3

Funções de uma Única Linha

Objetivos deste Capítulo

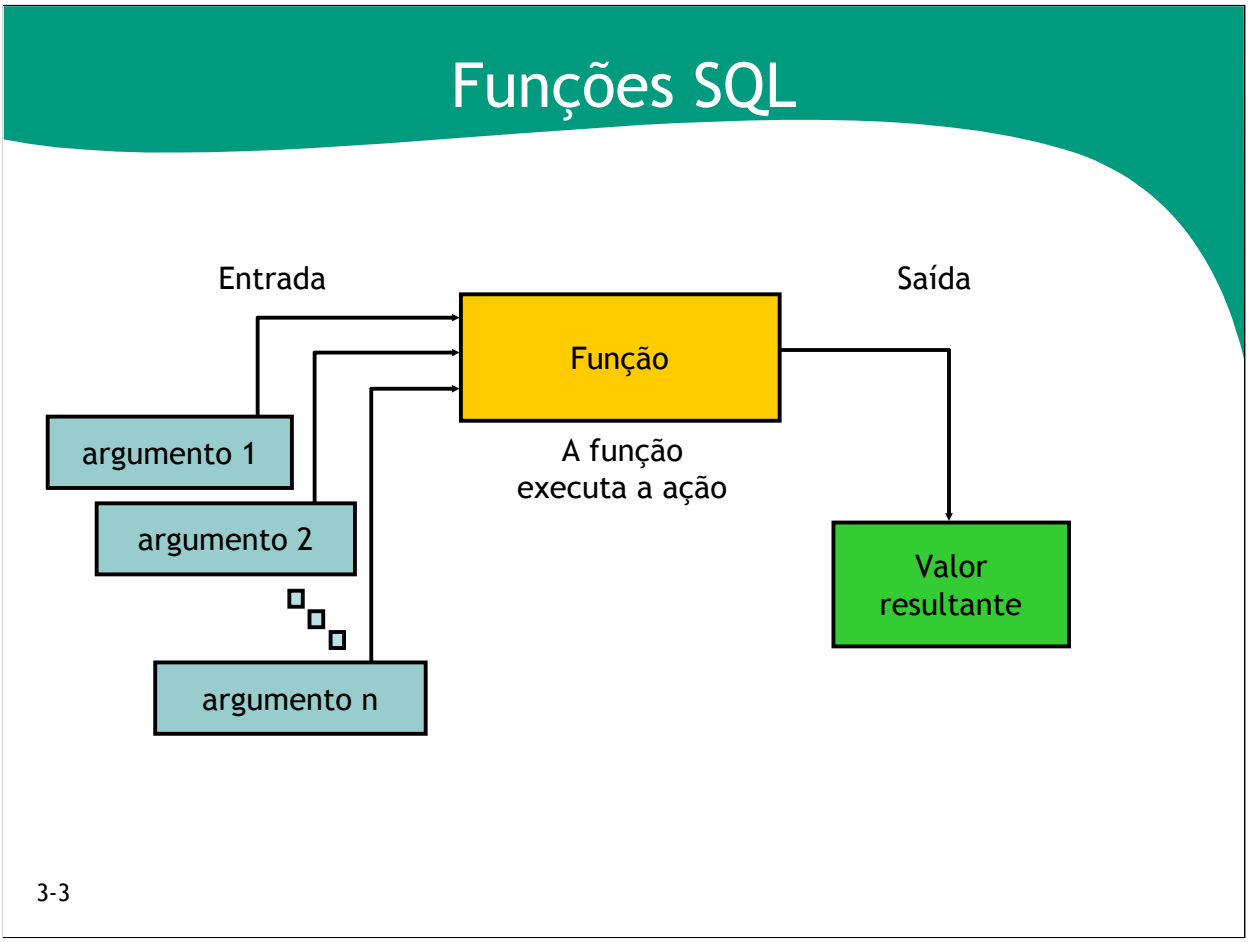
- Ao concluir este capítulo, você poderá:
  - Descrever vários tipos de funções disponíveis em SQL
  - Usar funções de caractere, número e data em instruções SELECT
  - Descrever o uso de funções de conversão

3-2

Objetivos deste Capítulo

As funções podem ser usadas para tornar os comandos SQL mais eficientes.

Este capítulo trata das funções de única linha de caractere, número e data. Ele trata também de funções de conversão de tipos de dados (por exemplo, conversão entre um tipo de dado de caractere em numéricos)



Funções SQL

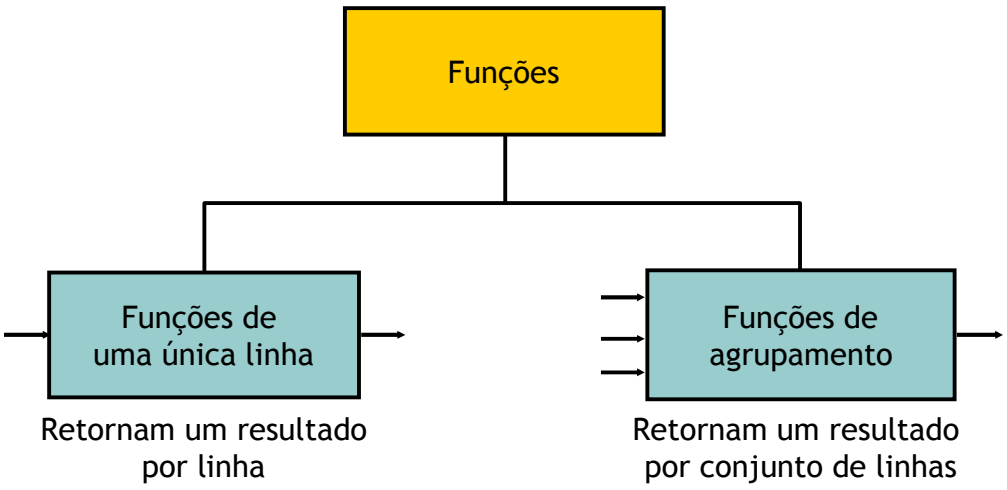
As funções são um recurso eficiente de SQL, elas aceitam zero, um ou mais argumentos de entrada e sempre retornam um parâmetro de saída.

É possível usar as funções com os seguintes objetivos:

- Executar cálculos em dados
- Modificar itens individuais de dados
- Manipular a saída de grupos de linhas
- Formatar datas e números para exibição
- Converter tipos de dados de colunas

Observação: A maioria das funções descritas neste capítulo são específicas da versão de SQL do Oracle.

Tipos de Funções SQL



3-4

Tipos de Funções SQL

Existem dois tipos de funções SQL:

- Funções de uma única linha
- Funções de agrupamento

Funções de uma Única Linha

Essas funções operam em uma linha por vez e retornam um valor de saída para cada linha de entrada. Existem tipos diferentes de funções de uma única linha. Este capítulo trata das seguintes funções:

- Caractere
- Número
- Data
- Conversão
- Geral
- Expressões Regulares

Funções de Agrupamento

As funções de agrupamento podem manipular grupos de linhas para fornecer um resultado por grupo. As funções de agrupamento serão abordadas em um capítulo posterior.

Observação: Para obter mais informações e uma lista completa das funções disponíveis e a sintaxe correspondente, consulte o manual *Oracle SQL Reference*.

Funções de uma Única Linha

- Funções de uma única linha:
  - Aceitam argumentos e retornam um valor
  - Agem sobre cada linha retornada
  - Retornam um resultado por linha
  - Podem modificar o tipo de dados
  - Podem ser aninhadas
  - Aceitam argumentos, como uma coluna ou uma expressão

```
nome_funcao [(argumento1, argumento2,...)]
```

3-5

Funções de uma Única Linha

As funções de uma única linha aceitam um ou mais argumentos e retornam um valor para cada linha retornada pela consulta. Um argumento pode ser:

- Uma constante fornecida pelo usuário
- Um valor variável
- Um nome de coluna
- Uma expressão

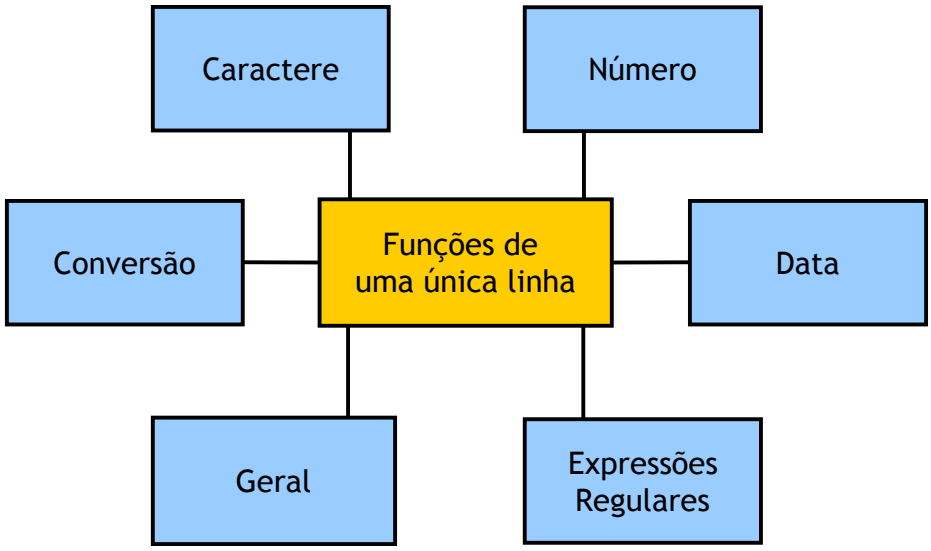
Os recursos de funções de uma única linha são:

- Agir sobre cada linha retornada pela consulta
- Retornar um resultado por linha
- Possibilitar o retorno de um valor de dados de um tipo diferente dos argumentos de entrada
- Possibilitar a entrada de um ou mais argumentos
- Ser usadas em cláusulas `SELECT`, `WHERE` e `ORDER BY`
- Ser aninhadas

Na sintaxe:

nome_funcao	é o nome da função
argumento1,argumento2	são argumentos a serem usados pela função. Pode ser representado por um nome de coluna ou uma expressão.

Funções de uma Única Linha

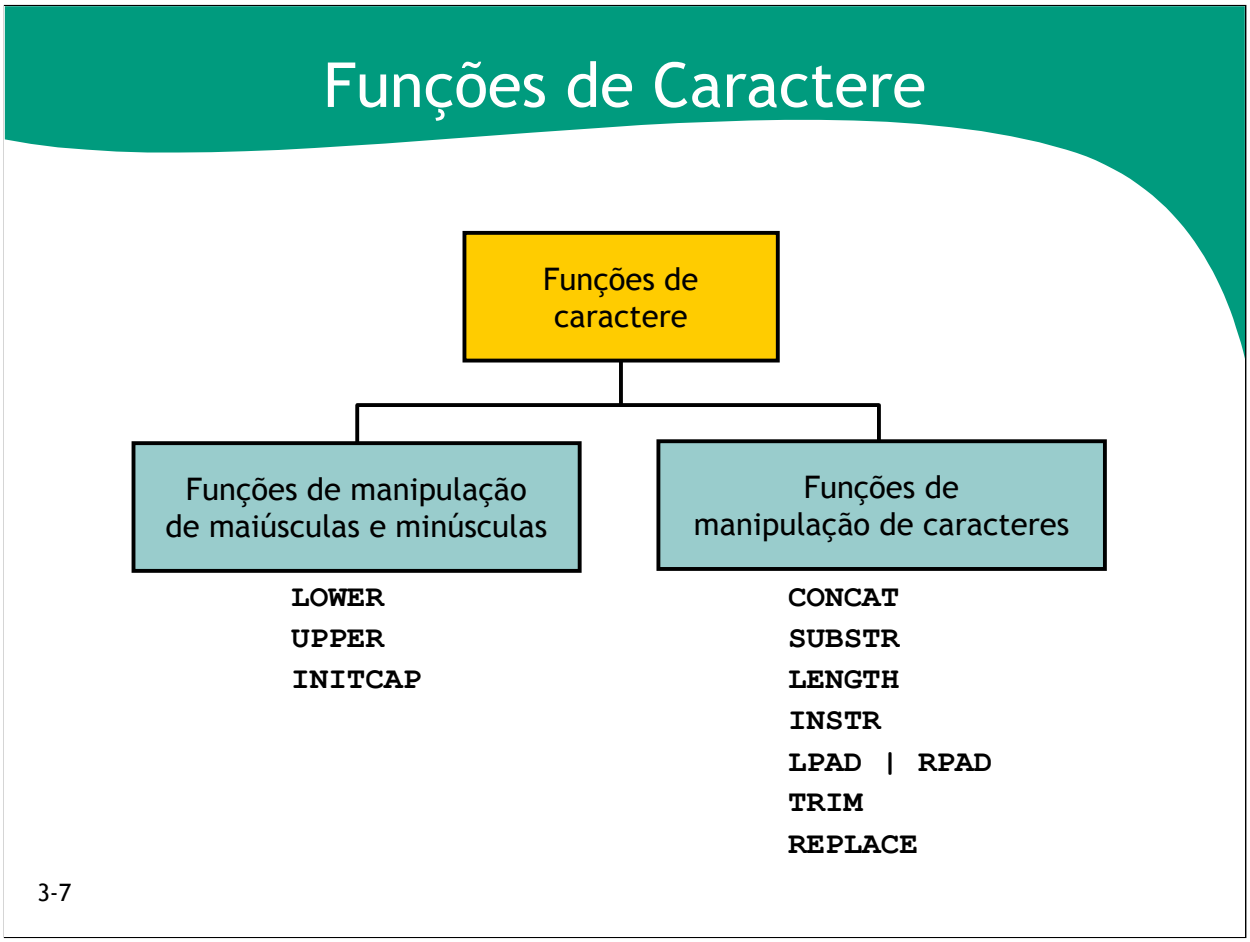


3-6

Funções de uma Única Linha (continuação)

Este capítulo aborda as seguintes funções de uma única linha:

- **Funções de caractere**: Aceitam a entrada de caracteres e podem retornar valores numéricos ou de caractere
- **Funções numéricas**: Aceitam a entrada numérica e retornam valores numéricos
- **Funções de data**: Operam em valores do tipo de dados `DATE` (Todas as funções de data retornam um valor de tipo de dados `DATE`, com exceção da função `MONTHS_BETWEEN`, que retorna um número.)
- **Funções de conversão**: Convertem um valor de um tipo de dados em outro
- **Funções gerais:** `NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.`
- **Funções de expressões regulares**: Usam expressões regulares para pesquisa de dados. As funções de expressões regulares serão vistas em um capítulo posterior.



Funções de Caractere

As funções de caractere de uma única linha aceitam dados de caractere como entrada e podem retornar valores numéricos ou de caractere. É possível dividir as funções de caractere em:

- Funções de manipulação de maiúsculas e minúsculas
- Funções de manipulação de caracteres

Função	Objetivo
LOWER(*coluna|expressão*)	Converte valores de caractere em minúsculas
UPPER(*coluna|expressão*)	Converte valores de caractere em maiúsculas
INITCAP(*coluna|expressão*)	Converte valores de caractere em maiúsculas (apenas a primeira letra de cada palavra; todas as outras letras permanecem minúsculas)
CONCAT(*coluna1|expressão1*, *coluna2|expressão2*)	Concatena o primeiro valor de caractere para o segundo valor de caractere; equivalente ao operador de concatenação (||)
SUBSTR(*coluna|expressão,m[,n]*)	Retorna caracteres especificados a partir do valor de caractere que inicia na posição *m*, *n* caracteres (Se *m* for negativo, a contagem iniciará a partir do final do valor do caractere. Se *n* for omitido, todos os caracteres do final da string serão retornados.)

Funções de Caractere (continuação)

Função	Objetivo
LENGTH(*coluna|expressão*)	Retorna a quantidade de caracteres na expressão.
INSTR(*coluna|expressão, 'texto', [,m], [n]*)	Retorna a posição numérica de um texto. Como opção, você pode fornecer uma posição *m* para iniciar a pesquisa e a ocorrência *n* da string. O default de *m* e *n* é 1, ou seja, começar a pesquisa desde o início e reportar a primeira ocorrência.
LPAD(*coluna|expressão, n,* '*texto*') RPAD(*coluna|expressão, n,* '*texto*')	Insere o valor de caractere justificado à direita com uma largura total de *n* posições de caractere Insere o valor do caractere justificado à esquerda com uma largura total de *n* posições de caractere.
TRIM(*LEADING|TRAILING|BOTH, caracter_a_ser_removido FROM texto_fonte*)	Permite reduzir os caracteres à direita (TRAILING), à esquerda (LEADING) ou nas duas direções (BOTH ou sem especificação) de uma string de caracteres. Se *caracter_a_ser_removido* não for especificado o padrão é um espaço em branco. Se for especificado apenas o *texto_fonte*, o Oracle remove espaços em branco a direita e a esquerda. Se *caracter_a_ser_removido* ou *texto_fonte* for NULL a função TRIM retorna NULL.
REPLACE(*texto, texto_de_pesquisa, texto_de_substituição*)	Procura uma string de caracteres em uma expressão de texto e substitui essa string por uma string de substituição especificada quando a encontra.

Observação: Esta lista contém somente algumas das funções de caractere disponíveis. O Manual "*Oracle SQL Reference"* contém uma lista completa.

Manipulação de Maiúsculas e Minúsculas

- Funções de conversão em caracteres de letras maiúsculas em minúsculas e vice-versa:

Função	Resultado
LOWER('TREINAMENTO sql')	Treinamento sql
UPPER('TREINAMENTO sql')	TREINAMENTO SQL
INITCAP('TREINAMENTO sql')	Treinamento Sql

3-9

Funções de Manipulação de Maiúsculas e Minúsculas

`LOWER`, `UPPER` e `INITCAP` são as três funções de conversão entre maiúsculas e minúsculas.

- **`LOWER`**: Converte o caractere de entrada em um caractere com todas as letras em minúsculas.
- **`UPPER`**: Converte o caractere de entrada em um caractere com todas as letras em maiúsculas.
- **`INITCAP`**: Converte a primeira letra de cada palavra em maiúscula e as letras restantes em minúsculas

```
SELECT 'O cargo de '||UPPER(nome)||' é '
||LOWER(cod_cargo) AS "DETALHE DO FUNCIONARIO“
FROM FUNCIONARIO;
```

DETALHE DO FUNCIONARIO
O cargo de ROBERTO é ad_pres
O cargo de NAIR é ad_vp
O cargo de LEONARDO é ad_vp
O cargo de ALEXANDRE é it_prog
O cargo de PEDRO é it_prog
O cargo de DILMA é it_prog
O cargo de MANUELA é es_ger
O cargo de TOMÉ é es_aux
O cargo de CRISTÓVÃO é es_aux

Manipulação de Maiúsculas e Minúsculas

- Exiba o número, o nome e o número de departamento do funcionário Higgins:

```
SQL> SELECT cod_funcionario, sobrenome, cod_departamento
  2  FROM   funcionario
  3  WHERE  sobrenome = 'chaves';

não há linhas selecionadas
```

```
SQL> SELECT cod_funcionario, sobrenome, cod_departamento
  2  FROM   funcionario
  3  WHERE  Lower(sobrenome) = 'chaves';

COD_FUNCIONARIO SOBRENOME   COD_DEPARTAMENTO
--------------- ----------- ----------------
            144 Chaves                   50
```

3-10

Funções de Manipulação de Maiúsculas e Minúsculas (Continuação)

O exemplo do slide exibe o código, o sobrenome e o código do departamento do funcionário de sobrenome Chaves.

A cláusula `WHERE` da primeira instrução SQL especifica o nome do funcionário como `chaves`. Como todos os dados da tabela `FUNCIONARIO` estão armazenados com as iniciais das palavras em letras maiúsculas, o nome `chaves` não detecta uma correspondência na tabela e, como resultado, nenhuma linha é selecionada.

A cláusula `WHERE` do segundo comando SQL especifica que o nome do funcionário na tabela `FUNCIONARIO` seja comparado a `chaves`, convertendo a coluna `SOBRENOME` em letras minúsculas para fins de comparação. Como agora os dois nomes estão em letras minúsculas, uma correspondência é detectada e uma linha é selecionada. É possível recriar a cláusula `WHERE` da seguinte maneira para produzir o mesmo resultado:

```
...WHERE last_name = 'Chaves'
```

O nome na saída aparece como foi armazenado no banco de dados. Para exibir o nome apenas com a primeira letra maiúscula, use a função `UPPER` no comando `SELECT`.

```
SELECT cod_funcionario, UPPER(sobrenome), cod_departamento
FROM funcionario
WHERE  INITCAP(sobrenome) = 'Chaves';
```

Funções de Manipulação de Caracteres

- Estas funções manipulam strings de caracteres:

Função	Resultado
CONCAT('Ola', 'Mundo')	OlaMundo
SUBSTR('OlaMundo',4,5)	Mundo
LENGTH('OlaMundo')	8
INSTR('OlaMundo', 'u')	5
LPAD(salario,10,'*')	******5000
RPAD(salario, 10, '*')	5000******
REPLACE ('FRANCO e FOSCO','F','B')	BRANCO e BOSCO
TRIM('O' FROM 'OlaMundo')	laMundo

3-11

Funções de Manipulação de Caracteres

`CONCAT`, `SUBSTR`, `LENGTH`, `INSTR`, `LPAD`, `RPAD` e `TRIM` são as funções de manipulação de caracteres abordadas neste capítulo.

- **`CONCAT`**: Une valores (Você está limitado a usar dois parâmetros com `CONCAT`.)
- **`SUBSTR`**: Extrai uma string de tamanho determinado
- **`LENGTH`**: Mostra o tamanho de uma string como um valor numérico
- **`INSTR`**: Localiza a posição numérica de um caractere
- **`LPAD`**: Preenche o valor do caractere à direita
- **`RPAD`**: Preenche o valor do caractere à esquerda
- **`TRIM`**: Reduz os caracteres à direita ou à esquerda (ou nas duas direções) de uma string de caracteres

Observação: É possível usar funções como `UPPER` e `LOWER` com a substituição de E comercial. Por exemplo, use `UPPER('&cargo')` para que o usuário não precise especificar o cargo em letras maiúsculas ou minúsculas.

Funções de Manipulação de Caracteres

```
SELECT cod_funcionario, CONCAT(nome, sobrenome) NOME,
       cod_cargo, LENGTH (sobrenome),
       INSTR(sobrenome, 'a') "Contem 'a'?"
FROM   funcionario
WHERE  SUBSTR(cod_cargo, 4) = 'REP';
```

COD_FUNCIONARIO	NOME	COD_CARGO	LENGTH(SOBRENOME)	Contem 'a'?
174	PamelaSue	VE_REP	3	0
176	EliasVoorhees	VE_REP	8	0
178	SadakoYamamura	VE_REP	8	2

3-12

Funções de Manipulação de Caracteres (Continuação)

O exemplo do slide exibe os nomes e os sobrenomes dos funcionários unidos, o tamanho do sobrenome do funcionário e a posição numérica da letra *a* no sobrenome de todos os funcionários cujo código do cargo contém a string `REP` a partir da quarta posição.

Exemplo

Modifique a instrução SQL no slide para exibir os dados relativos aos funcionários cujos sobrenomes terminam com a letra *o*.

```
SELECT cod_funcionario, CONCAT(nome, sobrenome) NOME,
       cod_cargo, LENGTH (sobrenome),
       INSTR(sobrenome, 'a') "Contem 'a'?"
FROM   funcionario
WHERE  SUBSTR(sobrenome, -1, 1) = 'o';
```

COD_FUNCIONARIO	NOME	COD_CARGO	LENGTH(SOBRENOME)	Contem 'a'?
103	AlexandreHonorato	IT_PROG	8	6
104	PedroOsterno	IT_PROG	7	0
200	LucianaTrajano	AD_ASST	7	3
206	RobertoNascimento	CTPUB_GER	10	2

Funções de Número

- ROUND: Arredonda o valor até o decimal especificado
- TRUNC: Trunca o valor até o decimal especificado
- MOD: Retorna o resto da divisão

Função	Resultado
`ROUND(45.926, 2)`	`45.93`
`TRUNC(45.926, 2)`	`45.92`
`MOD(1600, 300)`	`100`

3-13

Funções de Número

As funções de número aceitam a entrada numérica e retornam valores numéricos. Segue a descrição de algumas funções de número:

Função	Objetivo
ROUND(*coluna|expressão, n*)	Arredonda a coluna, a expressão ou o valor para *n* casas decimais; se *n* for omitido, não haverá casas decimais (Se *n* for negativo, os números à esquerda da vírgula decimal serão arredondados).
TRUNC(*coluna|expressão, n*)	Trunca a coluna, a expressão ou o valor para *n* casas decimais; se *n* for omitido, *n* assumirá o padrão zero.
MOD(*m,n*)	Retorna o resto de *m* dividido por *n*

Observação: Esta lista contém somente algumas das funções de número disponíveis. O Manual "*Oracle SQL Reference*" contém uma lista completa.

Função ROUND

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2         ROUND(45.923,-1)
  3  FROM   DUAL;

ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
          45,92              46               50
```

3-14

Função ROUND

A função ROUND arredonda a coluna, a expressão ou o valor até *n* casas decimais. Se o segundo argumento for 0 ou não estiver presente, o valor será arredondado até zero casas decimais. Se o segundo argumento for 2, o valor será arredondado até duas casas decimais. Por outro lado, se o segundo argumento for -2, o valor será arredondado até duas casas decimais à esquerda (arredondado para a unidade mais próxima de 10).

Também é possível usar a função ROUND com funções de data. São fornecidos exemplos posteriormente neste capítulo.

Função TRUNC

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
  2         TRUNC(45.923,-1)
  3  FROM   DUAL;

TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-1)
--------------- ------------- ----------------
          45,92            45               40
```

3-15

Função TRUNC

A função TRUNC trunca a coluna, a expressão ou o valor até n casas decimais.

Essa função opera com argumentos semelhantes aos da função ROUND. Se o segundo argumento for 0 ou não estiver presente, o valor será truncado até zero casas decimais. Se o segundo argumento for 2, o valor será truncado até duas casas decimais. Por outro lado, se o segundo argumento for -2, o valor será truncado até duas casas decimais à esquerda. Se o segundo argumento for -1, o valor será truncado até uma casa decimal à esquerda.

Assim como a função ROUND, é possível usar a função TRUNC com funções de data.

Função MOD

- Para todos os funcionários com o cargo de representante de vendas, calcule o resto do salário após dividi-lo por 5.000.

```
SELECT sobrenome, salario, MOD(salario, 5000)
FROM   funcionario
WHERE  cod_cargo = 'VE_REP';
```

SOBRENOME	SALARIO	MOD(SALARIO,5000)
Sue	11000	1000
Voorhees	8600	3600
Yamamura	7000	2000

3-16

Função MOD

A função MOD localiza o resto do primeiro argumento dividido pelo segundo argumento. O exemplo do slide calcula o resto do salário após a divisão por 5.000 para todos os funcionários cujo código de cargo é VE_REP.

A função MOD é usada com freqüência para determinar se o valor é par ou ímpar.

Formato de Datas

- O banco de dados Oracle armazena datas em um formato numérico interno: século, ano, mês, dia, horas, minutos e segundos.
- O formato padrão de exibição de data é DD-MON-RR.
  - Permite armazenar as datas do século XXI no século XX especificando apenas os últimos dois dígitos do ano
  - Permite armazenar as datas do século XX no século XXI da mesma forma

```
SQL> SELECT sobrenome, data_admissao
  2  FROM   funcionario
  3  WHERE  data_admissao < '01-JUL-97';

SOBRENOME   DATA_ADMISSAO
----------- -------------
Carlos      17-JUN-97
```

3-17

Formato de Datas

O banco de dados Oracle armazena datas em um formato numérico interno, que representa o século, o ano, o mês, o dia, as horas, os minutos e os segundos.

A exibição e o formato de entrada padrão de qualquer data correspondem a DD-MON-RR. As datas válidas no Oracle são de 1 de janeiro de 4712 A.C. a 31 de dezembro 9999 D.C.

No exemplo do slide, a saída da coluna `DATA_ADMISSAO` é exibida no formato padrão DD-MON-RR. No entanto, as datas não são armazenadas no banco de dados nesse formato. Todos os componentes da data e do horário são armazenados. Portanto, embora um valor de `DATA_ADMISSAO` como 17-JUN-97 seja exibido no formato dia, mês e ano, também há informações sobre *hora* e *século* associadas à data.

Formato de Datas (continuação)

Estes dados são armazenados internamente da seguinte maneira:

SÉCULO - ANO – MÊS – DIA – HORA – MINUTO - SEGUNDO

19 87 06 17 17 10 43

Séculos e o Ano 2000

Quando um registro com uma coluna de data é inserido em uma tabela, as informações sobre o *século* são obtidas da função `SYSDATE`. No entanto, quando a coluna de data é exibida na tela, o componente de século não é mostrado (por padrão).

O tipo de dados `DATE` sempre armazena internamente as informações sobre o ano como um número de quatro dígitos: dois dígitos para o século e dois para o ano. Por exemplo, o banco de dados Oracle armazena o ano como 1987 ou 2004, e não apenas como 87 ou 04.

Função SYSDATE

- SYSDATE é uma função que retorna as seguintes informações do servidor de banco de dados:
  - Data
  - Hora

3-19

Função SYSDATE

SYSDATE é uma função de data que retorna a data e o hora atuais do servidor de banco de dados. Você pode usar SYSDATE como qualquer outro nome de coluna. Por exemplo, para exibir a data atual, selecione SYSDATE em uma tabela. É comum selecionar SYSDATE em uma tabela fictícia denominada DUAL.

Exemplo

Exiba a data atual usando a tabela DUAL.

```
SELECT  SYSDATE
FROM    DUAL;
```

SYSDATE
08-MAI-09

Aritmética com Datas

- Some um número a uma data ou subtraia-o dessa data para obter um valor de data resultante.
- Subtraia uma data de outra para descobrir o número de dias entre elas.
- Some horas a uma data dividindo o número de horas por 24.

3-20

Aritmética com Datas

Como o banco de dados armazena datas como números, você pode realizar cálculos usando operadores aritméticos como os de adição e subtração. É possível somar e subtrair constantes de números e datas.

Você pode executar as seguintes operações:

Operação	Resultado	Descrição
data + número	Data	Adiciona um número de dias a uma data
data – número	Data	Subtrai um número de dias de uma data
data – data	Número de dias	Subtrai uma data de outra
data + número/24	Data	Adiciona um número de horas a uma data

Aritmética com Datas

```
SQL> SELECT sobrenome, (SYSDATE-data_admissao)/7 AS SEMANAS
  2  FROM   funcionario
  3  WHERE  cod_departamento = 90;

SOBRENOME       SEMANAS
----------- ----------
Carlos      620,429856
Martins     502,429856
da Silva    329,572713
```

3-21

Aritmética com Datas (continuação)

O exemplo do slide exibe o sobrenome e o número de semanas desde a admissão de todos os funcionários do departamento 90. Ele subtrai a data na qual o funcionário foi admitido da data atual (`SYSDATE`) e divide o resultado por 7 para calcular há quantas semanas a pessoa está admitida.

`SYSDATE` é uma função SQL que retorna a data e o hora atuais. Os resultados podem diferir do exemplo.

Se uma data mais recente for subtraída de uma data mais antiga, a diferença será um número negativo.

Funções de Data

Função	Resultado
MONTHS_BETWEEN	Número de meses entre duas datas
ADD_MONTHS	Adiciona meses do calendário à data
NEXT_DAY	Dia seguinte ao da data especificada
LAST_DAY	Último dia do mês
ROUND	Arredonda a data
TRUNC	Trunca a data

3-22

Funções de Data

Todas as funções de data retornam um valor do tipo de dados DATE, com exceção de MONTHS_BETWEEN, que retorna um valor numérico.

- **MONTHS_BETWEEN(*data1, data2*)**: Obtém o número de meses entre *data1* e *data2*. O resultado pode ser positivo ou negativo. Se *data1* for posterior a *data2*, o resultado será positivo; caso contrário, o resultado será negativo. A parte decimal do resultado representa uma parte do mês.
- **ADD_MONTHS(*data, n*)**: Adiciona *n* meses do calendário à *data*. O valor *n* deve ser inteiro e pode ser negativo.
- **NEXT_DAY(*data, 'char'*)**: Obtém a data do próximo dia especificado da semana ('*char*') após a *data* em questão. O valor de *char* pode ser um número que represente um dia ou uma string de caracteres.
- **LAST_DAY(*data*)**: Obtém a data do último dia do mês que contém a *data* em questão.
- **ROUND(*data*[,'*fmt*'])**: Retorna a *data* arredondada até a unidade especificada pelo modelo de formato *fmt*. Se o modelo de formato *fmt* for omitido, a *data* será arredondada até o dia mais próximo.
- **TRUNC(*data*[, '*fmt*'])**: Retorna a *data* com a parte do horário do dia truncada até a unidade especificada pelo modelo de formato *fmt*. Se o modelo de formato *fmt* for omitido, a data será truncada até o dia mais próximo.

A lista é um subconjunto de funções de data disponíveis. Os modelos de formato são abordados posteriormente neste capítulo. Os exemplos de modelos de formato são month e year.

Funções de Data

Função	Resultado
MONTHS_BETWEEN ('01-SET-95','11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY ('01-SET-95','SEXTA-FEIRA')	'08-SET-95'
LAST_DAY ('01-FEV-95')	'28-FEV-95'

3-23

Funções de Data (continuação)

Por exemplo, exiba o número de funcionário, a data de admissão, o número de meses desde a admissão, a data da revisão semestral, a primeira sexta-feira após a data de admissão e o último dia do mês da admissão relativos a todos os funcionários admitidos há menos de 36 meses.

```
SELECT cod_funcionario, data_admissao,
       MONTHS_BETWEEN (SYSDATE, data_admissao) MESES,
       ADD_MONTHS (data_admissao, 6) REVISAO,
       NEXT_DAY (data_admissao, 'SEXTA-FEIRA'),
       LAST_DAY(data_admissao)
FROM   funcionario
WHERE  MONTHS_BETWEEN (SYSDATE, data_admissao) < 36;
```

```
COD_FUNCIONARIO DATA_ADMISSAO      MESES REVISAO   NEXT_DAY( LAST_DAY(
--------------- ------------ ---------- --------- --------- ---------
            107 07-FEV-09     3,03277106 07-AGO-09 13-FEV-09 28-FEV-09
            124 16-NOV-08     5,74244848 16-MAI-09 21-NOV-08 30-NOV-08
            142 29-JAN-07     27,3230936 29-JUL-07 02-FEV-07 31-JAN-07
            143 15-MAR-08     13,7747065 15-SET-08 21-MAR-08 31-MAR-08
            144 09-JUL-08     9,96825493 09-JAN-09 11-JUL-08 31-JUL-08
            174 11-MAI-06     35,9037388 11-NOV-06 12-MAI-06 31-MAI-06
            202 17-AGO-07     20,7101904 17-FEV-08 24-AGO-07 31-AGO-07
```

Funções de Data

- Suponha que SYSDATE = '25-JUL-08':

Função	Resultado
`ROUND(SYSDATE,'MONTH')`	`01-AGO-08`
`ROUND(SYSDATE ,'YEAR')`	`01-JAN-09`
`TRUNC(SYSDATE ,'MONTH')`	`01-JUL-08`
`TRUNC(SYSDATE ,'YEAR')`	`01-JAN-08`

3-24

Funções de Data (continuação)

É possível usar as funções `ROUND` e `TRUNC` para valores de número e data. Quando usadas com datas, essas funções arredondam ou truncam o valor até o modelo de formato especificado. Portanto, você pode arredondar as datas até o mês ou o ano mais próximo.

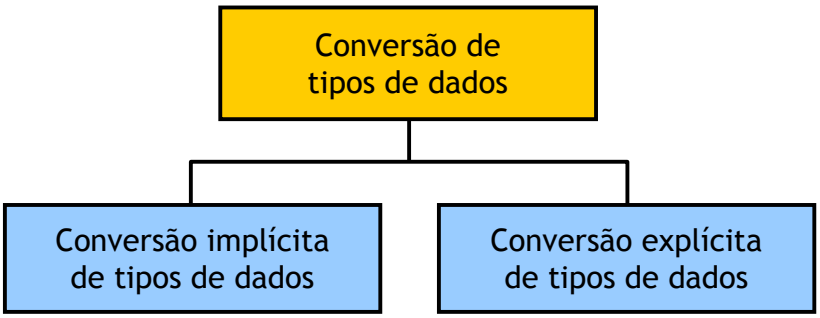
Exemplo

Compare as datas de admissão de todos os funcionários admitidos em 2007. Exiba o código do funcionário, a data de admissão e o mês da admissão usando as funções `ROUND` e `TRUNC`.

```
SELECT cod_funcionario, data_admissao,
       ROUND(data_admissao, 'MONTH'),
       TRUNC(data_admissao, 'MONTH')
FROM   funcionario
WHERE  data_admissao LIKE '%07';
```

```
COD_FUNCIONARIO DATA_ADMISSAO ROUND(DAT TRUNC(DAT
--------------- ------------- --------- ---------
            142 29-JAN-07     01-FEV-07 01-JAN-07
            202 17-AGO-07     01-SET-07 01-AGO-07
```

Funções de Conversão



3-25

Funções de Conversão

Algumas vezes você precisa converter um valor de um tipo de dado para outro. O banco de dados Oracle converte automaticamente um valor de um tipo para outro quando essa conversão faz sentido. Porém, recomenda-se que se utilize conversões explícitas pelas seguintes razões:

- Os comandos SQLs são mais fáceis de se entender quando você usa conversão explícita;
- Conversão implícita pode ter um efeito negativo na performance de execução do comando;
- Dependendo do contexto, uma conversão implícita pode não funcionar da mesma forma em todos os casos, por exemplo, uma conversão implícita de uma data para caractere pode retornar um ano diferente dependendo da variável NLS_DATE_FORMAT;
- Algoritmos de conversão implícita podem mudar de uma versão para outra do Oracle, isso faria com que um comando não funcione corretamente após uma atualização de versão de banco de dados.

As conversões explícitas de tipos de dados são feitas por meio das funções de conversão. Em geral, a forma dos nomes das funções segue a convenção `tipo de dados TO tipo de dados`. O primeiro tipo de dados é o da entrada, e o segundo, o da saída

As conversões implícitas de tipos de dados funcionam de acordo com as regras explicadas nos próximos dois slides.

Conversão Implícita de Tipos de Dados

- Parte da matriz de conversão implícita do Oracle:

	CHAR	VARCHAR2	DATE	NUMBER	LONG	RAW	ROWID	CLOB	BLOB
CHAR	--	X	X	X	X	X	--	X	X
VARCHAR2	X	--	X	X	X	X	X	X	--
DATE	X	X	--	--	--	--	--	--	--
NUMBER	X	X	--	--	--	--	--	--	--
LONG	X	X	--	--	--	X	--	X	--
RAW	X	X	--	--	X	--	--	--	X
ROWID	--	X	--	--	--	--	--	--	--
CLOB	X	X	--	--	X	--	--	--	--
BLOB	--	--	--	--	--	X	--	--	--

3-26

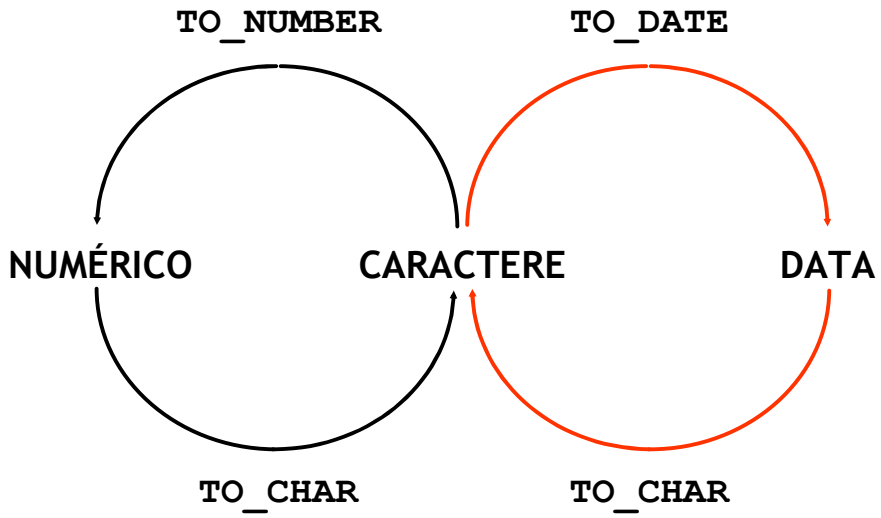
Conversão Implícita de Tipos de Dados

A tabela do slide mostra os principais tipos de conversão implícita possíveis, sem levar em consideração a direção da conversão ou o contexto no qual ela é feita.

A tabela completa pode ser encontrada no manual “Oracle SQL Reference”.

Por exemplo, a expressão `data_nascimento > '01-JAN-90'` resulta na conversão implícita da string `'01-JAN-90'` em uma data.

Conversão Explícita de Tipos de Dados



3-27

Conversão Explícita de Tipos de Dados

O SQL fornece três funções para converter um valor de um tipo de dados em outro:

Função	Objetivo
`TO_CHAR(`*`número|data`*`,[ `*`fmt`*`], [`*`nlsparams`*`])`	Converte um valor de número ou data em uma string de caracteres VARCHAR2 com o modelo de formato *fmt* **Conversão de número:** O parâmetro nlsparams especifica os seguintes caracteres, que são retornados pelos elementos de formato numérico: • Caractere decimal • Separador de grupos • Símbolo de moeda local • Símbolo de moeda internacional Se nlsparams ou qualquer outro parâmetro for omitido, esta função utilizará os valores de parâmetro padrão para a sessão.

Conversão Explícita de Tipos de Dados (continuação)

Função	Objetivo
TO_CHAR(*número*|*data*,[*fmt*], [*nlsparams*])	**Conversão de data:** O parâmetro nlsparams especifica o idioma em que são retornados as abreviações e os nomes de dias e meses. Se esse parâmetro for omitido, esta função usará os idiomas de data padrão para a sessão.
TO_NUMBER(*texto*,*[fmt]*, *[nlsparams]*)	Converte uma string de caracteres com dígitos em um número no formato especificado pelo modelo de formato opcional *fmt*. Nesta função, o parâmetro nlsparams desempenha o mesmo papel que na função TO_CHAR no que diz respeito à conversão de número.
TO_DATE(texto,[*fmt*], [*nlsparams*])	Converte uma string de caracteres que representa uma data em um valor de data de acordo com o *fmt* especificado. Se o *fmt* for omitido, o formato será DD-MON-YY. Nesta função, o parâmetro nlsparams desempenha o mesmo papel que na função TO_CHAR no que diz respeito à conversão de data.

Observação: A lista de funções mencionada neste capítulo contém apenas algumas das funções de conversão disponíveis.

Para obter mais informações consulte o manual *Oracle SQL Reference*.

Função TO_CHAR com Datas

```
TO_CHAR(data, 'modelo_de_formato')
```

- O modelo de formato:
  - Deve ser delimitado por aspas simples
  - Faz distinção entre maiúsculas e minúsculas
  - Pode incluir qualquer elemento de formato de data válido
  - Tem um elemento fm para remover os espaços vazios preenchidos ou para suprimir os zeros à esquerda
  - É separado do valor da data por vírgula

3-29

Função TO_CHAR com Datas

Anteriormente, todos os valores de data do Oracle eram exibidos no formato DD-MON-RR. Você pode usar a função `TO_CHAR` para converter uma data desse formato padrão em um formato especificado.

Diretrizes

- O modelo de formato deve ser delimitado por aspas simples e fazer distinção entre maiúsculas e minúsculas.
- O modelo de formato pode incluir qualquer elemento de formato de data válido. Separe o valor da data do modelo de formato por vírgula.
- Os espaços vazios em nomes de dias e meses na saída serão preenchidos automaticamente.
- Para remover os espaços vazios preenchidos ou suprimir os zeros à esquerda, use o elemento fm do modo de preenchimento.
- Você pode formatar o campo de caracteres resultante com o comando SQL*Plus `COLUMN` (abordado em um capítulo posterior).

```
SELECT cod_funcionario, TO_CHAR(data_admissao,
'MM/YY') MES_ADMISSAO
FROM   funcionario
WHERE  sobrenome = 'Chaves';
```

COD_FUNCIONARIO	MES_ADMISSAO
144	07/08

Elementos do Modelo de Formato de Data

Elemento	Resultado
YYYY	Ano completo em números
YEAR	Ano por extenso (em inglês)
MM	Valor de dois dígitos para o mês
MONTH	Nome completo do mês
MON	Abreviação de três letras do mês
DY	Abreviação de três letras do dia da semana
DAY	Nome completo do dia da semana
DD	Dia numérico do mês

3-30

Elementos do Modelo de Formato de Data

Um modelo de formato é um caractere literal que descreve o formato que se deseja exibir uma data.

Quando o modelo de formato é usado numa função TO_CHAR parta converter uma data em caractere, ele indica o formato resultante do caractere já convertido. Porém, o uso de um modelo de formato não altera a representação interna do valor no banco de dados.

A tabela do slide mostra alguns exemplos de literais que podem compor um modelo de formato de data.

Exemplo de Elementos de Formatos de Data Válidos

Elemento	Descrição
SCC ou CC	Século; o servidor prefixa a data A.C. com -
Anos em datas YYYY ou SYYYY	Ano; o servidor prefixa a data A.C. com -
YYY ou YY ou Y	Últimos três ou dois dígitos do ano, ou apenas o último dígito
Y.YYY	Ano com ponto nesta posição
IYYY, IYY, IY, I	Ano de quatro, três, dois ou um dígito baseado no padrão ISO
SYEAR ou YEAR	Ano por extenso; o servidor prefixa a data A.C. com -
BC ou AD	Indica o ano A.C. ou D.C.
B.C. ou A.D.	Indica o ano A.C. ou D.C. usando pontos
Q	Trimestre do ano
MM	Mês: valor de dois dígitos
MONTH	Nome do mês preenchido com espaços em branco (até nove caracteres)
MON	Nome do mês; abreviação de três letras.
RM	Mês em numeral romano
WW ou W	Semana do ano ou mês
DDD ou DD ou D	Dia do ano, do mês ou da semana
DAY	Nome do dia preenchido com espaços em branco (até nove caracteres)
DY	Nome do dia; abreviação de três letras
J	Dia Juliano; o número de dias desde 31 de dezembro de 4713 A.C.

Elementos do Modelo de Formato de Data

- Os elementos de horário formatam a parte relativa ao horário da data:

`HH24:MI:SS AM`	`15:45:32 PM`

- Adicione strings de caracteres delimitando-as por aspas duplas:

`DD "de" MONTH`	`12 de OUTUBRO`

- Os sufixos de números exibem os números por extenso:

`ddspth`	`fourteenth`

3-32

Elementos de Formato de Data: Formatos de Horário

Use os formatos listados a seguir para exibir informações sobre horário.

Elemento	Descrição
AM ou PM	Indicador de meridiano
A.M. ou P.M.	Indicador de meridiano com pontos
HH ou HH12 ou HH24	Hora do dia, hora (1-12) ou hora (0-23)
MI	Minuto (0-59)
SS	Segundo (0-59)
SSSSS	Segundos após meia-noite (0-86399)

Outros Formatos:

Elemento	Descrição
/ . ,	Pontuação é reproduzida no resultado.
“de”	String entre aspas é reproduzida no resultado.

Especificando Sufixos para Influenciar a Exibição de Números:

Elemento	Descrição
TH	Número ordinal (por exemplo, DDTH para 4TH)
SP	Números por extenso (por exemplo, DDSP para FOUR)
SPTH or THSP	Número ordinal por extenso (por exemplo, DDSPTH para FOURTH)

Função TO_CHAR com Datas: Exemplo

```
SELECT sobrenome,
       TO_CHAR(data_admissao, 'fmDD Month YYYY')
       AS DATA_ADMISSAO
FROM   funcionario;
```

SOBRENOME	DATA_ADMISSAO
Carlos	17 Junho 1997
Martins	21 Setembro 1999
da Silva	13 Janeiro 2003
Honorato	3 Janeiro 2000
Osterno	21 Maio 2001
Barata	7 Fevereiro 2009
Brunni	16 Novembro 2008
Lopes	17 Outubro 2005
Cabral	29 Janeiro 2007
Miranda	15 Marco 2008

3-33

Função `TO_CHAR` com Datas: Exemplo

O comando SQL do slide exibe os sobrenomes e as datas de admissão de todos os funcionários. A data de admissão aparece como 15 Março 2008.

Função TO_CHAR com Números

```
TO_CHAR(número, 'modelo_de_formato')
```

- Estes são alguns dos elementos do modelo de formato que você pode usar com a função TO_CHAR para exibir um valor de número como um caractere:

Elemento	Resultado
9	Representa um número
0	Impõe a exibição de um zero
$	Insere um sinal de dólar flutuante
L	Usa o símbolo da moeda local flutuante
.	Imprime uma casa decimal
,	Imprime uma vírgula como indicador de milhar

3-34

Função TO_CHAR com Números

Quando estiver trabalhando com valores de números como strings de caracteres, converta esses números no tipo de dados de caractere usando a função TO_CHAR, que converte um valor do tipo de dados NUMBER no tipo de dados VARCHAR2. Essa técnica é especialmente útil para realizar a concatenação de caracteres.

Função TO_CHAR com Números (continuação)

Elementos de Formato de Número

Se você estiver convertendo um número no tipo de dados de caractere, poderá usar os seguintes elementos de formato:

Elemento	Descrição	Exemplo	Resultado
9	Posição numérica (o número de 9s determina a largura da exibição)	999999	1234
0	Exibe zeros à esquerda	099999	001234
$	Símbolo de dólar flutuante	$999999	$1234
L	Símbolo de moeda local flutuante	L999999	R$1234
D	Retorna o caractere decimal na posição especificada. O default é uma vírgula (,).	99,99	99D99
.	Vírgula decimal na posição especificada	999999,99	1234,00
G	Retorna o separador de grupos na posição especificada. Você pode especificar vários separadores de grupos em um modelo de formato numérico.	9.999	9G999
,	Ponto na posição especificada	999.999	1.234
MI	Sinal de subtração à direita (valores negativos)	999999MI	1234-
PR	Números negativos entre colchetes	999999PR	<1234>
EEEE	Notação científica (o formato deve especificar quatro Es)	99,999EEEE	1,234E+03
U	Retorna o símbolo monetário dual "Euro" (ou outro) na posição especificada	U9999	€1234
V	Multiplica por 10 *n* vezes (*n* = número de 9s após V)	9999V99	123400
S	Retorna o valor negativo ou positivo	S9999	-1234 ou +1234
B	Exibe valores zero em branco, e não 0	B9999,99	1234,00

Função TO_CHAR com Números

```
SQL> SELECT TO_CHAR(salario, '$99,999.00') SALARIO
  2  FROM   funcionario
  3  WHERE  nome = 'Alexandre';

SALARIO
-----------
  $9,000.00
```

3-36

Função TO_CHAR com Números (Continuação)

O slide mostra exemplos de uso da função TO_CHAR para converter números.

O Oracle exibe uma string de símbolos de número (#) no lugar de um número inteiro cujos dígitos ultrapassam o número de dígitos fornecido no modelo de formato.

O Oracle arredonda o valor decimal armazenado até o número de casas decimais do modelo de formato.

Funções TO_NUMBER e TO_DATE

- Converta uma string de caracteres em um formato de número usando a função TO_NUMBER:

```
TO_NUMBER(texto[, 'modelo_de_formato'])
```

- Converta uma string de caracteres em um formato de data usando a função TO_DATE:

```
TO_DATE(texto[, 'modelo_de_formato'])
```

- Estas funções têm um modificador fx. Esse modificador especifica a correspondência exata para o argumento de caractere e o modelo de formato de data de uma função TO_DATE.

3-37

Funções TO_NUMBER e TO_DATE

É possível converter uma string de caracteres em um número ou uma data. Para realizar essa tarefa, use a função `TO_NUMBER` ou `TO_DATE`. O modelo de formato escolhido baseia-se nos elementos de formato demonstrados anteriormente.

O modificador `fx` especifica a correspondência exata para o argumento de caractere e o modelo de formato de data de uma função `TO_DATE`:

- A pontuação e o texto devem corresponder exatamente (exceto em relação a maiúsculas e minúsculas) às partes associadas do modelo de formato.
- O argumento de texto não pode conter espaços em branco adicionais. Sem `fx`, o Oracle ignora os espaços em branco adicionais.
- Os dados numéricos no argumento de caractere devem ter o mesmo número de dígitos do elemento correspondente no modelo de formato. Sem `fx`, os números no argumento de caractere podem omitir os zeros à esquerda.

Funções TO_NUMBER e TO_DATE (continuação)

Exemplo:

Exiba o nome e a data de admissão de todos os funcionários admitidos em 24 de maio de 1999. Como o modificador `fx` é usado, uma correspondência exata é necessária e os espaços após a palavra *May* não são reconhecidos:

```
SQL> SELECT nome, data_admissao
  2  FROM   funcionario
  3  WHERE  data_admissao = TO_DATE('Maio  24,
1999', 'fxMonth DD, YYYY');
 WHERE  data_admissao = TO_DATE('Maio  24, 1999',
'fxMonth DD, YYYY')
                                *
ERRO na linha 3:
ORA-01858: a non-numeric character was found where
a numeric was expected
```

Elemento de Formato de Data RR

		Se o ano de dois dígitos especificado for:	
		0-49	50-99
Se os dois últimos dígitos do ano atual forem:	0-49	A data retornada estará contida no século atual	A data retornada estará contida no século imediatamente anterior ao atual
	50-99	A data retornada estará contida no século imediatamente posterior ao atual	A data retornada estará contida no século atual

Ano Atual	Data Especificada	Formato RR	Formato YY
1995	27/10/95	1995	1995
1995	27/10/17	2017	1917
2001	27/10/17	2017	2017
2001	27/10/95	1995	2095

3-39

Elemento de Formato de Data RR

O formato de data `RR` é semelhante ao elemento `YY`, mas pode ser usado para especificar séculos diferentes. Use o elemento de formato de data `RR`, em vez de `YY`, para que o século do valor retornado varie de acordo com o ano de dois dígitos especificado e com os últimos dois dígitos do ano atual. A tabela do slide resume o comportamento do elemento `RR`.

Elemento de Formato de Data RR

- Para localizar os funcionários admitidos antes de 1999, use o formato de data RR, que produz os mesmos resultados quer o comando seja executado em 1999 ou hoje:

```
SELECT sobrenome, TO_CHAR(data_admissao, 'DD-Mon-YYYY')
FROM  funcionario
WHERE data_admissao < TO_DATE('01-Jan-99','DD-Mon-RR');
```

SOBRENOME	TO_CHAR(DATA_ADMISSAO,'DD-MON-YYYY')
Carlos	17-Jun-1997
Voorhees	24-Mar-1998
Trajano	17-Set-1997

3-40

Elemento de Formato de Data RR (continuação)

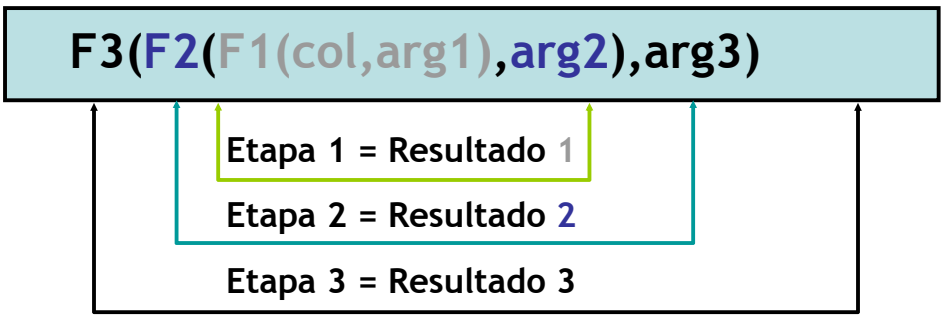
Para localizar os funcionários admitidos antes de 1999, você poderá usar o formato `RR`. Como o ano atual é maior que 1999, o formato `RR` interpretará a parte da data relativa ao ano de 1950 a 1999.

O comando a seguir, por outro lado, tratá todos os funcionários mesmo contratados depois de 1999, porque ele fará a comparação com o ano de 2099.

```
SELECT sobrenome,
       TO_CHAR(data_admissao, 'DD-Mon-YYYY')
FROM   funcionario
WHERE  data_admissao < TO_DATE('01-Jan-99','DD-Mon-YY');
```

Aninhando Funções

- É possível aninhar as funções de uma única linha em quantos níveis forem necessários.
- As funções aninhadas são avaliadas a partir do nível mais interno para o nível mais externo.



3-41

Aninhando Funções

É possível aninhar as funções de uma única linha em quantos níveis forem necessários. As funções aninhadas são avaliadas do nível mais interno para o nível mais externo. Veja a seguir alguns exemplos que mostram a flexibilidade dessas funções.

Aninhando Funções

```
SQL> SELECT sobrenome,
  2         UPPER(CONCAT(SUBSTR (sobrenome, 1, 8), '_UK'))
  3  FROM   funcionario
  4  WHERE  cod_departamento = 20;
```

SOBRENOME	UPPER(CONCAT(SUBSTR(SOBRENOME,1,8),'_UK'))
Thacker	THACKER_UK
Kramer	KRAMER_UK

3-42

Aninhando Funções (continuação)

O exemplo do slide exibe os sobrenomes dos funcionários do departamento 20. A avaliação do comando SQL abrange três etapas:

1. A função interna recupera os oito primeiros caracteres do sobrenome.

```
Resultado1 = SUBSTR (SOBRENOME, 1, 8)
```

2. A função externa concatena o resultado com _UK.

```
Resultado2 = CONCAT(Resultado1, '_US')
```

3. A função mais externa converte os resultados em letras maiúsculas.

A expressão inteira torna-se o cabeçalho da coluna, pois não foi fornecido um apelido para essa coluna.

Exemplo

Exiba a data da próxima sexta-feira que está a seis meses da data de admissão. A data resultante deverá aparecer como Sexta-Feira, Agosto 13, 1999. Ordene os resultados por data de admissão.

```
SELECT    TO_CHAR(NEXT_DAY(ADD_MONTHS
          (data_admissao, 6), 'SEXTA-FEIRA'),
          'fmDay, Month DD, YYYY')
          "Revisão do Sexto Mês"
FROM      funcionario
ORDER BY data_admissao;
```

Funções Gerais

- As funções a seguir agem com qualquer tipo de dados e estão relacionadas ao uso de valores nulos:
  - NVL (expressão1, expressão2)
  - NVL2 (expressão1, expressão2, expressão3)
  - NULLIF (expressão1, expressão2)
  - COALESCE (expressão1, expressão2, ..., expressãoN)

3-43

Funções Gerais

Essas funções agem com qualquer tipo de dados e estão relacionadas ao uso de valores nulos na lista de expressões.

Função	Descrição
NVL	Converte um valor nulo em um valor definido
NVL2	Se expressão1 não for nulo, NVL2 retornará expressão2. Se expressão1 for nulo, NVL2 retornará expressão3. É possível expressar o argumento expressão1 em qualquer tipo de dados.
NULLIF	Compara duas expressões; se elas forem iguais, retornará um valor nulo e, se forem diferentes, retornará a primeira expressão.
COALESCE	Retorna a primeira expressão não nula da lista de expressões

Observação: Para obter mais informações sobre as centenas de funções disponíveis, consulte o manual *Oracle SQL Reference*.

Função NVL

- Converte um valor nulo em um valor determinado:
- É possível usar os tipos de dados de data, caractere e número.
- Os argumentos expressão1 e expressão2 podem ser de qualquer tipo de dados, porém se forem tipos diferentes o Oracle fará uma conversão implícita:
  - NVL(comissao,0)
  - NVL(data_nascimento,'01-JAN-97')
  - NVL(cargo,'Nenhum cargo')

3-44

Função NVL

É uma função condicional que retorna a expressão2 apenas no caso da expressão1 ser nulo, senão expressão1 é retornada.Para converter um valor nulo em um valor determinado, use a função `NVL`.

Sintaxe

`NVL (`*`expressão1`*`, `*`expressão2`*`)`

Na sintaxe:

- *`expressão1`* é o valor de origem ou a expressão que terá seu valor testado.
- *`expressão2`* é o valor de retorno apenas no caso de expressão1 conter um valor nulo.

Você pode aplicar a função `NVL` em qualquer tipo de dados, mas o valor retornado é sempre igual ao do tipo de dados de *`expressão1`*.

Execução da Função NVL com Vários Tipos de Dados

Tipo de Dados	Exemplo de Uso da Função
NUMBER	NVL(coluna_numerica, 9)
DATE	NVL(coluna_data, '01-JAN-95')
CHAR ou VARCHAR2	NVL(coluna_caractere,'Indeterminado')

Função NVL

```
SELECT nome, salario, NVL(comissao, 0),
       (salario*13) + (salario*12*NVL(comissao, 0)) SALARIO_ANUAL
FROM funcionario;
```

NOME	SALARIO	NVL(COMISSAO,0)	SALARIO_ANUAL
Roberto	24000	0	312000
Nair	17000	0	221000
Leonardo	17000	0	221000
Alexandre	9000	0	117000
Pedro	6000	0	78000
Dilma	4200	0	54600
Manuela	5800	0	75400
Tomé	3500	0	45500
Cristóvão	3100	0	40300
Rafael	2600	0	33800
Pedro	2500	0	32500
Paula	10500	,2	161700
Pamela	11000	,3	182600
Elias	8600	,2	132440
Sadako	7000	,15	103600
Luciana	4400	0	57200
William	13000	0	169000
Cosmo	6000	0	78000
Sheila	12000	0	156000
Roberto	8300	0	107900

3-45

Função `NVL` (continuação)

Para calcular a remuneração anual de todos os funcionários, você precisa multiplicar o salário mensal por 12 e adicionar o percentual de comissão ao resultado:

```
SELECT nome, salario, NVL(comissao, 0),
       (salario*13) + (salario*12*comissao) SALARIO_ANUAL
FROM funcionario;
```

NOME	SALARIO	NVL(COMISSAO,0)	SALARIO_ANUAL
Roberto	24000	0	
Nair	17000	0	
Leonardo	17000	0	
Alexandre	9000	0	
Pedro	6000	0	
Dilma	4200	0	
Manuela	5800	0	
Tomé	3500	0	
Cristóvão	3100	0	
Rafael	2600	0	
Pedro	2500	0	
Paula	10500	,2	161700
Pamela	11000	,3	182600

Observe que a remuneração anual é calculada apenas para os funcionários que recebem comissão. Se o valor de uma coluna da expressão for nulo, o resultado será nulo. Para calcular valores relativos a todos os funcionários, transforme o valor nulo em um número determinado antes de aplicar o operador aritmético. No exemplo do slide, a função `NVL` é usada para transformar valores nulos em zero.

Função NVL2

```
SELECT sobrenome,  salario, comissao,
       NVL2(comissao,'SAL+COMM', 'SAL') recebimento
FROM   funcionario WHERE cod_departamento IN (50, 80)
```

SOBRENOME	SALARIO	COMISSAO	RECEBIMENTO
Brunni	5800		SAL
Lopes	3500		SAL
Cabral	3100		SAL
Miranda	2600		SAL
Chaves	2500		SAL
Schultz	10500	,2	SAL+COMM
Sue	11000	,3	SAL+COMM
Voorhees	8600	,2	SAL+COMM

3-46

Função NVL2

A função NVL2 examina a primeira expressão. Se a primeira expressão não for nula, a função NVL2 retornará a segunda expressão. Se a primeira expressão for nula, a terceira expressão será retornada.

Sintaxe

```
NVL2 (expressão1, expressão2, expressão3)
```

Na sintaxe:

- *expressão1* é o valor de origem ou a expressão que será testada se possui um valor nulo.
- *expressão2* é o valor retornado quando *expressão1* não é nulo
- *expressão3* é o valor retornado quando *expressão2* é nulo.

No exemplo mostrado no slide, a coluna COMISSAO é examinada. Se for detectado um valor, a segunda expressão SALARIO+COMISSAO será retornada. Se a coluna COMISSAO contiver um valor nulo, a terceira expressão SALARIO será retornada.

É possível expressar o argumento *expressão1* em qualquer tipo de dados. Os argumentos *expressão2* e *expressão3* podem ser expressos em qualquer tipo de dados, exceto LONG. Se os tipos de dados de *expressão2* e *expressão3* forem diferentes, o servidor Oracle converterá *expressão3* no tipo de dados de *expressão2* antes de compará-los, a menos que *expressão3* seja uma constante nula. No último caso, não será necessária uma conversão de tipo de dados. O tipo de dados do valor retornado será sempre igual ao tipo de dados de *expressão2*, a menos que *expressão2* seja expresso em dados de caractere; nesse caso, o tipo de dados do valor retornado será VARCHAR2.

Função NULLIF

```
SELECT nome, LENGTH(nome) "expressão1",
       sobrenome,  LENGTH(sobrenome)  "expressão2",
       NULLIF(LENGTH(nome), LENGTH(sobrenome)) resultado
FROM   funcionario;
```

NOME	expressão1	SOBRENOME	expressão2	RESULTADO
Sheila	6	Almeida	7	6
Dilma	5	Barata	6	5
Manuela	7	Brunni	6	7
Cristóvão	9	Cabral	6	9
Roberto	7	Carlos	6	7
Pedro	5	Chaves	6	5
Alexandre	9	Honorato	8	9
Cosmo	5	Kramer	6	5
Tomé	4	Lopes	5	4
Nair	4	Martins	7	4
Rafael	6	Miranda	7	6
Roberto	7	Nascimento	10	7
Pedro	5	Osterno	7	5
Paula	5	Schultz	7	5
Pamela	6	Sue	3	6
William	7	Thacker	7	
Luciana	7	Trajano	7	
Elias	5	Voorhees	8	5
Sadako	6	Yamamura	8	6
Leonardo	8	da Silva	8	

3-47

Função NULLIF

A função NULLIF compara duas expressões. Se elas forem iguais, a função retornará um valor nulo. Se elas forem diferentes, a função retornará a primeira expressão. Não é possível especificar o literal NULL para a primeira expressão.

Sintaxe

```
NULLIF (expressão1, expressão2)
```

Na sintaxe:

- *expressão1* é o valor de origem comparado a *expressão2*
- *expressão2* é o valor de origem comparado a *expressão1* (Se ele não for igual a *expressão1*, *expressão1* será retornada.)

No exemplo do slide, o tamanho do nome na tabela FUNCIONAIO é comparado ao tamanho do sobrenome nessa mesma tabela. Quando os tamanhos dos nomes são iguais, um valor nulo é exibido. Quando eles são diferentes, é exibido o tamanho do nome.

A função NULLIF equivale logicamente à expressão CASE a seguir. A expressão CASE será abordada posteriormente neste capítulo:

```
CASE
WHEN expressão1 = expressão2 THEN NULL
ELSE expressão1 END
```

Função COALESCE

- A vantagem da função COALESCE em relação à função NVL é que a primeira pode assumir diversos valores alternativos.
- Se a primeira expressão não for nula, a função COALESCE retornará essa expressão; caso contrário, será usada a função COALESCE para as expressões restantes.

3-48

Função COALESCE

A função `COALESCE` retorna a primeira expressão não nula da lista.

Sintaxe

```
COALESCE (expressão1, expressão2, ... expressãoN)
```

Na sintaxe:

- *expressão1* retornará essa expressão se ela não for nula
- *expressão2* retornará essa expressão se ela não for nula e a primeira expressão for nula
- *expressãoN* retornará essa expressão se as expressões anteriores forem nulas

Todas as expressões devem ter o mesmo tipo de dados.

Função COALESCE

```
SELECT sobrenome,
       COALESCE(cod_gerente,comissao, -1) comissao
FROM   funcionario
ORDER BY comissao;
```

SOBRENOME	COMISSAO
Carlos	-1
Martins	100
da Silva	100
Thacker	100
Schultz	100
Brunni	100
Almeida	101
Trajano	101
Honorato	102
Osterno	103
Barata	103
Lopes	124
Chaves	124
Cabral	124
Miranda	124
Yamamura	149
Sue	149
Voorhees	149
Kramer	201
Nascimento	205

3-49

Função COALESCE (continuação)

No exemplo mostrado no slide, se o valor de COD_GERENTE não for nulo, ele será exibido. Se o valor de COD_GERENTE for nulo, COMISSAO será exibido. Se os valores de COD_GERENTE e COMISSAO forem nulos, será exibido o valor –1.

Expressões Condicionais

- Permitem usar a lógica IF-THEN-ELSE em uma instrução SQL
- Usam dois métodos:
  - Expressão CASE
  - Função DECODE

3-50

Expressões Condicionais

Os dois métodos usados para implementar o processamento condicional (a lógica IF-THEN-ELSE) em um comando SQL são a expressão `CASE` e a função `DECODE`.

Observação: A expressão `CASE` está de acordo com o padrão ANSI SQL. A função `DECODE` é específica da sintaxe do Oracle.

Expressão CASE

- Facilita consultas condicionais executando o trabalho de uma instrução IF-THEN-ELSE:

```
CASE expressão WHEN expressão_comparação1 THEN retorno1
               [WHEN expressão_comparação2 THEN retorno2
                WHEN expressão_comparaçãoN THEN retornoN
                ELSE retorno_else]
END
```

3-51

Expressão CASE

As expressões `CASE` permitem usar a lógica IF-THEN-ELSE em instruções SQL sem que sejam usados procedimentos.

Em uma expressão `CASE` simples, o servidor Oracle pesquisa o primeiro par `WHEN ... THEN` no qual expressão é igual a `expressão_comparação` e retorna `retorno`. Se nenhum par `WHEN ... THEN` atender a essa condição e existir uma cláusula `ELSE`, o servidor Oracle retornará `retorno_else`. Caso contrário, ele retornará um valor nulo. Não é possível especificar o literal `NULL` para todos os valores de `retorno` e `retorno_else`.

Todas as expressões (`expressão`, `expressão_comparação` e `retorno`) devem ter o mesmo tipo de dados, que pode ser `CHAR`, `VARCHAR2`, `NCHAR` ou `NVARCHAR2`.

Expressão CASE

- Facilita consultas condicionais executando o trabalho de uma instrução IF-THEN-ELSE:

```
SELECT sobrenome, cod_cargo, salario,
       CASE cod_cargo WHEN 'IT_PROG' THEN  1.10*salario
                      WHEN 'ES_AUX'  THEN  1.15*salario
                      WHEN 'VE_REP'  THEN  1.20*salario
       ELSE salario   END  "SALARIO_REVISADO"
FROM   funcionario;
```

SOBRENOME	COD_CARGO	SALARIO	SALARIO_REVISADO
Carlos	AD_PRES	24000	24000
Martins	AD_VP	17000	17000
da Silva	AD_VP	17000	17000
Honorato	IT_PROG	9000	9900
Osterno	IT_PROG	6000	6600
Barata	IT_PROG	4200	4620
Brunni	ES_GER	5800	5800
Lopes	ES_AUX	3500	4025
Cabral	ES_AUX	3100	3565

3-52

Expressão CASE (continuação)

No comando SQL do slide, o valor de COD_CARGO é decodificado. Se o valor de `COD_CARGO` for `IT_PROG`, o aumento de salário será de 10%; se esse valor for `ES_AUX`, o aumento de salário será de 15%; se o valor for `VE_REP`, o aumento de salário será de 20%. Para todos os outros cargos, não haverá aumento de salário.

É possível criar a mesma instrução com a função `DECODE`.

Este é um exemplo de uma expressão `CASE` pesquisada. Nessa expressão, a pesquisa é feita da esquerda para a direita até que seja localizada uma ocorrência da condição listada. Em seguida, obtém-se a expressão de retorno. Se nenhuma condição for considerada verdadeira e existir uma cláusula `ELSE`, a expressão de retorno nessa cláusula será exibida; caso contrário, será retornado `NULL`.

```
SELECT sobrenome,salario,
(CASE WHEN salario<5000 THEN 'Baixo'
      WHEN salario<10000 THEN 'Medio'
      WHEN salario<20000 THEN 'Bom'
      ELSE 'Excelentet'
END) salario_qualificado
FROM funcionario;
```

Função DECODE

- Facilita consultas condicionais executando o trabalho de uma expressão CASE ou de uma instrução IF-THEN-ELSE:

```
DECODE(coluna|expressão, pesquisa1, resultado1
                        [, pesquisa2, resultado2,...,]
                        [, retorno_else])
```

3-53

Função DECODE

A função DECODE decodifica uma expressão de maneira semelhante à lógica IF-THEN-ELSE usada em várias linguagens. Essa função decodifica a *expressão* depois de compará-la a cada valor da *pesquisa*. Se a expressão for igual à da *pesquisa*, o *resultado* será retornado.

Se o *retorno_else* for omitido, um valor nulo será retornado quando um valor da pesquisa não corresponder a nenhum dos valores do resultado.

Função DECODE

```
SELECT sobrenome, cod_cargo, salario,
       DECODE(cod_cargo,'IT_PROG',1.10*salario,
                        'ES_AUX', 1.15*salario,
                        'VE_REP', 1.20*salario,
              salario)
       SALARIO_REVISADO
FROM   funcionario;
```

SOBRENOME	COD_CARGO	SALARIO	SALARIO_REVISADO
Carlos	AD_PRES	24000	24000
Martins	AD_VP	17000	17000
da Silva	AD_VP	17000	17000
Honorato	IT_PROG	9000	9900
Osterno	IT_PROG	6000	6600
Barata	IT_PROG	4200	4620
Brunni	ES_GER	5800	5800
Lopes	ES_AUX	3500	4025
Cabral	ES_AUX	3100	3565
Miranda	ES_AUX	2600	2990
Chaves	ES_AUX	2500	2875
Schultz	VE_GER	10500	10500
Sue	VE_REP	11000	13200
Voorhees	VE_REP	8600	10320
Yamamura	VE_REP	7000	8400
Trajano	AD_ASST	4400	4400
Thacker	MK_GER	13000	13000
Kramer	MK_ANA	6000	6000
Almeida	CT_GER	12000	12000
Nascimento	CTPUB_GER	8300	8300

Função DECODE (continuação)

No comando SQL do slide, o valor de COD_CARGO é testado. Se o valor de COD_CARGO for IT_PROG, o aumento de salário será de 10%; se esse valor for ES_AUX, o aumento de salário será de 15%; se o valor for VE_REP, o aumento de salário será de 20%. Para todos os outros cargos, não haverá aumento de salário.

É possível expressar a mesma instrução em pseudocódigo como uma instrução IF-THEN-ELSE:

```
IF cod_cargo = 'IT_PROG'   THEN  salario = salario*1.10
IF cod_cargo = 'ES_AUX'    THEN  salario = salario*1.15
IF cod_cargo = 'VE_REP'    THEN  salario = salario*1.20
ELSE salario = salario
```

Função DECODE

- Exiba a alíquota de imposto aplicável para cada funcionário do departamento 80:

```
SELECT sobrenome, salario,
       DECODE (TRUNC(salario/2000, 0),
                          0, 0.00,
                          1, 0.09,
                          2, 0.20,
                          3, 0.30,
                          4, 0.40,
                          5, 0.42,
                          6, 0.44,
                             0.45) TAXA
FROM   funcionario
WHERE  cod_departamento = 80;
```

3-55

Função DECODE (continuação)

Este slide mostra outro exemplo com a função DECODE. Nesse exemplo, determinamos a alíquota de imposto de cada funcionário do departamento 80 com base no salário mensal. As alíquotas de imposto são as seguintes:

Faixa de Salário Mensal	*Alíquota de Imposto*
R$ 0,00 - 1.999,99	00%
R$ 2.000,00 - 3.999,99	09%
R$ 4.000,00 - 5.999,99	20%
R$ 6.000,00 - 7.999,99	30%
R$ 8.000,00 - 9.999,99	40%
R$ 10.000,00 - 11.999,99	42%
R$ 12.200,00 - 13.999,99	44%
R$14.000,00 ou mais	45%

SOBRENOME	SALARIO	TAXA
Schultz	10500	,42
Sue	11000	,42
Voorhees	8600	,4

Exercício 3

Exercício 3

1. Crie uma consulta para exibir a data atual. Atribua o nome a coluna de retorno `Data`.

```
Data
---------
08-MAI-09
```

2. Faça uma consulta para exibir o código do funcionário, o sobrenome, o salário e o salário com 15,5% de aumento (especificado como um número inteiro) de cada funcionário. Atribua nome `Novo Salario` à coluna. Salve o comando SQL no arquivo de texto `cap_03_02.sql` e a execute.

```
COD_FUNCIONARIO SOBRENOME      SALARIO Novo Salario
--------------- ---------- ---------- ------------
            100 Carlos          24000        27720
            101 Martins         17000        19635
            102 da Silva        17000        19635
            103 Honorato         9000        10395
            104 Osterno          6000         6930
            107 Barata           4200         4851
            124 Brunni           5800         6699
            141 Lopes            3500         4043
            142 Cabral           3100         3581
            143 Miranda          2600         3003
            144 Chaves           2500         2888
            149 Schultz         10500        12128
            174 Sue             11000        12705
            176 Voorhees         8600         9933
            178 Yamamura         7000         8085
            200 Trajano          4400         5082
            201 Thacker         13000        15015
            202 Kramer           6000         6930
            205 Almeida         12000        13860
            206 Nascimento       8300         9587

20 linhas selecionadas.
```

3. Modifique a consulta `cap_03_02.sql` para adicionar uma coluna que subtraia o salário antigo do novo salário. Atribua o nome `Aumento` à coluna. Salve o conteúdo do arquivo como `cap_03_03.sql` e execute a consulta revisada.

```
COD_FUNCIONARIO SOBRENOME      SALARIO Novo Salario    Aumento
--------------- ---------- ---------- ------------ ----------
            100 Carlos          24000        27720       3720
            101 Martins         17000        19635       2635
            102 da Silva        17000        19635       2635
            103 Honorato         9000        10395       1395
            104 Osterno          6000         6930        930
            107 Barata           4200         4851        651
            124 Brunni           5800         6699        899
            141 Lopes            3500         4043        543
            142 Cabral           3100         3581        481
            143 Miranda          2600         3003        403
            144 Chaves           2500         2888        388
            149 Schultz         10500        12128       1628
            174 Sue             11000        12705       1705
            176 Voorhees         8600         9933       1333
            178 Yamamura         7000         8085       1085
            200 Trajano          4400         5082        682
            201 Thacker         13000        15015       2015
            202 Kramer           6000         6930        930
            205 Almeida         12000        13860       1860
            206 Nascimento       8300         9587       1287

20 linhas selecionadas.
```

Exercício 3 (continuação)

4.

a) Crie uma consulta que exiba o sobrenome (com a primeira letra maiúscula e todas as outras minúsculas) e o tamanho do sobrenome de todos os funcionários cujos nomes comecem com a letra *J*, *A* ou *M*. Atribua um nome apropriado a cada coluna. Ordene resultados pelos sobrenomes dos funcionários.

Nome	Tamanho
Almeida	7
Martins	7
Miranda	7

b) Recrie a consulta para que o usuário seja solicitado a informar a letra inicial do sobrenome. Por exemplo, se o usuário informar C quando uma letra for solicitada, a saída deverá mostrar todos os funcionários cujos sobrenomes começam com a letra *H*.

Nome	Tamanho
Cabral	6
Carlos	6
Chaves	6

Exercício 3 (continuação)

5. Para cada funcionário, exiba o sobrenome e calcule o número de meses entre hoje e a data de admissão do funcionário. Atribua o nome `MESES_TRABALHADOS` à coluna. Ordene os resultados pelo número de meses em que o funcionário está empregado. Arredonde o número de meses para o número inteiro mais próximo.

Observação: Os resultados serão diferentes do mostrado abaixo.

SOBRENOME	MESES_TRABALHADOS
Barata	3
Brunni	6
Chaves	10
Miranda	14
Kramer	21
Cabral	27
Sue	36
Thacker	39
Lopes	43
Almeida	59
Nascimento	59
da Silva	76
Osterno	96
Schultz	111
Honorato	112
Martins	116
Yamamura	119
Voorhees	133
Trajano	140
Carlos	143

Exercício 3 (continuação)

6. Crie um relatório que produza estas informações para cada funcionário: `<sobrenome do funcionário> recebe <salário>` mensalmente, mas deseja `<3 vezes o salário>`. Atribua o nome `Salário dos Sonhos` à coluna.

Salário dos Sonhos
Carlos recebe $24,000.00 mensalmente, mas deseja $72,000.00.
Martins recebe $17,000.00 mensalmente, mas deseja $51,000.00.
da Silva recebe $17,000.00 mensalmente, mas deseja $51,000.00.
Honorato recebe $9,000.00 mensalmente, mas deseja $27,000.00.
Osterno recebe $6,000.00 mensalmente, mas deseja $18,000.00.
Barata recebe $4,200.00 mensalmente, mas deseja $12,600.00.
Brunni recebe $5,800.00 mensalmente, mas deseja $17,400.00.
Lopes recebe $3,500.00 mensalmente, mas deseja $10,500.00.
Cabral recebe $3,100.00 mensalmente, mas deseja $9,300.00.

7. Crie uma consulta que exiba o sobrenome e o salário de todos os funcionários. Formate o salário para defini-lo com um tamanho de 15 caracteres e preenchê-lo à esquerda com o símbolo $. Atribua o nome `SALARIO` à coluna.

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

Exercício 3 (continuação)

8. Exiba o sobrenome, a data de admissão e a data de revisão de salário de cada funcionário, que é a primeira segunda-feira após seis meses de serviço. Atribua o nome `REVISAO` à coluna. Formate as datas para que sejam exibidas no formato semelhante a "Segunda-feira, 31 de Julho, 2000".

SOBRENOME	DATA_ADMISSAO	REVISAO
Carlos	17-JUN-97	Segunda-Feira, 22 de Dezembro, 1997
Martins	21-SET-99	Segunda-Feira, 27 de Março, 2000
da Silva	13-JAN-03	Segunda-Feira, 14 de Julho, 2003
Honorato	03-JAN-00	Segunda-Feira, 10 de Julho, 2000
Osterno	21-MAI-01	Segunda-Feira, 26 de Novembro, 2001
Barata	07-FEV-09	Segunda-Feira, 10 de Agosto, 2009
Brunni	16-NOV-08	Segunda-Feira, 18 de Maio, 2009
Lopes	17-OUT-05	Segunda-Feira, 24 de Abril, 2006
Cabral	29-JAN-07	Segunda-Feira, 30 de Julho, 2007
Miranda	15-MAR-08	Segunda-Feira, 22 de Setembro, 2008

9. Exiba o sobrenome, a data de admissão e o dia da semana em que o funcionário começou a trabalhar. Atribua o nome `DIA` à coluna. Ordene os resultados pelo dia da semana, começando por segunda-feira.

SOBRENOME	DATA_ADMISSAO	DIA
Cabral	29-JAN-07	SEGUNDA-FEIRA
Nascimento	07-JUN-04	SEGUNDA-FEIRA
da Silva	13-JAN-03	SEGUNDA-FEIRA
Honorato	03-JAN-00	SEGUNDA-FEIRA
Osterno	21-MAI-01	SEGUNDA-FEIRA
Almeida	07-JUN-04	SEGUNDA-FEIRA
Yamamura	24-MAI-99	SEGUNDA-FEIRA
Lopes	17-OUT-05	SEGUNDA-FEIRA
Carlos	17-JUN-97	TERÇA-FEIRA
Martins	21-SET-99	TERÇA-FEIRA

Exercício 3 (continuação)

10. Crie uma consulta que exiba os sobrenomes e as comissões dos funcionários. Se um funcionário não ganhar comissão, a informação “Sem Comissão" deverá ser exibida. Atribua o nome `COMISSAO` à coluna.

SOBRENOME	COMISSAO
Carlos	Sem Comissão
Martins	Sem Comissão
da Silva	Sem Comissão
Honorato	Sem Comissão
Osterno	Sem Comissão
Barata	Sem Comissão
Brunni	Sem Comissão
Lopes	Sem Comissão

11. Crie uma consulta que exiba os oito primeiros caracteres dos sobrenomes dos funcionários e indique os valores dos salários com asteriscos. Cada asterisco representa mil reais. Ordene os dados em ordem decrescente de salário. Atribua o nome `FUNCIONARIO_E_SEUS_SALARIOS` à coluna.

FUNCIONARIOS_E_SEUS_SALARIOS
Carlos ************************
Martins *****************
da Silva ******************
Thacker **************
Almeida ************
Sue ***********
Schultz **********
Honorato *********
Voorhees ********
Nascimen ********
Yamamura *******
Osterno ******
Kramer ******
Brunni *****

Exercício 3 (continuação)

12. Com a função `DECODE`, crie uma consulta que exiba o nível de todos os funcionários com base no valor da coluna `COD_CARGO`. Use estes dados:

Cargo	*Nível*
AD_PRES	A
ES_GER	B
IT_PROG	C
VE_REP	D
ES_AUX	E
Nenhuma das opções anteriores	0

SOBRENOME	COD_CARGO	NIVEL
Carlos	AD_PRES	A
Martins	AD_VP	0
da Silva	AD_VP	0
Honorato	IT_PROG	C
Osterno	IT_PROG	C
Barata	IT_PROG	C
Brunni	ES_GER	B
Lopes	ES_AUX	E
Cabral	ES_AUX	E
Miranda	ES_AUX	E
Chaves	ES_AUX	E
Schultz	VE_GER	0
Sue	VE_REP	D
Voorhees	VE_REP	D
Yamamura	VE_REP	D
Trajano	AD_ASST	0
Thacker	MK_GER	0
Kramer	MK_ANA	0
Almeida	CT_GER	0
Nascimento	CTPUB_GER	0

13. Recrie o comando do exercício anterior usando a sintaxe `CASE`.