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
?	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 = '...'

String	Match
AB	✗
ABCS	✗
ABC	✓
1A4	✓
S 3	✓

PATTERN = '.ab+'

String	Match
2AB	✗
2ab	✓
xabb	✓
abb	✗
aab	✓

PATTERN = '1a?b'

String	Match
1aaab	✗
1b	✓
2	✗
1a	✗
aa	✗

PATTERN = '1a*b'

String	Match
1aaab	✓
1b	✓
2	✗
1a	✗
aa	✗

Pattern Matching - Intervals

Metacharacter	Operator Name	Description
{*m*}	Interval--Exact Count	Matches exactly *m* occurrences of the preceding subexpression
{*m*,}	Interval--At Least Count	Matches at least *m* occurrences of the preceding subexpression
{*m*,*n*}	Interval--Between Count	Matches at least *m*, but not more than *n* occurrences of the preceding subexpression

\d\d\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	✓
11abc	✓
34Dwc	✓
Ad4	✗
24d e	✗

PATTERN = ‘\d{1,}\w?’

String	Match
245er	✗
2	✓
2a	✓
abb	✗
343523e	✓

PATTERN = ‘\d{1,3}-\d{2}’

String	Match
234-0	✗
23-42	✓
225-23	✓
2-3	✗
E-35	✗

Pattern Matching – Lists and Groups

Metacharacter	Operator Name	Description
[…]	Matching Character List	Matches any character in list …
[^ …]	Non-Matching Character List	Matches any character not in list …
(…)	Subexpression or Grouping	Treat expression … as a unit. The 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-z][a-b]’

String	Match
adt	✗
zza	✓
aab	✓
4ax	✗
24d e	✗

PATTERN = ‘[ade][^erf]’

String	Match
d	✗
d3	✓
ah	✓
ef	✗
a-	✓

PATTERN = ‘(erf)(oo)’

String	Match
eroo	✗
erfoo	✓
Erfoo	✗
2-3	✗
E-35	✗

PATTERN = ‘[abcde][^a-z](end)’

String	Match
a2end	✓
E7end	✗
eeend	✗
A2end	✗
22end	✗

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

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
?	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*}	Interval--Exact Count	Matches exactly *m* occurrences of the preceding subexpression
{*m*,}	Interval--At Least Count	Matches at least *m* occurrences of the preceding subexpression
{*m*,*n*}	Interval--Between 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 nth preceding subexpression, where *n* is an integer from 1 to 9.
\	Escape Character	Treat the subsequent metacharacter in the expression as a literal.
^	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’

String	Match
11abc	✗
ab2	✓
aA3	✓
Ad4	✗
Ad 1a 3	✗

PATTERN = ‘\d+\w?’

String	Match
asc	✗
2	✓
234	✓
abb	✗
1d	✓

PATTERN = ‘\d\d\w*’

String	Match
1aaab	✗
11Z	✓
22	✓
1a	✗
aa	✗

PATTERN = ‘2+\w?\d*’

String	Match
22a1	✓
21	✓
2aaab1	✗
1a	✗
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	✗
abd	✓
def	✓
cdef	✗
abc	✗

PATTERN = ‘(a|b|c|d){4}’

String	Match
aaaa	✓
abcs	✗
aaad	✓
aaa	✗
cddd	✓

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)

1 2 3

\2 = (AVD)

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’

String	Match
aabbaa	✗
aabbbb	✓
AABBBB	✗
abb	✗
aaa	✗

PATTERN = ’[abc](aa)(bb)\2’

String	Match
caabbaa	✗
caabbbb	✓
baabbbb	✓
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*	✓
Aabb*	✗
aab	✗
*	✗

Pattern Matching – Line Anchors

Metacharacter	Operator Name	Description
^	Beginning of Line Anchor	Match the subsequent expression only when it 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]'

String	Match
abe1	✗
abez	✓
abeA	✗
abez13	✓
abe3a	✗

PATTERN = 'col[123]$'

String	Match
col5	✗
xcol1	✓
col2	✓
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

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

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

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)

start_position
Position of the string where the search will start. Defaults to 0 if omitted

string
This is the string to search

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_INSTR(string, pattern [, start_position [, nth_appearance [, return_option [, match_parameter [, sub_expression]]]]]])

pattern
The regular expression

nth_appearance
Defaults to 0 if omitted

REGEXP_SUBSTR

- Extracts a substring from a string

string
This is the string to search

REGEXP_SUBSTR(string, pattern [, start_position [, nth_appearance [, return_option [, match_parameter [, sub_expression]]]]])

pattern
The regular expression

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

REGEXP_REPLACE(string, pattern [, replacement_string [, start_position [, nth_appearance [, match_parameter]]]])

pattern
The regular expression