01

# Conhecendo as funções existentes no Oracle

### Transcrição

Nós já vimos as principais funções para trabalhar com textos, porém, tão importante é saber como trabalhar com números nos nossos bancos de dados. Iremos começar a conhecer as funções usadas com números. Para isto, precisaremos conhecer o sinal dos mesmos: temos positivos e negativos. No Oracle, temos algumas funções para trabalhar com isto.

A primeira é `sign()`. O que ela faz? Nós podemos informar qualquer valor, caso o número for positivo, a função irá retornar `1`; se ele for igual a zero, ela irá retornar `0`. E se o número for negativo, ela irá retornar `-1`. Vamos fazer um teste com o valor `50` e pesquisar na tabela `dual`.

```
SQL> select sign(50) from dual;

SIGN(50)
---------
        1
```

A função também retornaria o valor `1`, se usassemos o número `135`.

```
SQL> select sign(135) from dual;

 SIGN(135)
----------
         1
```

Mas se passarmos um valor `-80`, ele irá retornar o valor `-1`.

```
SQL> select sign(-80) from dual;

 SIGN(-80)
----------
        -1
```

O resultado seria o mesmo para qualquer número negativo.

E o que acontece quando usamos `0`?

```
SQL> select sign(0) from dual;

 SIGN(0)
----------
         0
```

Nós iremos usar a função `sign()` sempre que quisermos saber se um número é positivo, negativo ou igual a zero.

Se passarmos um valor `null` para função, ela também terá o mesmo comportamento que vimos anteriormente.

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

SIGN(NULL)
----------
```

Além de queremos identificar o sinal de um número, podemos querer ignorá-lo. Agora. usaremos a função `abs`, que irá selecionar o valor absoluto.

```
SQL> select abs(-35) from dual;

 ABS(-35)
---------
       35
```

Ela retornou o número `35`, que é o valor absoluto. Se passássemos o valor `35`, o resultado seria o mesmo.

```
SQL> select abs(35) from dual;

 ABS(35)
---------
       35
```

Independente se passamos um número positivo ou negativo, ele irá retornar o número positivo. Se o valor for igual a `0`, o resultado também será `0`. E se passarmos o `null`, o retorno será igual.

```
SQL> select abs(0) from dual;

 ABS(0)
---------
        0
```

Além de trabalharmos com sinais, é comum trabalharmos com números decimais. Por exemplo, vamos selecionar o valor `π` (Pi), que é igual a `22/7`.

```
SQL> select 22/7 from dual;

      22/7
----------
3,14285714
```

Às vezes, a parte decimal de um número é irrelevante e queremos selecionar apenas a parte inteira. Podemos usar uma função que arredonte o número. Com `round()` podemos fazer isto.

```
SQL> select round(22/7) from dual;

      22/7
----------
         3
```

Ele retornou um número o número `3`, porque é o valor inteiro mais próximo de `3,14`. Mas o resultado será diferente, se passarmos o valor `3,55`.

```
SQL> select round(3.55) from dual;

ROUND(3.55)
-----------
          4
```

O critério é que, na casa decimal, de 0 até 4, o arredondamento será para baixo. Porém, o arredondamento da casa decimal, de 5 em diante, será para mais.

No entanto, em alguns casos, não queremos arredondar para um número inteiro. Na escola, por exemplo, era comum considerarmos o valor de `π` igual a `3,14`. E se quisermos que o número decimal seja mantido com duas casas. Por padrão, a função `round()` tem um segundo parâmetro que é igual a zero. É este valor que define a quantidade de casas decimais. Mas se queremos que o nosso número tenha duas casas decimais, podemos especificá-lo na consulta:

```
SQL> select round(22/7,2) from dual;

ROUND(22/7,2)
-------------
         3,14
```

A regra do arredondamento continuará valendo.

```
SQL> select round(3.145,2) from dual;

ROUND(3.145,2)
-------------
         3,15
```

Agora, o arredondamento ocorrerá apenas na segunda casa decimal.

Um uso menos comum da função `round()` é quando temos um número grande. Por exemplo, `1235.55`. Se pedirmos para arredontar com as duas casas decimais, o retorno será o próprio número.

```
SQL> select round(125.55,2) from dual;

ROUND(1235.55,2)
----------------
         1235,55
```

Podemos também passar um valor negativo para o segundo parâmetro. O que acontece se usarmos o valor o `-2`, por exemplo?

```
SQL> select round (1235.55,-2) from dual;

ROUND(1235.55,-2)
```

```
-----------------
             1200
```

Quando peço para ele arredondar um valor negativo, a função irá trabalhar com os números posicionados antes da vírgula. No exemplo, o valor mais próximo de `35` é zero, ele irá arredondar para 1200. Se tivéssemos trabalhado com o número `1255`, o arredondamento teria virado mais uma centena.

```
SQL> select round(1255.55,-2) from dual;

ROUND(1255.55,-2)
-----------------
             1300
```

Então, quando determinamos um parâmetro negativo, nos referimos a quantas casa queremos andar antes da vírgula.

Temos ainda uma função bastante parecida com a `round()`, mas que possui uma pequena diferença, é a `trunc()`. Testaremos o comportamento da função com o valor `3,1415`.

```
SQL> select trunc(3.1415) from dual;

TRUNC(3.1415)
-------------
            3
```

O retorno seria apenas o valor inteiro `3`. Porém, se trabalharmos com o número `3,9415`, que está mais próximo do número `4`, a função igualmente irá arredondar para `3`.

```
SQL> select trunc(3.9415) from dual;

TRUNC(3.9415)
-------------
            3
```

Ele apenas ignora as casas decimais e não se preocupa em arredondá-las. Mas nós podemos definir quantas casas queremos que seja ignorada especificando um segundo parâmetro.

```
SQL> select trunc(3.9415,2) from dual;

TRUNC(3.9415,2)
---------------
           3,94
```

Quando usamos o valor `2` no segundo parâmetro, ele ignorou duas casas decimais.

Também precisamos saber o que acontece quando passamos valores nulos para os parâmetros das duas funções anteriores. Na função `trunc()`, quando um dos parâmetros for um valor nulo, o resultado também será.

```
SQL> select trunc(3.9415,null) from dual;

TRUNC(3.9415,NULL)
------------------
```

O mesmo acontecerá se os dois parâmetros forem nulos.

```
SQL> select trunc(null,2) from dual;

TRUNC(NULL,2)
-------------
```

Vale lembrar que os dois parâmetros também precisam ser números. Se adicionarmos como parâmetro a letra `a`, ele dirá que é um número inválido.

```
SQL> select trunc(3.9415,'a') from dual;
select trunc(3.9415,'a') from dual

ERRO na linha 1:
ORA-01722:numero invalido
```

O mesmo vale para função `round()`, os dois parâmetros precisam ser números. Caso algum dos parâmetros seja nulo, o resultado também será.

Se informarmos um número decimal, no segundo parâmetro, a função irá considerar apenas o número inteiro.

Até aqui, vimos as principais funções para trabalharmos sinais dos números, além de arredondamentos e truncamentos. Em seguida, veremos outras funções.