

Banco de Dados

Aula 09 - Linguagem SQL – criação, inserção e
modificação de tabelas

Apresentação

Na aula anterior, você aprendeu que existe uma linguagem padrão de acesso aos bancos de dados denominada de SQL e conheceu um pouco de sua história. Verificou que para usar um banco de dados é necessário instalar um Sistema Gerenciador de Banco de Dados (SGBD) e que existem diversas opções com características variadas. Por fim, realizou os procedimentos de instalação e configuração do MySQL.

Nesta aula, daremos continuidade ao estudo da linguagem SQL. Você vai aprender como criar um banco de dados e suas tabelas. Aprenderá como inserir, atualizar e apagar dados nas tabelas.



Vídeo 01 - Apresentação

Objetivos

- Criar banco de dados e tabelas.
- Inserir dados em tabelas.
- Atualizar e apagar dados em tabelas.

Criação de tabelas

Já vimos, em aulas anteriores, que em um banco de dados relacional, uma tabela é um conjunto de dados organizado em uma estrutura de linhas e colunas. Em uma tabela, cada linha (registro) contém todas as informações sobre um único objeto. As colunas (atributos) caracterizam os tipos de dados que deverão constar na tabela (numéricos, alfanuméricos, datas etc.).

Agora, vamos aprender a criar tabelas com a linguagem SQL? Primeiro você precisará criar um banco de dados para armazenar todas as suas tabelas. Vamos relembrar o comando que é usado para criar bancos de dados, visto na aula anterior? Para criar um banco de dados, digite o comando apresentado no quadro a seguir, não se esqueça de teclar ENTER após o sinal de ponto-e-vírgula.

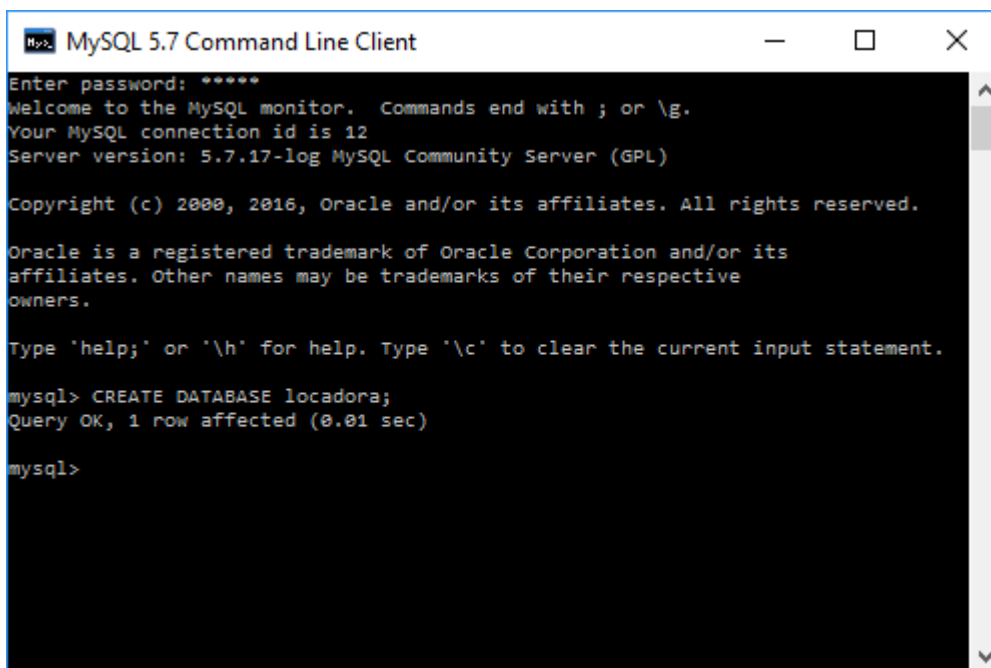
```
1 mysql> CREATE DATABASE nome_do_banco_de_dados;
```

Agora, vamos praticar os conceitos aprendidos criando um banco de dados chamado **locadora**, no qual posteriormente vamos criar nossas tabelas. Para criarmos esse banco de dados, vamos digitar o comando a seguir:

```
1 mysql> CREATE DATABASE locadora;
```

A resposta do SGBD, no caso do MySQL, ao comando CREATE DATABASE **locadora** é ilustrada na **Figura 1**.

Figura 01 - Tela do MySQL após a criação do banco de dados **locadora**.



```
MySQL 5.7 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.17-log MySQL Community Server (GPL)

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> CREATE DATABASE locadora;
Query OK, 1 row affected (0.01 sec)

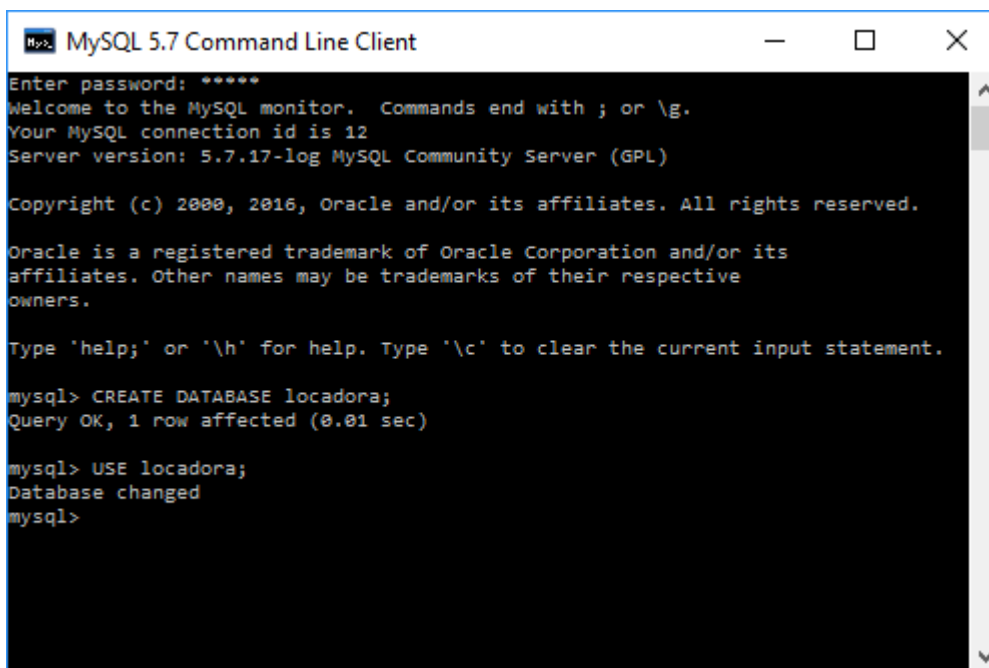
mysql>
```

O passo seguinte é dizer ao sistema que você quer utilizar o banco de dados **locadora**, através do comando:

```
1 mysql>USE locadora;
```

A mensagem fornecida pelo sistema ao comando USE **locadora** nos informa que o banco de dados corrente foi alterado (*Database changed*), conforme é ilustrado na **Figura 2**.

Figura 02 - Tela do MySQL após o comando USE **locadora**.



```
MySQL 5.7 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.17-log MySQL Community Server (GPL)

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> CREATE DATABASE locadora;
Query OK, 1 row affected (0.01 sec)

mysql> USE locadora;
Database changed
mysql>
```

Fonte: MySQL 5.7

Em geral, a maioria dos SGBDs possui editores gráficos de banco de dados que permitem a criação rápida e simples de qualquer tipo de tabela com qualquer tipo de formato. Entretanto, iremos estudar os comandos diretamente na linguagem SQL, ou seja, do modo como devem ser digitados na linha de comando.

Para criar uma tabela, devemos especificar diversos dados: o nome que queremos atribuir a essa tabela, seus atributos e seus tipos. Ademais, pode ser necessário especificar quais desses campos serão índices (chave primária, chave estrangeira,...) e as restrições de integridade, a fim de evitar a inconsistência dos dados nas tabelas.

A sintaxe de criação pode variar ligeiramente entre os diferentes SGBDs, já que os tipos de campos aceitos não são completamente padronizados. O comando para criar uma tabela é semelhante ao comando de criação de um banco de dados (CREATE DATABASE nome_do_banco;), conforme é apresentado no quadro a seguir.

```
1 mysql>CREATE TABLE nome_da_tabela
2 (
3     atributo 1 tipo1,
4     atributo 2 tipo 2,
5     ...
6     atributo N tipo N
7 );
```

É importante que você se lembre de usar um parêntese aberto antes do início da lista de atributos e um parêntese de fechamento após o final da definição dos atributos. Certifique-se de separar cada definição de coluna com uma vírgula. Lembre-se: todas as declarações SQL devem terminar com um ";".

No momento da criação de uma tabela em um banco de dados, devemos definir para cada atributo o seu respectivo tipo. Os tipos mais comuns de dados são:

- **CHAR (tamanho):** sequência de caracteres (string) de comprimento fixo. O tamanho é especificado entre parênteses. O tamanho máximo permitido é de 255 caracteres.
- **VARCHAR (tamanho):** sequência de caracteres (string) com tamanho variável. a quantidade máxima de caracteres que poderá ser armazenada no campo é especificada entre parênteses. O tamanho era limitado entre 0 e 255 até o MySQL 5.0.3. Após isto, o limite superior foi alterado para 65,535.
- **INT:** tipo numérico que aceita valores inteiros. Podemos representar com esse tipo qualquer valor inteiro na faixa entre -2.147.483.648 e 2.147.483.647.
- **NUMERIC (n,d):** tipo numérico que aceita valores reais (**n** indica a quantidade total de números e **d** indica a quantidade do total que corresponde a casas decimais). Exemplo: NUMERIC(5,2) corresponde a números com 5 dígitos com até duas casas decimais, como 256,12.
- **TIME:** tipo tempo no formato hora:minuto:segundo.
- **DATE:** tipo data no formato ano-mês-dia.

Quando as tabelas são criadas, é comum que uma ou mais colunas tenham **restrições** que lhes estão associadas. Restrição é basicamente uma regra associada a uma coluna que diz quais as limitações dos dados inseridos nessa coluna. Por exemplo, a restrição UNIQUE especifica que dois registros não podem ter o mesmo valor em uma determinada coluna. Eles devem ser todos originais. As restrições mais populares são NOT NULL e PRIMARY KEY. A restrição NOT NULL especifica que

uma coluna não pode ser deixada em branco. E a restrição PRIMARY KEY (chave primária) define uma identificação única de cada registro (ou linha) em uma tabela. Iremos aprender mais sobre restrições no decorrer da disciplina.



Vídeo 02 - Criação BD e Tabelas

Atividade 01

1. Nesta aula, alguns dos tipos de dados mais comuns foram apresentados. Faça uma busca na internet e descubra outros tipos de dados permitidos pelo MySQL.

Sugerimos o seguinte site para pesquisa:
<<http://dev.mysql.com/doc/refman/5.7/en/data-type-overview.html>>.

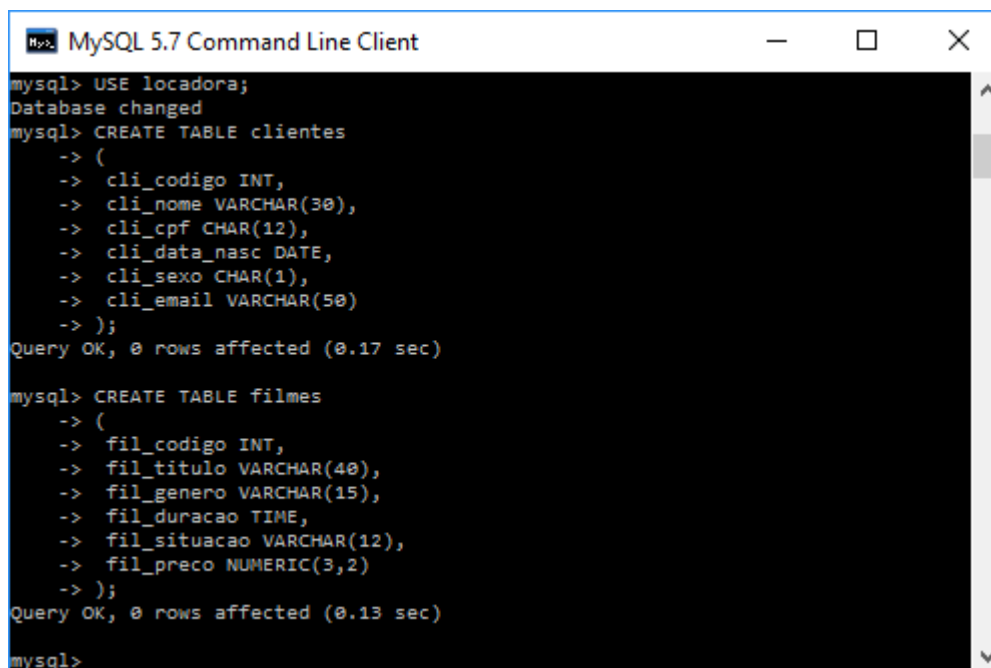
Neste ponto, estamos aptos a criar algumas tabelas no nosso banco de dados chamado **locadora**. Duas tabelas importantes no banco de dados da nossa locadora são as que contêm as informações sobre os clientes, denominada **clientes**, e a outra com as informações sobre os filmes, denominada **filmes**. Os comandos para criação dessas duas tabelas são apresentados a seguir. Examine com cuidado e não deixe de praticar em seu banco de dados. Lembre-se que a prática leva à perfeição!

```
1 mysql>CREATE TABLE clientes
2 (
3     cli_codigo INT,
4     cli_nome VARCHAR(30),
5     cli_cpf CHAR(12),
6     cli_data_nasc DATE,
7     cli_sexo CHAR(1),
8     cli_email VARCHAR(50)
9 );
10 mysql>CREATE TABLE filmes
11 (
12     fil_codigo INT,
13     fil_titulo VARCHAR(40),
14     fil_genero VARCHAR(15),
15     fil_duracao TIME,
16     fil_situacao VARCHAR(12),
17     fil_preco NUMERIC(3,2)
18 );
```

É uma boa prática de programação colocar na frente do atributo uma informação que permita identificar de forma simples, por exemplo, a qual tabela aquele atributo pertence. No exemplo, na tabela **clientes** foi adicionada uma abreviação da palavra clientes (cli) antes de cada atributo da tabela. Essa prática evita confusões de atributos iguais (por exemplo, codigo) nas tabelas **clientes** e **filmes**.

As respostas do SGBD, no caso o MySQL, aos comandos CREATE TABLE **clientes** e CREATE TABLE **filmes** são ilustradas na **Figura 3**.

Figura 03 - Tela do MySQL após os comandos CREATE TABLE **clientes** e CREATE TABLE **filmes**.



```
mysql> USE locadora;
Database changed
mysql> CREATE TABLE clientes
-> (
->   cli_codigo INT,
->   cli_nome VARCHAR(30),
->   cli_cpf CHAR(12),
->   cli_data_nasc DATE,
->   cli_sexo CHAR(1),
->   cli_email VARCHAR(50)
-> );
Query OK, 0 rows affected (0.17 sec)

mysql> CREATE TABLE filmes
-> (
->   fil_codigo INT,
->   fil_titulo VARCHAR(40),
->   fil_genero VARCHAR(15),
->   fil_duracao TIME,
->   fil_situacao VARCHAR(12),
->   fil_preco NUMERIC(3,2)
-> );
Query OK, 0 rows affected (0.13 sec)

mysql>
```

A mensagem “Query OK” informa que as tabelas **clientes** e **filmes** foram criadas corretamente.

Atividade 02

1. Vamos praticar um pouco? Para que você se familiarize com os comandos de criação de banco de dados e tabelas, crie um banco de dados da sua lanchonete preferida, por exemplo. Adicione a esse banco de dados as tabelas contendo informações sobre os funcionários e sobre o estoque de produtos, definindo seus atributos e seus respectivos tipos. Se tiver dúvidas, consulte o material das aulas anteriores que lhe ensinaram a modelar um banco de dados. Mãos a obra!

Inserir dados nas tabelas

Inserir dados em uma tabela significa preencher as linhas de uma tabela com dados correspondentes aos tipos determinados durante a criação da tabela. A sintaxe para incluir dados em uma tabela é descrita no quadro a seguir.

```
1 mysql>INSERT INTO nome_da_tabela (atributo1, atributo2, ...)
2   VALUES (valor1, valor2, ...);
```

Os dados do tipo **CHAR**, **VARCHAR**, **DATE**, **TIME** (texto em geral, ainda que seja apenas um caractere como A ou B) devem ser representados entre aspas simples (' '). Os dados do tipo **INT** ou **NUMERIC** não são representados por aspas simples. Observe ainda que as casas decimais dos números devem ser separadas por pontos ao invés de vírgulas, e os valores do tipo **VARCHAR** podem conter acentos e espaços em branco.

É importante ressaltar que os valores valor1, valor2, ..., seguem a mesma ordem dos atributos listados na primeira linha do comando **INSERT**. Essa lista de atributos é usada para indicar os atributos da tabela que devem ser preenchidos e com que valores. Essa lista é obrigatória quando alguns campos não forem preenchidos, ou quando a ordem dos valores informados for diferente da ordem definida na criação da tabela.

Vamos praticar o comando **INSERT** adicionando dados na tabela **clientes** criada anteriormente?

```
1 mysql>INSERT INTO clientes ( cli_codigo, cli_nome, cli_cpf, cli_data_nasc, cli_sexo, cli_email)
2   VALUES ( 1, 'José da Silva', '123456789-10', '1980-12-10', 'M', 'joseSilva@cursoSQL.com');
3
4 mysql>INSERT INTO clientes (cli_codigo, cli_email, cli_nome, cli_cpf, cli_data_nasc, cli_sexo)
5   VALUES (2, 'mariaSilva@cursoSQL.com', 'Maria da Silva', '012345678-99', '1982-02-28', 'F');
6
7 mysql>INSERT INTO clientes
8   VALUES ( 3, 'Francisco da Silva', '109876543-21', '1990-01-01', 'M', 'franciscoSilva@cursoSQL.com');
9
10 mysql>INSERT INTO clientes (cli_codigo, cli_nome, cli_sexo, cli_email)
11   VALUES (4, 'Francisca da Silva', 'F', 'franciscaSilva@cursoSQL.com');
```

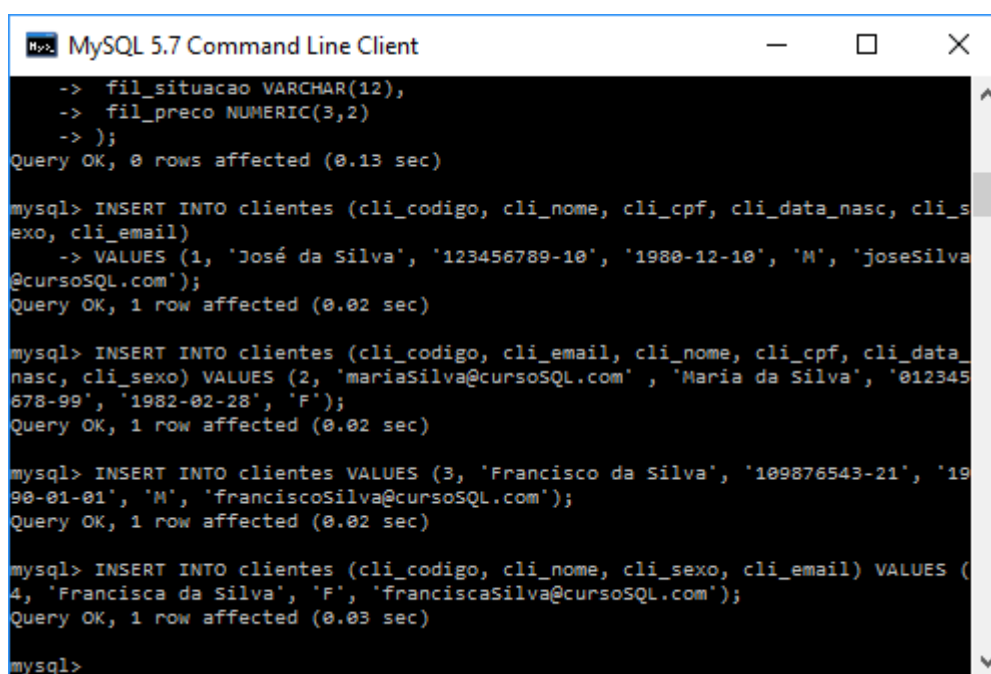
Observe com cuidado as duas primeiras inserções de dados na tabela. Percebeu a ordem dos atributos? Agora olhe para os valores, eles estão na mesma ordem dos atributos listados. Desde que os valores informados sejam referentes aos respectivos atributos listados, a ordem utilizada no **INSERT** não importa. Agora, analise a terceira inserção de dados na tabela **clientes**. A lista de atributos foi omitida no comando **INSERT**. Nesse caso, os valores devem estar todos ali, e na mesma ordem que os atributos foram definidos durante a criação da tabela. No último exemplo de inserção, alguns valores não foram inseridos. Como o seu sistema **SQL** não sabe a que atributos esses valores pertencem, você deve

especificá-los na lista de atributos (nesse exemplo são as colunas cli_codigo, cli_nome, cli_sexo e cli_email indicadas dentro dos parentes logo após o nome da tabela clientes).

Você pode estar se perguntando o que acontece com os atributos que não tiveram seus campos preenchidos com o comando INSERT. Nesse caso, os campos não informados serão preenchidos com NULL. Um valor NULL é um valor indefinido.

A mensagem “QUERY OK, 1 ROW AFFECTED” fornecida pelo sistema ao comando INSERT INTO **clientes** nos informa que uma linha de dados foi corretamente inserida na tabela **clientes**, conforme é ilustrado na **Figura 4**.

Figura 04 - Tela do MySQL após os comandos INSERT INTO **clientes**.



```
MySQL 5.7 Command Line Client
-> fil_situacao VARCHAR(12),
-> fil_preco NUMERIC(3,2)
-> );
Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO clientes (cli_codigo, cli_nome, cli CPF, cli_data_nasc, cli_s
exo, cli_email)
-> VALUES (1, 'José da Silva', '123456789-10', '1980-12-10', 'M', 'joseSilva
@cursoSQL.com');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO clientes (cli_codigo, cli_email, cli_nome, cli CPF, cli_data_
nasc, cli_sexo) VALUES (2, 'mariaSilva@cursoSQL.com', 'Maria da Silva', '012345
678-99', '1982-02-28', 'F');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO clientes VALUES (3, 'Francisco da Silva', '109876543-21', '19
90-01-01', 'M', 'franciscoSilva@cursoSQL.com');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO clientes (cli_codigo, cli_nome, cli_sexo, cli_email) VALUES (
4, 'Francisca da Silva', 'F', 'franciscaSilva@cursoSQL.com');
Query OK, 1 row affected (0.03 sec)

mysql>
```

Fonte: MySQL 5.7



Vídeo 03 - Inserção de Registros

Atividade 03

1. Seu SGBD avisa quando algo está errado no seu código, mas às vezes a resposta é meio vaga. Examine a seguir os comandos INSERT e tente descobrir o que há de errado com o comando, em seguida digite no seu sistema e observe a mensagem exibida.
 - a. INSERT INTO filmes (fil_codigo, fil_titulo, fil_genero, fil_duracao, fil_situacao, fil_preco) VALUES (1, 'E o vento Levou', 'romance', 'alugado', 5.00);
 - b. INSERT INTO filmes (fil_codigo, fil_titulo, fil_genero, fil_duracao, fil_situacao) VALUES (2, 'O silêncio dos inocentes', 'policial', '0:02:00', 'disponível', 02.50);
 - c. INSERT INTO filmes VALUES (3, 'Procurando Nemo', 'animação', '0:01:40' 'alugado', 02.50);
 - d. INSERT INTO filmes (fil_codigo, fil_titulo, fil_genero, fil_situacao, fil_duracao) VALUES (4, 'Cidade de Deus', 'ação', 'disponível', 0:02:10);
2. Corrija os comandos do item anterior e insira de forma correta os dados na tabela filmes do banco de dados locadora.

Modificação de tabelas

Vamos supor que um cliente da nossa locadora, o Sr. José da Silva, mudou seu e-mail. Sendo assim, para que nosso banco de dados não fique desatualizado, devemos atualizar a nossa tabela **clientes**. Para isso, podemos utilizar o comando UPDATE (atualizar), que altera apenas os valores das colunas especificadas.

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

1	mysql>UPDATE nome_da_tabela
2	SET atributo = valor
3	WHERE condição

A palavra SET diz ao SGBD para alterar o atributo antes do sinal de igual para conter o dado cujo valor é especificado depois do sinal de igualdade. A cláusula WHERE é opcional, mas quando está presente diz ao sistema para mudar somente as linhas que tenham a condição atendida. **Se a condição não for informada, a atualização será realizada em toda a tabela, então, você deve tomar muito cuidado ao atualizar um campo sem a cláusula WHERE.**

Você pode utilizar o UPDATE com um único atributo ou com um conjunto de atributos, mas para isso é preciso adicionar mais pares de **atributo = valor** na cláusula SET. Não se esqueça de colocar vírgula após cada um dos pares de **atributo = valor**.

Vamos exercitar a utilização do comando UPDATE fazendo pequenas atualizações nas tabelas **clientes** e **filmes** do nosso banco de dados **locadora**.

Inicialmente, vamos mudar o e-mail do cliente com nome José da Silva para silvajose@cursosql.com do seguinte modo:

```
1 mysql>UPDATE clientes
2 SET cli_email = 'silvajose@cursosql.com'
3 WHERE cli_nome = 'José da Silva';
```

Veja que especificamos na cláusula WHERE que o e-mail a ser atualizado era do cliente José da Silva, caso não tivesse especificado, todos os e-mails cadastrados na tabela clientes teriam sido alterados.

Se quisermos alterar o valor cobrado para locação de um filme, por exemplo, oferecendo um desconto de R\$1,00 em todos os filmes, podemos usar a seguinte estrutura:

```
1 mysql>UPDATE filmes
2 SET fil_preco = fil_preco - 1;
```

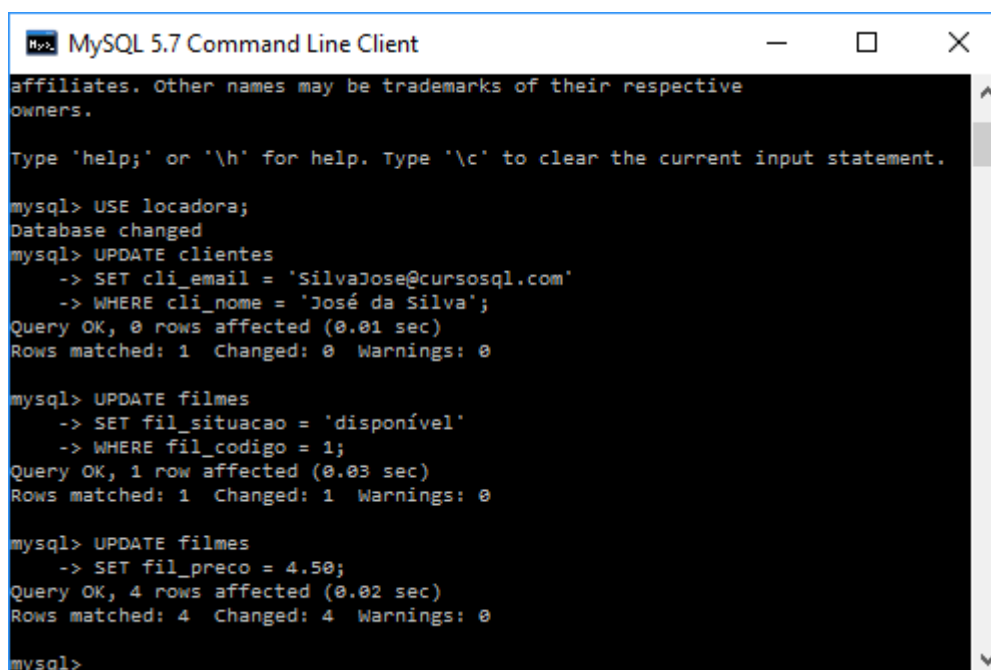
A estrutura (fil_preco = fil_preco - 1) é válida, pois fil_preco é um atributo numérico. Qualquer operação aritmética básica pode ser utilizada na construção desse tipo de estrutura (+ para adição, - para subtração, * para multiplicação e / para divisão).

Caso queira especificar o preço de todos os filmes para, por exemplo, R\$ 4,50, faça a atualização do seguinte modo:

```
1 mysql>UPDATE filmes
2 SET fil_preco = 4.50;
```

A mensagem fornecida pelo sistema ao comando UPDATE informa se o comando foi realizado com sucesso e quantas linhas da tabela foram alteradas, conforme é ilustrado na **Figura 5**.

Figura 05 - Tela do MySQL após diversos comandos UPDATE.



```
MySQL 5.7 Command Line Client
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> UPDATE clientes
  -> SET cli_email = 'SilvaJose@cursosql.com'
  -> WHERE cli_nome = 'José da Silva';
Query OK, 0 rows affected (0.01 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> UPDATE filmes
  -> SET fil_situacao = 'disponível'
  -> WHERE fil_codigo = 1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE filmes
  -> SET fil_preco = 4.50;
Query OK, 4 rows affected (0.02 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql>
```

Fonte: MySQL 5.7

O próximo comando a ser estudado é o DELETE, que exclui linhas simples ou linhas múltiplas dependendo da cláusula WHERE.

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

```
1 mysql>DELETE FROM nome_da_tabela
2 WHERE condição;
```

Você não pode utilizar o comando DELETE para apagar o valor de um atributo ou de uma porção de atributos, mas sim para apagar linhas simples ou múltiplas dependendo da cláusula WHERE. **Atenção: a cláusula WHERE é opcional no comando DELETE. Se não for informada, você pode excluir todas as linhas de**

uma tabela, mas não exclui a tabela do banco de dados. Para excluir a tabela inteira (dados e estrutura) do banco de dados você deve utilizar o comando DROP, o qual tem sua sintaxe descrita no quadro a seguir.

1	mysql>DROP TABLE nome_da_tabela;
---	----------------------------------

Exemplos

- Apagar cadastros de todos os clientes do sexo masculino:

1	mysql>DELETE FROM clientes
2	WHERE cli_sexo = 'M';

- Apagar cadastros de todos os filmes de terror:

1	mysql>DELETE FROM filmes
2	WHERE fil_genero= 'terror';

- Apagar o cadastro de todos os filmes:

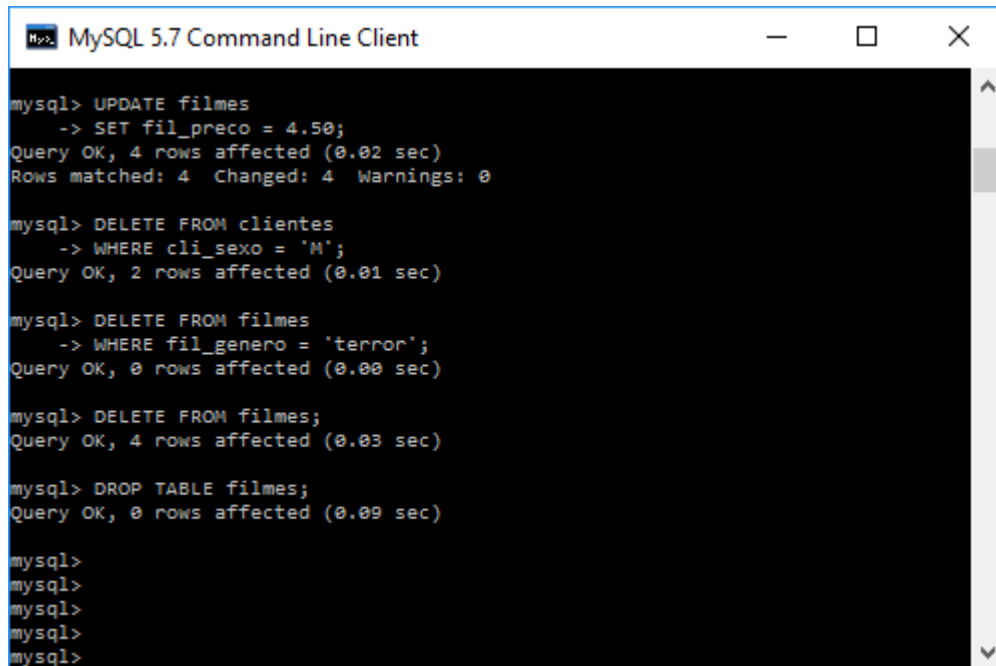
1	mysql>DELETE FROM filmes;
---	---------------------------

- Excluir a tabela filmes do banco de dados:

1	mysql>DROP TABLE filmes;
---	--------------------------

As respostas do SGBD, no caso o MySQL, aos comandos DELETE FROM e DROP TABLE são ilustradas na **Figura 6**.

Figura 06 - Tela do MySQL após os comandos DELETE FROM e DROP TABLE.



```
mysql> UPDATE filmes
-> SET fil_preco = 4.50;
Query OK, 4 rows affected (0.02 sec)
Rows matched: 4 Changed: 4 Warnings: 0

mysql> DELETE FROM clientes
-> WHERE cli_sexo = 'M';
Query OK, 2 rows affected (0.01 sec)

mysql> DELETE FROM filmes
-> WHERE fil_genero = 'terror';
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM filmes;
Query OK, 4 rows affected (0.03 sec)

mysql> DROP TABLE filmes;
Query OK, 0 rows affected (0.09 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
```

Fonte: MySQL 5.7



Vídeo 04 - Alteração e Exclusão de Registros

Atividade 04

1. A cláusula WHERE é opcional no comando UPDATE e DELETE. O que acontece se essa cláusula não for utilizada no comando UPDATE? E no comando DELETE?
2. Como utilizar o comando UPDATE para atualizar múltiplos atributos em várias linhas?
3. Qual a diferença entre o comando DELETE e o comando DROP?
4. Você pode utilizar o comando DELETE para deletar o valor de uma simples coluna? Por quê?

Conclusão

Concluimos por aqui nossa segunda aula sobre a linguagem SQL, mas temos um longo caminho pela frente. Na próxima aula, você vai conhecer o poderoso comando SELECT, que lhe permitirá o acesso as informações inseridas nas tabelas, possibilitando a construção das nossas tão desejadas consultas. Faça a autoavaliação com atenção e veja se precisa parar e refletir mais um pouco sobre o que estudamos. É uma boa escrever no seu caderno todos os comandos SQL (e respectivas funções) que está aprendendo para não esquecer.

Bons estudos e boa sorte!

Resumo

Nesta aula, você viu que para criar um banco de dados utilizamos o comando `CREATE DATABASE` e para entrar efetivamente no banco de dados é necessário utilizarmos o comando `USE`. 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. Todos os comandos aprendidos foram praticados na ferramenta MySQL através da criação e inserção de dados nas tabelas clientes e filmes do banco de dados locadora.

Autoavaliação

1. Crie um banco de dados chamado CursoX.
2. Nesse banco, crie as tabelas a seguir, de acordo com as informações do Dicionário de Dados (visto na Aula 3).

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: Disciplina

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

3. Escreva os comandos SQL necessários para inserir nessas tabelas as seguintes informações:

- dados de 10 alunos;
- dados de 5 disciplinas distintas;
- dados de 5 professores distintos.

4. Atualize a tabela disciplina, aumentando a carga horária de cada disciplina em 10 horas/ aulas.

5. Apague da tabela **disciplina** todas as disciplinas ministradas por um determinado professor.

Referências

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

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