CONSULTA COM FILTRAGEM DE LINHAS
ORDENAÇÃO DE RESULTADOS
CLÁUSULA DISTINCT
FUNÇÕES UPPER E LOWER
APELIDOS PARA TABELAS E ATRIBUTOS

CONSULTA COM FILTRAGEM DE LINHAS

→ Selecionando apenas algumas linhas/registros de uma tabela

A cláusula WHERE aplicada em um comando SELECT estabelece quais linhas/registro de uma tabela deseja-se obter.

Sintaxe básica:

```
SELECT <lista_de_colunas>
FROM <nome_tabela>
WHERE <condição_de_seleção>;
```

Onde <condição_de_seleção>, pode ser:

```
WHERE <nome_do_atributo> <operador> <valor>;
```

Operadores Relacionais:

=	Igual
<	menor que
>=	maior ou igual a
<>	ou != diferente
>	maior que
<=	menor ou igual a

Observação: A linguagem SQL diferencia caracteres maiúsculos de minúsculos. Assim, um atributo do tipo **caracter** pede que o <valor> esteja entre aspas simples ('). Por exemplo: 'gato' é diferente de 'GATO' e, diferente, de 'Gato'.

Exemplo:

1) Quais são os cursos que têm mensalidade superior à R$ 800,00?

```
SELECT nm_curso
FROM curso
WHERE mensalidade > 800;
```

2) Quais são os cursos que têm turno igual à 'D' (diurno)?

```
SELECT nm_curso
FROM curso
WHERE turno = 'D';
```

Operadores Lógicos:

AND	"e" lógico
OR	"ou" lógico
NOT	Negação

Exemplos:

1) Quais são os cursos que têm mensalidade entre R$ 800,00 e R$1.500,00?

```
SELECT nm_curso
FROM curso
WHERE mensalidade >= 800 AND mensalidade <= 1500;
```

2) Quais cursos que NÃO possuem mensalidade igual à R$ 1.000,00?

```
SELECT nm_curso
FROM curso
WHERE NOT (mensalidade > 1000);
```

ou

```
SELECT nm_curso
FROM curso
WHERE mensalidade != 1000;
```

Operador [NOT] BETWEEN:

Com o uso deste operador, é possível estabelecer a seleção de uma faixa de valores sem o uso dos operadores >=, <= e do AND, sendo que <valor1> e <valor2> devem possuir o mesmo tipo de dado que <nome_atributo>.

```
WHERE <nome_atributo> BETWEEN <valor1> AND <valor2>
WHERE <nome_atributo> NOT BETWEEN <valor1> AND <valor2>
```

Exemplo:

Quais são os cursos que têm mensalidade entre R$ 800,00 e R$1.500,00?

```
SELECT nm_curso
FROM curso
WHERE mensalidade BETWEEN 800 AND 1500;
```

Operador IS [NOT] NULL:

O uso desse operador permite a busca por valores *nulos* (NULL) ou de valores *não nulos* (NOT NULL) em atributos de uma tabela, selecionado as linhas correspondentes.

```
WHERE <nome_atributo> IS NULL
WHERE <nome_atributo> IS NOT NULL
```

Exemplos:

1) Quais alunos não possuem responsável cadastrado no BD?

```
SELECT nome
FROM aluno
WHERE responsavel IS NULL;
```

Quais os alunos possuem responsável cadastrado no BD?

```
SELECT nome
FROM aluno
WHERE responsavel IS NOT NULL;
```

Operador [NOT] LIKE:

O operador [NOT] LIKE somente pode ser utilizado em atributos do tipo caractere.

```
WHERE <nome_coluna> LIKE <valor>
WHERE <nome_atributo> NOT LIKE <valor>
```

Observação: O operador funciona de forma semelhante aos operadores = e <>. No entanto, o LIKE e o NOT LIKE permitem a utilização dos símbolos **%** e **_**, que desempenham papel de "coringa":

- **%** Substitui uma palavra.
- **_** Substitui um caractere qualquer.

Exemplos:

1)
```
SELECT *
FROM aluno
WHERE nome LIKE 'Ana%';
```

Essa consulta teria como resultado todos os nomes de disciplinas que iniciam com as letras especificadas na cadeia de caracteres entre aspas após o LIKE. O resultado poderia ser:

- ✓ 'Ana Paula'
- ✓ 'Ana Maria'
- ✓ 'Ana Carolina'

2)
```
SELECT *
FROM aluno
WHERE nome LIKE '%n_';
```

Essa consulta teria como resultado o nome dos alunos que após uma letra "n" de seu nome, existe apenas mais um caractere:

- ✓ 'Ana'

3) Listar os dados dos alunos cujos os nome NÃO comecem com a letra 'A'.

```
SELECT *
FROM aluno
WHERE nome NOT LIKE 'A%';
```

4) Listar os dados dos alunos cujos os nome terminem com a letra 'A'.

```
SELECT *
FROM aluno
WHERE nome NOT LIKE '%a';
```

5) Listar os dados dos alunos cujos os nome possuam a letra 'A' em sua composição.

```
SELECT *
FROM aluno
WHERE nome NOT LIKE '%a%';
```

Operador [NOT] IN:

Lista registros cujo valor do atributo informado (<nome_atributo>) esteja contido na <lista_de_valores>.

```
WHERE <nome_atributo> IN <lista_de_valores>
WHERE <nome_atributo> NOT IN (<lista_de_valores>)
```

Exemplo:

Listar os cursos que possuem mensalidade igual a R$ 300,00 e R$ 500,00.

```
SELECT *
FROM curso
WHERE mensalidade IN ('300','500');
```

ORDENAÇÃO DE RESULTADOS

Cláusula ORDER BY.

Ordena a listagem de acordo com o(s) atributo(s) definido(s) na cláusula. A ordenação pode ser ascedente (ASC) ou descendente (DESC). Caso não seja informada a cláusula ASC ou DESC, por padrão é assumida a primeira (ASC).

Sintaxe:

```
SELECT <lista_de_atributos>
FROM <nome_tabela>
[WHERE <condição_de_seleção>]
ORDER BY {<nome_atributo1>[, <nome_atributo2>, ...][ASC|DESC]}
```

Exemplo:

1) Mostrar em ordem alfabética o nome dos alunos e suas respectivas datas de nascimento.

```
SELECT nome, dt_nascimento
FROM aluno
ORDER BY nome;
```

2) Listar o nome dos alunos, do mais velho ao mais jovem.

```
SELECT nome
FROM aluno
ORDER BY dt_nascimento ASC;
```

3) Listar o nome dos alunos que possuem responsável cadastrado e suas respectivas datas de nascimento. Ordenar de maneira crescente por data de nascimento, seguida da ordenação alfabética dos nomes.

```
SELECT dt_nascimento, nome
FROM aluno
WHERE responsavel IS NOT NULL
ORDER BY dt_nascimento, nome ASC;
```

4) Listar os dados dos alunos que nasceram entre os anos de 2000 e 2005. Os resultados devem ser organizados por ordem crescente de data.

```
SELECT *
FROM aluno
WHERE dt_nascimento IN ('01/01/2000','31/12/2005')
ORDER BY dt_nascimento;
```

CLÁUSULA DISTINCT

Com exceção da chave primária, o valor de um registro pode aparecer em várias linhas de uma tabela podem possuir os mesmos valores. Para contornar essa situação e eliminar redundância de valores em uma consulta, pode-se utilizar a cláusula DISTINCT logo após a palavra SELECT.

Exemplo:
Quais os turnos dos cursos oferecidos na escola?

```
SELECT DISTINCT turno
FROM curso;
```

FUNÇÃO UPPER

Retorna caracteres em MAIÚSCULO.

Sintaxe:

```
UPPER (nome_atributo);
```

Exemplos:

```
SELECT UPPER (nome)
FROM aluno;

SELECT *
FROM aluno
WHERE UPPER (nome) NOT LIKE 'A%';
```

FUNÇÃO LOWER

Retorna caracteres em minúsculo.

Sintaxe:

```
LOWER (nome_atributo);
```

Exemplo:

```
SELECT LOWER (nome)
FROM aluno;

SELECT *
FROM aluno
WHERE LOWER (nome) NOT LIKE 'a%';
```

APELIDOS PARA TABELAS E ATRIBUTOS

Pode ser interessante dar "apelidos" para uma tabela ou atributo. Para isso, utiliza-se o ALIASES. Para tabelas, a definição dos ALIASES é feita na cláusula FROM e, para o atributo, ao lado de seu nome.

Exemplos:

1) Aliases aplicado ao nome da tabela:

```
SELECT nome
FROM aluno AS a;
```

2) Aliases aplicado ao nome do atributo:

```
SELECT nome AS "Nome do aluno"
FROM aluno;
```

Observação: O uso da cláusula **AS** é opcional, pode ser omitida. Como no exemplo a seguir:

```
SELECT nome "Nome do aluno"
FROM aluno;
```

REFERÊNCIAS

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson, 2018.

MACHADO, Felipe; ABREU, Maurício. **Projeto de Banco de Dados**: uma visão prática. São Paulo: Ética, 2009.