Logotipo, nome da empresa

Descrição gerada automaticamente

Apostila – 08

**[Programação Para Dispositivos Móveis – PDM]**

**Cursos:**

**Tecnologia em Análise e Desenvolvimento de Sistemas**

**Ciência da Computação**

# **Visão Geral do SQLite**

Vamos entender o que é o **SQLite**, como ele difere do **SQL**, por que é necessário e como ele lida com o banco de dados de aplicativos. **SQLite** é uma biblioteca de software que implementa um mecanismo de banco de dados **SQL** autônomo, sem servidor, de configuração zero e transacional. **SQLite** é um dos motores de banco de dados de crescimento mais rápido em termos de popularidade, não em termos de seu tamanho. O código-fonte do **SQLite** está em domínio público.

## O Que é o SQLite?

Como vimos anteriormente, **SQLite** é uma biblioteca de software que implementa um mecanismo de banco de dados **SQL** autônomo, sem servidor, de configuração zero e transacional. Como podemos perceber, é o único banco de dados, que não precisa ser configurado, o que significa não vamos precisar configurá-lo em nosso sistema. O motor do **SQLite** não é um processo autônomo como outros bancos de dados, podemos vinculá-lo estaticamente ou dinamicamente conforme a exigência da nossa aplicação.

## Por que usar SQLite?

* O **SQLite** não requer um processo ou sistema de servidor separado para operar (sem servidor – serverless).
* O **SQLite** vem com zero-configuração, o que significa que nenhuma configuração ou administração é necessária.
* Um banco de dados **SQLite** completo é armazenado em um único arquivo de disco multi-plataforma.
* O **SQLite** é muito pequeno e peso leve, menos de 400KB totalmente configurado ou menos de 250KB com recursos opcionais omitidos.
* O **SQLite** é auto-suficiente, o que significa que não há dependências externas.
* As transações **SQLite** são totalmente compatíveis com ACID, permitindo o acesso seguro de vários processos ou threads.
* O **SQLite** suporta a maioria dos recursos de linguagem de consulta encontrados no padrão SQL92 (SQL2).
* O **SQLite** é escrito em ANSI-C e fornece uma API simples e fácil de usar.
* O **SQLite** está disponível em UNIX (Linux, Mac OS-X, Android, iOS) e Windows (Win32, Win64, WinCE, WinRT).

## História

2000 – Drº Richard Hipp tinha projetado o **SQLite** com a finalidade de nenhuma configuração necessária para operar um programa.

2000 - Em agosto, o **SQLite** 1.0 foi lançado com o GNU Database Manager.

2011 - Drº Richard Hipp anunciou para adicionar UNQl interface para SQLite DB e desenvolver UNQLite (Document orientado banco de dados).

## Limitações do SQLite

Existem poucos recursos não suportados do **SQL92** no **SQLite** que são mostrados na tabela abaixo:

|  |  |
| --- | --- |
| **Característica** | **Descrição** |
| **RIGHT OUTER JOIN** | Somente LEFT OUTER JOIN é implementado. |
| **FULL OUTER JOIN** | Somente LEFT OUTER JOIN é implementado. |
| **ALTER TABLE** | As variantes RENAME TABLE e ADD COLUMN do comando ALTER TABLE são suportadas. DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT não é suportada. |
| **Suporte a Trigger** | As linhas de triggers são suportadas, mas declarações separadas de triggers não são suportadas. |
| **VIEWs** | VIEWs em **SQLite** são somente leitura. Não podemos executar uma instrução DELETE, INSERT ou UPDATE em uma VIEW. |
| **GRANT e REVOKE** | As permissões de acesso apenas podem ser aplicadas as permissões de acesso de arquivos normais do sistema operacional em uso. |

## Comandos SQLite

Os comandos padrão **SQLite** para interagir com bancos de dados relacionais são semelhantes aos comandos **SQL**, que são **CREATE**, **SELECT**, **INSERT**, **UPDATE**, **DELETE** e **DROP**. Estes comandos podem ser classificados em grupos com base na sua natureza operacional, conforme abaixo:

**DDL - Data Definition Language – Linguagem de Definição de Dados** é o subconjunto da **SQL** que vamos utilizar para gerenciar a estrutura do banco de dados. Com a **DDL** podemos criar, alterar e remover objetos (tabelas, visões, funções, etc.) no banco de dados. Os comandos deste subconjunto são:

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| **CREATE** | Cria uma nova tabela, uma view de uma tabela ou outro objeto no banco de dados. |
| **ALTER** | Modifica um objeto de banco de dados existente, como uma tabela. |
| **DROP** | Exclui uma tabela inteira, uma view de uma tabela ou outro objeto no banco de dados. |

**DML - Data Manipulation Language – Linguagem de Manipulação de Dados** é o subconjunto que nós vamos utilizar com mais frequência da linguagem **SQL**, pois é através da **DML** que vamos operar sobre os dados dos bancos de dados com instruções de inserção, atualização, exclusão e consulta de informações. Os comandos **SQL** desse subconjunto são:

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| **INSERT** | Cria um registro. |
| **UPDATE** | Altera os registros. |
| **DELETE** | Deleta os registros. |

Só para lembrar que **Registro**, **Linha** e **Tupla** são palavras sinônimas para referenciar a uma linha da tabela.

**DQL - Data Query Language – Linguagem de Consulta de Dados**

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| **SELECT** | Recupera certos registros de uma ou mais tabelas. |

**DCL – DATA CONTROL LANGUAGE – Linguagem de Controle de Dados** é o subconjunto da **SQL** que vamos utilizar para controlar o acesso aos dados, basicamente com dois comandos que permite ou bloqueia o acesso de usuários a dados. Vejamos estes comandos:

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| **GRANT** | Autoriza um usuário a executar alguma operação. |
| **REVOKE** | Restringe ou remove a permissão de um usuário executar alguma operação. |

**DTL – DATA TRANSACTION LANGUAGE – Linguagem de controle de transações** é o subconjunto da **SQL** que vai nos fornecer mecanismos para controlar transações no banco de dados. São 3 comandos:

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| **BEGIN TRANSACTION** | Iniciar uma transação. |
| **COMMIT** | Efetivar as alterações no banco de dados. |
| **ROLLBACK** | Cancelar as alterações. |

Veremos mais adiante o uso destes comandos.

## Instalação do SQLite

O **SQLite** é famoso por sua principal característica de zero-configuração, o que significa que não é necessário configuração complexa ou de administração. Veremos o processo de configuração do **SQLite** no **Windows**.

## Tipo de dados SQLite

No **SQLite** o tipo de dados é um atributo que especifica o tipo de dados de qualquer objeto. Cada coluna, variável e expressão tem tipo de dados relacionados no **SQLite**. Vamos usar esses tipos de dados ao criarmos as tabelas do nosso banco de dados. O **SQLite** usa um sistema de tipo dinâmico mais geral. No **SQLite**, o tipo de dados de um valor está associado ao próprio valor, e não ao seu contêiner.

## Classes de Armazenamento SQLite

Cada valor armazenado em um banco de dados **SQLite** tem uma das seguintes classes de armazenamento, cnforme tabela abaixo:

|  |  |
| --- | --- |
| **Classe de Armazenamento** | **Descrição** |
| **NULL** | O valor é um valor NULL. |
| **INTEGER** | O valor é um inteiro assinado, armazenado em 1, 2, 3, 4, 6 ou 8 bytes, dependendo da grandeza do valor. |
| **REAL** | O valor é um valor de ponto flutuante, armazenado como um número de ponto flutuante IEEE de 8 bytes. |
| **TEXT** | O valor é uma seqüência de caracteres de texto, armazenada usando a codificação de banco de dados (UTF-8, UTF-16BE ou UTF-16LE). |
| **BLOB** | O valor é um blob de dados, armazenado exatamente como foi inserido. |

A Classe de armazenamento do **SQLite** é um pouco mais geral do que um tipo de dados. A classe de armazenamento **INTEGER**, por exemplo, inclui 6 diferentes tipos de dados inteiros de comprimentos diferentes.

## Tipo de Afinidade no SQLite

O **SQLite** suporta o conceito de afinidade de tipo em colunas. Qualquer coluna ainda pode armazenar qualquer tipo de dados, mas a classe de armazenamento preferida para uma coluna é chamada de sua afinidade. Cada coluna das tabelas em um banco de dados **SQLite3** é atribuída um tipo de afinidade, conforme tabela abaixo:

|  |  |
| --- | --- |
| **Afinidade** | **Descrição** |
| **TEXT** | Esta coluna armazena todos os dados usando classes de armazenamento **NULL**, **TEXT** ou **BLOB**. |
| **NUMERIC** | Esta coluna pode conter valores usando todas as cinco classes de armazenamento. |
| **INTEGER** | Comporta-se como uma coluna com afinidade NUMÉRICA com uma exceção em uma expressão CAST. |
| **REAL** | Comporta-se como uma coluna com afinidade NUMÉRICA exceto que força valores inteiros em representação em ponto flutuante. |
| **NONE** | Uma coluna com afinidade **NONE** não prefere uma classe de armazenamento sobre outra e nenhuma tentativa é feita para coagir os dados de uma classe de armazenamento para outra. |

## Nomes de Afinidade e Tipo no SQLite

Na tabela a seguir, são listados vários nomes de tipos de dados que podem ser usados durante a criação de tabelas no **SQLite3** e a correspondente afinidade aplicada, conforme tabela abaixo:

|  |  |
| --- | --- |
| **Tipo de Dados** | **Afinidade** |
| **INT**  **INTEGER**  **TINYINT**  **SMALLINT**  **MEDIUMINT**  **BIGINT**  **UNSIGNED BIG INT**  **INT2**  **INT8** | **INTEGER** |
| **CHARACTER(20)**  **VARCHAR(255)**  **VARYING CHARACTER(255)**  **NCHAR(55)**  **NATIVE CHARACTER(70)**  **NVARCHAR(100)**  **TEXT**  **CLOB** | **TEXT** |
| **BLOB**  **Nenhum tipo de dado especificado.** | **NONE** |
| **REAL**  **DOUBLE**  **DOUBLE PRECISION**  **FLOAT** | **REAL** |
| **NUMERIC**  **DECIMAL(10,5)**  **BOOLEAN**  **DATE**  **DATETIME** | **NUMERIC** |

## Tipo de Dados booleano

O **SQLite** não tem uma classe de armazenamento **Boolean** separado. Em vez disso, os valores booleanos são armazenados como inteiros 0 (falso) e 1 (verdadeiro).

## Tipo de Dados Data e Hora

O **SQLite** não tem uma classe de armazenamento separada para armazenar datas e / ou horas, mas o **SQLite** é capaz de armazenar datas e horas como valores **TEXT**, **REAL** ou **INTEGER**, conforme tabela abaixo:

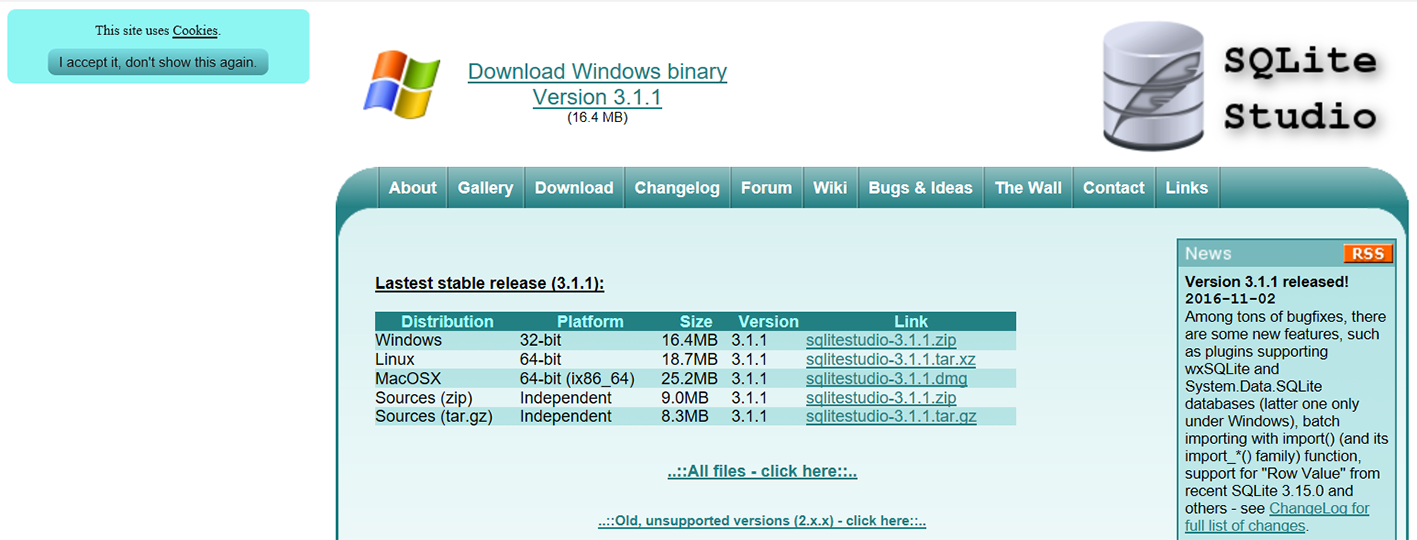
|  |  |
| --- | --- |
| **Classe de Armazenamento** | **Formato de Data** |
| **TEXT** | Uma data em um formato como **"YYYY-MM-DD HH:MM:SS.SSS"**. |
| **REAL** | O número de dias desde o meio-dia em **Greenwich em 24 de novembro de 4714 B.C.** |
| **INTEGER** | O número de segundos desde **1970-01-01 00:00:00 UTC**. |

Podemos escolher armazenar datas e horas em qualquer um desses formatos e converter livremente entre formatos usando as funções de data e hora incorporadas.

# **Capitulo 05: Trabalhando com Banco de Dados SQLite**

## Usando Versão Gráfica do SQLite

Veja que trabalhar no **prompt** de comando, não é nada agradável e nem produtivo. Por este motivo vamos baixar uma versão gráfica do **SQLite**. Para isto, abra seu navegador web e digite o seguinte endereço: **http://sqlitestudio.pl/?act=download**, veja que vai aparecer o site para download, conforme figura abaixo:



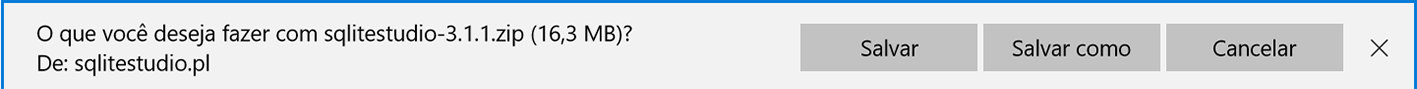
**Figura 18**

Clique no link **Download Windows binary Version 3.1.1**, conforme figura abaixo:



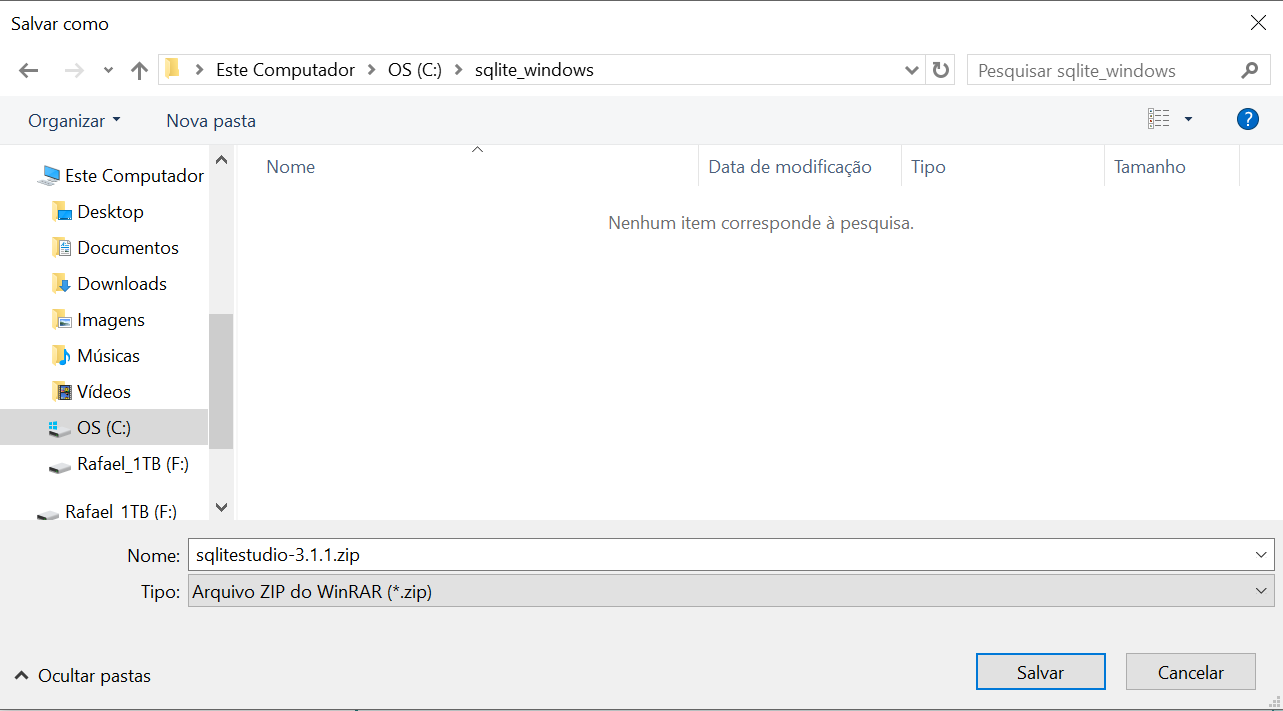
**Figura 19**

Vai aparecer a opção para download, conforme figura abaixo:



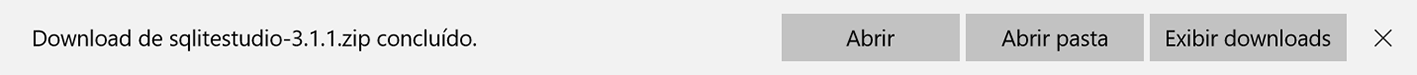
**Figura 20**

Esta mensagem é referente ao Navegador **Microsoft Edge**, dependendo do navegador esta mensagem vai ser diferente, clique no botão **Salva como**, vai abrir a janela **Salvar como**, conforme figura abaixo:



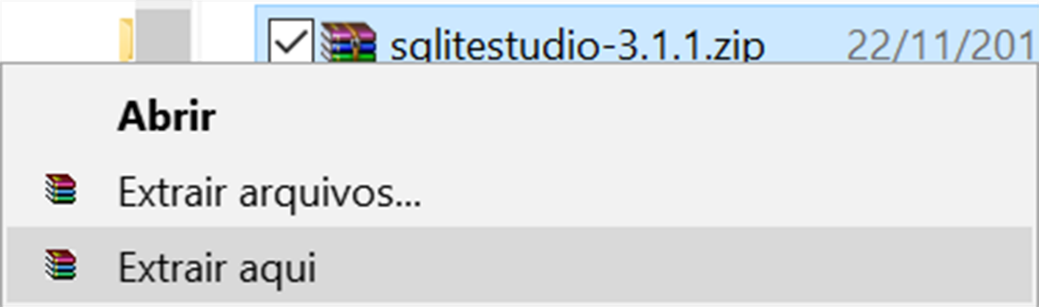
**Figura 21**

Selecione a pasta que criou, no meu caso, criei uma pasta chamada **sqlite\_windows**, no meu **C:\**, e clique no botão **Salvar** e aguarde o término do **download**, vai aparecer uma mensagem, conforme figura abaixo:



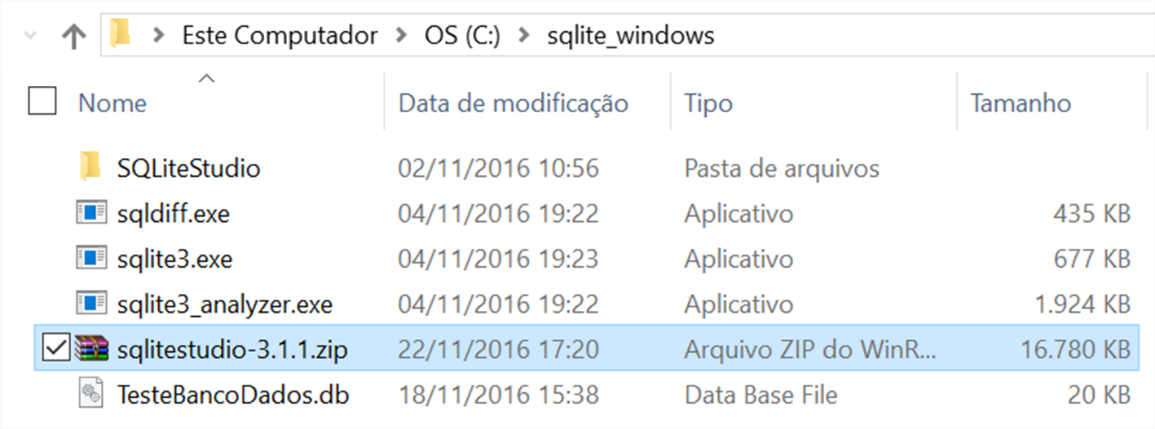
**Figura 22**

Clique no botão **Abrir pasta**, e vai ser aberta a pasta onde o arquivo foi salvo, clique com o botão do mouse sobre o arquivo **sqlitestudio-3.1.1.zip**, e selecione o item **Extrair aqui**, conforme figura abaixo:



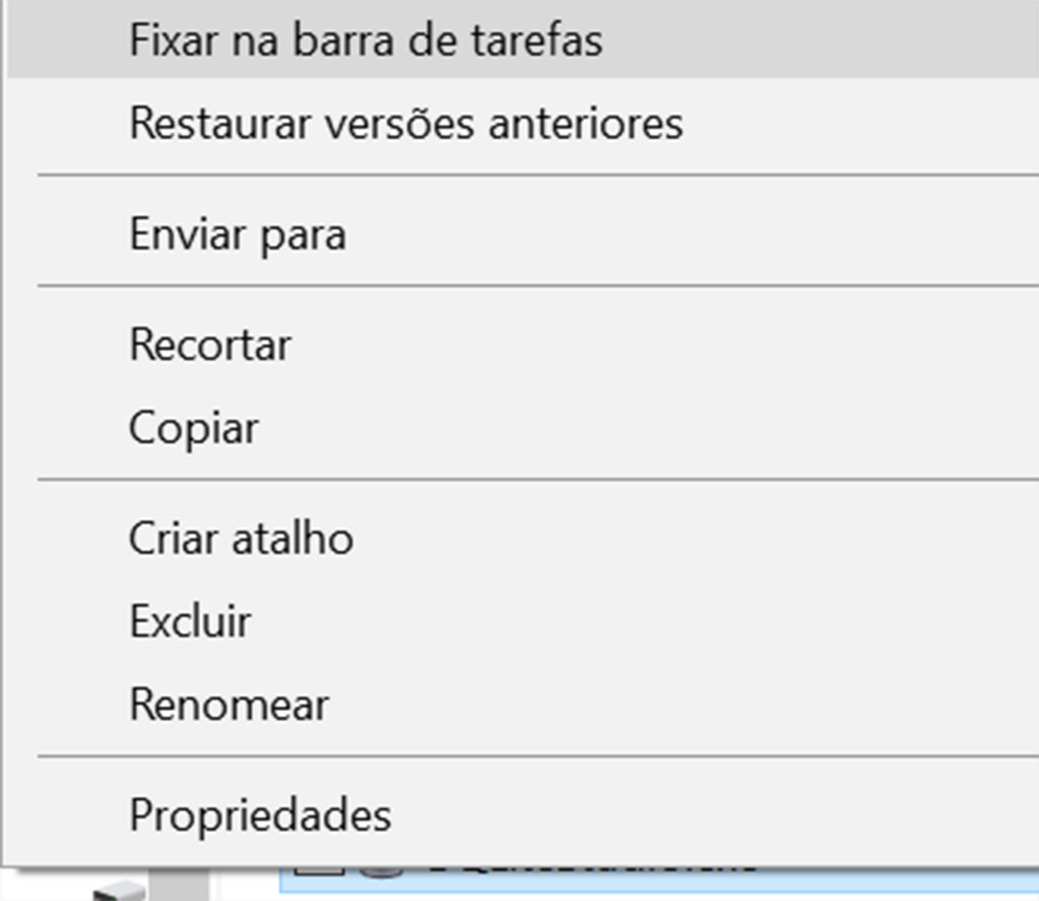
**Figura 23**

Aguarde o término da extração dos arquivos e veja como ficou, conforme figura abaixo:



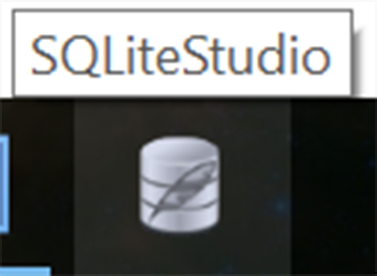
**Figura 24**

Em seguida selecione o arquivo **sqlitestudio-3.1.1.zip**, e pressione a tecla **Delete** do teclado. Em seguida clique duas vezes sobre a pasta **SQLiteStudio**, procure o arquivo **SQLiteStudio.exe**, clique com o botão do mouse sobre o mesmo e selecione o item **Fixar na barra de tarefas**, conforme figura abaixo:



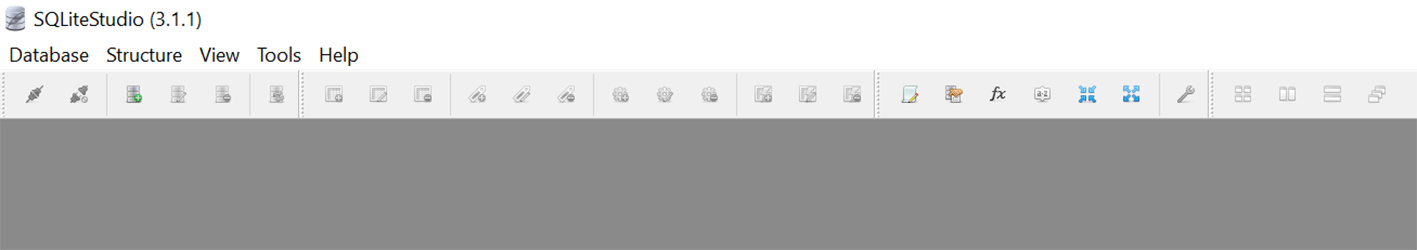
**Figura 25**

Veja como ficou, conforme figura abaixo:



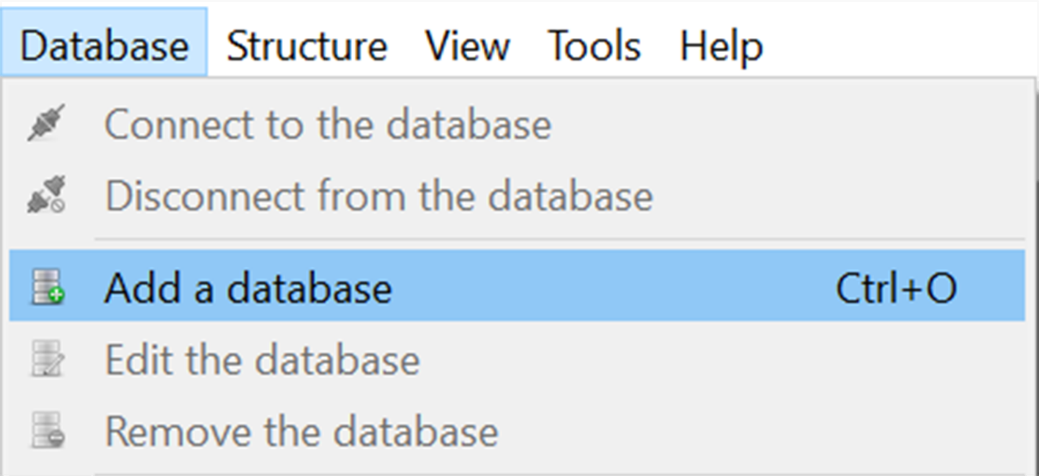
**Figura 26**

Desta forma teremos um atalho para chamar o programa, agora basta clicar sobre o ícone do **SQLiteStudio**, e vai abrir a janela do mesmo, conforme figura abaixo:



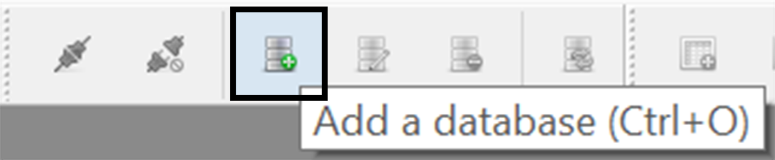
**Figura 27**

Para criarmos um banco de dados no ambiente visual, chamado **verdurasImaginaria**, clique no menu **Database**, em seguida selecione o item **Add a database**, conforme figura abaixo:



**Figura 28**

Ou pressione ao mesmo tempo as teclas **Ctrl+O**, ou ainda clique sobre o ícone da barra de ferramentas, conforme figura abaixo:



**Figura 29**

Independentemente da forma usada, vai abrir a janela **Database**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Figura 30**

No item **File**, clique sobre a pasta amarela, conforme figura abaixo:

Uma imagem contendo Padrão do plano de fundo

Descrição gerada automaticamente

**Figura 31**

Vai abrir a janela **Database file**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 32**

Selecione a pasta onde vamos salvar nosso banco de dados, neste caso vamos deixar na pasta principal do programa. Na caixa de texto **Nome:**, digite **verdurasImaginaria**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 33**

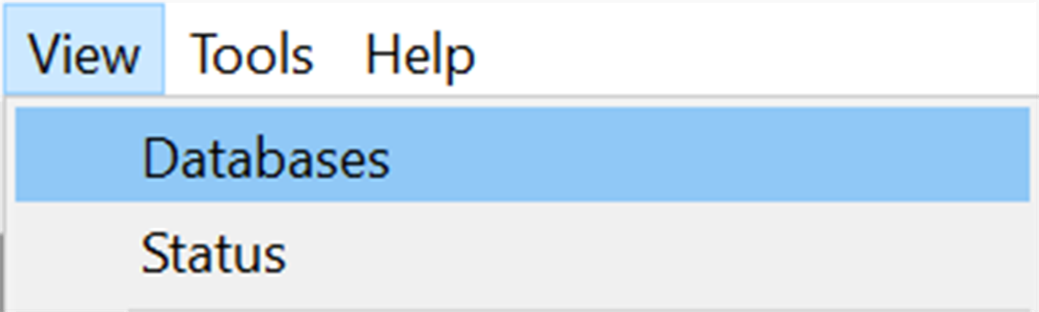
Em seguida clique no botão **Selecionar**. Veja como ficou, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Figura 34**

Clique no botão **OK**. Para ter certeza que o banco de dados foi criado, clique no menu **View**, em seguida clique no item **Databases**, conforme figura abaixo:



**Figura 35**

Vai abrir a janela **Databases**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Figura 36**

Para utilizar o nosso banco de dados, temos que conectá-lo, para isto clique com o botão do mouse sobre o nosso banco de dados, e selecione o item **Connect the database**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 37**

Veja como ficou, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Figura 38**

Agora estamos aptos a trabalhar com nosso banco de dados **verdurasImaginaria**, na criação das tabelas, visões, entre outros.

## Criar Tabela no SQLite

Vimos até o presente momento como criar nossa base de dados, repositório ou banco de dados. Agora vamos aprender a crias nossas **Tabelas**, temos duas formas de fazê-lo: no ambiente visual, e através do código **SQL**. Veremos as duas formas. A primeira é clicando com o botão do mouse sobre o item **Tables** da janela **Databases**, e selecionar a opção **Create a table**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Figura 39**

Vai abrir a janela **Strucuture**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Figura 40**

No item **Table name:**, digite **Produtos**, em seguida clique no item **Add column**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Email

Descrição gerada automaticamente

**Figura 41**

Vai abrir a janela **Column**, para digitação dos dados, conforme figura abaixo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Figura 42**

No item **Name and type**, na caixa de texto **Column name**, digite **proID**, na caixa de seleção **Data type:**, selecione o tipo **INTEGER**, no item **Constraints**, selecione o quadradinho **Primary Key**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 43**

Clique no botão **Configure**, que fica ao lado de **Primary Key**, e vai abrir a janela **Edit constraint**, nesta janela marque o quadradinho **Autoincrement**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Figura 44**

Em seguida clique no botão **Apply**. Observem que vamos retornar para a janela **Column**, ainda no item **Constraints**, selecione o quadradinho **Not Null**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 45**

Clique no botão **OK**, e veja que criamos nosso primeiro campo da tabela **verdurasImaginaria**, conforme figura abaixo:

Tabela

Descrição gerada automaticamente com confiança média

**Figura 46**

Vamos criar o próximo campo, para isto repita o procedimento para inserir outra coluna, conforme vimos no tópico anterior, na janela **Column**, no item **Name and type**, na caixa de texto **Column name**, digite **proNome**, na caixa de seleção **Data type:**, selecione o tipo **TEXT**, no item **Size**:, digite 255, no item **Constraints**, selecione o quadradinho **Not Null**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 47**

Clique no botão **OK**, e veja que criamos nosso segundo campo da tabela **verdurasImaginaria**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Figura 48**

Vamos criar o próximo campo, para isto repita o procedimento para inserir outra coluna, conforme vimos no tópico anterior, na janela **Column**, no item **Name and type**, na caixa de texto **Column name**, digite **proPrecoUnitario**, na caixa de seleção **Data type:**, selecione o tipo **REAL**, no item **Constraints**, selecione o quadradinho **Not Null**, conforme figura abaixo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Figura 49**

Clique no botão **OK**, e veja que criamos nosso terceiro campo da tabela **verdurasImaginaria**, conforme figura abaixo:

Aplicativo

Descrição gerada automaticamente com confiança média

**Figura 50**

Pronto, nossa tabela **Produtos** está completa. Para salvar a tabela, vamos clicar no botão **Commit structure changes**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente

**Figura 51**

Vai aparecer a janela **Queries to be executed**, com o código **SQL** da criação da tabela, conforme figura abaixo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Figura 52**

Clique no botão **OK**. Observem que na janela **Databases**, no item **Tables**, temos nossa tabela, basta clicar na setinha do item **Tables**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Figura 53**

A outra forma de criar uma nova tabela é usando a instrução **SQL CREATE TABLE** em qualquer banco de dados fornecido. Veja a sintaxe conforme abaixo:

**CREATE TABLE database\_name.table\_name(**

**column1 datatype PRIMARY KEY(one or more columns),**

**column2 datatype,**

**column3 datatype,**

**.....**

**columnN datatype,**

**);**

Onde **CREATE TABLE** é a palavra-chave dizendo ao sistema de banco de dados para criar uma nova tabela. O nome exclusivo ou identificador para a tabela segue após a instrução **CREATE TABLE**. Opcionalmente, podemos especificar o nome do banco de dados **database\_name** juntamente com o nome da tabela, **table\_name**. Para criar nossas tabelas, precisamos verificar se nosso banco de dados **verdurasImaginaria.db**, está conectado, para isto clique com o botão do mouse sobre o nome do nosso banco de dados, e se o item **Connect to database** estiver apagado, nosso banco de dados está conectado, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Figura 54**

Caso contrário, ou seja, se o mesmo estiver ativo, basta clicar sobre ele para conectar o banco de dados, conforme vimos na **Figura** **37**. Vamos criar mais uma tabela, para digitar o código da nova tabela no **SQL**, vamos abrir a janela **Open SQL editor**, para isto vamos até o menu **Tools**, em seguida vamos clicar no item **Open SQL editor**, ou pressione ao mesmo tempo as teclas **Alt+E**, conforme figura abaixo:

Interface gráfica do usuário

Descrição gerada automaticamente

**Figura 55**

Veja como ficou, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Figura 56**

Vamos criar uma nova tabela que vamos chamar de **DestinosExportacao**, com um campo **desID** do tipo **INTEGER** como chave primária, e **NOT NULL** e **desNome**, do tipo **VARCHAR**, com tamanho de 255 caracteres, e **NOT NULL**, são as restrições que mostram que estes campos não podem ser nulos durante a criação de registos nesta tabela, na aba **Query**, vamos digitar o código, conforme abaixo:

CREATE TABLE DestinosExportacao (

desID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

desNome VARCHAR (255) NOT NULL

);

Veja como ficou, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**Figura 57**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme figura abaixo:

Texto

Descrição gerada automaticamente

**Figura 58**

Vamos criar uma nova tabela que vamos chamar de **Vendas**, com um campo **venID** do tipo **INTEGER** como chave primária, e **NOT NULL** e **venData**, do tipo **DATE**, e um campo **desID** do tipo **INTEGER**, e **NOT NULL**, este campo é a chave estrangeira da tabela **DestinosExportacao**. São as restrições que mostram que estes campos não podem ser nulos durante a criação de registos nesta tabela, na aba **Query**, vamos digitar o código, conforme abaixo:

**CREATE TABLE** Vendas

(

venID **INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT**,

venData **DATE,**

desID **INTEGER NOT NULL**

);

Veja como ficou, conforme vimos na **Figura 57**, para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme Vimos na **Figura 58**. Vamos criar uma nova tabela que vamos chamar de **DetalhesVendas**, com um campo **venID** do tipo **INTEGER** como chave primária, e **NOT NULL** e um campo **proID** do tipo **INTEGER** como chave primária, e **NOT NULL**, e um campo **devQtd** do tipo **INTEGER**, e **NOT NULL**. Observem que temos uma chave primária dupla, desta forma ela não pode ser auto incremento. Estes dois campos, são as chaves estrangeiras das tabelas **Vendas** e **Produtos**. São as restrições que mostram que estes campos não podem ser nulos durante a criação de registos nesta tabela, na aba **Query**, vamos digitar o código, conforme abaixo:

**CREATE TABLE** DetalhesVendas

(

venID **INTEGER NOT NULL**,

proID **INTEGER NOT NULL**,

devQtd **INTEGER NOT NULL**,

**CONSTRAINT** Key1 **PRIMARY KEY** (proID,venID)

);

Veja como ficou, conforme vimos na **Figura 57**, para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme Vimos na **Figura 58**. Vejam que na janela **Databases**, no item **Tables**, temos nossas tabelas, basta clicar na setinha do item **Tables**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Figura 59**

Você pode obter informações completas sobre uma tabela basta clicar duas vezes com o botão do mouse sobre o nome da tabela na janela **Databases**, no item **Tables**, e vai aparecer os dados da tabela, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**Figura 60**

Veja que temos várias abas com todos os detalhes da tabela, basta navegar nas mesmas para obter as informações que precisa. Para fechar a janela, basta clicar com o botão do mouse sobre a aba com o nome da tabela que fica no canto inferior esquerdo e selecione o item **Close all window**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Figura 61**

Isto vale para todas as janelas. Para fechar todas as janelas, basta selecionar o item **Close all window**. Acabamos de criar nossas tabelas, agora vamos popular ou inserir dados nas mesmas.

## Inserção de Dados – SQLite INSERT INTO Query

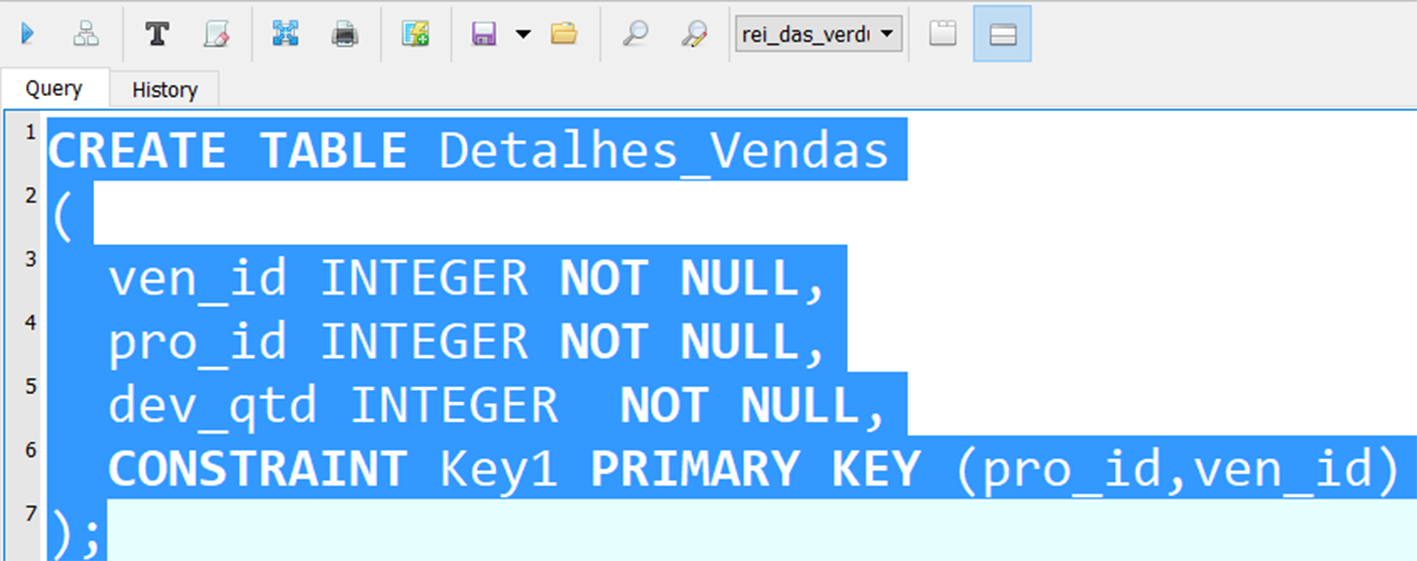
O comando **INSERT INTO** é usado para adicionar novas linhas de dados em uma tabela no banco de dados, a sintaxe é a seguinte:

**INSERT INTO** tabela (coluna1, coluna2, coluna3, colunaX) **VALUES** (XX, XX, XX);

Ou

**INSERT INTO** tabela (coluna1, coluna2, coluna3, colunaX) **VALUES** (XX, XX, XX), (XX, XX, XX);

Então vamos inserir os dados na tabela **DestinosExportacao**. Verifique se o banco de dados **verdurasImaginaria.db**, está conectado conforme vimos na **Figura 54**, em seguida vamos abrir a janela **Open SQL editor**, para isto vamos até o menu **Tools**, em seguida vamos clicar no item **Open SQL editor**, ou pressione ao mesmo tempo as teclas **Alt+E**, conforme vimos na **Figura 55**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme figura abaixo:



**Figura 62**

Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, usando o primeiro método de inserção de dados, vamos digitar o código, conforme abaixo:

**INSERT INTO** DestinosExportacao (desNome) **VALUES** ('EUA')

Veja como ficou, conforme figura abaixo:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

**Figura 63**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se os dados foram inclusos na tabela **DestinosExportacao**, vamos utilizar o comando **SELECT**, que nada mais é do que uma conversa com o banco de dados, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**SELECT \* FROM** DestinosExportacao

Veja como ficou, conforme figura abaixo:

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

**Figura 64**

Este comando **SQL**, é formado por dois grupos de palavras: **SELECT** **\*** e **FROM** Produtos, também conhecidos como cláusulas, onde estamos especificando que queremos que seja mostrado todas as colunas, através da cláusula **SELECT** **\***, e o nome da tabela, com a cláusula **FROM**. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:

Interface gráfica do usuário

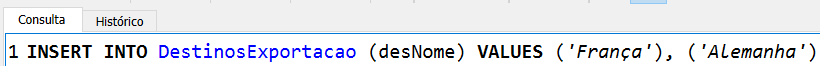
Descrição gerada automaticamente

**Figura 65**

Observem que com este método vamos ter que inserir os dados um a um, o que seria muito cansativo e impensável, caso tivermos muitas linhas ou tuplas a serem incluídas. Por este motivo vamos utilizar o segundo método. Vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**INSERT INTO** DestinosExportacao (desNome) **VALUES** ('França'), ('Alemanha')

Veja como ficou, conforme figura abaixo:



**Figura 66**

Observem que não inserimos os campo **desID**, pois o mesmo é chave primária auto incremento, ou seja, o valor vai ser adicionado automaticamente. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se todos os dados foram inclusos na tabela **DestinosExportacao**, vamos utilizar o comando **SELECT**, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme vimos na **Figura 64**. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:

Interface gráfica do usuário, Texto, Aplicativo

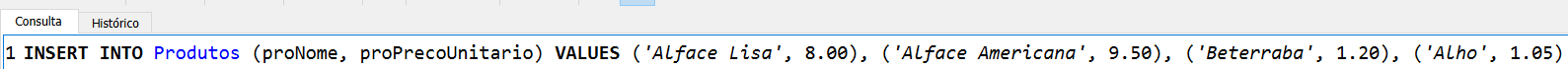
Descrição gerada automaticamente

**Figura 67**

Veremos o comando **SELECT** em detalhes, mais adiante. Agora vamos inserir os dados da tabela **Produtos**, para isto, vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**INSERT INTO** Produtos (proNome, proPrecoUnitario) **VALUES** ('Alface Lisa', 8.00), ('Alface Americana', 9.50), ('Beterraba', 1.20), ('Alho', 1.05)

Veja como ficou, conforme figura abaixo:



**Figura 68**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se todos os dados foram inclusos na tabela **Produtos**, vamos utilizar o comando **SELECT**, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**SELECT \* FROM** Produtos

Deve ficar conforme vimos na **Figura 64**. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:

Tabela

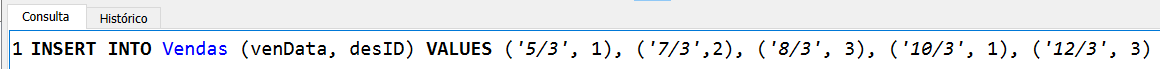
Descrição gerada automaticamente com confiança média

**Figura 69**

Agora vamos inserir os dados da tabela **Vendas**, para isto, vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**INSERT INTO** Vendas (venData, desID) **VALUES** ('5/3', 1), ('7/3',2), ('8/3', 3), ('10/3', 1), ('12/3', 3)

Veja como ficou, conforme figura abaixo:



**Figura 70**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se todos os dados foram inclusos na tabela **Vendas**, vamos utilizar o comando **SELECT**, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**SELECT \* FROM** Vendas

Deve ficar conforme vimos na **Figura 64**. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:

Interface gráfica do usuário, Aplicativo, Tabela

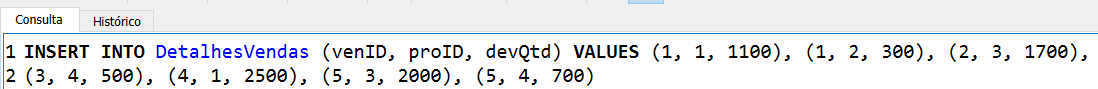
Descrição gerada automaticamente

**Figura 71**

Agora vamos inserir os dados da tabela **DetalhesVendas**, para isto, vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**INSERT INTO** DetalhesVendas (venID, proID, devQtd) **VALUES** (1, 1, 1100), (1, 2, 300), (2, 3, 1700), (3, 4, 500), (4, 1, 2500), (5, 3, 2000), (5, 4, 700)

Veja como ficou, conforme figura abaixo:



**Figura 72**

Um detalhe que deve ser observado nesta tabela, é que os campos **venID** e **proID**, apesar de serem chaves primárias, não são auto incremento, desta forma obrigatoriamente temos que informar os valores dos mesmos. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se todos os dados foram inclusos na tabela **DetalhesVendas**, vamos utilizar o comando **SELECT**, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**SELECT \* FROM** DetalhesVendas

Deve ficar conforme vimos na **Figura 64**. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:

Tabela

Descrição gerada automaticamente

**Figura 73**

Pronto, todas as nossas tabelas estão populadas, vamos agora estudar os comandos **SQL** do **SQLite**, mais a fundo.

Parei aqui

## Alteração de Dados – SQLite UPDATE Query

Esta query é usada para modificar ou alterar os registros existentes em uma tabela. Devemos usar a cláusula **WHERE** com **UPDATE** para atualizar as linhas selecionadas, caso contrário, todas as linhas serão atualizadas. Veja a sintaxe conforme abaixo:

**UPDATE** nome\_tabela

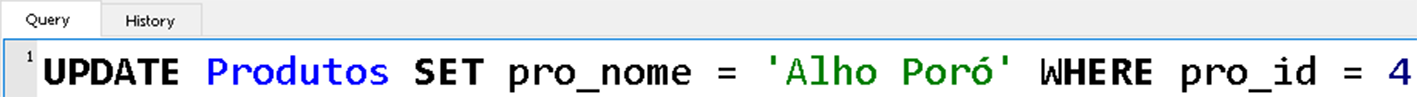
**SET** column1 = value1, column2 = value2...., columnN = valueN

**WHERE** [condição]

Podemos combinar **N** números de condições usando os operadores **AND** ou **OR**. Vamos utilizar a tabela **Produtos**, para nossos exemplos, vamos alterar o nome da verdura com **proID** igual 4, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

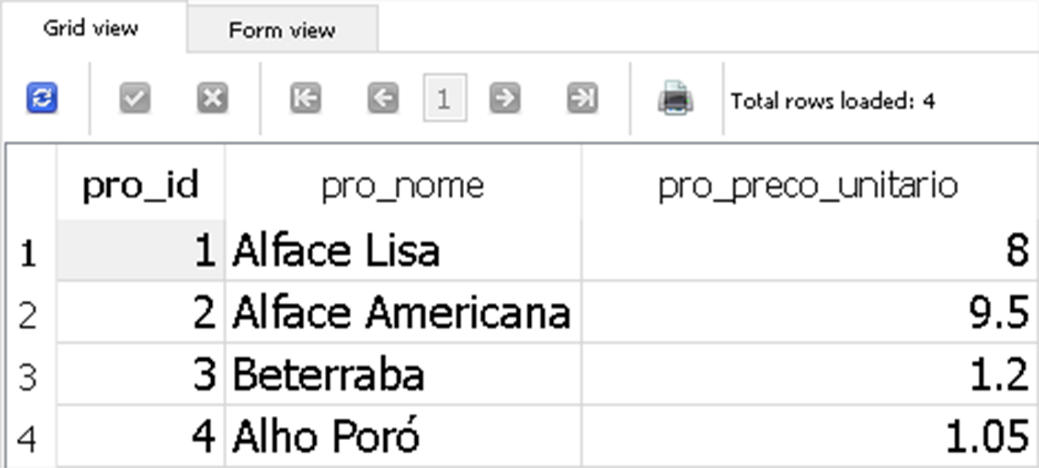
**UPDATE** Produtos **SET** proNome = 'Alho Poró' **WHERE** proID = 4

Veja como ficou, conforme figura abaixo:



**Figura 113**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se os dados foram alterados, use o comando **SELECT** para exibir todos os dados da tabela **Produtos**, conforme vimos na **Figura 69**. Vejam que os dados solicitados e alterados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:



**Figura 114**

## Excluir Dados – SQLite DELETE Query

Esta query é usada para excluir os registros existentes em uma tabela. Devemos usar a cláusula **WHERE** com a cláusula **DELETE** para excluir linhas selecionadas, caso contrário, todos os registros seriam excluídos. Veja a sintaxe conforme abaixo:

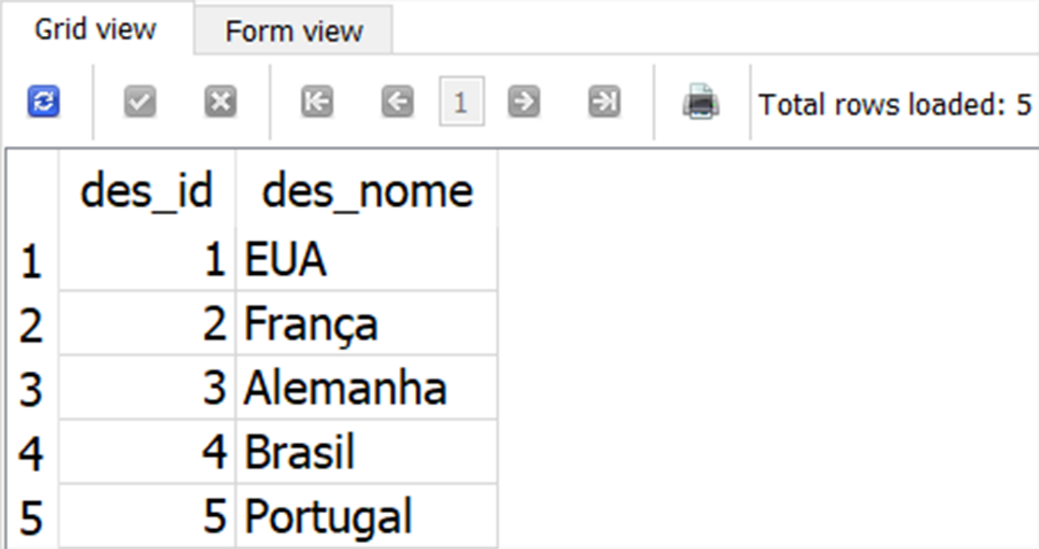
**DELETE** **FROM** nome\_tabela

**WHERE** [condição]

Podemos combinar **N** números de condições usando os operadores **AND** ou **OR**. Vamos utilizar a tabela **DestinosExportacao**, para nossos exemplos, vamos inserir mais dois países nesta tabela. Vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**INSERT INTO** DestinosExportacao (desNome) **VALUES** ('Brasil'), ('Portugal')

Veja como ficou, conforme vimos na **Figura 66**. Para verificar se todos os dados foram inclusos na tabela **DestinosExportacao**, vamos utilizar o comando **SELECT**, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme vimos na **Figura 64**. Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:

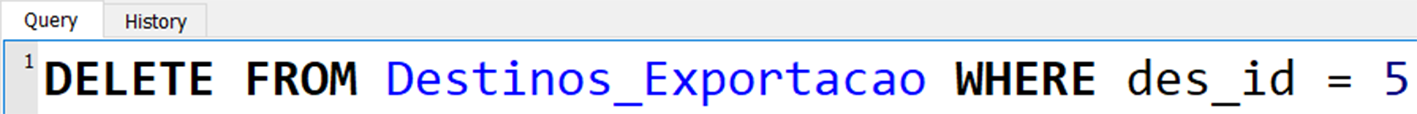


**Figura 115**

Vamos excluir o país com o código número 5, para isto vamos retornar à janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

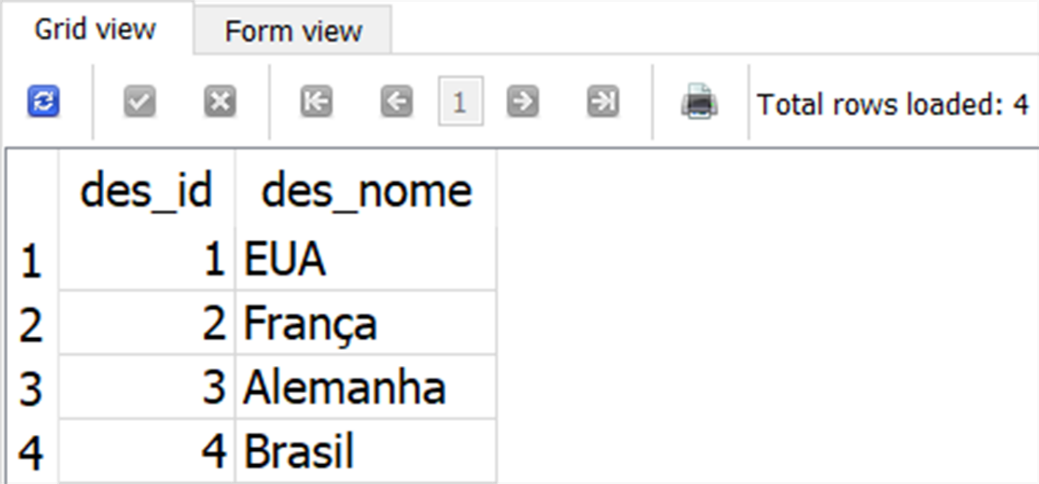
**DELETE FROM** DestinosExportacao **WHERE** desID = 5

Veja como ficou, conforme figura abaixo:



**Figura 116**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Para verificar se os dados foram excluídos, use o comando **SELECT** para listar os dados, conforme vimos na **Figura 64**. Vejam que os dados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:



**Figura 117**

Observem que uma vez deletado um registro, o código deste registro não será mais utilizado. Caso tenhamos que inserir um novo registro, a sequência seria o código de número 6, e não número 5, pois os códigos são únicos. Muito cuidado ao excluir registros, o ideal é fazer um backup das tabelas antes de usar este comando. Outra ponto que devemos tomar cuidado e que se esquecermos de usar a cláusula **WHERE**, vamos deletar todos os registros da tabela.

## Cláusula Joins no SQLite

Esta cláusula é usada para combinar registros de duas ou mais tabelas em um banco de dados. Um **JOIN** ou **junção**, é um meio para combinar campos de duas tabelas usando valores comuns a cada um. O **SQL** define três tipos principais de junções:

* A CROSS JOIN.
* A INNER JOIN.
* A OUTER JOIN.

Antes de prosseguir, vamos usar como exemplo todas as tabelas que criamos.

## Usando CROSS JOIN – Junção Cruzada

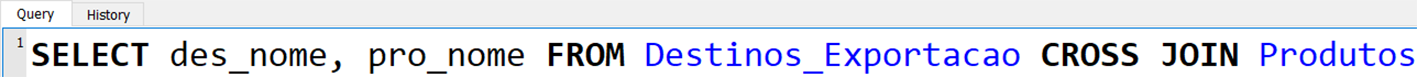
Corresponde a cada linha da primeira tabela com cada linha da segunda tabela. Se as tabelas de entrada tiverem colunas x e y, respectivamente, a tabela resultante terá colunas x + y. Como **CROSS JOINs** tem o potencial de gerar tabelas extremamente grandes, deve-se ter cuidado e usá-las somente quando apropriado. Veja a sintaxe conforme abaixo:

**SELECT** ... **FROM** tabela1 **CROSS JOIN** tabela2 ...

Com base nas tabelas que criamos, vamos fazer uma junção cruzada com as tabelas **DestinosExportacao** e **Produtos**, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

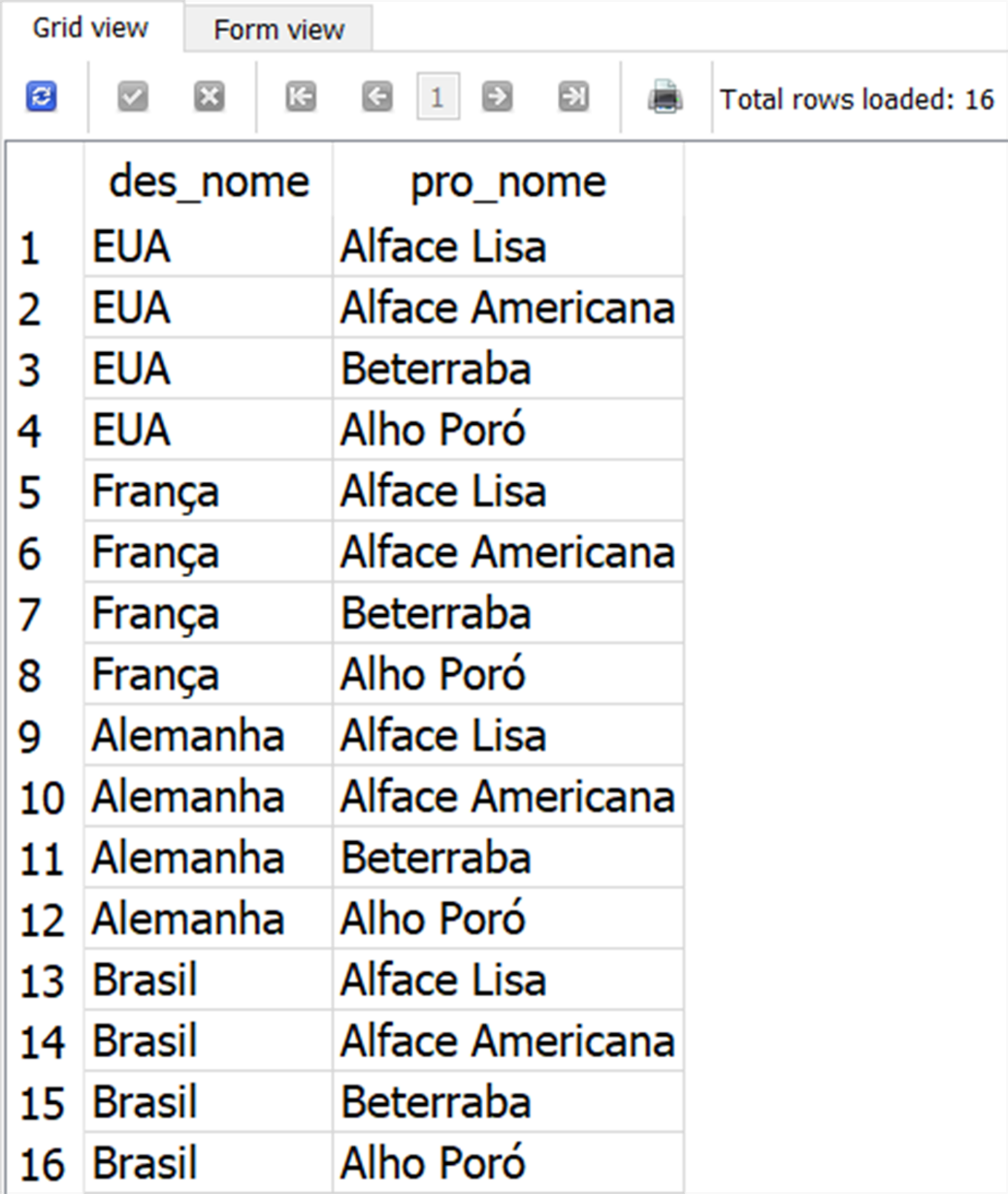
**SELECT** desNome, proNome **FROM** DestinosExportacao **CROSS** JOIN Produtos

Veja como ficou, conforme figura abaixo:



**Figura 132**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:



**Figura 133**

## Usando INNER JOIN – Junção

Cria uma nova tabela de resultados, combinando valores de coluna de duas tabelas (**tabela1** e **tabela2**) com base no predicado de junção. A consulta compara cada linha da **tabela1** com cada linha da **tabela2** para encontrar todos os pares de linhas, que satisfazem o predicado de junção. Quando o predicado de junção é satisfeito, valores de coluna para cada par de linhas correspondentes de A e B são combinados em uma linha de resultado. Este é o tipo mais comum de associação e é o tipo padrão de junção. Podemos usar a palavra-chave **INNER** opcionalmente. Veja a sintaxe conforme abaixo:

**SELECT** ... **FROM** tabela1 **[INNER] JOIN** tabela2 **ON** expressão\_condicional...

Para evitar redundância e manter a expressão mais curta, as condições **INNER JOIN** podem ser declaradas com uma expressão **USING**. Esta expressão especifica uma lista de uma ou mais colunas, veja a sintaxe conforme abaixo:

**SELECT** ... **FROM** tabela1 **JOIN** tabela2 **USING** ( column1 ,... ) ...

Um **NATURAL JOIN** é similar a um **JOIN...USING**, a diferença básica e que ele automaticamente testa a igualdade entre os valores de cada coluna que existe em ambas as tabelas, veja a sintaxe conforme abaixo:

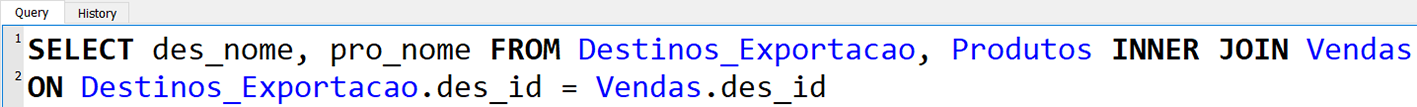
**SELECT** ... **FROM** tabela1 **NATURAL** **JOIN** tabela2...

Com base nas tabelas **DestinosExportacao** e **Produtos**, vamos fazer uma junção com estas tabelas, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**SELECT** desNome, proNome **FROM** DestinosExportacao, Produtos **INNER JOIN** Vendas

**ON** DestinosExportacao.desID = Vendas.desID

Veja como ficou, conforme figura abaixo:



**Figura 134**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:



**Figura 135**

Com base em todas as tabelas que criamos, vamos fazer uma junção com estas tabelas, onde vamos buscar todos os produtos vendidos para a Alemanha, no dia 12/3, para isto vamos retornar a janela **Open SQL editor**, em seguida vamos selecionar o código que está digitado nesta janela, se tiver algum, basta pressionar ao mesmo tempo as teclas **Ctrl+A**, e veja que o código foi selecionado, conforme vimos na **Figura 62**. Agora pressione a tecla **Delete**, do teclado e veja que nossa janela **Open SQL editor**, vai ficar vazia novamente, no local onde o cursor do mouse está piscando, vamos digitar o código, conforme abaixo:

**SELECT** Vendas.desID, venData, desNome, DetalhesVendas.proID, proNome, proPrecoUnitario, devQtd

**FROM** Vendas, Produtos, DestinosExportacao **INNER JOIN** DetalhesVendas

**ON** DestinosExportacao.desID = Vendas.desID

**AND**

DetalhesVendas.proID = Produtos.proID

**AND**

DestinosExportacao.desID = 3

**AND**

Vendas.desID = 3

**AND**

Vendas.venData = **'12/3'**

Outra forma de digitar o código, conforme abaixo:

**SELECT** Vendas.desID, venData, desNome, DetalhesVendas.proID, proNome, proPrecoUnitario, devQtd

**FROM** Vendas, Produtos, DestinosExportacao, DetalhesVendas

**WHERE** DestinosExportacao.desID = Vendas.desID

**AND**

DetalhesVendas.proID = Produtos.proID

**AND**

DestinosExportacao.desID = 3

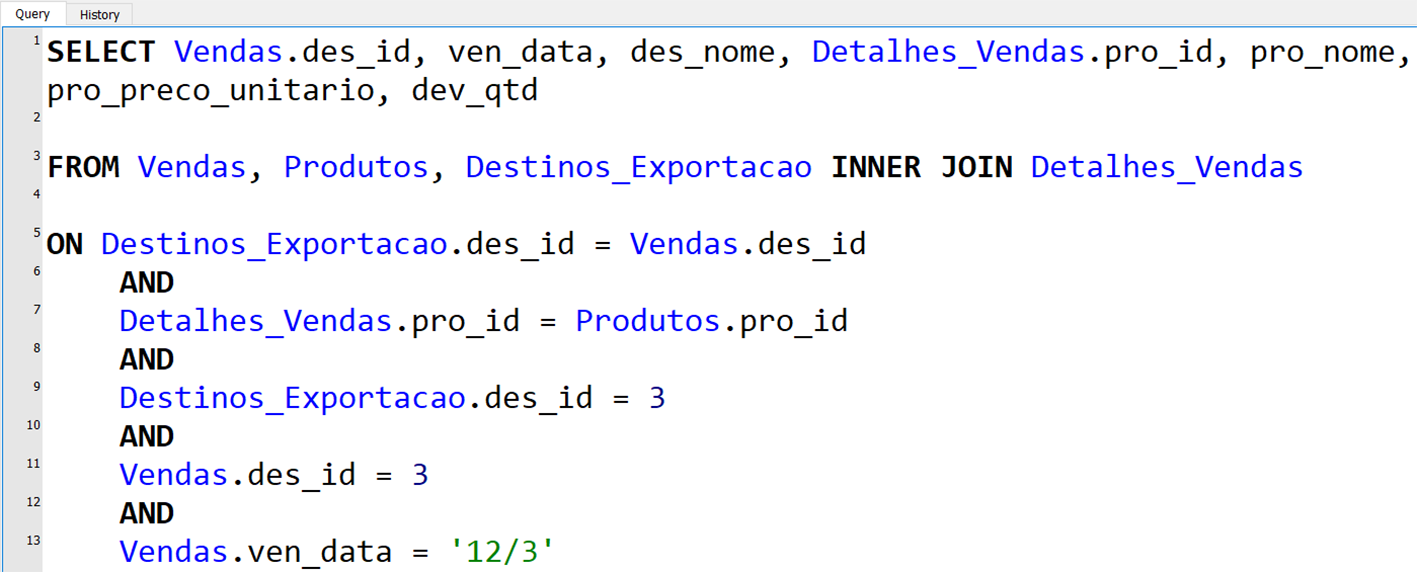
**AND**

Vendas.desID = 3

**AND**

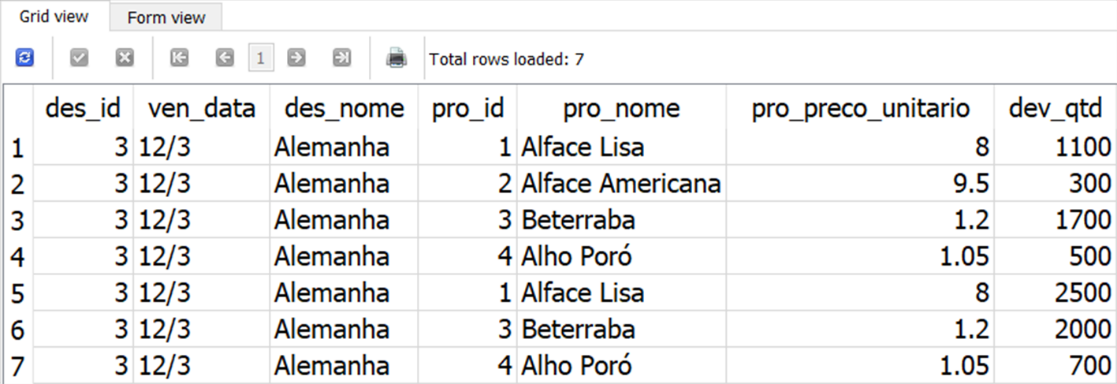
Vendas.venData = **'12/3'**

Independe da forma como foi digitado, vai ter o mesmo resultado. Veja como ficou, conforme figura abaixo:



**Figura 136**

Para executar a **Query**, vamos clicar no botão do triangulo azul **Execute query(F9) – Executar query(F9)**, ou pressionar a tecla **F9**, conforme vimos na **Figura 58**. Vejam que os dados solicitados vão aparecer na janela **Grid view**, que fica logo abaixo da janela **Open SQL editor**, conforme figura abaixo:



**Figura 137**