Pattern Matching (. +?*)

Metacharacter	Operator Name	Description
•	Any Character Dot	Matches any character
+	One or More Plus Quantifier	Matches one or more occurrences of the preceding subexpression
Ś	Zero or One Question Mark Quantifier	Matches zero or one occurrence of the preceding subexpression
*	Zero or More Star Quantifier	Matches zero or more occurrences of the preceding subexpression

Pattern Matching (. + ? *)

For these examples we are returning matches of the specified pattern in its entirety.

Case sensitive.

PATTERN = '...'

PATTERN = '.ab+'

PATTERN = '1a?b'

PATTERN = '1a*b'

String	Match
AB	×
ABCS	×
ABC	\checkmark
1A4	\checkmark
S 3	\checkmark

String	Match
2AB	×
2ab	\checkmark
xabb	\checkmark
abb	×
aab	\checkmark

String	Match
1aaab	×
1 b	\checkmark
2	×
1 a	×
aa	×

String	Match
1aaab	\checkmark
1 b	\checkmark
2	×
1 a	×
aa	×

Pattern Matching - Intervals

Metacharacter	Operator Name	Description
{m}	IntervalExact Count	Matches exactly <i>m</i> occurrences of the preceding subexpression
{m,}	IntervalAt Least Count	Matches at least <i>m</i> occurrences of the preceding subexpression
{m,n}	IntervalBetween Count	Matches at least m , but not more than n occurrences of the preceding subexpression

 $\d = \d{3}$

Pattern Matching - Intervals

For these examples we are returning matches of the specified pattern in its entirety. \d and \w are NOT Case sensitive but text literals are.

PATTERN = $'\d{2}\w{3}'$

String	Match
234sf	\checkmark
11abc	\checkmark
34Dwc	\checkmark
Ad4	×
24d e	×

PATTERN = '\d{1,}\w?'

String	Match
245er	×
2	\checkmark
2 a	\checkmark
abb	×
343523e	\checkmark

PATTERN = '\d{1,3}-\d{2}'

String	Match
234-0	X
23-42	\checkmark
225-23	\checkmark
2-3	×
E-35	X

Pattern Matching – Lists and Groups

Metacharacter	Operator Name	Description
[]	Matching Character List	Matches any character in list
[^]		Matches any character not in list
	Character List	
()	Subexpression or	Treat expression as a unit. The
	Grouping	subexpression can be a string of literals or a
		complex expression containing operators.

Pattern Matching – Lists and Groups

For these examples we are returning matches of the specified pattern in its entirety.

Text literals are case sensitive.

PATTERN = '[a-z][a-b]'

PATTERN = '[ade][^erf]'

PATTERN = '(erf)(oo)'

PATTERN = '[abcde][^a-z](end)'

String	Match
adt	×
zza	\checkmark
aab	\checkmark
4ax	×
24d e	×

String	Match
d	×
d3	\checkmark
ah	$\overline{\mathbf{V}}$
ef	×
a-	\checkmark

String	Match
eroo	×
erfoo	\checkmark
Erfoo	×
2-3	×
E-35	×

String	Match
a2end	\checkmark
E7end	X
eeend	X
A2end	X
22end	X

REGEXP_REPLACE

• Replaces a sequence of characters with another sequence of characters

```
replacement_string
                                   If omitted the matched
                                   pattern will be removed.
                                   Otherwise the pattern will
                                   be replaced with the string
                                   specified
             string
             This is the string to
             search A
REGEXP_REPLACE( string, pattern [, replacement_string [, start_position [, nth_appearance [, match_parameter ]]]])
                     pattern
                     The regular
                     expression
```

What are Regular Expressions

- Regular Expressions aka RegEx
- Used for pattern matching in text strings
- Uses a combination of characters, metacharacters and operators to define a search pattern
- Regex is extremely powerful in searching for and manipulating text
- There are specific functions in SQL that utilize RegEx

Metacharacters – Oracle SQL

Metacharacter	Operator Name	Description
•	Any Character Dot	Matches any character
+	One or More Plus Quantifier	Matches one or more occurrences of the preceding subexpression
Ś	Zero or One Question Mark Quantifier	Matches zero or one occurrence of the preceding subexpression
*	Zero or More Star Quantifier	Matches zero or more occurrences of the preceding subexpression
\d	Any Character Digit	Matches any digit character
\w	Any Character Word	Matches any word character. Not case sensitive
{m}	IntervalExact Count	Matches exactly m occurrences of the preceding subexpression
{m,}	IntervalAt Least Count	Matches at least m occurrences of the preceding subexpression
{m,n}	IntervalBetween Count	Matches at least m , but not more than n occurrences of the preceding subexpression
[]	Matching Character List	Matches any character in list
[^]	Non-Matching Character List	Matches any character not in list
	Or	'a b' matches character 'a' or 'b'.
()	Subexpression or Grouping	Treat expression as a unit. The subexpression can be a string of literals or a complex expression containing operators.
\n	Backreference	Matches the n^{th} preceding subexpression, where n is an integer from 1 to 9.
\	Escape Character	Treat the subsequent metacharacter in the expression as a literal.
٨	Beginning of Line Anchor	Match the subsequent expression when it occurs at the start of the string.
\$	End of Line Anchor	Match the preceding expression when it occurs at the end of the string.

Pattern Matching (\d \w)

Metacharacter	Operator Name	Description
\d	Any Character Digit	Matches any digit character
\w	Any Character Word	Matches any word character
		[A-Z][a-z][0-9]. Not case sensitive

Pattern Matching (\d \w)

For these examples we are returning matches of the specified pattern within each string. \d and \w are NOT Case sensitive but text literals are.

PATTERN = 'a\w\d'

PATTERN = '\d+\w?'

PATTERN = '\d\d\w*'

PATTERN = '2+\w?\d*'

String	Match
11abc	×
ab2	\checkmark
aA3	\checkmark
Ad4	×
Ad 1a 3	X

String	Match
asc	×
2	$\overline{\mathbf{V}}$
234	$\overline{\mathbf{V}}$
abb	×
1 d	✓

String	Match
1aaab	×
11 Z	\checkmark
22	\checkmark
1 a	×
aa	×

String	Match
22 a1	\checkmark
21	\checkmark
2aaab1	×
1 a	×
aa	×

Pattern Matching – Or

Metacharacter	Operator Name	Description
	Or	'a b' matches character 'a' or 'b'.

Pattern Matching – Or

For these examples we are returning matches of the specified pattern in its entirety.

Text literals are case sensitive.

PATTERN = '(abd | def)'

String	Match
ba	X
abd	\checkmark
def	\checkmark
cdef	X

abc

PATTERN = '(a|b|c|d){4}'

String	Match
aaaa	\checkmark
abcs	×
aaad	\checkmark
aaa	×
cddd	\checkmark

Pattern Matching – Back Reference

Metacharacter	Operator Name	Description
\n	Backreference	Matches the nth preceding subexpression, where
		n is an integer from 1 to 9.

(ABC)(AVD)(SET)







Pattern Matching – Back Reference

For these examples we are returning matches of the specified pattern in its entirety.

Text literals are case sensitive.

PATTERN = '(aa)(bb)\2'

PATTERN = '[abc](aa)(bb)\2'

String	Match
aabbaa	×
aabbbb	\checkmark
AABBBB	×
abb	×
aaa	×

String	Match
caabbaa	×
caabbbb	\checkmark
baabbbb	\checkmark
3aabbbb	×
baabbBb	×

Pattern Matching – Escape Characters

Metacharacter	Operator Name	Description
	Escape Character	Treat the subsequent metacharacter in the
		expression as a literal.

Pattern Matching – Escape Characters

For these examples we are returning matches of the specified pattern in its entirety.

Text literals are case sensitive.

PATTERN = '(aa)(bb)*'

String	Match
aabb	×
aabb*	\checkmark
Aabb*	×
aab	×
*	×

Pattern Matching – Line Anchors

Metacharacter	Operator Name	Description
^	Beginning of Line	Match the subsequent expression only when it
	Anchor	occurs at the beginning of the string.
\$	End of Line Anchor	Match the preceding expression only when it
		occurs at the end of the string.

Pattern Matching – Line Anchors

This time we are not matching the entire string – just the beginning or end.

Text literals are case sensitive.

PATTERN = '^(abe)[a-z]'

PATTERN = 'col[123]\$'

String	Match
abe1	×
abez	\checkmark
abeA	X
abez13	ightharpoons
abe3a	×

String	Match
col5	×
xcol1	\checkmark
col2	\checkmark
tocol5	×
cola5	×

REGEXP_LIKE

- Performs regular expression matching and will return records containing that pattern –
 does not need to match entire string just a part of it
- Used in the 'WHERE' clause

expression

This is the character expression to be used as the search value. It is a column or field

Value	Description
'c'	Perform case-sensitive matching.
'i '	Perform case-insensitive matching.
'n'	Allows the period character (.) to match the newline character. By default, the period is a wildcard.
'm'	expression is assumed to have multiple lines, [^] is the start of a line, \$ is the end of a line. By default, expression is assumed to be a single line.
'x'	Whitespace characters are ignored. Disabled by default

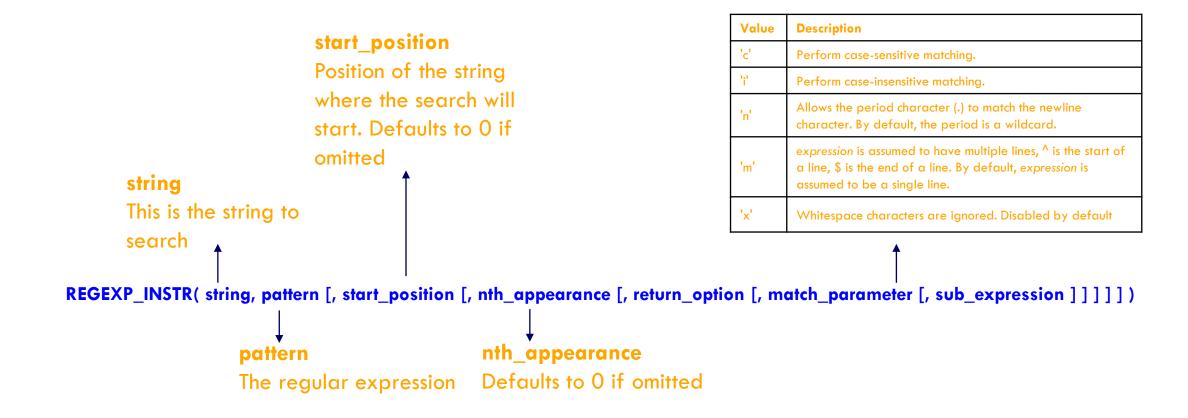
REGEXP_LIKE(expression, pattern [,match parameter])

pattern

The regular expression

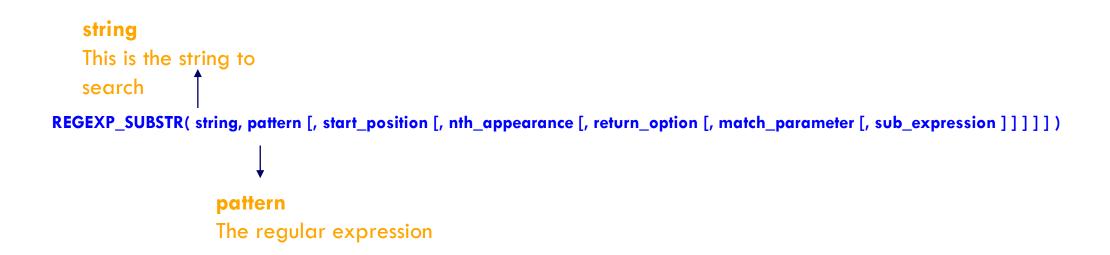
REGEXP_INSTR

Returns the location of a regular expression pattern in the string (starts from 0)



REGEXP_SUBSTR

Extracts a substring from a string



REGEXP_REPLACE

• Replaces a sequence of characters with another sequence of characters

```
replacement_string
                                   If omitted the matched
                                   pattern will be removed.
                                   Otherwise the pattern will
                                   be replaced with the string
                                   specified
             string
             This is the string to
             search A
REGEXP_REPLACE( string, pattern [, replacement_string [, start_position [, nth_appearance [, match_parameter ]]]])
                     pattern
                     The regular
                     expression
```