

[◀ VOLTAR](#)

Criação de relatórios utilizando filtros, operadores relacionais e operadores lógicos

Criar relatórios a partir de dados existentes no Banco de dados utilizando comandos de seleção e aplicando filtros ou condições.

NESTE TÓPICO



Marcar
tópico



Antes de iniciarmos esta aula, devemos relembrar o uso dos operadores existentes em banco de dados: operadores relacionais, operadores lógicos e operadores aritméticos. Após esta breve revisão, podemos dar início às instruções que geram relatórios, utilizando condições tais como aquelas que são aplicadas nos comandos de atualização e eliminação de dados. É necessário relembrar a estrutura do comando que cria relatórios ou consultas em uma base de dados e, por fim, mostrar dados com o comando SELECT.

A função do comando SELECT é buscar e mostrar dados armazenados, em forma de tabelas, dentro do Banco de Dados.

Sintaxe:

1. `SELECT coluna (s) FROM nome da tabela WHERE condições ORDER BY expresssão`

Onde:

SELECT - especifica qual (is) coluna (s) será (ao) lida (s).

FROM - especifica qual (is) tabela(s) possuem estas colunas.

WHERE - especifica critérios de seleção de linhas (opcional), se não colocada mostrará todos os dados da tabela.

ORDER BY - ordem que os dados devem ser mostrados. (opcional).

Mostrando todos os dados de uma tabela:

Exemplo:

1. `SELECT * FROM nome da tabela`

Exemplo:

1. `SELECT * FROM funcionario;`



```
SQL> select * from funcionario;
```

MATRICULA	NOME	SALARIO	DT_ADMIS	C
1	Claudia	50	10/10/10	A
2	Silvio	50	07/12/00	B
3	Vanessa	50	23/08/11	A
4	Ricardo	50	01/04/00	C

4 linhas selecionadas.

```
SQL>
```

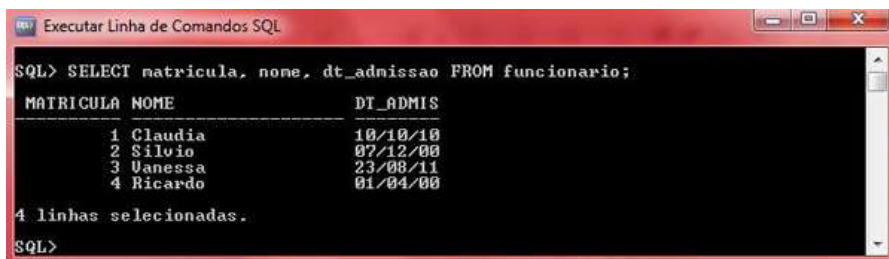
Para selecionar colunas em específico:

Sintaxe:

1. `SELECT nome da coluna1, nome da coluna 2, ... FROM nome da tabela`

Exemplo:

1. `SELECT matricula, nome, dt_admissao FROM funcionario;`



```
SQL> SELECT matricula, nome, dt_admissao FROM funcionario;
```

MATRICULA	NOME	DT_ADMIS
1	Claudia	10/10/10
2	Silvio	07/12/00
3	Vanessa	23/08/11
4	Ricardo	01/04/00

4 linhas selecionadas.

```
SQL>
```

A ordenação de um resultado não influencia na ordem de gravação dos dados.

Pode-se definir dois sentidos de ordenação: ascendente (ASC) e descendente (DESC).

A cláusula utilizada para ordenação é ORDER BY.

Sintaxe:

1. `SELECT { * [nome da coluna] } FROM nome da tabela WHERE condição (opcional) ORDER BY nome da coluna ASC/DESC`

Exemplo:

1. `SELECT nome, dt_admissao FROM funcionario WHERE dt_admissao = '10/11/2011' ORDER BY nome;`

Exemplo:

1. `SELECT nome, dt_admissao FROM funcionario WHERE dt_admissao = '10/11/2011'`
2. `ORDER BY nome DESC;`

Obs.1: Note que apenas o operador descendente deve ser digitado.

Obs.2: Não é necessária a coluna utilizada no ORDER BY estar no SELECT.

Clausula WHERE

O filtro ou condição WHERE, serve para seleção linhas em uma tabela de acordo com a condição estabelecida.

Sintaxe:

1. `SELECT { * [nome da coluna] } FROM nome da tabela WHERE condição?`

Esta condição pode ser, por exemplo, `matricula = 1` ou `dt_admissao <> '10/10/2010'`

A seguir exemplos cada condição:

Igual (=) >> Retorna as linhas onde os valores de uma coluna são iguais aos estabelecidos.

Exemplo:

`SELECT matricula, nome from funcionario WHERE matricula = 1;`



Para comparações com valores em caracteres é necessário tomar o cuidado de descrever o valor tal como foi armazenado, além de colocar este valor entre 'aspas simples'.

Quando a comparação é em colunas que armazenam datas devemos respeitar a máscara default do ORACLE que é DD/MM/YYYY. (dia – mês – ano)

Exemplo com data:

1. `SELECT dt_admissao from funcionario where dt_admissao = '10/10/2010';`



Os valores da data também devem estar entre 'aspas simples'.

Diferente de (<> ou !=) >> Retorna as linhas com valores diferentes do valor estipulado na condição WHERE.

Exemplo:

1. `SELECT nome, salario from funcionario WHERE salario <> 5000;`

Maior que (>) >> Retorna linhas que possuam valores maiores que o estipulado na coluna envolvida na condição WHERE.

Exemplo:

1. `SELECT nome, matricula, salario FROM funcionario WHERE matricula > 2;`

É possível comparar data e caracteres também.

Maior ou igual a (>=) >> Retorna linhas que possuam valores maiores ou iguais ao determinado.

Exemplo:

1. `SELECT matricula, salario from funcionario WHERE dt_admissao >= '10/07/2007';`

Menor que (<) >> Retorna linhas que possuam valores menores que o estabelecido.

Exemplo:

1. `SELECT matricula,nome from funcionario WHERE dt_admissao < '01/12/2000';`

Menor igual a (<=) >> Retorna linhas que possuam valores, menores ou iguais ao valor de comparação.

Exemplo:

1. `SELECT nome, dt_admissao, salario from funcionario WHERE nome <= 'Ricardo';`



Valores nulos (IS NULL)

As colunas que se encontram vazias, para o banco de dados, possuem valores chamados NULL. Este valor possui características específicas que geram a necessidade de comando particular para estes casos.

Para a visualização de linhas onde uma determinada coluna está vazia devemos utilizar o IS NULL.

Exemplo:

1. `SELECT * FROM funcionario WHERE dt_admissao IS NULL;`

O exemplo acima mostrará todos os funcionários que não possuem data de admissão cadastrada.

Valores diferentes de NULL (IS NOT NULL)

Da mesma forma para visualizar só as linhas que possuem dados (quaisquer) em uma coluna.

Exemplo:

1. `SELECT * FROM funcionario WHERE dt_admissao IS NULL;`

1. `SELECT * FROM funcionario WHERE salario IS NOT NULL;`

Complexidade nas condições:

A cláusula WHERE pode ter mais que uma condição no mesmo comando. Para isto devem ser usados os operadores lógicos AND e OR onde:

AND - Mostra a linha se todas as condições usadas no AND são verdadeiras.

OR - Mostra a linha se uma das condições for verdadeira.

Exemplo:

```
1. SELECT nome, salario FROM funcionario WHERE matricula >= 2 AND salario > 5000;
```

Apenas os funcionários de matrícula maior ou igual a 2 e salário superior a R\$ 5.000,00 serão apresentados.

Exemplo:

```
1. SELECT * FROM funcionario WHERE dt_admissao = '10/10/2010' OR nome > 'Ricardo';
```

Apenas os funcionários que possuem data de admissão igual 10/10/2010 ou que o nome é maior que Ricardo serão listados.

A combinação de duas ou mais condições de busca obedecem a uma regra de precedentes. O resultado da pesquisa depende desta ordem.

Para quebrar esta precedência, basta utilizar parênteses.

Exemplo:

```
1. SELECT nome, salario, matricula FROM funcionario WHERE salario >= 1000 AND (matricula = 4 or matricula = 2);
```

Neste caso apenas funcionários de matrícula 4 e 2, com salário maior ou igual a 1000, serão listados.

Referências

BEIGHLEY, Lynn. *Use a Cabeça SQL*. Rio de Janeiro: Alta Books, 2008.

FANDERUFF, Damaris. *Dominando o Oracle 9i: Modelagem e Desenvolvimento*, São Paulo: Makron, 2003.

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

MORELLI, Eduardo Terra. *Oracle 9i Fundamental: SQL, PL/SQL e Administração*, São Paulo, Editora Érica, 2002.

PRICE, Jason. *Oracle Database 11g SQL*. (tradução: João Eduardo Nóbrega Tortello). Porto Alegre: Bookman, 2009.

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



Avalie este tópico



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

Criação de relatórios utilizando filtros, operadores aritméticos e de banco de dados