15

Recuperação Hierárquica

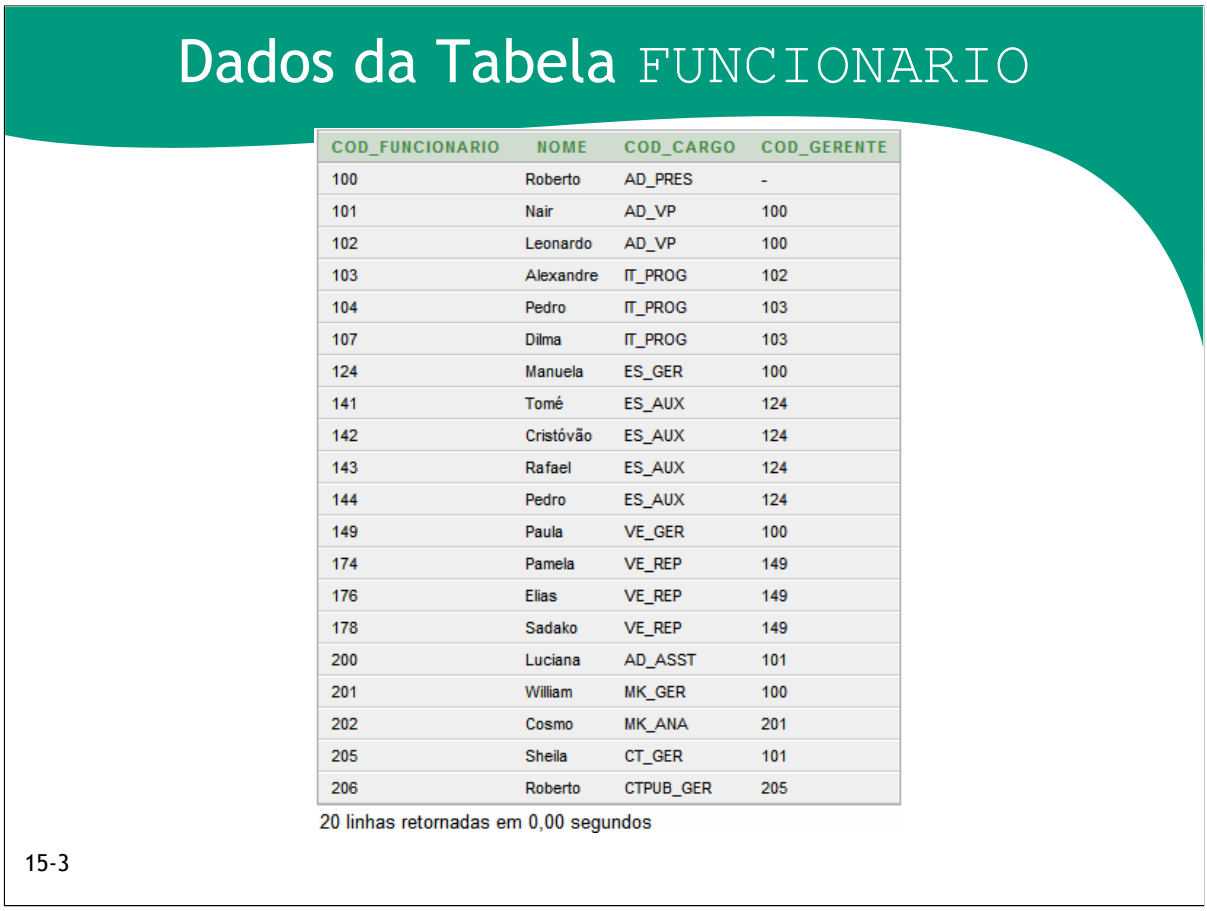
Objetivos deste Capítulo

- Ao concluir este capítulo, você poderá:
  - Interpretar o conceito de uma consulta hierárquica
  - Criar um relatório estruturado em árvore
  - Formatar dados hierárquicos
  - Excluir ramificações da estrutura em árvore

15-2

Objetivos deste Capítulo

Neste capítulo, você aprenderá a usar consultas hierárquicas para criar relatórios estruturados em árvore.

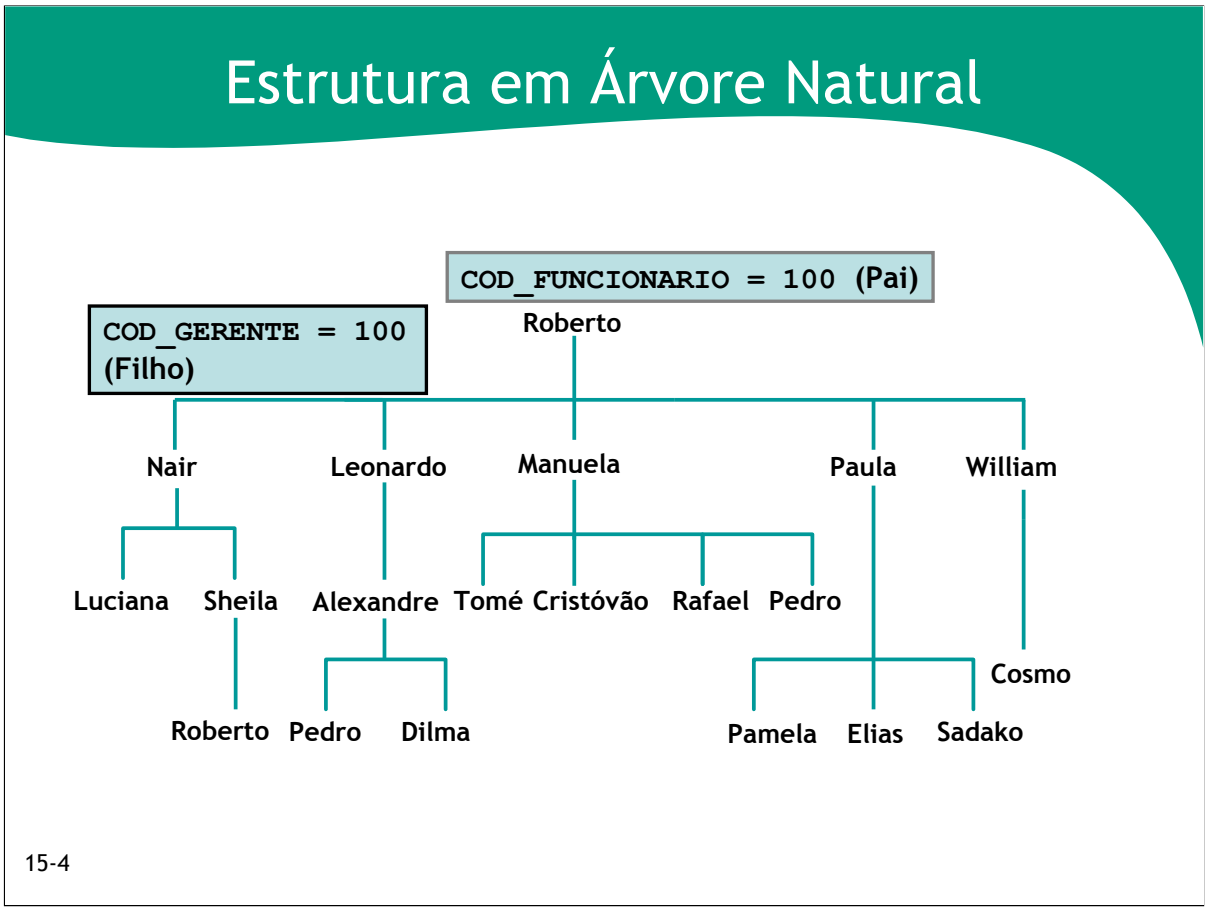


Dados da Tabela `FUNCIONARIO`

Ao usar consultas hierárquicas, você poderá recuperar dados com base em um relacionamento hierárquico natural entre as linhas de uma tabela. Um banco de dados relacional não armazena registros de forma hierárquica. No entanto, quando existe um relacionamento hierárquico entre as linhas de uma tabela, um processo denominado percurso da árvore permite que a hierarquia seja construída. Uma consulta hierárquica é um método de organizar os galhos de uma árvore em uma ordem específica.

Imagine uma árvore genealógica onde os membros mais velhos ficam na base da árvore, e os mais novos representam os galhos. Os galhos podem ter suas próprias ramificações e assim por diante.

Pode ser feita uma consulta hierárquica quando existe um relacionamento entre as linhas de uma tabela. Por exemplo, no slide, é possível verificar que os funcionários com códigos de cargos `AD_VP`, `ES_GER`, `VE_GER` e `MK_GER` estão diretamente subordinados ao presidente da empresa. Sabemos disso porque a coluna `COD_GERENTE` desses registros contém um código de funcionário igual a 100, que pertence ao presidente (`AD_PRES`).



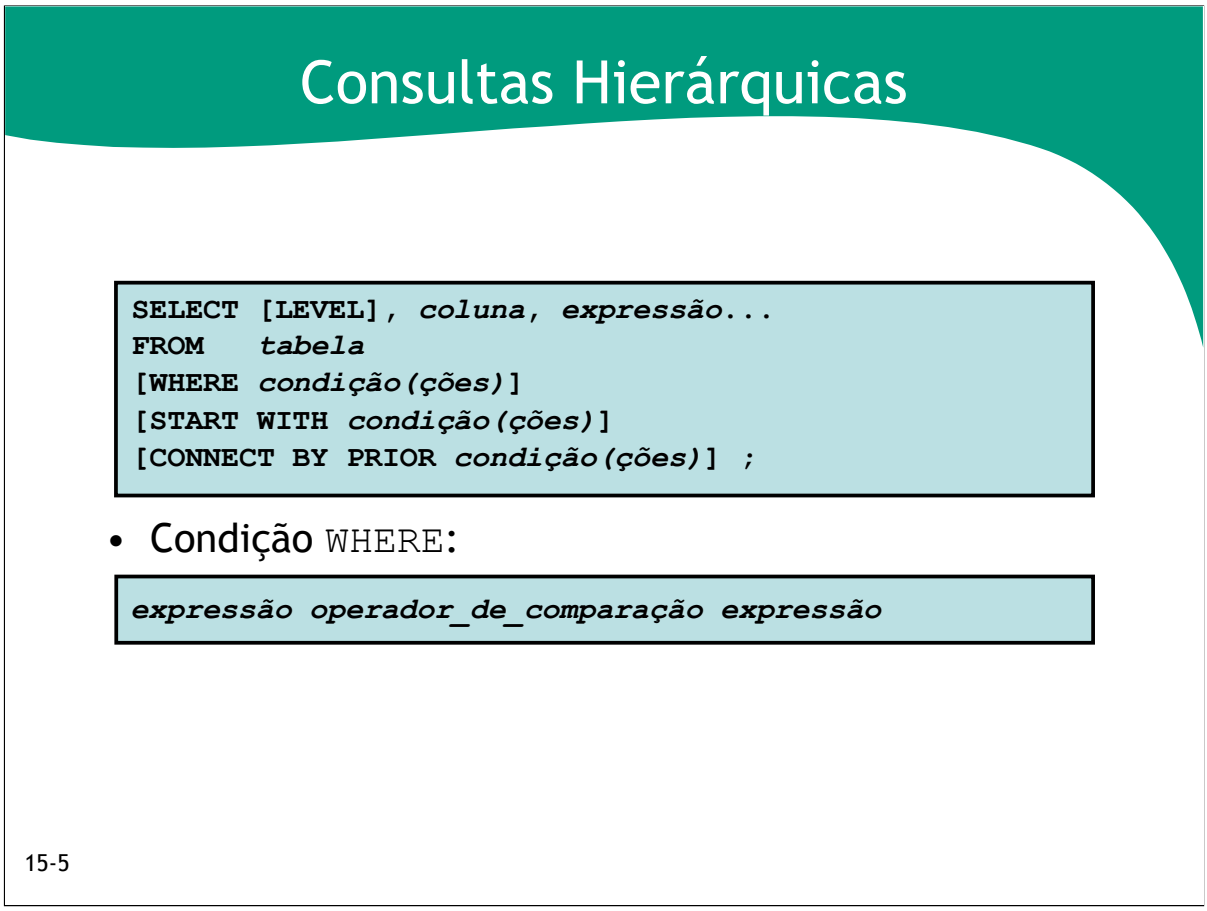
Estrutura em Árvore Natural

A tabela `FUNCIONARIO` tem uma estrutura em árvore que representa a estrutura organizacional da empresa. É possível criar a hierarquia examinando o relacionamento entre os valores equivalentes nas colunas `COD_FUNCIONARIO` e `COD_GERENTE`. Para explorar esse relacionamento, é possível unir a tabela a ela mesma. A coluna `COD_GERENTE` contém o número de funcionário do gerente do funcionário.

O relacionamento pai/filho de uma estrutura em árvore permite controlar:

- A direção em que a hierarquia é percorrida
- A posição inicial na hierarquia

O slide exibe uma estrutura em árvore invertida da hierarquia de gerenciamento dos funcionários na tabela `FUNCIONARIO`.

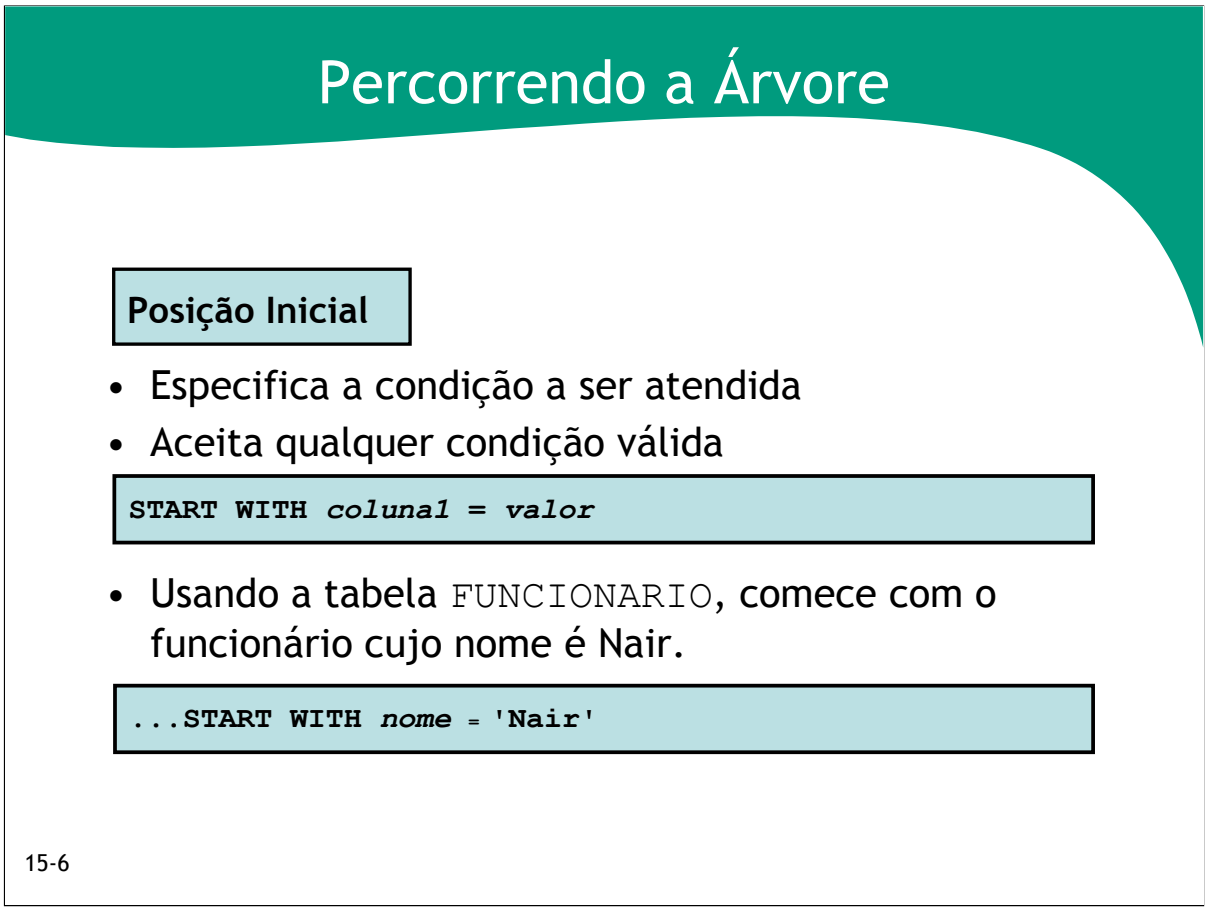


Consultas Hierárquicas

Na sintaxe:

`LEVEL`	Para cada linha retornada por uma consulta hierárquica, a pseudocoluna `LEVEL` retorna 1 para uma linha-raiz, 2 para uma linha filho da raiz, e assim por diante.
`FROM` *`tabela`*	Especifica a tabela, a visão ou o visão materializada que contém as colunas. Você só pode fazer seleções em uma tabela.
`WHERE`	Restringe as linhas retornadas pela consulta sem afetar outras linhas da hierarquia
`condition`	É uma comparação com expressões
`START WITH`	Especifica as linhas-raiz da hierarquia (onde começar). Esta cláusula é necessária para uma consulta hierárquica verdadeira.
`CONNECT BY PRIOR`	especifica as colunas nas quais existe o relacionamento entre linhas pais e filhos. Esta cláusula é necessária para uma consulta hierárquica.

O comando `SELECT` não pode conter uma join ou uma consulta de uma visão que contém uma join.



Percorrendo a Árvore

As linhas a serem usadas como raiz da árvore são determinadas pela cláusula `START WITH`. É possível usar a cláusula `START WITH` com qualquer condição válida.

Exemplos

Usando a tabela `FUNCIONARIO`, comece com Roberto, o presidente da empresa.

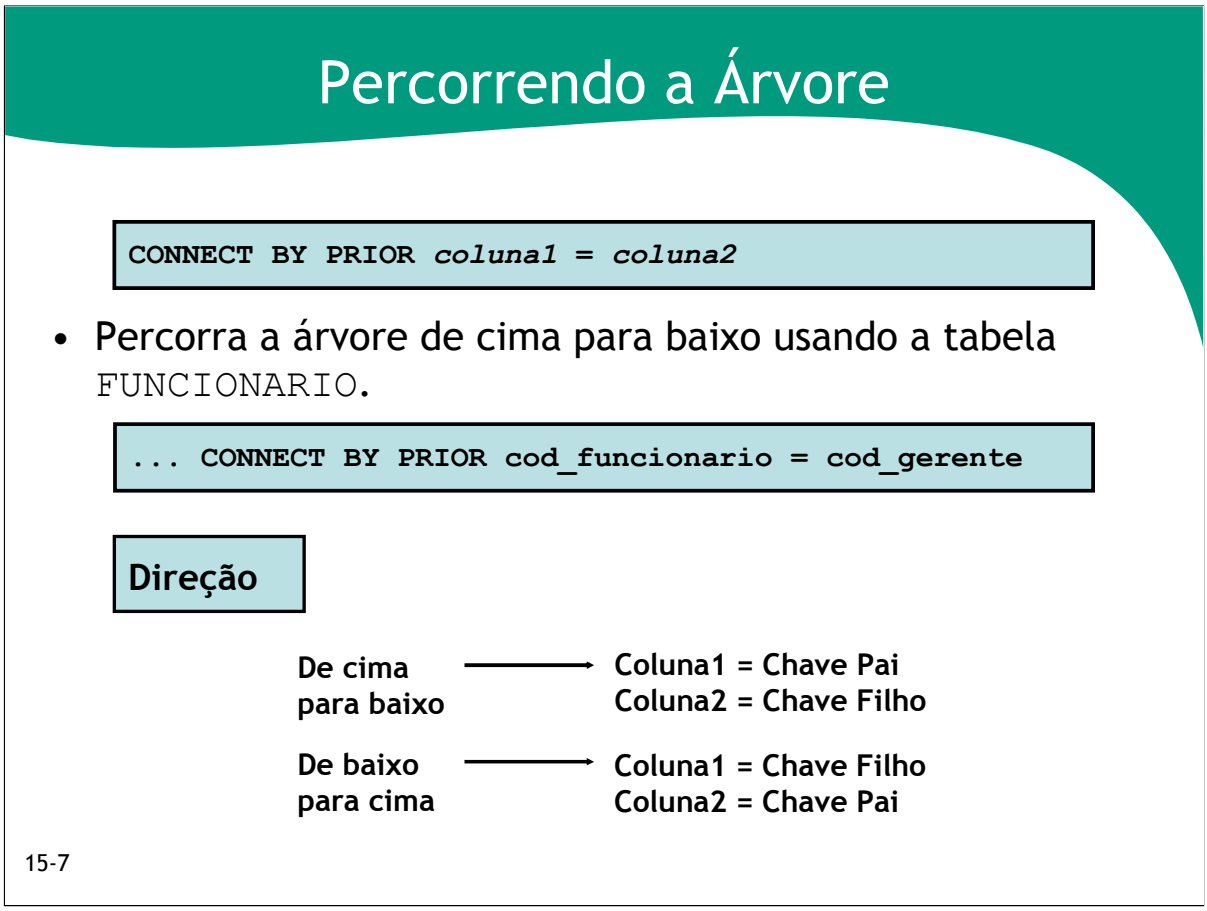
```
... START WITH cod_gerente IS NULL
```

Usando a tabela `FUNCIONARIO`, comece com o funcionário Nair. Uma condição `START WITH` pode conter uma subconsulta.

```
... START WITH cod_funcionario = (SELECT cod_funcionario
                                  FROM   funcionario
                                  WHERE  sobrenome = 'Nair')
```

Se a cláusula `START WITH` for omitida, o percurso da árvore começará com todas as linhas da tabela como linhas-raiz. Se for usada uma cláusula `WHERE`, o percurso começará com todas as linhas que atenderem à condição `WHERE`. Esse percurso não refletirá mais a verdadeira hierarquia.

As cláusulas `CONNECT BY PRIOR` e `START WITH` não seguem o padrão ANSI SQL.



Percorrendo a Árvore (continuação)

A direção da consulta, de pai para filho ou de filho para pai, é determinada pela posição da coluna `CONNECT BY PRIOR`. O operador `PRIOR` faz referência à linha pai. Para localizar as linhas filhas de uma linha pai, o Oracle avalia a expressão `PRIOR` da linha pai e as outras expressões de cada linha da tabela. As linhas para as quais a condição é verdadeira são as linhas filhas da linha pai. O Oracle sempre seleciona as linhas filhas avaliando a condição `CONNECT BY` relativa a uma linha pai atual.

Percorra a árvore de cima para baixo usando a tabela `FUNCIONARIO`. Defina um relacionamento hierárquico no qual o valor de `COD_FUNCIONARIO` da linha pai seja igual ao valor de `COD_GERENTE` da linha filho.

```
... CONNECT BY PRIOR cod_funcionario = cod_gerente
```

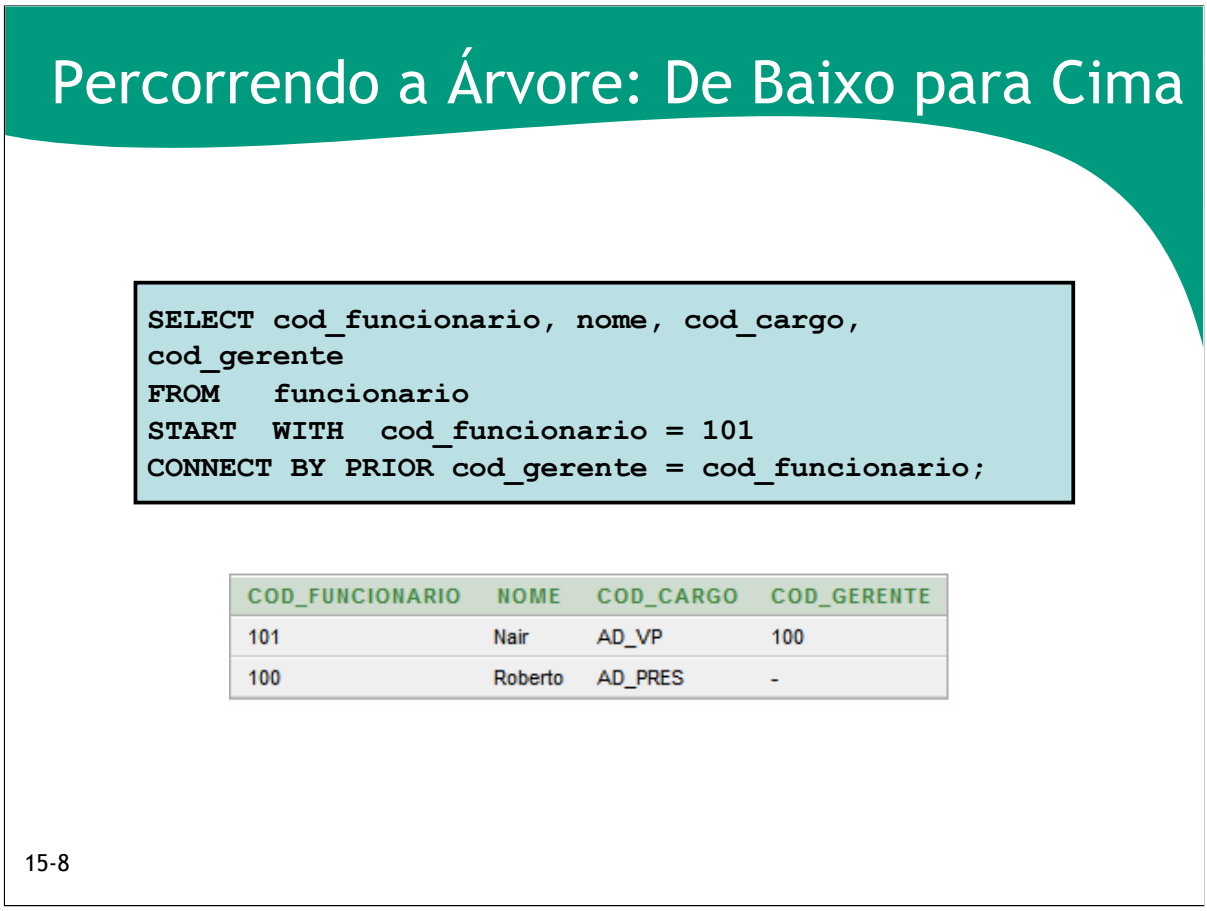
Percorra a árvore de baixo para cima usando a tabela `FUNCIONARIO`.

```
... CONNECT BY PRIOR cod_gerente = cod_funcionario
```

O operador `PRIOR` não precisa ser necessariamente codificado logo após `CONNECT BY`. Portanto, a cláusula `CONNECT BY PRIOR` abaixo fornece o mesmo resultado que o exemplo anterior.

```
... CONNECT BY cod_funcionario = PRIOR cod_gerente
```

Observação: A cláusula `CONNECT BY` não pode conter uma subconsulta.



Percorrendo a Árvore: De Baixo para Cima

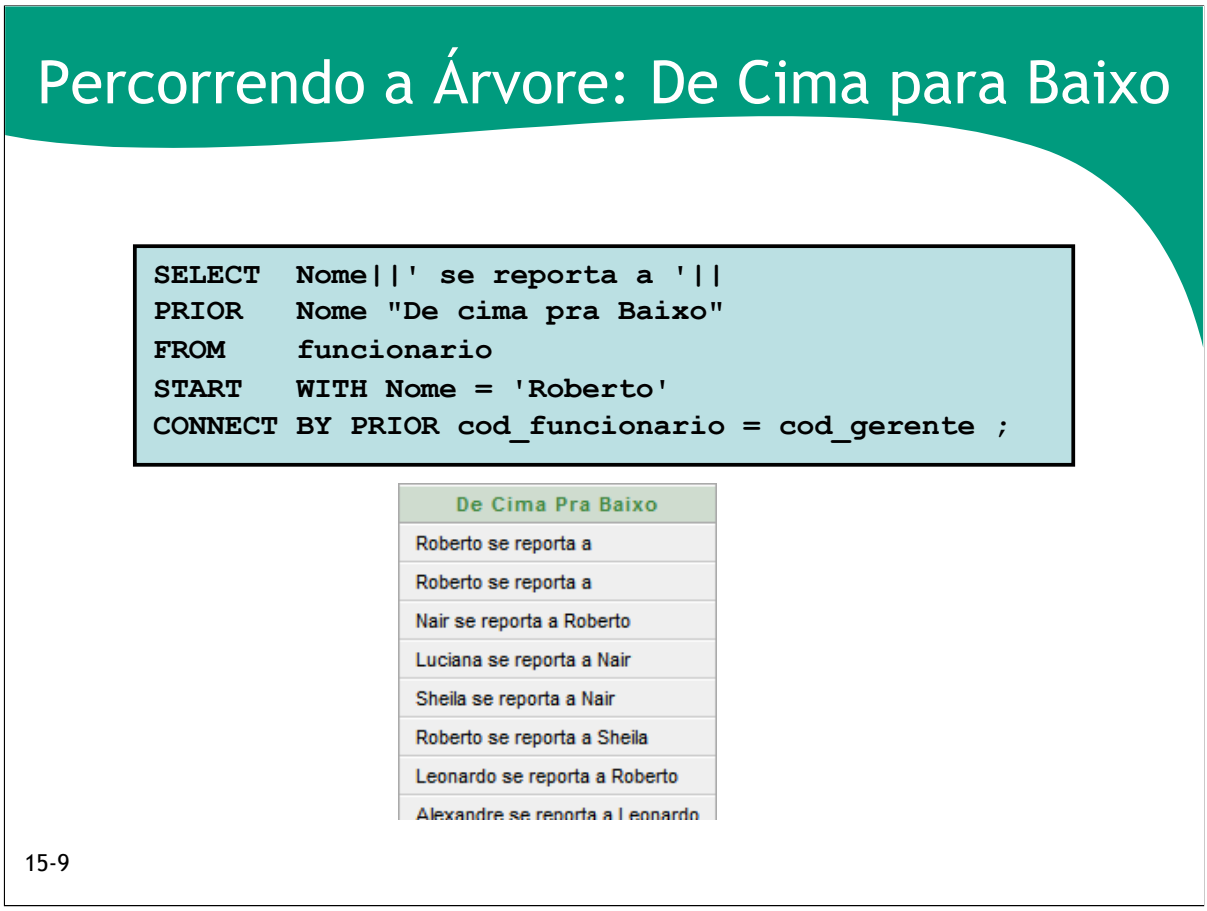
O exemplo do slide exibe uma lista de gerentes começando com o funcionário cujo código é 101.

Exemplo

No exemplo a seguir, os valores de `COD_FUNCIONARIO` são avaliados para a linha pai e os valores de `COD_GERENTE` e `SALARIO` são avaliados para as linhas filhos. O operador `PRIOR` só se aplica ao valor de `COD_FUNCIONARIO`.

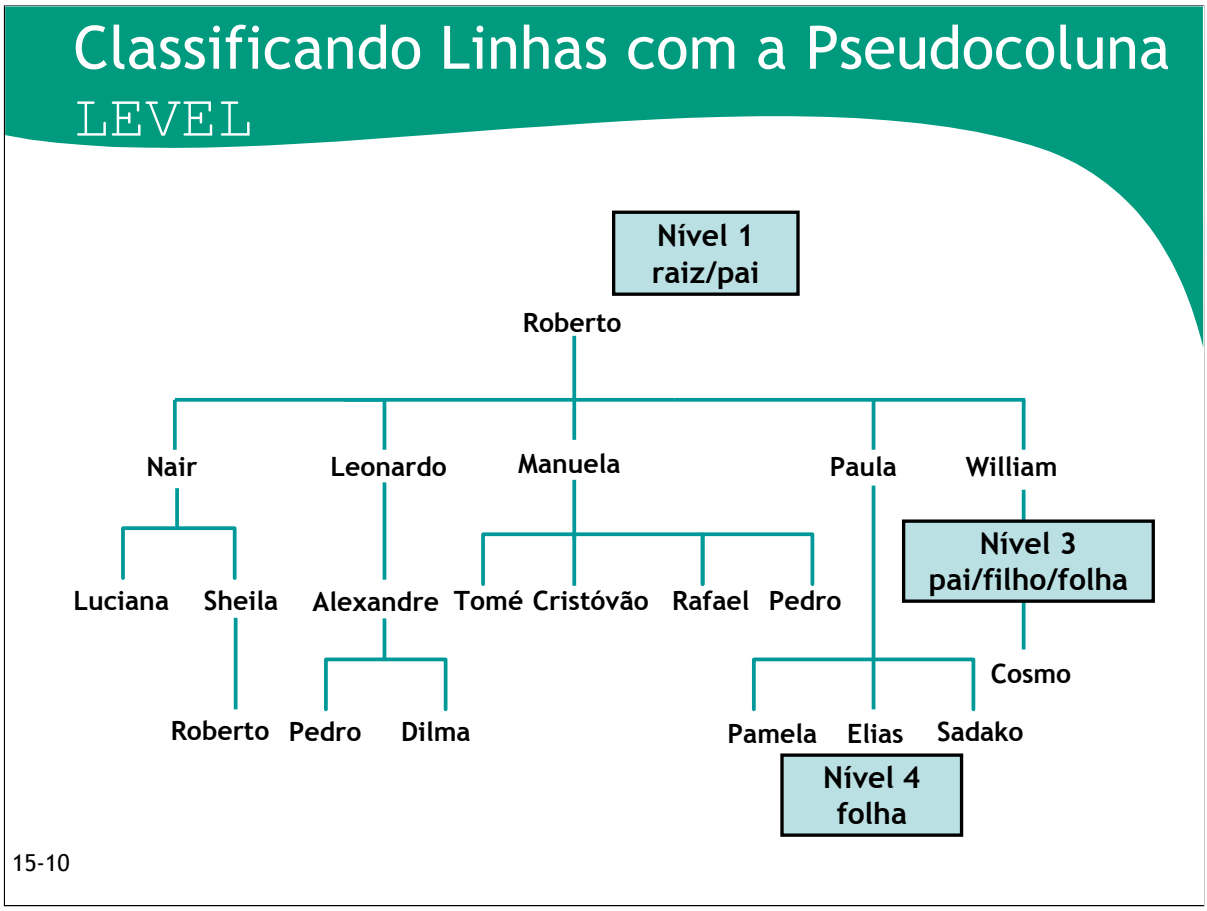
```
... CONNECT BY PRIOR cod_funcionario = cod_gerente
                     AND salario > 15000;
```

Para qualificar-se como uma linha filho, a linha deve ter um valor de `COD_GERENTE` igual ao valor de `COD_FUNCIONARIO` da linha pai e um valor de `SALARIO` maior que R$15.000.



Percorrendo a Árvore: De Cima para Baixo

Se a árvore for percorrida de cima para baixo, os nomes dos funcionários e respectivos gerentes serão exibidos. Use o funcionário King como a posição inicial. Imprima apenas uma coluna.



Classificando Linhas com a Pseudocoluna `LEVEL`

Você pode mostrar explicitamente a classificação ou o nível de uma linha na hierarquia usando a pseudocoluna `LEVEL`. Assim, seu relatório será mais legível. As bifurcações onde uma ou mais ramificações separam-se de uma ramificação maior são chamadas de nós, e a extremidade de uma ramificação é chamada de folha ou nó-folha. O diagrama do slide mostra os nós da árvore invertida com os respectivos valores de `LEVEL`. Por exemplo, o funcionário William é um nó pai e um nó filho; enquanto o funcionário Elias é um nó filho e um nó-folha.

A Pseudocoluna `LEVEL`

Valor	Nível
1	Um nó-raiz
2	Um filho de um nó-raiz
3	Um filho de um filho, e assim por diante

No slide, Roberto é o nó-raiz ou pai (`LEVEL = 1`). Nair, Leonardo, Manuela, Paula, William, Sheila e Alexandre são nós filhos e também nós pais (`LEVEL = 2`). Luciana, Tomé, Cristóvão, Rafael, Pedro, Roberto, Pedro, Dilma, Pamela, Elias, Sadako e Cosmo são nós filhos e nós-folha. (`LEVEL = 3` e `LEVEL = 4`)

Observação: Um *nó-raiz* é o nó mais alto em uma árvore invertida. Um *nó filho* é qualquer nó diferente do nó-raiz. Um nó pai tem nós filhos. Um nó-folha não tem nós filhos. O número de níveis retornados por uma consulta hierárquica pode ser limitado pela memória disponível do usuário.

Formatando Relatórios Hierárquicos Usando LEVEL e LPAD

- Crie um relatório exibindo os níveis de gerenciamento da empresa, começando com o nível mais elevado e recuando cada nível subseqüente.

```
COLUMN organograma FORMAT A12
SELECT LPAD(nome||' '||sobrenome,
       LENGTH(nome ||' '||sobrenome)+(LEVEL*2)-2,'_')
       AS organograma
FROM   funcionario
START WITH sobrenome='Carlos'
CONNECT BY PRIOR cod_funcionario=cod_gerente
```

15-11

Formatando Relatórios Hierárquicos Usando a Pseudocoluna LEVEL

São designados números de nível aos nós de uma árvore a partir da raiz. Use a function LPAD com a pseudocoluna LEVEL para exibir um relatório hierárquico como uma árvore recuada.

No exemplo do slide:

- LPAD(*char1*,*n* [,*char2*]) retorna *char1*, preenchido à esquerda até o tamanho *n* com a seqüência de caracteres em *char2*. O argumento *n* é o tamanho total do valor retornado como aparece na tela do terminal.
- LPAD(sobrenome, LENGTH(sobrenome)+(LEVEL*2)-2,'_') define o formato da exibição.
- *char1* é o valor de SOBRENOME, *n* é o tamanho total do valor retornado, o tamanho de SOBRENOME +(LEVEL*2)-2 e *char2* é '_'.

Em outras palavras, o exemplo informa ao SQL para preencher à esquerda o valor de SOBRENOME com o caractere '_' até que o tamanho da string de caracteres resultante seja igual ao valor determinado por LENGTH(sobrenome)+(LEVEL*2)-2. Para Roberto Carlos, LEVEL = 1. Portanto, (2 * 1) -2 = 2 -2 = 0. Assim, Roberto Carlos não é preenchido com caracteres '_' e é exibido na coluna 1. Para Nair Martins, LEVEL = 2. Portanto, (2 * 2) -2 = 4 - 2 = 2. Assim, Nair Martins é preenchido com 2 caracteres '_' e exibido recuado. Os outros registros da tabela FUNCIONARIO são exibidos de forma semelhante.

Formatando Relatórios Hierárquicos Usando `LEVEL` (continuação)

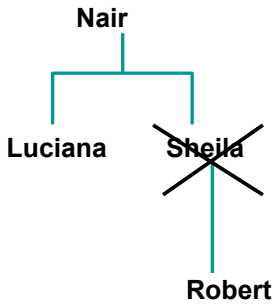
ORGANOGRAMA
Roberto Carlos
__Nair Martins
____Luciana Trajano
____Sheila Almeida
______Roberto Nascimento
__Leonardo da Silva
____Alexandre Honorato
______Pedro Osterno
______Dilma Barata
__Manuela Brunni
____Tomé Lopes
____Cristóvão Cabral
____Rafael Miranda
____Pedro Chaves
__Paula Schultz
____Pamela Sue
____Elias Voorhees
____Sadako Yamamura
__William Thacker
____Cosmo Kramer

20 linhas retornadas

Reduzindo Ramificações

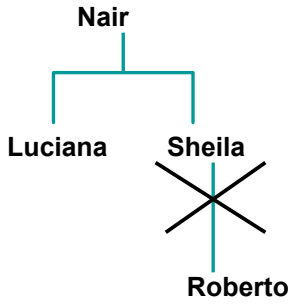
Use a cláusula WHERE para eliminar um nó.

```
WHERE nome != 'Sheila'
```



Use a cláusula CONNECT BY para eliminar uma ramificação.

```
CONNECT BY PRIOR
cod_funcionario = cod_gerente
AND nome != 'Sheila'
```



15-13

Reduzindo Ramificações

Você pode usar as cláusulas `WHERE` e `CONNECT BY` para reduzir a árvore, isto é, para controlar quais nós ou linhas serão exibidos. O predicado usado age como uma condição booleana.

Exemplos

Começando na raiz, percorra a árvore de cima para baixo e elimine o funcionário Sheila do resultado, mas processe as linhas filhos.

```
     SELECT  cod_departamento, cod_funcionario,sobrenome,
cod_cargo, salario
     FROM    funcionario
     WHERE   nome  != 'Sheila'
     START   WITH cod_gerente IS NULL
     CONNECT BY PRIOR cod_funcionario = cod_gerente;
```

Começando na raiz, percorra a árvore de cima para baixo e elimine o funcionário Sheila e todas as linhas filhos.

```
     SELECT  cod_departamento, cod_funcionario,sobrenome,
cod_cargo, salario
     FROM    funcionario
     START   WITH cod_gerente IS NULL
     CONNECT BY PRIOR cod_funcionario = cod_gerente
     AND     nome != 'Sheila';
```

Exercício 15

15-14

Exercício 15 (continuação)

1. Gere um relatório que mostre um organograma do departamento de Manuela. Imprima os sobrenomes, os salários e os códigos dos departamentos.

SOBRENOME	SALARIO	COD_DEPARTAMENTO
Brunni	5800	50
Lopes	3500	50
Cabral	3100	50
Miranda	2600	50
Chaves	2500	50

5 linhas retornadas em 0,00 segundos

2. Crie um relatório que mostre a hierarquia de gerentes para a funcionário de sobrenome Nascimento. Exiba primeiramente o seu gerente imediato.

NOME	SOBRENOME
Sheila	Almeida
Nair	Martins
Roberto	Carlos

Exercício 15 (continuação)

3. Crie um relatório recuado mostrando a hierarquia de gerenciamento, começando pelo funcionário cujo `SOBRENOME` é Martins. Imprima o sobrenome, o código do gerente e o código do departamento do funcionário. Defina apelidos para as colunas, conforme indicado no exemplo de saída.

NOME	GER	CODDEPT
Martins	100	90
__Trajano	101	10
__Almeida	101	110
____Nascimento	205	110

4 linhas retornadas em 0,00 segundos

4. Produza um organograma que mostre a hierarquia de gerenciamento da empresa. Comece pela pessoa que está no nível mais alto e exclua todas as outras que tenham um código do cargo igual a `IT_PROG`. Exclua também Leonardo e respectivos subordinados.

SOBRENOME	COD_FUNCIONARIO	COD_GERENTE
Carlos	100	-
Martins	101	100
Trajano	200	101
Almeida	205	101
Nascimento	206	205
Brunni	124	100
Lopes	141	124
Cabral	142	124
Miranda	143	124
Chaves	144	124
Schultz	149	100
Sue	174	149
Voorhees	176	149
Yamamura	178	149
Thacker	201	100
Kramer	202	201

16 linhas retornadas em 0,01 segundos