

Oracle 10G Regular Expressions Lecture

Lev Molytner
 May 1, 2004

Regular Expressions

- Used to search and manipulate text data
 - o Many useful applications (eg. search engines)
 - o Supported by many word processors and programming languages
 - Eg. shell, awk, perl, and java
- Use special characters to represent chunks of text
 - o For example, '[0-9]?[0-9]:[0-9]:[0-9][0-9] [A|a|P|p][M|m]' represents a time string
 - Covers string like '12:01:01 AM'

Regular Expressions in Oracle

- Oracle Database 10g includes support for IEEE/POSIX standard native regular expressions in SQL
 - o Compatible with other programming environments such as Unix, perl and Java
 - o Part of the database SQL engine
- Oracle 9i contains OWA_PATTERN package to support regular expressions
 - o Not compatible with POSIX regular expressions and has many limitations
- Oracle has implemented regular expressions via the following four new SQL functions
 - o REGEXP_LIKE
 - o REGEXP_INSTR
 - o REGEXP_SUBSTR
 - o REGEXP_REPLACE
- Regular Expression Operators are required to support these functions

Commonly Used Regular Expression Operators

| Value | Description | Example |
|-------|---|--|
| . | Any character (except NULL) | a. - matches 'a' one time |
| * | Zero or more occurrences | a* - matches 'a' zero or more times |
| + | One or more occurrences | A+ - matches 'a' one or more times |
| ? | Zero or one occurrence | A? - matches 'a' zero or one time |
| {m} | Exactly m occurrences | a{3} - matches 'aaa' |
| | Alternation operator | a b - matches either 'a' or 'b' |
| \$ | End-of-line character | b\$ - matches lines ending with 'b' |
| ^ | Beginning-of-line character | ^a - matches lines beginning with 'a' |
| [] | Any character specified within brackets | [abc] - matches either 'a' or 'b' or 'c' |
| [^] | All except the listed characters | [^ab] - matches any character except 'a' and 'b' |

REGEXP_LIKE Function

- Applies a LIKE function to a regular expression pattern
- Simplified Syntax:


```
REGEXP_LIKE (source string, pattern)
```
- Rules:
 - o Source string specifies source data to be scanned
 - o Pattern is the regular expression to search within the source string
 - o Returns true or false indicating whether the pattern matched the data.
 - o Used primarily in the WHERE clause

REGEXP_LIKE Example

- Assume we have a postal code column with A9A-9A9 format for Canadian companies
- Get all companies in Canada with a valid postal code

```
SELECT company_name
FROM company
WHERE REGEXP_LIKE(postal_code, '^[A-Z][0-9][A-Z][ |-][0-9][A-Z][ 0-9]$');
```

REGEXP_INSTR Function

- Returns the beginning position of the pattern within the string
 - o Extension to the INSTR function but it does not work from the end of the string
- Simplified Syntax:


```
REGEXP_INSTR (source string, pattern
               [,start position [,occurrence ]])
```
- Rules:
 - o Source string specifies source data to be scanned
 - o Pattern is the regular expression to search within the source string
 - o Start position specifies position within source string where search should begin. (default is 1)
 - o Occurrence indicates which occurrence to search for (default is 1)
 - o Returns beginning position of the pattern within the string

REGEXP_INSTR Example 1

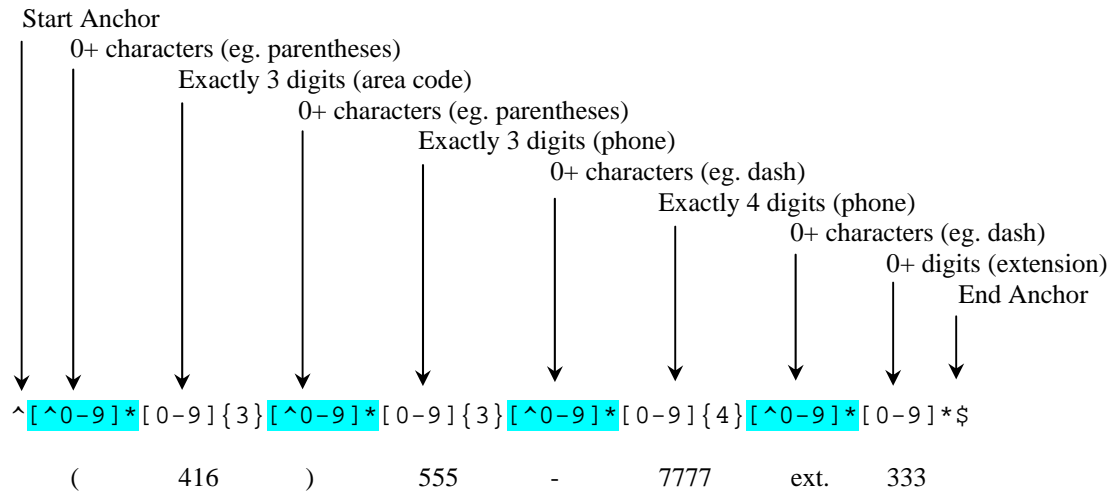
- Assume student tel_no column follows the following format:
 - Area code - first three digits
 - Could be in parenthesis or have a space or a dash after it
 - Phone number - next seven digits
 - Could have a space or a hyphen after the first three digits
 - Extension - remaining digits
 - Could be 3, 4 or 5 digits and could be prefixed by characters
- Examples: (415) 5763 217, 8881219912 121, 415-576-3223 ext.121

- Define a regular expression to describe phone numbers

```
'^[^0-9]*[0-9]{3}[^0-9]*[0-9]{3}[^0-9]*[0-9]{4}[^0-9]*[0-9]*$'
```

| | |
|----------|-------------------------------------|
| ^ and \$ | anchors at each end |
| [^0-9]* | Zero or more non-numeric characters |
| [0-9]{3} | Exactly three numeric digits |

REGEXP_INSTR Example 1 (Continued)



REGEXP_INSTR Example 2

- Select all students who have an extension
 - o To ensure that the extensions has at least 1 digit, change the last * to +

```
SELECT student_lname, tel_no
FROM student
WHERE REGEXP_INSTR(tel_no,
    '^[^0-9]*[0-9]{3}[^0-9]*[0-9]{3}[^0-9]*[0-9]{4}[^0-9]*[0-9]+$') <> 0;
```

| STUDENT_LNAM | TEL_NO |
|--------------|-----------------------|
| BROWN | (617)342-2345 ext.232 |
| COX | 619 433 6845 x.123 |
| GIBBS | (714)346-2896 42121 |

REGEXP_INSTR Example 3

- Select all students whose area code is 212
 - o Use REGEXP_INSTR to pass the starting location of the area code to the SUBSTR function
 - This assumes that the first three digits are area code

```
SELECT student_lname, tel_no
FROM student
WHERE SUBSTR(tel_no,
    ,REGEXP_INSTR(tel_no, '[0-9]{3}')
    , 3) = 212;
```

| STUDENT_LNAM | TEL_NO |
|--------------|----------------------|
| TYLER | 212 444 9769 |
| GREEN | (212)342-9621 |
| FRANKS | (212)234-8521 x.1231 |

REGEXP_SUBSTR Function

- Searches for the pattern and returns the matched portion of the string
- Simplified Syntax:


```
REGEXP_SUBSTR (source string, pattern
                [,start position [,occurrence]])
```
- Rules:
 - o Same parameters as REGEXP_INSTR
 - o Returns matched portion of the pattern from the string

REGEXP_SUBSTR Example

- Determine the number of street addresses contain the words STREET vs WAY

```
SELECT REGEXP_SUBSTR(street,'STREET|WAY') type
       ,COUNT(*) cnt
FROM person_address
WHERE REGEXP_INSTR(street,'STREET|WAY') > 0
GROUP BY REGEXP_SUBSTR(street,'STREET|WAY');
```

| TYPE | CNT |
|--------|-----|
| STREET | 5 |
| WAY | 4 |

REGEXP_REPLACE Function

- Searches a pattern in the string and replaces the matched string with the supplied replacement pattern
- Simplified Syntax:


```
REGEXP_REPLACE (source string, pattern
                [,replace string [,start position [,occurrence]]])
```
- Rules:
 - o Same parameters as REGEXP_SUBSTR with one extra parameter
 - o Replace string is the regular expression to replace the matched portion within the source string
 - o Returns replaced string

REGEXP_REPLACE Function (continued)

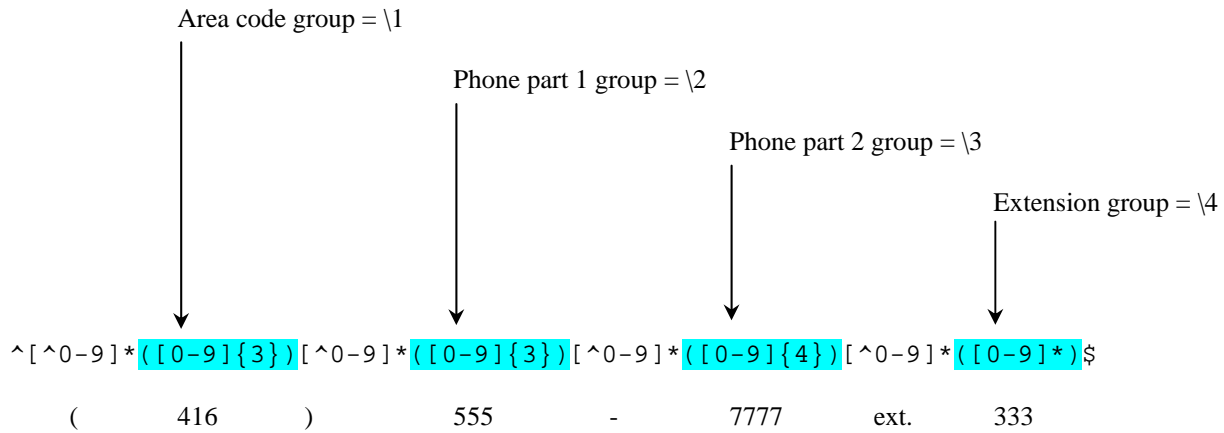
- Two new regular expression operators are useful with replace function
 - o Use (..) to define groups in the patter
 - o Use \n to the matched group characters

| Operator | Description | Example |
|----------|---|---|
| (..) | Group operator to identify sub-expression | (ab) - ab characters can be referenced as a group |
| \n | Matches the nth group | \2 - matches the second group |

REGEXP_REPLACE Example

- Display tel_no column as phone number formatted as (999) 999-9999 and extension without any text

| Tel No Column | Phone Number | Extension |
|---------------------|----------------|-----------|
| 415 5763218 | (415) 576-3218 | |
| (415) 576 3217 | (415) 576-3217 | |
| 8881219912 ext. 121 | (888) 121-9912 | 121 |
| 415-576-3223 x.2345 | (415) 576-3223 | 2345 |



REGEXP_REPLACE Example (continued)

- Use groups \1, \2 and \3 to construct the phone number with parenthesis, spaces and dashes

```

SELECT student_lname
      ,REGEXP_REPLACE(tel_no
      ,'^[\w0-9]^*([0-9]{3})[\w0-9]^*([0-9]{3})[\w0-9]^*([0-9]{4})[\w0-9]^*([0-9]^*)$'
      ,'\(\1\) \2-\3') phone
      ,REGEXP_REPLACE(tel_no
      ,'^[\w0-9]^*([0-9]{3})[\w0-9]^*([0-9]{3})[\w0-9]^*([0-9]{4})[\w0-9]^*([0-9]^*)$'
      ,'\4') ext
FROM student;
  
```

| STUDENT_LNAM | PHONE | EXT |
|--------------|----------------|-------|
| BROWN | (617) 342-2345 | 232 |
| ADAMS | (213) 334-2789 | |
| COX | (619) 433-6845 | 123 |
| TYLER | (212) 444-9769 | |
| GIBBS | (714) 346-2896 | 42121 |
| ROSE | (415) 334-2345 | |
| NELSON | (609) 345-2346 | |
| CRICK | (415) 345-2313 | |

Hint: Regular expression functions can be used in DDL and PL/SQL where they behave the same way.

Case Insensitive Pattern Matching

- All 4 regular expression functions have one additional optional parameter called match
- Syntax:
 REGEXP_function(parms, match)
- Commonly used values for match parameter are
 - i - case insensitive searching
 - c - case sensitive searching

Example:

- Assume postal codes in our previous example can be in mixed case (eg. A9a-9b9)
- Get all companies in Canada with a valid postal code

```
SELECT company_name
FROM company
WHERE REGEXP_LIKE(postal_code, '^[A-Z][0-9][A-Z][ |-][0-9][A-Z][ 0-9]$', 'i');
```

Case Insensitive Sorting

- Use the NLS_SORT parameter to make sort of all queries in the session case-insensitive
 - o BINARY makes the sort case sensitive (default)
 - o BINARY_CI make the sort case insensitive

Syntax:

```
ALTER SESSION SET NLS_SORT = BINARY;
ALTER SESSION SET NLS_SORT = BINARY_CI;
```

- Case sensitive – lower case sorts after upper case
- Case insensitive – lower case is mixed with upper case

| Case Sensitive | Case Insensitive |
|----------------|------------------|
| DOE | DOE |
| SMITH | Jones |
| jones | SMITH |

Sorting Example

Case Sensitive

```
SELECT student_lname
FROM student
WHERE upper(student_lname) < 'D'
ORDER BY student_lname;
```

```
STUDENT_LNAM
-----
ADAMS
ARCHER
BROWN
CARTER
COOPER
CRICK
cox
```

Case Insensitive

```
ALTER SESSION SET NLS_SORT =
BINARY_CI;

SELECT student_lname
FROM student
WHERE upper(student_lname) < 'D'
ORDER BY student_lname;
```

```
STUDENT_LNAM
-----
ADAMS
ARCHER
BROWN
CARTER
COOPER
Cox
CRICK
```