

< VOLTAR



Criação de relatórios utilizando filtros, operadores e funções em banco de dados

Criar relatórios a partir de dados existentes no Banco de dados, utilizando comandos de seleção, aplicando filtros, realizando cálculos, usando operadores e funções de banco de dados.

NESTE TÓPICO

- > Executando cálculos com datas
- > Resultado utilizando soma e subtração de datas
- > Funções de formatação
- > Concatenação de caracteres string
- > Contando linhas de uma tabela
- > Cláusula GROUP BY

Marcar
tópico



Como vimos na aula anterior, o uso das funções torna-se importante para a criação de relatórios mais detalhados e eficientes. Nesta aula vamos completar esse uso trabalhando com datas, realizando cálculos com datas e, mais importante, simplificando com a utilização de funções de grupo. Até agora as funções estão sendo aplicadas na base de dados e retornando a sua execução em todas as linhas. Com as funções de grupo, os relatórios ficarão mais simples para a leitura do resultado apresentado do que apenas usando comandos sem agrupamento. Para essa realização, vamos mudar alguns dados da tabela funcionário, que está sendo usada, e depois criaremos novas tabelas com mais colunas e mais complexas.

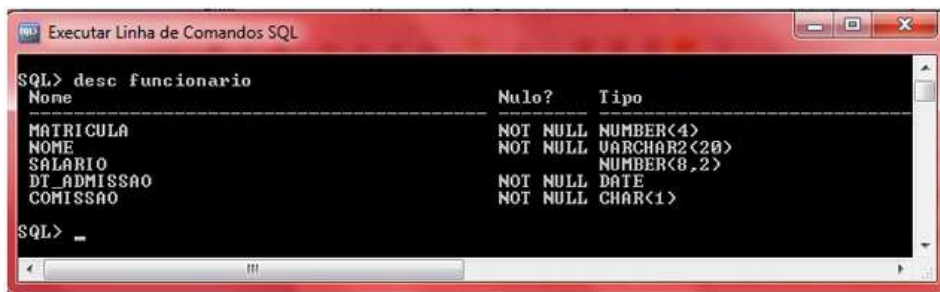


Faremos uma consulta na tabela funcionário para visualizar os novos dados e estrutura.

Visualizando a estrutura da tabela funcionário:

1. Desc funcionario





1. Select * from funcionario;



Agora imagine que precisamos trabalhar a data dos funcionários, realizar cálculos, projetar uma simulação de data para entrega de um projeto, vencimento de um contrato de trabalho, cheques pré-datados, tempo de trabalho para bonificação do salário etc.

Executando cálculos com datas

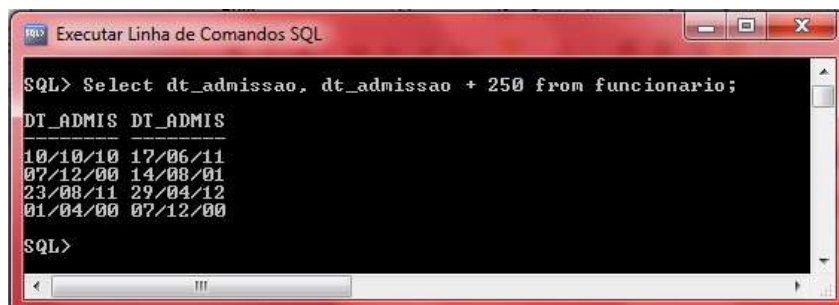
Podemos utilizar os operadores aritméticos também nas datas.

Expressão: Data + número – soma um número a uma data e devolve uma data.

Sintaxe: data + número

Exemplo:

1. Select dt_admissao, dt_admissao + 250 from funcionario;

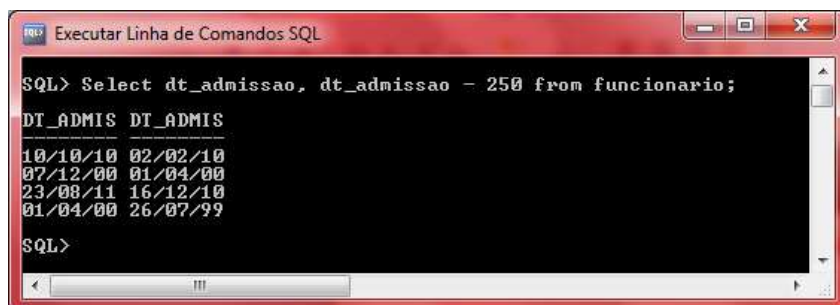


Expressão: Data - número – subtrai um número da data e devolve uma data.

Sintaxe: data - número

Exemplo:

1. `Select dt_admissao, dt_admissao - 250 from funcionario;`



Resultado utilizando soma e subtração de datas

Expressão: Data - data – subtrai uma data de outra e retorna o número de dias entre uma data e outra.

Sintaxe: data1 - data2

Exemplo:

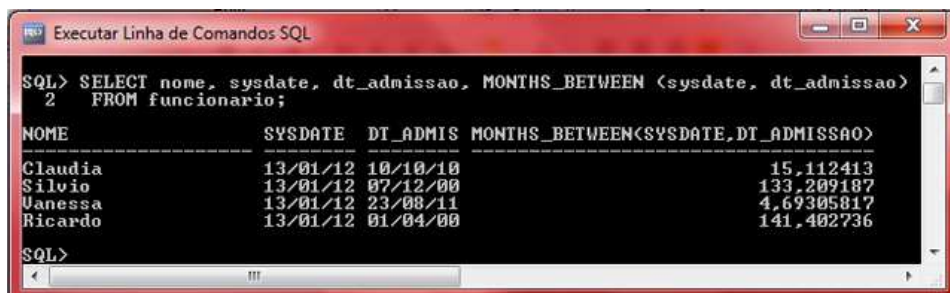
1. `SELECT nome, sysdate, dt_admissao, sysdate - dt_admissao FROM funcionario WHERE matricula < 4;`

Obs.: SYSDATE – retorna a data do sistema, data atual.

Expressão: MONTHS_BETWEEN – retorna o número de meses entre duas datas.

Sintaxe: MONTHS_BETWEEN(data1, data2)

1. `SELECT nome, sysdate, dt_admissao, MONTHS_BETWEEN (sysdate, dt_admissao) FROM funcionario;`



Expressão: NEXT_DAY – retorna o próximo dia da semana de acordo com uma data.

Sintaxe: NEXT_DAY(data1, 'dia da semana')

1. SELECT dt_admissao, NEXT_DAY (dt_admissao, 'segunda') FROM funcionario;



```

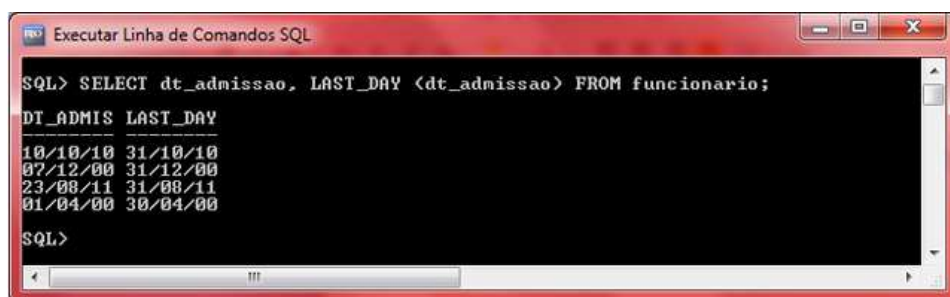
SQL> SELECT dt_admissao, NEXT_DAY (dt_admissao, 'segunda')
2 FROM funcionario;

DT_ADMIS NEXT_DAY
-----
10/10/10 11/10/10
07/12/00 11/12/00
23/08/11 29/08/11
01/04/00 03/04/00
SQL>
  
```

Expressão: LAST_DAY – retorna o último dia do mês corrente.

Sintaxe: LAST_DAY(data)

1. SELECT dt_admissao, LAST_DAY (dt_admissao) FROM funcionario;



```

SQL> SELECT dt_admissao, LAST_DAY (dt_admissao) FROM funcionario;

DT_ADMIS LAST_DAY
-----
10/10/10 31/10/10
07/12/00 31/12/00
23/08/11 31/08/11
01/04/00 30/04/00
SQL>
  
```

Funções de formatação

Para modificar o formato de datas, devemos utilizar a função TO_CHAR.

Sintaxe: TO_CHAR (coluna tipo data, máscara).

Máscaras:

Formato	Descrição
yyy ou yy	Três ou dois dígitos do ano
Mm, month	Número do mês, nome do mês
dd	Dia do mês
day	Nome do dia da semana
hh24	Formato da hora
mi	Formato minutos
ss	Formato segundos

Exemplos:

1. `SELECT nome, TO_CHAR(dt_admissao, 'MM/YYYY') Admissao FROM funcionario WHERE salario > 4500;`

```

SQL> SELECT nome, TO_CHAR(dt_admissao, 'MM/YYYY') Admissao
2 FROM funcionario WHERE salario > 4500;

NOME                ADMISSA
-----
Claudia             10/2010
Silvio              12/2000
Vanessa             08/2011
Ricardo             04/2000
SQL>

```

1. `SELECT nome, TO_CHAR(dt_admissao, 'fmDD MONTH YYYY') FROM funcionario WHERE matricula >= 2;`

```

SQL> SELECT nome, TO_CHAR(dt_admissao, 'fmDD MONTH YYYY')
2 FROM funcionario WHERE matricula >= 2;

NOME                TO_CHAR(DT_ADMISSAO, 'FMDDMONTHYYYY')
-----
Silvio              7 DEZEMBRO 2000
Vanessa            23 AGOSTO 2011
Ricardo             1 ABRIL 2000
SQL> _

```

Concatenação de caracteres string

Para concatenar caracteres string utilizamos o símbolo ||.

Exemplo:

1. `SELECT matricula || ' - ' || nome FROM funcionario`

```

SQL> SELECT matricula || ' - ' || nome FROM funcionario;

MATRICULA||'-'||NOME
-----
1 - Claudia
2 - Silvio
3 - Vanessa
4 - Ricardo
SQL> _

```

Funções de agrupamento: média (AVG), valor máximo (MAX), valor mínimo (MIN) e soma (SUM)

Sintaxe: AVG(nome da coluna), MAX(nome da coluna), MIN(nome da coluna), SUM (nome da coluna)

Exemplo:

1. `SELECT AVG(salario) Media, MAX(salario) Maior, MIN(salario) Menor, SUM (salario) Somatoria FROM funcionario;`



Contando linhas de uma tabela

A função de agrupamento que retorna o número de linhas existentes em uma tabela é o COUNT. Essa função pode ser usada com o nome da coluna ser realizado a contagem, COUNT(nome da coluna) ou COUNT(*), que processa a contagem em todas as colunas e retorna o maior valor.

Exemplo:

1. `SELECT count(nome) from funcionario;`



Cláusula GROUP BY

Quando desejamos obter resultados agrupados, o SQL nos fornece a cláusula GROUP BY, que permite que os valores singulares possam ser combinados com valores agrupados na cláusula SELECT.

Sintaxe:

`SELECT nome da coluna singular 1 [, nome da coluna singular2, ...], função de agrupamento [, função de agrupamento...] FROM nome da tabela WHERE condição (opcional) GROUP BY nome da coluna singular 1 [, nome da coluna singular2, ...]`

Exemplo: A partir de agora, usaremos outras tabelas; neste caso a tabela cliente segue um relatório de seus dados.



```

Executar Linha de Comandos SQL

SQL> desc cliente
None
-----
COD_CLIIE      Nulo?      Tipo
NOME_CLIIE     NOT NULL   VARCHAR2(20)
ENDERECO       NOT NULL   VARCHAR2(30)
CIDADE         VARCHAR2(15)
CEP            CHAR(8)
UF             CHAR(2)
CNPJ           CHAR(14)
IE             CHAR(12)

SQL>

```

```

Executar Linha de Comandos SQL

COD_CLIIE NOME_CLIIE      ENDERECO      CIDADE      CEP      UF      CNPJ      IE
-----
728 Ana      Rua 17 n.19    Niteroi      24358310    RJ      121132310000134 2134
878 Flavio   Av. Pres. Vargas, 10  Sao Paulo    22763931    SP      2253412693879 4631
118 Jorge    Rua Caiapo, 13   Curitiba     30078500    PR      1451276498349 2985
222 Lucia    Rua Itabira, 123  Belo Horizonte 22124391    MG      28315212393488 2985
338 Mauricio Av. Paulista, 1236 Sao Paulo     3812683     SP      3281698574656 2343
138 Edmar    Rua da Praia, s/n Salvador      30879100    BA      234632842349 7121
418 Rodolfo Largo da Lapa, 27 Rio de Janeiro 30878900    RJ      1283512823469 743
28 Beth      Rua Clinerio, 45   Sao Paulo     25679300    SP      3248512673268 9288
157 Paulo    Trav. Moraes, casa 3 Londrina      328482233242 1923
188 Livio    Av. Beira Mar, 1256 Florianopolis 30877500    SC      1273657123474 1111
268 Susana   Rua Lopes Mandes, 12 Niteroi       30846500    RJ      217635712329 2538
298 Renato   Rua Meireles, 123  Sao Paulo     30225900    SP      1327657112314 1820
398 Sebastiao Rua da Igreja, 18 Uberaba       38438700    MG      321765472133 9871
234 Jose     Quadra 3, Bl. 3, sl. 1003 Brasilia      22841650    DF      2176357612323 2931

14 linhas selecionadas.

SQL>

```

Imagine que alguém gostaria de saber quantos clientes moram no estado de São Paulo. Você não iria contar manualmente, iria? Então podemos elaborar um relatório com contador e função de grupo que fariam a contagem para nós.

1. `SELECT uf, COUNT(uf) Total from cliente where UPPER(uf) = 'SP'`

```

Executar Linha de Comandos SQL

SQL> SELECT uf, COUNT(uf) Total from cliente where UPPER(uf) = 'SP'
2 GROUP BY uf;

UF      TOTAL
-----
SP              4

SQL>

```

1. `GROUP BY uf;`

Criando um relatório que mostre quantos clientes existem por estado:

1. `SELECT uf, COUNT(uf) Total from cliente GROUP BY uf;`



```
SQL> SELECT uf, COUNT(uf) Total from cliente GROUP BY uf;
```

UF	TOTAL
RJ	3
MG	2
SP	4
PR	2
BA	1
SC	1
DF	1

7 linhas selecionadas.

```
SQL>
```

Cláusula HAVING

Esta cláusula restringe linhas mostradas pela cláusula GROUP BY. Podemos comparar analogamente a cláusula HAVING com a cláusula WHERE do SELECT.

Sintaxe:

SELECT nome da coluna singular 1 [, nome da coluna singular2, ...], função de agrupamento [, função de agrupamento...] FROM nome da tabela WHERE condição (opcional) GROUP BY nome da coluna singular 1 [, nome da coluna singular2, ...]

Exemplo:

1. SELECT uf, COUNT(uf) Total from cliente GROUP BY uf HAVING COUNT(uf) > 2;



```
SQL> SELECT uf, COUNT(uf) Total from cliente
2 GROUP BY uf HAVING COUNT(uf) > 2;
```

UF	TOTAL
RJ	3
SP	4

```
SQL>
```

Esta instrução limita a exibição quando o valor de contagem for maior que 2.

Referências

BEIGHLEY, Lynn. *Use a cabeça SQL*. Rio de Janeiro: AltaBooks.

FANDERUFF, Damaris. *Dominando o Oracle 9i: modelagem e desenvolvimento*. São Paulo: Makron.

GRAVES, Mark. *Projeto de banco de dados com XML*. 1. ed. São Paulo: Pearson, 2003.

MORELLI, Eduardo Terra. *Oracle 9i fundamental: SQL, PL/SQL e administração*, São Paulo: Érica.

PRICE, Jason. *Oracle Database 11g SQL*/Jason Price. Tradução: João Eduardo Nóbrega Tortello. Porto Alegre: Bookman, 2009.

SILVA, Robson S. *Oracle Database 10g Express Edition*. 1. ed. São Paulo: Érica, 2007.



Avalie este tópico



ANTERIOR

Criação de relatórios utilizando filtros, operadores e funções em banco de dados

Biblioteca
(https://www.uninove.br/conheca-
a-
uninove/biblioteca/sobre-
a-
biblioteca/apresentacao/)
Portal Uninove
(http://www.uninove.br)
Mapa do Site



Índice

Ajuda?
PRÓXIMO
(https://ava.un
inove.br/ava-
ta/)



Criação de relatórios utilizando mais de uma tabela

© Todos os direitos reservados

