

Banco de Dados

Aula 10 - Linguagem SQL – Consulta simples de tabelas

Apresentação

Na aula anterior, você viu os comandos `CREATE DATABASE` e `USE`, que são utilizados para criar e para utilizar efetivamente um banco de dados. Em seguida, estudou 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. Estudou, também, como fazer atualizações e apagar linhas nas tabelas de um banco de dados através dos comandos `UPDATE` e `DELETE FROM`, respectivamente. Finalizando a aula, você estudou 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. Também pode estar se perguntando como se faz 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, você vai estudar como visualizar a estrutura de uma tabela e depois verá como realizar pesquisas simples em uma tabela utilizando o famoso comando `SELECT`.



Vídeo 01 - Apresentação da Aula

Objetivos

- 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, você aprendeu 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:

```
1 | mysql> DESC nome_da_tabela;
```

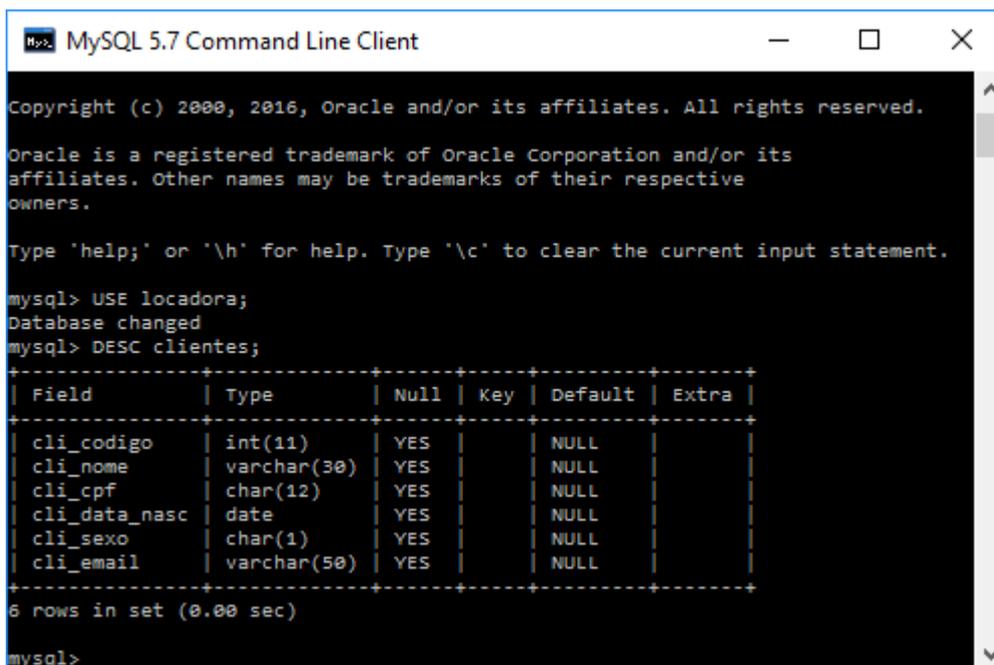
Ao executar esse comando corretamente, você obterá como resposta do sistema a estrutura da tabela, que informa quais são seus campos ou atributos (*Field*) e os seus respectivos tipos (*Type*). Além das informações sobre as restrições de cada atributo, tais como: valor padrão (*Default*), se é uma chave (*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 a seguir.

```
1 | 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.

Figura 01 - Tela do MySQL após o comando DESC **clientes**.



```
MySQL 5.7 Command Line Client
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> USE locadora;
Database changed
mysql> DESC clientes;
+-----+-----+-----+-----+-----+-----+
| 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 (0.00 sec)
mysql>
```

Fonte: MySQL 5.7 *Command Line Client*

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.

Atividade 01

1. 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 9) 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 9, que mostrou, dentre outras coisas, como criar e apagar tabelas.

Consulta simples em tabelas

Vamos supor que precisamos obter o e-mail 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 a seguir.

```
1 mysql> SELECT atributo1, atributo2, ...
2   FROM nome_da_tabela1, nome_da_tabela2, ...
3   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 turma 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.

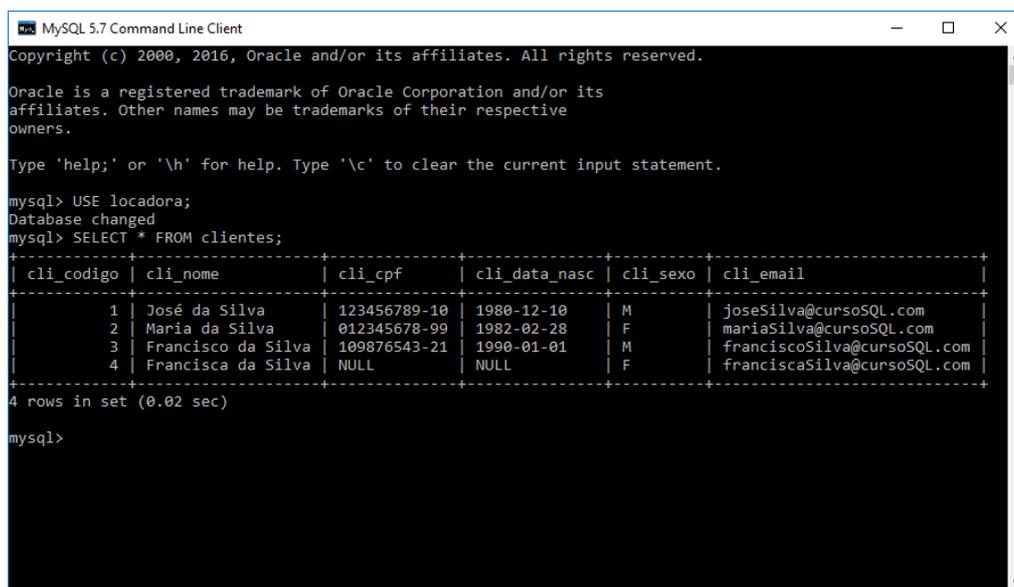
Vamos exercitar a utilização do comando SELECT fazendo algumas consultas aos dados da tabela **clientes** e **filmes** do nosso banco de dados **locadora**.

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

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

A resposta fornecida pelo sistema a essa consulta 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.

Figura 02 - Tela do MySQL após o comando `SELECT * FROM clientes`.



```
MySQL 5.7 Command Line Client
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

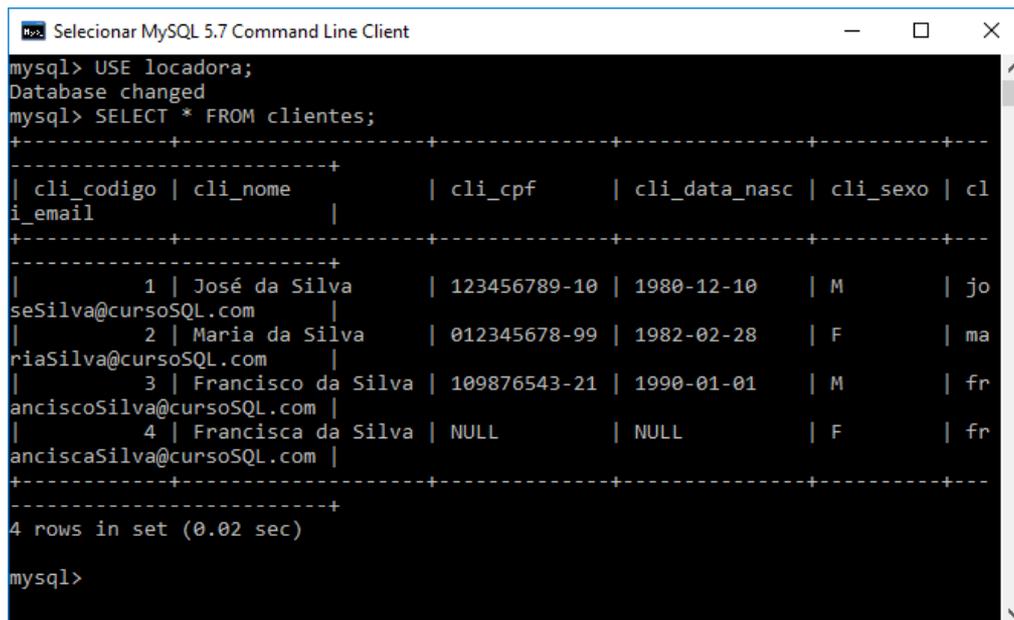
mysql> USE locadora;
Database changed
mysql> SELECT * FROM clientes;
+-----+-----+-----+-----+-----+-----+
| cli_codigo | cli_nome | cli_cpf | cli_data_nasc | cli_sexo | cli_email |
+-----+-----+-----+-----+-----+-----+
| 1 | José da Silva | 123456789-10 | 1980-12-10 | M | joseSilva@cursoSQL.com |
| 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 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)

mysql>
```

Fonte: MySQL 5.7 *Command Line Client*

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**.

Figura 03 - Tela do MySQL após o comando `SELECT * FROM clientes`.



```
mysql> USE locadora;
Database changed
mysql> SELECT * FROM clientes;
+-----+-----+-----+-----+-----+-----+
| 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 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)

mysql>
```

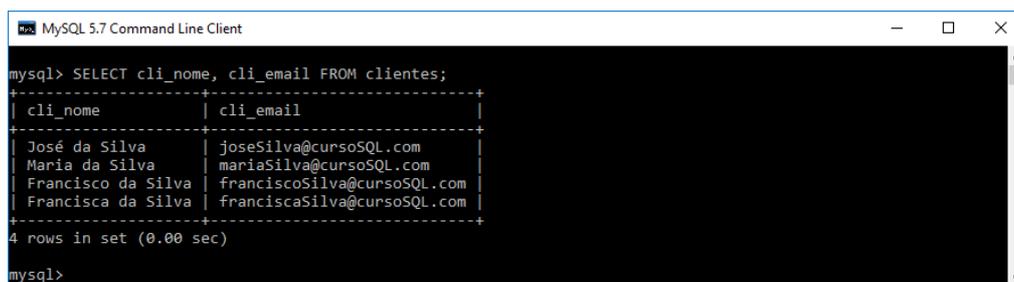
Fonte: MySQL 5.7 *Command Line Client*

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 e-mails, selecione as colunas `cli_nome` e `cli_email`, conforme apresentado a seguir.

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

Observe que apenas as informações sobre os nomes e e-mails 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.

Figura 04 - Tela do MySQL após o comando `SELECT cli_nome, cli_email FROM clientes`.



```
mysql> SELECT cli_nome, cli_email FROM clientes;
+-----+-----+
| 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 |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Fonte: MySQL 5.7 *Command Line Client*

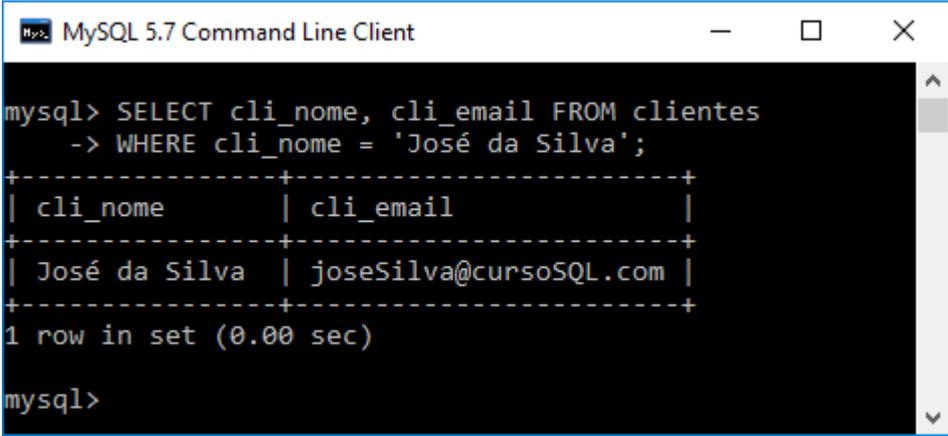
Agora, se você deseja só visualizar linhas específicas de sua tabela, você poderá utilizar a cláusula WHERE para limitar os registros a serem exibidos. Por exemplo, se você quiser pesquisar só o e-mail do cliente Sr. José da Silva, você executaria uma consulta como essa:

```
1 mysql> SELECT cli_nome, cli_email
2   FROM clientes
3   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, à esse comando, é ilustrada na Figura 5.

Figura 05 - Tela do MySQL após o comando **SELECT cli_nome, cli_email FROM clientes WHERE cli_nome='José da Silva'**.



```
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 (0.00 sec)

mysql>
```

Fonte: MySQL 5.7 *Command Line Client*



Vídeo 02 - Visualização de Estrutura de Consultas

Atividade 02

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

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:

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

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

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

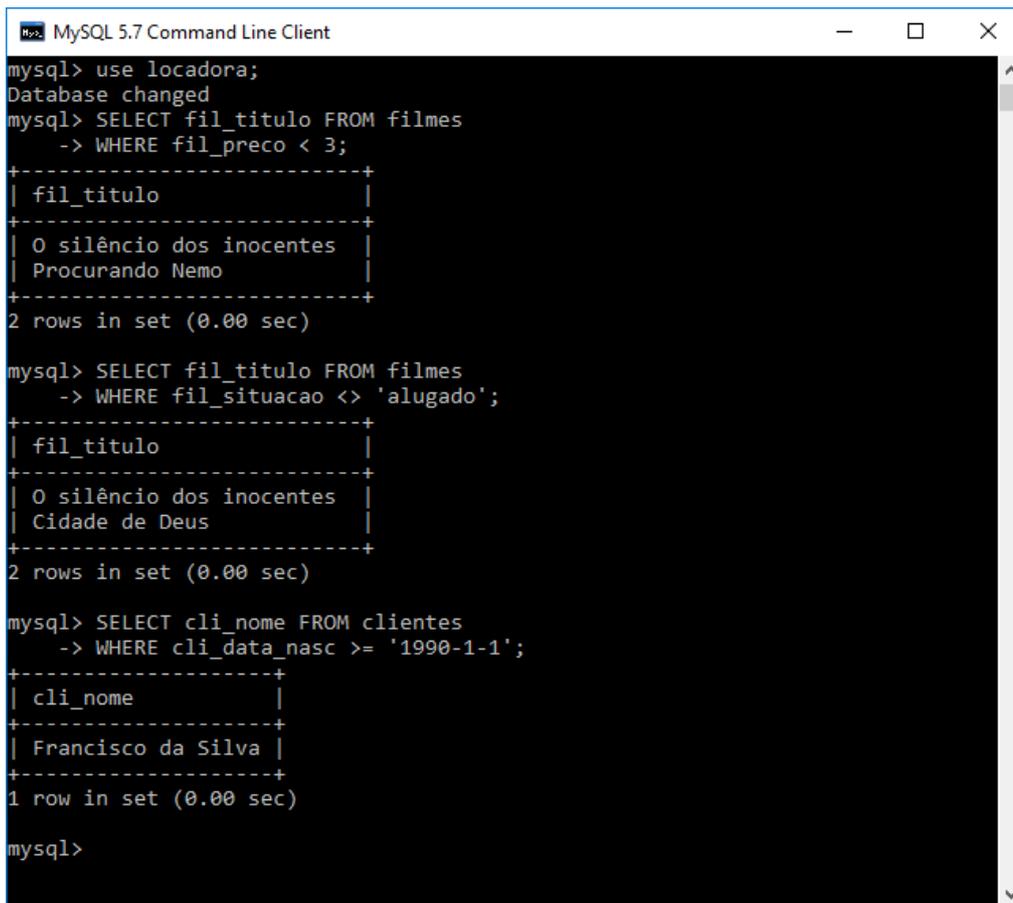
A estrutura **fil_situacao <> 'alugado'** é válida, pois os operadores de comparação podem ser utilizados para fazer comparações de valores alfanuméricos. Nesse caso, a comparação é feita em relação à ordem alfabética. Por exemplo, a estrutura **cli_nome > 'José da Silva'** retornará todos os registros 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:

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

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

Figura 06 - Tela do MySQL após diversas pesquisas com o comando SELECT, usando operadores de comparação na cláusula WHERE.



```
mysql> use locadora;
Database changed
mysql> SELECT fil_titulo FROM filmes
  -> WHERE fil_preco < 3;
+-----+
| fil_titulo |
+-----+
| O silêncio dos inocentes |
| Procurando Nemo          |
+-----+
2 rows in set (0.00 sec)

mysql> SELECT fil_titulo FROM filmes
  -> WHERE fil_situacao <> 'alugado';
+-----+
| fil_titulo |
+-----+
| O silêncio dos inocentes |
| Cidade de Deus           |
+-----+
2 rows 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>
```

Fonte: MySQL 5.7 *Command Line Client*

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:

```
1 mysql>SELECT fil_titulo
2   FROM filmes
3   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).

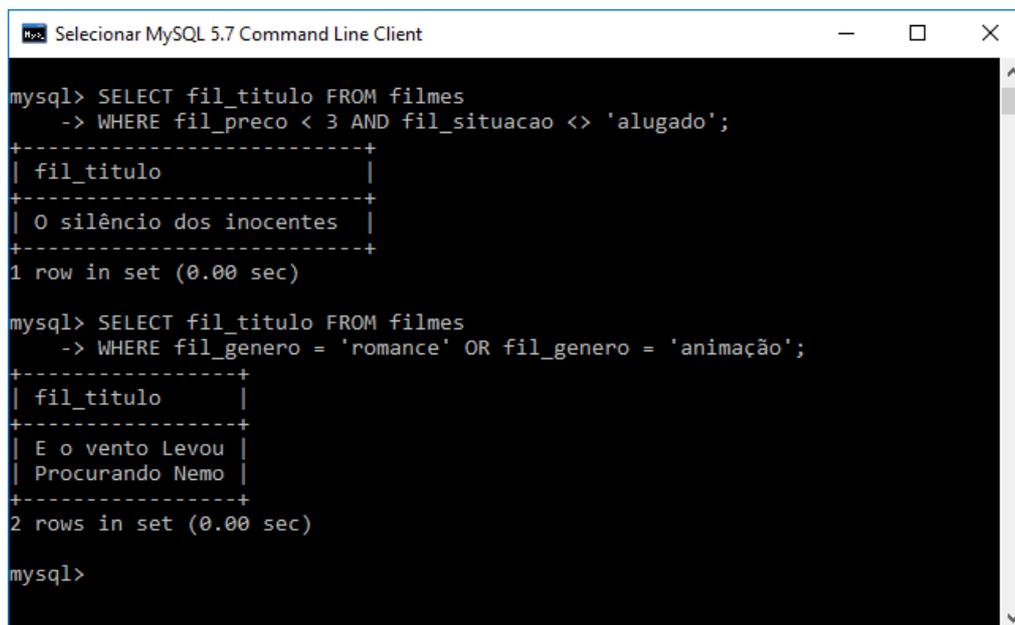
Exemplo 2

Pesquisar os filmes cujo gênero é romance ou animação:

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

Os resultados dessas pesquisas são ilustrados na **Figura 7**.

Figura 07 - Tela do MySQL após diversas pesquisas com o comando SELECT, usando operadores lógicos na cláusula WHERE.



```
Selecionar MySQL 5.7 Command Line Client

mysql> SELECT fil_titulo FROM filmes
-> WHERE fil_preco < 3 AND fil_situacao <> 'alugado';
+-----+
| fil_titulo |
+-----+
| O silêncio dos inocentes |
+-----+
1 row 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>
```

Fonte: MySQL 5.7 *Command Line Client*



Vídeo 03 - Aplicação de Filtros

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 entender 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.

```
1 mysql>SELECT fil_titulo  
2 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.

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

Exemplo 3

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

```
1 mysql>SELECT cli_email  
2 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 e-mail cadastrado no nosso banco de dados.

```
1 mysql>SELECT cli_nome  
2 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.

```
1 mysql>SELECT fil_titulo
2 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 e-mail cadastrado no nosso banco de dados.

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



Vídeo 04 - Uso de Operadores

Atividade 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.
 - a. SELECT fil_titulo FROM filmes WHERE NOT fil_preco <2.5;
 - b. SELECT cli_nome FROM clientes WHERE NOT cli_sexo = 'F';
 - c. SELECT fil_titulo FROM filmes WHERE fil_genero = 'comedia' OR fil_genero = 'policial';
 - d. SELECT cli_nome FROM clientes WHERE NOT cli_nome LIKE 'A' AND NOT cli_nome LIKE 'B';

Conclusão

Encerramos por aqui mais uma aula sobre a linguagem SQL. Na próxima aula, você vai aprender como 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, você estudou o comando DESC, que é utilizado para visualizar a estrutura de uma tabela em um banco de dados. Viu também 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, iniciou seu 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, viu 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. Finalizando a aula, você estudou as palavras-chaves BETWEEN, LIKE, IN e IS NULL, que são utilizadas na cláusula WHERE para facilitar a elaboração do comando SELECT.

Autoavaliação

1. Considere o banco de dados CursoX criado na aula anterior cuja estrutura de tabelas é mostrada a seguir:

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 - Alunos

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 - Disciplinas

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

TABELA - Professores

Resolva as consultas a seguir utilizando a linguagem SQL:

- a. Exibir código, nome e endereço de todos os professores do nosso colégio.
- b. Exibir os 3 primeiros alunos cadastrados.

- c. Exibir o nome de todas as disciplinas e seus respectivos professores.
- d. Mostrar os primeiros 50% de alunos cadastrados.
- e. Mostrar o nome das disciplinas que não possuem professor cadastrado.
- f. Exibir as disciplinas com código entre 5 e 15.
- g. Mostrar o nome dos alunos que possuem sobrenome Silva.

Referências

BEIGHLEY, L. **Use a cabeça SQL**. Rio de Janeiro: Alta Books, 2008.

MYSQL 5.7 **Reference Manual**. Disponível em:
<<http://dev.mysql.com/doc/refman/5.7/en/>>. Acesso em: 15 jan. 2017.