



# Sub-consultas



**Curso de Introdução a Oracle 11g:  
SQL/Avançado**

**Prof.: Marlon Mendes Minussi**

**[marlonminussi@gmail.com](mailto:marlonminussi@gmail.com)**



# Objetivos

- Descrever os tipos de problemas que sub-consultas podem resolver;
- Definir sub-consultas;
- Listar os tipos de sub-consultas;
- Escrever sub-consultas do tipo single-row e multiple-row.



# Utilizando uma Sub-consulta para resolver um problema

- “quais são os atendimentos de maior valor que os atendimentos do paciente 2?”
- Para resolver este problema, você precisa de duas consultas: uma consulta para encontrar o maior valor dos atendimentos do paciente 2 e uma segunda consulta para encontrar quem tem maior valor de atendimento que este valor.
- Você pode resolver este problema combinando as duas consultas e colocando uma consulta dentro da outra.
- Uma consulta interna ou sub-consulta retorna um valor que é utilizado pela consulta externa ou consulta principal. Utilizar uma sub-consulta é equivalente a executar duas consultas seqüenciais e utilizar o resultado da primeira consulta como o valor de procura da segunda consulta.



# Sub-consultas

- Sintaxe:

```
SELECT select_list  
FROM      table  
WHERE expr operator  
        (SELECT select_list  
          FROM table);
```

- Uma sub-consulta é um comando SELECT embutido em uma cláusula de outro comando SELECT. Você pode construir comandos poderosos utilizando sub-consultas. Eles podem ser muito úteis quando você precisa selecionar linhas de uma tabela com uma condição que depende dos dados da própria tabela.



# Sub-consultas

- Você pode colocar sub-consultas em várias cláusulas SQL:
  - Cláusula WHERE
  - Cláusula HAVING
  - Cláusula FROM
- Sintaxe:  
*operator*: inclui um operador de comparação como >, = ou IN
- Nota: operadores de comparação entram em duas classes: operadores do tipo single-row (>, =, >=, <, <>, <=) e operadores do tipo multiple-row (IN, ANY, ALL).
- A sub-consulta é freqüentemente chamada de SELECT aninhado, sub-consulta, ou comando SELECT interno. A sub-consulta geralmente é executada primeiro, e seu resultado é utilizado para completar a condição da consulta principal ou externa.



# Utilizando uma Sub-consulta

- Ex.:

```
SELECT cod_atendimento, valor  
FROM atendimento  
WHERE          valor >  
              (SELECT MAX (valor)  
                FROM atendimento  
                WHERE cod_pac = 2);
```

- No exemplo, a consulta interna determina o maior valor dos atendimento do paciente 2. A consulta externa recebe o resultado da consulta interna e utiliza este resultado para exibir todos os atendimentos que possuem valor maior que este valor.





# Diretrizes para Utilização de Sub-consultas

- Uma sub-consulta deve ser incluída entre parênteses;
- Uma sub-consulta deve estar no lado direito do operador de comparação;
- Sub-consultas não podem conter uma cláusula ORDER BY. Você pode ter somente uma cláusula ORDER BY para um comando SELECT, e se especificada ela deve ser a última cláusula do comando SELECT principal;
- Duas classes de operadores de comparação são utilizadas em sub-consultas: os operadores do tipo single-row e os operadores do tipo multiple-row.



# [ Tipos de Consultas

- Sub-consultas single-row: consultas que retornam apenas uma linha a partir do comando `SELECT` interno;
- Sub-consultas multiple-row: consultas que retornam mais de uma linha a partir do comando `SELECT` interno;





# Sub-consultas Single-Row

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



# [ Sub-consultas Single-Row

- Uma sub-consulta do tipo single-row retorna uma linha a partir do comando SELECT interno. Este tipo de sub-consulta utiliza um operador do tipo single-row. O gráfico acima exibe uma lista dos operadores single-row.
- Exemplo:
- Mostre os atendimentos cujo o valor seja maior que o máximo dos valores dos atendimentos efetuados pelo médico cujo código é 3.

```
SELECT cod_atendimento, valor
FROM atendimento
WHERE          valor >
              (SELECT MAX (valor)
               FROM atendimento
               WHERE cod_med = 3);
```



# [ Executando Sub-consultas Single-Row

Ex.:

```
SELECT cod_pac, nome
FROM paciente
WHERE estado =
      ( SELECT estado
        FROM paciente
        WHERE cod_pac = 5)
AND TO_CHAR (dt_nasc, 'MON') =
      (SELECT TO_CHAR (dt_nasc, 'MON')
       FROM paciente
       WHERE cod_pac = 11);
```

**RS**

**NOV**



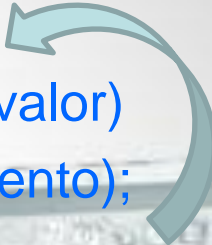
# [ Executando Sub-consultas Single-Row

- Um comando SELECT pode ser considerado como um bloco de consulta. O exemplo exhibe os pacientes cujo estado é o mesmo do paciente 5 e cujo mês de nascimento é igual que o do paciente 11.
- O exemplo consiste de três blocos de consulta: a consulta externa e as duas consultas internas. Os blocos de consulta internos são executados primeiro, produzindo os resultados da consulta: RS e NOV, respectivamente. O bloco de consulta externo é então processado e utiliza os valores retornados pelas consultas internas para completar as suas condições de pesquisa. Ambas as consultas internas retornam valores únicos (RS e NOV, respectivamente), sendo chamadas de sub-consultas single-row.
- Nota: as consultas externa e interna podem obter dados de tabelas diferentes.



# Utilizando Funções de Grupo em uma Sub-consulta

Ex.:

```
SELECT cod_atendimento, dt_atendimento, valor, desconto  
FROM      atendimento  
WHERE     valor >   
           (SELECT AVG (valor)  
            FROM atendimento);
```

**82,4166667**

- Você pode exibir dados de uma consulta principal utilizando uma função de grupo em uma sub-consulta para retornar uma única linha. A sub-consulta está entre parênteses e é colocada após o operador de comparação.
- O exemplo exibe os atendimentos que possuem o um valor maior que a média de todos os valores dos atendimentos da tabela ATENDIMENTO. A função de grupo AVG retorna um único valor (**82,4166667**) para a consulta externa.




# [ Cláusula HAVING com Sub-consultas

- Ex.: Encontre as datas e as médias dos valores dos atendimentos que sejam igual ao mínimo da média dos exames:

```
SELECT  dt_atendimento, AVG(valor)
FROM    atendimento
GROUP   BY
        dt_atendimento
HAVING  AVG (valor) =
        (SELECT MIN(AVG (valor))
         FROM  exame
         GROUP BY dt_exame);
```

70,5








# Cláusula HAVING com Sub-consultas

- Ex.:

```
SELECT dt_atendimento, MIN (valor)
FROM atendimento
GROUP BY
      dt_atendimento
HAVING MIN (valor) >
      (SELECT MIN (valor)
       FROM exame
       WHERE dt_exame < '31-MAR-11');
```



70,5

- O Servidor Oracle executa a sub-consulta, e os resultados são retornados para a cláusula HAVING da consulta principal. O comando SQL do exemplo exibe todos os atendimentos agrupados por data do atendimento que possuem um valor mínimo maior que o valor mínimo dos exames dos dias '31-MAR-11' (70,5).



# Cláusula HAVING com Sub-consultas

- Qual o Erro deste comando?

```
SELECT dt_atendimento, AVG (valor)
FROM      atendimento
GROUP BY
          dt_atendimento
HAVING AVG (valor) =
        (SELECT MAX (valor)
         FROM atendimento
         GROUP BY dt_atendimento);
```

- Um erro comum em sub-consulta é mais de uma linha ser retornada para a sub-consulta do tipo single-row.
- No comando SQL do exemplo, a sub-consulta possui uma cláusula GROUP BY (dt\_atendimento), que implica que a sub-consulta devolverá múltiplas linhas, uma para cada grupo encontrado.



# [ Cláusula HAVING com Sub-consultas

- Neste caso, o resultado da sub-consulta será:

MAX(V valor)

-----  
100  
100  
100  
70,5  
100  
75,5

- A consulta externa recebe os resultados da sub-consulta e utiliza estes resultados em sua cláusula HAVING . A cláusula HAVING contém um operador igual (=), operador de comparação do tipo single-row que compara apenas um valor. O operador (=) não aceita mais de um valor a partir da sub-consulta e conseqüentemente gera o erro.
- Para corrigir este erro, mude o operador (=) para IN.



# Este Comando Funcionará?

```
SELECT dt_atendimento, MIN (valor)
FROM atendimento
GROUP BY
      dt_atendimento
HAVING MIN (valor) >
      (SELECT MIN (valor)
       FROM atendimento
       WHERE dt_atendimento < '31-MAR-70');
```

**No rows selected**

- Um problema comum com sub-consulta é, nenhuma linha ser retornada pela consulta interna. No comando SQL do exemplo, a sub-consulta contém uma cláusula WHERE (dt\_atendimento < '31-MAR-70'). Presumivelmente, a intenção é achar o menor valor de atendimento do dia 31-MAR-70. O comando parece estar correto mas não seleciona nenhuma linha quando executado.
- O problema é que não existem atendimentos nesta data. Assim, a sub-consulta não retorna nenhuma linha. A consulta externa recebe os resultados da sub-consulta (null) e utiliza estes resultados na cláusula WHERE. A consulta externa não encontra nenhum atendimento de data menor que '31-MAR-70'.



# Sub-consultas do tipo Multiple-Row

Operador	Significado
IN	Igual a qualquer membro da lista
ANY	Compara um valor com cada valor retornado por uma sub-consulta
ALL	Compara um valor com todos os valores retornados por uma sub-consulta



# [ Sub-consultas do tipo Multiple-Row

- Sub-consultas que retornam mais que uma linha são chamadas sub-consultas multiple-row. Você utiliza operadores multiple-row, em vez de operadores single-row, com uma sub-consulta multiple-row. O operador multiple-row aceita um ou mais valores.
- Ex.:

```
SELECT cod_atendimento, cod_pac, valor  
FROM atendimento  
WHERE valor IN  
      (SELECT MIN(valor)  
        FROM atendimento  
        GROUP BY
```

```
dt_atendimento);
```

70,5 e 75,5





# [ Sub-consultas do tipo Multiple-Row

- Encontre os atendimentos que possuem valores de 70.5 ou 75.
- A lista fixa de valores produz um resultado que contém duas linhas: 70.5 ou 75.5. O bloco de consulta principal é então processada e utiliza os valores retornados pela consulta interna para contemplar sua condição de pesquisa.
- Ex.:


```
SELECT cod_atendimento, cod_pac, valor  
FROM atendimento  
WHERE valor IN (70.5, 75.5);
```



# Utilizando o Operador ANY em Sub-consultas Multiple-row

- Ex.:

```
SELECT cod_atendimento, dt_atendimento, valor, desconto
FROM      atendimento
WHERE      valor < ANY
           (SELECT valor
            FROM atendimento
            WHERE dt_atendimento = '31-03-11')
AND dt_atendimento <> '31-03-11';
```



**70.5 e 100**


- O operador ANY compara um valor retornado por uma sub-consulta. O exemplo exibe os atendimentos cujo valor é menor que o valor de qualquer atendimento do dia = '31-03-11' e que não são deste dia. O maior valor deste dia é 100. O comando SQL exibe todos os atendimentos que não possuem este dia, mas que o valor seja menor que 100.
  - < ANY menor que o máximo.
  - > ANY maior que mínimo.
  - = ANY: é equivalente a IN.



# Utilizando o Operador ALL em sub-consultas Multiple-row

- Ex.:

```
SELECT cod_atendimento, dt_atendimento, valor, desconto
FROM      atendimento
WHERE      valor > ALL
          (SELECT valor
           FROM atendimento
           WHERE dt_atendimento = '01-04-11');
```



**75.5 e 100**

- O operador ALL compara um valor com todos os valores retornados por uma sub-consulta. O exemplo exhibe os atendimentos cujo valor é maior que o valor de todos os atendimentos do dia '31-03-11'. O maior valor é 100, assim a consulta retorna aqueles atendimentos cujo total é maior que 100, neste exemplo nenhum valor.
  - > ALL significa mais que o máximo.
  - < ALL significa menos que mínimo.
- O operador NOT pode ser utilizado com os operadores IN, ANY e ALL.



# [ Sub-consultas Multiple-Column

- Objetivos
- Escrever uma sub-consulta multiple-column
- Descrever e explicar o comportamento de sub-consultas quando valores nulos são recuperados
- Escrever sub-consultas em uma cláusula FROM



# [ Sub-consultas Multiple-column

- Para comparar duas ou mais colunas, você deve escrever uma combinação na cláusula WHERE utilizando os operadores lógicos. Sub-consultas do tipo multiple-column permitem combinar condições WHERE duplicadas em uma única cláusula WHERE.
- Sintaxe:

```
SELECT column, column, ...  
FROM table  
WHERE (column, column, ...) IN  
      (SELECT column, column, ...  
       FROM table  
       WHERE condition);
```



# Utilizando Sub-consultas Multiple-column

- Ex.:

```
SELECT cod_atendimento, dt_atendimento, valor, desconto
FROM atendimento
WHERE (valor, NVL(desconto,0)) IN
      (SELECT valor, NVL(desconto,0)
       FROM atendimento
       WHERE valor < 100);
```

- O exemplo acima é uma sub-consulta do tipo multiple-column, uma vez que sub-consulta retorna mais de uma coluna. Ele compara a coluna valor e a coluna desconto. Ele exibe o cod\_atendimento, dt\_atendimento, desconto, valor de qualquer atendimento cujo o valor seja menor que 100.





# [ Comparação de Colunas

- Comparações de coluna em uma sub-consulta multiple-column podem ser do tipo pairwise ou nonpairwise.
- Se você quiser uma comparação tipo pairwise na cláusula WHERE, cada linha do comando SELECT deve ter o mesmo valor e o mesmo desconto, para atendimentos com descontos.
- E se você quiser uma comparação tipo nonpairwise (um cross product), você deve utilizar uma cláusula WHERE com múltiplas condições.



# [ Sub-consulta tipo Nonpairwise

- Ex.:

```
SELECT cod_atendimento, dt_atendimento, valor, desconto
FROM atendimento
WHERE valor IN
    (SELECT valor
     FROM atendimento
     WHERE dt_atendimento < '08-ABR-11')
AND NVL(desconto, 0) IN
    (SELECT NVL(desconto, 0)
     FROM atendimento
     WHERE dt_atendimento < '08-ABR-11');
```

- O exemplo faz uma comparação tipo nonpairwise das colunas. Exibe o cod\_atendimento, dt\_atendimento, desconto e o valor de qualquer atendimento cujo valor e o desconto correspondam ao valor e desconto de qualquer atendimento com data de atendimento menor que '08-ABR-11'.



# [ Sub-consultas Tipo Pairwise

- Ex.:

```
SELECT cod_atendimento, dt_atendimento, valor, desconto
FROM      atendimento
WHERE      (NVL(desconto, 0), valor) IN
           (SELECT NVL(desconto, 0), valor
            FROM      atendimento
            WHERE dt_atendimento < '08-ABR-11');
```

- Os resultados das duas últimas consultas são diferentes.
- Os resultados foram obtidos por causa dos dados específicos da tabela atendimento.



# Valores Nulos em uma Sub-consulta

- Ex.:

```
SELECT cod_atendimento, cod_pac
FROM    atendimento
WHERE   desconto NOT IN
        ( SELECT desconto
          FROM atendimento);
no rows selected.
```

- O comando SQL no exemplo tenta exibir todos os atendimentos que possuem desconto. Logicamente, este comando SQL deveria ter retornado linhas. Entretanto, o comando SQL não retorna nenhuma linha. Um dos valores retornados pela consulta interna é um valor nulo e conseqüentemente toda a consulta não retorna nenhuma linha. A razão é que todas as condições que comparam um valor nulo resultam um nulo. Portanto, sempre que for provável eu valores nulos façam parte do conjunto resultante de uma sub-consulta, não utilize o operador NOT IN. O operador NOT IN equivalente a != ALL.



# Valores Nulos em uma Sub-consulta

- Ex.:

```
SELECT cod_atendimento, cod_pac
FROM   atendimento
WHERE  desconto NOT IN
      ( SELECT desconto
        FROM atendimento
        WHERE cod_atendimento>9);
```

no rows selected.

- Para testar e se certificar que um dos valores retornados pela consulta interna sendo um valor nulo e conseqüentemente toda a consulta não retornará nenhuma linha.



# Valores Nulos em uma Sub-consulta

- Observe que o valor nulo como parte do conjunto de uma sub-consulta não será um problema se você estiver utilizando o operador IN. O operador IN é equivalente a = ANY.
- Ex.:  

```
SELECT cod_atendimento, cod_pac, desconto  
FROM atendimento  
WHERE desconto IN  
      (SELECT desconto  
        FROM atendimento);
```





# Utilizando uma Sub-consulta na Cláusula FROM

- Ex.:

```
SELECT at.cod_atendimento, at.cod_pac, at.valor, qry.media  
FROM atendimento at, (SELECT dt_exame, AVG(valor) MEDIA  
                      FROM exame  
                      GROUP BY  
                      dt_exame) qry  
WHERE at.dt_atendimento = qry.dt_exame  
AND at.valor > qry.media;
```

- Você pode utilizar uma sub-consulta na cláusula FROM de um comando SELECT. O exemplo acima exibe os atendimentos que possuem valor maior que média do valor dos exames feitos em um mesmo dia de atendimento.



# Bibliografias

- HEUSER, Carlos Alberto, “Projeto de Banco de Dados”. 4ª Edição (Série Livros Didáticos Nº 4), Porto Alegre: Instituto de Informática da UFRGS, 2001.
- RAMIREZ, Elmasri, S. Navathe a “Sistemas de Banco de Dados”. 4ª edição, São Paulo: Editora Pearson, 2005.
- SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S., “Sistemas de Banco de Dados”. Tradução da 5ª edição, Rio de Janeiro: Editora Campus, 2006.
- SILVA, Robson Soares, “Oracle 10g Express Edition: Guia de Instalação, Configuração e Administração”. 1ª edição, São Paulo: Érica, 2007.