

## ***Bases de Dados – Prof. Eduardo Corrêa***

### ***Aula 04 – Modificação de Dados na Linguagem SQL***

Na última aula, abordamos a definição de esquemas no através da SQL. Nesta aula, apresentaremos o conjunto de operações que permitem a **modificação da instância** de uma relação. São três operações básicas:

- **INSERT** : insere uma tupla;
- **DELETE**: exclui uma ou mais tuplas;
- **UPDATE**: atualiza os valores de alguns atributos de tuplas existentes.

Estas três instruções fazem parte da DML SQL. Serão apresentados exemplos de diversas formas de utilização destas instruções utilizando a base de dados do “minimundo IMDb”, criada na aula anterior (“Exemplo1.db”).

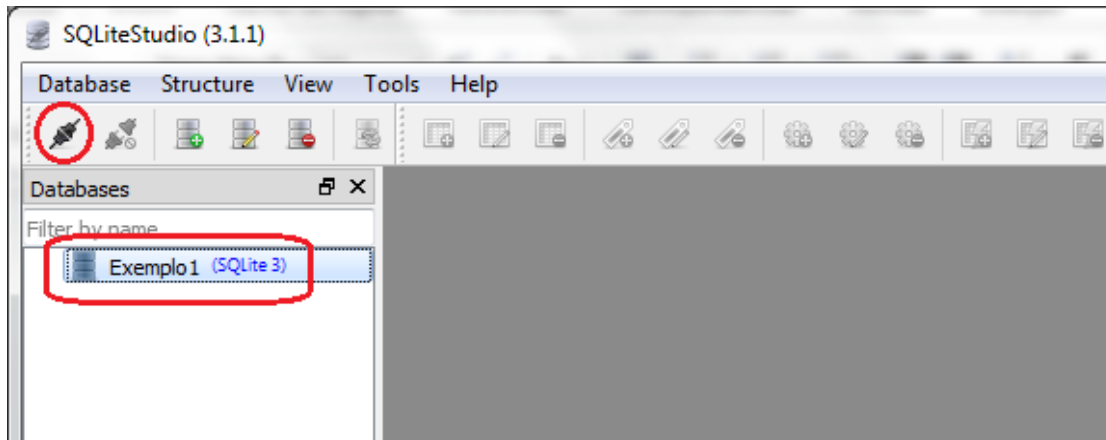
Adicionalmente, a aula aborda a **modificação de esquemas**, apresentando duas novas instruções DDL SQL: DROP TABLE e ALTER TABLE.

## **1. MODIFICAÇÃO DE DADOS: INSERT, DELETE e UPDATE**


### **PRÁTICA**

#### **PASSO 1 – Realizar a conexão com a Base de Dados “Exemplo1.db”**

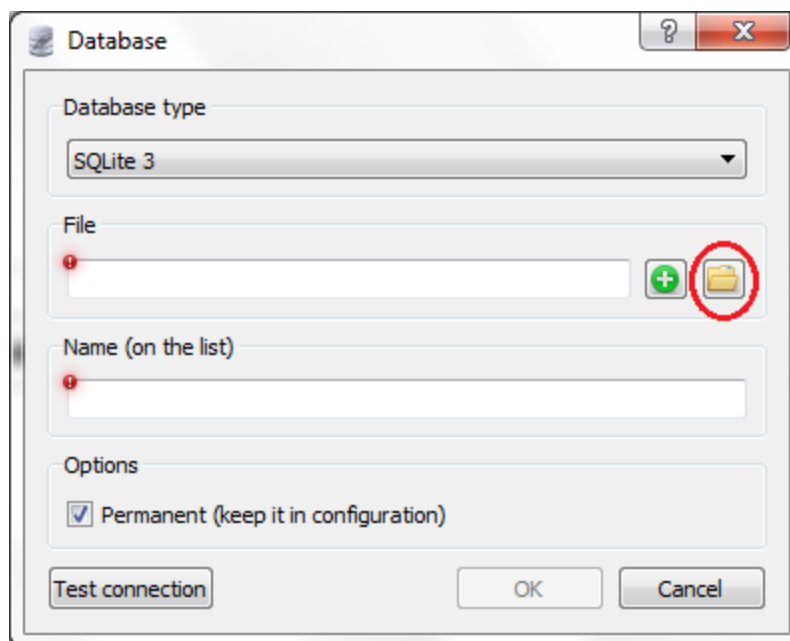
Abra o SQLite Studio e realize a conexão com o base de dados “Exemplo1.db” (BD criado na Aula 03). Caso esta base já esteja configurada no SQLite Studio, bastará selecioná-la e depois clicar no ícone de conexão, como mostra a Figura 1.



**Figura 1. Conexão com a base de dados Exemplo1**

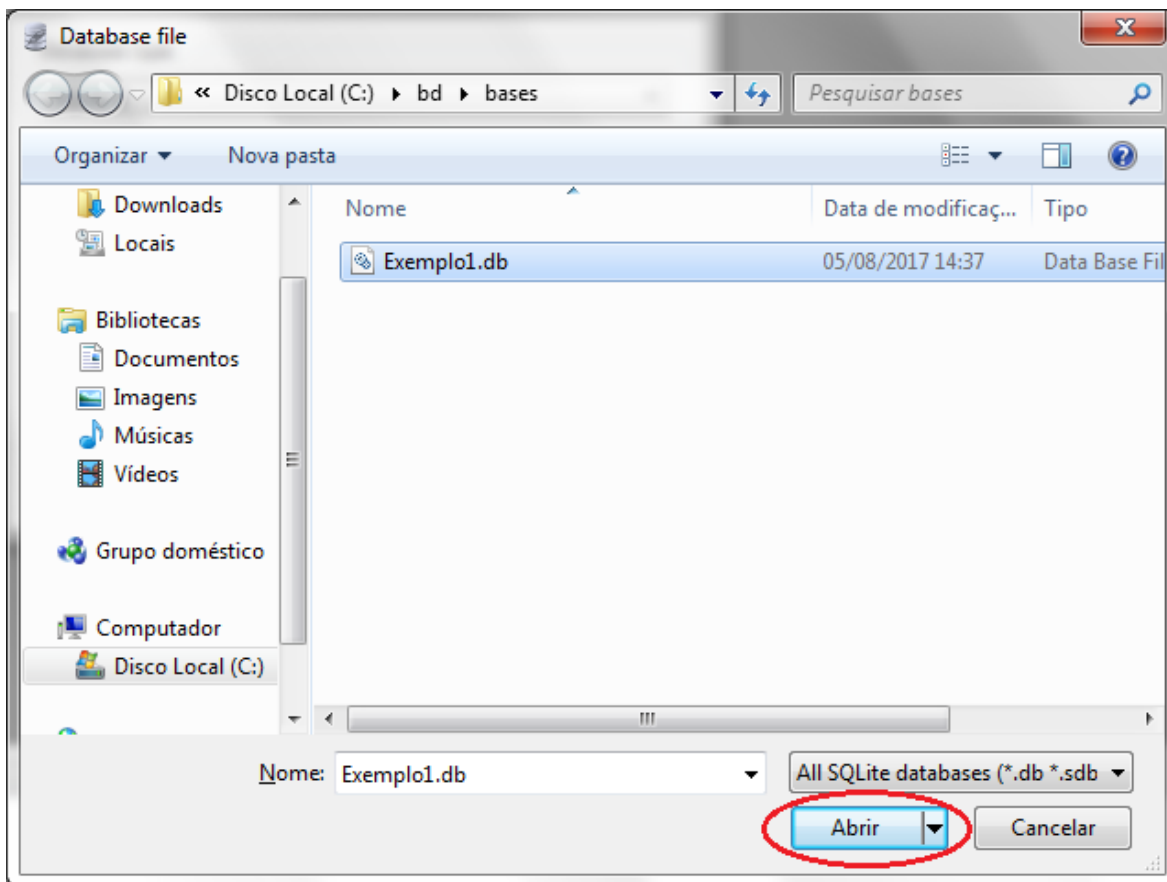
**Caso contrário**, será preciso adicionar a base ao SQLite Studio. Para fazer isso, escolha a opção **Database / Add a Database** ou clique no terceiro ícone da barra de ferramentas: 

Na nova janela, selecione o botão destacado na Figura 2.



**Figura 2. Criando uma conexão com um BD (parte 1)**


Navegue até a pasta onde o arquivo “Exemplo1.db” foi gravado e selecione o arquivo. Então, clique em “Abrir” (Figura 3). A seguir faça a conexão com a base conforme mostrado na primeira figura dessa aula.



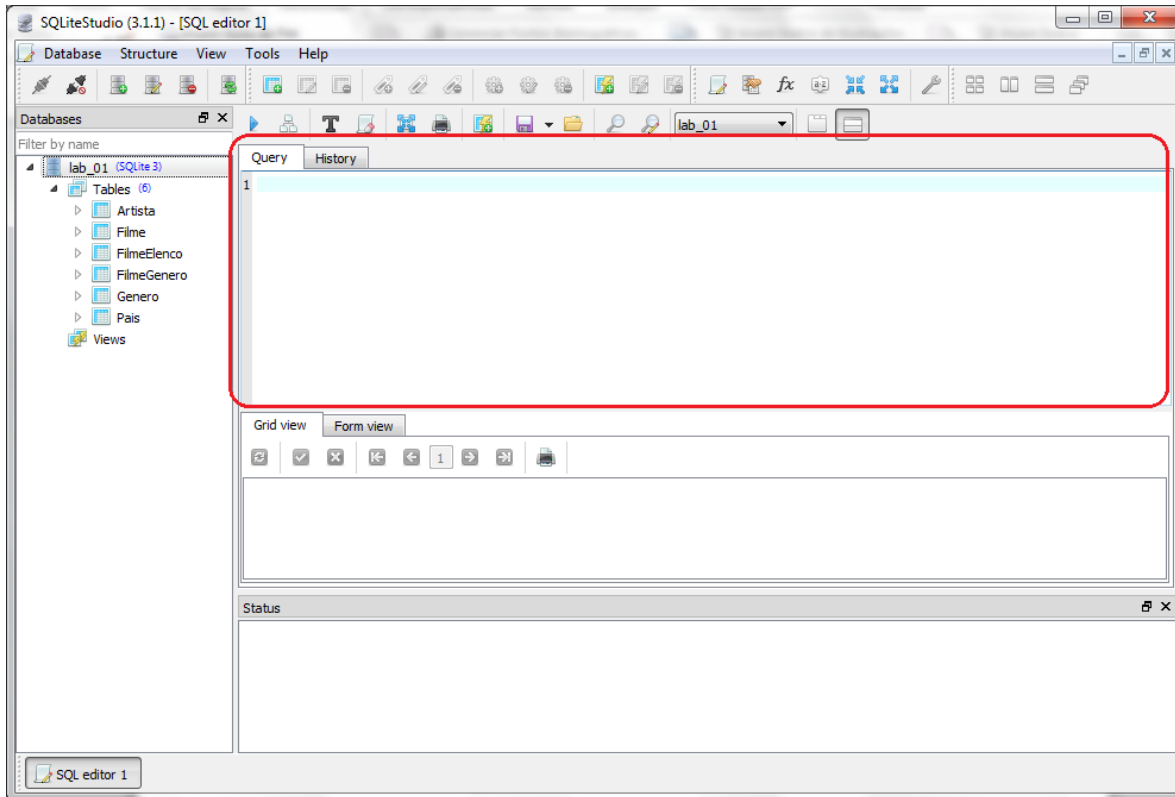
**Figura 3. Criando uma conexão com um BD (parte 2)**

## **PASSO 2 – Inserindo Registros com o comando INSERT**

A base de dados “Exemplo1” possui 6 tabelas: *Artista*, *Filme*, *FilmeElenco*, *FilmeGenero*, *Genero* e *Pais*.

Vamos agora utilizar o comando INSERT, que faz parte da DML SQL, para inserir registros em diversas tabelas desse BD<sup>1</sup>. Caso a janela de execução de comandos SQL não esteja aberta (Figura 4), abra a mesma selecionando a opção **Tools / Open SQL Editor** ou clicando no botão: 

<sup>1</sup>Ao longo do curso, utilizaremos os termos “base de dados” e “banco de dados” de forma intercambiável.



**Figura 4. Janela de Edição de Comandos SQL**

## **PASSO 2: Inserindo Registros com o Comando INSERT**

Inicialmente, utilizaremos o comando INSERT para inserir registros na tabela *Filme*. Este comando possui a sintaxe apresentada a seguir:

```
INSERT INTO tabela (coluna1, ..., colunan)  
VALUES (valor1, ..., valorn);
```

Na definição acima:

- *tabela*: representa o nome da tabela que vai receber o novo registro.
- *coluna*<sub>1</sub>, ..., *coluna*<sub>*n*</sub>: é uma lista com os nomes das colunas que serão afetadas (entre parênteses). Essa lista pode ser omitida caso todas as colunas estejam envolvidas na operação.
- *valor*<sub>1</sub>, ..., *valor*<sub>*n*</sub>: é uma lista com os valores que serão inseridos em cada coluna (deve respeitar a ordem estabelecida em *coluna*<sub>1</sub>, ..., *coluna*<sub>*n*</sub>).

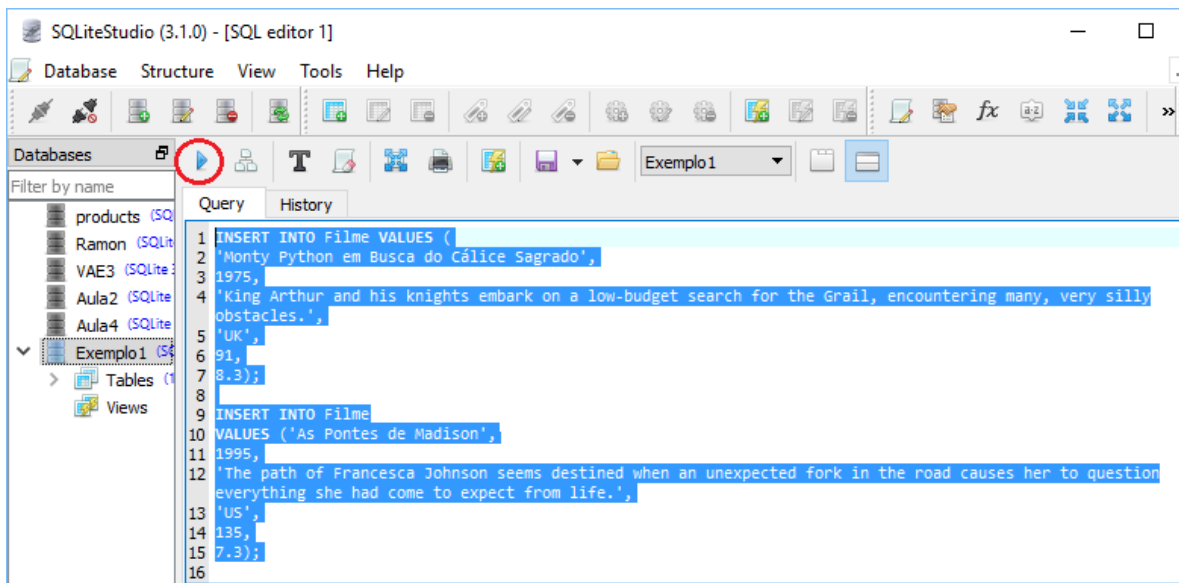
EXEMPLOS:

(a) **Inserindo dois filmes com informações completas (valores para todas as colunas).**

```
INSERT INTO Filme VALUES (  
'Monty Python em Busca do Cálice Sagrado',  
1975,  
'King Arthur and his knights embark on a low-budget search for the  
Grail, encountering many, very silly obstacles.',  
'UK',  
91,  
8.3);
```

```
INSERT INTO Filme VALUES ('As Pontes de Madison',  
1995,  
'The path of Francesca Johnson seems destined when an unexpected  
fork in the road causes her to question everything she had come to  
expect from life.',  
'US',  
135,  
7.3);
```

Após digitar estes comandos na janela de edição SQL, selecione-os com o mouse e clique no botão **Execute Query** conforme mostra a Figura 5. Você também pode selecionar os comandos e clicar em **F9**.



**Figura 5. Seleção e execução de dois comandos INSERT**

Observações:

- Como inserimos filmes contendo informações completas (valores para todos os campos da tabela) não foi preciso especificar os nomes das colunas no comando INSERT. Bastou especificar diretamente os valores, respeitando a ordem em que as colunas especificadas no CREATE TABLE da tabela *Filme* (esta ordem é: “título”, “ano”, “resumo”, “pais”, “duração” e “avaliacao”).
- Para inserir dois registros a abordagem mais comumente utilizada através de dois comandos INSERT. Porém, há uma sintaxe que permite inserir vários registros com um único INSERT, como veremos ainda nesta aula.

Para visualizar se a operação foi bem sucedida, digite e execute o seguinte comando:

```
SELECT * FROM Filme;
```

Obs.: No SQLite Studio, você pode executar o comando acima de diferentes formas: (i) apagando os comandos INSERT digitados anteriormente; (ii) abrindo uma nova janela de edição SQL; (iii) sem apagar os comandos INSERT, digitando o SELECT na mesma janela e depois o selecionando com o mouse para em seguida executá-lo. O mesmo procedimento poderá ser feito para todos os exemplos posteriores desta apostila.

O comando apresentado serve para recuperar<sup>2</sup> a instância (conteúdo corrente) de *Filme*. O resultado do SELECT é mostrado a seguir:

```
SELECT * FROM Filme
```

TITULO	ANO	RESUMO	PAIS	DURACAO	AVALIACAO
Monty Python em busca...	1975	King Arthur...	UK	91	8.3
As Pontes de Madison	1995	The path of...	US	135	7.3

### (b) Inserindo filmes com informações parciais

-- insere um filme sem resumo e sem duração.

```
INSERT INTO Filme (titulo, ano, pais, avaliacao)  
VALUES ('Contato de Risco', 2003, 'US', 2.4);
```

<sup>2</sup> O termo recuperar costuma ser bastante utilizado na área de BD. Quando você utiliza um comando para recuperar informações – o comando SELECT – isto significa que você está trazendo uma ou mais linhas e colunas de interesse de tabela(s) do BD. Estas poderão então ser visualizadas (como ocorre quando usamos o SQLite Studio) ou trabalhadas (como ocorre quando estamos criando um programa que acessa o BD).

-- *insere filmes apenas contendo título, ano e país.*

```
INSERT INTO Filme (titulo, ano, pais)
VALUES ('A Família Bélier', 2014, 'FR');
```

```
INSERT INTO Filme (titulo, ano, pais)
VALUES ('Caça-Fantasmas', 1984, 'US');
```

```
INSERT INTO Filme (titulo, ano, pais)
VALUES ('Caça-Fantasmas', 2016, 'US');
```

**(c) Inserindo um filme utilizando a seguinte ordem das colunas: ano, país, duração e título.**

-- *inserindo filme com ano, país, duração e título*

```
INSERT INTO Filme (ano, pais, duracao, titulo)
VALUES (2006, 'FR', 94, 'Meu melhor amigo');
```

**(d) Inserindo um filme cujo título tem apóstrofo (aspas simples).**

-- *é necessário colocar dois apóstrofes para contornar essa situação*

```
INSERT INTO Filme (titulo, ano, pais)
VALUES ('Joana D'arc', 1999, 'FR');
```

**(e) Tentativa de violar a chave primária.**

Agora tente executar o comando abaixo. Observe que ele tenta inserir uma segunda linha para o filme ‘As Pontes de Madison’ de 1995.

```
INSERT INTO Filme (titulo, ano, resumo, pais)
VALUES ('As Pontes de Madison', 1995, 'Um ótimo filme.', 'US');
```

Veja que a execução falhará!

[12:03:27] Error while executing SQL query on database 'Exemplo1': UNIQUE constraint failed: Filme.titulo, Filme.ano

Ao tentar inserir pela segunda vez um filme com o “título” igual a 'As Pontes de Madison' e o “ano” igual 1995, há **violação da restrição de chave primária**<sup>3</sup>. O SQLite

<sup>3</sup> Veja que, de fato, a mensagem exibida é “UNIQUE constraint failed”. Uma restrição do tipo UNIQUE, quando aplicada a uma ou mais colunas, garante que os dados adicionados a estas colunas serão únicos em cada linha. Toda coluna definida como PK é também uma coluna UNIQUE. Mas nem toda coluna UNIQUE precisa necessariamente ser uma PK. Você entenderá melhor esta situação quando estudar o conceito de **chave candidata** na Aula 05.

não permite que existam dois registros com o mesmo “título” e “ano” na tabela *Filme*, mesmo que o valor de qualquer um dos demais campos (“resumo”, “pais”, “duracao” e “avaliacao”) seja diferente.

#### **(e) Inserindo nas demais tabelas**

##### *--Pais*

```
INSERT INTO Pais VALUES('US','Estados Unidos');
INSERT INTO Pais VALUES('UK','Reino Unido');
INSERT INTO Pais VALUES('FR','França');
INSERT INTO Pais VALUES('BR','Brasil');
```

##### *--Artista*

```
INSERT INTO Artista VALUES('Graham Chapman','M');
INSERT INTO Artista VALUES('John Cleese','M');
INSERT INTO Artista VALUES('Eric Idle','M');
INSERT INTO Artista VALUES('Terry Gilliam','M');
INSERT INTO Artista VALUES('Terry Jones','M');
INSERT INTO Artista VALUES('Michael Palin','M');
INSERT INTO Artista VALUES('Jennifer Lopez','F');
INSERT INTO Artista VALUES('Clint Eastwood','M');
INSERT INTO Artista VALUES('Meryl Streep','F');
INSERT INTO Artista VALUES('Ben Affleck','M');
INSERT INTO Artista VALUES('Milla Jovovich','F');
```

##### *--Genero*

```
INSERT INTO Genero VALUES('Drama');
INSERT INTO Genero VALUES('Crime');
INSERT INTO Genero VALUES('Comédia');
INSERT INTO Genero VALUES('Aventura');
INSERT INTO Genero VALUES('Fantasia');
INSERT INTO Genero VALUES('Romance');
INSERT INTO Genero VALUES('Biografia');
INSERT INTO Genero VALUES('Sci-Fi');
```

##### *--Inserindo o Elenco do Filme “As Pontes de Madison”*

```
INSERT INTO FilmeElenco
VALUES('As Pontes de Madison',1995,'Meryl Streep');

INSERT INTO FilmeElenco
VALUES('As Pontes de Madison',1995,'Clint Eastwood');
```

##### *--Inserindo os Gêneros do Filme “As Pontes de Madison”*

```
INSERT INTO FilmeGenero
VALUES('As Pontes de Madison',1995,'Drama');

INSERT INTO FilmeGenero
VALUES('As Pontes de Madison',1995,'Romance');
```



### ATENÇÃO

O comando INSERT, conforme definido originalmente na linguagem SQL, **insere apenas uma linha por vez**. Ou seja: cada INSERT insere uma linha na tabela.

No entanto, atualmente a maioria dos SGBDs oferece uma sintaxe que permite a inclusão de múltiplas linhas em um único INSERT. O SQLite, por exemplo, suporta a sintaxe apresentada abaixo:

```
INSERT INTO Pais VALUES  
( 'IT', 'Itália' ),  
( 'AR', 'Argentina' ),  
( 'NZ', 'Nova Zelândia' );
```

Neste exemplo, três linhas são inseridas de uma só vez na tabela *Pais* com o uso de apenas um comando INSERT.

### **PASSO 4 – Excluindo Registros com o Comando DELETE**

O comando DELETE é utilizado quando desejamos excluir uma ou mais linhas de uma tabela. Este comando possui a sintaxe apresentada a seguir:

```
DELETE FROM tabela WHERE condicao;
```

Na definição acima:

- *tabela*: nome da tabela alvo.
- *condição*: teste realizado sobre uma ou mais colunas, que define as linhas que serão excluídas.
- 

### **EXEMPLO:**

#### **(a) Excluindo o Filme “Caça-Fantasmas” de 1984**

```
DELETE FROM Filme  
WHERE titulo = 'Caça-Fantasmas' AND ano = 1984;
```

Observe que:

- Utilizamos a chave primária completa da tabela (título + ano) para especificar que desejávamos excluir apenas o filme ‘Caça-Fantasmas’ de 1984, mantendo o filme de 2016 no BD. Se tivéssemos colocado apenas o título na cláusula WHERE, os dois filmes de título ‘Caça-Fantasmas’ teriam sido excluídos.
- A condição especificada na cláusula WHERE pode ser montada da forma que você desejar, o que inclui o uso dos operadores booleanos AND, OR e NOT para possibilitar testes em diversas colunas. O comando DELETE irá obedecer exatamente aquilo que for colocado nesta cláusula.
  - **Exemplo:** se você especificar `DELETE FROM Filme WHERE ano > 2012`, todos os filmes produzidos de 2013 em diante serão excluídos.

### **CUIDADO!!!**

Um comando DELETE sem uma condição associada irá **apagar todas as linhas** da tabela especificada (ex.: `DELETE FROM Filme;`). Portanto, é preciso ter muito cuidado ao utilizar esse comando. Ele deve ser usado apenas se nossa intenção é realmente esvaziar a tabela!

Também é importante observar que o comando DELETE não apaga a definição da tabela (ou seja, não é capaz de remover a tabela do BD). O DELETE apaga apenas linhas da mesma. Para remover definitivamente uma tabela utiliza-se o comando DROP (comando do tipo DDL) que será apresentado no final desta aula.

## **PASSO 4 – Atualizando Registros com o Comando UPDATE**

O comando UPDATE é utilizado quando desejamos modificar os valores de uma ou mais colunas de uma tabela. A operação pode afetar uma ou diversas linhas. Este comando possui a sintaxe apresentada a seguir:

```
UPDATE tabela
SET coluna1=valor1, ..., colunan=valorn
WHERE condição;
```

Na definição acima:

- *tabela*: nome da tabela que terá registros atualizados.
- *coluna*<sub>*i*</sub>=*valor*<sub>*i*</sub>: nome de uma coluna que será modificada e o novo valor que será a ela atribuído.

- *condição*: teste realizado sobre uma ou mais colunas, que define as linhas que serão atualizadas. Funciona da mesma forma que a condição da instrução DELETE.

### EXEMPLOS:

#### **(a) Modificando a avaliação do filme “As Pontes de Madison”**

```
UPDATE Filme  
SET avaliacao = 10  
WHERE titulo = 'As Pontes de Madison' AND ano=1995;
```

Utilizamos a chave primária completa da tabela (“titulo” + “ano”), mas nesse exemplo específico isso não seria preciso (bastaria fazer o WHERE em cima do título), pois só existe um filme cadastrado na base com o título ‘As Pontes de Madison’.

#### **(b) Modificando o país, avaliação e resumo do filme “Joana D’Arc”**

```
UPDATE Filme SET  
pais = 'US',  
avaliacao = 6.4,  
resumo = 'A young girl receives a vision that drives her to rid  
France of its oppressors.'  
WHERE titulo = 'Joana D''arc' AND ano=1999;
```

Neste exemplo, o valor do atributo “pais” estava anteriormente cadastrado como 'FR' e foi trocado para 'US'. O resumo e a avaliação estavam NULL (nulos) e foram modificados para os novos valores especificados. A duração continua com valor NULL, pois esta coluna não foi especificada no comando UPDATE.

### **CUIDADO!!!**

Um comando UPDATE sem parâmetro irá modificar todas as linhas da tabela especificada. Portanto, é preciso tomar muito cuidado ao usar o comando. Por exemplo, execute o seguinte comando: `UPDATE Filme SET pais = 'BR';`

Após a execução do mesmo, todos filmes a ficarão com o valor 'BR' na coluna “pais”.

## **EXERCÍCIOS PROPOSTOS**

1. Atualize a tabela *Filme*, colocando novamente FR como país do filme Joana D’arc

2. Insira o elenco e os gêneros do filme 'Monty Python em Busca do Cálice Sagrado'. Os artistas são os 6 primeiros que foram inseridos na tabela de Artista e os gêneros são 'Comédia', 'Aventura' e 'Fantasia'.
3. Apague todas as informações referentes ao filme "As Pontes de Madison" da base de dados (OBS: isso significa remover não apenas o filme, mas seu elenco e gêneros).

## 2. MODIFICAÇÃO DE ESQUEMAS: DROP TABLE e ALTER TABLE

Acabamos de aprender os comandos SQL para modificar o conteúdo de tabelas. Já tínhamos aprendido anteriormente a declarar esquemas usando SQL. Mas e se precisássemos alterar o esquema de uma tabela que já tenha sido definida, que esteja sendo utilizada há algum tempo e que já contenha diversas linhas?

Na prática vários tipos diferentes tipos de modificação de esquemas podem ser necessárias:

- Remover uma tabela (ex.: ela não precisa ser mais usada).
- Renomear uma tabela ou coluna.
- Adicionar uma nova coluna (a mais comum das alterações de esquema).
- Remover uma coluna existente.

Embora a modificação de esquemas seja menos comum do que a modificação de dados, eventualmente torna-se necessário modificar a definição de uma tabela. A prática a seguir apresenta os comandos SQL que realizam esse tipo de operação.

## PRÁTICA

### PASSO 1 – Removendo uma Tabela

Conforme mencionado anteriormente, um comando DELETE sem WHERE apaga todas as linhas de uma tabela. Por exemplo, se você executar o comando `DELETE FROM Pais;` acabará apagando todas as linhas da tabela *Pais*.

No entanto, mesmo com a utilização deste comando, a tabela *Pais* continuará existindo, pois DELETE é um comando DML, o que significa que ele atua apenas sobre dados das tabelas, mas nunca sobre a definição das mesmas. Para remover a tabela *Pais* do BD, é necessário utilizar o comando **DROP TABLE**:

```
DROP TABLE Pais;
```

Execute o comando acima. Para confirmar que a tabela foi realmente excluída, utilize o comando:

```
SELECT * FROM Pais;
```

O SQLite Studio apresentará a seguinte informação na janela de status:

Error while executing SQL query on database 'lab\_01': no such table: Pais

### **DROP TABLE – Sintaxe na SQL**

```
DROP TABLE tabela;
```

Na definição acima:

- *tabela*: nome da tabela que terá registros atualizados.

#### **CUIDADO!!!**

O comando DROP TABLE apaga a tabela do BD **mesmo que ela contenha dados**. Ao executar essa instrução a tabela especificada não mais fará parte do esquema do banco de dados e todos os seus registros serão eliminados.

### **PASSO 2 – Renomeando uma Tabela**

Mais frequentemente do que apagar uma tabela que é parte de um BD existente, podemos precisar modificar o esquema de uma tabela existente. Essas modificações são realizadas com o uso da instrução ALTER TABLE.

A instrução ALTER TABLE pode ser utilizada para fazer diversos tipos de alterações em uma tabela já criada. O exemplo abaixo mostra como renomear uma tabela no SQLite (Obs.: para outros SGBDs a sintaxe pode ser diferente). Neste exemplo, o nome da tabela *FilmeElenco* é alterado para *Elenco*.

```
ALTER TABLE FilmeElenco RENAME TO Elenco;
```

## **ALTER TABLE para renomear tabela – Sintaxe na SQL**

```
ALTER TABLE nome_ant TO nome_novo;
```

Na definição acima:

- *nome\_ant*: nome da tabela que será renomeada.
- *nome\_novo*: novo nome que será dado para a tabela

## **PASSO 3 – Inserindo uma Nova Coluna em uma Tabela**

Suponha que o sistema do IMDb evoluiu e que a partir de agora apresentará aos usuários a arrecadação obtida por cada filme. Nesse caso, será preciso acrescentar uma nova coluna do tipo numérica na tabela *Filme*. Essa operação também é realizada com o uso da instrução ALTER TABLE.

```
ALTER TABLE Filme ADD COLUMN arrecadacao NUM;
```

A nova coluna é inserida como a **última coluna** da tabela e com o valor NULL (ausente) em todos os registros. É possível também adicionar uma coluna com um valor default. Para isso, basta utilizar a palavra DEFAULT após o tipo da nova coluna. No exemplo abaixo, adicionamos a coluna “tipo” na tabela *Artista*. Imagine que esta coluna será utilizada para indicar se um artista é um ATOR/ATRIZ (tipo='A') ou um DIRETOR (tipo='D'). Como atualmente a tabela contém apenas ATORES e ATRIZES, podemos adicionar essa coluna especificando que o valor DEFAULT será “A”.

```
ALTER TABLE Artista ADD COLUMN tipo TEXT DEFAULT 'A';
```

**IMPORTANTE:** sempre que um comando INSERT for executado sobre a tabela *Artista*, caso o valor do campo “tipo” não seja especificado, será assumido que o mesmo é o valor DEFAULT (ou seja, “A”).

```
INSERT INTO Artista (nome, sexo) VALUES ('Alice Braga','F');
```

Para inserir um diretor, você terá que explicitar o valor “D” para o campo “tipo” no comando INSERT:

```
INSERT INTO Artista (nome, sexo, tipo)  
VALUES ('Damien Chazelle','M','D');
```

**ALTER TABLE para incluir coluna – Sintaxe na SQL**

```
ALTER TABLE tabela ADD COLUMN nome_coluna tipo_coluna;
```

Na definição acima:

- *tabela*: nome da tabela que receberá a nova coluna.
- *nome\_coluna*: nome da coluna que será inserida.
- *tipo\_coluna*: tipo da coluna que será inserida (ex.: TEXT, INT, NUM).
  - Opcionalmente, pode ser atribuído um valor default para a coluna, com o uso da palavra-chave DEFAULT seguida do valor.

**REMOVENDO COLUNAS**

Em alguns SGBDs, para remover uma coluna, utiliza-se:

```
ALTER TABLE tabela DROP nome_coluna;
```

Ao executar essa instrução, a coluna chamada “nome\_coluna” será removida da tabela especificada. Obviamente, todos os dados armazenados na coluna em questão são perdidos!

Entretanto, o SQLite não suporta a exclusão de colunas.