03

Filtrando dados usando o having

Filtrando dados usando o having

Continuando com a ideia de relatórios, vamos visualizar a soma do salário de cada setor. Usaremos a função `sum()` para a soma e o `group by` para definir que será da coluna `setor_id`.

```
SQL> select sum(salario),setor_id from funcionarios group by setor_id;

SUM(SALARIO)  SETOR_ID

------------ ---------
       17500         1
       13500         2
       10150         3
```

Agora, quero filtrar estas informações e quero identificar quais setores tem um gasto superior a R$15.000. Nós já vimos nos cursos anteriores o uso que podemos fazer do `where()`, vamos tentar aplicá-la na nossa *query*. A nossa regra será que a soma do salário precisa ser maior (>) do que `15000`.

```
SQL> select sum(salario), setor_id from funcionarios where sum(salario) > 15000 group by setor_id;
select sum(salario), setor_id from funcionarios where sum(salario) > 15000 group by setor_id

ERRO na linha 1:
ORA-00934: a funcao de grupo nao e permitida aqui
```

Ele retornou que a função de grupo não é permitida no `where()`. Por que isto acontece? A função `where()` é realmente usada para filtrar dados, porém, ela é executada nos dados por linha. No entanto, agora, queremos aplicar um filtro que seja aplicado nos dados agrupados. Então, **nunca** poderemos ter uma função de agrupamento dentro do `where()`. Para conseguirmos realizar a filtragem, teremos que usar o `having()`, que fará exatamente o mesmo que o `where()`, mas em dados agrupados.

```
SQL> select sum(salario), setor_id from funcionarios having sum(salario) > 15000 group by setor_id;

SUM(SALARIO)   SETOR_ID
------------ ----------
       17500          1
```

Ele retornou que o setor `1` tem um gasto superior a R$15.000.

E podemos usar `where()` juntamente com `having`? Sim, em um situação em que queremos desconsiderar os salários menores do que R$2000, iremos adicionar o `where()` na *query*.

```
SQL> sleect sum(salario), setor_id from funcionarios where salario > 2000 having sum(salario) > 1500

SUM(SALARIO)   SETOR_ID
```

```
----------- ----------
      16000          1
```

O gasto do setor `1` ficou menor agora, porque excluímos o registro de R$1500.

Vamos detalhar um pouco mais a razão pela qual não podemos usar `where()` na filtragem. Nós estamos usando o `SQL \*Plus` que é o padrão que vem com o Oracle. Com ele, temos a opção de setar o `autrace`, que irá explicar como a *query* está sendo executada. Queremos ligar o `autotrace`, então passaremos `on`, e queremos passar o identificador de sessão `ses`.

```
SQL> set autotrace on ses
SP2-0610: Nao e possivel localizar o Identificador de Sessao. verifique se PLUSTRACE esta ativado
SP2-0611: Erro de ativacao do relatorio STATISTICS
```

Ele disse que o Plustrace está ativado. Em seguida, iremos executar a *query* novamente.

```
SQL> select sum(salario),setor_id from funcionarios where salario > 2000 having sum(salario) > 15000
```

Agora, o Oracle irá explicar a *query*. Ele irá informar que a primeira ação realizada foi acessar a tabela (`TABLE ACCESS FILL`), depois acessou o `GROUP BY`. Em seguida, ele aplicou o filtro (`FILTER`) e por último, ele retornou os dados do `select`.

```
Plano de Execucao
----------------------------------------------------
Plan hash value: 2807415947

-------------------------------------------------------------------------------------------------------
| Id   | Operation          | Name           | Rows   | Bytes  | Cost (%CPU)| Time
   |
-------------------------------------------------------------------------------------------------------

|   0  | SELECT STATEMENT   |                |    11  |   286  |    4   (25)| 00:00:
01 |

|*  1  |  FILTER            |                |        |        |            |
   |

|   2  |   HASH GROUP BY    |                |    11  |   286  |    4   (25)| 00:00:
01 |

|*  3  |   TABLE ACCESS FULL|  FUNCIONARIOS  |    11  |   286  |    4    (0)| 00:00:
01 |

-------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("SALARIO")>15000)
   3 - filter("SALARIO">2000)

   \...
```

Observe que tanto a linha `3` como a linha `1` estão sinalizadas com um `*` (asterísco). O Oracle irá informar que são informações extras.

E na linha 3, ele explica que ao acessar a tabela, ele aplica o filtro do `where() : "SALARIO">2000` . Ou seja, a função é executada antes de agruparmos os dados. Por isso, não conseguimos usar uma função de agrupamento com ele, porque ser executado antes. A ordem é:

1) Execução da tabela 2) Execução do `where() ( "SALARIO">2000) 3) Agrupamento dos dados 4) Aplicação do filtro ( SUM("SALARIO")>15000`) 5) Retorno do resultado

Logo, vemos que o `where()` é excutado antes do `group by()` .

Esta é a maneira como trabalhamos com filtros para dados, usando funções de agrupamento.

Para desligar o `autotrace` , iremos setar novamente:

```
SQL> set autotrace off
```

Agora, se executarmos uma nova *query*, ele irá apresentar apenas os resultados.

Nós aprendemos a filtrar os dados usando `having()` e a sua diferença com a função `where()` - um ponto interessante que poderá ser cobrado no exame de certificação. Você precisará ficar atento: se o objetivo é filtrar o dado puro, usará `WHERE` , e se quiser filtrar dados agrupado, usará o `HAVING` .

Relatórios

Nós já vimos como criar um relatório do subtotal de gastos de cada setor.

```
SQL> select sum(salario),setor_id from funcionarios group by setor_id;

SUM(SALARIO)   SETOR_ID
------------ ----------
       17500          1
       13500          2
       10150          3
```

Vamos imaginar que além do subtotal, eu quero trazer a soma dos valores dos três setores. Nós poderiamos apenas retirar o `group by()` da nossa *query* e ele já retornaria o total.

```
SQL> select sum(salario) from funcionarios;

SUM(SALARIO)
------------
       41150
```

Precisaríamos executar duas *queries* para listar todos os valores. No Oracle, temos uma forma de resolver o problema com uma única *query*: com a função `rollup()` , ele irá calcular o subtotal de cada setor.

```
SQL> select sum(salario),setor_id from funcionarios group by rollup(setor_id);

SUM(SALARIO)   SETOR_ID
------------ ----------
       17500          1
       13500          2
       10150          3
       41150
```

Ele apresentou todos os resultados que foram retornados nas *queries* anteriores, em uma única.

Vamos agora criar uma *query* maior, na qual iremos pegar uma quantidade de funcionários, então incluiremos `count(*)` .

```
SQL> select sum(salario),count(*),setor_id from funcionarios group by rollup(setor_id);

SUM(SALARIO)   COUNT(*)   SETOR_ID
------------ ---------- ----------
       17500          5          1
       13500          5          2
       10150          4          3
       41150         14
```

Ele trouxe o `count()` do subtotal de funcionários.

O `rollup()` calcula o subtotal de todas as funções de agrupamento. Agora, além de agruparmos pelo subgrupo, queremos agrupar pelo `salario` .

```
SQL> select sum(salario),count(*),setor_id from funcionarios group by rollup(setor_id,salario);

SUM(SALARIO)   COUNT(*)    SETOR_ID
------------ ---------- -----------
        1500          1           1
        2600          1           1
        3000          1           1
        4200          1           1
        6200          1           1
       17500          5           1
        1800          1           2
        2000          1           2
        4400          2           2
        5300          1           2
       13500          5           2

SUM(SALARIO)   COUNT(*)    SETOR_ID
------------ ---------- -----------
        2100          1           3
        2200          1           3
        2400          1           3
        3450          1           3
       10150          4           3
       41150         14

14 linhas selecionadas.
```

Ele calcula o subtotal de cada setor, com os valores dos salários (`17500` , `13500` , `10150` e `41150`) e de funcionários (`5` , `5` , `4` e `14`).

Além do `rollup()` , temos a sub cláusula `cube()` , que irá trazer todas a combinações possíveis no agrupamento. Ela possibilita o cálculo de subtotais e totais para diferentes níveis dos dados. Vamos testá-la com o `setor_id` :

```
SQL> select sum(salario), setor_id from funcionarios group by cube(setor_id);

SUM(SALARIO)   SETOR_ID
------------ ----------
       41150
       17500          1
       13500          2
       10150          3
```

Ele retornou o subtotal dos três setores e depois a soma dos resultados. O `cube()` fica interessante se agruparmos mais de uma coluna no `group by()` . Adicionaremos também `count(*)` .

```
SQL> select sum(salario), setor_id from funcionarios group by cube(setor_id,salario);

SUM(SALARIO)   COUNT(*)    SETOR_ID
------------ ---------- ----------
       41150         14
        1500          1
        1800          1
        2000          1
        2100          1
        6600          3
        2400          1
        2600          1
        3000          1
        4200          1
        5300          1

SUM(SALARIO)   COUNT(*)   SETOR_ID
------------ ---------- ----------
        6200          1
        3450          1
       17500          5          1
        1500          1          1
        2600          1          1
        3000          1          1
        4200          1          1
        6200          1          1
       13500          5          2
        1800          1          2
        2000          1          2

SUM(SALARIO)   COUNT(*)   SETOR_ID
------------ ---------- ----------
        4400          2          2
        5300          1          2
       10150          4          3
```

```
2100          1          3
2200          1          3
2400          1          3
3450          1          4

29 linhas selecionadas.
```

Ele retornou primeiro o total de todos os salários. Depois o subtotal, por cada valor de salário. Mas observe, que ele somou os valores repetidos. Por exemplo, o valor `6600` é a soma de três registros de `2200`. Ele também calculou o subtotal para cada setor.

Agora, conhecemos estes os modificadores `rollup()` e o `cube()`.

Este foi o nosso quarto curso de Oracle, falando sobre funções de agrupamento e o `GROUP BY`.

Até o próximo curso!