Introducción al uso de las Expresiones Regulares en una Base de Datos Oracle

Por Fernando García
Publicado en marzo 2012

Prólogo

Históricamente Oracle había ofrecido prestaciones muy básicas para el uso de expresiones regulares y la manipulación de cadenas de caracteres. Aquellos desarrolladores que necesitaban hacer uso de expresiones regulares se veían prácticamente obligados a extraer los datos de la base para manipular las cadenas de caracteres en capas intermedias de la aplicación. A partir de la versión 10g, con la introducción de las llamadas funciones REGEXP (REGular EXPressions), Oracle nos ofrece una batería de sofisticadas y potentes herramientas nativas que permiten al desarrollador centralizar el procesamiento intensivo de cadenas de caracteres dentro de la base de datos Oracle y con los lenguajes SQL y PL/SQL.

Tabla de Contenidos

1. El metacaracter Suma (+)
2. El metacaracter Asterisco (*)
3. El metacaracter [char…]
4. El metacaracter [^char…]
5. El metacaracter subexpresión (expr)
6. El metacaracter de anclaje de principio de línea (^)
7. El metacaracter de anclaje de fin de línea ($)
8. El metacaracter de escape (\)
9. Construyendo Expresiones Regulares complejas
10. Uso de las Expresiones Regulares en Oracle
11. Profundizando el conocimiento y uso de las Expresiones Regulares en Oracle

¿Qué son las Expresiones Regulares?

Antes de introducirnos bien en el mundo de las expresiones regulares es necesario que le perdamos un poco el miedo. Lo más común es que un ser humano huya desesperadamente cuando se encuentra por primera vez con una expresión regular como la siguiente:

```
^Ab*[0-8]\*(ha|ve)n$
```

Sin embargo, un buen manejo de las expresiones regulares es algo tan poderoso que vale la pena que nos tomemos un tiempo para aprender a utilizarlas. Además, saber manejar expresiones regulares nos servirá no solamente en el mundo de las bases de datos Oracle; las podremos usar también en otros lenguajes de programación como Perl, Java, .Net, Php y Unix Shell Scripting, entre otros.

La idea de este documento es hacer una introducción al tema de manera práctica, comenzando con ejemplos sencillos para luego ir aumentando la complejidad.

Comenzamos entonces definiendo a una expresión regular como una cadena de caracteres que definen un patrón de búsqueda. En una expresión regular encontramos literales y metacaracteres. Los literales se leen al pie de la letra. Los metacaracteres son caracteres que tienen un significado especial.

Tomemos por ejemplo la siguiente expresión regular:

```
[n|p]ata
```

Los metacaracteres son los corchetes y el símbolo pipe (|). El resto son literales.

Los corchetes agrupan a varios caracteres en un solo caracter.

El símbolo pipe indica un símbolo u otro; es decir la 'n' o la 'p'.

Luego, la expresión regular:

```
[n|p]ata
```

Coincide con las cadenas:

```
nata pata
```

Seguramente el lector se preguntará cómo determinar qué elementos de una expresión regular son literales y qué elementos son metacaracteres. Más adelante encontrará la respuesta. Sigamos.

¿Para qué sirven las Expresiones Regulares?

Para que una cadena de caracteres coincida con dicho patrón:

- Debe estar el literal 'c'.
- Le debe seguir el literal 'a'
- Le debe seguir un y solamente un carácter cualquiera.
- Le debe seguir el literal 'a'

En la expresión regular encontramos 3 literales y el metacaracter punto. Analicemos las siguientes cadenas y veamos cuales coinciden con el patrón buscado y cuáles no.

Cadena	¿Coincide?	Análisis
casa	Si	Coinciden los literales y en la posición del operador punto aparece sólo un carácter
cana	Si	Coinciden los literales y en la posición del

		operador punto aparece sólo un carácter
cara	Si	Coinciden los literales y en la posición del operador punto aparece sólo un carácter
canta	No	En la posición del punto aparece más de un carácter
pala	No	El literal 'p' en la primer posición no respeta el literal ´c´ definido en la expresión regular

El metacaracter asterisco (*)

Coincide con cero o más ocurrencias de la subexpresión que le precede al asterisco. Analicemos la expresión:

```
cas*a
```

Para que una cadena de caracteres coincida con dicho patrón:

- Debe estar el carácter 'c' - Le debe seguir el carácter 'a' - Le debe seguir cero o mas ocurrencias del carácter 's'. Es decir que el carácter 's' puede o no estar. - Le debe seguir el carácter 'a'

Analicemos qué pasa con las mismas cadenas de caracteres del ejemplo anterior:

Cadena	¿Coincide?	Análisis
Casa cassssa	Si	El símbolo asterisco (*) es precedido por la subexpresión "s". Por lo tanto todas las cadenas que tienen cero o más "s" a partir de

cassssssssa		la tercera posición y respetan el resto de los literales, coinciden con el patrón representado en la expresión regular.
caa	Si	El patrón acepta la no existencia del carácter 's' en la tercera posición.
lasa	No	No respeta todos los literales de la expresión regular
tassa	No	No respeta todos los literales de la expresión regular

El metacaracter [char...]

Coincide con una sola ocurrencia de alguno de los caracteres de la lista entre corchetes. En la lista, todos los caracteres son tomados como literales excepto algunos caracteres que son interpretados como metacaracteres. Aquí nos limitaremos a mencionar el metacaracter '-' para representar rangos de caracteres. Analicemos la siguiente expresión regular:

```
ca[snt]a
```

Para que una cadena de caracteres coincida con dicho patrón:

- Debe estar el carácter 'c'
- Le debe seguir el carácter 'a'
- Le debe seguir uno y solamente uno de los siguientes caracteres: 's', 'n' o 't'.
- Le debe seguir el carácter 'a'

Cadena	¿Coincide?	Análisis
casa	Si	Coinciden los literales y en la posición del operador aparece sólo un carácter y es de los de la lista
cana	Si	Coinciden los literales y en la posición del operador aparece sólo un carácter y es de los de la lista
cata	Si	Coinciden los literales y en la posición del operador aparece sólo un carácter y es de los de la lista
cara	No	En la posición del operador aparece un solo carácter pero no está en la lista
canta	No	Si bien en la posición del operador aparecen literales de la lista; son más de uno
pasa	No	No respeta el primer literal de la lista

Como se dijo anteriormente, en la lista se puede utilizar el metacaracter rango '-'. Haciendo uso de este metacaracter la siguiente lista de caracteres:

```
[123456789]
```

Podemos expresarla como

```
[1-9]
```

Y la siguiente lista de caracteres

```
[abcdef]
```

Podemos expresarla como

```
[a-f]
```

Analicemos ahora la siguiente expresión regular:

```
a[3-7][f-i]9
```

Para que una cadena coincida con el patrón:

- Debe tener el literal 'a'
- Le debe seguir un numero de 3 a 7. Es decir: 3, 4, 5, 6 ó 7.
- Le debe seguir una letra de la f a la i. Es decir 'f', 'g', 'h' o 'i'
- Le debe seguir el numero 9

Cadena	¿Coincide?	Análisis
a3h9	Si	Coinciden los literales y en la posición de los operadores aparecen literales incluidos en los rangos especificados en el patrón de la expresión regular.
a33f9	No	En la posición del primer operador aparece más de un caracter
a3z9	No	La 'z' no está incluida en el rango del segundo operador

El metacaracter [^char...]

El sombrerito (^) que precede a la lista indica que los literales que le siguen no deben estar en la cadena de caracteres para que se produzca una coincidencia. Veamos,

```
ca[^snt]a
```

Para que una cadena de caracteres coincida con dicho patrón:

- Debe estar el carácter 'c'
- Le debe seguir el carácter 'a'
- Le debe seguir uno y solamente un carácter que no sea ni 's' ni 'n' ni 't'.
- Le debe seguir el carácter 'a'

Cadena	¿Coincide?	Análisis
casa	No	Coinciden los literales y en la posición del operador aparece un carácter de los negados en la lista
cana	No	Coinciden los literales y en la posición del operador aparece un carácter de los negados en la lista
cata	No	Coinciden los literales y en la posición del operador aparece un carácter de los negados en la lista

cara	Si	Coinciden los literales y en la poscion del operador aparece un carácter que no está en la lista de literales negados.
canta	No	En la posición del operador aparece más de un valor
para	No	No respeta el primer literal de la lista

El metacaracter subexpresión (expr)

Considera a toda la expresión entre paréntesis como una unidad. La expresión puede ser una simple cadena de literales o una expresión compleja conteniendo otros metacaracteres. Analicemos la siguiente expresión regular

```
chau(hola)*chau
```

Ahora el asterisco precede a la expresión

```
hola
```

El asterisco indica que la expresión

```
hola
```

Puede aparecer cero ó más veces

Cadena	¿Coincide?	Análisis

chauchau	Si	Coinciden los literales y la expresión 'hola' aparece cero veces
chauholachau	Si	Coinciden los literales y la expresión 'hola' aparece una vez, es decir cero o más veces.
chauholaholachau	Si	Coinciden los literales y la expresión 'hola' aparece dos veces, es decir cero o más veces.
holachau	No	No aparecen el literal chau' que precede al ‘hola’
chau	No	No aparece el literal 'chau' que precede al ‘hola’

El metacaracter de anclaje de principio de línea (^)

Coincide con el principio de línea y el operador está representado con el caracter sombrerito (^). En la expresión regular

```
^hola
```

Los literales que componen la palabra 'hola' deben estar al inicio de la línea para que se produzca la coincidencia con el patrón expresado.

Cadena	¿Coincide?	Análisis
hola	Si	Los literales que componen la palabra 'hola'

		están al inicio de la línea
chauhola	No	La línea no comienza con los literales que conforman la palabra 'hola'
holachau	Si	Los literales que componen la palabra 'hola' están al inicio de la línea
holahola	Si	Los literales que componen la palabra 'hola' están al inicio de la línea

El metacaracter de anclaje de fin de línea ($)

Coincide con el final de línea y el operador está representado con el carácter pesos ($). En la expresión regular

```
hola$
```

Los literales que componen la palabra 'hola' deben estar al final de la línea para que se produzca la coincidencia con el patrón expresado.

Cadena	¿Coincide?	Análisis
Hola	Si	Los literales que componen la palabra 'hola' están al final de la línea
chauhola	Si	La línea finaliza con los literales que conforman la palabra 'hola'

holachau	No	Los literales que componen la palabra 'hola' no están al final de la línea
holahola	Si	Los literales que componen la palabra 'hola' están al final de la línea

El metacaracter de escape (\)

Precediendo a un metacaracter con el símbolo de escape, el metacaracter será interpretado como un literal. El doble carácter de escape (\\) permite considerar al carácter de escape como literal.

En la expresión regular

```
hola\*
```

Cadena	¿Coincide?	Análisis
hola	No	Falta el literal '*' que sigue literal 'a'
Hola*	Si	Coinciden todos los literales
hol*	No	Faltan los literales 'a' y '*'

Construyendo Expresiones Regulares complejas

Hasta aquí hemos visto algunos de los metacaracteres que se usan con más frecuencia. Combinando varios de estos metacaracteres en una sola expresión regular, podemos hacer construcciones más complejas y muy poderosas. Analicemos la siguiente expresión regular en la que encontramos varios metacaracteres

```
^hola[0-9]*chau[^a]+$
```

Hagamos un desglose de la expresión regular:

^hola	Al principio de la línea debe estar la palabra 'hola'
[0-9]*	Luego de la palabra 'hola' deben aparecer cero o mas dígitos del 0 al 9
chau	Luego debe aparecer el literal 'chau'
[^a]+$	La línea debe continuar hasta el final con uno o más caracteres distintos de 'a'

Busquemos el patrón en algunas cadenas de caracteres

Cadena	¿Coincide?	Análisis
hola123chaub	Si	Comienza con el texto 'hola'. Le siguen varios dígitos entre 0 y 9. Le sigue el literal 'chau' y finaliza con al menos una letra distinta de 'a'.
holachauc	Si	Comienza con el texto 'hola'. No aparecen dígitos del 0 al 9 (el asterisco indica que esto

		es aceptado para que haya coincidencia). Luego aparece el literal 'chau' y se finaliza con una letra distinta de 'a'
hola0chaua	No	Finaliza con una ‘a’

Uso de las Expresiones Regulares en Oracle

Hasta aquí hemos visto un poco de teoría y algunos ejemplos prácticos para conocer y aprender un poco acerca de las expresiones regulares. Pero, ¿cómo podemos hacer uso de este conocimiento adquirido en una base de datos Oracle?

A partir de la versión 10g Oracle nos ofrece un grupo de nuevas funciones y condiciones para poder manejar expresiones regulares en el lenguaje SQL:

REGEXP_LIKE	Condición que se puede utilizar en la cláusula WHERE de una sentencia SQL SELECT y que permite retornar aquellas filas que coinciden con el patrón especificado en una expresión regular.
REGEXP_COUNT	Función que permite contar el número de veces que un patrón aparece en una cadena de caracteres.
REGEXP_INSTR	Función que permite determinar la posición de inicio de un patrón específico en una cadena de caracteres.
REGEXP_REPLACE	Función que permite hacer búsqueda y reemplazo en una cadena de caracteres utilizando expresiones regulares para la búsqueda.

REGEXP_SUBSTR	Función para extraer de una cadena una subcadena de caracteres que coincidan con un patrón especificado en una expresión regular.

Veamos algunos ejemplos de implementación de estas condiciones y funciones combinadas con el uso de expresiones regulares.

La condición REGEXP_LIKE es similar a la condición LIKE pero, a diferencia del LIKE, utiliza patrones basados en expresiones regulares para la búsqueda de coincidencias en las cadenas de caracteres analizadas.

En el siguiente ejemplo, vemos una tabla de empleados cuyos nombres fueron migrados de un sistema legacy. En muchos casos el nombre ha quedado separado del apellido por varios espacios en blanco:

Copy

```
SQL> select nombre from empleados;

NOMBRE
----------------------------------------
Fernando         Garcia
Marcelo  Burgos
       Marcelo         Ochoa
Gerardo     Tezza
Clarisa Maman Orfali
Rodolfo  Pajuelo Quispe

6 rows selected.
```

El único caso de un solo espacio entre el nombre y el apellido es el de Clarisa Maman Orfali. Veamos cómo podemos construir un query que detecte aquellas filas en que haya dos o más espacios separando nombres de apellidos.

Copy

```
SQL> select nombre
  2  from empleados
  3  where regexp_like(nombre, '[ ][ ][ ]*');

NOMBRE
----------------------------------------
Fernando         Garcia
Marcelo  Burgos
       Marcelo         Ochoa
Gerardo     Tezza
Rodolfo  Pajuelo Quispe
```

Corrijamos esta situación dejando un solo blanco de separación en donde hay dos o más espacios en blanco.

Copy

```
SQL> update empleados
  2  set nombre = regexp_replace(nombre, '[ ][ ][ ]*', ' ')
  3  where regexp_count(nombre, '[ ][ ][ ]*') > 0;

5 rows updated.
```

Veamos cómo quedo la tabla luego de la corrección

Copy

```
SQL> select nombre from empleados;

NOMBRE
----------------------------------------
Fernando Garcia
Marcelo Burgos
```

```
 Marcelo Ochoa
Gerardo Tezza
Clarisa Maman Orfali
Rodolfo Pajuelo Quispe

6 rows selected.
```

Bastante bien. Salvo el caso de Marcelo Ochoa, en donde nos quedó un espacio en blanco al principio de la línea. Eliminemos entonces los espacios en blanco que están al principio de la línea.

Copy

```
SQL> update empleados
  2  set nombre = regexp_replace(nombre, '^ ')
  3  where regexp_count(nombre, '^ ') > 0
  4  /

1 row updated.

SQL> select nombre from empleados;

NOMBRE
----------------------------------------
Fernando Garcia
Marcelo Burgos
Marcelo Ochoa
Gerardo Tezza
Clarisa Maman Orfali
Rodolfo Pajuelo Quispe

6 rows selected.
```

En este otro ejemplo vemos una tabla de impuestos en la que almacenamos un código de identificación tributaria numérico y que debe respetar el siguiente formato:

Copy

```
99-99999999-99

SQL> select cuit from impuestos;

CUIT
-------------
20-20198123-9
10-43112345-8
1-22123987-9
8883838382
aa-75212456-x
```

Busquemos aquellos códigos que no están respetando el formato requerido:

Copy

```
select cuit
from impuestos
where not regexp_like (cuit, '[0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]-[0-9]')

CUIT
-------------
1-22123987-9
8883838382
aa-75212456-x
```

A fin de evitar que la tabla siga admitiendo códigos que no respetan el formato, podemos crear una constraint que fuerce el ingreso de códigos que respeten el formato.

Copy

```
alter table impuestos
add constraint c_cuit
check (regexp_like(cuit,'[0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]-[0-9]'))
enable novalidate
/
Table altered.
```

Veamos qué ocurre al ingresar nuevas filas a la table de impuestos

Copy

```
SQL> insert into impuestos (cuit) values ('20-17979866-2');

1 row created.

SQL> insert into impuestos (cuit) values ('1x-17979866-2');
insert into impuestos (cuit) values ('1x-17979866-2')
*
ERROR at line 1:
ORA-02290: check constraint (SYS.C_CUIT) violated


SQL> insert into impuestos values ('12888122812');
insert into impuestos values ('12888122812')
*
ERROR at line 1:
ORA-02290: check constraint (SYS.C_CUIT) violated
```

Profundizando el conocimiento y uso de las expresiones regulares en Oracle

Hemos visto varios ejemplos para demostrar el potencial y utilidad que nos ofrecen las expresiones regulares en las bases de datos Oracle. Existen muchos más metacaracteres que no hemos mencionado aquí. Las funciones tampoco las hemos visto en profundidad; hay más argumentos para