7

Regular Expression Support

Objectives

After completing this lesson, you should be able to do the following:

- List the benefits of using regular expressions
- Use regular expressions to search for, match, and replace strings

Lesson Agenda

- Introduction to regular expressions
- Using metacharacters with regular expressions
- Using the regular expressions functions:
  - `REGEXP_LIKE`
  - `REGEXP_REPLACE`
  - `REGEXP_INSTR`
  - `REGEXP_SUBSTR`
- Accessing subexpressions
- Using the `REGEXP_COUNT` function
- Regular expressions and check constraints

What Are Regular Expressions?

- You use regular expressions to search for (and manipulate) simple and complex patterns in string data by using standard syntax conventions.
- You use a set of SQL functions and conditions to search for and manipulate strings in SQL and PL/SQL.
- You specify a regular expression by using:
  - Metacharacters, which are operators that specify the search algorithms
  - Literals, which are the characters for which you are searching

Benefits of Using Regular Expressions

Regular expressions enable you to implement complex match logic in the database with the following benefits:

- By centralizing match logic in Oracle Database, you avoid intensive string processing of SQL results sets by middle-tier applications.
- Using server-side regular expressions to enforce constraints, you eliminate the need to code data validation logic on the client.
- The built-in SQL and PL/SQL regular expression functions and conditions make string manipulations more powerful and easier than in previous releases of Oracle Database 11*g*.

Using the Regular Expressions Functions and Conditions in SQL and PL/SQL

Function or Condition Name	Description
REGEXP_LIKE	Is similar to the LIKE operator, but performs regular expression matching instead of simple pattern matching (condition)
REGEXP_REPLACE	Searches for a regular expression pattern and replaces it with a replacement string
REGEXP_INSTR	Searches a 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 extracts the matched substring
REGEXP_COUNT	Returns the number of times a pattern match is found in an input sting

Lesson Agenda

- Introduction to regular expressions
- Using metacharacters with regular expressions
- Using the regular expressions functions:
  - `REGEXP_LIKE`
  - `REGEXP_REPLACE`
  - `REGEXP_INSTR`
  - `REGEXP_SUBSTR`
- Accessing subexpressions
- Using the `REGEXP_COUNT` function

What Are Metacharacters?

- Metacharacters are special characters that have a special meaning such as a wildcard, a repeating character, a nonmatching character, or a range of characters.
- You can use several predefined metacharacter symbols in the pattern matching.
- For example, the `^(f|ht)tps?:$` regular expression searches for the following from the beginning of the string:
  - The literals `f` or `ht`
  - The `t` literal
  - The `p` literal, optionally followed by the `s` literal
  - The colon “`:`” literal at the end of the string

Using Metacharacters with Regular Expressions

Syntax	Description
.	Matches any character in the supported character set, except NULL
+	Matches one or more occurrences
?	Matches zero or one occurrence
*	Matches zero or more occurrences of the preceding subexpression
{m}	Matches exactly *m* occurrences of the preceding expression
{m, }	Matches at least *m* occurrences of the preceding subexpression
{m,n}	Matches at least *m*, but not more than *n*, occurrences of the preceding subexpression
[...]	Matches any single character in the list within the brackets
|	Matches one of the alternatives
(...)	Treats the enclosed expression within the parentheses as a unit. The subexpression can be a string of literals or a complex expression containing operators.

Using Metacharacters with Regular Expressions

Syntax	Description
^	Matches the beginning of a string
$	Matches the end of a string
\	Treats the subsequent metacharacter in the expression as a literal
\n	Matches the *n*th (1–9) preceding subexpression of whatever is grouped within parentheses. The parentheses cause an expression to be remembered; a backreference refers to it.
\d	A digit character
[:class:]	Matches any character belonging to the specified POSIX character class
[^:class:]	Matches any single character *not* in the list within the brackets

Lesson Agenda

- Introduction to regular expressions
- Using metacharacters with regular expressions
- Using the regular expressions functions:
  - `REGEXP_LIKE`
  - `REGEXP_REPLACE`
  - `REGEXP_INSTR`
  - `REGEXP_SUBSTR`
- Accessing subexpressions
- Using the `REGEXP_COUNT` function

Regular Expressions Functions and Conditions: Syntax

```
REGEXP_LIKE    (source_char, pattern [,match_option]
```

```
REGEXP_INSTR   (source_char, pattern [, position
                [, occurrence [, return_option
                [, match_option [, subexpr]]]]])
```

```
REGEXP_SUBSTR (source_char, pattern [, position
                [, occurrence [, match_option
                [, subexpr]]]])
```

```
REGEXP_REPLACE(source_char, pattern [,replacestr
                 [, position [, occurrence
                 [, match_option]]]])
```

```
REGEXP_COUNT (source_char, pattern [, position
                [, occurrence [, match_option]]])
```

Performing a Basic Search by Using the REGEXP_LIKE Condition

```
REGEXP_LIKE(source_char, pattern [, match_parameter ])
```

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

	FIRST_NAME	LAST_NAME
1	Steven	King
2	Steven	Markle
3	Stephen	Stiles

Replacing Patterns by Using the REGEXP_REPLACE Function

```
REGEXP_REPLACE(source_char, pattern [,replacestr
[, position [, occurrence [, match_option]]]])
```

```
SELECT REGEXP_REPLACE(phone_number, '\.','-') AS phone
FROM employees;
```

Original

	LAST_NAME	PHONE
1	OConnell	650.507.9833
2	Grant	650.507.9844
3	Whalen	515.123.4444
4	Hartstein	515.123.5555

Partial results

	LAST_NAME	PHONE
1	OConnell	650-507-9833
2	Grant	650-507-9844
3	Whalen	515-123-4444
4	Hartstein	515-123-5555

Finding Patterns by Using the REGEXP_INSTR Function

```
REGEXP_INSTR  (source_char, pattern [, position [,
   occurrence [, return_option [, match_option]]]])
```

```
SELECT street_address,
REGEXP_INSTR(street_address,'[[:alpha:]]') AS
   First_Alpha_Position
FROM  locations;
```

	STREET_ADDRESS	FIRST_ALPHA_POSITION
1	1297 Via Cola di Rie	6
2	93091 Calle della Testa	7
3	2017 Shinjuku-ku	6
4	9450 Kamiya-cho	6

Extracting Substrings by Using the REGEXP_SUBSTR Function

```
REGEXP_SUBSTR (source_char, pattern [, position
                [, occurrence [, match_option]]])
```

```
SELECT REGEXP_SUBSTR(street_address , ' [^ ]+ ') AS Road
FROM locations;
```

	ROAD
1	Via
2	Calle
3	(null)
4	(null)
5	Jabberwocky

Lesson Agenda

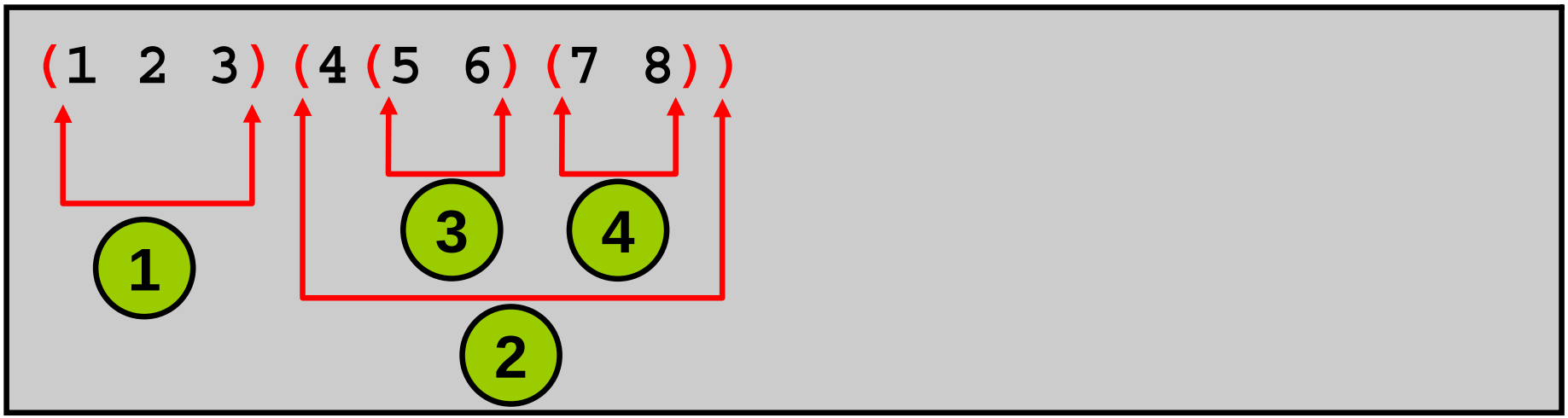
- Introduction to regular expressions
- Using metacharacters with regular expressions
- Using the regular expressions functions:
  - `REGEXP_LIKE`
  - `REGEXP_REPLACE`
  - `REGEXP_INSTR`
  - `REGEXP_SUBSTR`
- Accessing subexpressions
- Using the `REGEXP_COUNT` function

Subexpressions

Examine this expression:

```
(1 2 3)(4(5 6)(7 8))
```

The subexpressions are:



Using Subexpressions with Regular Expression Support

```
SELECT
  REGEXP_INSTR
① ('0123456789',       -- source char or search value
② '(123)(4(56)(78))', -- regular expression patterns
③ 1,                   -- position to start searching
④ 1,                   -- occurrence
⑤ 0,                   -- return option
⑥ 'i',                 -- match option (case insensitive)
⑦ 1)                -- sub-expression on which to search
    "Position"
FROM dual;
```

	Position
1	2

Why Access the *n*th Subexpression?

- A more realistic use: DNA sequencing
- You may need to find a specific subpattern that identifies a protein needed for immunity in mouse DNA.

```
SELECT
    REGEXP_INSTR('ccacctttccctccactcctcacgttctcacctgtaaagcgtccctc
    cctcatccccatgccccttaccctgcagggtagagtaggctagaaaccagagagctccaagc
    tccatctgtggagaggtgccatccttgggctgcagagagaggagaatttgccccaaagctgcc
    tgcagagcttcaccacccttagtctcacaaagccttgagttcatagcatttcttgagttttca
    ccctgcccagcaggacactgcagcacccaaagggcttcccaggagtagggttgccctcaagag
    gctcttgggtctgatggccacatcctggaattgttttcaagttgatggtcacagccctgaggc
    atgtaggggcgtggggatgcgctctgctctgctctcctctcctgaacccctgaacccctctggc
    taccccagagcacttagagccag',
                    '(gtc(tcac)(aaag))',
                     1, 1, 0, 'i',
                     1) "Position"
FROM dual;
```

	Position
1	195

REGEXP_SUBSTR: Example

```
SELECT
  REGEXP_SUBSTR
①('acgctgcactgca', -- source char or search value
②  'acg(.*)gca',    -- regular expression pattern
③  1,               -- position to start searching
④  1,               -- occurrence
⑤  'i',            -- match option (case insensitive)
⑥  1)               -- sub-expression
 "Value"
FROM dual;
```

Value
1 ctgcact

Lesson Agenda

- Introduction to regular expressions
- Using metacharacters with regular expressions
- Using the regular expressions functions:
  - `REGEXP_LIKE`
  - `REGEXP_REPLACE`
  - `REGEXP_INSTR`
  - `REGEXP_SUBSTR`
- Accessing subexpressions
- Using the `REGEXP_COUNT` function

Using the REGEXP_COUNT Function

```
REGEXP_COUNT (source_char, pattern [, position
                [, occurrence [, match_option]]])
```

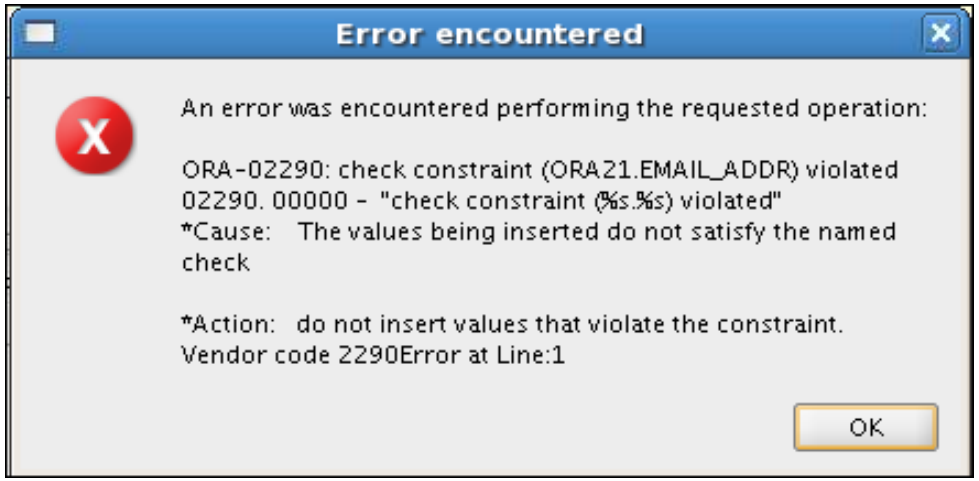
```
SELECT REGEXP_COUNT(
    'ccacctttccctccactcctcacgttctcacctgtaaagcgtccctccctcatccccatgccccttaccctgcag
    ggtagagtaggctagaaaccagagagctccaagctccatctgtggagaggtgccatccttgggctgcagagagaggag
    aatttgccccaaagctgcctgcagagcttcaccacccttagtctcacaaagccttgagttcatagcatttcttgagtt
    ttcaccctgcccagcaggacactgcagcacccaaagggcttcccaggagtagggttgccctcaagaggctcttgggtc
    tgatggccacatcctggaattgttttcaagttgatggtcacagccctgaggcatgtaggggcgtggggatgcgctctg
    ctctgctctcctctcctgaacccctgaacctctggctaccccagagcacttagagccag',
    'gtc') AS Count
FROM dual;
```

	COUNT
1	4

Regular Expressions and Check Constraints: Examples

```
ALTER TABLE emp8
  ADD CONSTRAINT email_addr
  CHECK(REGEXP_LIKE(email,'@')) NOVALIDATE;
```

```
INSERT INTO emp8 VALUES
  (500,'Christian','Patel','ChrisP2creme.com',
   1234567890,'12-Jan-2004','HR_REP',2000,null,102,40);
```



Quiz

With the use of regular expressions in SQL and PL/SQL, you can:

1. Avoid intensive string processing of SQL result sets by middle-tier applications
2. Avoid data validation logic on the client
3. Enforce constraints on the server

Summary

In this lesson, you should have learned how to use regular expressions to search for, match, and replace strings.

Practice 7: Overview

This practice covers using regular expressions functions to do the following:

- Searching for, replacing, and manipulating data
- Creating a new `CONTACTS` table and adding a `CHECK` constraint to the `p_number` column to ensure that phone numbers are entered into the database in a specific standard format
- Testing the adding of some phone numbers into the `p_number` column by using various formats