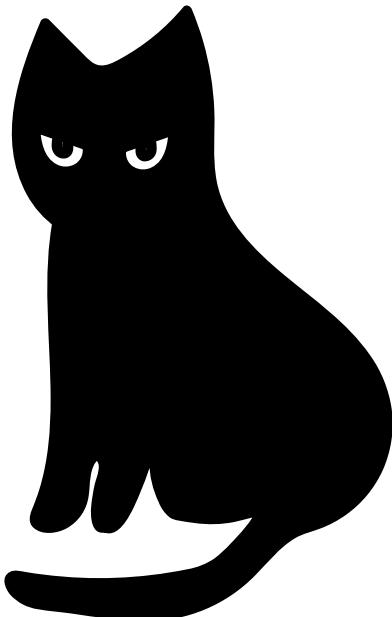


Vamos a seguir aprendiendo sobre SQL, esperemos que al final todo te sea claro. Una vez por semana estarás con nosotros donde te explicaremos consultas de manera sencilla o algún dato de SQL. Gracias por tu apoyo. Te agradecemos si te quedas con nosotros.
Hasta pronto.





Próxima Edición



En la próxima edición veremos
más ejemplos y teoría sobre
SQL.
Hasta la próxima.
Miau.



Gracias por leer

Si te gusto el contenido, agradecería mucho si pudieses compartirlo.

Espera las siguientes actualizaciones.

