5

Consulta de Dados de Várias Tabelas

Objetivos deste Capítulo

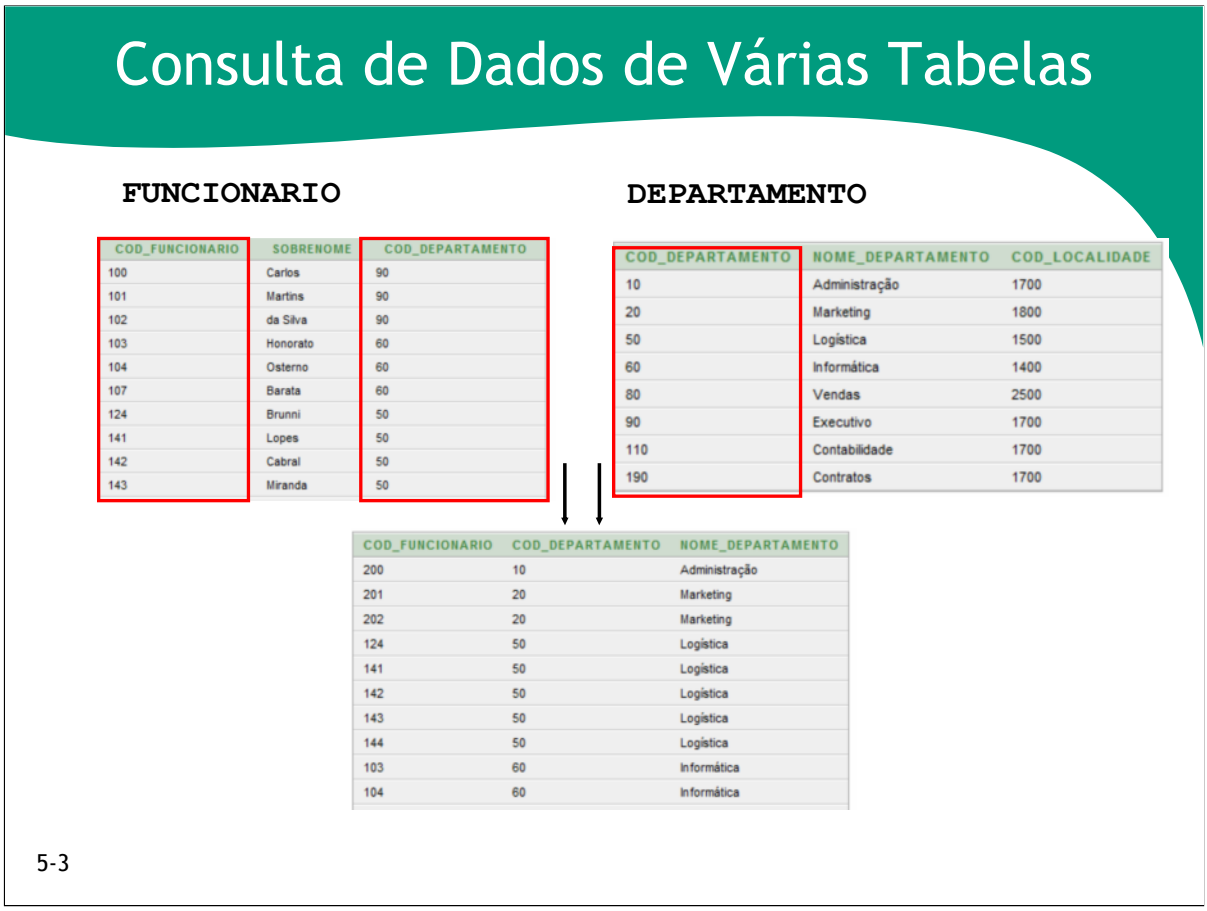
- Ao concluir este capítulo, você poderá:
  - Usar comandos `SELECT` para retornar dados de mais de uma tabela
  - Efetuar um Join de uma tabela com ela mesma através de Auto-Join
  - Retornar dados mesmo que eles não atendam a condição de join usando Outer Joins
  - Criar um produto cartesiano das linhas de duas ou mais tabelas

5-2

Objetivos deste Capítulo

Este capítulo trata de como você poderá obter dados de mais de uma tabela através de Joins.

Serão vistos os diversos tipos de Join, como por exemplo, Auto-Joins, Outer Joins e a criação de produtos cartesianos.



Consulta de Dados de Várias Tabelas

Quando for necessário criar um consulta que retorne dados de mais de uma tabela, você deverá usar o recurso do SQL de Join. No exemplo do slide, o relatório exibe dados de duas tabelas diferentes:

- Os códigos dos funcionários estão na tabela `FUNCIONARIO`.
- Os códigos dos departamentos estão nas tabelas `FUNCIONARIO` e `DEPARTAMENTO`.
- Os nomes de departamento estão na tabela `DEPARTAMENTOS`.

Para criar uma consulta que traga o código do funcionário, código do departamento e nome do departamento, você precisa juntar as tabelas `FUNCIONARIO` e `DEPARTAMENTO` e acessar os dados dessas duas tabelas formando um único resultado. A condição de junção entre as duas tabelas será o código do departamento.

Tipos de Join

- Estes são aos tipos de Join compatíveis com o padrão SQL:1999:
  - Cross Joins
  - Natural Joins
  - Joins com a Cláusula `USING`
  - Right Outer Join
  - Left Outer Join
  - Full Outer Join

5-4

Tipos de Join

Até a versão 9i a sintaxe de Joins de comandos SELECT no Oracle era proprietária, porém a partir da versão 9i o Oracle suporta o padrão ANSI SQL:1999.

Neste capítulo veremos as duas sintaxes, porém é preferível que você crie consultas usando a sintaxe ANSI SQL:1999 no desenvolvimento de novas consultas.

Os tipos de Join compatíveis com o padrão ANSI SQL:1999 estão listados no slide.

Sintaxe de Joins

- Padrão ANSI SQL:1999:

```
SELECT   tabela1.coluna, tabela2.coluna
FROM     tabela1
[NATURAL JOIN tabela2] |
[JOIN tabela2 USING (nome_de_coluna1)] |
[JOIN tabela2
  ON (tabela1.nome_de_coluna1 = tabela2.nome_de_coluna2)]|
[LEFT|RIGHT|FULL OUTER JOIN tabela2
  ON (tabela1.nome_de_coluna1 = tabela2.nome_de_coluna2)]|
[CROSS JOIN tabela2];
```

- Proprietária Oracle:

```
SELECT tabela1.coluna, tabela2.coluna
FROM   tabela1, tabela2
WHERE  tabela1.nome_de_coluna1 = tabela2.nome_de_coluna2;
```

5-5

Sintaxe de Joins

tabela1.coluna indica a tabela e a coluna das quais os dados são recuperados.

NATURAL JOIN une duas tabelas com base no mesmo nome de coluna.

JOIN *tabela* USING *nome_de_coluna1* executa uma operação de junção por igualdade com base no nome da coluna .

JOIN *tabela ON (tabela1.nome_de_coluna1 = tabela2.nome_de_coluna2)* executa uma operação de junção por igualdade com base na condição da cláusula ON.

LEFT/RIGHT/FULL OUTER executa Outer Joins.

CROSS JOIN retorna um produto cartesiano das duas tabelas.

A partir de agora as consultas estão se tornando mais complicadas. Para melhorar a legibilidade e manutenibilidade das consultas e para evitar erros por ambiguidade no nome das colunas, você deve usar apelidos de tabelas (também chamados de variáveis de tupla), por exemplo:

```
SELECT f.nome, f.sobrenome, f.cod_departamento
FROM funcionario f
WHERE f.data_nascimento > TO_DATE('01/01/1995','DD/MM/YYYY');
```

Natural Joins

- O uso da cláusula `NATURAL JOIN` faz com que o Join seja feito com base em todas as colunas de nomes iguais das duas tabelas.
- Serão selecionadas as linhas das duas tabelas que têm valores iguais em todas as colunas correspondentes.
- Caso as colunas com nomes iguais tenham tipos de dados diferentes, o Oracle retornará um erro.

5-6

Natural Joins

É possível unir tabelas automaticamente com base nas colunas das duas tabelas com tipos de dados e nomes correspondentes. Para uni-las você deverá usar a cláusula `NATURAL JOIN`.

A união entre as duas tabelas só pode ocorrer em colunas com os mesmos nomes e tipos de dados. Se as colunas tiverem nomes idênticos, mas tipos de dados diferentes o `NATURAL JOIN` trará um erro.

Natural Joins

```
SELECT cod_departamento, nome_departamento,
       cod_localidade, cidade
FROM   departamento
NATURAL JOIN localidade ;
```

COD_DEPARTAMENTO	NOME_DEPARTAMENTO	COD_LOCALIDADE	CIDADE
60	Informática	1400	Recife
50	Logística	1500	São Paulo
10	Administração	1700	Porto Alegre
90	Executivo	1700	Porto Alegre
110	Contabilidade	1700	Porto Alegre
190	Contratos	1700	Porto Alegre
20	Marketing	1800	Toronto
80	Vendas	2500	Oxford

5-7

Natural Joins (continuação)

No exemplo do slide, a tabela `LOCALIDADE` é unida à tabela `DEPARTMENTO` pela coluna `COD_LOCALIDADE`, que é a única coluna com o mesmo nome nas duas tabelas. Se houvessem outras colunas comuns, ou seja, de mesmo nome e tipo de dados, o NATURAL JOIN usaria todas elas.

Para implementar outras restrições em um Natural Join você pode usar a cláusula `WHERE`. O exemplo a seguir limita as linhas da saída àquelas com um código de departamento igual a 20 ou 50:

```
SELECT  cod_departamento, nome_departamento,
        cod_localidade, cidade
FROM    departamento
NATURAL JOIN localidade
WHERE   cod_departamento IN (20, 50);
```

Cláusula USING

- Para especificar as colunas a serem usadas na junção você poderá usar a cláusula USING.
- Quando houver correspondência em mais de uma coluna nas tabelas, mas você só deseja usar uma coluna, a cláusula USING deve ser usada ao invés do NATURAL JOIN.
- Não prefixe as colunas referenciadas na cláusula USING com o nome ou apelido de tabela.
- As cláusulas NATURAL JOIN e USING são mutuamente exclusivas.

5-8

Cláusula USING

O NATURAL JOIN usa todas as colunas com nomes e tipos de dados correspondentes para unir as tabelas. É possível usar a cláusula USING para especificar apenas as colunas que deverão ser usadas em uma junção. As colunas referenciadas na cláusula USING não devem ser prefixadas pelo nome ou apelido de tabela em qualquer ponto do comando SQL.

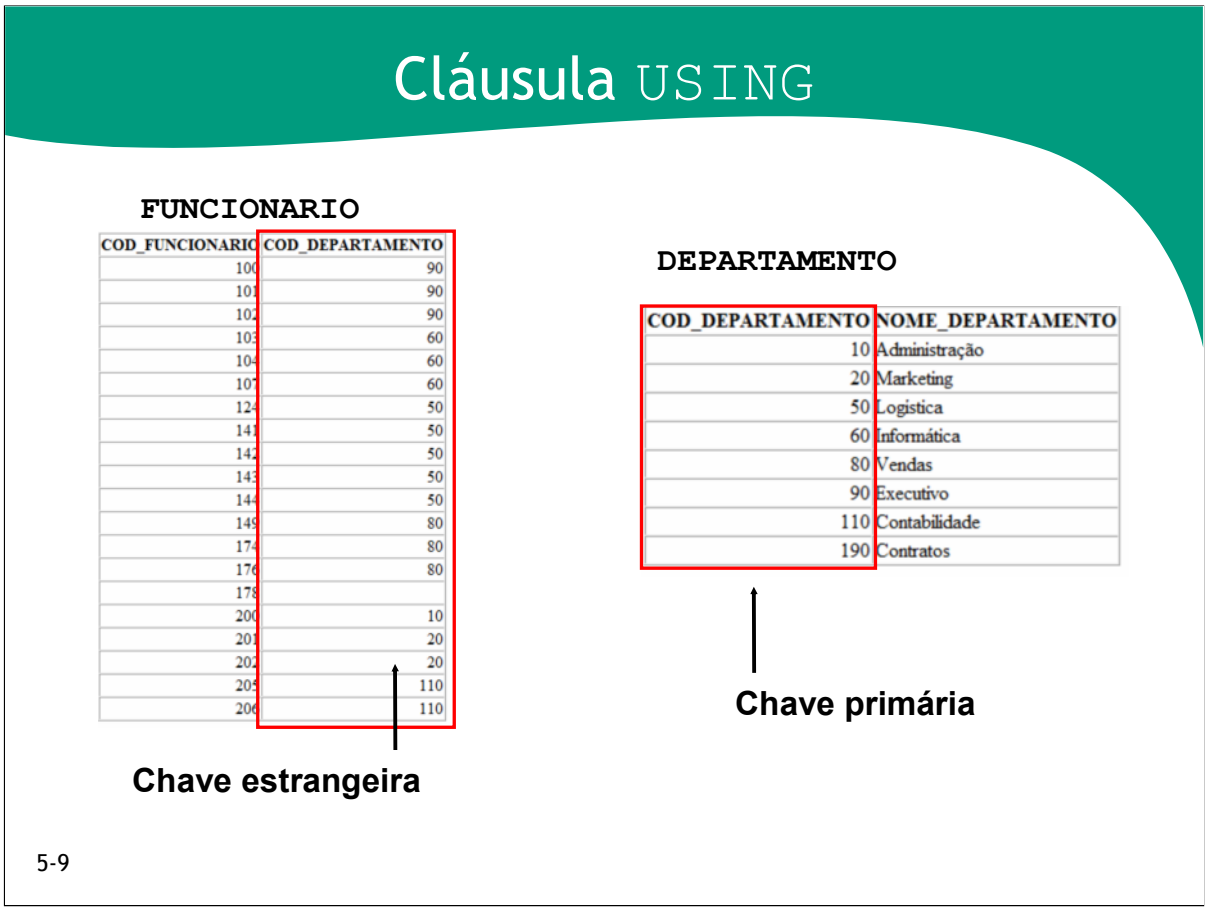
Por exemplo, este comando é válido:

```
SELECT l.cidade, d.nome_departamento
FROM   localidade l JOIN departamento d USING (cod_localidade)
WHERE  cod_localidade = 1400;
```

O comando a seguir é inválido, pois COD_LOCALIDADE está prefixado com o apelido da tabela na cláusula WHERE:

```
SELECT l.cidade, d.nome_departamento
FROM localidade l JOIN departamento d USING(cod_localidade)
WHERE d.cod_localidade = 1400;
ORA-25154: parte da coluna da cláusula USING não pode ter
qualificador
```

A mesma restrição também é aplicada aos NATURAL JOINs. Portanto, as colunas com o mesmo nome nas duas tabelas devem ser usadas sem prefixos de nome ou apelido de tabela.



Cláusula USING (continuação)

Para determinar o nome do departamento que um funcionário faz parte, compare o valor da coluna COD_DEPARTAMENTO na tabela FUNCIONARIO com o valor de COD_DEPARTAMENTO na tabela DEPARTAMENTO. O relacionamento entre as tabelas FUNCIONARIO e DEPARTAMENTO é uma junção simples (ou junção interna ou equijoin), isto é, os valores da coluna COD_DEPARTAMENTO nas duas tabelas devem ser iguais. Normalmente esse tipo de junção envolve uma chave estrangeira com a sua chave primária de referência.

Cláusula USING

```
SELECT funcionario.cod_funcionario, funcionario.sobrenome,
       departamento.cod_localidade, cod_departamento
FROM   funcionario JOIN departamento
USING (cod_departamento);
```

COD_FUNCIONARIO	SOBRENOME	COD_LOCALIDADE	COD_DEPARTAMENTO
200	Trajano	1700	10
201	Thacker	1800	20
202	Kramer	1800	20
124	Brunni	1500	50
141	Lopes	1500	50
142	Cabral	1500	50
143	Miranda	1500	50
144	Chaves	1500	50
103	Honorato	1400	60
104	Osterno	1400	60
107	Barata	1400	60
149	Schultz	2500	80
174	Sue	2500	80
176	Voorhees	2500	80
100	Carlos	1700	90
101	Martins	1700	90
102	da Silva	1700	90
205	Almeida	1700	110
206	Nascimento	1700	110

5-10

Cláusula USING (continuação)

O exemplo do slide une a coluna COD_DEPARTAMENTO das tabelas FUNCIONARIO e DEPARTAMENTO e indica o local de trabalho de um funcionário.

Nomes de Colunas Ambíguos

- Para qualificar nomes de colunas que fazem parte de mais de uma tabela, essas colunas devem ser prefixadas com nome ou apelido da tabela.
- O uso de prefixos de tabela melhora o desempenho de execução dos comandos SQL.
- Para distinguir as colunas com nomes idênticos mas que residem em tabelas diferentes, use apelidos de coluna.
- Não use apelidos em colunas identificadas na cláusula `USING` e listadas em alguma parte do comando SQL.

5-11

Nomes de Colunas Ambíguos

Deve-se qualificar os nomes das colunas com o nome ou apelido da tabela para evitar ambiguidades. Sem os prefixos das tabelas, a coluna `COD_DEPARTAMENTO` no comando `SELECT` poderá ser da tabela `FUNCIONARIO` ou `DEPARTAMENTO`. É necessário adicionar o prefixo da tabela para executar a consulta:

```
SELECT funcionario.cod_funcionario, funcionario.sobrenome,
       departamento.cod_departamento,
departamento.cod_localidade
  FROM   funcionario JOIN departamento
  ON     funcionario.cod_departamento =
departamento.cod_departamento;
```

Se não houver nomes de colunas comuns entre as duas tabelas, não será preciso qualificar as colunas. Porém, o uso do prefixo da tabela melhora o desempenho, pois você informa ao Oracle exatamente onde localizar as colunas e não deixa isso para que seja decidido por ele.

Ao efetuar uma operação de junção com a cláusula `USING`, você não poderá qualificar uma coluna usada nessa própria cláusula. Além disso, se essa coluna for usada em alguma parte do comando SQL, ela não poderá ser utilizada como apelido.

Apelidos de Tabelas

- Os apelidos de tabelas são usados para simplificar as consultas e para melhorar o seu desempenho.

```
SELECT e.cod_funcionario, e.sobrenome,
       d.cod_localidade, cod_departamento
FROM   funcionario e JOIN departamento d
USING (cod_departamento);
```

5-12

Apelidos de Tabelas

Prefixar os nomes das colunas com os nomes completos das tabelas pode consumir muito tempo, principalmente se os nomes das tabelas forem longos. Você pode usar os *apelidos das tabelas* ao invés dos nomes completos das tabelas. Assim como um apelido de coluna dá outro nome a uma coluna, um apelido de tabela dá outro nome a uma tabela dentro do comando. Os apelidos de tabelas reduzem o tamanho do comando SQL reduzindo consequentemente o uso de memória.

Os apelidos de tabelas são identificados na cláusula `FROM` do exemplo do slide. O nome da tabela é especificado por inteiro, seguido por um espaço e, depois, o apelido da tabela. A tabela `FUNCIONARIO` recebeu o apelido f, e a tabela `DEPARTAMENTO`, o apelido d.

- Um apelido de tabela pode conter até 30 caracteres, mas é recomendável especificar o menor nome possível.
- Se um apelido de tabela for usado para um nome de tabela específico na cláusula `FROM`, ele deverá ser substituído pelo nome da tabela em toda a instrução `SELECT`.
- Os apelidos de tabelas devem ser significativos.
- O apelido de tabela é válido somente para o comando `SELECT` atual.

Cláusula ON

- Use a cláusula ON para especificar condições arbitrárias ou colunas a serem utilizadas em Joins.
- O uso da cláusula ON faz com que a condição de Join fique separada de outros filtros de pesquisa da cláusula WHERE.
- A cláusula ON facilita a compreensão do comando SQL.

```
SELECT e.cod_funcionario, e.sobrenome, e.cod_departamento,
       d.cod_departamento, d.cod_localidade
FROM   funcionario e JOIN departamento d
ON     (e.cod departamento = d.cod departamento);
```

COD_FUNCIONARIO	SOBRENOME	COD_DEPARTAMENTO	COD_DEPARTAMENTO	COD_LOCALIDADE
200	Trajano	10	10	1700
201	Thacker	20	20	1800
202	Kramer	20	20	1800
124	Brunni	50	50	1500
141	Lopes	50	50	1500
142	Cabral	50	50	1500
143	Miranda	50	50	1500
144	Chaves	50	50	1500
103	Hongrato	60	60	1400

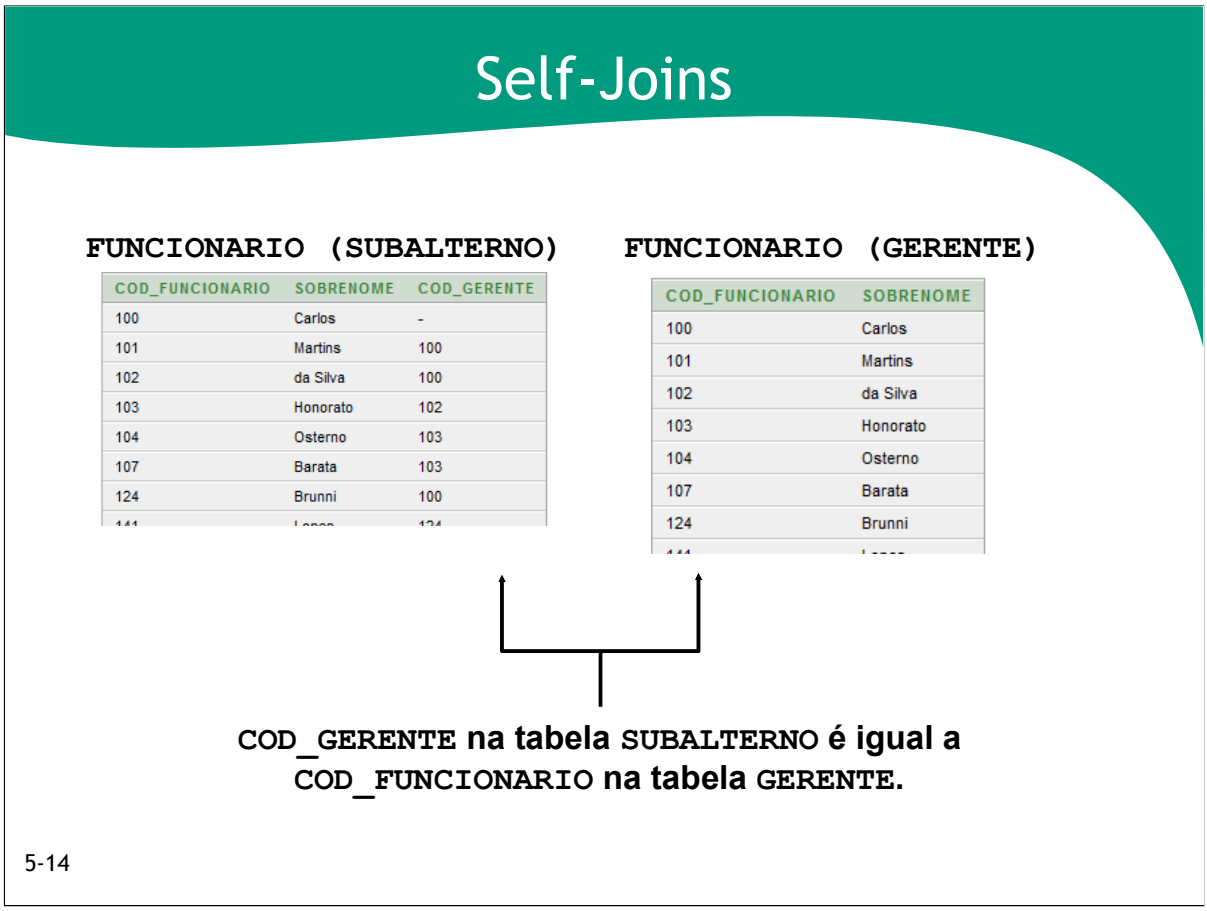
5-13

Cláusula ON

Use a cláusula ON para especificar uma condição arbitrária de Join. Assim, você pode especificar condições de join separadas de condições de filtro e pesquisa na cláusula WHERE.

No exemplo do slide, as colunas COD_DEPARTAMENTO das tabelas FUNCIONARIO e DEPARTAMENTO são unidas com a cláusula ON. Sempre que um código de departamento na tabela FUNCIONARIO for igual ao código de departamento na tabela DEPARTAMENTO, a linha será retornada.

Também é possível usar a cláusula ON para unir colunas com nomes diferentes.



Self-Joins

Para descobrir o sobrenome do gerente de cada funcionário, você precisa unir a tabela FUNCIONARIO a ela própria, isso é executar um self-join. Por exemplo, para descobrir o sobrenome do gerente do funcionário de sobrenome Brunni, você precisa:

- Localizar o sobrenome Brunni na tabela FUNCIONARIO examinando a coluna SOBRENOME.
- Localizar o número do gerente de Brunni examinando a coluna COD_GERENTE. O número do gerente de Brunni é 100.
- Localizar o sobrenome do gerente com o valor de COD_FUNCIONARIO 100 examinando a coluna SOBRENOME. O número de funcionário de Carlos é 100, portanto, Carlos é o sobrenome do gerente de Brunni.

Nesse processo, você examina a tabela duas vezes. Na primeira vez, você examina a tabela para localizar Brunni na coluna SOBRENOME e o valor 100 relativo a COD_GERENTE. Na segunda vez, você examina a coluna COD_FUNCIONARIO para localizar 100 e a coluna SOBRENOME para localizar Carlos.

Self-Joins Com a Cláusula ON

```
SELECT e.sobrenome funcionario, m.sobrenome gerente
FROM   funcionario e JOIN funcionario m
ON     (e.cod_gerente = m.cod_funcionario);
```

FUNCIONARIO	GERENTE
Martins	Carlos
da Silva	Carlos
Brunni	Carlos
Schultz	Carlos
Thacker	Carlos
Trajano	Martins
Almeida	Martins
Honorato	da Silva
Osterno	Honorato
Barata	Honorato
Lopes	Brunni
Cabral	Brunni
Miranda	Brunni
Chaves	Brunni
Sue	Schultz
Voorhees	Schultz
Yamamura	Schultz
Kramer	Thacker
Nascimento	Almeida

5-15

Self-Joins Com a Cláusula ON

Você pode usar a cláusula ON para unir colunas com nomes distintos na mesma tabela ou em uma tabela diferente.

O exemplo mostrado é um self-join da tabela FUNCIONARIO, com base nas colunas COD_FUNCIONARIO e COD_GERENTE.

Outras Condições em um Join

```
SELECT e.cod_funcionario, e.sobrenome, e.cod_departamento,
       d.cod_departamento, d.cod_localidade
FROM   funcionario e JOIN departamento d
ON     (e.cod_departamento = d.cod_departamento)
AND    e.cod_gerente = 149 ;
```

COD_FUNCIONARIO	SOBRENOME	COD_DEPARTAMENTO	COD_DEPARTAMENTO	COD_LOCALIDADE
174	Sue	80	80	2500
176	Voorhees	80	80	2500

5-16

Outras Condições em uma Join

Você pode aplicar outras condições a um comando com Join.

O exemplo do slide executa uma operação de Join nas tabelas FUNCIONARIO e DEPARTAMENTO, mas só exibe os funcionários cujo gerente tem o código de número 149. Para adicionar outras condições à cláusula ON, especifique cláusulas AND. Como opção, você pode usar uma cláusula WHERE para aplicar outras condições:

```
SELECT e.cod_funcionario, e.sobrenome,
e.cod_departamento,
       d.cod_departamento, d.cod_localidade
FROM   funcionario e JOIN departamento d
ON     (e.cod_departamento = d.cod_departamento)
WHERE  e.cod_gerente = 149;
```

Joins Tridimensionais

```
SELECT cod_funcionario, cidade, nome_departamento
FROM   funcionario e
JOIN   departamento d
ON     d.cod_departamento = e.cod_departamento
JOIN   localidade l
ON     d.cod_localidade = l.cod_localidade;
```

COD_FUNCIONARIO	CIDADE	NOME_DEPARTAMENTO
103	Recife	Informática
104	Recife	Informática
107	Recife	Informática
124	São Paulo	Logística
141	São Paulo	Logística
142	São Paulo	Logística
143	São Paulo	Logística
144	São Paulo	Logística
200	Porto Alegre	Administração

5-17

Joins Tridimensionais

Um join tridimensional é uma união de três tabelas. Na sintaxe compatível com o padrão SQL:1999, os joins são executados da esquerda para a direita. Portanto, o primeiro join a ser executado é `FUNCIONARIO JOIN DEPARTAMENTO`. A primeira condição de join pode fazer referência a colunas de `FUNCIONARIO` e `DEPARTAMENTO`, mas não a colunas de `LOCALIDADE`. A segunda condição de join pode fazer referência a colunas de todas as três tabelas.

Non-Equijoins

FUNCIONARIO

SOBRENOME	SALARIO
Carlos	24000
Martins	17000
da Silva	17000
Honorato	9000
Osterno	6000
Barata	4200
Brunni	5800
Lopes	3500
Cabral	3100
Miranda	2600

FAIXA_SALARIO

COD_FAIXA	MENOR_SALARIO	MAIOR_SALARIO
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

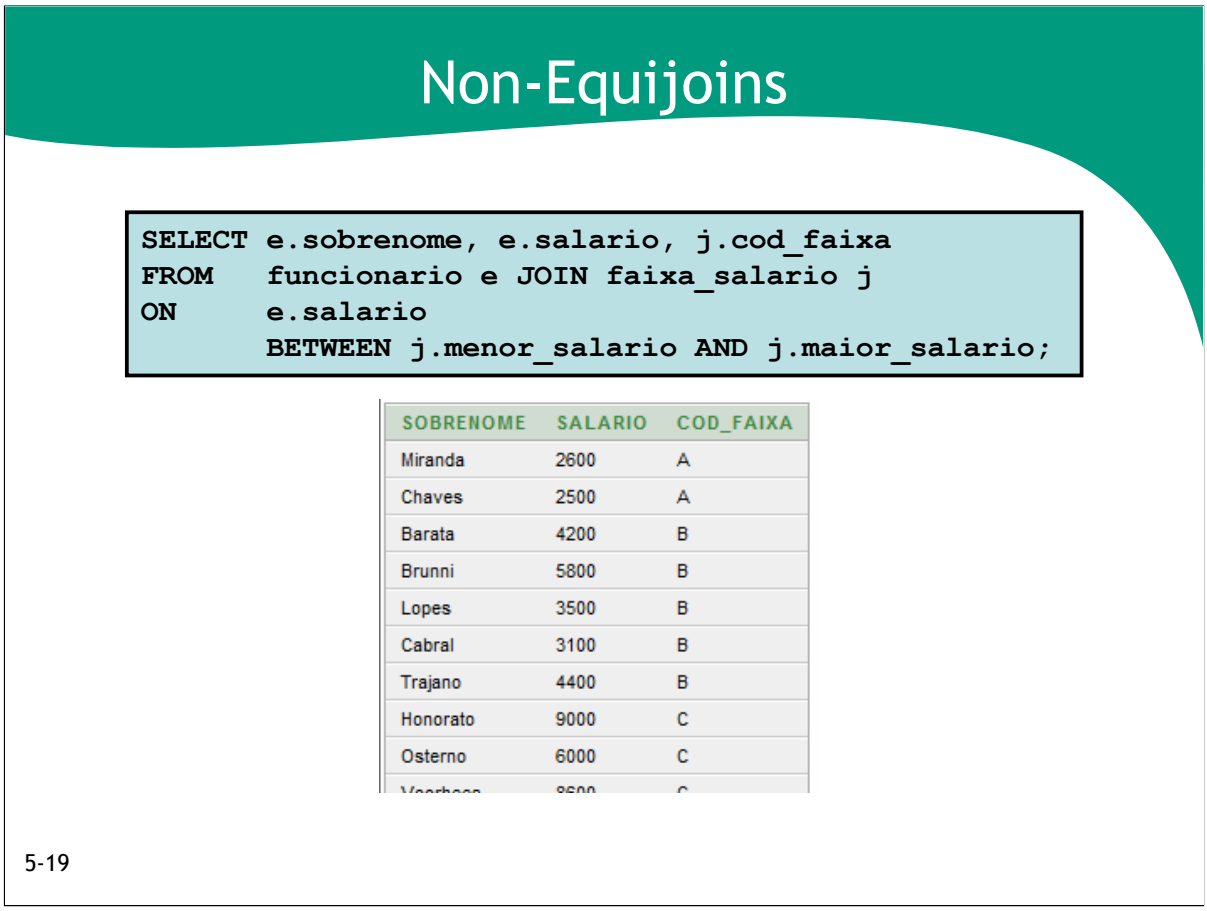
O salário na tabela FUNCIONARIO deve estar compreendido entre o menor e o maior salário na tabela FAIXA_SALARIO.

5-18

Non-Equijoins

Um non-equijoin é uma condição de join que contém algo diferente de um operador de igualdade.

O relacionamento entre as tabelas FUNCIONARIO e FAIXA_SALARIO é um exemplo de um non-equijoin. Em um relacionamento entre as duas tabelas, os valores da coluna SALARIO da tabela FUNCIONARIO devem estar compreendidos entre os valores das colunas MENOR_SALARIO e MAIOR_SALARIO da tabela FAIXA_SALARIO. O relacionamento é obtido com um operador diferente do operador de igualdade (=).



Non-Equijoins (continuação)

O exemplo do slide cria um non-equijoin para retornar o nível do salário de um funcionário. O salário deve estar compreendido entre o salário mais baixo e o salário mais alto de qualquer das faixas de salário definidas.

Observe que todos os funcionários não aparecem repetidos no resultado trazido quando essa consulta é executada. Os motivos para isso ocorrer são:

- Nenhuma das linhas da tabela de faixas de salário contém níveis sobrepostos, isto é, o valor de um determinado salário somente pode estar entre os valores de uma única faixa de salário.
- Os salários de todos os funcionários estão compreendidos entre os limites fornecidos pela tabela de níveis de cargos, ou seja, nenhum funcionário recebe menos que o valor mais baixo contido na coluna `MENOR_SALARIO` ou mais que o valor mais alto contido na coluna `MAIOR_SALARIO`.

É possível usar outras condições (como <= e >=), mas `BETWEEN` é a mais simples. Quando usar `BETWEEN`, lembre-se de informar primeiro o valor mais baixo e depois o valor mais alto.

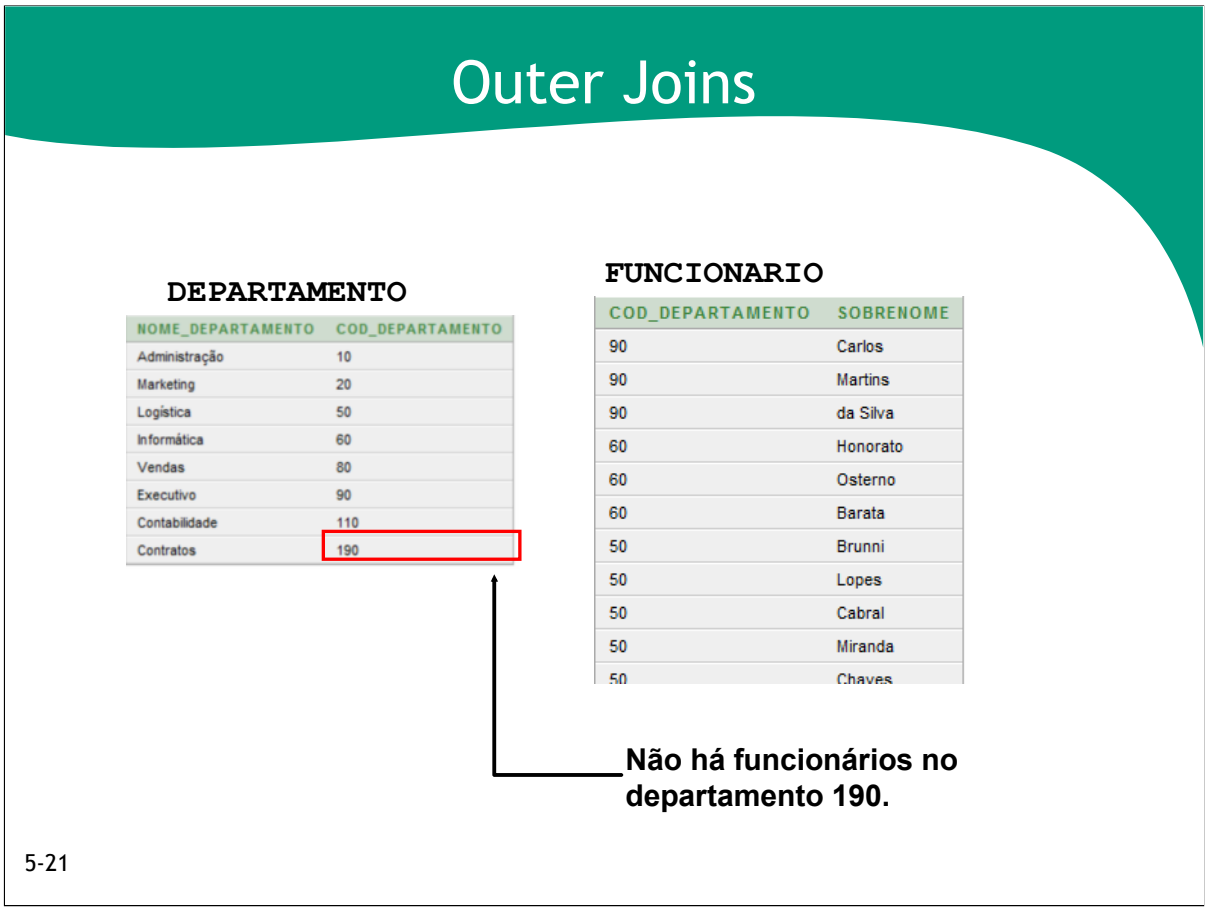
Exemplo de Join com a Sintaxe Proprietária Oracle

O exemplo do slide mostra como unir os dados das tabelas FUNCIONARIO e DEPARTAMENTO usando a sintaxe proprietária Oracle.

Caso seja necessária a inclusão de condições adicionais de filtro, basta acrescentar à cláusula WHERE uma condição AND com o filtro necessário.

Para realizar o Join com mais de duas tabelas, deve-se especificar o nome das tabelas na cláusula FROM separadas por vírgulas e as condições de Join devem ser especificadas na cláusula WHERE usando-se o operador AND:

```
SELECT e.sobrenome, d.nome_departamento, l.cidade
FROM   funcionario e, departamento d, localidade l
WHERE  e.cod_departamento = d.cod_departamento
AND    d.cod_localidade = l.cod_localidade;
```



Outer Joins

Quando uma linha não atender a uma condição de join, ela não aparecerá no resultado da consulta. Por exemplo, na condição de join das tabelas `FUNCIONARIO` e `DEPARTAMENTO`, o código do departamento 190 não é exibido, pois não existem funcionários com esse código registrado na tabela `FUNCIONARIO`. Em vez de conter 20 funcionários, o conjunto de resultados conterá 19 registros.

Porém, você pode querer retornar o registro de um departamento que não tenha funcionários, ou seja, mesmo que o departamento não possua um código correspondente na tabela de funcionários. Para trazer esse departamento na consulta, você terá que usar um Outer Join.

Outer Joins

- Left Outer Join: é um Outer Join entre duas tabelas que retorna, além dos resultados do Join normal, todas as linhas da tabela da esquerda mesmo que não possuam correspondência com a tabela da direita.
- Right Outer Join: é um Outer Join entre duas tabelas que retorna, além dos resultados do Join normal, todas as linhas da tabela da direita mesmo que não possuam correspondência com a tabela da esquerda.
- Full Outer Join: é um Outer Join entre duas tabelas que retorna, além dos resultados do Join normal, todas as linhas das tabelas da direita e esquerda mesmo que não possuam correspondência com a outra tabela.

5-22

Outer Joins

Um Outer Join retorna todas as linhas que atendem à condição de join, bem como algumas ou todas as linhas de uma tabela para as quais nenhuma linha da outra tabela atende à condição de join.

Existem três tipos de Outer Joins:

- Left Outer Join: quando além das linhas que possuem a correspondência de Join, o resultado traz todas as linhas da tabela da esquerda, mesmo que não possuam correspondência com a outra tabela.
- Right Outer Join: quando além das linhas que possuem a correspondência de Join, o resultado traz todas as linhas da tabela da direita, mesmo que não possuam correspondência com a outra tabela.
- Full Outer Join: quando além das linhas que possuem a correspondência de Join, o resultado traz todas as linhas das tabelas da direita e esquerda, mesmo que não possuam correspondência com a outra tabela.

LEFT OUTER JOIN

```
SELECT e.sobrenome, e.cod_departamento, d.nome_departamento
FROM   funcionario e LEFT OUTER JOIN departamento d
ON    (e.cod_departamento = d.cod_departamento);
```

SOBRENOME	COD_DEPARTAMENTO	NOME_DEPARTAMENTO
Carlos	90	Executivo
Martins	90	Executivo
da Silva	90	Executivo
Honorato	60	Informática
Osterno	60	Informática
Barata	60	Informática
Brunni	50	Logística
Lopes	50	Logística
Cabral	50	Logística
Miranda	50	Logística
Chaves	50	Logística
Schultz	80	Vendas
Sue	80	Vendas
Voorhees	80	Vendas
Yamamura	-	-

5-23

Left Outer Join

A consulta do slide recupera todas as linhas da tabela `FUNCIONARIO`, que é a tabela da esquerda, mesmo quando não há correspondência na tabela `DEPARTAMENTO`, que é a tabela da direita.

A consulta do slide escrita na sintaxe proprietária da Oracle fica da seguinte forma:

```
SELECT e.sobrenome, e.cod_departamento,
d.nome_departamento
FROM   funcionario e, departamento d
WHERE  e.cod_departamento = d.cod_departamento(+);
```

Observe que o símbolo de Outer Join (+) deve ser colocado na cláusula WHERE, ao lado da condição da tabela que não possui registros correspondentes, ou seja, serão retornadas todas as linhas da outra tabela.

RIGHT OUTER JOIN

```
SELECT e.sobrenome, e.cod_departamento, d.nome_departamento
FROM   funcionario e RIGHT OUTER JOIN departamento d
ON    (e.cod_departamento = d.cod_departamento);
```

SOBRENOME	COD_DEPARTAMENTO	NOME_DEPARTAMENTO
Trajano	10	Administração
Thacker	20	Marketing
Kramer	20	Marketing
Brunni	50	Logística
Lopes	50	Logística
Cabral	50	Logística
Miranda	50	Logística
Chaves	50	Logística
Honorato	60	Informática
Osterno	60	Informática
Barata	60	Informática
Schultz	80	Vendas
Sue	80	Vendas
Voorhees	80	Vendas
Carlos	90	Executivo
Martins	90	Executivo
da Silva	90	Executivo
Almeida	110	Contabilidade
Nascimento	110	Contabilidade
-	-	Contratos

5-24

Right Outer Join

A consulta do slide recupera todas as linhas da tabela DEPARTAMENTO, que é a tabela da direita, mesmo quando não há correspondência na tabela FUNCIONARIO, que é a tabela da esquerda.

A consulta do slide escrita na sintaxe proprietária da Oracle fica da seguinte forma:

```
SELECT e.sobrenome, e.cod_departamento,
d.nome_departamento
FROM   funcionario e, departamento d
WHERE  e.cod_departamento(+) = d.cod_departamento;
```

FULL OUTER JOIN

```
SELECT e.sobrenome, d.cod_departamento, d.nome_departamento
FROM   funcionario e FULL OUTER JOIN departamento d
ON    (e.cod_departamento = d.cod_departamento) ;
```

SOBRENOME	COD_DEPARTAMENTO	NOME_DEPARTAMENTO
Carlos	90	Executivo
Martins	90	Executivo
da Silva	90	Executivo
Honorato	60	Informática
Osterno	60	Informática
Barata	60	Informática
Brunni	50	Logística
Lopes	50	Logística
Cabral	50	Logística
Miranda	50	Logística
Chaves	50	Logística
Schultz	80	Vendas
Sue	80	Vendas
Voorhees	80	Vendas
Yamamura	-	-
Trajano	10	Administração
Thacker	20	Marketing
Kramer	20	Marketing
Almeida	110	Contabilidade
Nascimento	110	Contabilidade
-	190	Contratos

5-25

Full Outer Join

A consulta do slide recupera todas as linhas da tabela `FUNCIONARIO`, mesmo quando não há correspondência na tabela `DEPARTAMENTO`. A consulta também recupera todas as linhas da tabela `DEPARTAMENTO`, mesmo quando não há correspondência na tabela `FUNCIONARIO`.

Não existe correspondente ao `FULL OUTER JOIN` na sintaxe proprietária da Oracle. A única maneira de se fazer um Full Outer Join usando a sintaxe proprietária da Oracle é criar uma consulta com um Right Outer Join e unir o resultado a uma outra consulta com um Left Outer Join (o operador de união de consultas será visto em um capítulo posterior).

Produtos Cartesianos

- Um produto cartesiano ocorrerá quando todas as linhas da primeira tabela se uniram a todas as linhas da segunda tabela.
- Um produto cartesiano será formado quando:
  - Uma condição de join for omitida
  - Uma condição de join for inválida
- Para evitar um produto cartesiano, inclua sempre uma condição de join válida.

5-26

Produtos Cartesianos

Quando uma condição de join é inválida ou completamente omitida, o resultado é um *produto cartesiano*, no qual todas as combinações de linhas são exibidas. Todas as linhas da primeira tabela são unidas a todas as linhas da segunda tabela.

Um produto cartesiano tende a gerar um grande número de linhas e o resultado raramente é útil. Inclua sempre uma condição de join válida, a menos que exista uma necessidade específica de combinar todas as linhas de todas as tabelas.

Os produtos cartesianos são úteis em alguns testes quando é necessário gerar muitas linhas para simular um volume razoável de dados.

Produtos Cartesianos (continuação)

O exemplo do slide exibe o sobrenome e o nome do departamento dos funcionários com base nas tabelas `FUNCIONARIO` e `DEPARTAMENTO`. Como não foi especificada uma condição de join, todas as 20 linhas da tabela `FUNCIONARIO` são unidas a todas as 6 linhas da tabela `DEPARTAMENTO`, gerando 160 linhas na saída.

Cross Joins

- A cláusula `CROSS JOIN` gera o produto cartesiano de duas tabelas.

```
SELECT sobrenome, nome_departamento
FROM   funcionario
CROSS JOIN departamento;
```

160 registros

SOBRENOME	NOME_DEPARTAMENTO
Almeida	Administração
Barata	Administração
Brunni	Administração
Cabral	Administração
Carlos	Administração
Chaves	Administração
Honorato	Administração
Kramer	Administração

5-28

Cross Joins

Para criar intencionalmente um produto cartesiano entre duas tabelas você pode usar a cláusula `CROSS JOIN`. No exemplo do slide a cláusula `CROSS JOIN` resultará em um produto cartesiano das tabelas `FUNCIONARIO` e `DEPARTAMENTO`

Para criar um produto cartesiano usando a sintaxe de Join proprietária da Oracle você deve especificar o join sem as condições de junção das tabelas. O exemplo do slide ficaria:

```
SELECT sobrenome, nome_departamento
FROM   funcionario, departamento;
```

Exercício 5

5-29

Exercício 5

1. Crie uma consulta para gerar os endereços de todos os departamentos. Use as tabelas `LOCALIDADE` e `PAIS`. Mostre o código da localidade, o endereço, a cidade, o estado e o país na saída. Use `NATURAL JOIN` para gerar os resultados.

COD_LOCALIDADE	ENDERECO	CIDADE	ESTADO	COD_PAIS
1400	Rua da Aurora 174	Recife	PE	BR
1500	Rua Haddock Lobo 1048	São Paulo	SP	BR
1700	Rua Duque de Caxias 1187	Porto Alegre	RS	BR
1800	460 Bloor St. W.	Toronto	Ontário	CA
2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	UK

2. Gere um relatório de todos os funcionários. Crie uma consulta para exibir o sobrenome, o código do departamento e o nome do departamento de todos os funcionários.

SOBRENOME	COD_DEPARTAMENTO	NOME_DEPARTAMENTO
Trajano	10	Administração
Thacker	20	Marketing
Kramer	20	Marketing
Brunni	50	Logística
Lopes	50	Logística
Cabral	50	Logística
Miranda	50	Logística
Chaves	50	Logística
Honorato	60	Informática
Osterno	60	Informática
Barata	60	Informática
Schultz	80	Vendas
Sue	80	Vendas
Voorhees	80	Vendas
Carlos	90	Executivo
Martins	90	Executivo
da Silva	90	Executivo
Almeida	110	Contabilidade
Nascimento	110	Contabilidade

Exercício 5 (continuação)

3. Exiba o sobrenome, o código do cargo, o código do departamento e o nome do departamento de todos os funcionários que trabalham em Toronto.

SOBRENOME	COD_CARGO	COD_DEPARTAMENTO	NOME_DEPARTAMENTO
Thacker	MK_GER	20	Marketing
Kramer	MK_ANA	20	Marketing

4. Crie um relatório para exibir o sobrenome e o código dos funcionários, bem como o sobrenome e o código dos respectivos gerentes. Atribua às colunas os nome: `Funcionario`, `Func#`, `Gerente MGer#`, respectivamente. Salve o comando SQL no arquivo de texto `cap_05_04.sql`.

Funcionario	Func#	Gerente	Ger#
Martins	101	Carlos	100
da Silva	102	Carlos	100
Brunni	124	Carlos	100
Schultz	149	Carlos	100
Thacker	201	Carlos	100
Trajano	200	Martins	101
Almeida	205	Martins	101
Honorato	103	da Silva	102
Osterno	104	Honorato	103
Barata	107	Honorato	103
Lopes	141	Brunni	124
Cabral	142	Brunni	124
Miranda	143	Brunni	124
Chaves	144	Brunni	124
Sue	174	Schultz	149
Voorhees	176	Schultz	149
Yamamura	178	Schultz	149
Kramer	202	Thacker	201
Nascimento	206	Almeida	205

Exercício 5 (continuação)

5. Modifique `cap_05_04.sql` para exibir todos os sobrenomes dos funcionários, inclusive Carlos, que não possui gerente. Ordene os resultados pelo código do funcionário. Salve o comando SQL no arquivo de texto `cap_05_05.sql` e o execute.

Funcionario	Func#	Gerente	Ger#
Carlos	100	-	-
Martins	101	Carlos	100
da Silva	102	Carlos	100
Honorato	103	da Silva	102
Osterno	104	Honorato	103
Barata	107	Honorato	103
Brunni	124	Carlos	100
Lopes	141	Brunni	124
Cabral	142	Brunni	124

6. Crie um relatório para exibir os sobrenomes e os códigos de departamento dos funcionários, bem como todos os funcionários que trabalham no mesmo departamento . Atribua um nome apropriado a cada coluna. Salve o script no arquivo `cap_05_06.sql`.

DEPARTAMENTO	FUNCIONARIO	COLEGA
20	Kramer	Thacker
20	Thacker	Kramer
50	Brunni	Cabral
50	Brunni	Chaves
50	Brunni	Lopes
50	Brunni	Miranda
50	Cabral	Brunni
50	Cabral	Chaves
50	Cabral	Lopes
50	Cabral	Miranda
50	Chaves	Brunni
50	Chaves	Cabral

42 linhas retornadas em 0,00 segundos

Exercício 5 (continuação)

7. Crie um relatório sobre faixas de cargos e salários. Para se familiarizar com a tabela `FAIXA_SALARIO`, primeiro mostre a estrutura dessa tabela. Em seguida, crie uma consulta que exiba o sobrenome, o código do cargo, o nome do departamento, o salário e a faixa de salário de todos os funcionários.

Tipo de Objeto **TABLE** Objeto **FAIXA_SALARIO**

Table	Column	Tipo De Dados
FAIXA_SALARIO	COD_FAIXA	Varchar2
	MENOR_SALARIO	Number
	MAIOR_SALARIO	Number

	COD_CARGO	NOME_DEPARTAMENTO	SALARIO	COD_FAIXA
	ES_AUX	Logística	2600	A
	ES_AUX	Logística	2500	A
	IT_PROG	Informática	4200	B
Brunni	ES_GER	Logística	5800	B
Lopes	ES_AUX	Logística	3500	B
Cabral	ES_AUX	Logística	3100	B
Trajano	AD_ASST	Administração	4400	B
Honorato	IT_PROG	Informática	9000	C
Osterno	IT_PROG	Informática	6000	C
Voorhees	VE_REP	Vendas	8600	C
Kramer	MK_ANA	Marketing	6000	C
Nascimento	CTPUB_GER	Contabilidade	8300	C
Schultz	VE_GER	Vendas	10500	D
Sue	VE_REP	Vendas	11000	D
Thacker	MK_GER	Marketing	13000	D
Almeida	CT_GER	Contabilidade	12000	D
Carlos	AD_PRES	Executivo	24000	E
Martins	AD_VP	Executivo	17000	E
da Silva	AD_VP	Executivo	17000	E

19 linhas retornadas em 0.00 segundos Exportação para CSV

8. Crie uma consulta para exibir o sobrenome e a data de admissão de todos os funcionários admitidos após o funcionário de sobrenome Schultz.

SOBRENOME	DATA_ADMISSAO
da Silva	13/01/03
Osterno	21/05/01
Barata	07/02/09
Brunni	16/11/08
Lopes	17/10/05
Cabral	29/01/07
Miranda	15/03/08
Chaves	09/07/08
Sue	11/05/06
Thacker	17/02/06
Kramer	17/08/07
Almeida	07/06/04
Nascimento	07/06/04

13 linhas retornadas em 0,02 segundos

Exercício 5 (continuação)

9. Obtenha os sobrenomes e as datas de admissão de todos os funcionários admitidos antes dos seus respectivos gerentes, além dos sobrenomes e das datas de admissão desses gerentes. Salve o script no arquivo `cap5_09.sql`.

SOBRENOME	DATA_ADMISSAO	SOBRENOME	DATA_ADMISSAO
Trajano	17/09/97	Martins	21/09/99
Honorato	03/01/00	da Silva	13/01/03
Lopes	17/10/05	Brunni	16/11/08
Cabral	29/01/07	Brunni	16/11/08
Miranda	15/03/08	Brunni	16/11/08
Chaves	09/07/08	Brunni	16/11/08
Voorhees	24/03/98	Schultz	29/01/00
Yamamura	24/05/99	Schultz	29/01/00

8 linhas retornadas em 0,05 segundos Exportação para CSV