Arquitectura y Administración de bases de datos con SQL 2021

Héctor Manuel Garduño Castañeda

Diciembre, 2021

scidata

Contenido

Ya hemos tenido oportunidad de conocer el comando **LIKE**. Recordemos que tenemos dos comodines: % y _. El % permite que *entre sus extremos haya cualquier cantidad de caracteres, incluyendo espacios*, y _, que permite que *entre sus extremos haya un solo caracter*.

Por ejemplo:
A% significa que busque textos que comiencen con A, como 'ABC' o 'ABCDE'.
%A significa que busque textos que terminen con A, como 'BCA' o 'BCDE A'.
A%B significa que busque textos que comiencen con A y terminen con B, como 'AC HB' o 'AB'.

AB_C significa que busque textos que comiencen con AB, luego siga un único caracter, y luego una C, como 'AB C' o 'ABDC'.

NOTA. El operador **LIKE** busca coincidencias exactas. Esto significa que diferencía entre mayúsculas y minúsculas. Si no queremos hacer distinción, utilizamos **ILIKE**.

```
SELECT * FROM customer WHERE customer_name LIKE 'Jo%';
SELECT * FROM customer WHERE customer_name LIKE '%od%';
SELECT * FROM customer WHERE customer_name LIKE 'Jas_n%';
SELECT * FROM customer WHERE customer_name NOT LIKE
'J%';
SELECT * FROM customer WHERE customer_name LIKE 'A% ____';
```

Las *expresiones regulares* (regex) son una búsqueda de patrones por codificación. Sirven para realizar búsquedas más complejas como, por ejemplo, validar si una lista de correos está correctamente escrita, ya que un mail debe contener una secuencia de caracteres alfanuméricos incluyendo punto y guión bajo seguido de un @ seguido de una secuencia de caracteres alfanuméricos seguidos de un punto y seguido de una secuencia de caracteres alfabéticos de entre 2 y 8 caracteres.

Comodines en regex

Existen muchos comodines. Aquí mostraremos los más importantes y útiles.

$\|$	Es el operador 'o'
$*$	Denota repetición de la secuencia previa 0 o mas veces
$+$	Denota repetición de la secuencia previa 1 o mas veces
$?$	Denota repetición de la secuencia previa 0 o una vez
$\{m\}$	Denota repetición de la secuencia previa m veces
$\{m,\}$	Denota repetición de la secuencia previa m o más veces
$\{m,n\}$	Denota repetición de la secuencia previa al menos m pero no más de n veces
$\wedge$,\$	Denotan inicio y final del texto
$[texto]$	Una expresión de caracteres. Denota coincidencia con con cualquier elemento dentro de los corchetes
\s	Denota espacio en blano
$\sim$,*	Denotan caso sensitivo o insensitivo a mayúsculas

Ejercicios

https://regex101.com/

https://regex101.com/r/geGfa/1

para practicar expresiones en regex

Traducir qué significan los siguienes símbolos:

- ∼*'∧a+[a-z\s]*$'
- ∼*'∧(a|b|c|d)+[a-z\s]+$'
- ∼*'∧(a|b|c|d)[a-z]{3}\s[a-z]{4}$'
- ∼*'[a-z0-9\.\ − _]+@[a-z0-9\−]+\.[a-z]{2,5}'

+ Explicación de cada una en el minuto 46:27 de la Sesión 10

```
SELECT * FROM customer
WHERE customer_name ~*'^a+[a-z\s]*$'
```
+ Personas que empiecen con A

```
SELECT * FROM customer
WHERE customer_name ~*'^(a|b|c|d)+[a-z\s]+$'
```
+ Personas que empiecen con a,b,c o d

```
SELECT * FROM customer
WHERE customer_name ^(a|b|c|d)[a-z]{3}\s[a-z]{4}$'
```
+ Esta sintaxis está mal, está bien escrita en el script supermercado_clase10.sql

```
SELECT * FROM users
WHERE name ~*'^[a-z0-9\.\ - \_]+@[a-z0-9\-]+\.[a-z]{2,5}$'
```

~ se hace con la combinación de teclas alt gr + +

Nota:
create table users(id serial primary key, name character varying);
+ Cuando ponemos id serial estamos diciendo que ponga los números consecutivamente