06

Como cortar uma string

Transcrição

Conheceremos mais algumas funções para trabalharmos com texto. Vamos imaginar que no banco de dados, temos alguns registros salvos com pronomes de tratamento, como senhor ou senhor, doutor ou doutora. No entanto, eles aparecem abreviados nos registros, como "sr julio", e quero que eles apareçam sem abreviações. Iremos usar a função `replace()`. Usaremos como parâmetros: `sr julio`, `sr`, `senhor`.

```
SQL> select replace(`sr julio`, `sr`, `senhor`) from dual;

REPLACE('SR)
------------
senhor julio
```

Ele substituiu o `sr` por `senhor`.

Mas e o que acontece se um parâmetro é nulo? Vamos começar pelo último parâmetro. O resultado será que a *string* será removida.

```
SQL> select replace('sr julio', 'sr',null) from dual;

REPLAC
------
 julio
```

O comportamento padrão do terceiro parâmetro é `null`. Podemos inclusive omiti-lo, que o resultado será o mesmo.

E se o segundo parâmetro também for nulo? A *string* não será alterada.

```
SQL> select replace('sr julio',null) from dual;

REPLACE(
--------
sr julio
```

Como não procuramos nada para ser alterado, nada será afetado.

Mas, o que acontece quando o primeiro parâmetro é nulo?

```
SQL> select replace(null,'sr') from dual;

R
-
```

Ele irá retornar nulo, um comportamento esperado e que se enquandra na regra de outras funções.

A função replace é muito usada em situações quando queremos substituir `\` (barra invertida) por ´/´(barra), ou um quebra de linha `\n` por `br` .

Existe outra função bastante parecida: a `translate()` . Ela irá mapear um conjunto de mapear um conjunto de caracteres em outro. Vamos imaginar o a situação em que o nome `renan` está escrito com alguns numerais. `r3n4n` . Mas quero traduzir isto e transformar os números em vogais.

A função `translate()` aceita três parâmetros. Para fazer a troca, usaremos como segundo parâmetro os números `1340` , que são usados para substituir as letras `ieao` , respectivamente.

```
SQL> select translate('r3n4n','1340','ieao') from dual;

TRANS
-----
renan
```

Ele irá substituir os caracteres correspondentes na *string*. Se não tivessemos adicionado todos os caracteres correspondentes no terceiro parâmetro, os números simplesmente teriam sido omitidos no retorno.

```
SQL> select translate('r3n4n','1340','i') from dual;

TRA
---
rnn
```

Como o segundo caractere do segundo parâmetro não tinha correspondente no terceiro, ele foi removido.

Caso o terceiro parâmetro seja nulo, a *string* será apagada no retorno.

```
SQL> select translate('r3n4n', '1230',null) from dual;

T
-
```

O mesmo acontecerá se o segundo parâmetro for nulo.

```
SQL> select translate('r3n4n',null,'ieao') from dual;

T
-
```

Observe a diferença entre o `translate()` e `replace()` .

- A função `replace()` irá trocar uma sequência de caracteres por outra.

_ A função `translate()` irá mapear o primeiro caractere da primeira sequência com a da segunda. Depois, fará o mesmo com os demais caracteres de cada sequência.

A última função para trabalharmos com *string* que veremos é a `substr()`, usada nas situações quando queremos selecionar um substring - uma parte da *string*.

Vamos testá-la com o exemplo do "sr julio". Usaremos a função `substr()` e como parâmetro, queremos que ele considere só a partir do terceiro caractere.

```
SQL> select substr('sr julio','3') from dual;


SUBSTR
------
 julio
```

Por padrão, ele irá buscar a partir da posição indicada até o fim da *string*. Porém, o que aconteceria se quiséssemos apenas os três primeiro caracteres de `julio`? Após indicar a posição inicial da busca, determinaremos quantos caracteres queremos recortar.

```
SQL> select substr('sr julio',4,3) from dual;

SUB
---
jul
```

O segundo parâmetro é referente ao **inicio** da busca e o segundo, ao **termino**. A partir da letra `j`, ele contou `3` caracteres.

Vale lembrar que o último parâmetro precisa ter um valor positivo, caso contrário o retorno será nulo. No entanto, o segundo parâmetro admite valores negativos. Se quisermos começar a busca a partir da posição `-4`, ela será feita da direita para a esquerda.

```
SQL> select substr('sr julio',-4,3) from dual;

SUB
---
uli
```

Em quais situações é interessante usar o `substr()`? Anteriormente, vimos a função `instr()`. Utilizamos a função para descobrir qual era a posição de `.` no nome de um arquivo. Agora, podemos recortar o formato do arquivo.

```
SQL> select 'desktop/notas.mes.txt' from dual;

'DESKTOP/NOTAS.MES.TX
---------------------
desktop/notas.mes.txt
```

Nós precisaremos descobrir a posição do `.` no arquivo. Usaremos a função `instr()`.

```
SQL> select instr('desktop/notas.mes.txt','.',-1,1) from dual;

INSTR('DESKTOP/NOTAS.MES.TXT','.',-1,1)
```

```
---------------------------------------
                                     18
```

Quando usamos `substr()`, o segundo parâmetro informava a posição de onde queríamos começar. No caso do nosso arquivo, podemos começar do retorno do `instr()`. Ou seja, iremos usá-lo como segundo parâmetro da `substr()`. O primeiro parâmetro será o caminho do arquivo `desktop/notas.mes.txt`.

```
SQL> select substr('desktop/notas.mes.txt',instr('desktop/notas.mes.txt','.',-1,1)) from dual;

SUBS
----
.txt
```

Se adicionarmos `+1` ao retorno usado no segundo parâmetro, iremos remover o `.` do resultado.

```
SQL> select substr('desktop/notas.mes.txt',instr('desktop/notas.mes.txt','.',-1,1)+1) from dual;

SUBS
----
txt
```

A função retornou apenas a extensão do arquivo.

Nós podemos **combinar** as funções `instr()` e `substr()` para pesquisarmos a extensão de um arquivo. Lembrando que se estivéssemos selecionando de uma tabela, o texto do caminho do arquivo seria o nome da coluna em que ele estivesse inserido.

Vimos até aqui as principais funções que podem ser cobradas no exame de certificação do Oracle. Temos outras que não são muito utilizadas, que não vi serem cobradas em outros simulados ou quando eu fiz a prova.

Até a próxima aula.