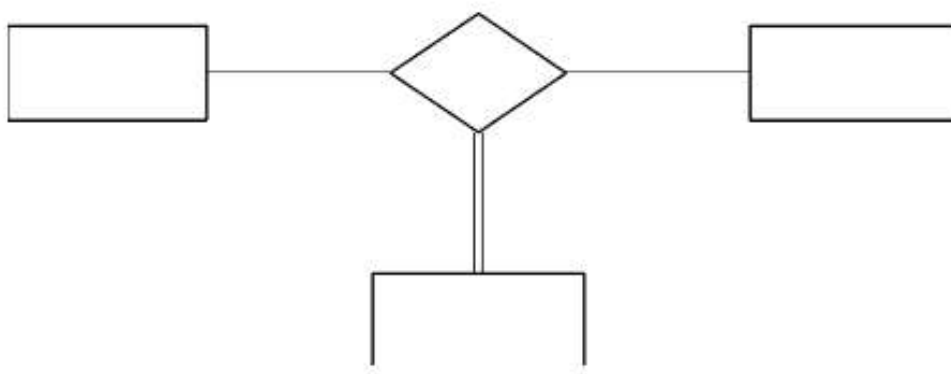


Consultas com Agrupamento, Totalização, Ordenação e Operadores IN, EXISTS, ANY e ALL

INTRODUÇÃO

Os exemplos a seguir utilizam as tabelas REPRESENTANTE, CLIENTE e PEDIDO que se relacionam conforme o diagrama abaixo. Os scripts de criação e inserção de dados encontram-se no material de apoio.



Resumidamente, representantes de vendas emitem pedidos. Um pedido corresponde à venda feita por um representante a um cliente em uma data. O mapeamento do relacionamento CRP foi feito através de chaves estrangeiras na tabela PEDIDO.

A linguagem SQL possui um conjunto de funções, chamadas de funções de coluna, que permitem que o resultado de uma consulta seja agrupado e sumariado. Basicamente, estas funções são aplicadas a uma ou mais colunas presentes no comando SELECT. As funções de sumarização encontram-se a seguir.

| Função | Descrição |
|----------|---|
| COUNT() | Conta o número de linhas da consulta |
| SUM() | Soma os valores da coluna |
| AVG() | Calcula a média dos valores da coluna. É equivalente a SUM()/COUNT(*) |
| MIN() | Valor mínimo da coluna |
| MAX() | Valor máximo da coluna |
| STDDEV() | Desvio padrão da coluna (populacional) |

Veja o resultado da consulta a seguir. A intenção é mostrar o representante que realizou a maior venda e o respectivo valor:

```
SELECT MAX(P.valor) AS 'MAIOR VENDA', R.nome AS NOME FROM pedido AS P, representante AS R WHERE P.rep_id = R.rep_id;
```

| MAIOR VENDA | NOME |
|-------------|---------------|
| 16000.00 | João da Silva |

Este não é o resultado esperado. Há dois pedidos de 16.000,00. E João da Silva não foi responsável por nenhum dos dois. O que aconteceu?

Consultas com funções de coluna retornam uma única linha. Pode-se usar diversas funções de coluna em uma única consulta, aplicadas a diferentes colunas, porém o valor de colunas simples, como R.nome na consulta anterior, retornam valores imprevisíveis. Deve-se, portanto, evitar misturar colunas simples e funções de coluna na mesma consulta.

Consulta a seguir retorna o valor médio dos pedidos:



```
SELECT AVG(P.valor) AS 'MÉDIA' FROM pedido AS P;
```

| MÉDIA |
|-------------|
| 8080.000000 |

A CLÁUSULA ORDER BY

A cláusula ORDER BY permite que a saída de uma consulta seja ordenada de acordo com as colunas retornadas. A sua forma geral é:

```
SELECT . . . [ORDER BY {coluna1 | expressão1} [ASC | DESC] [, {coluna2 | expressão2} [ASC | DESC],] ...]
```

Os modificadores ASC ou DEC especificam se a ordenação é por ordem crescente (valor padrão) ou decrescente, respectivamente. A consulta a seguir ordena o resultado da consulta por representante, cliente e data do pedido:

A inclusão da cláusula GROUP BY, apresentada mais adiante, permite o retorno de várias linhas.

```
SELECT R.nome AS NOME, C.nome AS cliente, P.data AS DATA, FORMAT(P.valor, 2, 'de_DE') AS VALOR FROM pedido AS P, representante AS R, cliente AS C WHERE P.rep_id = R.rep_id AND P.cliente_id = C.cliente_id ORDER BY R.nome, C.nome, P.data;
```

Na consulta, utilizou-se a função `FORMAT()` para exibir valor com o formato brasileiro. A função recebe 3 argumentos: o valor a ser formatado, o número de casas decimais e o formato a ser utilizado. No MySQL 8.0, o formato 'pt_BR' não inclui os separadores de milhar, por isso foi utilizado outro padrão.

A CLÁUSULA GROUP BY

Até o momento, as consultas utilizando funções de coluna retornavam uma única linha de resultado com o resultado da função. A cláusula `GROUP BY` permite sumarizar grupos de linhas de uma consulta em “subtotais”. Sua forma geral é:

```
SELECT . . . [GROUP BY {coluna1 | expressão1} [, {coluna2 | expressão2}] ...  
[WITH ROLLUP]]
```

Os exemplos a seguir ilustram o funcionamento da cláusula `GROUP BY` e do modificador `WITH ROLLUP`.

Consulta para calcular o valor total de pedidos por representante, ordenado por nome:

```
SELECT R.nome AS NOME, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL FROM pedido AS P,
representante AS R WHERE P.rep_id = R.rep_id GROUP BY R.nome ORDER BY R.nome;
```



| NOME | TOTAL |
|---------------|-----------|
| Ana Matos | 39.000,00 |
| Joana Santos | 20.000,00 |
| João da Silva | 24.000,00 |
| Jorge Jesus | 23.000,00 |
| Marcos Leite | 39.000,00 |
| Maria Carmo | 57.000,00 |

Neste último exemplo, foi possível mesclar colunas simples com funções de coluna graças à cláusula GROUP BY. Ela cria uma hierarquia de colunas que serão utilizadas para a subtotalização. No exemplo acima, as linhas selecionadas pela cláusula WHERE são ordenadas por R.nome. P.valor é totalizado para cada valor distinto de R.nome. O resultado final corresponde a uma linha para cada valor de R.nome com a soma de R.valor. Importante: a cláusula ORDER BY deve sempre vir depois da GROUP BY.

Consulta para calcular o valor total de pedidos por representante e cliente, ordenado por nome do representante e nome do cliente (a tabela da direita é a continuação do resultado da consulta):

```
SELECT R.nome AS NOME, C.nome AS CLIENTE, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL
FROM pedido AS P, representante AS R, cliente AS C WHERE P.rep_id = R.rep_id AND
P.cliente_id = C.cliente_ID GROUP BY R.nome, C.nome ORDER BY R.nome, C.nome;
```

| NOME | CLIENTE | TOTAL |
|---------------|---------------------|-----------|
| Ana Matos | Máximo Transportes | 13.000,00 |
| Ana Matos | ServCentro Serviços | 26.000,00 |
| Joana Santos | Indústrias União | 14.000,00 |
| Joana Santos | ServCentro Serviços | 6.000,00 |
| João da Silva | Máximo Transportes | 21.000,00 |
| João da Silva | MaxLog Logística | 3.000,00 |
| Jorge Jesus | Máximo Transportes | 9.000,00 |
| Jorge Jesus | MaxLog Logística | 2.000,00 |

| NOME | CLIENTE | TOTAL |
|--------------|---------------------|-----------|
| Jorge Jesus | ServCentro Serviços | 12.000,00 |
| Marcos Leite | Indústrias União | 25.000,00 |
| Marcos Leite | Máximo Transportes | 1.000,00 |
| Marcos Leite | ServCentro Serviços | 13.000,00 |
| Maria Carmo | Indústrias União | 3.000,00 |
| Maria Carmo | Máximo Transportes | 13.000,00 |
| Maria Carmo | MaxLog Logística | 19.000,00 |
| Maria Carmo | ServCentro Serviços | 22.000,00 |

Utilizando a consulta anterior, deseja-se incluir o valor total de pedidos por representante, ou seja, a cada mudança de representante, deseja-se incluir o total daquele representante. O modificador WITH ROLLUP é

utilizado para isto: quando uma coluna da cláusula GROUP BY tem o seu valor alterado, é incluída uma linha com a totalização daquela coluna. As colunas que não tiveram seu valor alterado são preenchidas com NULL.

```
SELECT R.nome AS NOME, C.nome AS CLIENTE, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL
FROM pedido AS P, representante AS R, cliente AS C WHERE P.rep_id = R.rep_id AND
P.cliente_id = C.cliente_ID GROUP BY R.nome, C.nome WITH ROLLUP ORDER BY R.nome,
C.nome;
```

| NOME | CLIENTE | TOTAL |
|---------------|---------------------|------------|
| NULL | NULL | 202.000,00 |
| Ana Matos | NULL | 39.000,00 |
| Ana Matos | Máximo Transportes | 13.000,00 |
| Ana Matos | ServCentro Serviços | 26.000,00 |
| Joana Santos | NULL | 20.000,00 |
| Joana Santos | Indústrias União | 14.000,00 |
| Joana Santos | ServCentro Serviços | 6.000,00 |
| João da Silva | NULL | 24.000,00 |
| João da Silva | Máximo Transportes | 21.000,00 |
| João da Silva | MaxLog Logística | 3.000,00 |
| Jorge Jesus | NULL | 23.000,00 |
| Jorge Jesus | Máximo Transportes | 9.000,00 |

| NOME | CLIENTE | TOTAL |
|--------------|---------------------|-----------|
| Jorge Jesus | MaxLog Logística | 2.000,00 |
| Jorge Jesus | ServCentro Serviços | 12.000,00 |
| Marcos Leite | NULL | 39.000,00 |
| Marcos Leite | Indústrias União | 25.000,00 |
| Marcos Leite | Máximo Transportes | 1.000,00 |
| Marcos Leite | ServCentro Serviços | 13.000,00 |
| Maria Carmo | NULL | 57.000,00 |
| Maria Carmo | Indústrias União | 3.000,00 |
| Maria Carmo | Máximo Transportes | 13.000,00 |
| Maria Carmo | MaxLog Logística | 19.000,00 |
| Maria Carmo | ServCentro Serviços | 22.000,00 |

Não era exatamente isto que se desejava. As linhas de totalização aparecem antes das linhas de subtotalização. Isto ocorreu por causa dos valores NULL durante a ordenação do resultado. Se tirarmos a cláusula ORDER BY o resultado aparece na ordenação correta (verifique!). Na realidade, o resultado de uma consulta com a cláusula GROUP BY sempre estará ordenado pelas colunas desta cláusula, portanto não é necessária a utilização da cláusula ORDER BY.

Observe que os valores NULL que aparecem em uma consulta podem ser originados da cláusula GROUP BY . . . WITH ROLLUP ou por causa de

colunas com o valor NULL nas linhas selecionadas. A função de coluna GROUPING() permite saber qual é a origem do NULL:



```
SELECT R.nome AS NOME, C.nome AS CLIENTE, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL,
GROUPING(R.nome) AS R, GROUPING(C.nome) AS C FROM pedido AS P, representante AS R,
cliente AS C WHERE P.rep_id = R.rep_id AND P.cliente_id = C.cliente_ID GROUP BY
R.nome, C.nome WITH ROLLUP;
```

| NOME | CLIENTE | TOTAL | R | C |
|---------------|---------------------|-----------|---|---|
| Ana Matos | Máximo Transportes | 13.000,00 | 0 | 0 |
| Ana Matos | ServCentro Serviços | 26.000,00 | 0 | 0 |
| Ana Matos | NULL | 39.000,00 | 0 | 1 |
| Joana Santos | Indústrias União | 14.000,00 | 0 | 0 |
| Joana Santos | ServCentro Serviços | 6.000,00 | 0 | 0 |
| Joana Santos | NULL | 20.000,00 | 0 | 1 |
| João da Silva | Máximo Transportes | 21.000,00 | 0 | 0 |
| João da Silva | MaxLog Logística | 3.000,00 | 0 | 0 |
| João da Silva | NULL | 24.000,00 | 0 | 1 |
| Jorge Jesus | Máximo Transportes | 9.000,00 | 0 | 0 |
| Jorge Jesus | MaxLog Logística | 2.000,00 | 0 | 0 |
| Jorge Jesus | ServCentro Serviços | 12.000,00 | 0 | 0 |

| NOME | CLIENTE | TOTAL | R | C |
|--------------|---------------------|------------|---|---|
| Jorge Jesus | NULL | 23.000,00 | 0 | 1 |
| Marcos Leite | Indústrias União | 25.000,00 | 0 | 0 |
| Marcos Leite | Máximo Transportes | 1.000,00 | 0 | 0 |
| Marcos Leite | ServCentro Serviços | 13.000,00 | 0 | 0 |
| Marcos Leite | NULL | 39.000,00 | 0 | 1 |
| Maria Carmo | Indústrias União | 3.000,00 | 0 | 0 |
| Maria Carmo | Máximo Transportes | 13.000,00 | 0 | 0 |
| Maria Carmo | MaxLog Logística | 19.000,00 | 0 | 0 |
| Maria Carmo | ServCentro Serviços | 22.000,00 | 0 | 0 |
| Maria Carmo | NULL | 57.000,00 | 0 | 1 |
| NULL | NULL | 202.000,00 | 1 | 1 |

A função GROUPING() retorna 1 sempre que a coluna correspondente recebeu o valor NULL por conta da cláusula GROUP BY . . . WITH ROLLUP. O resultado pode ainda ficar melhor com o uso da função IF(). A sua forma geral é IF(condição, valor_se_verdadeiro, valor_se_falso):


```
SELECT IF(GROUPING(R.nome, C.nome)=3, 'TOTAL GERAL:', IF(GROUPING(C.nome)=1, 'TOTAL:', R.nome)) AS NOME, IF(GROUPING(C.nome)=1, '', C.nome) AS CLIENTE, FORMAT(SUM(P.valor) 2, 'de_DE') AS TOTAL
FROM pedido AS P, representante AS R, cliente AS C WHERE P.rep_id = R.rep_id AND P.cliente_id = C.cliente_ID GROUP BY R.nome, C.nome WITH ROLLUP;
```



| NOME | CLIENTE | TOTAL |
|---------------|---------------------|-----------|
| Ana Matos | Máximo Transportes | 13.000,00 |
| Ana Matos | ServCentro Serviços | 26.000,00 |
| TOTAL: | | 39.000,00 |
| Joana Santos | Indústrias União | 14.000,00 |
| Joana Santos | ServCentro Serviços | 6.000,00 |
| TOTAL: | | 20.000,00 |
| João da Silva | Máximo Transportes | 21.000,00 |
| João da Silva | MaxLog Logística | 3.000,00 |
| TOTAL: | | 24.000,00 |
| Jorge Jesus | Máximo Transportes | 9.000,00 |
| Jorge Jesus | MaxLog Logística | 2.000,00 |
| Jorge Jesus | ServCentro Serviços | 12.000,00 |

| NOME | CLIENTE | TOTAL |
|--------------|---------------------|------------|
| TOTAL: | | 23.000,00 |
| Marcos Leite | Indústrias União | 25.000,00 |
| Marcos Leite | Máximo Transportes | 1.000,00 |
| Marcos Leite | ServCentro Serviços | 13.000,00 |
| TOTAL: | | 39.000,00 |
| Maria Carmo | Indústrias União | 3.000,00 |
| Maria Carmo | Máximo Transportes | 13.000,00 |
| Maria Carmo | MaxLog Logística | 19.000,00 |
| Maria Carmo | ServCentro Serviços | 22.000,00 |
| TOTAL: | | 57.000,00 |
| TOTAL GERAL: | | 202.000,00 |

A função GROUPING() pode ter vários argumentos. Cada bit do valor retornado corresponde ao resultado relacionado a um dos argumentos.

A CLÁUSULA HAVING

A cláusula HAVING está para a cláusula GROUP BY assim como a cláusula WHERE está para o SELECT. Sua função é selecionar as linhas que formarão o resultado após a totalização da cláusula GROUP BY. Sua forma geral é:

Para se saber que argumentos retornam o valor 1 deve-se converter o valor retornado pela função para a base 2. Por exemplo, GROUPING(a, b, c) retornando o valor 5; 5 = b00000101; os valores NULL das colunas a e c são devido à cláusula GROUP BY . . . WITH ROLLUP.


```
SELECT . . . [GROUP BY {coluna1 | expressão1}, [{coluna2 | expressão2},] ...  
[WITH ROLLUP]] [HAVING condição]
```



Utilizando a consulta que calcula o valor total de pedidos por representante, selecionar apenas aqueles com total de pedidos superior a R\$ 25.000.

```
SELECT R.nome AS NOME, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL  
FROM pedido AS P, representante AS R  
WHERE P.rep_id = R.rep_id  
GROUP BY R.nome HAVING SUM(P.valor) > 25000;
```

| NOME | TOTAL |
|--------------|-----------|
| Ana Matos | 39.000,00 |
| Marcos Leite | 39.000,00 |
| Maria Carmo | 57.000,00 |

As cláusulas WHERE e HAVING são utilizadas para selecionar linhas em uma consulta. A primeira seleciona linhas individuais oriundas da junção das tabelas participantes enquanto a segunda seleciona linhas oriundas da subtotalização de linhas. Por isso, não faz muito sentido utilizar a cláusula HAVING sem GROUP BY. Outra diferença entre WHERE e HAVING é que a primeira não permite o uso de funções de coluna, enquanto a segunda, sim. Na realidade, a cláusula HAVING quase sempre tem uma função de coluna ou faz referência a uma função de coluna em SELECT.

No último exemplo, a condição de HAVING utilizou a mesma função de coluna que constava no SELECT. Por que não utilizar a condição `TOTAL > 25000` no lugar? Porque o codinome TOTAL faz referência à `FORMAT(SUM(P.valor), 2, 'de_DE')`, cujo tipo é texto e não numérico. Se no SELECT tivesse sido usado simplesmente `SUM(P.valor)`, a condição `TOTAL > 25000` poderia ser utilizada sem problema (verifique!).



- ELMASRI, R. e NAVATHE, S. B. **Sistemas de Banco de Dados**. 7ª Ed., São Paulo: Pearson, 2011.

- RAMAKRISHNAN, R., GEHRKE, J. **Sistemas de Gerenciamento de Banco de Dados**. 3ª Ed., São Paulo: McGraw-Hill, 2008.

- CORONEL, C. e ROB, P. **Sistemas de Banco de Dados - Projeto, Implementação e Gerenciamento**. 1ª Ed., São Paulo: Cengage, 2010.

- GROFF, J. R., WEINBERG, P. N. e OPPEL, A. J. **SQL: The Complete Reference**. 3ª Ed., Nova York: McGraw-Hill, 2009.

[Ir para exercício](#)