

Bases de Dados

Módulo 8B: Modificação de dados com uso da SQL

Prof. André Bruno de Oliveira

Data 14/12/2023 19:54

Modelo Relacional – Modificação de dados com uso da SQL

Objetivo: Aula prática de como modificar os dados das relações com uso de SQL.

TÓPICOS

- Aprender comando básicos SQL
 - INSERÇÃO, EXCLUSÃO, ATUALIZAÇÃO
 - EXERCÍCIOS
 - ALTERAÇÃO DE ESQUEMAS
 - EXERCÍCIOS

INTRODUÇÃO

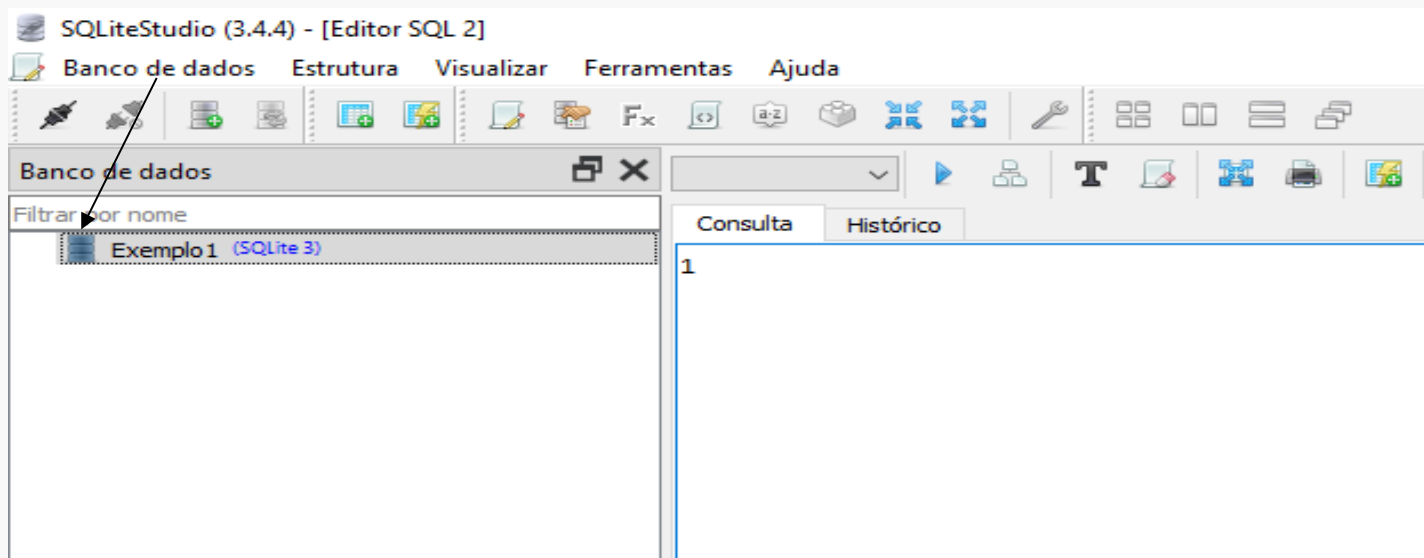
- Na aula anterior, abordamos a definição de esquemas com uso da SQL. Nesta aula, veremos um conjunto de operações que permitem a **modificação da instância (conjunto de linhas denominadas tuplas ou registro)** de uma relação. São três operações básicas da DML SQL:
- **INSERT**: insere uma tupla ou registro ou linha.
- **DELETE**: excluir um ou mais registros de acordo com a condição.
- **UPDATE**: atualiza os valores dos atributos de acordo com a condição.
- Serão apresentados exemplos de diversas formas de utilização destas instruções utilizando a base de dados da “Censo_escolar”, 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**.

MODIFICAÇÃO DE DADOS: INSERT, DELETE e UPDATE

Etapa 1: Realize a conexão com o BD “Exemplo1.db”

- Abra o SQLite Studio e realize a conexão com o BD “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.

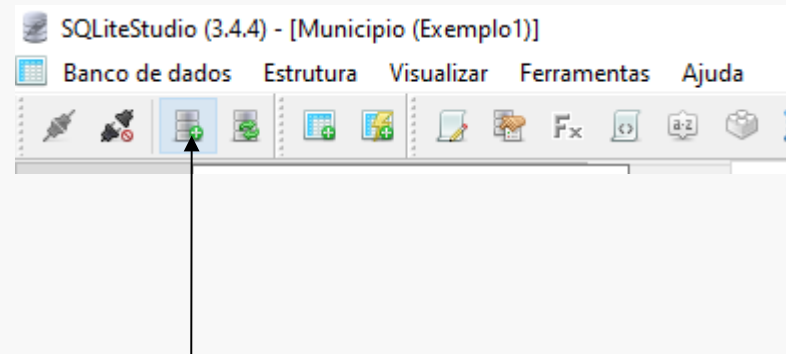
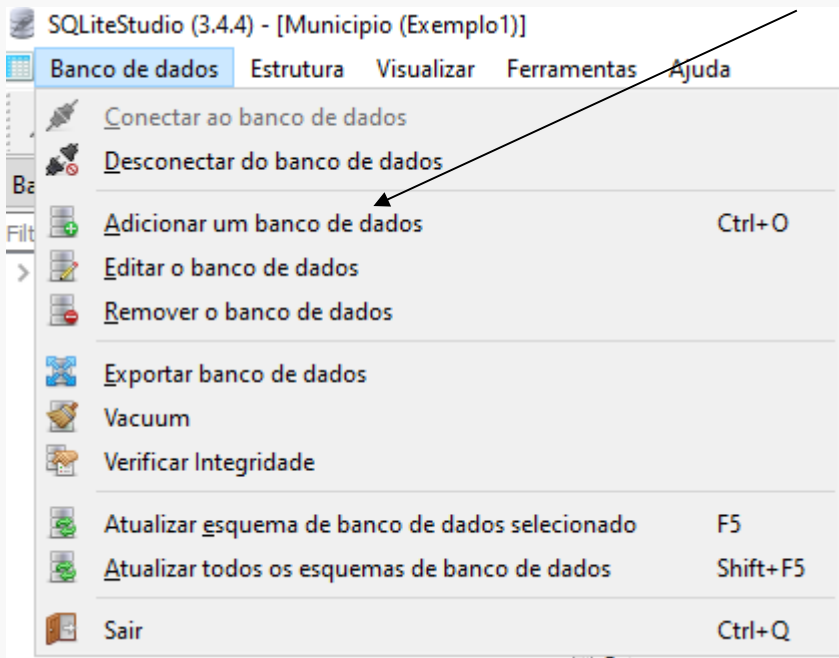
Figura 1 – Conexão com o banco de dados Exemplo1



MODIFICAÇÃO DE DADOS: INSERT, DELETE e UPDATE

- **Caso o BD não esteja adicionado**, será preciso adicioná-lo ao SQLite Studio. Para fazer isso, escolha a opção **Banco de dados** ou clique no terceiro ícone da barra de ferramentas.

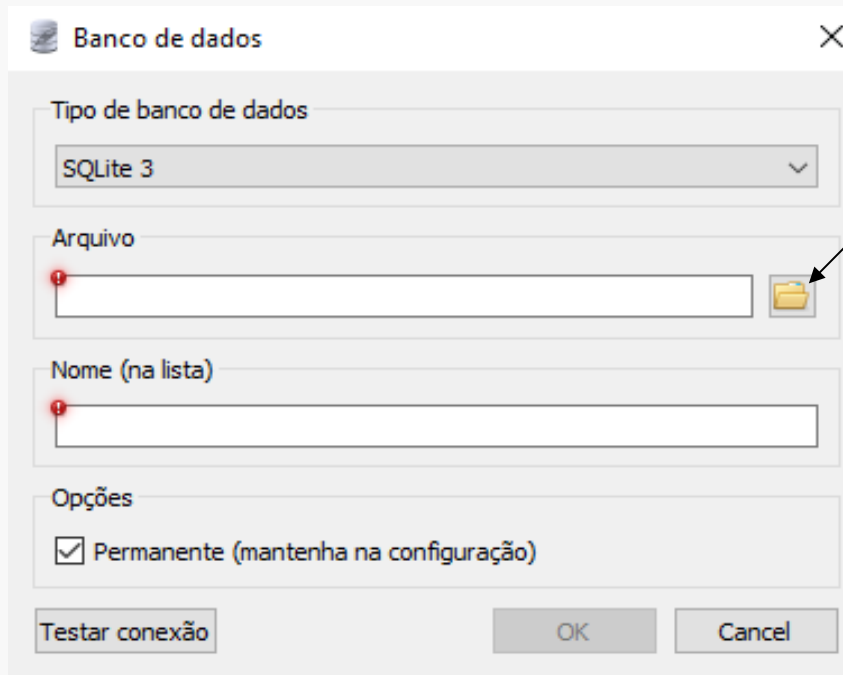
Figura 2 – Adicionar um BD



MODIFICAÇÃO DE DADOS: INSERT, DELETE e UPDATE

- Na janela que se abre (Figura 2.1), selecione o ícone de pasta para selecionar o local onde está o arquivo do BD.

Figura 2.1 – Criar conexão com um BD



Banco de dados

Tipo de banco de dados

SQLite 3

Arquivo

Nome (na lista)

Opções

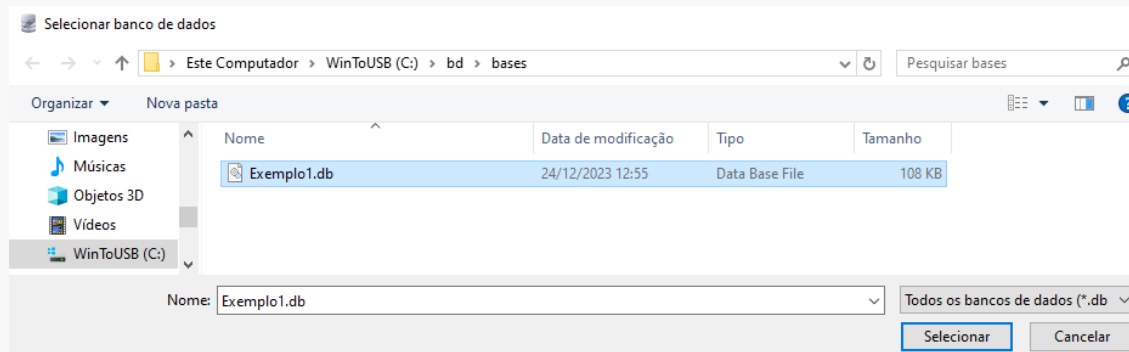
☒ Permanente (mantenha na configuração)

Testar conexão OK Cancel

MODIFICAÇÃO DE DADOS: INSERT, DELETE e UPDATE

- Na nova janela aberta (Figura 3), selecione o arquivo de banco de dados no local que está localizado. Em seguida clique em selecionar.

Figura 3 – Selecionar arquivo de BD

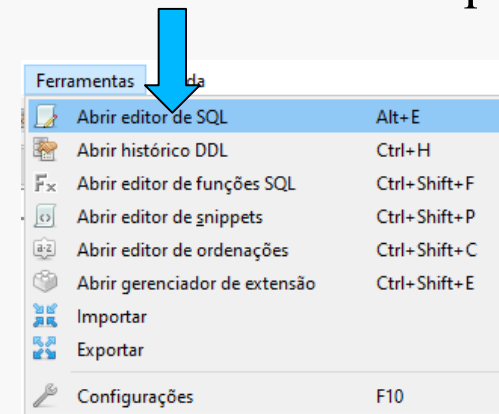
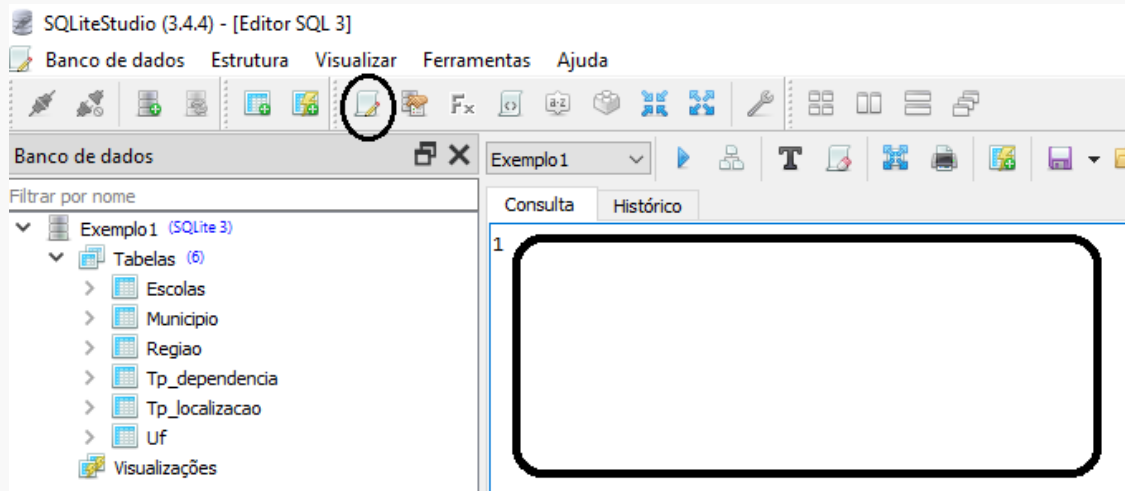


INSERIR REGISTROS

A base de dados “Exemplo1” possui 6 tabelas: Escolas, Municipio, Regiao, Tp_dependencia, Tp_localizacao e Uf.

- Vamos agora utilizar o comando INSERT, que faz parte da DML SQL, para inserir registros em diversas tabelas desse BD. Caso a janela de execução de comandos SQL não esteja aberta (Figura 4), abra-a selecionando a opção **Ferramentas=>Abrir Editor de SQL** ou clicando no ícone em destaque na figura 4.

Figura 4 – Aba de edição de SQL



INSERIR REGISTROS

Use o comando INSERT conforme a sintaxe descrita a seguir:

```
INSERT INTO tabela (coluna1, ..., colunan)
```

```
VALUES (valor1, ..., valorn);
```

Na especificação acima:

tabela: representa o nome da tabela que vai receber o novo registro.

*coluna*₁, ..., *coluna*_n: é uma lista com os nomes das colunas que serão afetadas (entre parênteses). Esta lista pode ser omitida caso todas as colunas estejam envolvidas na operação. Caso esta lista seja omitida, fica definido por padrão uma lista que segue a ordem dos atributos definidos na relação.

*valor*₁, ..., *valor*_n: é uma lista com os valores que serão inseridos em cada coluna (deve respeitar a ordem estabelecida na lista com *coluna*₁, ..., *coluna*_n).

INSERIR REGISTROS

EXEMPLOS

Inserir linhas na tabela Escola

--Inserindo três escolas com todas as colunas sem declarar a lista.

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia, tp_localizacao,  
qt_mat_bas_fem, qt_mat_bas_masc, qt_mat_bas, qt_doc_bas, cod_regiao,  
cod_uf, cod_municipio) VALUES (11002158, 'EMEF SAO FRANCISCO DE ASSIS',  
3, 1, 70, 58, 128, 7, 1, 11, 1100205);
```

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia, tp_localizacao,  
qt_mat_bas_fem, qt_mat_bas_masc, qt_mat_bas, qt_doc_bas, cod_regiao,  
cod_uf, cod_municipio) VALUES (11003391, 'EMEF MANOEL MACIEL NUNES',  
3, 2, 21, 22, 43, 2, 1, 11, 1100205);
```

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia, tp_localizacao,  
qt_mat_bas_fem, qt_mat_bas_masc, qt_mat_bas, qt_doc_bas, cod_regiao,  
cod_uf, cod_municipio) VALUES (35578678, 'CENTRO EDUCACIONAL  
INFANTIL ALMERINA PEREIRA DOS SANTOS', 3, 1, 54, 50, 103, 13, 3, 35,  
3548401);
```

INSERIR REGISTROS

Observações:

- Como inserimos escolas 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 foram especificadas no CREATE TABLE da tabela Escolas nesta ordem é:
 - co_entidade,
 - no_entidade,
 - cod_regiao,
 - cod_uf,
 - cod_municipio,
 - tp_dependencia,
 - tp_localizacao,
 - qt_mat_bas_fem,
 - qt_mat_bas_masc,
 - qt_mat_bas,
 - qt_doc_bas

INSERIR REGISTROS

Observações:

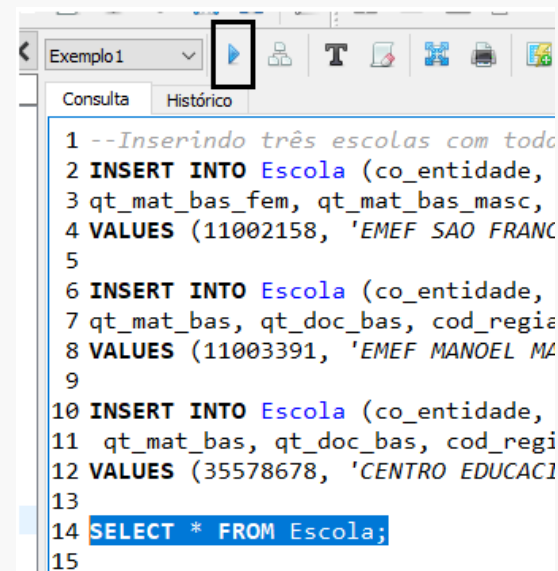
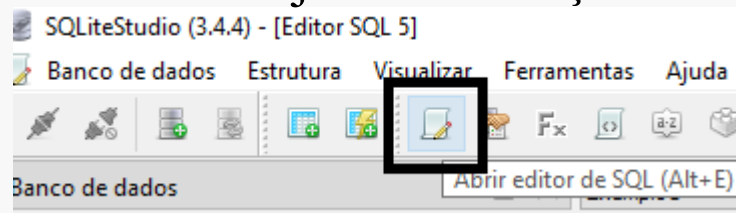
- Para inserir dois registros a sintaxe mais comum é através de dois comandos INSERT. Contudo, há uma sintaxe que permite inserir vários registros com um único INSERT, como veremos ainda nesta aula.
- Para visualizar se a operação de inserção teve sucesso, digite e execute o seguinte comando:
 - `SELECT * FROM Escola;`

Atenção

Sobre o uso do SQLite Studio

- No SQLite Studio, o comando anterior poder ser executado de diferentes modos:
- (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 selecionando-o com o mouse para em seguida executá-lo. O mesmo procedimento poderá ser feito para todos os exemplos posteriores desta apostila.

Abrindo nova janela de edição



Selecionando o
comando

INSERIR REGISTROS

--Inserindo mais três escolas com todas as colunas declarando a lista.

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia, tp_localizacao,  
qt_mat_bas_fem, qt_mat_bas_masc, qt_mat_bas, qt_doc_bas, cod_regiao,  
cod_uf, cod_municipio) VALUES (53017943, 'ESC CR PARTIMPIM BABY KIDS',  
4, 1, 51, 26, 77, 3, 5, 53, 5300108);
```

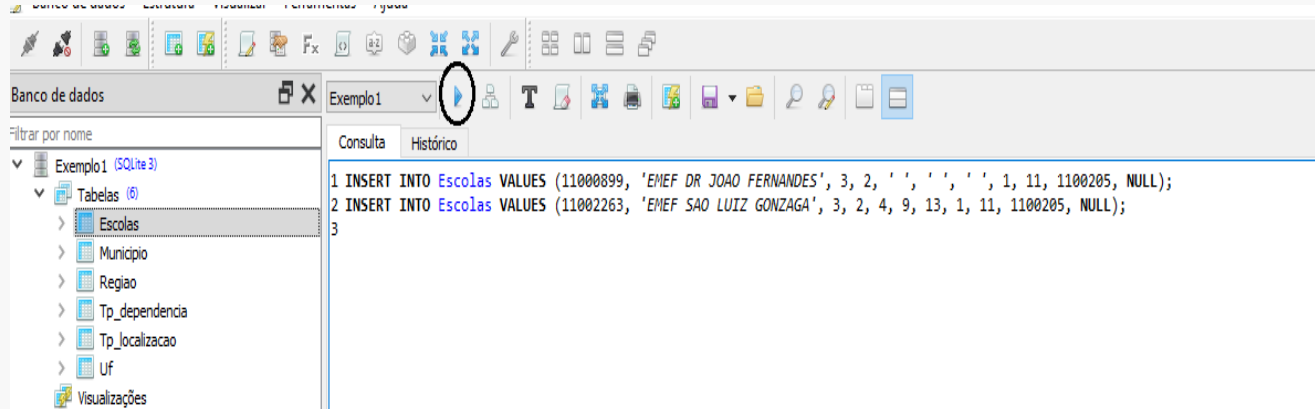
```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia, tp_localizacao,  
qt_mat_bas_fem, qt_mat_bas_masc, qt_mat_bas, qt_doc_bas, cod_regiao,  
cod_uf, cod_municipio) VALUES (53018605, 'CEF 113 DO RECANTO DAS  
EMAS', 2, 1, 685, 732, 1417, 63, 5, 53, 5300108);
```

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia, tp_localizacao,  
qt_mat_bas_fem, qt_mat_bas_masc, qt_mat_bas, qt_doc_bas, cod_regiao,  
cod_uf, cod_municipio) VALUES (53018990, 'COL ABCZINHO', 4, 1, 45, 41, 86,  
3, 5, 53, 5300108);
```

INSERIR REGISTROS

- Após copiar e colar este conjunto de comandos na janela de edição SQL. Selecione-os com o mouse e clique no ícone **execute consulta** conforme mostra a Figura 6. Você também pode selecionar estes comandos e clicar em **F9**.

Figura 6 – Executar query de inserção da tabela Escola



CONFERIR INSERÇÃO

- O comando apresentado serve para recuperar as linhas (conteúdo atual) de Escolas. O resultado do SELECT é mostrado a seguir na figura 7.

Figura 7 – Resultado do comando SELECT * FROM *Escola*;

Exibição em grade		Visualização do formulário									
		Total de linhas carregadas: 3									
	co_entidad	no_entidade	cod_regiao	cod_uf	cod_munici	tp_depend	tp_localiza	qt_mat_ba	qt_mat_ba	qt_mat_ba	qt_doc_ba
1	11002158	EMEF SAO FRANCISCO DE ASSIS	1	11	1100205	3	1	70	58	128	7
2	11003391	EMEF MANOEL MACIEL NUNES	1	11	1100205	3	2	21	22	43	2
3	35578678	CENTRO EDUCACIONAL INFANTIL ALMERINA PEREIRA DOS SANTOS	3	35	3548401	3	1	54	50	103	13

INSERIR LINHAS COM INFORMAÇÕES PARCIAIS

--Inserir escolas somente com os atributos (co_entidade, no_entidade, tp_dependencia)

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia) VALUES
(11009888, 'EEEFM LAURINDO RABELO', 2);
```

```
INSERT INTO Escola (co_entidade, no_entidade, tp_dependencia) VALUES
(11017112, 'EMEIEF MARACATIARA', 3);
```

Execute o comando a seguir para verificar:

```
SELECT * FROM Escola;
```

INSERIR LINHAS COM INFORMAÇÕES PARCIAIS

- Inserir registro com informações parciais na seguinte ordem:

cod_uf, qt_mat_bas, co_entidade, escola

--inserir escolas contendo apenas co_entidade, no_entidade, cod_uf, qt_mat_bas

```
INSERT INTO Escola (cod_uf,qt_mat_bas , co_entidade, no_entidade)
```

```
VALUES (11,155,11050020, 'EMEF CECILIA MEIRELES');
```

```
INSERT INTO Escola (co_entidade, no_entidade, cod_uf,qt_mat_bas )
```

```
VALUES (11, 28 ,11050527, 'ESCOLA CASTELINHO DO SABER');
```

Atenção: Veja que a ordem dos campos na lista do INSERT está diferente da ordem padrão e somando isso é fundamental que os valores contidos no VALUES respeite a ordem da lista. **Uma boa prática é sempre declarar a lista para evitar erros.**

INSERIR REGISTRO COM APÓSTROFO

- Inserir registro com apóstrofo (aspas simples) no valor do atributo

--É necessário colocar dois apóstrofes para que fique gravado somente um.

```
INSERT INTO Escola (co_entidade, no_entidade, cod_uf, qt_mat_bas )  
VALUES (11050021, 'EMEF CECILIA D"ALEMBERT', 11,155);
```

TENTAR INSERIR REGISTRO REPETIDO

- Tentar inserir um registro com chave primária repetida

-- Registro com chave única repetida

```
INSERT INTO Escola (co_entidade, no_entidade, cod_uf, qt_mat_bas )  
VALUES (11050021, 'EMEF CECILIA D"ALEMBERT', 11, 155);
```

A seguinte mensagem de erro é retornada:

[21:28:44] Erro ao executar consulta SQL no banco de dados 'Exemplo1':
UNIQUE constraint failed: Escolas.co_entidade

- A definição de chave primária não permite que seja incluída um registro com valor do atributo chave identificador da escola já existente na relação Escolas.

INSERIR NAS DEMAIS TABELAS

--Inserir registro na tabela Regiao

```
INSERT INTO Regiao (cod_regiao, nome_regiao) VALUES (1, 'NORTE');  
INSERT INTO Regiao (cod_regiao, nome_regiao) VALUES (2, 'NORDESTE');  
INSERT INTO Regiao (cod_regiao, nome_regiao) VALUES (4, 'SUL ');  
INSERT INTO Regiao (cod_regiao, nome_regiao) VALUES (3, 'SUDESTE');  
INSERT INTO Regiao (cod_regiao, nome_regiao) VALUES (5, 'CENTRO OESTE');
```

--Inserir registro na relação Uf

```
Insert into UF (COD_UF,NOME_UF,SIGLA_UF) values ('11','RONDÔNIA','RO');  
Insert into UF (COD_UF,NOME_UF,SIGLA_UF) values ('35','SÃO PAULO','SP');  
Insert into UF (COD_UF,NOME_UF,SIGLA_UF) values ('13','AMAZONAS','AM');
```

INSERIR NAS DEMAIS TABELAS

--Inserir registros na tabela Municipio

```
INSERT INTO Municipio (cod_municipio, nome_municipio) VALUES (1301506, 'Envira');  
INSERT INTO Municipio (cod_municipio, nome_municipio) VALUES (1304302, 'Urucará');  
INSERT INTO Municipio (cod_municipio, nome_municipio) VALUES (1304104, 'Tapauá');  
INSERT INTO Municipio (cod_municipio, nome_municipio) VALUES (1303106, 'Nova  
Olinda do Norte');  
INSERT INTO Municipio (cod_municipio, nome_municipio) VALUES (5300108, 'Brasília');
```

--Inserir registros do tipo de dependência administrativa

```
INSERT INTO Tp_dependencia (tp_dependencia, dependencia) VALUES (1, 'Federal');  
INSERT INTO Tp_dependencia (tp_dependencia, dependencia) VALUES (2, 'Estadual');  
INSERT INTO Tp_dependencia (tp_dependencia, dependencia) VALUES (3, 'Municipal');  
INSERT INTO Tp_dependencia (tp_dependencia, dependencia) VALUES (4, 'Privada');
```

-- Inserir registros do tipo de localização

```
INSERT INTO Tp_localizacao (tp_localizacao, localizacao) VALUES (1, 'Urbana');  
INSERT INTO Tp_localizacao (tp_localizacao, localizacao) VALUES (2, 'Rural');
```

INSERIR MÚLTIPLAS LINHAS

- O comando INSERT originalmente na linguagem SQL **insere apenas uma linha por vez**. Cada INSERT insere uma linha na tabela. Só que atualmente muitos dos SGBDs oferecem uma sintaxe que permite a inclusão de múltiplas linhas com um único INSERT. O SQLite oferece suporte a esta sintaxe conforme exemplo a seguir:

```
INSERT INTO Municipio (cod_municipio, nome_municipio)  
VALUES (1506500, 'Santa Izabel do Pará'),  
(1507003, 'Santo Antônio do Tauá'),  
(1505809, 'Portel');
```


INSERIR MÚLTIPLAS LINHAS

- Exemplo de INSERT múltiplo com erro de PK. Todas as inserções são canceladas porque houve erro.

```
INSERT INTO Municipio (cod_municipio, nome_municipio)  
VALUES (5107248, 'Santa Carmem'),  
(5107941, 'Tabaporã'),  
(5107248, 'Itanhangá');
```



[11:41:16] Erro ao executar consulta SQL no banco de dados 'Exemplo1': UNIQUE constraint failed: Municipio.cod_municipio

INSERIR MÚLTIPLAS LINHAS

- Neste exemplo, três linhas são inseridas de uma só vez na tabela *Municipio* com o uso de apenas um comando `INSERT`. Note que o comando `INSERT` anterior inclui três registros:
 - Cada registro inclui valores que respeita a ordem de atributos declarada,
 - Cada registro está separado do outro por vírgula.
 - O ponto e vírgula é usado somente para dizer que o comando terminou caso queira executar outros comandos após este.

```
INSERT INTO Municipio (cod_municipio, nome_municipio)  
VALUES (1506500, 'Santa Izabel do Pará'),  
(1507003, 'Santo Antônio do Tauá'),  
(1505809, 'Portel');
```

EXCLUIR REGISTROS

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*;

No comando delete acima:

tabela: nome da tabela alvo.

condicao: um ou mais testes realizado sobre uma ou mais colunas (campos, atributos) para definir as linhas que devem excluídas.

EXCLUIR REGISTROS

EXEMPLOS:

*/*Excluir linha da tabela Escola com menos de 30 matrículas do ensino básico e que o código do município esteja vazio.*/**

DELETE FROM Escola

WHERE qt_mat_bas < 30 AND cod_municipio IS NULL;

- Observe que neste exemplo a lógica usada para excluir linhas pode selecionar uma ou mais linhas. O teste feito para `cod_municipio` faz uso do conectivo AND se a primeira e a segunda condição forem iguais então é verdadeira a proposição. O Segundo teste é feito com o uso de `IS NULL` para verificar se campo `cod_municipio` é igual nulo (com ausência de valor).
- Atenção: Está construção `atributo = NULL` vai retornar falso, pois a forma correta é `atributo IS NULL`.

EXCLUSÃO DE REGISTROS

EXEMPLOS:

/*Excluir registro da relação Escola que o valor de cod_uf não esteja nulo e o atributo escola seja igual a 'EMEF CECILIA MEIRELES'.*/

DELETE FROM *Escola*

WHERE escola = 'EMEF CECILIA MEIRELES' AND cod_uf IS NOT NULL;

- **observe que que neste exemplo** é usado um teste para verificar se o atributo está sem um valor gravado. A construção **atributo IS NOT NULL** é verdadeira se não há valor gravado.

Atenção: O valor nulo (NULL) é diferente de uma cadeia de caracteres de comprimento zero (" – abrir e fechar aspas simples sem espaço).

EXCLUSÃO DE REGISTROS

- A lógica definida na cláusula WHERE pode ser montada de várias formas que for preciso. Isto inclui o uso dos operadores booleanos como AND, OR e NOT para possibilitar testes em diversas colunas. O comando DELETE irá obedecer exatamente aquilo que for colocado nesta proposição.
- **DELETE FROM *tabela* WHERE *condicao*;**

Tabela 1 – Operadores lógicos

Conectivo	Operador	Operação lógica	Valor
E	AND	Conjunção	Retorna o valor verdadeiro somente quando todas as proposições são verdadeiras.
OU	OR	Disjunção	Retorna verdadeiro quando pelo menos uma proposição for verdadeira.
NÃO	NOT	Negação	Retorna o valor falso caso a proposição seja verdadeira e vice-versa.

EXCLUSÃO DE REGISTROS

- Exemplos:
 - Verdadeiro – v; falso – f com uso do parênteses. Em SGBD relacionais é como elaborar construções de condição com vários testes. O uso dos parênteses indica a ordem que os testes devem ocorrer.

Tabela 2 – Operadores lógicos

Proposições	Resultado
v AND v	Verdadeiro
v AND v AND f	Falso
v OR v OR f	Verdadeiro
NOT f AND v AND v	Verdadeiro
(f OR f) AND (f OR f)	Falso
(f OR v) AND (v OR f)	Verdadeiro
(NOT (f OR v)) AND (v OR f)	Falso

ATUALIZAR REGISTRO

- Atualizando registros com o comando UPDATE
- O comando UPDATE é utilizado para modificar os valores de uma ou mais colunas de uma tabela. A operação pode atualizar uma ou diversas linhas. O comando atende a sintaxe a seguir:

UPDATE *tabela*

SET $coluna_1=valor_1, \dots, coluna_n=valor_n$

WHERE *condição*;

ATUALIZAR REGISTRO

Na definição do UPDATE

tabela: nome da tabela que terá registros atualizados.

coluna_i=valor_i: nome de uma coluna que será modificada e o novo valor que será a ela atribuída.

condição: proposição com uma ou mais colunas para definir as linhas que serão atualizadas. Funciona da mesma forma que a condição da instrução DELETE.

UPDATE *tabela*

SET *coluna₁=valor₁, ..., coluna_n=valor_n*

WHERE *condição*;

ATUALIZAR REGISTRO

EXEMPLOS:

Modificar cod_uf da escola para 33 com a identificação única 11050021.

--Atualizar o código da unidade federativa para 33 quando identificação única da escola for igual a 11050021

```
UPDATE Escola SET cod_uf=33 WHERE co_entidade=11050021;
```

- O comando faz uso da tabela *Escola* para aplicar uma atualização no atributo cod_uf quando a proposição com a chave única for satisfeita. Assim, somente uma única linha será afetada pelo comando UPDATE.

ATUALIZAR REGISTRO

Tentar atualizar tabela utilizando uma condição falsa.

--Tentar atualizar o código da unidade federativa para uma condição falsa

```
UPDATE Escola SET cod_uf=33 WHERE co_entidade=9999999;
```

O SQLite Studio emite a mensagem a seguir para informar que há zero linhas afetadas.

[20:28:22] Consulta finalizada em 0.000 segundo(s). Linhas afetadas: 0.

Atenção: O uso do comando UPDATE sem WHERE faz atualizações em todas as linhas para a coluna especificada. É fundamental ter muito cuidado para não atualizar todas as linhas da tabela por engano. **UPDATE SET COD_UF=33!**

ATUALIZAR REGISTRO

- **Modificar mais de uma coluna**

/*Atualizar as colunas tp_dependencia, tp_localizacao e escola quando a PK (PRIMARY KEY) for igual 11050021*/

```
UPDATE Escola SET
```

```
tp_dependencia=2,
```

```
tp_localizacao=2,
```

```
escola='EMEF CECILIA DALEMBERT'
```

```
WHERE co_entidade=11050021;
```

- No caso do exemplo anterior, tp_dependencia muda de nulo para 2, tp_localizacao de nulo para 2 e mudou o nome da escola 'EMEF CECILIA D'ALEMBERT' para 'EMEF CECILIA DALEMBERT' (retirou o plique).

EXERCÍCIOS PROPOSTOS 1

1) Inserir a escola ESC RAIMUNDO QUIRINO NOBRE situada na região sudeste (3) unidade federativa São Paulo, município de São Paulo, tipo de localização urbana, tipo de dependência federal. A escola possui 50 matrículas na educação básicas masculina (qt_mat_bas_masc), 60 femininas (qt_mat_bas_fem). O total de docentes é 10 (qt_doc_bas). A identificação do INEP sugeria para esta escola é 400.

```
INSERT INTO escola (no_entidade, cod_regiao, cod_uf, tp_localizacao,  
tp_dependencia, qt_mat_bas_masc, qt_mat_bas_fem, qt_doc_bas,  
co_entidade)
```

```
Values ('ESC RAIMUNDO QUIRINO NOBRE', 3, 35, 1,1, 50,60,10,400 );
```

EXERCÍCIOS PROPOSTOS 1

2) Quando não tiver município informado na tabela Escola. Atualize as colunas tipo de dependência (tp_dependencia) para Municipal, tipo de localização para Rural, qt_mat_bas_masc para 10, qt_mat_bas_fem para 11 e qt_mat_bas para a soma de qt_mat_bas_mas + qt_mat_bas_fem.

•(Atenção, quando for o caso, use os respectivos códigos numéricos de cada valor e não o nome para atualizar as colunas.)

```
UPDATE      escola      set      tp_localizacao=2,      qt_mat_bas_masc=10,  
qt_mat_bas_fem=11, qt_mat_bas= (qt_mat_bas_masc + qt_mat_bas_fem)
```

```
WHERE cod_municipio IS NULL;
```

EXERCÍCIOS PROPOSTOS 1

- 3) Corrija os registros da tabela Escola que tenha total de matrículas na educação básica (qt_mat_bas) diferente de qt_mat_bas_masc + qt_mat_bas_fem.

```
UPDATE Escola SET qt_mat_bas = (1.0*qt_mat_bas_masc +  
1.0*qt_mat_bas_fem)  
WHERE (qt_mat_bas <> (qt_mat_bas_masc + qt_mat_bas_fem))  
OR (qt_mat_bas IS NULL AND (qt_mat_bas_masc + qt_mat_bas_fem)>0);
```

EXERCÍCIOS PROPOSTOS 1

4) Exclua o registro de escola que tenha identificação única 400.

```
DELETE FROM ESCOLA WHERE cod_entidade=400;
```


MODIFICAÇÃO DE ESQUEMAS: DROP TABLE e ALTER TABLE

Recapitulando, até aqui vimos comandos SQL para modificar o conteúdo de tabelas e declarar esquemas usando SQL. Imagine caso seja necessário alterar o esquema de uma tabela que já tenha sido definida, que esteja em uso há algum tempo e que já contenha várias linhas. Como resolver?

- Na prática vários tipos diferentes de modificação de esquemas podem ser necessários:
- Remover uma tabela (ex.: ela não precisa ser mais usada).
- Renomear uma tabela ou coluna (ex.: o nome não está adequado)
- Adicionar uma nova coluna (a mais comum das alterações de esquema).
- Remover uma coluna existente.

APAGAR TABELA

- A modificação de esquemas é algo pouco frequente, mas eventualmente torna-se necessário modificar a definição de uma tabela. Os comandos SQL a seguir apresentam como realizar este tipo de operação.

PRÁTICA

- Os comandos DML utilizados anteriormente atuam somente no conteúdo das tabelas. Até mesmo o comando `DELETE FROM Escolas` elimina somente as tuplas sem que a tabela deixe de existir. Para remover a tabela *Tp_dependencia* do BD, é necessário utilizar o comando `DROP TABLE`:

```
DROP TABLE Tp_dependencia;
```

Para confirmar se a tabela foi realmente excluída, use o comando a seguir.

```
SELECT * FROM Tp_dependencia;
```

- Como a tabela não existe mais, o SQLite Studio apresentar a seguinte informação na janela de status:
 - [20:39:23] Erro ao executar consulta SQL no banco de dados 'Exemplo1': no such table: *Tp_dependencia*

APAGAR TABELA

DROP TABLE – Sintaxe na SQL

DROP TABLE *tabela*;

Na definição acima:

tabela: nome da tabela que será apagada.

Atenção: O comando DROP TABLE apaga a tabela do BD **mesmo que ela contenha linhas**. Após execução com sucesso desta instrução sobre a tabela especificada, esta tabela deixara de fazer parte do esquema do banco de dados assim como todos os seus registros.

RENOMEAR TABELA

Renomear uma Tabela

- A necessidade de mudar o nome de uma tabela pode não ser algo tão comum quando ela já estiver em uso. Este tipo de mudança é mais factível na fase de projeto, quando o projetista está construindo os elementos do esquema e avaliando as necessidades que precisa atender. Contudo, há situações de manutenção que isto se torna necessário. Essa modificação é realizada com o uso do comando ALTER TABLE.

RENOMEAR TABELA

Atenção, o comando 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 Escola RENAME TO Escolas;
```

ALTER TABLE para renomear tabela – Sintaxe na SQL

```
ALTER TABLE nome_anterior TO nome_novo;
```

Na definição acima:

nome_anterior: nome da tabela que será renomeada.

nome_novo: novo nome que será dado para a tabela

INSERIR UMA COLUNA NOVA

- Suponha que houve a inclusão de uma variável no Censo Escolar e que a partir de agora haverá uma indicação se a escola está ativa ou inativa. 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 *Escola* ADD desativada INTEGER NULL;
- O exemplo insere uma nova coluna como a **última coluna** da tabela e com o valor NULL (ausente) em todos os registros.

INSERIR UMA COLUNA NOVA

- Suponha que seja necessário incluir **uma coluna nova com a informação padrão** sobre quadra de esporte.
- Para isso, basta utilizar o comando de alteração de tabela incluindo o termo DEFAULT após o tipo da coluna que será adicionada.

ALTER TABLE *Escola* ADD in_quadra_esportes INTEGER DEFAULT 9.

- Neste exemplo adicionamos a coluna “in_quadra_esportes” na tabela *Escola* com um valor padrão 9. Após a execução deste comando todas as linhas da tabela incluem o atributo in_quadra_esportes com o valor 9.

INSERIR UMA COLUNA NOVA

IMPORTANTE: sempre que um comando INSERT for executado sobre a tabela *Escola*, caso o valor do campo “in_quadra_esportes” não seja especificado, será assumido que o mesmo é o valor DEFAULT (ou seja, 9).

```
INSERT INTO Escola (co_entidade, escola) VALUES (35661831, 'CEMEIEF PROFA  
DINA FERREIRA CURY DIAS BAPTISTA');
```


INSERIR UMA COLUNA NOVA

Para inserir uma escola que tenha quadra de esportes, você terá que explicitar o valor desejado para o campo “in_quadra_esportes” no comando INSERT:

```
INSERT INTO Escola (co_entidade, escola, in_quadra_esportes)
```

```
VALUES (35661855, 'CEMEIEF PROFA MARIA APARECIDA MARGARIDO COSTA',1);
```

INSERIR UMA COLUNA NOVA

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, INTEGER, REAL).
 - Opcionalmente, pode ser atribuído um valor default para a coluna, com o uso da palavra-chave DEFAULT seguida do valor.

REMOVER COLUNA

ALTER TABLE para remover coluna – Sintaxe na SQL

ALTER TABLE *tabela* DROP *nome_coluna*;

Na definição acima:

- *tabela*: nome da tabela que terá a coluna apagada.
- *nome_coluna*: nome da coluna que será apagada.
- A exclusão de coluna é um recurso mais recente incluído na estrutura do SQL do SQLite. Em versões anteriores a solução usada era guardar os dados da tabela, reconstruir a tabela sem a coluna desejada e reinserir as informações na tabela nova descartando a coluna antiga.

REMOVER COLUNA

- Suponha que a coluna incluída “in_quadra_esportes” por questões de prazo e orçamento não deve ser incluída no questionário da pesquisa. Assim, devemos excluir a coluna da tabela Escola.

```
ALTER TABLE Escola DROP in_quadra_esportes;
```

EXERCÍCIOS PROPOSTOS 2

- 1) Suponha que seja necessário mudar o tipo da coluna nome_municipio definido na tabela Municipio de VARCHAR(60) para VARCHAR(50) mantendo os valores das colunas existentes. Como você pode resolver isso? (Escreva o código SQL da solução.)

```
ALTER TABLE Escola ADD nome_municipio_a VARCHAR(50);
```

```
UPDATE Escola SET nome_municipio_a = nome_municipio;
```

```
ALTER TABLE Escola DROP nome_municipio;
```

```
ALTER TABLE Escola ADD nome_municipio VARCHAR(50);
```

```
UPDATE Escola SET nome_municipio = nome_municipio_;
```

```
ALTER TABLE Escola DROP nome_municipio_a;
```

EXERCÍCIOS PROPOSTOS 2

- 2) Durante a fase de coleta algumas dificuldades ocorreram para o preenchimento do questionário do Censo Escolar. O Responsável pela coleta decidiu incluir os textos com observações das dificuldades encontradas. Um pedido foi enviado para o setor de informática para que um novo campo fosse incluído na solução. Você como responsável pelo projeto do banco de dados recebeu a tarefa de incluir uma coluna do tipo TEXT denominada *observacao* na tabela *Escola*. Como você escreveria o SQL desta modificação? (Escreva o código SQL da solução.)

```
ALTER TABLE Observacao ADD TEXT;
```

EXERCÍCIOS PROPOSTOS 3

3) Siga os seguintes passos a seguir e *descreva o que você compreendeu*.

--Execute o comando Begin

Begin transaction

--Faça a inserção de duas tuplas na tabela escola

```
INSERT INTO Escola (co_entidade, no_entidade) VALUES (4500001, 'ESCOLA TESTE 1'
);
```

```
INSERT INTO Escola (co_entidade, no_entidade) VALUES (4500002, 'ESCOLA TESTE
2');
```

-- Execute o comando SELECT

```
SELECT * FROM Escola;
```

Identifique as duas escolas incluídas

--Execute o comando Rollback;

```
Rollback;
```

-- Execute o comando SELECT

```
SELECT * FROM Escola;
```

Obrigado