

## Transformar para maiúscula e minúscula

### Transcrição

Vamos imaginar que quero gerar uma nota fiscal e preciso seguir uma regra de que o nome do cliente deve aparecer em caixa alta. No Oracle, temos uma função que transforma todas as letras maiúsculas: trata-se da `upper()`. Iremos testá-la com o nome `renan`:

```
SQL> select upper('renan') from dual;
```

```
UPPER  
-----  
RENAN
```

Ele retornou todas as letras em caixa alta.

Se passarmos um valor nulo, ele retornará mesmo tipo de valor.

```
SQL> select upper(null) from dual;
```

```
U  
-
```

Podemos usar também uma função que transformará todos os caracteres em caixa baixa. Com a função `lower()` todas os caracteres ficarão em minúsculo. Vamos testar a função com o endereço do cliente.

```
SQL> select lower('Rua Vergueiro 3185') from dual;
```

```
LOWER('RUAVERGUEIR  
-----  
rua vergueiro 3185
```

Temos outra função que define o tamanho ocupado pelo texto. Vamos imaginar que queremos todos os nomes ocupando um espaço específico na folha.

Vamos primeiramente, listar todos os nomes de funcionários.

```
SQL> select nome from funcionarios;
```

```
NOME  
-----  
Kauan  
Jose  
Eduarda  
Leonardo  
Thais  
Nicole  
Leticia
```

```

Joao
Igor
Diogo
Karlos

```

```

NOME
-----

```

```

Stevan
Otavio

```

13 linhas selecionadas.

Para garantir que todos eles tenham 20 caracteres, iremos usar a função `lpad()`. Iremos informar que o parâmetro `nome` tenha 20 caracteres. Caso o nome seja menor, que ele preencha o restante do espaço com asterisco ( `*` ).

```
SQL> select lpad(nome,20,'*') from funcionarios;
```

```
LPAD(NOME,20,'*')
```

```

-----
*****Kauan
*****Jose
*****Eduarda
*****Leonardo
*****Thais
*****Nicole
*****Leticia
*****Joao
*****Igor
*****Diogo
*****Karlos

```

```
LPAD(NOME,20,'*')
```

```

-----
*****Stevan
*****Otavio

```

13 linhas selecionadas.

A lista incluiu diversos `*` à esquerda dos nomes - lembrando que o significado de `LPAD` é *left-pad*. Se não adicionarmos um valor para o preenchimento, ele deixará o espaço vazio.

```
SQL> select lpad(nome,20) from funcionarios;
```

```
LPAD(NOME,20)
```

```

-----
          Kauan
          Jose
        Eduarda
        Leonardo
          Thais
          Nicole
        Leticia
          Joao

```

Igor  
Diogo  
Karlos

```
LPAD(NOME,20)
-----
      Stevan
      Otavio
```

13 linhas selecionadas.

Neste caso, os nomes ocupam menos do que 20 caracteres. Temos a opção também de usar menos caracteres. Quando usamos o `lpad()`, a função irá garantir que o campo tenha o tamanho especificado. Se o nome ultrapassar o tamanho, ele cortará a palavra.

```
SQL> select lpad(nome,3) from funcionarios;
```

```
LPA
---
Kau
Jos
Edu
Leo
Tha
Nic
Let
Joa
Igo
Dio
Kar
```

```
LPA
---
Ste
Ota
```

13 linhas selecionadas.

A função irá tanto adicionar como diminuir caracteres.

Outro padrão é que se adicionarmos um parâmetro nulo, ele irá retornar todos os campos nulos também.

Agora, iremos conhecer a função `rpad()` - que significa *right pad*. A função terá um comportamento semelhante à `lpad`, mas, caso seja necessário adicionar caracteres, eles serão preenchidos à direita.

```
SQL> select rpad(nome,20,'*') from funcionarios;
```

```
RPAD(NOME,20,'*')
-----
Kauan*****
Jose*****
Eduarda*****
Leonardo*****
```

```
Thais*****
Nicole*****
Leticia*****
Joao*****
Igor*****
Diogo*****
Karlos*****
```

```
RPAD(NOME,20,'*')
```

```
-----
```

```
Stevan*****
Otavio*****
```

13 linhas selecionadas.

Da mesma forma, se definirmos um número específico de caracteres, ele irá usar o espaço determinado.

```
SQL> select rpad(nome,5) from funcionarios;
```

```
RPAD(
```

```
-----
```

```
Kauan
```

```
Jose
```

```
Eduar
```

```
Leona
```

```
Thais
```

```
Nicol
```

```
Letic
```

```
Joao
```

```
Igor
```

```
Diogo
```

```
Karlo
```

```
RPAD(
```

```
-----
```

```
Steva
```

```
Otavi
```

13 linhas selecionadas.

Se podemos adicionar caracteres, temos também a opção de excluí-los. Por exemplo, ao adicionar o nome "felipe", inclui diversos espaços antes do nome.

```
SQL> select `felipe` from dual;
```

```
'FELIPE'
```

```
-----
```

```
felipe
```

Agora, queremos limpar o espaço vazio. Para isto, temos a função `ltrim()`. Com ela podemos remover um caractere específico do texto.

```
SQL> select ltrim('      felipe', ' ') from dual;

LTRIM(
-----
felipe
```

Agora, ele retornou apenas a palavra `felipe`. Se removermos o segundo parâmetro, a função por padrão remove espaçamento.

```
SQL> select ltrim('      felipe') from dual;

LTRIM(
-----
felipe
```

E se além do espaço em branco, quisermos remover um caractere? Por exemplo, a letra `f`.

```
SQL> select ltrim('      felipe', ' f') from dual;

LTRIM(
-----
elipe
```

A função removeu os dois. Observe que no segundo parâmetro passamos um espaço em branco e a letra "f". Com o `ltrim()`, todos os espaços foram removidos. A letra "f" também foi, porque ele era o próximo caractere depois do espaçamento. Mas se tivéssemos usado como parâmetro `e`, o resultado não seria o mesmo:

```
SQL> select ltrim('      felipe', ' e') from dual;

LTRIM(
-----
felipe
```

Como a letra `e` não era o próximo caractere, não foi removida.

Com uma comportamento semelhante ao `ltrim()`, temos a função `rtrim()`, que terá a mesma utilidade, porém, espelhada. Ou seja, ela removerá caracteres a partir da direita. Faremos o teste com a palavra `felipe` com um espaçamento à direita.

```
SQL> select rtrim('felipe      ') from dual;

RTRIM(
-----
felipe
```

Também existem temos uma função que combina o comportamento da `ltrim()` e a `rtrim()`: trata-se da `trim()`. Por padrão, ela igualmente irá remover o espaço em branco. Mas ela não receberá um segundo parâmetro. Para definir qual caractere queremos remover, precisaremos usar um expressão.

```
SQL> select trim('*' from '*****Vinicius*****') from dual;
```

```
TRIM('*'
```

```
-----
```

```
Vinicius
```

**Fique atento**, porque quando eu fiz a prova de certificação, caiu uma questão referente ao tema. É fácil deduzirmos que a função `trim()` funcionaria como `rtrim()` e `ltrim()`. Porém, não é assim. Com a `trim()`, não podemos passar um segundo parâmetro definindo qual caractere queremos remover. Precisamos informar o caractere, usar a cláusula `from` e depois especificar a *string*.

Nós conhecemos mais algumas funções para trabalharmos com texto, todas elas utilizam a regra de que se um dos parâmetros é nulo, o retorno também será.

Em seguida, veremos outras com novas utilidades.

