



# Exibindo Dados a partir de Múltiplas Tabelas



Curso de Introdução a Oracle 10g:  
SQL

Prof.: Marlon Mendes Minussi  
[marlonminussi@gmail.br](mailto:marlonminussi@gmail.br)

- Escrever comandos **SELECT** para acessar dados de mais de uma tabela utilizando diversos tipos de **JOINS**;
- Visualizar dados que geralmente não corresponde a condição de **JOIN** utilizando **OUTER JOINS**;
- Executar um **JOIN** de uma tabela com ela mesma (**SELF JOIN**);
- Este capítulo discute como obter dados de várias tabelas, utilizando os diferentes métodos possíveis.



# [ Obtendo Dados a partir de Múltiplas Tabelas ]

Executar Linha de Comandos SQL

```
SQL> desc paciente
```

| Nome      | Nulo?    | Tipo         |
|-----------|----------|--------------|
| -----     |          |              |
| COD_PAC   | NOT NULL | NUMBER(38)   |
| NOME      | NOT NULL | VARCHAR2(30) |
| DT_NASC   |          | DATE         |
| RG        |          | NUMBER(10)   |
| SEXO      |          | CHAR(1)      |
| FONE      |          | VARCHAR2(12) |
| EST_CIVIL |          | VARCHAR2(10) |
| END       |          | VARCHAR2(50) |
| CIDADE    |          | VARCHAR2(30) |
| ESTADO    |          | CHAR(2)      |

SQL>

Executar Linha de Comandos SQL

```
SQL> desc atendimento
```

| Nome            | Nulo?    | Tipo         |
|-----------------|----------|--------------|
| -----           |          |              |
| COD_ATENDIMENTO | NOT NULL | NUMBER(38)   |
| COD_PAC         | NOT NULL | NUMBER(38)   |
| COD_MED         | NOT NULL | NUMBER(38)   |
| DT_ATENDIMENTO  |          | DATE         |
| DIAGNOSTICO     |          | VARCHAR2(50) |
| VALOR           |          | NUMBER(10,2) |
| DESCONTO        |          | NUMBER(10,2) |

SQL>

- Às vezes você precisa utilizar dados de mais de uma tabela. No exemplo acima, você tem duas tabelas diferentes onde precisamos exibir o nome dos pacientes com atendimento em um determinado dia.
  - COD\_PAC (código paciente) existe nas tabelas PACIENTE E ATENDIMENTO.
  - COD\_ATENDIMENTO existe na tabela ATENDIMENTO.
  - NOME (nome paciente) existe na tabela PACIENTE.
- Para produzir uma consulta, você precisa unir as tabelas PACIENTE e ATENDIMENTO e acessar os dados a partir de ambas.



# O que é um Join?

- Sintaxe:

```
SELECT table1.column,    table2.column  
FROM    table1, table2  
WHERE    table1.column1 = table2.column2;
```

- Quando dados de mais de uma tabela são requeridos, uma condição de JOIN é utilizada.
- Linhas em uma tabela podem ser unidas a linhas em outra tabela de acordo com valores comuns que existem em colunas correspondentes, que normalmente são colunas de chaves primarias e estrangeiras.



# O que é um Join?

- Para exibir dados de duas ou mais tabelas relacionadas, escreva uma condição de JOIN simples na cláusula WHERE.
- Sintaxe:
- *table.column*: denota a tabela e coluna a partir da qual os dados são recuperados.
- *table1.column1* = *table2.column2*: é a condição que une (ou relaciona) as tabelas.

- Quando escrever um comando `SELECT` que relaciona tabelas, preceda o nome das colunas com o nome da tabela para obter maior clareza e melhorar o acesso ao banco de dados;
- Se o mesmo nome de coluna existir em mais de uma tabela, o nome de coluna deve ser prefixado com o nome da tabela;
- Para unir  $n$  tabelas, você precisa de um mínimo de  $(n-1)$  condições de join. Portanto, para unir quatro tabelas, um mínimo de três joins é necessário.





# Produto Cartesiano

- Quando uma condição de JOIN é inválida ou completamente omitida, o resultado é um produto cartesiano no qual serão exibidas todas as combinações das linhas.
- Todas as linhas da primeira tabela são unidas a todas as linhas da segunda tabela.
- Um produto cartesiano tende a gerar um número grande de linhas, e seu resultado é raramente útil.
- Você sempre deveria incluir uma condição de JOIN válida na cláusula WHERE.



# Gerando um Produto Cartesiano

- Um produto cartesiano é gerado se uma condição de JOIN for omitida.
- O exemplo abaixo exibira os nomes dos pacientes e os descontos nos atendimentos.
- Uma vez que nenhuma cláusula WHERE foi especificada, todas as linhas (11 linhas) da tabela paciente são unidas as linhas (12 linhas) da tabela atendimento, gerando um total de 132 linhas na consulta.
- Ex.:  

```
SELECT nome, desconto  
FROM paciente, atendimento;
```



# Tipos de Joins

- Existem dois tipos principais de condições de join:
  - Equijoins
  - Non-equijoins
- Métodos adicionais de JOIN incluem o seguinte:
  - Outer joins
  - Self joins
  - Set Operators



# O que é um Equijoin?

- Para determinar o nome de um paciente em um atendimento, você compara o valor na coluna `cod_pac` da tabela `PACIENTE` com os valores de `cop_pac` da tabela `ATENDIMENTO`.
- A relação entre as tabelas `PACIENTE` e `ATENDIMENTO` é um equijoin, ou seja os valores na coluna `paciente.cop_pac` e `atendimento.cod_pac` em ambas as tabelas devem ser iguais. Frequentemente, este tipo de JOIN envolve complementos de chave primária e chave estrangeira.
- NOTA: Equijoins também são chamados de joins simples ou de inner joins.

# Recuperando registros com Equijoins

- Ex.:

```
SELECT paciente.cod_pac CODIGO_PACIENTE,  
       paciente.nome NOME,  
       atendimento.cod_atendimento CODIGO_ATENDIMENTO,  
       atendimento.cod_pac CODIGO_PACIENTE  
FROM   paciente, atendimento  
WHERE  paciente.cod_pac=atendimento.cod_pac;
```

```
SELECT paciente.cod_pac CODIGO_PACIENTE,  
       paciente.nome NOME,  
       atendimento.cod_atendimento CODIGO_ATENDIMENTO,  
       atendimento.cod_pac CODIGO_PACIENTE  
FROM   paciente INNER JOIN atendimento  
ON     paciente.cod_pac = atendimento.cod_pac;
```

# Recuperando registros com Equijoins

- No exemplo acima:
- A cláusula **SELECT** especifica os nomes de coluna a recuperar:
  - Nome do paciente e o código do paciente são colunas da tabela **PACIENTE**;
  - Número do atendimento e o código do paciente que são colunas da tabela **ATENDIMENTO**;
- A cláusula **FROM** especifica as duas tabelas que o banco de dados deve acessar:
  - Tabela **PACIENTE**;
  - Tabela **ATENDIMENTO**;
- A cláusula **WHERE** especifica como as tabelas serão unidas:
  - `paciente.cod_pac=atendimento.cod_pac`
- Uma vez que as colunas de mesmo nome em tabelas realcionadas, deve ser prefixada com o nome da tabela para evitar ambigüidade.



# [ Qualificando Nomes de Colunas Ambíguos ]

- Você precisa qualificar os nomes das colunas na cláusula WHERE com o nome da tabela para evitar ambigüidade. Sem os prefixos de tabelas, a coluna VALOR e DESCONTO por exemplo, poderia ser da tabela EXAME ou da tabela ATENDIMENTO. É necessário adicionar o prefixo da tabela para executar a consulta.
- Se não existir nenhum nome de coluna comum entre as duas tabelas, não há necessidade de qualificar as colunas. Entretanto, você ganhará desempenho utilizando o prefixo de tabela porque você informa exatamente para o Servidor Oracle onde procurar pelas colunas.
- A exigência para qualificar os nomes de colunas ambíguas também é aplicável para colunas que podem ser ambíguas em outras cláusulas como SELECT ou ORDER BY.

# Condições Adicionais de Pesquisa com o Operador AND

- Em adição ao join, você pode ter critérios adicionais na cláusula WHERE.
- Por exemplo, para exibir o código do paciente, o nome, o código dos atendimentos e os valores, apenas para o cliente 'Renato Gaúcho', você precisa de uma condição adicional na cláusula WHERE.

• Ex.:

```
SELECT paciente.cod_pac CODIGO_PACIENTE,  
       paciente.nome NOME,  
       atendimento.cod_atendimento CODIGO_ATENDIMENTO,  
       atendimento.valor VALOR_ATENDIMENTO  
FROM   paciente, atendimento  
WHERE  paciente.cod_pac=atendimento.cod_pac  
AND nome='Renato Gaúcho';
```





# Utilizando Alias de Tabela

- Ex.:

```
SELECT paciente.cod_pac CODIGO_PACIENTE,  
       paciente.nome NOME,  
       atendimento.cod_atendimento CODIGO_ATENDIMENTO,  
       atendimento.valor VALOR_ATENDIMENTO  
  
FROM   paciente, atendimento  
WHERE  paciente.cod_pac=atendimento.cod_pac  
AND    nome='Renato Gaucho';  
  
SELECT pac.cod_pac CODIGO_PACIENTE,  
       pac.nome NOME,  
       at.cod_atendimento CODIGO_ATENDIMENTO,  
       at.valor VALOR_ATENDIMENTO  
  
FROM   paciente pac, atendimento at  
WHERE  pac.cod_pac=at.cod_pac  
AND    pac.nome='Renato Gaucho';
```



# Utilizando Alias de Tabela

- Qualificar os nomes de coluna com os nomes de tabela pode consumir muito tempo, particularmente se os nomes de tabelas forem longos.
- Você pode utilizar alias de tabela em vez de nomes de tabela. Da mesma maneira que um alias de coluna fornece um outro nome para uma coluna, um alias de tabela fornece um outro nome para uma tabela.
- Alias de tabela ajudam a manter o código SQL menor, utilizando menos memória.
- Observe que no exemplo os alias de tabela são identificados na cláusula FROM. O nome da tabela é especificado por completo, seguido por um espaço e então pelo alias da tabela. A tabela PACIENTE recebeu o alias PAC, enquanto que a tabela ATENDIMENTO recebeu o alias AT.



# Utilizando Alias de Tabela

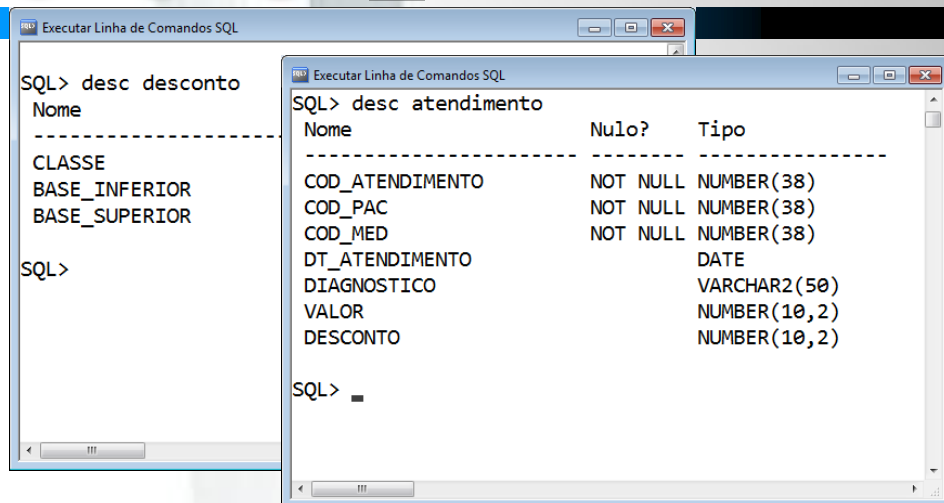
- **Diretrizes**
- Alias de tabelas podem ter até 30 caracteres de tamanho, porém, quanto menor melhor;
- Se um alias de tabela for utilizado para um nome de tabela específico na cláusula FROM, então este alias de tabela deve ser utilizado para substituir o nome da tabela em todo o comando SELECT;
- Alias de tabelas devem ser significativos;
- O alias de tabela só é válido para o comando SELECT no qual foi declarado.
-

# [ Relacionando mais de duas Tabelas ]

- Às vezes você pode precisar unir mais de duas tabelas. Por exemplo, para exibir o nome do paciente, os atendimentos, o valor da atendimento e o valor do exame do paciente 'Steve Jobs' você terá que relacionar as tabelas PACIENTE, ATENDIEMNT0 e EXAME.
- Ex.:

```
SELECT p.nome NOME, a.cod_atendimento ATENDIMENTO,  
       e.cod_exame EXAME,  
       a.valor VALOR_ATENDIMENTO,  
       e.valor VALOR_EXAME  
FROM   paciente p, atendimento a, exame e  
WHERE  p.cod_pac = a.cod_pac  
AND    a.cod_atendimento = e.cod_atendimento  
AND    p.nome = 'Steve Jobs';
```

# Non-Equi Joins



The image shows two overlapping SQL command windows. The background window displays the structure of the 'desconto' table, and the foreground window displays the structure of the 'atendimento' table.

```
SQL> desc desconto
Nome
-----
CLASSE
BASE_INFERIOR
BASE_SUPERIOR
SQL>
```

```
SQL> desc atendimento
Nome          Nulo?   Tipo
-----
COD_ATENDIMENTO NOT NULL NUMBER(38)
COD_PAC        NOT NULL NUMBER(38)
COD_MED        NOT NULL NUMBER(38)
DT_ATENDIMENTO          DATE
DIAGNOSTICO             VARCHAR2(50)
VALOR                   NUMBER(10,2)
DESCONTO                NUMBER(10,2)
SQL>
```

- Uma relação entre tabela ATENDIMENTO e a tabela DESCONTO pode ser definida como um non-equi join, significando que nenhuma coluna da tabela ATENDIMENTO corresponde diretamente a uma coluna da tabela DESCONTO.
- A relação entre as duas tabelas é a coluna DESCONTO da tabela ATENDIMENTO que está entre os valores definidos nos campos BASE\_INFERIOR e BASE\_SUPERIOR da tabela DESCONTO.
- A relação é obtida utilizando um operador diferente de igual (=).



# Recuperando Registros com Non-Equi Joins

- Ex.:

```
SELECT a.cod_atendimento ATENDIMENTO,  
       d.classe DESCONTO  
FROM   atendimento a, desconto d  
WHERE  NVL(a.desconto,0) BETWEEN d.base_inferior  
                                AND d.base_superior;
```

- O exemplo acima cria um non-equi join para avaliar o nível de desconto de um atendimento. O desconto deve estar entre qualquer faixa de valor de menor e maior valor.
- É importante observar que todos os atendimentos aparecem precisamente uma única vez quando a consulta é executada. Nenhum atendimento é repetido na lista. Existem duas razões para isto.





# Recuperando Registros com Non-Equi Joins

- Nenhuma das linhas da tabela de níveis de desconto contém graus que se sobrepõem. Ou seja, o valor desconto de um atendimento está entre os valores de menor e maior de uma das linhas da tabela de níveis de desconto;
- Todos os descontos dos atendimentos estão dentro dos limites fornecidos pela tabela de níveis de desconto. Ou seja, nenhum atendimento tem desconto menos que o menor valor contido na coluna BASE\_INFERIOR ou mais que o maior valor contido na coluna BASE\_SUPERIOR.
  - Nota: outros operadores como ' $\leq$ ' e ' $\geq$ ' podem ser utilizados, porém o operador BETWEEN é o mais simples. Lembre-se de especificar o menor valor primeiro e o maior valor por ultimo quando utilizar o operador BETWEEN. Alias de tabela foram especificados por razões de desempenho, não por causa de possíveis ambigüidades.
  - Utilizamos a função NVL para garantir a comparação com os valores diferentes de nulo.

# Outer Joins

- Se uma linha não satisfaz a condição de join, esta linha não aparecerá no resultado da consulta.
- 
- Por exemplo na condição de equijoin das tabelas **PACIENTE** e **ATENDIMENTO** alguns pacientes não possuem nenhum atendimento.



# Recuperando Registros sem Correspondência Direta Utilizando Outer Joins

- Sintaxe:

```
SELECT table.column,      table.column  
FROM   table1, table2  
WHERE  table1.column (+) = table2.column;  
  
SELECT table.column,      table.column  
FROM   table1, table2  
WHERE  table1.column = table2.column (+);
```

- A(s) linha(s) sem correspondência podem ser recuperadas se um operador de OUTER JOIN for utilizado na condição de join. O operador é o sinal de adição colocado entre parênteses (+), e é colocado no lado “lado” do JOIN que é deficiente de informação. Este operador possui o efeito de criar uma ou mais linhas nulas, para as quais uma ou mais linhas da tabela não deficiente podem ser unidas.



# Recuperando Registros sem Correspondência Direta Utilizando Outer Joins

- Sintaxe:
  - *Table1.column = :* é a condição que relaciona (joins) as tabelas
  - *Table2.column (+):* é o símbolo de outer join; ele pode ser colocado em qualquer lado da condição da cláusula, porém, não pode ser colocado em ambos os lados. Coloque o símbolo onde OUTER JOIN logo após o nome da coluna da tabela que pode não possuir dados para corresponder as linhas da outra tabela.



# Utilizando Outer Joins

- Ex.:

```
SELECT p.nome, a.cod_atendimento  
FROM paciente p, atendimento a  
WHERE p.cod_pac(+)=a.cod_pac;
```

- O exemplo acima exibe o nome do paciente, o código do atendimento.
- Os pacientes que não possuem atendimento são exibidos.



# Restrições do Outer Join

- O operador OUTER JOIN só pode aparecer em um dos lados da expressão, o lado que não possui informação correspondente.
- Ele retorna essas linhas a partir de uma tabela que não possui correspondência direta para a outra tabela.
- Uma condição envolvendo um OUTER JOIN não pode utilizar o operador IN ou ser unida para outra condição pelo operador OR.



# LEFT JOIN e RIGHT JOIN

- LEFT JOIN seleciona os dados da esquerda de relação.

- Ex.:

```
SELECT p.nome, a.*
```

```
FROM paciente p LEFT JOIN atendimento a
```

```
ON p.cod_pac = a.cod_pac;
```

- equivale.

- Ex.:

```
SELECT p.nome, a.cod_atendimento
```

```
FROM paciente p, atendimento a
```

```
WHERE p.cod_pac=a.cod_pac(+);
```



# LEFT JOIN e RIGHT JOIN

- RIGHT JOIN seleciona os dados da direita de relação.

- Ex.:

```
SELECT p.nome, a.*
```

```
FROM paciente p RIGHT JOIN atendimento a
```

```
ON p.cod_pac = a.cod_pac;
```

- equivale

- Ex.:

```
SELECT p.nome, a.cod_atendimento
```

```
FROM paciente p, atendimento a
```

```
WHERE p.cod_pac(+) = a.cod_pac;
```

# CROSS JOIN

- Suponhamos que temos uma tabela PACIENTE e ATENDIMENTO. Se desejássemos obter uma listagem combinando ambas tabelas de tal forma que cada paciente aparecesse junto a cada atendimento, utilizaríamos a seguinte exemplo:

```
SELECT p.nome, a.diagnostico  
FROM paciente p CROSS JOIN atendimento a;
```

# Relacionando uma Tabela com Ela Mesma (Self Joins)

- As vezes você precisa relacionar um tabela com ela mesma.

- Ex.:

```
SELECT a.cod_atendimento ATENDIMENTO, a.diagnostico ||  
    possui valor de R$' ||at.valor DIAGNOSTICO_VALOR  
FROM atendimento a, atendimento at  
WHERE a.cod_atendimento = at.cod_atendimento;
```

- O exemplo acima relaciona a tabela ATENDIMENTO com ela mesma. Para simular duas tabelas na cláusula FROM, existem dois alias, chamados ATENDIMENTO e DIAGNOSTICO\_VALOR, para a mesma tabela, ATENDIMENTO.