Bases de Datos 2022

Consultas con Like

Vamos a realizar consultas que no son exactas, quiere decir que no se utilizan los operadores ‘=’ ó ‘<>’.

Esto se puede utilizar por ejemplo para buscar todos los apellidos que contengan las letras ‘un’.

Por ejemplo:

SELECT first_name FROM employee WHERE first_name like 'Sh%';

¿qué devuelve esta sentencia?

Vemos aparece el operador %, ¿para qué sirve?

Manos a la obra

En primer lugar vamos a buscar la documentación de postgresql, así como los sitios de ejemplos compartidos en el classroom para ver como se realizan las búsquedas por patrones.

Elaborar un documento que contenga:

1) La descripción del operador **like**, posibles usos, comodines, tipos de datos sobre los que trabaja, cómo hacer para trabajar con tipos de datos que no son string y todo otro comentario o cita que les parezca de interés.

   PostgreSQL proporciona tres enfoques separados para la coincidencia de patrones: el operador SQL LIKE tradicional, el operador SIMILAR TO más reciente (agregado en SQL: 1999) y las expresiones regulares de estilo POSIX. Aparte del básico "¿coincide esta cadena con este patrón?" operadores, las funciones están disponibles para extraer o reemplazar subcadenas coincidentes y para dividir una cadena en ubicaciones coincidentes.

   Tip

   Si tiene necesidades de coincidencia de patrones que van más allá de esto, considere escribir una función definida por el usuario en Perl o Tcl.

   Precaución

   Si bien la mayoría de las búsquedas de expresiones regulares se pueden ejecutar muy

rápidamente, se pueden idear expresiones regulares que toman cantidades arbitrarias de tiempo y memoria para procesar. Tenga cuidado al aceptar patrones de búsqueda de expresiones regulares de fuentes hostiles. Si debe hacerlo, es aconsejable imponer un tiempo de espera de declaración.

Las búsquedas que usan patrones SIMILAR TO tienen los mismos riesgos de seguridad, ya que SIMILAR TO proporciona muchas de las mismas capacidades que las expresiones regulares de estilo POSIX.
Las búsquedas LIKE, al ser mucho más simples que las otras dos opciones, son más seguras de usar con fuentes de patrones posiblemente hostiles.

Los operadores de coincidencia de patrones de los tres tipos no admiten intercalaciones no deterministas. Si es necesario, aplique una intercalación diferente a la expresión para evitar esta limitación.

```
string LIKE pattern [ESCAPE escape-character]
string NOT LIKE pattern [ESCAPE escape-character]
```

La expresión LIKE devuelve verdadero si la cadena coincide con el patrón proporcionado. (Como era de esperar, la expresión NOT LIKE devuelve falso si LIKE devuelve verdadero, y viceversa. Una expresión equivalente es NOT (patrón de cadena LIKE).)

Si el patrón no contiene signos de porcentaje o guiones bajos, entonces el patrón solo representa la cadena en sí; en ese caso, LIKE actúa como el operador igual. Un guión bajo (_) en el patrón representa (coincide) con cualquier carácter único; un signo de porcentaje (%) coincide con cualquier secuencia de cero o más caracteres.
Algunos ejemplos

```
'abc' LIKE 'abc'    true
'abc' LIKE 'a%'     true
'abc' LIKE '_b_'    true
'abc' LIKE 'c'      false
```

La coincidencia de patrones LIKE siempre cubre toda la cadena. Por lo tanto, si se desea

hacer coincidir una secuencia en cualquier lugar dentro de una cadena, el patrón debe comenzar y terminar con un signo de porcentaje.

Para hacer coincidir un guión bajo literal o un signo de porcentaje sin hacer coincidir otros caracteres, el carácter respectivo en el patrón debe estar precedido por el carácter de escape. El carácter de escape predeterminado es la barra invertida, pero se puede seleccionar uno diferente mediante la cláusula ESCAPE. Para hacer coincidir el propio carácter de escape, escriba dos caracteres de escape.

SIMILAR TO Regular Expressions

El operador SIMILAR A devuelve verdadero o falso dependiendo de si su patrón coincide con la cadena dada. Es similar a LIKE, excepto que interpreta el patrón utilizando la definición estándar de SQL de una expresión regular. Las expresiones regulares de SQL son un cruce curioso entre la notación LIKE y la notación de expresión regular común (POSIX).

Al igual que LIKE, el operador SIMILAR TO tiene éxito solo si su patrón coincide con la cadena completa; esto es diferente al comportamiento común de las expresiones regulares donde el patrón puede coincidir con cualquier parte de la cadena. También como LIKE, SIMILAR TO usa _ y % como caracteres comodín que denotan cualquier carácter individual y cualquier cadena, respectivamente (estos son comparables a . y .* en las expresiones regulares POSIX).

Además de estas funciones prestadas de LIKE, SIMILAR TO admite estos metacaracteres de coincidencia de patrones tomados de expresiones regulares POSIX:

| denota alternancia (cualquiera de dos alternativas).

* denota repetición del elemento anterior cero o más veces.

+ denota repetición del elemento anterior una o más veces.

? denota repetición del elemento anterior cero o una vez.

{m} denota la repetición del elemento anterior exactamente m veces.

{m,} denota la repetición del elemento anterior m o más veces.

{m,n} denota la repetición del elemento anterior al menos m y no más de n veces.

Los paréntesis () se pueden utilizar para agrupar elementos en un único elemento lógico.

Una expresión de paréntesis [...] especifica una clase de carácter, al igual que en las expresiones regulares POSIX.

2) Buscar alternativas al operador **like** en sql.
La otra alternativa al like es usar los operadores que usamos siempre en conjunto, como el igual que, mayor que, menor que, diferente que junto a operadores lógicos puesto a que es probable que tengamos que buscar entre ciertos elementos

3) En el pgadmin en un base de datos, ejecute el script bd-ejemplo.sql, que contiene tres tablas:

```
CREATE TABLE "autor" (
    "idautor" smallint NOT NULL,
    "autor" character varying(32) NOT NULL,
    Constraint "autor_pkey" Primary Key ("idautor")
);

CREATE TABLE "editorial" (
    "editorial" character varying(30) NOT NULL,
    "direccion" character varying(40) NOT NULL,
    "telefono" character varying(32) NOT NULL,
    Constraint "editorial_pkey" Primary Key ("editorial")
);



CREATE TABLE "libro" (
    "isbn" character(10) NOT NULL,
    "titulo" character varying(80) NOT NULL,
    "fpublicacion" date NOT NULL,
    "paginas" smallint NOT NULL,
    "editorial" character varying(32) NOT NULL,
    Constraint "libro_pkey" Primary Key ("isbn"),
    foreign key (editorial) references editorial deferrable
);
```

4) Realizar al menos cuatro consultas distintas utilizando el operador like y algunos de los comodines, al menos una de la consulta debe ser sobre un dato que no sea carácter.

Todos los libros que comienzen con A
```
SELECT * FROM libro WHERE libro.titulo LIKE 'A%'
```

Todos los libros cuyas editoriales terminen con ll o contengan ARGENTINA como palabra

en el titulo

```
SELECT * FROM libro
INNER JOIN editorial ON editorial.editorial = libro.editorial
WHERE libro.editorial LIKE '%ll' OR libro.titulo LIKE '%ARGENTINA%';
```

Todos los libros que contengan el numero 3 como numero inicial de cantidad de paginas

```
SELECT * FROM libro WHERE  CAST(libro.paginas AS TEXT) LIKE '3%';
```

Todos los libros que no sean del 1996-04-07

```
SELECT * FROM libro WHERE  CAST(libro.fpublicacion AS TEXT) NOT LIKE '1996-04-07';
```