## oracle：正则表达式

从 oracle database 10gsql 开发指南中 copy 的。

**正则表达式：**

本节介绍正则表达式及相关的 Oracle 数据库函数。使用这些函数可以在字符串中搜索字符模式。例如，假设有下列年份：

1965

1968

1971

1970

如果希望获得 1965 年和 1968 年之间的年份(包括 1965 年和 1968 年)，就可以使用下面的正则表达式实现这种功能：

^196[5-8]$

正则表达式中包含许多元字符(metacharacter)。在上面这个例子中，^、[5-8]以及$都是元字符。^可以匹配一个字符串的开头；[5-8]可以匹配介于 5~8 之间的数字；$可以匹配一个字符串的结尾。因此，^196 可以匹配以 196 开头的字符串；[5-8]$可以匹配以 5、6、7 或 8 结尾的字符串。而^196[5-8]$就可以匹配 1965、1966、1967 和 1968，这就是想要的结果。

在下面这个例子中，假设有如下字符串，其中引用了莎士比亚的《罗密欧与朱丽叶》中的一句台词：

But, soft! What light through yonder window breaks?

如果想查找子字符串 light，可以对引用的字符串应用下面的正则表达式：

l[[:alpha:]]{4}

在这个例子中，[[:alpha:]]和{4}都是元字符。[[:alpha:]]可以匹配 A-Z 或 a-z 之间的字符；{4}表示前面的匹配模式可以重复 4 次。当 l、[[:alpha:]]和{4}一起使用时，可以匹配以 l 开头的 5 个字母组成的序列。因此，当对这个字符串应用正则表达式 l[[:alpha:]]{4}时，就可以匹配子字符串 light。

表 4-7 列出了在正则表达式中可以使用的部分元字符，同时还给出了这些元字符的意思以及使用这些元字符的简单例子。

表 4-7　正则表达式中的元字符

| 元　字　符 | 意　　思 | 例　　子 |
| --- | --- | --- |
| \ | 说明要匹配的字符是一个特殊字符、常量或者反向引用。(反向引用重复上一次匹配。) | \n 匹配换行符<br>\\ 匹配 \<br>\( 匹配(<br>\) 匹配) |
| ^ | 匹配字符串的开头位 | 如果 A 是字符串中的第一个字 |

| | | |
|---|---|---|
| | 置 | 符，^A 匹配 A |
| $ | 匹配字符串的末尾位置 | 如果 B 是字符串中的最后一个字符，$B 匹配 B |
| * | 匹配前面的字符 0 次或多次。 | ba*rk 可以匹配 brk、bark、baark 等等 |
| + | 匹配前面的字符 1 次或多次。 | ba+rk 可以匹配 bark、baark 等等，但是不能匹配 brk |
| ? | 匹配前面的字符 0 次或 1 次 | ba?rk 只能匹配 brk 和 bark |
| {n} | 匹配一个字符恰好 n 次，其中 n 是一个整数 | hob{2}it 可以匹配 hobbit |

(续表)

| 元　字　符 | 意　　思 | 例　　子 |
|---|---|---|
| {n,m} | 匹配一个字符至少 n 次，最多 m 次，其中 n 和 m 都是整数 | hob{2,3}it 只能匹配 hobbit 和 hobbbit |
| . | 匹配除 null 之外的任意单个字符 | hob.it 可以匹配 hobait、hobbit 等等 |
| (pattern) | 匹配指定模式的一个子表达式。可以使用子表达式构成复杂的正则表达式。在这种子表达式中，可以访问单次的匹配，称为捕获(capture) | anatom(y\|ies) 可以匹配 anatomy 和 anatomies |
| x\|y | 匹配 x 或 y，其中 x 和 y 是一个或多个字符 | war\|peace 可以匹配 war 或 peace |
| [abc] | 匹配中括号内的任意一个字符 | [ab]bc 可以匹配 abc 和 bbc |

| | | |
|---|---|---|
| [a-z] | 匹配指定范围内的任意一个字符 | [a-c]bc 可以匹配 abc、bbc 和 cbc |
| [: :] | 指定一个字符类，可以匹配该类中的任何字符 | [:alphanum:] 可以匹配字符 0-9、A-Z 和 a-z<br>[:alpha:] 可以匹配字符 A-Z 和 a-z<br>[:blank:] 可以匹配空格或 tab 键<br>[:digit:] 可以匹配数字 0-9<br>[:graph:] 可以匹配非空字符<br>[:lower:] 可以匹配小写字母 a-z<br>[:print:] 与 [:graph:] 类似，不同之处在于[:print:] 包括空格字符<br>[:punct:] 可以匹配标点符号.,’’’等等<br>[:space:] 可以匹配所有的空白字符<br>[:upper:] 可以匹配所有的大写字母 A~Z<br>[:xdigit:] 可以匹配十六进制数字 0~9、A~F 和 a~f |
| [..] | 匹配一个组合元素，例如多字符元素 | 无 |
| [==] | 指定等价类 | 无 |
| \n | 这是对前一次捕获的一个反向引用，其中 n 是一个正整数 | (.)\1 可以匹配两个连续相同的字符。(.)可以匹配除 null 之外的任何单个字符，而 \1 则重复上次匹配的内容，即再次匹配相同的字符，因此可以匹配两个连续相同的字符 |
| | | |

Oracle Database10gRelease 2 新增加了很多类似于 Perl 的元字符，如表 4-8 所示。

表 4-8　类似于 Perl 的元字符

| 元　字　符 | 含　义 |
| --- | --- |
| \d | 数字字符 |
| \D | 非数字字符 |
| \w | 字母字符 |
| \W | 非字母字符 |
| \s | 空白字符 |
| \S | 非空白字符 |

(续表)

| 元　字　符 | 含　义 |
| --- | --- |
| \A | 只匹配字符串的开头位置 |
| \Z | 只匹配字符串的末尾位置或者字符串末尾的换行符之前的位置 |
| *? | 匹配前面的模式元素 0 次或多次 |
| +? | 匹配前面的模式元素 1 次或多次 |
| ?? | 匹配前面的模式元素 0 次或 1 次 |
| {n} | 匹配前面的模式元素恰好 n 次 |
| {n,} | 匹配前面的模式元素至少 n 次 |
| {n,m} | 匹配前面的模式元素至少 n 次，但不超过 m 次 |

表 4-9 列出了正则表达式函数。正则表达式函数是在 Oracle Database10g 中新增加的， Oracle Database11g 中又增加了一些条目，如下表所示。

表 4-9　正则表达式函数

| 函　　数 | 说　　明 |
| --- | --- |
| REGEXP_LIKE( x, pattern<br>[, match_option]) | 从 x 中搜索 pattern 参数中定义的正则表达式。可以使用 match_option 修改默认匹配选项，该参数可以被设置为：<br>• 'c'，说明在匹配时区分大小写(默认选项)<br>• 'I'，说明在匹配时不区分大小写<br>• 'n'，允许使用可以匹配任意字符的操作符<br>• 'm'，将 x 作为一个包含多行的字符串 |
| REGEXP_INSTR( x, pattern<br>[, start<br>[, occurrence<br>[, return_option<br>[, match_option<br>[, subexp_option]]]]]) | 在 x 中查找 pattern，并返回 pattern 所在的位置。可以指定以下的可选参数：<br>• start 开始查找的位置。默认值是 1，指 x 的第一个字符。<br>• occurrence 说明应该返回第几次出现 pattern 的位置。默认值是 1，这意味着函数返回 pattern 第一次在 x 中出现的位置。<br>• return_option 说明应该返回什么整数。若该参数为 0，则说明要返回的整数是 x 中的第一个字符的位置；若该参数为非 0 的整数，则说明要返回的整数为 x 中出现在 pattern 之后的字符的位置<br>• match_option 修改默认的匹配设置，其工作方式与 REGEXP_LIKEK()中指定的方式相同。<br>• subexp_option 是 Oracle Database 11g 新增加的，其工作方式如下：对于具有子表达式的模式， subexp_option 是 0~9 之间的一个非负数，指出 pattern 中的哪个子表达式是函数的目标。例如，考虑表达式 0123(((abc)(de)f)ghi)45(678)，此表达式有 5 个子表达式，分别是："abcdefghi"、"abcdef"、"abc"、"de"和"678"。<br>如果 subexp_option 是 0，则返回 pattern |

| | |
|---|---|
| | 的位置。如果 pattern 没有正确的子表达式数字，则函数返回 0。subexp_option 为空值则返回空。subexp_option 的默认值是 0 |

(续表)

| 函　　数 | 说　　明 |
|---|---|
| REGEXP_REPLACE(x, pattern<br>[, replace_string<br>[, start<br>[, occurrence<br>[, match_option]]]]) | 在 x 中查找 pattern，并将其替换为 replace_string。其他选项的意思与 REGEXP_INSTR()函数的参数完全相同 |
| REGEXP_SUBSTR( x, pattern<br>[, start<br>[, occurrence<br>[, match_option<br>[, subexp_option]]]]) | 返回 x 中可以匹配 pattern 的一个子字符串，其开始位置由 start 指定。其他选项的意思与 REGEXP_INSTR()函数的参数完全相同。Oracle Database11g 新增加的 subexp_option 其工作方式与 REGEXP_INSTR()函数中相同 |
| REGEXP_COUNT( x, pattern<br>[, start<br>[, match_option]]) | 这是 Oracle Database11g 新增加的一个函数。在 x 中查找 pattern，并返回 pattern 在 x 中出现的次数。可以提供以下两个可选参数：<br>• start 开始查找的位置。默认值是 1，指 x 的第一个字符。<br>• match_option 修改默认的匹配设置，其工作方式与 REGEXP_LIKEK()中相同 |

接下来的几节将会介绍更多有关正则表达式函数的知识。

## 1. REGEXP_LIKE()

REGEXP_LIKE(x, pattern [, match_option])用于在 x 中查找 pattern 参数中定义的正则表达式，该函数还可以提供一个可选参数 match_option，它可以设置为下面几个字符之一：

- 'c'，说明在匹配时区分大小写(默认选项)
- 'I'，说明在匹配时不区分大小写
- 'n'，允许使用可以匹配任意字符的操作符
- 'm'，将 x 作为一个包含多行的字符串

下面这个查询使用 REGEXP_LIKE 函数检索生日在 1965 年到 1968 年之间的顾客：

```
SELECT customer_id, first_name, last_name, dob
FROM customers
WHERE REGEXP_LIKE(TO_CHAR(dob, 'YYYY'), '^196[5-8]$');
```

```
CUSTOMER_ID FIRST_NAME LAST_NAME     DOB
----------- ---------- ---------- ---------
          1 John             Brown        01-JAN-65
          2 Cynthia          Green        05-FEB-68
```

下面这个查询检索名字以 J 或 j 开头的顾客。注意传递给 REGEXP_LIKE()的正则表达式是 ^j，匹配选项是 i，这说明不区分大小写，因此在本例中，^j 可以匹配 J 或 j：

```
SELECT customer_id, first_name, last_name, dob
FROM customers
WHERE REGEXP_LIKE(first_name, '^j', 'i');
```

```
CUSTOMER_ID FIRST_NAME LAST_NAME     DOB
----------- ---------- ---------- ---------
          1 John             Brown        01-JAN-65
```

## 2. REGEXP_INSTR()

REGEXP_INSTR(x, pattern [, start [, occurrence [, return_option [, match_option]]]])用于在 x 中查找 pattern；REGEXP_INSTR()返回 pattern 出现的位置。匹配位置从 1 开始。

下面这个查询使用 REGEXP_INSTR 函数返回匹配正则表达式 l[[:alpha:]]{4}的位置：

```
SELECT
REGEXP_INSTR('But, soft! What light through yonder window breaks?',
'l[[:alpha:]]{4}') AS result
FROM dual;
```

RESULT

---

17

注意返回值为 17，这是 light 中 l 的位置。

下面这个查询返回第二次匹配正则表达式 s[[:alpha:]]{3}的位置，匹配位置从 1 开始：

```
SELECT
REGEXP_INSTR('But, soft! What light through yonder window softly breaks?',
's[[:alpha:]]{3}', 1, 2) AS result
FROM dual;
```

RESULT

---

45

下面这个查询使用 REGEXP_INSTR 函数返回第二次匹配字母 o 的位置，匹配位置从 10 开始：

```
SELECT
REGEXP_INSTR('But, soft! What light through yonder window breaks?',
'o', 10, 2) AS result
FROM dual;
```

RESULT

---

32

## 3. REGEXP_REPLACE()

REGEXP_REPLACE(x, pattern [, replace_string [, start [, occurrence[, match_option]]]]) 用于在 x 中查找 pattern，并将其替换为 replace_string。

下面这个查询使用 REGEXP_REPLACE 函数将匹配正则表达式 l[[:alpha:]]{4}的子字符串替换为字符串 sound：

```
SELECT
REGEXP_REPLACE('But, soft! What light through yonder window breaks?',
'l[[:alpha:]]{4}', 'sound') AS result
FROM dual;
```

RESULT

---

But, soft! What sound through yonder window breaks?

注意 light 已经被替换为 sound。

## 4. REGEXP_SUBSTR()

REGEXP_SUBSTR(x, pattern[, start [, occurrence[, match_option]]])用于在 x 中查找匹配 pattern 的子字符串，开始位置由 start 指定。

下面这个查询使用 REGEXP_SUBSTR 函数返回匹配正则表达式 l[[:alpha:]]{4}的子字符串：

```
SELECT
REGEXP_SUBSTR('But, soft! What light through yonder window breaks?',
'l[[:alpha:]]{4}') AS result
FROM dual;
```

RESUL

---

light

## 5. REGEXP_COUNT()

REGEXP_COUNT()是 Oracle Database11g 新增加的一个函数。REGEXP_COUNT(x, pattern[, start [,match_option]])用于在 x 中查找 pattern，并返回 pattern 在 x 中出现的次数。可以提供可选参数 start，指出要从 x 中开始查找 pattern 的那个字符；也可以提供可选的 match_option 字符串，指出匹配选项。

下面这个查询使用 REGEXP_COUNT 函数返回正则表达式 s[[:alpha:]]{3}出现的次数：

```
SELECT
 REGEXP_COUNT('But, soft! What light through yonder window softly breaks?',
  's[[:alpha:]]{3}') AS result
FROM dual;
```

RESULT

---

2

注意返回结果是 2，这表明正则表达式在提供的字符串中有两次匹配。