



## Cursos

Listagem de disciplinas

Selecione uma disciplina

## Aulas

- 01 Introdução a Banco de Dados
- 02 Modelo de Entidade e Relacionamento
- 03 Modelo Relacional
- 04 Transformações ER para MR
- 05 Transformações ER para MR e dicionário de dados
- 06 Normalização básica
- 07 Normalização avançada
- 08 Introdução à Linguagem SQL e Sistemas Gerenciadores de Banco de Dados
- 09 Linguagem SQL - criação, inserção e modificação de tabelas
- 10 Linguagem SQL - Consulta simples de tabelas
- 11 Linguagem SQL - Consulta avançada de tabelas
- 12 Linguagem SQL - Alteração da estrutura de tabelas e ambientes de múltiplas tabelas
- 13 Linguagem SQL - Subconsultas
- 14 Linguagem SQL - VISÕES
- 15 Linguagem SQL - STORED PROCEDURES
- 16 Linguagem SQL - Funções
- 17 Linguagem SQL - Segurança
- 18 Engenharia Reversa
- 19 Utilizando SQL em Java
- 20 Utilizando conceitos avançados de SQL em Java

Voltar Imprimir Topo



### Sistemas de Banco de Dados


#### Aula 10 – Linguagem SQL - Consulta simples de tabelas

**Professores autores**  
José Josemar de Oliveira Júnior (josemar@ect.ufrr.br)  
Luciana Ribeiro Veloso (luciana.veloso@globlo.com)



## Apresentacao


Na aula anterior, vimos os comandos CREATE DATABASE e USE que são utilizados para criar e para utilizar efetivamente um banco de dados. A seguir, foram estudados os comandos CREATE TABLE, que cria a estrutura de uma tabela, e o INSERT INTO, que é utilizado para preencher a tabela com os dados. Estudamos, também, como fazemos atualizações e apagamos linhas nas tabelas de um banco de dados através dos comandos UPDATE e DELETE FROM, respectivamente. E finalizamos nossa aula com o comando DROP TABLE, que exclui tanto os dados como a estrutura de uma tabela. Agora, você deve estar se perguntando como podemos fazer para visualizar a estrutura e os dados de uma tabela, para conferir, por exemplo, qual é o tipo de um determinado atributo ou se, realmente, a tabela foi criada corretamente ou ainda fazer uma pesquisa sobre um determinado registro em uma tabela. No primeiro momento desta aula, vamos aprender como visualizar a estrutura de uma tabela e depois estudaremos como realizar pesquisas simples em uma tabela utilizando o famoso comando SELECT.



## Objetivo

**Ao final desta aula, você será capaz de:**

- executar comandos para visualizar a estrutura de uma tabela;
- consultar dados em tabelas;
- entender e aplicar a cláusula WHERE.



### Visualização da estrutura de uma tabela

Na aula anterior aprendemos a criar e inserir dados em tabelas. Mas, como podemos verificar se as tabelas criadas estão realmente como desejávamos? A resposta está no comando DESC, que descreve a estrutura de uma tabela. Ele possui a seguinte sintaxe:

```
mysql> DESC nome_da_tabela;
```

Ao executar esse comando corretamente, você obterá como resposta do sistema a estrutura da tabela, que informa quais são seus campos ou atributos (*Field*) e os seus respectivos tipos (*Type*). Além das informações sobre as restrições de cada atributo, tais como: valor padrão (*Default*), se é uma chave (*Key*) ou não, e se o atributo aceita valores nulos (*Null*). Não se preocupe com essas informações adicionais, falaremos delas em breve.

Agora vamos praticar esse comando verificando a estrutura da tabela **clientes** do banco de dados da nossa **locadora**. Não se esqueça que antes de começar é preciso dizer ao *software* do SGBD qual o banco de dados que você quer trabalhar, utilizando o comando USE. O comando para visualizar a estrutura da tabela **clientes** é exemplificado no quadro abaixo.

```
mysql>DESC clientes;
```

A resposta do SGBD, no caso do MySQL, ao comando DESC **clientes** é ilustrada na Figura 1. Ela fornece a informação, por exemplo, que a tabela **clientes** possui um atributo denominado de cli\_nome do tipo VARCHAR com no máximo 30 caracteres, e que esse atributo aceita um valor nulo e o seu valor padrão (caso nenhum outro lhe seja atribuído) é nulo.

Field	Type	Null	Key	Default	Extra
cli_codigo	int(11)	YES		NULL	
cli_nome	varchar(30)	YES		NULL	
cli_cpf	char(12)	YES		NULL	
cli_data_nasc	date	YES		NULL	
cli_sexo	char(1)	YES		NULL	
cli_email	varchar(50)	YES		NULL	

6 rows in set (0.01 sec)

**Figura 1** – Tela do MySQL após o comando DESC **clientes**.  
Fonte: MySQL Server 5.1

Além do comando DESC, os comandos SHOW DATABASES e SHOW TABLES também podem ser usados para visualização das estruturas de um banco de dados. A diferença é que o SHOW DATABASES mostra os bancos de dados que estão cadastrados no seu diretório e SHOW TABLES mostra as tabelas que foram criadas no banco de dados em uso.

### Pratique 01

Vamos praticar um pouco para que você se familiarize com o comando de visualização de estruturas das tabelas. Entre no banco de dados da sua lanchonete preferida (Atividade 2 da Aula 10) e visualize as estruturas das tabelas que contêm as informações sobre os funcionários e sobre o estoque dos produtos. Caso alguma tabela tenha algum problema, apague-a (eliminando os dados e excluindo sua estrutura), a seguir recrie-a e insira alguns dados. Cheque novamente a estrutura criada. Se tiver dúvidas, consulte o material da Aula 10, que mostrou, dentre outras coisas, como criar e apagar tabelas.

## Consulta simples em tabelas

Vamos supor que precisamos obter o email do cliente Sr. José da Silva para lhe enviar uma correspondência. Não podemos usar o comando DESC, porque precisamos ter acesso visual aos dados inseridos na tabela e não a sua estrutura. Para isso, utilizamos o comando SELECT, que permite visualizar, consultar, pesquisar ou selecionar os dados de uma tabela.

A sintaxe do comando SELECT é descrita no quadro a seguir.

```
mysql> SELECT atributo1, atributo2, ...
      FROM nome_da_tabela1, nome_da_tabela2, ...
      WHERE condição;
```

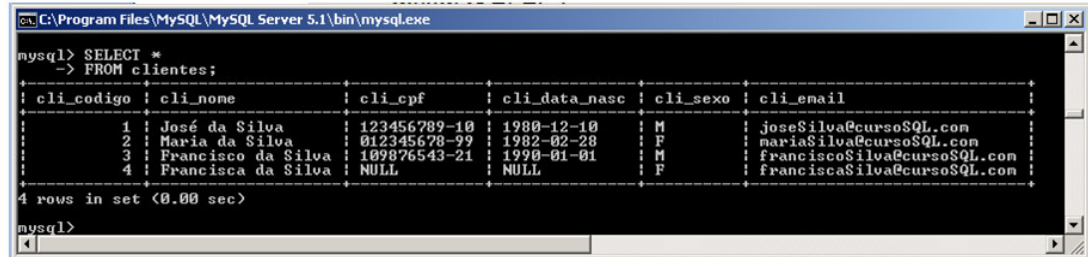
A expressão de consulta SELECT é composta por três cláusulas: SELECT – FROM – WHERE. Na cláusula SELECT, os valores representados por atributo1, atributo2, ... compõem a lista de atributos que você deseja consultar. Quando se deseja consultar todos os atributos de uma tabela, ao invés de informar toda a lista de atributos, podemos substituir por um "\*" (asterisco ou, como é mais conhecido pela galera do SQL, estrela). A cláusula FROM informa de quais tabelas os dados serão recuperados. E por fim tem-se a cláusula WHERE, que é opcional, mas quando presente especifica quais as condições que um determinado registro deve satisfazer para qualificar a recuperação. Ela limita os resultados, exibindo somente os registros que são compatíveis com a condição estabelecida.

Vamos exercitar a utilização do comando SELECT fazendo algumas consultas aos dados da tabela **clientes** e **filmes** do nosso banco de dados **locadora**.

Inicialmente, vamos realizar uma simples consulta a todos os dados da tabela **clientes**, através do seguinte comando:

```
mysql> SELECT *
FROM clientes;
```

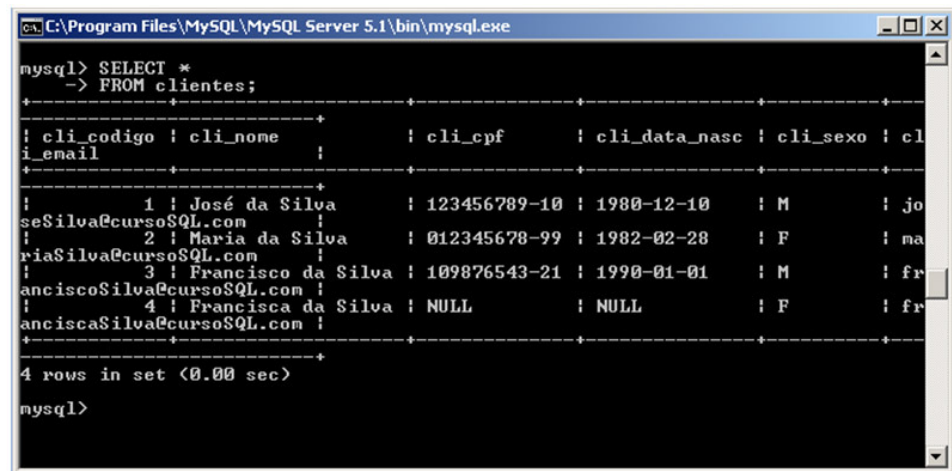
A resposta fornecida pelo sistema à consulta acima é ilustrada na Figura 2. Observe que todos os atributos inseridos para todos os registros foram visualizados. Os atributos `cli_cpf` e `cli_data_nasc` contêm o valor NULL para o registro de Francisca da Silva. Esses valores NULLs foram inseridos pelo SGBD, devido à ausência de valores informados para esses campos durante a inclusão desse registro na tabela. Sendo assim, o SGBD atribui a esses campos o valor padrão (default) NULL.



cli_codigo	cli_nome	cli_cpf	cli_data_nasc	cli_sexo	cli_email
1	José da Silva	123456789-10	1980-12-10	M	joseSilva@cursoSQL.com
2	Maria da Silva	012345678-99	1982-02-28	F	mariaSilva@cursoSQL.com
3	Francisco da Silva	109876543-21	1990-01-01	M	franciscoSilva@cursoSQL.com
4	Francisca da Silva	NULL	NULL	F	franciscaSilva@cursoSQL.com

**Figura 2** – Tela do MySQL após o comando `SELECT * FROM clientes`  
Fonte: MySQL Server 5.1

Muitas vezes, quando a tabela possui muitos atributos, a visualização de todos os dados com o comando `SELECT` gera um resultado visual desagradável e de difícil interpretação. Isso porque a largura da tabela é grande demais para caber na janela do sistema e as informações de cada registro acabam sendo apresentadas na linha seguinte, conforme ilustrado na Figura 3.



cli_codigo	cli_nome	cli_cpf	cli_data_nasc	cli_sexo	cli_email
1	José da Silva	123456789-10	1980-12-10	M	joseSilva@cursoSQL.com
2	Maria da Silva	012345678-99	1982-02-28	F	mariaSilva@cursoSQL.com
3	Francisco da Silva	109876543-21	1990-01-01	M	franciscoSilva@cursoSQL.com
4	Francisca da Silva	NULL	NULL	F	franciscaSilva@cursoSQL.com

**Figura 3** – Tela do MySQL após o comando `SELECT * FROM clientes`  
Fonte: MySQL Server 5.1

Caso você não deseje ver todas as colunas de sua tabela, simplesmente forneça os nomes das colunas que você tiver interesse, separando os campos por vírgulas. Por exemplo, se você deseja obter uma lista dos nomes de seus clientes com seus respectivos emails, selecione as colunas `cli_nome` e `cli_email`, conforme apresentado no quadro abaixo.

```
mysql> SELECT cli_nome, cli_email
FROM clientes;
```

Observe que apenas as informações sobre os nomes e emails foram listados, conforme ilustrado na Figura 4. Essa é uma boa prática de programação, pois além de permitir uma melhor visualização dos dados necessários, ela acelera a recuperação de seus resultados.



cli_nome	cli_email
José da Silva	joseSilva@cursoSQL.com
Maria da Silva	mariaSilva@cursoSQL.com
Francisco da Silva	franciscoSilva@cursoSQL.com
Francisca da Silva	franciscaSilva@cursoSQL.com

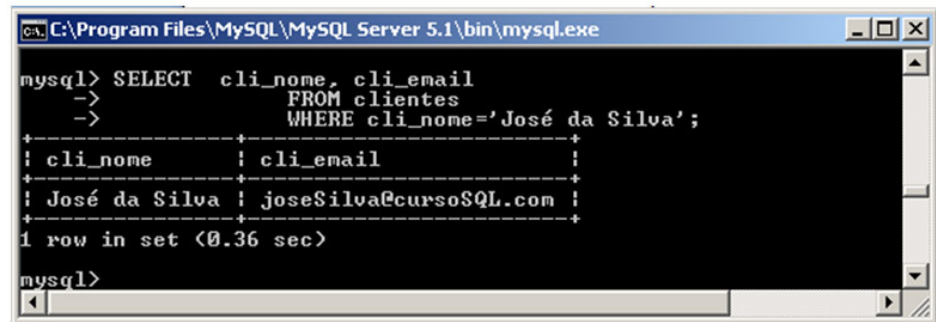
**Figura 4** – Tela do MySQL após o comando `SELECT cli_nome, cli_email FROM clientes`  
Fonte: MySQL Server 5.1

Agora, se você deseja só visualizar linhas específicas de sua tabela, você poderá utilizar a cláusula WHERE para limitar os registros a serem exibidos. Por exemplo, se você quiser pesquisar só o email do cliente Sr. José da Silva, você executaria uma consulta como essa:

```
mysql> SELECT cli_nome, cli_email  
        FROM clientes WHERE cli_nome='José da Silva';
```

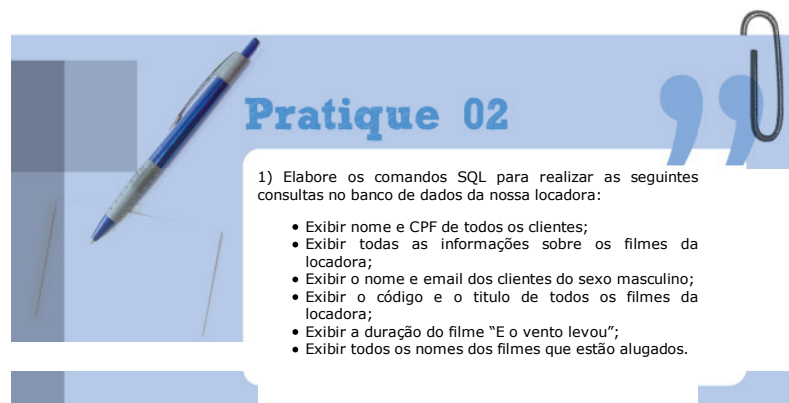
Observe que o nome José da Silva foi colocado entre aspas simples, pois o atributo cli\_nome é do tipo VARCHAR. Lembre que os tipos de dados textuais como VARCHAR, CHAR, DATE e TIME precisam de aspas simples. E os tipos numéricos como DEC e INT não precisam.

A resposta do SGBD, no caso do MySQL, ao comando acima, é ilustrada na Figura 5.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe  
mysql> SELECT cli_nome, cli_email  
        FROM clientes  
        WHERE cli_nome='José da Silva';  
+-----+-----+  
| cli_nome | cli_email |  
+-----+-----+  
| José da Silva | joseSilva@cursoSQL.com |  
+-----+-----+  
1 row in set (0.36 sec)  
mysql>
```

Figura 5 – Tela do MySQL após o comando `SELECT cli_nome, cli_email FROM clientes WHERE cli_nome='José da Silva'`  
Fonte: MySQL Server 5.1



## Pratique 02

- 1) Elabore os comandos SQL para realizar as seguintes consultas no banco de dados da nossa locadora:
  - Exibir nome e CPF de todos os clientes;
  - Exibir todas as informações sobre os filmes da locadora;
  - Exibir o nome e email dos clientes do sexo masculino;
  - Exibir o código e o título de todos os filmes da locadora;
  - Exibir a duração do filme "E o vento levou";
  - Exibir todos os nomes dos filmes que estão alugados.

## A cláusula WHERE

Como já foi visto, a cláusula WHERE é utilizada para restringir os dados que serão listados. Até agora, temos usado o sinal de igualdade (=) para listar apenas os registros que possuem atributos com valores exatos aos desejados. Entretanto, qualquer um dos operadores de comparação (<, <=, >, >=, <>, =), que você aprendeu a utilizar nas disciplinas de programação, pode ser utilizado para testar os valores dos atributos na cláusula WHERE.

O sinal de menor (<) é utilizado na cláusula WHERE do comando SELECT, para visualizar os registros que possuem os valores menores que a condição testada. Funcionalidade semelhante tem o operador menor ou igual (<=) que quando utilizado permite selecionar apenas os registros que possuem os valores menores ou iguais à condição testada.

Os sinais de maior (>) e maior ou igual (>=) são utilizados quando desejamos visualizar apenas os registros que possuem os valores maiores e maiores ou iguais à condição testada, respectivamente.

Na cláusula WHERE, quando queremos listar todos os registros que não condizem com a condição testada, ou seja, os registros que possuem valores diferentes da condição testada, utiliza-se o operador de comparação de diferença (<>).

Vamos praticar a utilização dos operadores de comparação fazendo pequenas consultas nas tabelas **clientes** e **filmes** do nosso banco de dados **locadora**.

Inicialmente, vamos realizar uma consulta à tabela **filmes** para determinar quais os filmes cujo valor do aluguel é menor que 3,00 reais, através do seguinte comando:

```
mysql> SELECT fil_titulo  
        FROM filmes WHERE fil_preco < 3;
```

Para pesquisar pelos filmes que não estão alugados, utilizamos a seguinte estrutura:

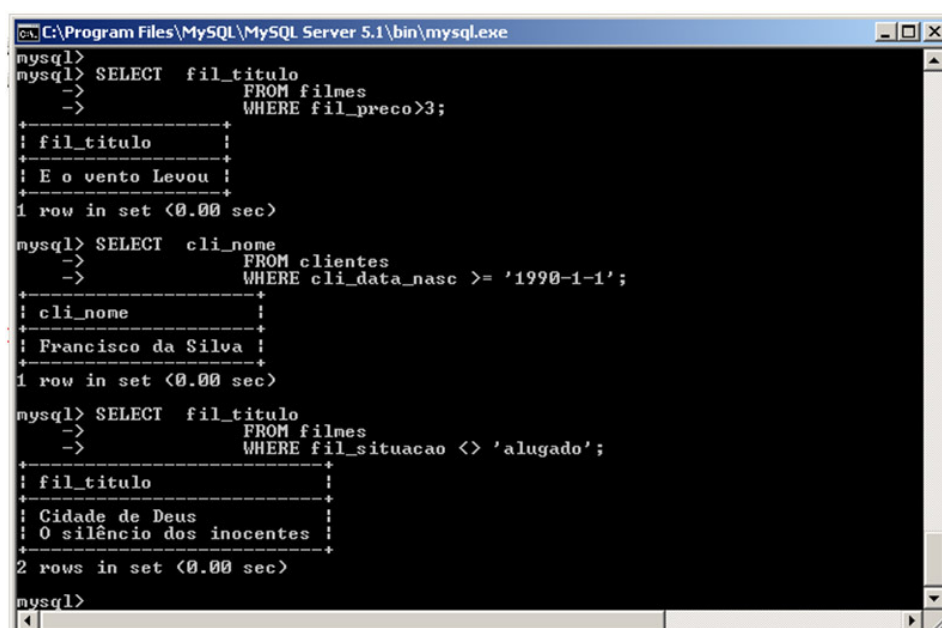
```
mysql> SELECT fil_titulo  
        FROM filmes  
        WHERE fil_situacao <> 'alugado';
```

A estrutura `fil_situacao <> 'alugado'` é válida, pois os operadores de comparação podem ser utilizados para fazerem comparações de valores alfanuméricos. Nesse caso, a comparação é feita em relação à ordem alfabética. Por exemplo, a estrutura `cli_nome > 'José da Silva'` retornará todos os registros cujo nome do cliente vem após José da Silva, considerando os nomes organizados em ordem alfabética.

E, finalmente, vamos realizar uma consulta à tabela **clientes** para determinar quais clientes nasceram depois ou no ano de 1990, usando o seguinte comando:

```
mysql>SELECT cli_nome
      FROM clientes WHERE cli_data_nasc >= '1990-1-1';
```

As informações fornecidas pelo sistema às pesquisas realizadas estão ilustradas na Figura 6.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql>
mysql> SELECT  fil_titulo
      ->        FROM filmes
      ->        WHERE fil_preco>3;
+-----+
| fil_titulo |
+-----+
| E o vento Levou |
+-----+
1 row in set (0.00 sec)

mysql> SELECT  cli_nome
      ->        FROM clientes
      ->        WHERE cli_data_nasc >= '1990-1-1';
+-----+
| cli_nome |
+-----+
| Francisco da Silva |
+-----+
1 row in set (0.00 sec)

mysql> SELECT  fil_titulo
      ->        FROM filmes
      ->        WHERE fil_situacao <> 'alugado';
+-----+
| fil_titulo |
+-----+
| Cidade de Deus |
| O silêncio dos inocentes |
+-----+
2 rows in set (0.00 sec)

mysql>
```

**Figura 6** – Tela do MySQL após diversas pesquisas com o comando SELECT, usando operadores de comparação na cláusula WHERE  
Fonte: MySQL Server 5.1

Além dos operadores de comparação, podemos utilizar os operadores lógicos AND (E) e OR (OU) para fazermos combinações de condições na cláusula WHERE.

O operador lógico AND deve ser utilizado quando se quer fazer uma consulta a registros que obedeçam a todas as condições impostas. Diferentemente do operador lógico OR que deve ser utilizado para exibir registros quando quaisquer das condições sejam atendidas, ou seja, pelo menos uma condição seja verdadeira. Analise os seguintes exemplos.

#### Exemplo 1

Selecionar os filmes cujo valor do aluguel é menor que 3,00 reais e que não estão alugados:

```
mysql>SELECT fil_titulo
      FROM filmes
      WHERE fil_preco< 3 AND fil_situacao<>'alugado';
```

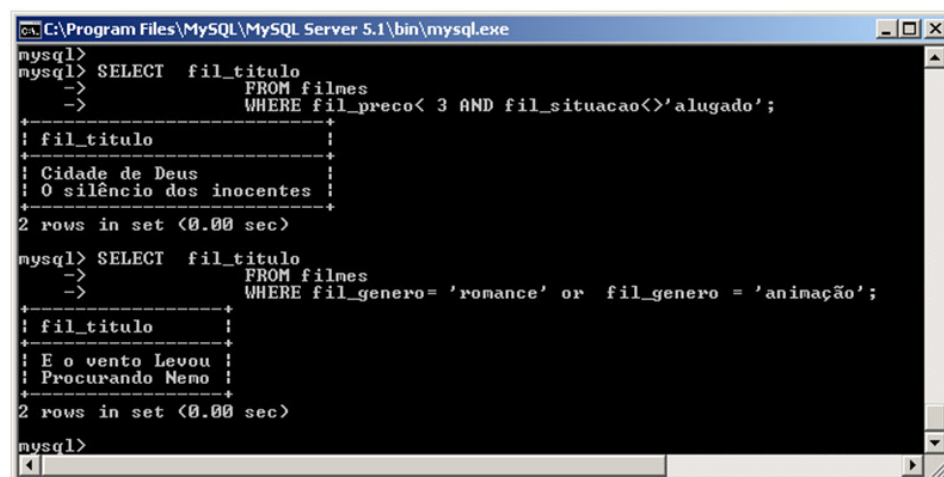
Observe nesse exemplo que utilizamos na mesma cláusula WHERE os operadores de comparação (< e <>) e operador lógico (AND).

#### Exemplo 2

Pesquisar os filmes cujo gênero é romance ou animação:

```
mysql>SELECT fil_titulo
      FROM filmes WHERE fil_genero= 'romance' OR fil_genero = 'animação';
```

Os resultados dessas pesquisas são ilustrados na Figura 7.



```

C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql>
mysql> SELECT  fil_titulo
->            FROM filmes
->            WHERE fil_preco < 3 AND fil_situacao <> 'alugado';
+-----+
| fil_titulo |
+-----+
| Cidade de Deus |
| O silêncio dos inocentes |
+-----+
2 rows in set (0.00 sec)

mysql> SELECT  fil_titulo
->            FROM filmes
->            WHERE fil_genero = 'romance' or fil_genero = 'animação';
+-----+
| fil_titulo |
+-----+
| E o vento Levou |
| Procurando Nemo |
+-----+
2 rows in set (0.00 sec)

mysql>

```

**Figura 7** – Tela do MySQL após diversas pesquisas com o comando SELECT, usando operadores lógicos na cláusula WHERE  
Fonte: MySQL Server 5.1

Além dos operadores de comparação e dos operadores lógicos, podemos utilizar algumas palavras chaves na cláusula WHERE para facilitar a elaboração dos comandos, tais como:

- BETWEEN - Usado para verificar se o valor de um atributo está em um intervalo de valores. Especifica um intervalo a ser testado;
- LIKE - Utilizada para comparar cadeias de caracteres usando padrões de comparação para um ou mais caracteres. Normalmente, o coringa percentual (%) substitui zero, um ou mais caracteres e o coringa sublinha (.) substitui um único caractere.
- IN - Usado para verificar se o valor de um atributo está em um conjunto de valores entre parênteses. Quando o valor for compatível com um dos valores do conjunto, o registro é exibido.
- IS NULL - Usado para selecionar diretamente um valor NULL.

Para entendermos melhor a utilização dessas palavras-chaves, vamos analisar os seguintes exemplos. Examine com cuidado e não deixe de praticar em seu banco de dados. Lembre-se de que a prática leva à perfeição!!!

#### Exemplo 1

Pesquisar os títulos dos filmes que possuem código entre 2 e 20.

```
mysql>SELECT fil_titulo
      FROM filmes WHERE fil_codigo BETWEEN 2 AND 20;
```

#### Exemplo 2

Pesquisar os nomes dos clientes que nasceram entre 1º de janeiro de 1990 e 1º de janeiro de 2000.

```
mysql>SELECT cli_nome
      FROM clientes WHERE cli_data_nasc BETWEEN '1990-1-1' AND '2000-1-1';
```

#### Exemplo 3

Pesquisar o email dos clientes que possuam a primeira letra do seu nome entre A e G.

```
mysql>SELECT cli_email
      FROM clientes WHERE cli_nome BETWEEN 'A' AND 'G';
```

#### Exemplo 4

Pesquisar o nome dos clientes que usem o *gmail* como um dos seus servidores de emails, cadastrados no nosso banco de dados.

```
mysql>SELECT cli_nome
      FROM clientes WHERE cli_email LIKE '%gmail.com';
```

Lembre-se de que o sinal de percentagem é um substituto para qualquer número de caracteres desconhecidos. E a sublinha é um substituto para apenas um caractere desconhecido. Dessa forma, se tivéssemos LIKE '\_A', estaríamos pesquisando *string* de dois caracteres cujo primeiro caractere podia ser qualquer um desde que a última letra fosse A.

#### Exemplo 5

Pesquisar o nome dos filmes cujo gênero é comédia, romance ou ação.

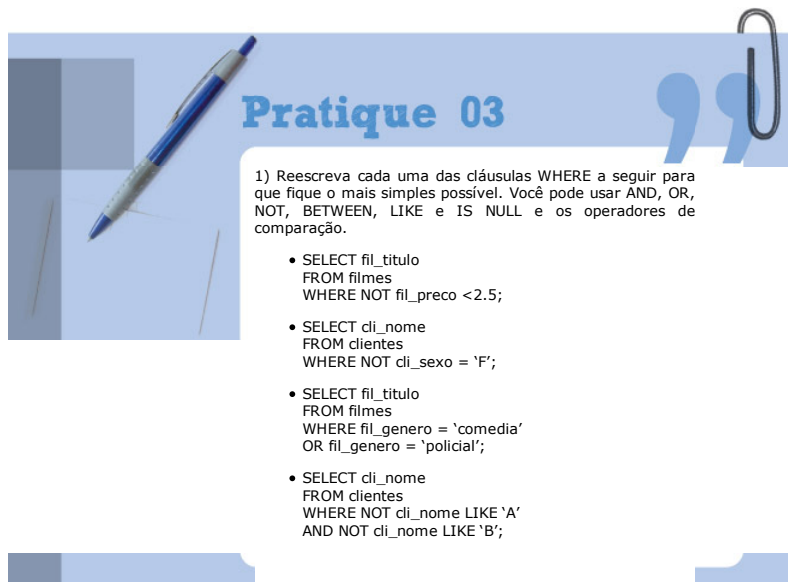
```
mysql>SELECT fil_titulo
      FROM filmes WHERE fil_genero IN ('comedia', 'romance', 'acao');
```

Para pesquisarmos todos os filmes exceto aqueles de comédia, romance ou ação, bastaria adicionar a palavra NOT na nossa declaração IN. Ou seja, a palavra-chave NOT IN diz ao sistema que recupere os resultados que não estão no conjunto de termos informados.

#### Exemplo 6

Pesquisar o nome dos clientes que não possuam *email* cadastrado no nosso banco de dados.

```
mysql>SELECT cli_nome  
      FROM clientes WHERE cli_email IS NULL;
```



### Pratique 03

1) Reescreva cada uma das cláusulas WHERE a seguir para que fique o mais simples possível. Você pode usar AND, OR, NOT, BETWEEN, LIKE e IS NULL e os operadores de comparação.

- SELECT fil\_titulo  
 FROM filmes  
 WHERE NOT fil\_preco <2.5;
- SELECT cli\_nome  
 FROM clientes  
 WHERE NOT cli\_sexo = 'F';
- SELECT fil\_titulo  
 FROM filmes  
 WHERE fil\_genero = 'comedia'  
 OR fil\_genero = 'policial';
- SELECT cli\_nome  
 FROM clientes  
 WHERE NOT cli\_nome LIKE 'A'  
 AND NOT cli\_nome LIKE 'B';

Encerramos por aqui mais uma aula sobre a linguagem SQL. Na próxima aula, aprenderemos a fazer consultas mais avançadas, bem como realizar operações matemáticas nos seus resultados. Lembre-se de fazer sua autoavaliação. E se precisar pare e reflita mais um pouco sobre o que estudamos. Bons estudos e boa sorte!

## Resumo

Nesta aula, apresentamos o comando DESC que é utilizado para visualizar a estrutura de uma tabela em um banco de dados. Também vimos os comandos SHOW DATABASES e SHOW TABLES para visualizar os bancos de dados presentes num diretório e as tabelas associadas a um banco de dados. A seguir, iniciamos o nosso estudo sobre o poderoso comando SELECT que é composto por três cláusulas: SELECT, FROM e WHERE. A cláusula WHERE foi estudada em maiores detalhes. Sendo assim, vimos que podemos utilizar os operadores de comparação (<, <=, >, >=, <>, =) para testar os valores dos atributos e os operadores lógicos (AND e OR) para fazermos combinações de condições. E finalizamos nossa aula estudando as palavras-chaves (BETWEEN, LIKE, IN e IS NULL), que são utilizadas na cláusula WHERE para facilitar a elaboração do comando SELECT.



## Evolucao

Considere o banco de dados CursoX criado na aula anterior cuja estrutura de tabelas é mostrada abaixo:

**TABELA alunos**

ATRIBUTO	TIPO	DESCRIÇÃO
aluno_cod	Número inteiro	Código do aluno
aluno_nome	Alfanumérico	Nome do aluno
aluno_endereco	Alfanumérico	Endereço do aluno
aluno_cidade	Alfanumérico	Cidade do aluno

**TABELA disciplina**

ATRIBUTO	TIPO	DESCRIÇÃO
----------	------	-----------

dis_cod	Número inteiro	Código da disciplina
dis_nome	Alfanumérico	Nome da disciplina
dis_carga	Número inteiro	Carga horária da disciplina
dis_professor	Alfanumérico	Professor da disciplina

**TABELA professores**

ATRIBUTO	TIPO	DESCRIÇÃO
prof_cod	Número inteiro	Código do professor
prof_nome	Alfanumérico	Nome do professor
prof_endereco	Alfanumérico	Endereço do professor
prof_cidade	Alfanumérico	Cidade do professor

Resolva as consultas abaixo utilizando a linguagem SQL:

1. Exibir código, nome e endereço de todos os professores do nosso colégio.
2. Exibir os 3 primeiros alunos cadastrados.
3. Exibir o nome de todas as disciplinas e seus respectivos professores.
4. Mostrar os primeiros 50% de alunos cadastrados.
5. Mostrar o nome das disciplinas que não possuem professor cadastrado.
6. Exibir as disciplinas com código entre 5 e 15.
7. Mostrar o nome dos alunos que possuem sobrenome Silva.

**Referencias**

BEIGHLEY, L. **Use a cabeça SQL**. Rio de Janeiro: Editora AltaBooks, 2008.

ySQL 5.1 Reference Manual. Disponível em: <<http://dev.mysql.com/doc/refman/5.1/en/>>. Acesso em: 24 set. 2010.

 Voltar  Imprimir  Topo