
ORACLE
ACADEMY

Database Programming with SQL

17-3
Regular Expressions

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Describe regular expressions
- Use regular expressions to search, match, and replace strings in SQL statements
- Construct and execute regular expressions and check constraints

Purpose

- Sometimes you have to find or replace a particular piece of text in a column, text string, or document.
- You already know how to perform simple pattern matching using LIKE and wildcards.
- Sometimes you might need to look for very complex text strings such as extracting all URLs from a piece of text.
- Other times you might be asked to do a more complex search such as finding all words whose every second character is a vowel.

Purpose

- Regular expressions are a method of describing both simple and complex patterns for searching and manipulating.
- They are used widely in the computing industry, and are not limited to Oracle.
- Oracle's implementation of regular expressions is an extension of the POSIX (Portable Operating System for UNIX) and are, as such, fully compatible with the POSIX standard, as controlled by the Institute of Electrical and Electronics Engineers (IEEE).

Regular Expressions

- The use of regular expressions is based on the use of meta characters.
  - Meta characters are special characters that have a special meaning, such as a wildcard character, a repeating character, a non-matching character, or a range of characters.
  - You can use several predefined meta character symbols in the pattern matching.
  - The next slides list some of the meta characters and provide a brief explanation of each.

META Characters

Symbol	Description
. (dot)	Matches any character in the supported character set, except NULL
?	Matches zero or one occurrence
*	Matches zero or more occurrences
+	Matches one or more occurrences
()	Grouping expression, treated as a single sub-expression
\	Escape character
|	Alternation operator for specifying alternative matches
^/$	Matches the start-of-line/end-of-line
[ ]	Bracket expression for a matching list matching any one of the expressions represented in the list

Regular Expression Examples

- A simple regular expression is very similar to the wildcard searches you are already familiar with.
- Let's take a look at an example: let's use the dot operator to look for the letter 'a' followed by any one character followed by the letter 'c'.
- As a regular expression, this would be done as: 'a.c'.
- The same expression as a standard SQL wildcard search would be: WHERE column LIKE 'a_c'.

Regular Expression Examples

- Which of the following strings would match 'a.c'?
  1. 'ABC',
  2. 'abc',
  3. 'aqx',
  4. 'axc',
  5. 'aBc',
  6. 'abC'
  7. 'Amc'
  8. 'amrc'

Regular Expression Examples

- The strings in red would match the search string 'a.c'
- 'ABC', 'abc', 'aqx', 'axc', 'aBc', 'abC' 'Amc' 'amrc'
- The other examples fail either due to their having the character in the wrong position or in the wrong case (uppercase not lowercase as specified in the search string).

Regular Expression Functions

- Oracle provides a set of SQL functions that you can use to search and manipulate strings using regular expressions.
- You can use these functions on any data type that holds character data such as CHAR, CLOB, and VARCHAR2.
- A regular expression must be enclosed in single quotation marks.

Regular Expression Functions

Name	Description
REGEXP_LIKE	Similar to the LIKE operator, but performs regular expression matching instead of simple pattern matching
REGEXP_REPLACE	Searches for a regular expression pattern and replaces it with a replacement string
REGEXP_INSTR	Searches for a given string for a regular expression pattern and returns the position where the match is found
REGEXP_SUBSTR	Searches for a regular expression pattern within a given string and returns the matched substring
REGEXP_COUNT	Returns the number of times a pattern appears in a string. You specify the string and the pattern. You can also specify the start position and matching options (for example, c for case sensitivity).

Regular Expression Function Examples

- Assume you were asked to list all employees with a first name of Stephen or Steven.

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE(first_name, '^Ste(v|ph)en$');
```

- With regular expressions, you could simply use the REGEXP_LIKE function and the search string: '^Ste(v|ph)en$'
  - "^" specifies the start of the string that is being searched
  - Uppercase "S", followed by
  - lowercase "t", followed by
  - lowercase "e", followed by

Regular Expression Function Examples

- With regular expressions, you could simply use the REGEXP_LIKE function and the search string: '^Ste(v|ph)en$'
  - "(" starts a sub-expression
  - lowercase "v"
  - "|" specifies an OR
  - lowercase "p" followed by Lowercase "h"
  - ")" finishes the group of choices,
  - lowercase "e"
  - lowercase "n"
  - "$" specifies the end of the string that is being searched

Regular Expression Function Examples

- Example:

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE(first_name, '^Ste(v|ph)en$');
```

- Result:

FIRST_NAME	LAST_NAME
Steven	King

Regular Expression Function Examples

- Regular expression REPLACE function will replace one string pattern with another.
- This example looks for an "H" followed by any vowel, and replaces them with two "*" symbols.

```
SELECT last_name, REGEXP_REPLACE(last_name, '^H(a|e|i|o|u)', '**')
  AS "Name changed"
FROM employees;
```

LAST_NAME	Name changed
Hunold	**nold
Ernst	Ernst
Davies	Davies
Lorentz	Lorentz
…	…

Regular Expression Function Examples

- Regular expression COUNT function returns the number of times a pattern appears in a string.
- This example searches for the subexpression "ab"

```
SELECT country_name, REGEXP_COUNT(country_name, '(ab)')
   AS "Count of 'ab'"
FROM wf_countries
WHERE REGEXP_COUNT(country_name, '(ab)')>0;
```

COUNTRY_NAME	Count of 'ab'
Kingdom of Saudi Arabia	1
Republic of Zimbabwe	1
Arab Republic of Egypt	1
Great Socialist Peoples Libyan Arab Jamahiriya	1
Syrian Arab Republic	1
Gabonese Republic	1
United Arab Emirates	1

Regular Expressions in Check Constraints

- Regular expressions could also be used as part of the application code to ensure that only valid data is stored in the database.
- It is possible to include a call to a regular expression function in, for instance, a CHECK constraint.

Regular Expressions in Check Constraints

- So if you want to ensure that no email addresses without '@' were captured in a table in your database, you could simply add the following check constraint:

```
ALTER TABLE employees
ADD CONSTRAINT email_addr_chk
CHECK(REGEXP_LIKE(email,'@'));
```

- This would ensure that all email addresses include an "@" sign.

Regular Expressions in Check Constraints

- Using Regular expressions, you can check the format of email addresses more thoroughly to check they are valid.
- A valid email address would have one or more characters then an @, followed by one or more characters then a . (dot), followed by one or more characters.

```
CREATE TABLE my_contacts
(first_name VARCHAR2(15),
 last_name VARCHAR2(15),
 email VARCHAR2(30) CHECK(REGEXP_LIKE(email, '.+@.+\..+')));
```

Regular Expressions in Check Constraints

- Syntax definitions:
  - .+ means one or more characters
  - @ an @ symbol
  - \. a . (a dot) (here the backslash is an escape character)

```
CREATE TABLE my_contacts
(first_name VARCHAR2(15),
 last_name VARCHAR2(15),
 email VARCHAR2(30) CHECK(REGEXP_LIKE(email, '.+@.+\..+')));
```

Terminology

Key terms used in this lesson included:

- REGULAR EXPRESSIONS
- META Characters
- Subexpressions

Summary

In this lesson, you should have learned how to:

- Describe regular expressions
- Use regular expressions to search, match, and replace strings in SQL statements
- Construct and execute regular expressions and check constraints


ORACLE
ACADEMY