Bases de Datos I
28/10/2020
Apuntes de la clase 28/10
IS-501
José Inestroza

El método JSON_VALUE, el cual en MySQL sería equivalente a ->>, o a JSON_UNQUOTE en combinación con JSON_EXTRACT, El rango entre **[0-9]** es lo mismo que el meta carácter **\d**. Existen distintos meta caracteres que son caracteres especiales que se usan para buscar patrones.

Expresiones regulares:
Clases de caracteres

1. **Conjunto de caracteres [aeiou]** → Coincide con cualquier carácter del conjunto.

2. **Conjunto de caracteres negados [^aeiou]** → Coincide con cualquier carácter que no esté en el conjunto.

3. **Rango [g-s]** → Coincide con un carácter que tiene un código de carácter entre los dos caracteres especificados es inclusivo.

4. **Punto .** → Coincide con cualquier carácter excepto los saltos de línea. Equivalente a [^ \ n \ r].

5. **Coincidir con cualquiera** [\s\S] → Un conjunto de caracteres que se puede utilizar para hacer coincidir cualquier carácter, incluidos los saltos de línea, sin las banderas dotall. Una alternativa es [^], pero no es compatible con todos los navegadores.

6. **Palabra \w** → Coincide con cualquier carácter de palabra (alfanumérico y subrayado). Solo coincide con caracteres de bajo ascii (sin acentos ni caracteres no romanos). Equivalente a [A-Za-z0-9_].

7. **No palabra \W** → Coincide con cualquier carácter que no sea un carácter de palabra (alfanumérico y subrayado). Equivalente a [^ A-Za-z0-9_].

8. **Digito \d** → Coincide con cualquier carácter de dígito (0-9). Equivalente a [0-9].

9. **No digito \D** → Coincide con cualquier carácter que no sea un carácter de dígito (0-9). Equivalente a [^ 0-9].

10. **Espacio en blanco \s** → Coincide con cualquier carácter de espacio en blanco (espacios, tabulaciones, saltos de línea).

11. **No espacio en blanco \S** → Coincide con cualquier carácter que no sea un carácter de espacio en blanco (espacios, tabulaciones, saltos de línea).

12. **Inicio o Cabeza \^w+** → Coincide con el comienzo de la cadena, o el comienzo de una línea si la bandera multilínea (m) está habilitada. Esto coincide con una posición, no con un personaje.

13. **Final o Cola \w+$** → Coincide con el final de la cadena, o el final de una línea si la bandera multilínea (m) está habilitada. Esto coincide con una posición, no con un personaje.

14. **Límite de palabra s\b** → Coincide con la posición del límite o posición final de una palabra entre un carácter de palabra y un carácter o posición que no es de palabra (inicio / final de cadena). Consulte la palabra clase de caracteres (w) para obtener más información.

15. **No un límite de palabra s\B** → Coincide con cualquier posición que no sea un límite de palabras. Esto coincide con una posición, no con un caracter.

16. **Caracteres reservados \+** → El siguiente carácter tiene un significado especial y debe ir precedido de una \ (barra invertida) para representar un carácter literal:+ *? ^ $ \. [] {} () | /
    Dentro de un conjunto de caracteres, solo es necesario escapar de \, - y].

17. **Carácter de escape octal \000** → El valor debe ser menor que 255 (\ 377).
    MATCH : ©

18. **Carácter de escape hexadecimal xFF** → MATCH : ©.

19. **Tab \t** → Coincide con un carácter TAB (código de carácter 9).

20. **Salto de línea \n** → Coincide con un carácter LINE FEED (código de carácter 10).

21. **Tab vertical \v** → Coincide con un carácter TAB VERTICAL (código de carácter 11).

22. **Null \0** → Coincide con un carácter NULL (código de carácter 0).

23. **Grupo de captura (ABC)** → Agrupa varios tokens y crea un grupo de captura para extraer una subcadena o usar una referencia inversa.

24. **Nombre del grupo de captura (?<name>ABC)** → Crea un grupo de captura al que se puede hacer referencia mediante el nombre especificado.

25. **Referencias numéricas \1** → Coincide con los resultados de un grupo de captura. Por ejemplo, \ 1 coincide con los resultados del primer grupo de captura y \ 3 coincide con el tercero.

26. **Grupos no capturables (?:ABC)** → Agrupa varios tokens sin crear un grupo de captura.

27. **Anticipación positiva (?=ABC)** → Coincide con un grupo después de la expresión principal sin incluirlo en el resultado. \d(?=px)

28. **Anticipación negativa (?!ABC)** → Especifica un grupo que no puede coincidir después de la expresión principal (si coincide, el resultado se descarta).
29. **Anticipación positiva hacia atrás (?<=ABC)** → Coincide con un grupo antes de la expresión principal sin incluirlo en el resultado.

30. **Anticipación negativa hacia atrás (?<!ABC) →** Especifica un grupo que no puede coincidir antes de la expresión principal (si coincide, el resultado se descarta).

31. **Plus o Más + →** Coincide con 1 o más de los tokens anteriores.

32. **Star o Cruz * →** Coincide con 0 o más del token anterior.

33. **Cuantificador {1,3} →** Coincide con la cantidad especificada del token anterior. {1,3} coincidirá con 1 a 3. {3} coincidirá exactamente con 3. {3,} coincidirá con 3 o más.

34. **Opcional ? →** Coincide con 0 o 1 del token anterior, lo que lo hace opcional.

35. **Alternación | →** Actúa como un OR booleano. Coincide con la expresión anterior o posterior a |. Puede operar dentro de un grupo o en una expresión completa. Los patrones se probarán en orden.

Expresiones regulares en Mariadb y MySQL
En muchos casos, la simple coincidencia de patrones proporcionada por **LIKE** es suficiente. **LIKE** realiza dos tipos de partidos:

1. **- El subrayado, que coincide con un solo carácter.**
2. **% El signo de porcentaje, que coincide con cualquier número de caracteres.**

Las coincidencias de expresiones regulares se realizan con la función REGEXP. RLIKE es sinónimo de REGEXP.

Sin caracteres especiales, una coincidencia de expresión regular es verdadera si los caracteres coinciden. La coincidencia no distingue entre mayúsculas y minúsculas, excepto en el caso de cadenas BINARIAS.

```
SELECT BINARY 'Maria' REGEXP 'maria';
+-------------------------------+
| BINARY 'Maria' REGEXP 'maria' |
+-------------------------------+
|                             0 |
+-------------------------------+
```

Tenga en cuenta que la palabra coincidente debe coincidir con el patrón completo:

```
SELECT 'Maria' REGEXP 'Mari';
+-----------------------+
| 'Maria' REGEXP 'Mari' |
+-----------------------+
|                     1 |
+-----------------------+

SELECT 'Mari' REGEXP 'Maria';
+-----------------------+
| 'Mari' REGEXP 'Maria' |
+-----------------------+
|                     0 |
+-----------------------+
```

El primero devuelve verdadero porque el patrón "Mari" existe en la expresión "Maria". Cuando se invierte el orden, el resultado es falso, ya que el patrón "María" no existe en la expresión "Mari"

Se puede realizar una coincidencia con más de una palabra con el | personaje. Por ejemplo:

```
SELECT 'Maria' REGEXP 'Monty|Maria';
+-------------------------------+
| 'Maria' REGEXP 'Monty|Maria' |
+-------------------------------+
|                             1 |
+-------------------------------+
```

Caracteres especiales

^ → coincide con el comienzo de una cadena (dentro de los corchetes también puede significar NO):

```
SELECT 'Maria' REGEXP '^Ma';
+-----------------------+
| 'Maria' REGEXP '^Ma' |
+-----------------------+
|                     1 |
+-----------------------+
```

$ → coincide con el final de una cadena:

```
SELECT 'Maria' REGEXP 'ia$';
+-----------------------+
| 'Maria' REGEXP 'ia$' |
+-----------------------+
|                     1 |
+-----------------------+
```

. → coincide con cualquier carácter individual:

```
SELECT 'Maria' REGEXP 'Ma.ia';
+-------------------------+
| 'Maria' REGEXP 'Ma.ia' |
+-------------------------+
|                       1 |
+-------------------------+

SELECT 'Maria' REGEXP 'Ma..ia';
+--------------------------+
| 'Maria' REGEXP 'Ma..ia' |
+--------------------------+
|                        0 |
+--------------------------+
```

***** → coincide con cero o más de un carácter x. En los ejemplos siguientes, es el carácter.

```
SELECT 'Maria' REGEXP 'Mar*ia';
+-------------------------+
| 'Maria' REGEXP 'Mar*ia' |
+-------------------------+
|                       1 |
+-------------------------+

SELECT 'Maia' REGEXP 'Mar*ia';
+------------------------+
| 'Maia' REGEXP 'Mar*ia' |
+------------------------+
|                      1 |
+------------------------+

SELECT 'Marrria' REGEXP 'Mar*ia';
+---------------------------+
| 'Marrria' REGEXP 'Mar*ia' |
+---------------------------+
|                         1 |
+---------------------------+
```

+ → coincide con uno o más caracteres x. En los ejemplos siguientes, es el carácter r.

```
SELECT 'Maria' REGEXP 'Mar+ia';
+-------------------------+
| 'Maria' REGEXP 'Mar+ia' |
+-------------------------+
|                       1 |
+-------------------------+

SELECT 'Maia' REGEXP 'Mar+ia';
+------------------------+
| 'Maia' REGEXP 'Mar+ia' |
+------------------------+
|                      0 |
+------------------------+

SELECT 'Marrria' REGEXP 'Mar+ia';
+---------------------------+
| 'Marrria' REGEXP 'Mar+ia' |
+---------------------------+
|                         1 |
+---------------------------+
```

? → coincide con cero o uno de un carácter x. En los ejemplos siguientes, es el carácter r.

```
SELECT 'Maria' REGEXP 'Mar?ia';
+-------------------------+
| 'Maria' REGEXP 'Mar?ia' |
+-------------------------+
|                       1 |
+-------------------------+

SELECT 'Maia' REGEXP 'Mar?ia';
+------------------------+
| 'Maia' REGEXP 'Mar?ia' |
+------------------------+
|                      1 |
+------------------------+

SELECT 'Marrria' REGEXP 'Mar?ia';
+---------------------------+
| 'Marrria' REGEXP 'Mar?ia' |
+---------------------------+
|                         0 |
+---------------------------+
```

() → combina una secuencia, por ejemplo (xyz) + o (xyz) *

```
SELECT 'Maria' REGEXP '(ari)+';
+-------------------------+
| 'Maria' REGEXP '(ari)+' |
+-------------------------+
|                       1 |
+-------------------------+
```

{} → x{n} y esta notación se utiliza para hacer coincidir muchas instancias de la . En el caso del partido debe ser exactamente que muchas veces. En el caso de , la coincidencia puede ocurrir desde tiempos. Por ejemplo, para hacer coincidir cero o una instancia de la cadena (que es idéntica a), se puede utilizar lo siguiente:x{m,n}xx{n}x{m,n}mnari(ari)?

```
SELECT 'Maria' REGEXP '(ari){0,1}';
+-----------------------------+
| 'Maria' REGEXP '(ari){0,1}' |
+-----------------------------+
|                           1 |
+-----------------------------+
```

[] → [xy] agrupa los caracteres para fines coincidentes. Por ejemplo, para que coincida con el carácter o el:pr

```
SELECT 'Maria' REGEXP 'Ma[pr]ia';
+--------------------------+
| 'Maria' REGEXP 'Ma[pr]ia' |
+--------------------------+
|                        1 |
+--------------------------+
```

Los corchetes también permiten que una coincidencia de rango, por ejemplo, coincida con cualquier carácter de a-z, se utiliza. También se permiten rangos numéricos.[a-z]

```
SELECT 'Maria' REGEXP 'Ma[a-z]ia';
+----------------------------+
| 'Maria' REGEXP 'Ma[a-z]ia' |
+----------------------------+
|                          1 |
+----------------------------+
```

Lo siguiente no coincide, ya que queda fuera del intervalo .ra-p

```
SELECT 'Maria' REGEXP 'Ma[a-p]ia';
+----------------------------+
| 'Maria' REGEXP 'Ma[a-p]ia' |
+----------------------------+
|                          0 |
+----------------------------+
```

^ → El carácter significa que coincide, por ejemplo:^NOT

```
SELECT 'Maria' REGEXP 'Ma[^p]ia';
+----------------------------+
| 'Maria' REGEXP 'Ma[^p]ia' |
+----------------------------+
|                          1 |
+----------------------------+

SELECT 'Maria' REGEXP 'Ma[^r]ia';
+----------------------------+
| 'Maria' REGEXP 'Ma[^r]ia' |
+----------------------------+
|                          0 |
+----------------------------+
```

El y los personajes por sí solos se pueden emparejar literalmente dentro de un bloque, sin escapar, siempre y cuando coincidan inmediatamente con el soporte de apertura:[ ][]

```
SELECT '[Maria' REGEXP '[[]';
+-----------------------+
| '[Maria' REGEXP '[[]' |
+-----------------------+
|                     1 |
+-----------------------+

SELECT '[Maria' REGEXP '[]]';
+-----------------------+
| '[Maria' REGEXP '[]]' |
+-----------------------+
|                     0 |
+-----------------------+
```

```
SELECT ']Maria' REGEXP '[]]';
+----------------------+
| ']Maria' REGEXP '[]]' |
+----------------------+
|                    1 |
+----------------------+

SELECT ']Maria' REGEXP '[]a]';
+-----------------------+
| ']Maria' REGEXP '[]a]' |
+-----------------------+
|                     1 |
+-----------------------+
```

Orden incorrecto, por lo que no coincide:

```
SELECT ']Maria' REGEXP '[a]]';
+-----------------------+
| ']Maria' REGEXP '[a]]' |
+-----------------------+
|                     0 |
+-----------------------+
```

El carácter también se puede emparejar de la misma manera: -

```
SELECT '-Maria' REGEXP '[1-10]';
+--------------------------+
| '-Maria' REGEXP '[1-10]' |
+--------------------------+
|                        0 |
+--------------------------+

SELECT '-Maria' REGEXP '[-1-10]';
+---------------------------+
| '-Maria' REGEXP '[-1-10]' |
+---------------------------+
|                         1 |
+---------------------------+
```

Límites de palabras

Los patrones y coinciden con el principio y el final de una palabra respectivamente. Por ejemplo::<::>:

```
SELECT 'How do I upgrade MariaDB?' REGEXP '[[:<:]]MariaDB[[:>:]]';
+-------------------------------------------------------------+
| 'How do I upgrade MariaDB?' REGEXP '[[:<:]]MariaDB[[:>:]]' |
+-------------------------------------------------------------+
|                                                           1 |
+-------------------------------------------------------------+
```

```
SELECT 'How do I upgrade MariaDB?' REGEXP '[[:<:]]Maria[[:>:]]';
+-----------------------------------------------------------+
| 'How do I upgrade MariaDB?' REGEXP '[[:<:]]Maria[[:>:]]' |
+-----------------------------------------------------------+
|                                                         0 |
+-----------------------------------------------------------+
```

Clases de caracteres
Hay una serie de accesos directos para que coincidan con clases de caracteres predefinidas. Estos coinciden con el patrón (dentro de un conjunto). Existen las siguientes clases de caracteres:[:character_class:][]

Character Class	Description
alnum	Alphanumeric
alpha	Alphabetic
blank	Whitespace
cntrl	Control characters
digit	Digits
graph	Graphic characters
lower	Lowercase alphabetic
print	Graphic or space characters
punct	Punctuation
space	Space, tab, newline, and carriage return
upper	Uppercase alphabetic
xdigit	Hexadecimal digit

```
SELECT 'Maria' REGEXP 'Mar[[:alnum:]]*';
+---------------------------------+
| 'Maria' REGEXP 'Mar[:alnum:]*' |
+---------------------------------+
|                               1 |
+---------------------------------+
```

Recuerde que las coincidencias no distinguen entre mayúsculas y minúsculas de forma predeterminada, a menos que se utilice una cadena binaria, por lo que el siguiente ejemplo, que busca específicamente una mayúscula, coincide con un carácter en minúsculas:

```
SELECT 'Mari' REGEXP 'Mar[[:upper:]]+';
+---------------------------------+
| 'Mari' REGEXP 'Mar[[:upper:]]+' |
+---------------------------------+
|                               1 |
+---------------------------------+

SELECT BINARY 'Mari' REGEXP 'Mar[[:upper:]]+';
+---------------------------------------+
| BINARY 'Mari' REGEXP 'Mar[[:upper:]]+' |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
```

Combinar

El verdadero poder de las expresiones regulares se libera cuando se combina lo anterior, para formar ejemplos más complejos. La reputación de complejidad de la expresión regular se deriva de la aparente complejidad de múltiples expresiones regulares combinadas, cuando en realidad, es simplemente una cuestión de entender los personajes y cómo se aplican:

El primer ejemplo no puede coincidir, ya que mientras que las coincidencias, o solo coincide una vez antes de los caracteres al final. Ma i r ia

```
SELECT 'Maria' REGEXP 'Ma[ir]{2}ia';
+------------------------------+
| 'Maria' REGEXP 'Ma[ir]{2}ia' |
+------------------------------+
|                            0 |
+------------------------------+
```

Este ejemplo coincide, como o coincide exactamente dos veces después de , en este caso uno y uno .irMari

```
SELECT 'Maria' REGEXP 'Ma[ir]{2}';
+----------------------------+
| 'Maria' REGEXP 'Ma[ir]{2}' |
+----------------------------+
|                          1 |
+----------------------------+
```

Escapar

Con el gran número de caracteres especiales, se debe tener cuidado para escapar correctamente de los personajes. Se requieren dos caracteres de barra diagonal invertida (uno para el analizador MariaDB, otro para la biblioteca regex), para escapar correctamente de un carácter. Por ejemplo:

Para que coincida con el literal:(Ma

```
SELECT '(Maria)' REGEXP '(Ma';
ERROR 1139 (42000): Got error 'parentheses not balanced' from regexp

SELECT '(Maria)' REGEXP '\(Ma';
ERROR 1139 (42000): Got error 'parentheses not balanced' from regexp

SELECT '(Maria)' REGEXP '\\(Ma';
+---------------------------+
| '(Maria)' REGEXP '\\(Ma' |
+---------------------------+
|                         1 |
+---------------------------+
```

Para hacer coincidir : Los dos primeros ejemplos son incorrectos, ya que coinciden con una o más veces, no:r+rr+

```
SELECT 'Mar+ia' REGEXP 'r+';
+----------------------+
| 'Mar+ia' REGEXP 'r+' |
+----------------------+
|                    1 |
+----------------------+
```

```
SELECT 'Maria' REGEXP 'r+';
+---------------------+
| 'Maria' REGEXP 'r+' |
+---------------------+
|                   1 |
+---------------------+

SELECT 'Maria' REGEXP 'r\\+';
+-----------------------+
| 'Maria' REGEXP 'r\\+' |
+-----------------------+
|                     0 |
+-----------------------+

SELECT 'Maria' REGEXP 'r+';
+---------------------+
| 'Maria' REGEXP 'r+' |
+---------------------+
|                   1 |
+---------------------+
```

Regexp

Sintaxis

```
expr REGEXP pat, expr RLIKE pat
```

Realiza una coincidencia de patrón de una expresión de cadena con un patrón . El patrón puede ser una expresión regular extendida. Devuelve si coincide o si no coincide. Si cualquiera o son NULL, el resultado es NULL. **1 expr pat 0 expr pat**
La forma negativa NOT REGEXP también existe, como alias para . RLIKE y NOT RLIKE son sinónimos de REGEXP y NOT REGEXP, originalmente proporcionados para la compatibilidad con mSQL. **NOT (string REGEXP pattern)**

El patrón no tiene por qué ser una cadena literal. Por ejemplo, se puede especificar como una expresión de cadena o una columna de tabla.

Nota: Dado que MariaDB utiliza la sintaxis de escape de C en cadenas (por ejemplo, "n" para representar el carácter de nueva línea), debe duplicar cualquier "- que utilice en las cadenas REGEXP. REGEXP no distingue mayúsculas de minúsculas, excepto cuando se utiliza con cadenas binarias.

Ejemplos

```
SELECT 'Monty!' REGEXP 'm%y%%';
+-------------------------+
| 'Monty!' REGEXP 'm%y%%' |
+-------------------------+
|                       0 |
+-------------------------+
```

```
SELECT 'Monty!' REGEXP '.*';
+----------------------+
| 'Monty!' REGEXP '.*' |
+----------------------+
|                    1 |
+----------------------+

SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
+---------------------------------------+
| 'new*\n*line' REGEXP 'new\\*.\\*line' |
+---------------------------------------+
|                                     1 |
+---------------------------------------+

SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';
+----------------+-----------------------+
| 'a' REGEXP 'A' | 'a' REGEXP BINARY 'A' |
+----------------+-----------------------+
|              1 |                     0 |
+----------------+-----------------------+

SELECT 'a' REGEXP '^[a-d]';
+---------------------+
| 'a' REGEXP '^[a-d]' |
+---------------------+
|                   1 |
+---------------------+
```