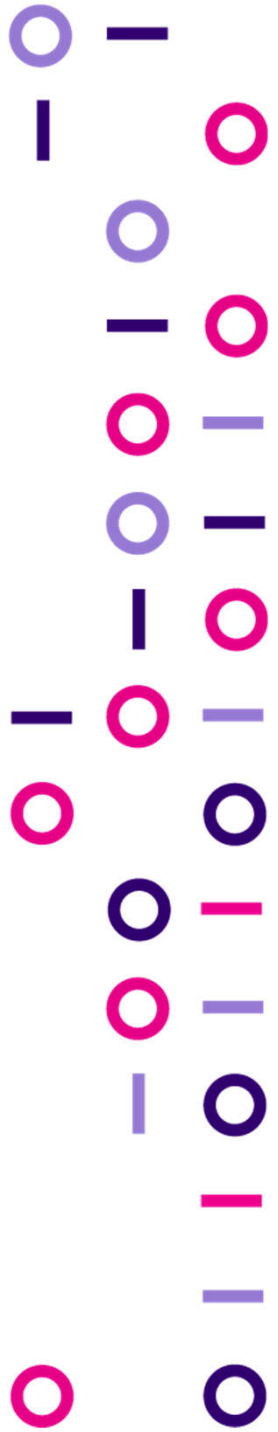
COTIVITI

Oracle Regular Expressions

UTOUG Training Days 2019

Scott Heffron

Senior Business Systems Analyst



Definition

A regular expression, regex or regexpr (sometimes called a rational expression) is a sequence of characters that define a search pattern. Usually this pattern is used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

Source: Wikipedia

What you will learn today

1. What are Regular Expressions
2. Where Do We Use them
3. Executing Regular Expressions

Agenda

- ✓ Where to use Regular Expressions
- ✓ Understanding Metacharacters
- ✓ Using the REGEXP_LIKE() function
- ✓ Using the REGEXP_REPLACE() function
- ✓ Using the REGEXP_SUBSTR() function
- ✓ Using the REGEXP_INSTR() function
- ✓ Using the REGEXP_COUNT() function
- ✓ Q & A

When and Where to Use Regular Expressions

- ✓ Searching and manipulating simple or complex patterns in strings
- ✓ Searching and manipulating string in SQL and PL/SQL, using SQL functions
- ✓ Using match logic in strings like URL, email, phone number
- ✓ Using in constraints to provide data consistency
- ✓ Oracle has functions called REPLACE, SUBSTR, INSTR and COUNT. But regular expressions can be more powerful to use.

Meta Characters

Regular expression metacharacters are used to identify a specific aspect of a string being processed. Such as a wildcard or repeating character. It can also provide control of the behavior or the parsing, such as grouping a pattern or identifying alternates.

There are a considerable number of metacharacters available. Too many for one session. We will look at a some that are used quite often.

Common metacharacters

Metacharacter	Description
*	Matches zero or more occurrences
+	Matches one or more occurrences
?	Matches zero or one occurrences
|	Alternation operator for specific alternative patterns
^	Matches the beginning of a string. In multiline, matches the beginning of any line with the string
$	Matches the end of the string
.	Matches any character in the supported character set except null
()	Grouping expression
{m}	Matches exactly m times
{m,}	Matches at least m times
{m,n}	Matches at least m times, but no more then n times

Common metacharacters - Continued

Metacharacter	Description
[]	Used to specify a matching list where you are trying to match an one of the characters in the list
[==]	Specifies equivalence classes.
a-z	Any lowercase letter
A-Z	Any uppercase letter
\s	Matches a whitespace character
\S	Matches a non-whitespace character
\d	Digit
[:alnum:]	Alphanumeric characters
[:alpha:]	Alphabetic characters
[:ascii:]	ASCII characters
[:blank:]	Space or tab
[:cntrl:]	Control Characters
[:digit:]	Digits (0-9)

Oracle Regular Expression Functions

- ✓ There are 5 regular expression functions that can be used for searching and manipulating strings.
- ✓ We can use character and CLOB data types in regular expressions

Oracle Regular Expression Functions

Function	Description
REGEXP_LIKE	Used in searching for patterns in strings. Used in WHERE clause
REGEXP_SUBSTR	Returns the substring matching with the pattern. Used in the SELECT or WHERE clause
REGEXP_REPLACE	Replaces the text matched with the pattern with a specified string. Used in SELECT or WHERE clause and PL/SQL code
REGEXP_INSTR	Returns an integers value of the position of the searched pattern in the target string. When no match found, it returns zero
REGEXP_COUNT	Returns the number of occurrences of a specified pattern in the target string. Zero if no match is found.

Using the REGEXP_LIKE Function

- ✓ Use REGEXP_LIKE for matching a regular expression with our string
- ✓ REGEXP_LIKE is very similar to the LIKE condition
- ✓ The difference is REGEXP_LIKE uses powerful regular expression matching instead of simple pattern matching in LIKE.
- ✓ Used in WHERE clause

REGEXP_LIKE Parameters

Syntax: REGEXP_LIKE(<string>, <pattern>, <match_parameter>)

String: data field to search

Pattern: The regular expression that we are searching for in the string

Optional Options

Match Parameters:

1. ‘c’: Case sensitivity matching (default)
2. ‘i’: No-case sensitive matching
3. ‘n’: Allows match any character operator
4. ‘m’: Treats source string as multiple lines
5. ‘x’: Whitespace characters are ignored. By default, whitespace characters are matched like any other character.

Special Notes for REGEXP_LIKE

- ✓ The REGEXP_LIKE condition uses the input character pattern to set the evaluating strings
- ✓ If you specify match_parameter values that conflict. The REGEXP_LIKE condition will use the last value to break the conflict
- ✓ If the match_parameter is omitted, the REGEXP_LIKE condition will use the case-sensitivity as determined by the NLS_SORT parameter.

Using Cases - REGEXP_LIKE Function

- ✓ Employees with last name starting with ‘A’
- ✓ Employees with last name starting with ‘A’ and ending with ‘n’.
- ✓ Employees with last name starting with ‘A’ or ‘D’ and ending with ‘n’.

Use Cases – REGEXP_LIKE function with SQL

```
-- ================================================
-- Find records starting with A
-- ================================================
SELECT first_name, last_name FROM hr.employees
 WHERE last_name LIKE 'A%'
;

SELECT first_name, last_name FROM hr.employees
 WHERE REGEXP_LIKE(last_name, '^A')
;

-- ================================================
-- Find records starting with A and ending with n
-- ================================================
SELECT first_name, last_name FROM hr.employees
 WHERE last_name LIKE 'A%' AND last_name LIKE '%n'
;

SELECT first_name, last_name FROM hr.employees
 WHERE REGEXP_LIKE(last_name, '^A.*n$')
;

-- ================================================
-- Find records starting with A or D and ending with n
-- ================================================
SELECT first_name, last_name FROM hr.employees
 WHERE ((last_name LIKE 'A%' AND last_name LIKE '%n')
      OR (last_name LIKE 'D%'AND last_name LIKE '%n'))
;

SELECT first_name, last_name FROM hr.employees
 WHERE REGEXP_LIKE(last_name, '^A.*n$|^D.*n$')
;
```

Using the REGEXP_REPLACE Function

- ✓ Use REGEXP_REPLACE for matching a regular expression with our string
- ✓ REGEXP_REPLACE is very similar to the REPLACE condition
- ✓ The difference is REGEXP_REPLACE uses powerful regular expression matching instead of simple pattern matching in REPLACE.
- ✓ Used in SELECT and WHERE clause
- ✓ Used in PL/SQL

REGEXP_REPLACE Parameters

Syntax: REGEXP_REPLACE(<string>, <pattern>, <str_to_replace>, <position>, <occurence>, <match_parameter>)

string: Text to be changed
pattern: The regular expression that we are searching for in the string
str_to_replace: The new string

Optional Options
position: Position (number) to start searching
occurrence: Which occurrence to be changed
match Parameters:

1. 'c': Case sensitivity matching (default)
2. 'i': No-case sensitive matching
3. 'n': Allows match any character operator
4. 'm': Treats source string as multiple lines
5. 'x': Whitespace characters are ignored.

Special Notes for REGEXP_REPLACE

- ✓ If there are conflicting values provided for the match parameters, the function will use the last value

Using Cases for REGEXP_REPLACE Function

- ✓ Replace periods with dashes within the phone number
- ✓ Replace the vowels with dashes with the first name
- ✓ Working with digits in a string
- ✓ Remove the extra spaces within the string

Use Cases – REGEXP_REPLACE w/SQL

```
-- ================================================
-- Replace periods with dashes
-- ================================================
SELECT phone_number
     , REPLACE(phone_number, '.', '-') AS function_phone
     , REGEXP_REPLACE(phone_number, '\.', '-') AS regexpr_phone
  FROM EMPLOYEES;

-- ================================================
-- Replace replace all vowels with dashes
-- ================================================
SELECT first_name
     , REPLACE(InitCap(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(LOWER(first_name), 'a', '0'), 'e', '0'), 'i', '0'), 'o', '0'),
'u', '0')), '0', '-') AS function_first_name
     , REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(first_name, 'a', '-'), 'e', '-'), 'i', '-'), 'o', '-'), 'u', '-') AS
function_first_name
     , REGEXP_REPLACE(first_name, 'a|e|i|o|u', '-', 1, 0, 'i') AS regexpr_first_name
  FROM EMPLOYEES;

-- ================================================
-- work on digits
-- ================================================
SELECT 'I earned $200 in 2 hours!'  AS BEFORE
     , REPLACE('I earned $200 in 2 hours!', '2', '3') AS function_try_1_AFTER
     , REPLACE('I earned $200 in 2 hours!', '2 h', '3 h') AS function_try_2_AFTER
     , SUBSTR('I earned $200 in 2 hours!', 1, 13) || REPLACE(SUBSTR('I earned $200 in 2 hours!', 14), 2, 3) AS
function_substr_after
     , REGEXP_REPLACE('I earned $200 in 2 hours!', '\d', '3', 15) AS regexpr_AFTER
  FROM dual;

-- ================================================
-- Remove extra spaces
-- ================================================
SELECT 'Hi     There,  today   is great'  AS BEFORE
     , REPLACE(REPLACE(REPLACE('Hi     There,  today   is great', '  ', ' '), '  ', ' '), '  ', ' ') AS FUNCTION_AFTER
     , REGEXP_REPLACE('Hi     There,  today   is great', '[ ]{2,}', ' ') AS REGEXPR_AFTER
  FROM dual;
```

Using the REGEXP_SUBSTR Function

- ✓ Purpose: Allows you to extract a substring from a string
- ✓ So REGEXP_SUBSTR is very similar to the SUBSTR condition
- ✓ The difference is that the REGEXP_SUBSTR uses powerful regular expression matching instead of the simple pattern matching of the SUBSTR function.
- ✓ Used in SELECT and WHERE clause

REGEXP_SUBSTR Parameters

Syntax: REGEXP_REPLACE(<string>, <pattern>, <str_to_replace>, <position>, <occurence>, <match_parameter>)

string: Text to be string

pattern: The regular expression that we are searching for in the string

Optional Options

position: Position (number) to start searching

occurrence: Which occurrence to be changed

match Parameters:

1. 'c': Case sensitivity matching (default)
2. 'i': No-case sensitive matching
3. 'n': Allows match any character operator
4. 'm': Treats source string as multiple lines
5. 'x': Whitespace characters are ignored. By default whitespace characters are matched like any other characters

Special Notes for REGEXP_SUBSTR

- ✓ If there are conflicting values provided for the match parameters, the function will use the last value
- ✓ If you omit a match parameters, the function will use the NLS_SORT parameter to determine case-sensitivity search. It will assume single line string

Using Cases for REGEXP_SUBSTR Function

- ✓ Find the first vowel in the name
- ✓ Find the 2nd occurrence of the #2 in the string
- ✓ Find the 1st occurrence of a 3 digit string
- ✓ Find the first word surrounds by spaces

Use Cases – REGEXP_SUBSTR w/SQL

```
-- ================================================
-- Find the first vowel in the name
-- ================================================
SELECT first_name
--     , INSTR(first_name, 'a', 1) instr_function
     , SUBSTR(first_name, INSTR(first_name, 'a', 1), 1) AS function_first_vowel_in_name
     , SUBSTR(first_name,
              CASE INSTR(first_name, 'a', 1)
                 WHEN 0 THEN LENGTH(first_name) + 1
                 ELSE INSTR(first_name, 'a', 1)
              END,
              1) AS function_first_vowel_in_name_2
     , REGEXP_SUBSTR(first_name, 'a', 1, 1, 'i') AS regexpr_first_name_a
     , REGEXP_SUBSTR(first_name, 'a|e|i|o|u', 1, 1, 'i') AS regexpr_first_name_any_vowel
  FROM EMPLOYEES;
```

```
-- ================================================
-- Find the 2nd occurrence of the #2 in the string
-- ================================================
SELECT 'I earned $200 in 2 hours!'  AS BEFORE
     , SUBSTR('I earned $200 in 2 hours!', INSTR('I earned $200 in 2 hours!', '2 h'), 3) AS function_try_1_AFTER
     , REGEXP_SUBSTR('I earned $200 in 2 hours!', '2 h', 2) AS regexpr_AFTER
  FROM dual;
```

Use Cases – REGEXP_SUBSTR w/SQL

```
-- ================================================
-- Find the 1st occurrence of a 3 digit number in the string
-- ================================================
SELECT 'I earned $200 in 2 hours!'  AS BEFORE
     , SUBSTR('I earned $200 in 2 hours!', INSTR('I earned $200 in 2 hours!', '200'), 3) AS function_3_digit_word
     , REGEXP_SUBSTR('I earned $200 in 2 hours!', '(\d)(\d)(\d)') AS regexpr_3_digit_word
  FROM dual;


-- ================================================
-- Looking for the first word with spaces around it
-- ================================================
SELECT 'Hello my coding language is SQL'  AS BEFORE
     , INSTR('Hello my coding language is SQL', ' ', 1) first_occurrence
     , INSTR('Hello my coding language is SQL', ' ', 1, 2) second_occurrence
     , SUBSTR(
              'Hello my coding language is SQL',   -- Target String
              INSTR('Hello my coding language is SQL', ' ', 1),   -- first occurrence of a space
              INSTR('Hello my coding language is SQL', ' ', 1, 2) - INSTR('Hello my coding language is SQL', ' ',
1) -- second occurence, minus first occurrence
             ) AS FUNCTION_AFTER
     , REGEXP_SUBSTR('Hello my coding language is SQL', ' [^ ]+ ') AS REGEXPR_AFTER
  FROM dual;
```

Using the REGEXP_INSTR Function

- ✓ Use REGEXP_INSTR for matching a regular expression with our string
- ✓ REGEXP_INSTR is very similar to the INSTR condition
- ✓ The difference is REGEXP_INSTR uses powerful regular expression matching instead of simple pattern matching in INSTR.
- ✓ Used in SELECT and WHERE clause

REGEXP_INSTR Parameters

Syntax: REGEXP_INSTR(<string>, <pattern>, <position>, <occurence>, <match_parameter>)

string: Text to be string

pattern: The regular expression that we are searching for in the string

Optional Options

position: Position (number) to start searching

occurrence: Which occurrence to be returned

return_option: 0 (current position) – 1 (one position to the right)

match Parameters:

1. ‘c’: Case sensitivity matching (default)
2. ‘i’: No-case sensitive matching
3. ‘n’: Allows match any character operator
4. ‘m’: Treats source string as multiple lines
5. ‘x”; Whitespace characters are ignored.

Special Notes for REGEXP_INSTR

- ✓ If there are conflicting values provided for the match parameters, the function will use the last value
- ✓ If you omit a match parameters, the function will use the NLS_SORT parameter to determine case-sensitivity search. It will assume single line string
- ✓ If the function does not find a match in the pattern, it will return 0.

Use Cases for REGEXPR_INSTR

- ✓ Find the first vowel
- ✓ View position of target and position after the target
- ✓ Find the first digit in a string
- ✓ Find the first word with spaces surrounding it.
- ✓ Search for roles

Use Case REGEXP_INSTR w/SQL

```
-- ================================================
-- Find first vowel position
-- ================================================
SELECT first_name
     , REGEXP_INSTR(first_name, 'a|e|i|o|u') AS first_name_vowel_position
  FROM EMPLOYEES;

-- ================================================
-- Vowel position of target - 0
-- Vowel next position of target - 1
-- ================================================
SELECT first_name
     , REGEXP_INSTR(first_name, 'a|e|i|o|u', 1, 1, 0) AS actual_position_of_vowel
     , REGEXP_INSTR(first_name, 'a|e|i|o|u', 1, 1, 1) AS next_position_after_vowel
  FROM EMPLOYEES;

-- ================================================
--  Find the first digit
-- ================================================
SELECT 'I earned $200 in 2 hours!' String
     , REGEXP_INSTR('I earned $200 in 2 hours!', '\d') actual_of_1st_digit
     , REGEXP_INSTR('I earned $200 in 2 hours!', '\d', 1, 1, 1) next_position_after_1st_digit
  FROM dual;
```

Use Case REGEXP_INSTR w/SQL

```
-- ================================================
-- Looking the first word with spaces around it
-- ================================================
SELECT 'Hello my name is Scott' Target_String
     , REGEXP_INSTR('Hello my name is Scott', ' [^ ]+ ') RegExpr_for_1st_occurrence
     , REGEXP_INSTR('Hello my name is Scott', ' [^ ]+ ', 1, 2) RegExpr_for_2nd_occurrence
  FROM dual;

SELECT 'Hello my name is Scott' Target_String
     , REGEXP_INSTR('Hello my name is Scott', '(\s)(\S*)(\s)') RegExpr_for_1st_occurence
     , REGEXP_INSTR('Hello my name is Scott', '(\s)(\S*)(\s)', 1, 2) RegExpr_for_2nd_occurence
  FROM dual;

-- ================================================
-- Look for VP's
-- ================================================
SELECT *
  FROM EMPLOYEES
 WHERE REGEXP_INSTR(job_id, 'VP') > 0
;
```

Using the REGEXP_COUNT Function

- ✓ The REGEXP_COUNT function will count the number of times that a pattern occurs in a string
- ✓ Used in SELECT and WHERE clause

REGEXP_COUNT Parameters

Syntax: REGEXP_INSTR(<string>, <pattern>, <position>, <occurence>, <match_parameter>)

string: Text to be string

pattern: The regular expression that we are searching for in the string

Optional Options

position: Position (number) to start searching

match Parameters:

1. ‘c’: Case sensitivity matching (default)
2. ‘i’: No-case sensitive matching
3. ‘n’: Allows match any character operator
4. ‘m’: Treats source string as multiple lines
5. ‘x’: Whitespace characters are ignored.

Special Notes for REGEXP_COUNT

- ✓ If there are conflicting values provided for the match parameters, the function will use the last value
- ✓ If you omit a match parameters, the function will use the NLS_SORT parameter to determine case-sensitivity search. It will assume single line string
- ✓ If the function does not find a match in the pattern, it will return 0.

Use Cases for REGEXPR_COUNT

- ✓ Count occurrences of separators in the phone number
- ✓ Count occurrences of vowels in the first name
- ✓ Count occurrences of digits
- ✓ Count occurrences of 2 or more spaces together

Use Case REGEXP_INSTR w/SQL

```
-- ================================================
-- How many separators are there in the phone number
-- ================================================
SELECT phone_number
     , REGEXP_COUNT(phone_number, '\.') AS regexpr_phone
  FROM EMPLOYEES;

-- ================================================
-- Count vowels in the first name
-- ================================================
SELECT first_name
     , REGEXP_COUNT(first_name, 'a|e|i|o|u', 1, 'c') AS firstname_lwr_case_vowels_cnt
     , REGEXP_COUNT(first_name, 'a|e|i|o|u', 1, 'i') AS firstname_no_case_vowels_cnt
  FROM EMPLOYEES;

-- ================================================
-- work on digits
-- ================================================
SELECT 'I earned $200 in 2 hours!'  AS BEFORE
     , REGEXP_COUNT('I earned $200 in 2 hours!', '(\d)') AS single_digit_count
     , REGEXP_COUNT('I earned $200 in 2 hours!', '(\d)(\d)') AS double_digit_count
     , REGEXP_COUNT('I earned $200 in 2 hours!', '(\d)(\d)(\d)') AS triple_digit_count
  FROM dual;

-- ================================================
-- Count of more that 2 spaces together
-- ================================================
SELECT 'Hi     There,  today   is great'  AS BEFORE
     , REGEXP_COUNT('Hi     There,  today   is great', '[ ]{2,}', 1) AS REGEXPR_AFTER
  FROM dual;
```

Q & A

THANK YOU

COTIVITI