Fundamentos de Bancos de Dados: Consultas com Agrupamento, Totalização e Ordenação

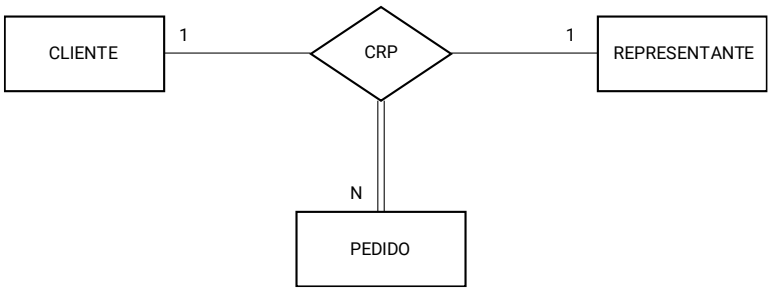
Resumo

Referências Bibliográficas

- 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.

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.

Funções de coluna

A linguagem SQL possui um conjunto de funções, chamadas de **funções de coluna**, que permitem que o resultado de uma consulta seja agrupado e sumarizado. 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**[1]**.** 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:

```
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".

[1] A inclusão da cláusula GROPU BY, apresentada mais adiante, permite o retorno de várias linhas.

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 o 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 ficar ainda 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.[2]

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:

```
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

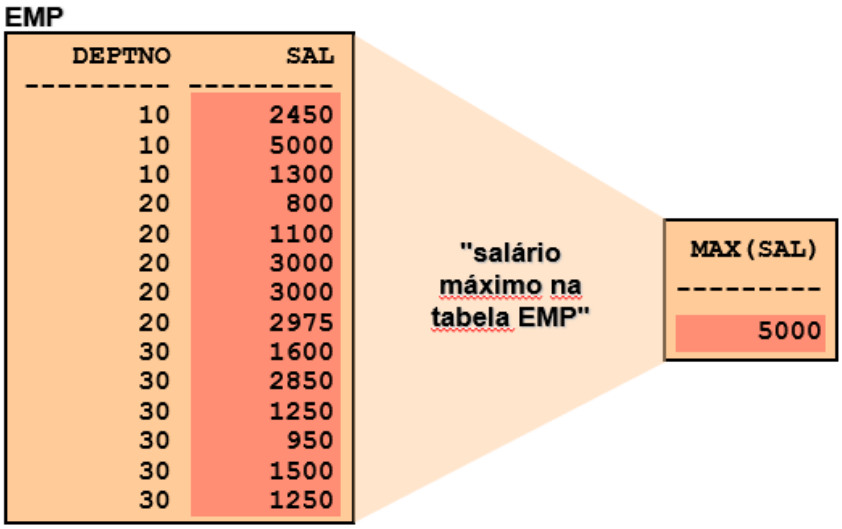
As cláusulas WHERE e HAVING são utilizadas para selecionar linhas em uma consulta. A primeira seleciona linhas individuais oriundas da junção das tabelas participantes, enquanto a segunda seleciona linhas oriundas da subtotalização de linhas. Por isso, não faz muito sentido utilizar a cláusula HAVING 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.

[2] Para se saber que argumentos retornaram 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.

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!).

Mais exemplos sobre funções de grupo

As funções de grupo operam em conjuntos de linhas para fornecer um resultado por grupo. Como no exemplo abaixo, a função Max analisa o grupo de dados do campo sal e retorna o maior valor encontrado nesse grupo.



Funções de grupo mais usadas

- **Min**: retorna o menor valor;
- **Max**: retorna o maior valor;
- **Sum:** Calcula a somatória de valores;
- **Count**: Conta a quantidade de valores;
- **Avg:** Calcula a média de valores.

É possível usar várias funções de grupo no mesmo comando, como no exemplo abaixo.

```
select count(salary) "Quantidade de Valores", sum(salary) Total, min(salary) "Menor Salário", max(salary) "Maior Salário",round(avg(salary)) "Média Salarial"
from employees;
```

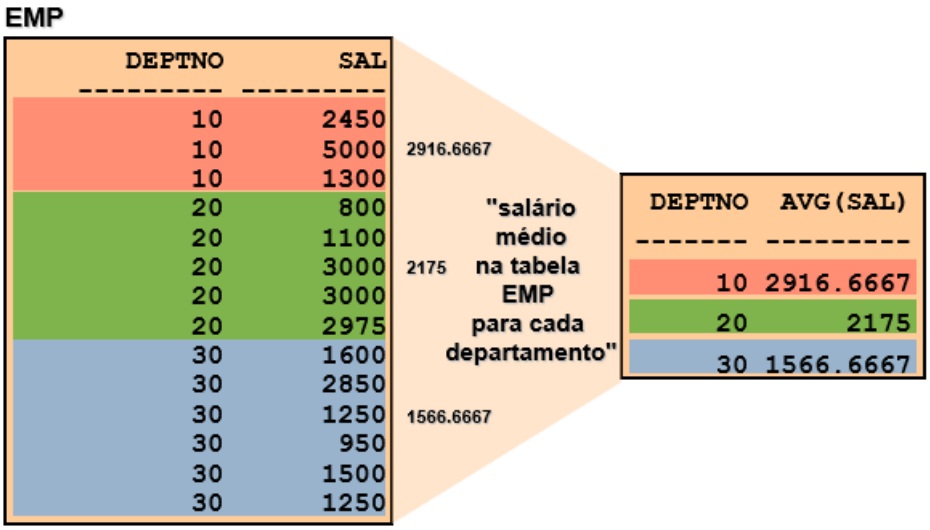
Resultado da Consulta

Todas as Linhas Extraídas: 1 em 0,003 segundos

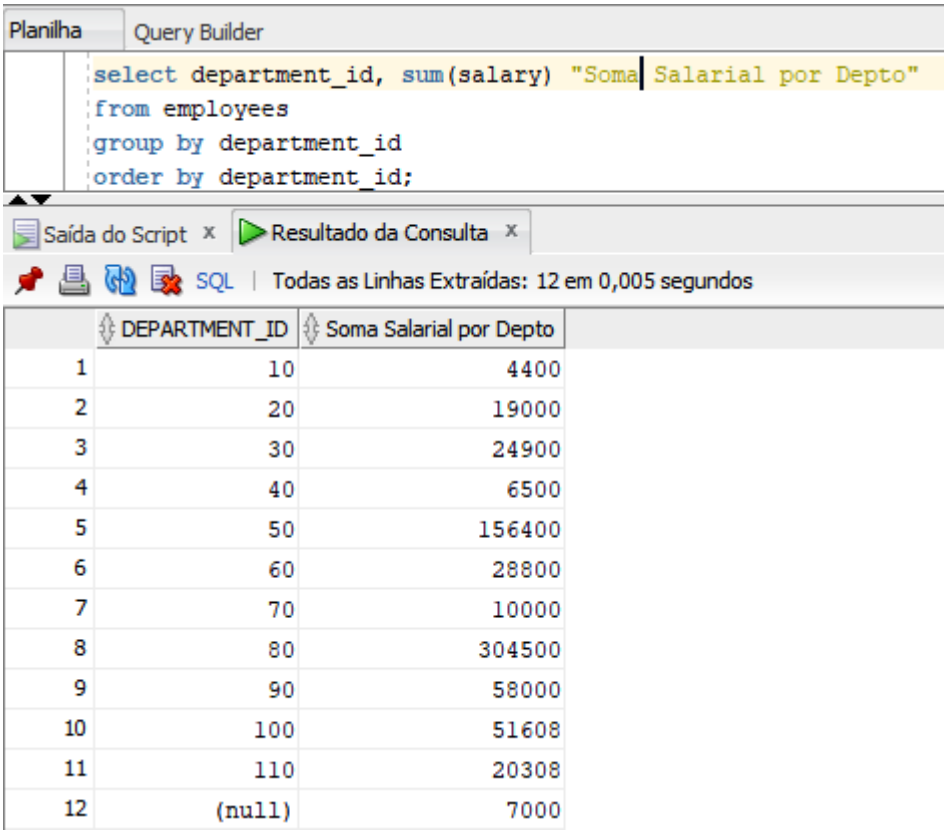
Quantidade de Valores	TOTAL	Menor Salário	Maior Salário	Média Salarial
107	691416	2100	24000	6462

Criando Grupos de Dados

É possível criar grupos de dados dentro do campo. No exemplo da figura abaixo, verificamos que são criados grupos por departamento, e dentro de cada grupo é aplicada a função avg que calcula a média salarial de cada departamento.



Divida linhas de uma tabela em grupos menores usando a cláusula GROUP BY. Observe o exemplo abaixo.



Planilha | Query Builder

```
select department_id, sum(salary) "Soma Salarial por Depto"
from employees
group by department_id
order by department_id;
```

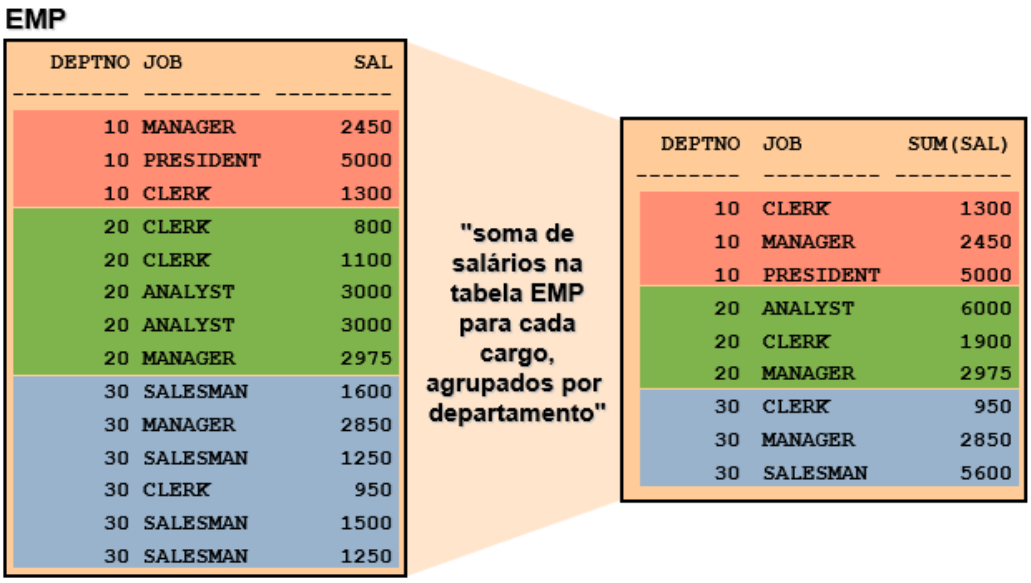
Saída do Script | Resultado da Consulta

SQL | Todas as Linhas Extraídas: 12 em 0,005 segundos

	DEPARTMENT_ID	Soma Salarial por Depto
1	10	4400
2	20	19000
3	30	24900
4	40	6500
5	50	156400
6	60	28800
7	70	10000
8	80	304500
9	90	58000
10	100	51608
11	110	20308
12	(null)	7000

Agrupando por várias colunas

Podemos criar grupos e subgrupos. Perceba que, no exemplo da figura abaixo, primeiro são criados os grupos por departamento e dentro desses grupos são criados subgrupos por cargo.



Observe o exemplo do comando abaixo.

```
select department_id,job_id,sum(salary)
from employees
group by department_id, job_id
order by department_id,job_id;
```

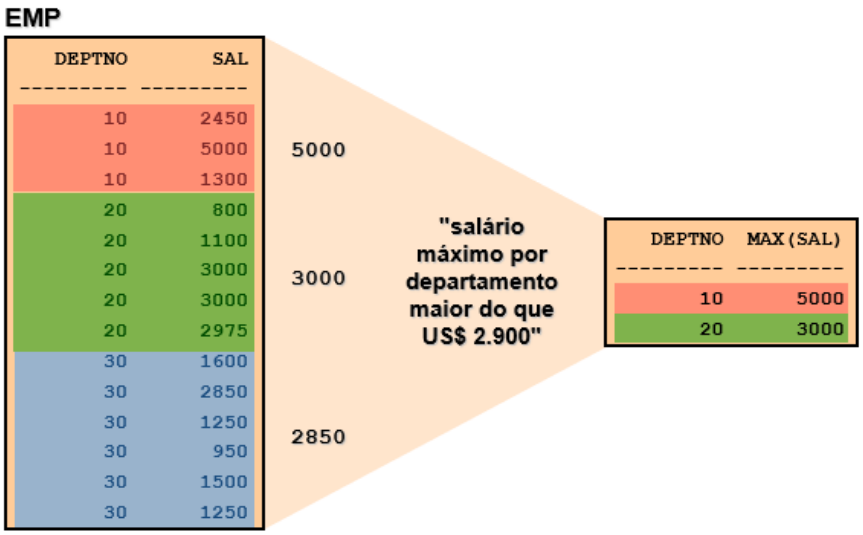
a do Script × Resultado da Consulta ×

SQL | Todas as Linhas Extraídas: 20 em 0,023 segundos

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
30	PU_CLERK	13900
30	PU_MAN	11000
40	HR_REP	6500
50	SH_CLERK	64300
50	ST_CLERK	55700
50	ST_MAN	36400
60	IT_PROG	28800
70	PR_REP	10000
80	SA_MAN	61000
80	SA_REP	243500
90	AD_PRES	24000
90	AD_VP	34000
100	FI_ACCOUNT	39600

Restringindo grupos usando a clausula HAVING

É possível impor condições aos grupos. Para tal, é necessário usar a clausula HAVING.



Usando a cláusula HAVING para restringir grupos, as linhas são agrupadas, a função de grupo é aplicada. Os grupos que correspondem à clausula HAVING são exibidos. Observe que no exemplo abaixo só serão exibidos os grupos que tiverem soma salarial maior que 20.000, que é condição imposta à função de grupo.

```
select department_id,sum(salary)
from employees
group by department_id
having sum(salary)<20000
order by department_id;
```

Resultado da Consulta | SQL | Todas as Linhas Extraídas: 5 em 0,004 segundos

	DEPARTMENT_ID	SUM(SALARY)
1	10	4400
2	20	19000
3	40	6500
4	70	10000
5	(null)	7000

O que é uma subconsulta?

De forma bem resumida, subconsultas são consultas dentro de consultas. Como o resultado de uma consulta é uma tabela, subconsultas podem ser utilizadas em qualquer parte do comando SELECT onde tabelas são utilizadas (FROM, WHERE e HAVING). De uma forma geral, subconsultas são utilizadas em duas situações:

(i) tornar mais "legível" uma consulta mais complexa e
(ii) criar consultas que não seriam possíveis com o SQL padrão.

Uma subconsulta tem exatamente a mesma forma de uma consulta normal, porém colocada entre parênteses:

```
SELECT . . . FROM {tabela1 | (subconsulta1)} [, {tabela2 | (subconsulta2)}] . . . WHERE
{condição1 | (subconsulta1)} [operador {condição2 | (subconsulta2)}] . . . GROUP BY
{coluna1 | expressão1} [, {coluna2 | expressão2}] ...
[WITH ROLLUP]] [HAVING condição];
```

Considere um esquema com três tabelas: REPRESENTANTE, CLIENTE e PEDIDO. 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 entre as três entidades foi feito através de chaves estrangeiras na tabela PEDIDO (ver material complementar).
A consulta a seguir relaciona todos os pedidos feitos no segundo semestre de 2019:

```
SELECT R.nome AS NOME, P.pedido_id, 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 AND P.data BETWEEN '2019-07-
   01' AND '2019-12-31' ORDER BY R.nome, C.nome, P.data, P.valor DESC;
```

Para ilustrar o uso de subconsultas, a mesma consulta poderia ser realizada da forma a seguir:

```
SELECT R.nome AS NOME, S.pedido_id, S.nome AS CLIENTE, S.data AS DATA, FORMAT(S.valor,
2, 'de_DE') AS VALOR
FROM representante AS R,
   (SELECT C.nome, P.data, P.valor, P.rep_id, P.pedido_id
    FROM cliente AS C, pedido AS P
    WHERE P.cliente_id = C.cliente_id AND P.data BETWEEN '2019-07-01' AND '2019-12-
    31') AS S
WHERE R.rep_id = S.rep_id ORDER BY R.nome, S.nome, S.data, S.valor DESC;
```

A junção entre CLIENTE e PEDIDO foi feita na subconsulta. Seu resultado é tratado como uma tabela qualquer. A diferença é que o codinome é obrigatório (S, no caso). Observe que não é necessário atribuir codinomes às suas colunas para referenciá-las. Entretanto, em subconsultas que tenham duas colunas com o mesmo nome, é necessário o codinome.
A subconsulta anterior produz o seguinte resultado:

nome	data	valor	rep_id	pedido_id
Máximo Transportes	2019-07-12	8000.00	4	8
Máximo Transportes	2019-07-17	12000.00	1	10
Máximo Transportes	2019-07-10	1000.00	4	11
ServCentro Serviços	2019-07-15	10000.00	5	12
ServCentro Serviços	2019-07-13	12000.00	4	21
Máximo Transportes	2019-07-12	6000.00	1	23

A junção com REPRESENTANTE produz o resultado desejado.

Subconsultas na cláusula FROM

O exemplo anterior apresentou a utilização de subconsultas na cláusula FROM. No entanto, a mesma consulta poderia ser realizada sem sua utilização. Porém, há situações em que isso não é possível ou, se é, a consulta resultante fica muito complexa e pouco legível.

Considere a consulta a seguir, que relaciona representante e clientes, totalizando por cliente, representante e total geral.

```
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;
```

NOME	CLIENTE	TOTAL
Ana Matos	Máximo Transportes	13.000,00
Ana Matos	ServCentro Serviços	26.000,00
Ana Matos	NULL	39.000,00
Joana Santos	Indústrias União	14.000,00
Joana Santos	ServCentro Serviços	6.000,00
Joana Santos	NULL	20.000,00
João da Silva	Máximo Transportes	21.000,00
João da Silva	MaxLog Logística	3.000,00
João da Silva	NULL	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
Jorge Jesus	NULL	23.000,00
Marcos Leite	Indústrias União	25.000,00
Marcos Leite	Máximo Transportes	1.000,00
Marcos Leite	ServCentro Serviços	13.000,00
Marcos Leite	NULL	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
Maria Carmo	NULL	57.000,00
NULL	NULL	202.000,00

Agora, deseja-se o mesmo formato de resultado, porém apenas com valor total de pedidos > R$ 25.000. Ou seja, a condição de seleção de linhas depende do resultado das totalizações realizadas pelo modificador WITH ROLLUP. Se a condição fosse colocada como HAVING SUM(P.valor) > 25000, seriam selecionadas todas as linhas com a coluna TOTAL > 25000 (são apenas 5 linhas; verifique!).
A solução é a utilização de uma subconsulta que seleciona apenas os representantes com valor total de pedidos > 25000 e, na consulta principal, são feitas as totalizações desejadas:

```
SELECT S.nome AS NOME, C.nome AS CLIENTE, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL
FROM representante AS R, cliente AS C, pedido AS P,
    (SELECT R.rep_id, R.nome, SUM(valor) AS TOTAL
    FROM pedido AS P, representante AS R
    WHERE P.rep_id = R.rep_id
    GROUP BY rep_id HAVING TOTAL > 25000) as S
WHERE P.rep_id = R.rep_id AND P.cliente_id = C.cliente_ID AND R.rep_id = S.rep_id
GROUP BY S.nome, C.nome WITH ROLLUP;
```

NOME	CLIENTE	TOTAL
Ana Matos	Máximo Transportes	13.000,00
Ana Matos	ServCentro Serviços	26.000,00
Ana Matos	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
Marcos Leite	NULL	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
Maria Carmo	NULL	57.000,00
NULL	NULL	135.000,00

Subconsultas na cláusula WHERE

Uma outra forma de executar a consulta anterior seria colocar, na cláusula WHERE, uma subconsulta que calculasse o valor total dos pedidos para cada representante:

```
SELECT R.nome AS NOME, C.nome AS CLIENTE, FORMAT(SUM(P.valor), 2, 'de_DE') AS TOTAL
FROM representante AS R, cliente AS C, pedido AS P,
WHERE P.rep_id = R.rep_id AND P.cliente_id = C.cliente_ID AND
    (SELECT SUM(valor)
    FROM pedido as P
    WHERE P.rep_id = R.rep_id) > 25000
GROUP BY R.nome, C.nome WITH ROLLUP;
```

Neste último exemplo, na cláusula WHERE da subconsulta, há uma referência qualificada à coluna *P.rep_id*. Apesar de a tabela REPRESENTANTE, utilizada na consulta principal, ter uma coluna com o mesmo nome, o qualificador não é necessário. Quando o SGBDR encontra o nome de uma coluna em uma subconsulta, primeiramente ele busca esta coluna nas tabelas dentro da subconsulta. Já a coluna *R.rep_id* deve ser qualificada, indicando que pertence à tabela REPRESENTANTE e não à tabela PEDIDO.
Subconsultas podem ser aninhadas sem restrição. É preciso tomar cuidado com a referência a colunas externas: sempre use qualificadores.

Os operadores IN E EXISTS

- **O operador IN**
  Os operadores IN e sua forma negativa, NOT IN, testam se um valor pertence/não pertence a um conjunto. Normalmente, estes operadores são utilizados juntamente com uma subconsulta que produz o conjunto de valores, porém ele pode ser utilizado em qualquer situação. A consulta abaixo relaciona todos os pedidos realizados pelos clientes nos dias 12, 13, 17 e 18/05/2019 utilizando o operador IN:

```
SELECT P.data AS DATA, R.nome AS NOME, C.nome AS CLIENTE
FROM representante AS R, cliente AS C, pedido AS P
WHERE P.rep_id = R.rep_id AND P.cliente_id = C.cliente_ID AND
P.data IN ('2019-05-12', '2019-05-13', '2019-05-17', '2017-05-18');
```

DATA	NOME	CLIENTE
2019-05-17	Maria Carmo	Indústrias União
2019-05-17	Marcos Leite	Indústrias União
2019-05-12	Maria Carmo	MaxLog Logística
2019-05-13	Joana Santos	ServCentro Serviços

O seu uso com uma subconsulta é apresentado a seguir, onde são relacionados os representantes que não têm pedidos para o cliente 3:

```
SELECT R.nome AS NOME FROM representante AS R
WHERE R.rep_id NOT IN (SELECT DISTINCT rep_id FROM pedido WHERE cliente_id = 3);
```

NOME
Joana Santos
Ana Matos
Marcos Leite

A subconsulta seleciona os representantes que têm pedidos para o cliente 3 e a consulta principal testa se o representante está no resultado da subconsulta.
O operador IN pode ser usado para testar tuplas[3] de valores. A consulta a seguir relaciona os representantes que têm pedidos para o cliente 2 em julho de 2019:

```
SELECT R.nome AS NOME FROM representante AS R
WHERE (R.rep_id, 7) IN (SELECT DISTINCT rep_id, MONTH(data) FROM pedido WHERE cliente_id
= 2);
```

NOME
João da Silva
Jorge Jesus

O modificador DISTINCT utilizado na subconsulta não é necessário. O resultado seria o mesmo se uma tupla aparecesse repetida. O uso deste modificador exige que o resultado seja ordenado, implicando um tempo maior de execução.

- **O operador EXISTS**

O operador EXISTS e sua forma negativa, NOT EXISTS, verificam se uma subconsulta produziu algum resultado ou não. Ele só pode ser utilizado com subconsultas e jamais retorna o valor NULL (por quê?). A consulta abaixo verifica relaciona, para cada representante, os clientes para os quais não há pedido:

```
SELECT R.nome AS NOME, C.nome AS CLIENTE
FROM representante AS R, cliente AS C
WHERE NOT EXISTS
   (SELECT * FROM pedido WHERE cliente_id = C.cliente_id AND rep_id = R.rep_id);
```

NOME	CLIENTE
João da Silva	Indústrias União
João da Silva	ServCentro Serviços
Joana Santos	Máximo Transportes
Joana Santos	MaxLog Logística
Jorge Jesus	Indústrias União
Ana Matos	Indústrias União
Ana Matos	MaxLog Logística
Marcos Leite	MaxLog Logística

Na consulta anterior, para cada par (*rep_id*, *cliente_id*), a subconsulta retorna os pedidos correspondentes. Se há algum pedido, o par não é incluído no resultado da consulta principal. Observe que foram feitas referências a duas colunas de tabelas da consulta principal que não estavam relacionadas na cláusula SELECT. Isto é perfeitamente possível porque, a partir do momento que uma tabela é incluída na cláusula FROM, todas as

[3] Aqui, o termo *tupla* representa um conjunto ordenado de valores: (*a*, *b*, ..., *n*).

suas colunas ficam disponíveis, mesmo que não apareçam no SELECT. Observe também que não importam as colunas que compõem o resultado da subconsulta. A única coisa que importa é se a subconsulta retorna alguma linha ou não.

Os modificadores ANY E ALL

Estes dois modificadores atuam em conjunto com os operadores relacionais =, <>, <, <=, > e >= na comparação de valores com o resultado de subconsultas.

O modificador ANY

Os operadores relacionais retornam VERDADEIRO ou FALSO na comparação de dois valores. O modificador ANY permite que um valor seja comparado com todas as linhas de uma subconsulta. Se pelo menos uma comparação retornar VERDADEIRO, o resultado é VERDADEIRO. Para isto, a subconsulta deve retornar apenas uma coluna. A consulta a seguir relaciona os representantes que tenham pelo menos um pedido que represente mais de 2% de sua meta:

```
SELECT R.nome AS NOME FROM representante AS R
WHERE 0.02 * R.meta < ANY (SELECT valor FROM pedido WHERE rep_id = R.rep_id);
```

NOME
João da Silva
Marcos Leite

Sempre se pode substituir o modificador ANY pelo operador EXISTS. Basta colocar a condição testada pelo ANY na cláusula WHERE da subconsulta (verifique!):

```
SELECT R.nome AS NOME FROM representante AS R
WHERE EXISTS (SELECT * FROM pedido WHERE rep_id = R.rep_id AND valor > 0.02 * R.meta);
```

O modificador ALL

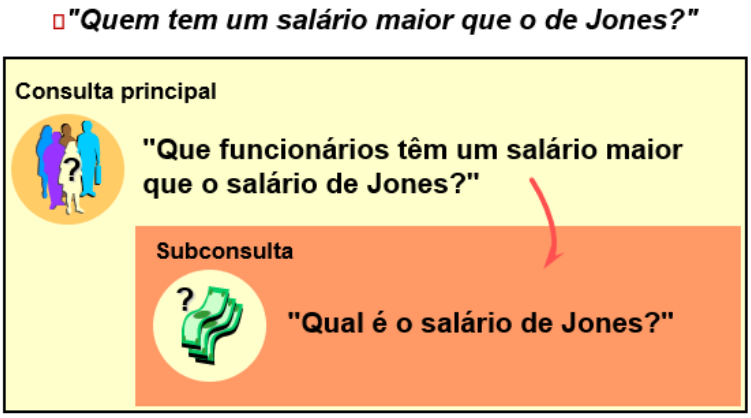
O modificador ANY retorna VERDADEIRO se pelo menos uma comparação retornar VERDADEIRO. O modificador ALL retorna VERDADEIRO se o resultado de todas as comparações for VERDADEIRO. A consulta a seguir relaciona os clientes em que todos os pedidos tenham valor superior a R$ 5.000 :

```
SELECT C.nome AS NOME FROM cliente AS C
WHERE 5000 < ALL (SELECT valor FROM pedido WHERE cliente_id = C.cliente_id);
```

NOME
ServCentro Serviços

Mais exemplos com subconsulta

Subconsulta, que também pode ser chamada de subselect, trata-se de uma consulta dentro da outra. Esse recurso é utilizado para resolver um problema. Observe o exemplo da figura abaixo.



A subconsulta (consulta interna) é executada uma vez antes da consulta principal. O resultado da subconsulta é usado pela consulta principal (consulta externa).

Estrutura do comando da subconsulta

```
SELECT select_list
FROM   tabela
HERE   operador expr
```

Consulta externa

```
(SELECT    select_list
   FROM          tabela);
```

Consulta interna

O problema a ser resolvido é descobrir quais funcionários ganham mais que o funcionário Jones. Esse problema precisa ser resolvido em duas etapas, ou seja, a consulta interna descobre quanto ganha o funcionário Jones e a consulta externa, com o valor retornado pela consulta interna, descobre quais funcionários ganham mais que o funcionário Jones.
O salário do funcionário Jones é 2975. Portanto, o comando da subconsulta abaixo retorna quais funcionários ganham mais que 2975.

```
select ename,sal
from emp
where sal> (select sal
            from emp
            where ename='JONES');
```

esultado da Consulta

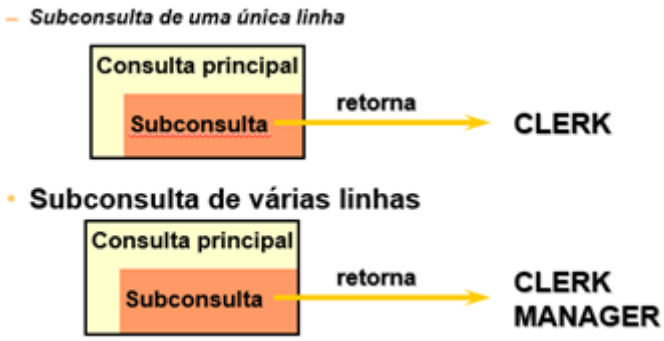
SQL | Todas as Linhas Extraídas: 3 em 0,376

	ENAME	SAL
1	SCOTT	3000
2	KING	5000
3	FORD	3000

Diretrizes para o uso de subconsultas

- Coloque as subconsultas entre parênteses.
- Não adicione uma cláusula ORDER BY a uma subconsulta.
- Use operadores de uma única linha com subconsultas que retornam uma única linha.
- Use operadores de várias linhas com subconsultas que retornam várias linhas.

Tipos de subconsultas



Operadores para subconsulta de uma única linha

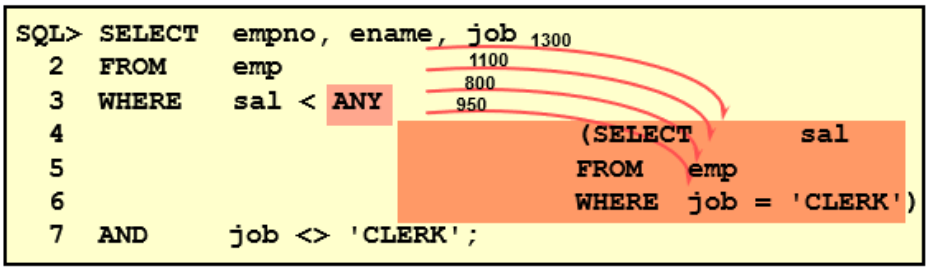
Operador	Significado
=	Igual a
>	Maior do que
>=	Maior do que ou igual a
<	Menor do que
<=	Menor ou igual a
<>	Diferente de

Operadores para subconsulta de várias linhas

Operador	Significado
IN	Igual a qualquer membro na lista
ANY	Compare o valor a cada valor retornado pela subconsulta
ALL	Compare o valor a todo valor retornado pela subconsulta

Exemplo de uso do operador ANY em subconsulta de várias linhas

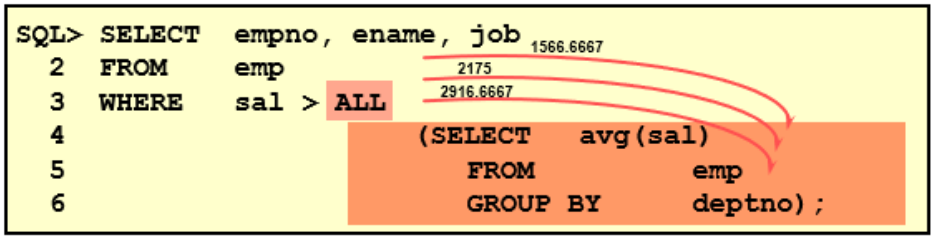
A leitura desse operador é “qualquer valor” retornado pela consulta interna. Observe que, no exemplo abaixo, serão selecionados os funcionários que ganham mais que qualquer funcionário que tenha o cargo clerk.



EMPNO	ENAME	JOB
7654	MARTIN	SALESMAN
7521	WARD	SALESMAN

Exemplo de uso do operador ALL em subconsulta de várias linhas

A leitura desse operador é “todos os valores” retornados pela consulta interna. Observe que, no exemplo abaixo, serão selecionados os funcionários que ganham mais que todas as médias salariais por departamento.



EMPNO	ENAME	JOB
7839	KING	PRESIDENT
7566	JONES	MANAGER
7902	FORD	ANALYST
7788	SCOTT	ANALYST

Exemplo de uso da subconsulta com o comando create table (“cópia de tabela”)

Perceba que, no exemplo abaixo, a tabela copia_emp é criada com toda a estrutura e os dados da tabela emp. É importante ressaltar que esse comando não copia as constraints.

```
create table copia_emp as (select * from emp);

select * from copia_emp;
```

Resultado da Consulta | Saída do Script

Tarefa concluída em 1,068 segundos

```
Table COPIA_EMP criado.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7788	SCOTT	ANALYST	7566	19/04/87	3000		20
7839	KING	PRESIDENT		17/11/81	5000		10

Exemplo de Uso da subconsulta com o comando insert (“cópia de registros”)

Perceba que, no exemplo abaixo, os registros da tabela dept são inseridos na tabela copia_dept.

```
insert into copia_dept (select * from dept);
select * from copia_dept;
```

do Script | Resultado da Consulta

SQL | Todas as Linhas Extraídas: 8 em 0,049 segundos

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON