

Guia SQL



Gabriel Richter

Sumário

Motivo de Aprender SQL	5
Introdução ao Pensamento Analítico	5
Como Manipular esses Dados.....	6
Introdução a Modelagem de Dados.....	7
O que é o Modelo Conceitual	7
O que é o Modelo Lógico	8
O que é o Modelo Físico.....	9
Normalização dos Dados.....	10
Significado da sigla SQL.....	10
Qual Ferramenta Usar?.....	11
SQL Fundamentos dos Dados	12
Tipos de dados Literais e Caracteres.....	12
Tipos de dados Numéricos.....	13
Tipos de dados Temporais.....	14
Tipos de dados Booleanos.....	15
Palavras Reservadas.....	15
Conceito de Chaves	16
Operadores Relacionais e Matemáticos	17
Operadores Lógicos	18
SQLite Funcionalidades.....	19
Interface do SQLite.....	19
Primeira Base de Dados	20
Consultas Básicas.....	23
Syntax e Select.....	23
Consultas com Where.....	26
Trabalhando com Ordenação.....	29
Instruções com Tabelas.....	30
Relacionando Tabelas	33
SQL Server Funcionalidades	39
Contexto de Negócio em Mercado	39
Fundamentos das Tabelas.....	40
Palavras Reservadas para Tabelas.....	41
Alterar e Modificar Colunas.....	43
Funções e Apelidos.....	46
Instruções de Agrupamento	48

Sub Consulta.....	49
Junções	50
Restaurando uma base de dados.....	56
Glossário.....	60
Bibliografia.....	62
Informações Adicionais.....	63



Objetivo

Resumir a linguagem SQL para leigos, aprender com exemplos simples, rápidos e com alguns exercícios, na verdade vários!

Resumo

Esse mini guia está dividido em três partes.

1. Introdução: O capítulo um, nesse capítulo será citado um pouco sobre negócio e SQL, como ela funciona e download do SQLite. Recomendo acessar o glossário no final desse guia para entender um pouco mais dos termos que irei utilizar nesse guia. *Talvez eu venha futuramente complementar com NoSQL como o MongoDB.*
2. Primeiros Comandos: No capítulo dois, será a parte introdutória as primeiras consultas e criar a primeira Base de Dados tudo com um software leve, grátis e móvel, o SQLite.
3. Softwares Profissionais: Não que o SQLite não seja um software poderoso, você vai entender ainda nesse capítulo um, o motivo de usar o SQLite, mas, também durante o decorrer desse mini guia, será utilizado um software novo, o SQL Server, o terceiro software mais utilizado no mercado.

Abstract

This mini guide is divided into three parts.

1. Introduction: Chapter one, in this chapter will be mentioned a little about bussiness and SQL, how it works and download SQLite. I recommend accessing the glossary at the end of this guide to understand a little more of the terms will be using in this guide. Maybe in the future I will complement it with NoSQL like MongoDB.
2. First Commands: In chapter two, will be like first queries and creating a first Database all with a free, soft and mobile software, the SQLite.
3. Professional Software's: Second data base software used in this guide is SQL Server, a complete *dbsm* for more complex queries.

Motivo de Aprender SQL

SQL é uma das poucas coisas desse mundo que consegue chegar próximo à unanimidade, desenvolvida na década de 1970 por funcionários da atual IBM, e até hoje é utilizada em qualquer Banco de Dados Relacional.

Qualquer estabelecimento, escola e até empresas, possuem e armazenam seus dados de várias formas, mas, todas armazenam.

Introdução ao Pensamento Analítico

Realizar uma consulta em uma base de dados é fácil, você vai aprender nesse guia, mas você deve entender o porquê você está fazendo essa consulta, você está ajudando um cientista de dados em um projeto que requer tais manipulações, você apenas quer analisar os dados pois um superior pediu um resultado, ou apenas treinar SQL.

A maioria das ferramentas no mercado nasceu com o intuito de resolver um problema que o indivíduo estava passando no momento em que foi proposto a ideia dessa ferramenta, se ainda não estiver convencido, olhe o exemplo do *visual studio code*, um ambiente de desenvolvimento integrado muito completo por sinal.

Fazendo uma pequena pesquisa pela internet, vemos muito ênfase na frase “Loja de extensões imensa”, entre outras frases, ou seja, essa ferramenta, possui muitas extensões onde por meio delas é possível manipular *scripts* de diversas formas, não vou entrar mais em detalhes pois acho que você já entendeu.

Recomendo que ao decorrer das atividades desse documento, você imagine-se em um ambiente de trabalho, e após realizar as determinadas atividades, faça um relatório, ou entenda mais por que você fez aquele exercício.

Fazendo esses exercícios extra de pensar no negócio, garanta uma preparação antes mesmo de desenvolver outros projetos sem ser especificamente com SQL, ou seja, você recebeu um problema, tente interpretar esse problema que você recebeu e quebre em tarefas menores, “Quero que você faça uma base de dados para frutas”, você realmente entendeu essa pergunta?

Pergunte a unidade de negócio que lhe fez essa pergunta o motivo, entender a causa raiz ajuda a encontrar potenciais métodos e formas para conseguir concluir esse desafio, pois você pode desenvolver uma solução que na maioria das vezes não vai servir ao propósito, pois existem várias formas de desenvolver uma base de dados para frutas, mas que tipos de frutas, ou o que deve conter nas tabelas dessa base, inclusive, essa unidade que lhe pediu esse problema irá validar suas propostas, definir um escopo fechado para futuramente não entrar em um ciclo infinito o que era para ser apenas uma base de dados.

Existem muitas formas de organizar um problema de negócio, mas não irei abordar mais esse assunto aqui, somente nas atividades, onde recomendo entender a pergunta, pois existem

perguntas que requerem um relatório, ou seja, você escolhe qual ferramenta fazer esse relatório, e como você vai fazer, qual ferramenta vai utilizar, Excel, Word, vai fazer uma aplicação, vai enviar por e-mail, enfim, muitas possibilidades.

Como Manipular esses Dados

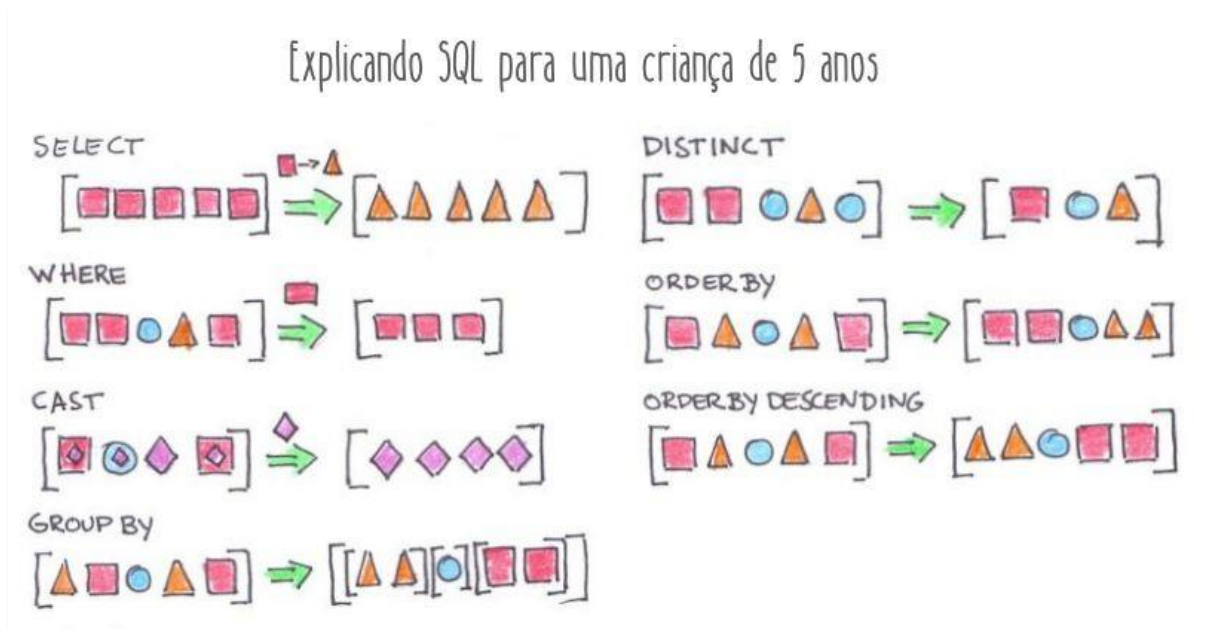


Figura 1: SQL Para Crianças, Imagem do LinkedIn, com algumas instruções importantes.

Para conseguir manipular esses tais dados, usamos um SGBD que é um software, com esse software para gerenciar nossas bases de dados, existem inúmeras vantagens, como:

- Visualização interativa dos dados.
- Compartilhamento dos dados.
- Backup, segurança e recuperação de falhas.
- Flexibilidade e padronização.

Atualmente existem vários bancos de dados, cada banco de dado com suas características e peculiaridades, por isso uma boa análise é essencial antes de querer estudar ou até mesmo montar uma base de dados para uma empresa específica.

Ao longo do documento vai ser apresentado dois desses softwares para que você possa ter duas ferramentas na sua caixinha de ferramentas, mas existem muitas outras ferramentas legais que podem ser estudadas e aplicadas na indústria, mas a melhor entre elas para começar seus estudos é o *SQLite* que vamos abordar ao decorrer do documento.

Além da ferramenta tem os indivíduos que utilizam essas ferramentas, como o *DBA*, “*Database Administrator*” ou em português, o administrador de banco de dados, a responsabilidade desse indivíduo é toda a administração, instalação, implementação e configuração do banco de dados garantindo continuamente uma melhor performance e monitoramento.

Além do *DBA*, existe também o Analista de dados, que é responsável por projetar a base de dados, ou seja, o que realmente vai ser salvo na base de dados, o desenvolvimento das tabelas e estruturas em um SGBD escolhido, juntamente com o Analista, vem o desenvolvedor, que vai usar de interações dos usuários para desenvolver uma forma de ligar a base de dados com a aplicação em específico que ele está desenvolvendo, por exemplo um formulário de inscrição, que nada mais é que uma aplicação ligada a um banco de dados com as informações de outros usuários.

E por fim o usuário, o indivíduo que vai utilizar a aplicação desenvolvida pelo programador no seu dia a dia.

Introdução a Modelagem de Dados

Antes de desenvolver um banco de dados, é essencial que eu tenha o projeto ou seja o modelo do qual vai ser criar bem elaborado tanto para manutenção, futuras implementações com o intuito de minimizar quaisquer futuros erros ou problemas durante a fase de desenvolvimento do banco de dados com a linguagem *sql* até a fase de inserção de dados na aplicação que o programador desenvolveu.

Resumidamente, como vai ser feito a relação das tabelas seguindo algumas normas, como a inserção vai ser feita, como eliminar duplicações e redundâncias, todos esses passos são muito importantes antes mesmo de desenvolver a base de dados que vão suportar essas tabelas, esse conceito está diretamente ligado com o pensamento analítico, ou seja, como eu vou resolver o determinado problema que requer uma base de dados, mas como vou desenvolver essa base de dados, o objetivo é gerar informação com os dados.

Uma modelagem mal feita pode acarretar inúmeros problemas e até mesmo comprometer o projeto, aplicação não funcional, baixo desempenho, entre outros, já uma modelagem bem feita o projeto vai ter muito mais eficiência e menor índice de erros durante qualquer etapa do desenvolvimento da base, inserção dos dados e ligação com a aplicação real.

Mas o que é um modelo afinal, podemos associar um modelo como um caderno, onde nele eu guardo as informações necessárias que irão me ajudar na tomada de decisão para concluir algum problema, por exemplo eu anoto no meu caderno algumas fórmulas matemáticas, com essas formulas consigo desenvolver os problemas da formula especifica e até generalizar para outros problemas, ou seja, nesse caderno deve ter detalhes de como eu uso essa formula, para que serve, o que eu espero, detalhadamente, para que outras pessoas usem o caderno e consigam tirar alguma possível conclusão do problema que estão no momento.

O que é o Modelo Conceitual

Resumidamente o modelo conceitual é a primeira etapa de um projeto de banco de dados, o principal objetivo do analista é recolher as informações do modelo de negócio que o mesmo está inserido para desenvolver os primeiros modelos do banco de dados, ou seja, ele descreve os fatos do mundo real, ou seja, o que realmente vai ser armazenado.

Um exemplo em uma lojinha de peças de roupa, vai ser armazenado as peças de roupa, a cor, tamanho, modelo, fabricante, tecido, entre outros dados que fazem sentido serem armazenados para o negócio, como também os dados do cliente em específico que comprou aquela peça de roupa, ou seja, o analista também trabalha com os possíveis relacionamentos das tabelas.

O modelo conceitual não está ligado com o banco de dados, ele na verdade é uma forma de graficamente representar a realidade e seus relacionamentos.

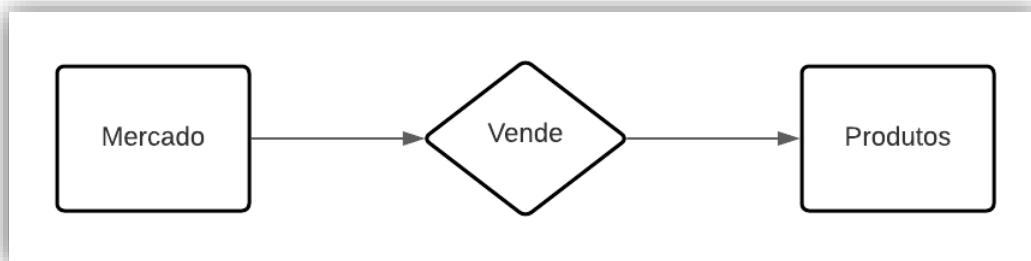


Figura 2: Diagrama Conceitual simples desenvolvido no LucidCharts.

Uma boa pratica é sempre utilizar palavras no singular, ou seja, mercado vende produto, mas no diagrama foi representado produtos no plural.

O que é o Modelo Lógico

O modelo lógico nada mais é que a estrutura que vai armazenar os dados dentro do banco de dados, nessa etapa do desenvolvimento vai ser levado em consideração todo o desenvolvimento do modelo conceitual, nessa etapa é desenvolvido todo o relacionamento das tabelas como no diagrama abaixo.

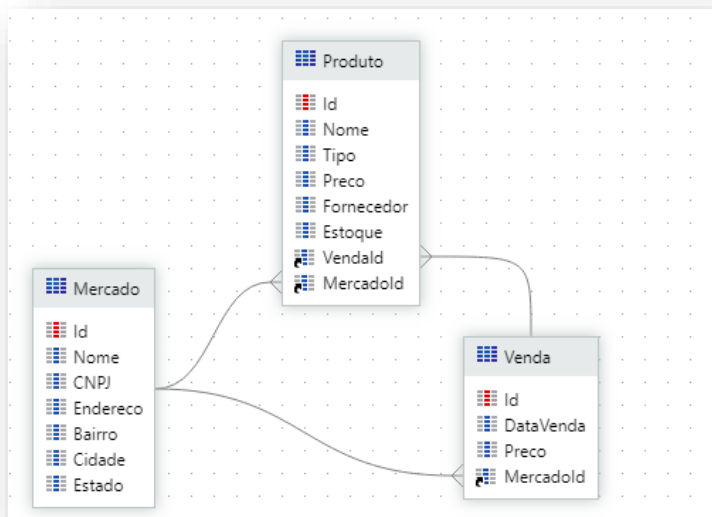


Figura 2: Diagrama Lógico simples.

Nesse diagrama foi utilizado as ligações popularmente chamadas de pé de galinha, esse diagrama de entidade basicamente fala que tenho muitas ligações, por exemplo, tem muitas vendas do mercado em específico representado pelo final da setinha que parece o pé da galinha, também isso para venda, ou seja, tenho uma venda de vários produtos, esse é o conceito um para muitos, consequentemente existe mais conceitos, como muitos para um, muitos para muitos, etc.

Também está representado no diagrama da figura 2 as chaves primárias e estrangeiras, onde o Id é a chave primária da tabela venda, e o MercadoId é a chave estrangeira na tabela venda, ou seja, o id do mercado está na tabela venda fazendo uma referência do mercado na venda específica, entraremos mais em detalhes sobre chaves primárias e estrangeiras.

Nessa fase já é escolhido qual *SGBD* vai suportar a base de dados para começar a desenvolver no modelo físico.

O que é o Modelo Físico

O modelo físico nada mais é que como irá ser as configurações da tabela e seus devidos relacionamentos, com o modelo lógico usa-se o mesmo para escrever com a linguagem SQL a criação da tabela, os relacionamentos, etc.

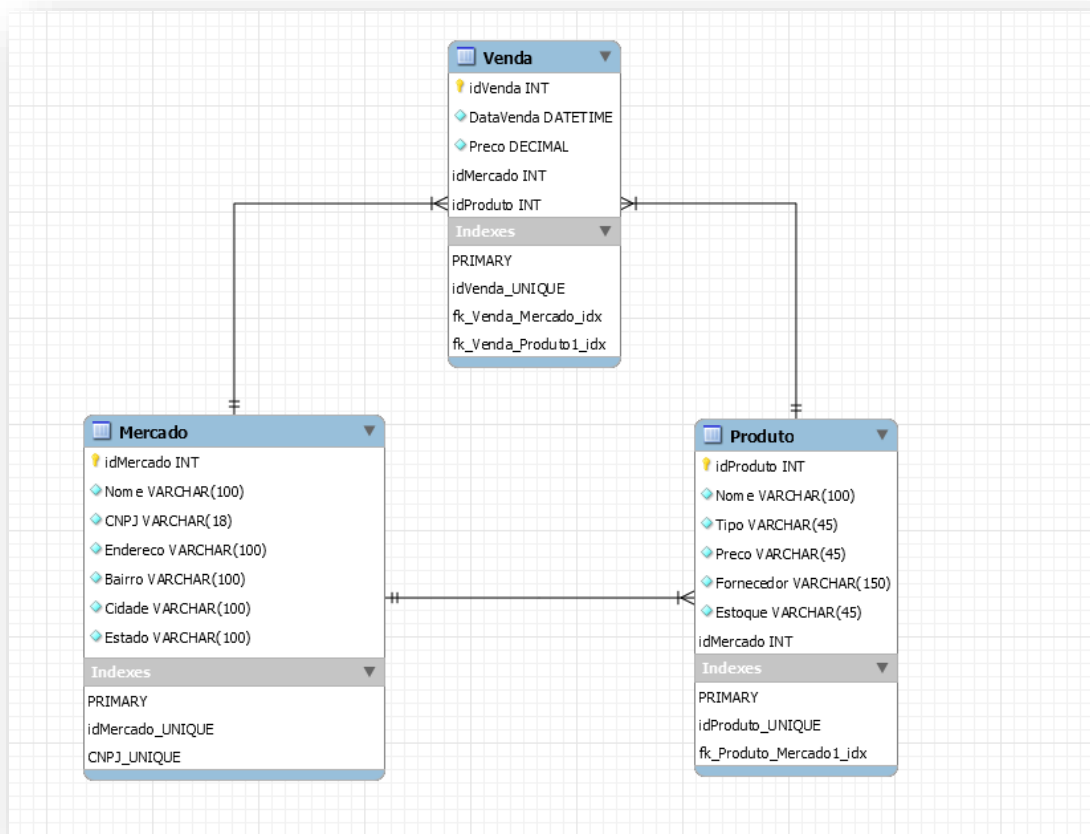


Figura 3: Diagrama Físico desenvolvido no MySQL Workbench.

Agora basta pegar esse modelo e desenvolver com a linguagem SQL.

Normalização dos Dados

Para um projeto de banco de dados tenha sucesso e os bugs e erros sejam a última preocupação é preciso proporcionar integridade e consistência para isso que existe o processo de normalização, ele é um tema bem profundo que seu objetivo principal é um conjunto de regras aplicadas sobre as tabelas e seus relacionamentos buscando qualquer anomalia.

Essas anomalias podem gerar vários problemas como grupos repetidos de dados, dependências, redundância de dados desnecessários, perdas de informações etc.

O processo de normalização segue alguns objetivos como analisar as tabelas e organizá-las de forma que sua estrutura seja simples, evitar repetições, uma forma adequada de representar o que é realmente importante, garantir a integridade e a organização das tabelas de uma forma mais eficiente.

Por sua vez podemos dividir as normalizações em algumas etapas, como a primeira forma normal, que nada mais é dividir a tabela em uma ou mais tabelas para evitar repetições, estabelecer o relacionamento e a cardinalidade entre as tabelas geradas, verificar relacionamentos um para muitos, ou seja, uma venda de muitos produtos, etc.

Já a aplicação da segunda forma normal é mais relacionada a chaves primarias concatenadas, um conceito que não será abordado nesse guia, mas destacar os atributos que tenham dependências em relação a chave primária, enfim, existem vários conceitos, mas a forma normal um está bom para nós já, caso queira aprofundar nos estudos em um próximo ciclo de estudo, recomendo a leitura de mais algumas das formas de normalização.

Significado da sigla SQL

Significa "*Structured Query Language*", ou "Linguagem de Consulta Estruturada". É uma linguagem para a manipulação de uma Base de Dados Relacional.

Possui diversos comandos, que são denominados como instruções, as instruções retornam à consulta, ou seja, os dados da minha base de dados. Para realizar uma consulta, antes é preciso entender as instruções, as instruções estão divididas em quatro principais grupos:

1. DML (Data Manipulation Language).

- SELECT -> Pesquisa.
- UPDATE -> Atualização.
- DELETE -> Deletar.
- INSERT -> Inserir.

2. DDL (Data Definition Language).

- CREATE -> Criar.
- ALTER -> Alterar.
- DROP -> Deletar.

3. DCL (Data Control Language).

- GRANT -> Fornecer Privilégios.
- REVOKE -> Remover Privilégios. (Ex: Acesso)

4. DTL (Data Transaction Language).

- COMMIT -> Salvar.
- ROLLBACK -> Voltar ao ponto antes de inserir os registros.

Usamos os comandos DDL, geralmente em Objetos (Tabelas), tente não decorar esses termos, pois vamos trinar os mesmos passo a passo ao decorrer do documento, vamos abordar necessariamente a sintaxe da linguagem SQL, que nada mais é que a forma de escrever o que você quer para que o computador entenda.

Qual Ferramenta Usar?

Existem inúmeras ferramentas para manipular uma base de dados com uma interface gráfica.

A mais simples e leve é o SQLite, ser simples não quer dizer que é ruim ou muito básica, e sim, ser fácil de usar, extremamente leve, esse sistema é usado nos smartphones, não requer instalação, registro de root, etc. Baixou, funcionou.

Link para o Download do SQLite Browser: <https://sqlitebrowser.org/dl/>

Windows: Basta baixar a versão conforme o tipo do seu sistema.

Ubuntu: Abra o terminal, para abrir o terminal (Ctrl + Alt + T) e digite *sudo apt-get install sqlitebrowser*, mas antes, uma boa pratica sempre que abrir o terminal, no Ubuntu é atualizar o gerenciador de pacotes usando o comando *sudo apt-get update*. Caso necessário do uso do *Personal Package*, use o comando *sudo add-apt-repository -y ppa:linuxgndu/sqlitebrowser*.

Mac: Com o homebrew fica tudo mais fácil, caso precise instalar o mesmo, no próprio site oficial do brew tem a sua instalação, <https://docs.brew.sh/Installation> .

brew install --cask db-browser-for-sqlit.

SQL Fundamentos dos Dados

A Linguagem SQL é como outra linguagem, tem seus tipos de dados e formas específicas para tais manipulações, um exemplo para citar esses tipos é que não podemos colocar uma sequência de caracteres textuais em uma coluna que só aceita números por exemplo.

Você precisa especificar na criação de uma tabela, os tipos de dado que serão aceitos durante a manipulação do banco de dados, na linguagem SQL, existem quatro famílias principais desses tipos de dados.

Tipos de dados Literais e Caracteres

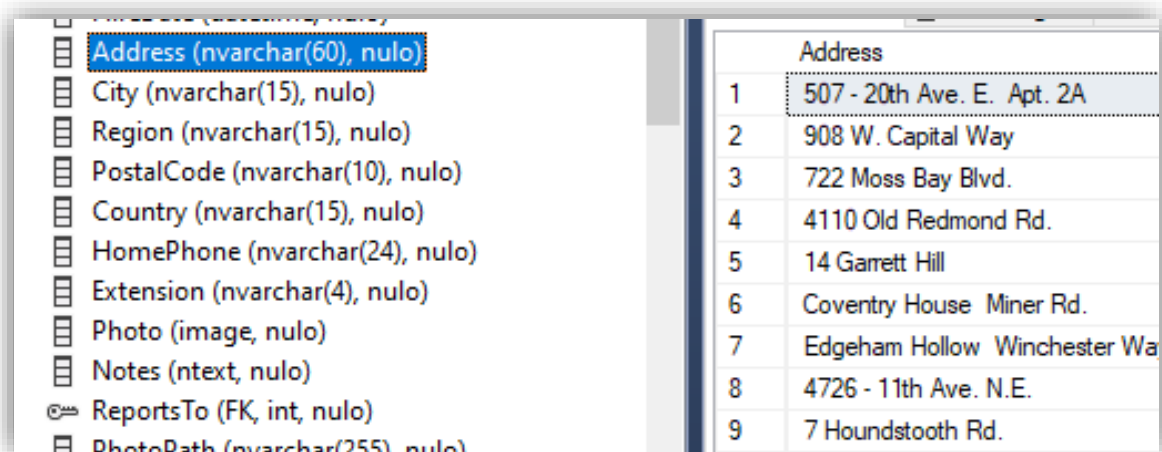
Para qualquer linguagem de programação, no contexto de programação de computadores, o conjunto de caracteres é denominado *String*, utilizada para representar palavras, frases e textos. Os tipos de dados em texto, sempre são representados com aspas simples ou duplas.

Nome: 'Zacarias',

Sobrenome: 'Junior'.

Mas eu posso expressar de outra forma utilizando as famosas *varchar*s.

Para o SQL, pode ter valores numerais e símbolos incluídos em uma sequência de caracteres que podem ser armazenados em variáveis, são um tipo diferente de dados, que juntam símbolos e valores decimais.



	Address
1	507 - 20th Ave. E. Apt. 2A
2	908 W. Capital Way
3	722 Moss Bay Blvd.
4	4110 Old Redmond Rd.
5	14 Garrett Hill
6	Coventry House Miner Rd.
7	Edgeham Hollow Winchester Wa
8	4726 - 11th Ave. N.E.
9	7 Houndstooth Rd.

Figura 4: Informações e alguns registros na coluna *Address*, uma das colunas em uma tabela de dados.

Repare na coluna *Address*, contam letras, símbolos e números, mais seu tipo de dado é *nvarchar*. Que é um tipo de variável no SQL para *string* e outros valores em conjuntos, o outro valor dentro dos parênteses, representa 'nulo', mas não quer dizer que os valores ali são nulos, veremos nos capítulos seguintes.

Instruções geralmente dentro de parênteses são denominamos **parâmetros**, provavelmente você já sabia disso quando estudou funções lá no ensino médio, caso não lembre, aqui vai um exemplo bem básico. $f(x) = x^2$.

Se eu mudar o x para dois, tenho a função $f(2) = 2^2$, e o dois está entre parênteses e onde tiver a letra x na minha função, altero para dois.

Mas não é só esses valores que existem.

Char	Variável que armazena letras, números e caracteres especiais, contém um valor fixo que é um parâmetro, caso não seja preenchido com a sequência de caracteres, o SQL preencherá com espaços em branco.
Varchar e Nvarchar	Mesmo conceito anterior, porém, é flexível em relação aos valores armazenados, ou seja, caso o parâmetro for (100), e os valores armazenados não alcançarem um total de 100 caracteres, a variável vai adequar-se ao tamanho dos valores inseridos.
TinyText e LongText	Tipo de dado somente texto, varia entre TinyText, Text, MediumText e LongText, a sua diferença é a quantidade de caracteres que você quer armazenar.
TinyBlob e LongBlob	Valores Binários como arquivos, fotos, etc. o tamanho também varia como os estilos textuais.
Set e Enum	Ambos são para uma lista de valores, contudo, o Enum é uma lista mais robusta e armazena mais valores.

Esses são todos os tipos de dados para valores caracteres e literais. Durante a criação de uma tabela no próximo capítulo, vai ser especificado esses tipos de dados que são muito importantes, principalmente as variáveis.

Tipos de dados Numéricos

Durante a criação das colunas e das tabelas, o tipo de dado numérico é bem importante, com esse tipo de dado, podemos armazenar dados de vendas, realizar contas matemáticas, entre outras coisas.

☞ SupplierID (FK, int, nulo)		UnitPrice	UnitsInStock	Discontinued
☞ CategoryID (FK, int, nulo)	1	18,00	39	0
☞ QuantityPerUnit (nvarchar(20), nulo)	2	19,00	17	0
☞ UnitPrice (money, nulo)	3	10,00	13	0
☞ UnitsInStock (smallint, nulo)	4	22,00	53	0
☞ UnitsOnOrder (smallint, nulo)	5	21,35	0	1
☞ ReorderLevel (smallint, nulo)	6	25,00	120	0
☞ Discontinued (bit, não nulo)	7	30,00	15	0

Figura 5: Tipos de dados das colunas *UnitPrice*, *UnitsInStock*, e *Discontinued*.

Nessa seleção de colunas, temos três tipos de valores numéricos veja mais alguns.

O parâmetro ‘Nulo’, não está relacionado ao valor numérico para ser nulo, é um outro conceito que será abordado ao decorrer desse capítulo, vamos primeiro entender os tipos numéricos usados na linguagem SQL.

Bit	Número inteiro de 0 ou 1 e nulo.
Int	É subdividido em TinyInt, SmallInt, Int e BigInt, o que muda é o espaço alocado para armazenar o conjunto numérico.
Decimal	Permite valores que serão convertidos, você especifica as escalas como parâmetros. Valores exatos com partes fracionárias.
Numeric	Mesmo conceito do decimal, contudo, não possui valor depois do ponto. (Precisão, Escala), Exemplo, (2, 5), 20,12753.
Money	Formato de dados para valores monetários.
Float	Valores com ponto flutuante, seguem um padrão de arquitetura (IEEE 754).
Real	Mesmo conceito do <i>float</i> , a variação é a precisão.

Tipos de dados Temporais

Um tipo de dado especialmente desenvolvido para armazenar valores do tipo de datas e tempos, que podem variar de diversas formas.

CustomerID (FK, nchar(5), nulo)		OrderDate	RequiredDate	ShippedDate
EmployeeID (FK, int, nulo)	1	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000
OrderDate (datetime, nulo)	2	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000	1996-07-10 00:00:00.000
RequiredDate (datetime, nulo)	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-12 00:00:00.000
ShippedDate (datetime, nulo)	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-15 00:00:00.000
	5	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-11 00:00:00.000

Figura 6: Tipos de dados das colunas *OrderDate*, *RequiredDate* e *ShippedDate*.

Reparem que mesmo em uma base de dados estrangeira, o formato de datas é o mesmo, sendo primeiro ano, depois mês e dia, com os horários separados por dois pontos.

Time	Armazena somente os valores de Tempo.
Datetime	Armazena o formato da data em (AAAA-MM-DD HH:MM:SS) Ano-Mês-Dia Hora:Minuto:Segundo
Datetime2	Armazena o formato da data em (AAAA-MM-DD HH:MM:SS:MS) Ano-Mês-Dia Hora:Minuto:Segundo:Milisegundo
SmallDatetime	Armazena o formato de Datetime, a partir de um limite de datas estabelecidos pelo próprio SQL.
Date	Armazena o formato da data em (AAAA-MM-DD) sem os horários.
Datetimeoffset	Mesma coisa que o Datetime2, porem com adição do Fuso horário

Tipos de dados Booleanos

Esse é o tipo de dado mais fácil, é como o BIT, ele armazena apenas dois valores, 1, 0 ou nulo, quando o valor é nulo, quer dizer que não tem nenhum registro nessa determinada coluna, muito comum durante o processo de junções de tabelas. Caso o parâmetro seja 'Não Nulo' ou 'Nulo', quer dizer que o formato da coluna aceita valores nulos de inputs ou inserção de dados, ou não aceita registros nulos.

O booleano, sendo seu valor zero é considerado como falso e o valor um é considerado verdadeiro. Exemplo, 'Na minha base de dados, os valores da coluna *casado*, está com valores um ou zero, ou seja, sim, está casado ou, não está casado'.

Palavras Reservadas

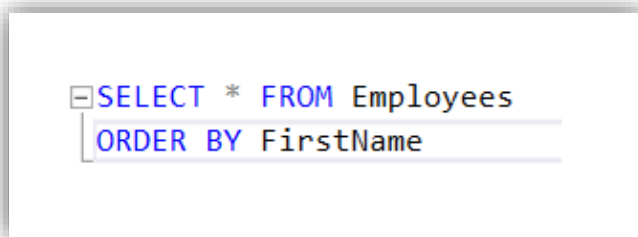
Em toda a linguagem SQL ou até mesmo em outras linguagens de programação como Java Script, Python, possuem suas palavras reservadas, ou seja, palavras que usamos para realizar nossas instruções, no exemplo do Java Script, não podemos utilizar a palavra 'Let', pois é

uma palavra reservada da linguagem para indicar que nas próximas sequencias da linha, tem uma variável.

Em SQL, existem três formas de diferenciar as palavras reservadas, são elas, a coloração que o SGBD nos indica quando digitamos os comandos no software, ele não é *case sensitive*, mas é possível escrever nossas instruções em letra maiúscula e minúsculas e todas as palavras são, a maioria auto explicativas e diferenciáveis pelo SGBD.

CREATE	DATABASE	SELECT	FROM	WHERE	DROP	DELETE	UPDATE
Criar	Base de Dados	Selecionar	A partir de	Onde	Largar / Jogar	Deletar	Atualizar

Esses são alguns exemplos de palavras reservadas da linguagem SQL (As que estão em Negrito), existem muitas outras palavras reservadas que eventualmente vão aparecer durante as consultas.



```
SELECT * FROM Employees  
ORDER BY FirstName
```

Figure 7: Exemplo de uma Instrução simples.

Conceito de Chaves

Durante a criação das nossas tabelas, todas as tabelas devem possuir uma chave, denominada Id, geralmente essa chave tem o mesmo nome da Tabela, juntamente com a palavra Id no final e seus devidos tipos de dados especiais.

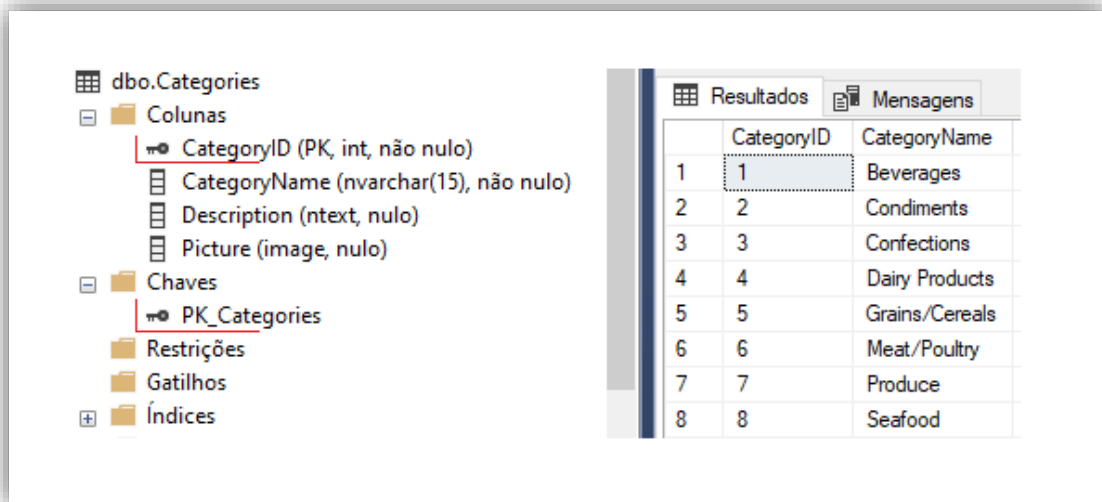


Figura 8: Chaves primárias e chaves estrangeiras.

Reparem que a chave até recebe um apelido de PK, essa chave é o indicador dessa tabela para as outras tabelas.

Quando uma chave primária está pertencendo a uma linha de outra tabela, ou seja, existe a tabela X que possui sua chave primária, porem ela aparece na tabela Y, a tabela Y usa e referência a tabela X, quando a PK de uma tabela aparece em outra tabela, ela é chamada de *Foreign Key*, ou chave estrangeira.

Observação, a PK, não pode ter um valor nulo, os valores da coluna do Id, sempre são únicos, pois são os Index, os valores tem que ser incrementais e sempre começam com o índice 1.

Vai ser abordado a criação de base de dados e tabelas com os identificadores e chaves estrangeiras ao decorrer do documento.

Operadores Relacionais e Matemáticos

A linguagem SQL suporta alguns símbolos matemáticos, mas os principais são as condicionais, que usamos em consultas.

Igualdade =	Igualdade, essa condicional vai verificar se o valor X é igual ao valor Y ($X = Y$), nesses casos, retorna uma condição booleana (Sim ou Não). ('A' = 'A'), o resultado vai ser positivo.
Diferente <>	Diferente, essa condicional vai verificar se o valor X é diferente do valor Y ($X <> Y$), nesses casos, retorna uma condição booleana (Sim ou Não). (3 <> 1), o resultado vai ser positivo.
Maior Que >	Maior que, essa condicional vai verificar se o valor X é maior do que valor Y ($X > Y$), nesses casos, retorna uma condição booleana (Sim ou Não). (7 > 6), o resultado vai ser positivo.

Menor Que <	Menor que, essa condicional vai verificar se o valor X é menor do que valor Y ($X < Y$), nesses casos, retorna uma condição booleana (Sim ou Não). ($10 < 100$), o resultado vai ser positivo.
Maior e Igual >=	Esse conceito se aplica também a o menor ou igual , ele verifica se o valor X é maior ou idêntico ao valor Y. ($10 \leq 50$) Positivo, ($20 \geq 20$) Positivo.
Matemáticos	Adição (+), Subtração (-), Divisão (/), Multiplicação (*). Veremos mais adiante as próprias funções do SQL para aplicar outras funções matemáticas.

Operadores Lógicos

Diferente dos operadores relacionais, os operadores lógicos comparam resultados booleanos, (*True* ou *False*) nos operadores relacionais, seu resultado é um valor booleano, com esse valor podemos utilizar operadores lógicos para que seja ainda mais cauteloso na nossa consulta.

Os operadores lógicos são divididos em três.

Expressões1	Expressões2	AND / E	OR / OU	
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	
Falso	Falso	Falso	Falso	NOT / Não
Falso	Verdadeiro	Falso	Verdadeiro	Verdadeiro -> Falso
Verdadeiro	Falso	Falso	Verdadeiro	Falso -> Verdadeiro

AND / E, retorna verdadeiro se os dois valores retornam verdadeiros, caso um deles não seja verdadeiro, ele não retorna verdadeiro.

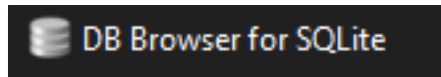
OR / OU, retorna verdadeiro se pelo menos um dos valores forem verdadeiros, caso ambos sejam falsos, ele retorna falso.

NOT / NÃO, inverte o valor, NÃO Verdadeiro, vai ser falso.

Para o SQL, a função AND, retornara todos os resultados em que seus registros sejam positivos.

SQLite Funcionalidades

Caso estiver usando o Windows, basta localizar o diretório de download do seu computador e abrir o executável DB Browser for SQLite. Para rodar no Ubuntu / Mac, basta usar o mesmo diretório ou o comando *sqlitebrowser* no terminal.



Interface do SQLite

A interface do SQLite é bem simples. Antes de começar o desenvolvimento, vamos conhecer um pouquinho mais sobre essa ferramenta.

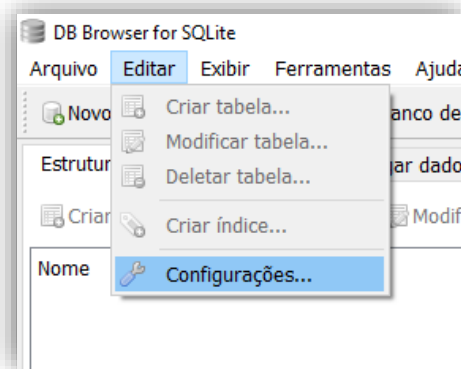


Figura 9: Guia de configurações no SQL Lite.

Editar, Configurações, para checar os primeiros requisitos, se estiver em outra linguagem, tem a tradução para português.

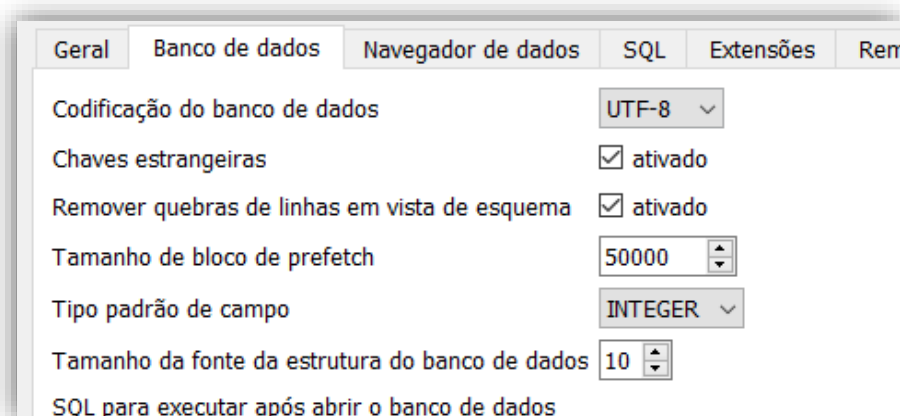


Figura 10: Configurações de SQL Lite.

Na guia Banco de Dados, certifique-se que a configuração está UTF-8, assim, você pode acrescentar acentos em seus registros.

Primeira Base de Dados

Para criar um novo banco de dados, basta clicar no botão ‘**Novo Banco de Dados**’ no canto superior esquerdo, irá subir um *popup* para você escolher onde quer salvar seu banco de Dados, recomendo salvar na Área de Trabalho ou no local padrão do SQLite, com algum nome.

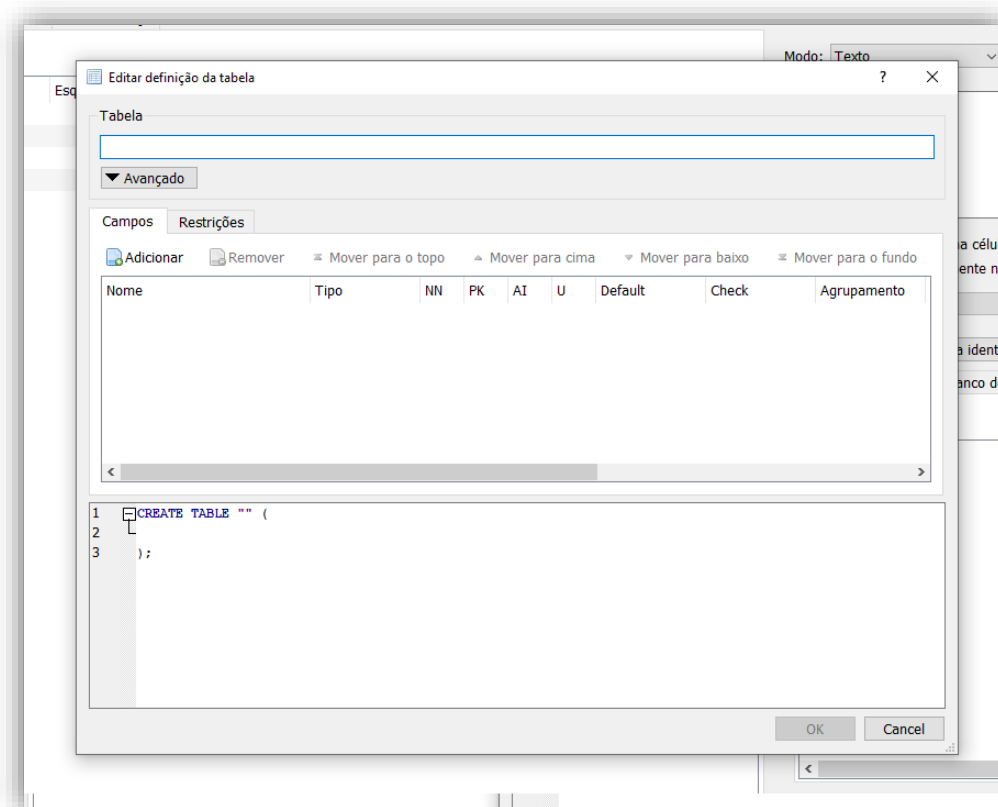


Figura 11: Criando uma tabela no SQL Lite.

Após selecionar a área onde deseja salvar seu banco de dados, novamente outro *popup* aparecerá, nesse *popup*, são as configurações básicas de sua nova tabela, pois a base de dados você já criou, sim o sqlite você cria a base automaticamente, só precisa criar as tabelas.

Informações da nossa tabela:

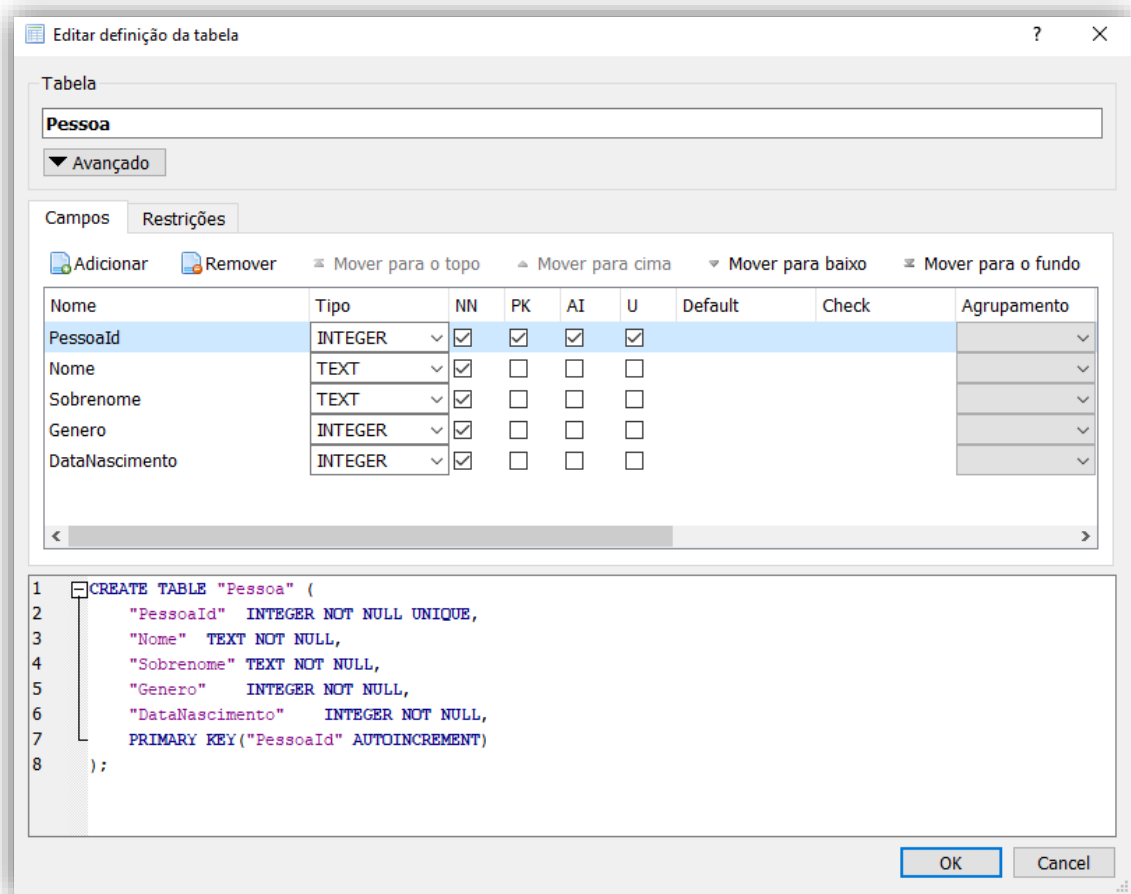


Figura 12: Criando colunas na tabela Pessoa.

No SQLite você consegue visualmente adicionar as colunas para começar criando sua tabela, basta clicar em Adicionar, repare que sempre começa com o **NomeDaTabelaId**, que no caso foi a tabela Pessoa, sendo a primeira coluna, a PK.

Reparem que a *checkbox* **NN** significa, **NOT NULL**, ou seja, não pode possuir valor nulo, como foi citado anteriormente, a *checkbox* com a opção **PK**, é justamente a opção para identificar que aquela coluna da tabela é a chave primária, e opção **U**, é para que todos os valores sejam únicos ‘Unique’, repare que na instrução abaixo, tudo que foi checado, aparece ali também, e tudo que foi citado no primeiro tópico, fundamentos de SQL, aparece também.

Futuramente iremos entrar em mais detalhes, mas a lógica inicial é:

*Coluna Nome é do tipo **Text**, Not Null, pois não pode pessoas sem nome na nossa tabela.*

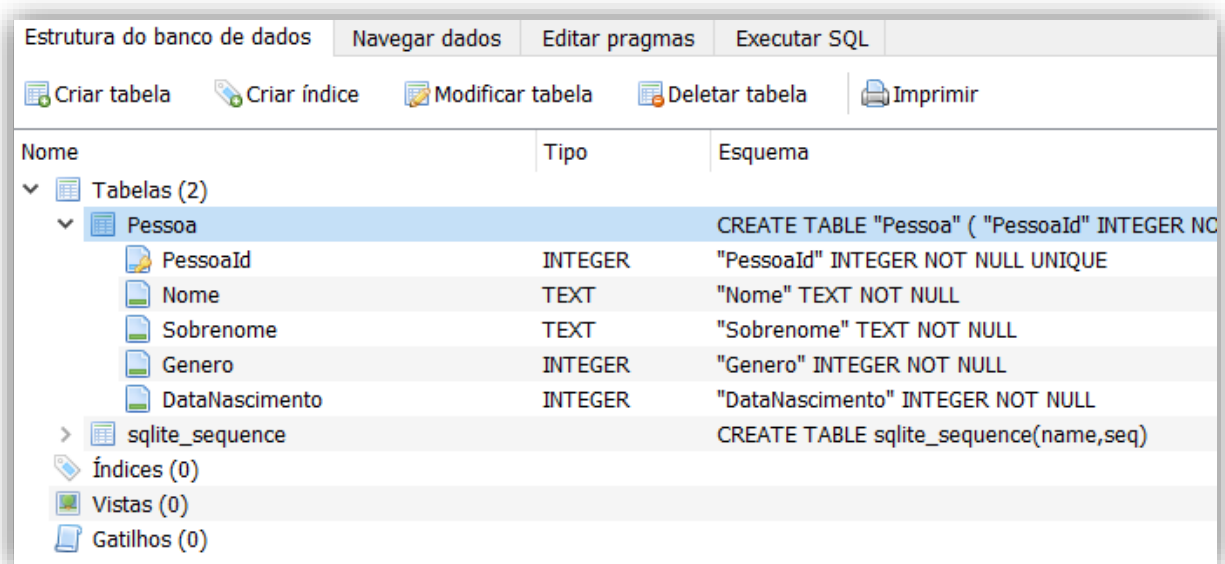
Coluna Sobrenome, segue o mesmo estilo da coluna nome.

*Coluna Genero, podia ser **Text** também, mas estou usando **INT**, sendo 1 masculino e 0 feminino, e sem acento.*

*Coluna **DataNascimento**, como não tem a opção **date no sqlite**, optei em deixar **Text**, esse estilo de escrita é chamado de **CamelCase**, ou seja, as primeiras letras maiúsculas e sem espaço, existe também a notação **snake_case**, onde é separado por **_** tudo minúsculo.*

Entenda que, você escolhe como vai desenvolver, pois você é o analista, para isso deve atender aos requisitos do escopo que foi proposto pela unidade de negócio que lhe passou esse problema, você passou por todas as etapas da modelagem antes mesmo de começar a desenvolver a tabela. Só uma observação para não fazer exatamente igual ao meu exemplo, você tem total liberdade para desenvolver da forma que melhor atende a necessidade do negócio ou para fins acadêmicos.

Agora com a primeira tabela criada, ela não possui valor nenhum, pode-se navegar para visualizar os registros clicando em 'Navegar Dados', mas não registramos nada ainda.

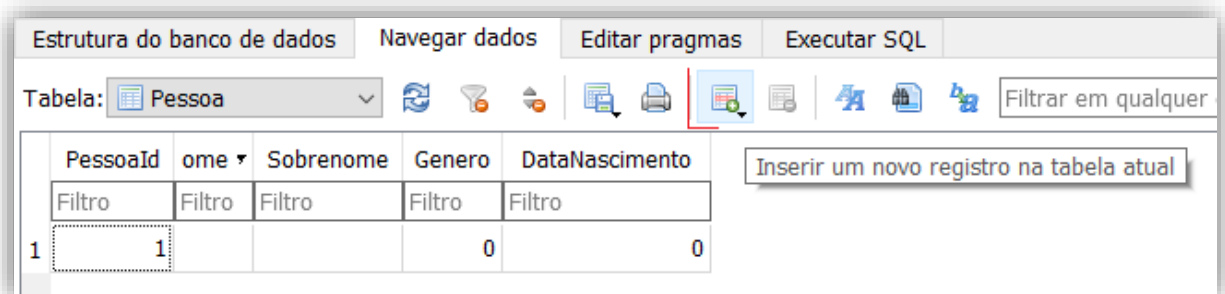


The screenshot shows the 'Estrutura do banco de dados' window with the 'Pessoa' table selected. The table structure is as follows:

Nome	Tipo	Esquema
CREATE TABLE "Pessoa" ("PessoaId" INTEGER NOT NULL UNIQUE		
PessoaId	INTEGER	"PessoaId" INTEGER NOT NULL UNIQUE
Nome	TEXT	"Nome" TEXT NOT NULL
Sobrenome	TEXT	"Sobrenome" TEXT NOT NULL
Genero	INTEGER	"Genero" INTEGER NOT NULL
DataNascimento	INTEGER	"DataNascimento" INTEGER NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

Figura 13: Colunas da tabela Pessoa.

Para popular a base de dados, no SQLite, basta clicar na aba denominada de Navegar Dados.



The screenshot shows the 'Navegar dados' window with the 'Pessoa' table selected. The table view shows one record with the following data:

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1			0	0

Figura 11: Adicionando registros na tabela.

Reparem que, o Index e a PK já foram automaticamente preenchidos, devido as constraints que foram aplicadas durante a criação da tabela. Para preencher os valores, basta clicar na célula onde você quer preencher os registros.

Alguns registros não nulos, o proprio SQLite vai preencher essas colunas.

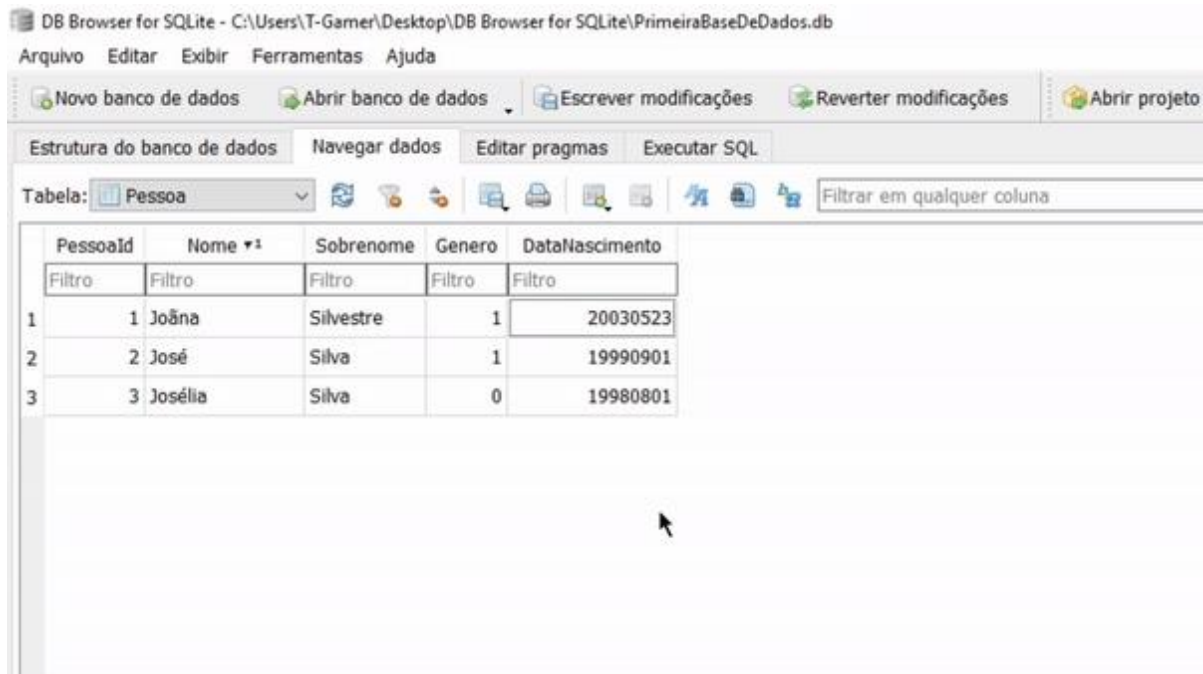


Figura 14: Registros manuais.

Podemos criar automaticamente as linhas na base de dados, basta clicar no ícone ‘**Inserir um novo registro na tabela atual**’ e você pode clicar nas linhas para adicionar as informações e digitar manualmente, após isso, basta clicar em aplicar.

Existem formas de inserir dados com próprias instruções sql, mas como o sqlite é intuitivo, recomendo começar inserindo os dados visualmente.

Consultas Básicas

Agora que foi criado a tabela e já possuem dados, está na hora de realizar as consultas com instruções, na lista de abas superior, tem a aba “**Executar SQL**”, na primeira caixa, é onde são inseridos as instruções.

Neste primeiro tópico, será abordado questões simples como sintaxe e os primeiros comandos.

Sintax e Select

Lembra que no primeiro capítulo, foi comentado conceitos sobre os tipos de instruções, pois bem, vamos utilizar a mais básica de todas, uma instrução DML para selecionar todas as colunas e registros da minha tabela pessoa.

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	1	20030523
2	2	José	Silva	0	19990901
3	3	Josélia	Silva	0	19980801
4	4	Juca	Silvestre	1	17540812

Figura 15: Adicionando registros na tabela.

Para aumentar a letra das instruções, basta clicar no CTRL e usar o Scroll do Mouse. Em notebooks, basta apertar Fn +

- Apertando TAB, o SQLite autocompleta.
- Para a instrução funcionar, basta apertar F5.
- Para comentar as instruções no SQL, ou seja, as instruções comentadas não serão executadas, basta usar /* Comentário */ ou -- Comentário.

SELECIONE * (Todos os Dados) Da minha tabela 'Pessoa'.

Esse é a instrução SQL mais fácil, porem é uma das mais perigosas, imagine-se em um cenário que você esta manipulando uma base de dados imensa, caso usar um comando desse sem especificação, poderá travar a consulta ou até mesmo acabar com a memória do computador.

1	SELECT * FROM Pessoa LIMIT 2;				
	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	1	20030523
2	2	José	Silva	1	19990901
Execução finalizada sem erros. Resultado: 2 linhas retornadas em 4 ms Na linha 1: SELECT * FROM Pessoa LIMIT 2;					

Figura 16: Instrução Limit.

Para conseguir reverter esse problema de selecionar muitos dados de uma vez, existe uma instrução que pode limitar essa consulta para retornar determinados registros, esse valor é arbitrário, na imagem 16 foi selecionado apenas 2 registros.

Dois pontos, é bem simples não é, apenas acrescentar a palavra LIMIT e a quantidade de registros que eu quero visualizar.

O ponto e virgula ‘ ; ’ no final, é uma sintaxe do próprio SQL, e é opcional.

Vamos supor que na minha tabela, tenha vários valores iguais, para eu separar esses valores, e buscar apenas o valor único, uso a instrução para distinguir esses registros, parecido com o comando limite, essa instrução de distinguir funciona da mesma forma, a palavra em inglês para distinto é ‘*Distinct*’, essa é a instrução no SQL.

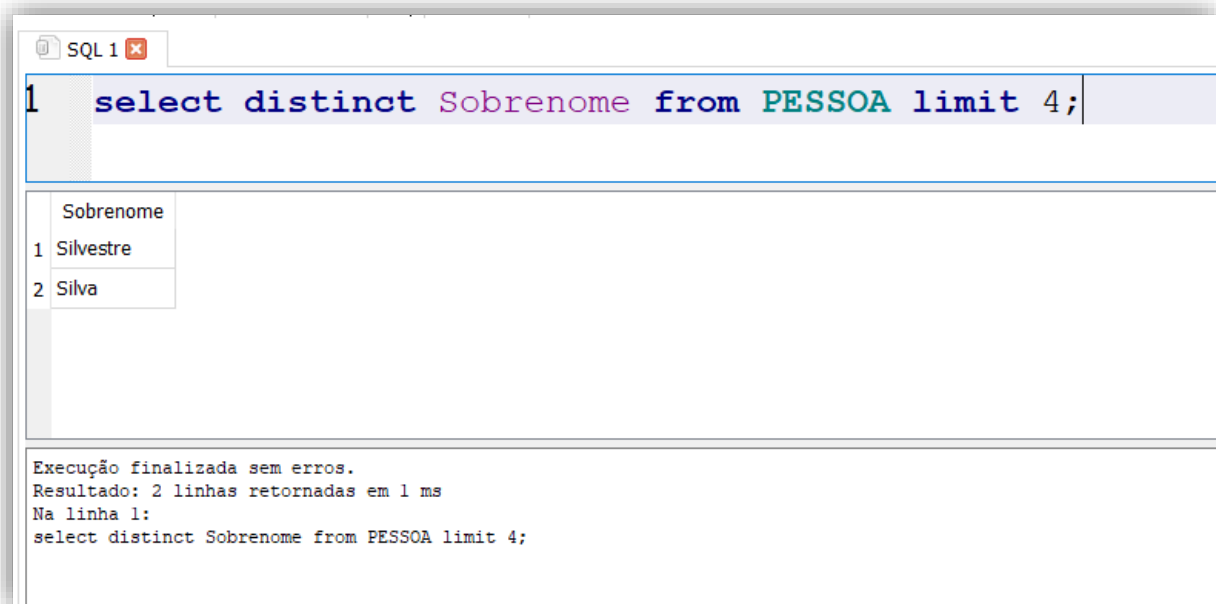


Figura 17: Instrução *Select* em minúsculo.

SQL não é case sensitive, ou seja, letras maiúsculas e minúsculas funcionam igual. Na primeira instrução, estou selecionando todos os sobrenomes distintos da minha tabela pessoa, onde distintos é únicos, com um limite de 4 registros, porém, tenho somente dois registros únicos de sobrenome na minha tabela que serão retornados.

Consultas com Where

A instrução **WHERE**, a tradução básica, seria '**Onde**', podemos fazer uma ligação com linguagens de programação, seria parecido com o *if*, que retorna determinados valores, *se* seu resultado for positivo.

O *Where* funciona da mesma forma, passamos uma **condição**, ele vai percorrer em todos os registros, quando encontrar valores que correspondem a essa condição, vai retornar o determinado registro.

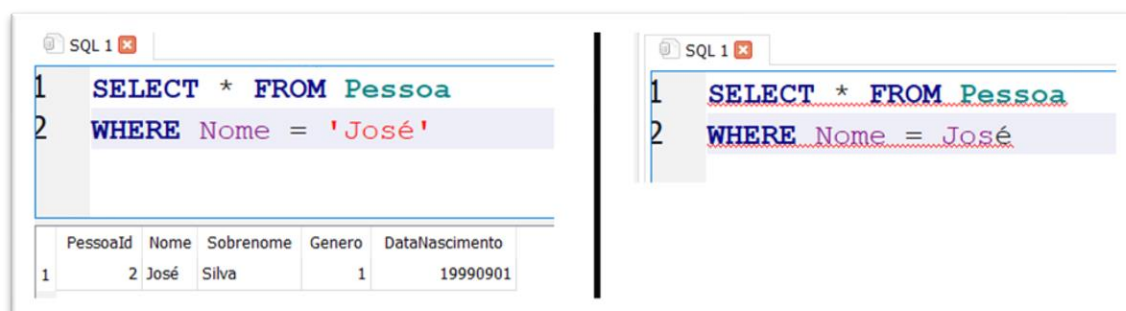


Figura 18: Instrução *Select* em maiúsculo com a instrução *Where*.

SELECIONE todos os registros DA tabela Pessoa

ONDE Nome seja igual a 'José'.

Na segunda imagem, a instrução esta errada, o interpretador SQL até ficou com um erro, pois o nome José, não esta sendo representado com aspas, ainda especifiquemos durante a criação dessa tabela, que apenas aceitaríamos registros do tipo string, e como foi visto no primeiro capítulo os tipos de dados, representa-se *String* com aspas.

Pode tambem ser usado os operadores lógicos durando as intruções WHERE.

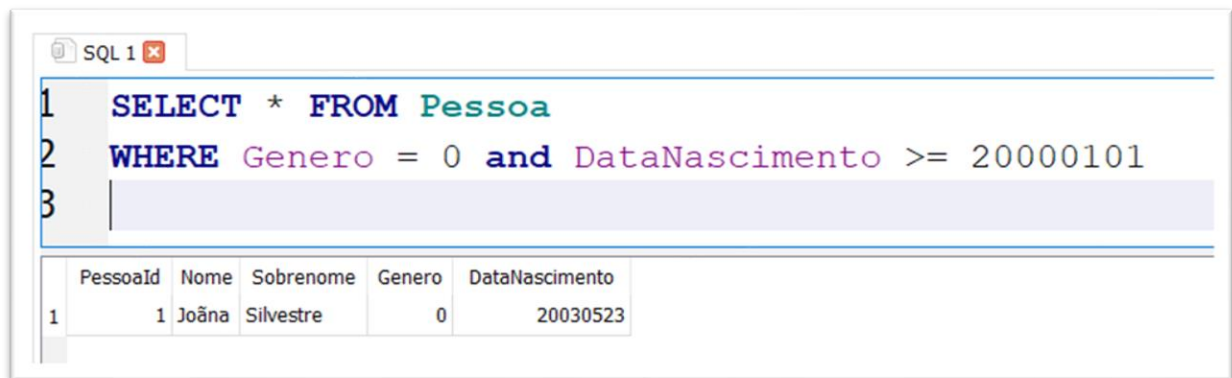


Figura 19: Instrução *Select* em maiúsculo com a instrução *Where* e *And*.

Selecione todos os registros da tabela pessoa.

Onde o genero é igual a zero **E** a DataNascimento é maior ou igual a 20000101.

Caso no lugar do operador **E**, estivesse o operador **OR**, minha consulta seria retornada com todos os registros que tenham o genero igual a um tambem.

Dica: Para salvar a base de dados, basta clicar em “**Escrever Modificações**”.

Palavra Reservada Like

Mas, caso eu não saiba exatamente o registro de uma tabela, então e usado a instrução **LIKE**, essa instrução, é geralmente utilizada em textos, quando não sabemos exatamente oque tinha escrito no texto, apenas uma parte, passamos junto uma mascara, o ‘%’, que é denominado padrão, existe muitos outros padroes, mas vou citar aqui apenas esse.

Esse padrão com as ‘%’, podem ser antes das aspas do texto ou depois, ou antes e depois, nesses casos, usar a porcentagem antes do texto com o Like, afirma que, estou procurando palavras que começam com algum valor, porem terminam com oque eu digitei, mesma coisa para os outros casos.

The screenshot shows a SQL editor window titled 'SQL 1'. The query entered is:

```
1 SELECT * FROM Pessoa
2 WHERE Nome LIKE '%ãn%'
```

Below the query, the result is displayed in a table with the following columns: PessoaId, Nome, Sobrenome, Genero, and DataNascimento.

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	0	20030523

Figura 20: Instrução *Select* em maiúsculo com a instrução *Where* e *Like*.

Selecione todos os registros da tabela Pessoa, onde seu nome é **parecido / como** ‘algum valor antes de **ãn** e algum valor depois de **ãn**’.

Palavra Reservada BETWEEN

Podemos localizar registros passando uma lista de registros, que no caso é a instrução **BETWEEN**, que significa ‘Entre’.

The screenshot shows a SQL editor window titled 'SQL 1'. The query entered is:

```
1 SeLeCt * FROM Pessoa
2 where PessoaId BeTWEn 1 and 4
```

Below the query, the result is displayed in a table with the following columns: PessoaId, Nome, Sobrenome, Genero, and DataNascimento.

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	0	20030523
2	2	José	Silva	1	19990901
3	3	Josélia	Silva	0	19980801
4	4	Juca	Silvestre	1	17540812

Figura 21: Instrução *Select* em maiúsculo e minúsculo com a instrução *Where*, *Between* e *And*.

Reparem que, SQL, funciona com letras maiúsculas e minúsculas juntos, mas não é muito agradável de ler essas instruções dessa forma, apenas coloquei para você visualizar.

Selecione todos os registros da tabela pessoa, onde o id da pessoa esteja **EnTrE**, um e quatro.

Palavra Reservada IN

Com a instrução **IN**, ‘Em’, passamos realmente uma lista de valores para serem selecionados dentro de parenteses.



```
1 SELECT * FROM Pessoa
2 WHERE PessoaId IN (1, 3, 4)
```

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	0	20030523
2	3	Josélia	Silva	0	19980801
3	4	Juca	Silvestre	1	17540812

Figura 22: Instrução *Select* em maiúsculo com a instrução *Where* e *In*.

Selecione todas as pessoas da tabela pessoas, onde o id da pessoa esteja **em** (1, 3, 4).

Trabalhando com Ordenação

Pode também ordenar os resultados das consultas com o comando **ORDER BY**, ‘ordenar por’, como tinha comentado antes, a sintaxe dessa linguagem é bem fácil de entender e auto explicativas.

Essa instrução para ordenar, devemos passar a coluna na qual vai ser ordenada e o sentido, por padrão é ASC, crescente, ou seja, começando do menor para o maior, mas podemos passar DESC, para decrescente.

Repare que a tabela agora possui alguns valores a mais, no próximo tópico, vamos adicionar mais registros a nossa tabela.

1	SELECT	*	FROM	Pessoa
2	WHERE	Genero	=	1
3	ORDER BY	Nome	DESC	

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	7	Marcos	Santos	1	20000101
2	5	Luiz	Lima	1	19850323
3	4	Juca	Silvestre	1	17540812
4	2	José	Silva	1	19990901

Figura 23: Instrução *Select* em maiúsculo com a instrução *Where*, *In Order By* e *Desc*.

A instrução de ordenação também funciona com o índice da coluna, ou seja, a PessoaId é o índice um, o Nome, essa coluna é o índice dois, e assim por diante, faça o teste alterando a palavra Nome, pelo valor dois, caso a coluna sua base de dados, cuja posição dois é a coluna com os registros de Nome.

Instruções com Tabelas

Palavra Reservada INSERT INTO

Para adicionar mais valores a tabela, usa-se a instrução “**Insert Into**”, ‘insira em’ seguido da tabela, as colunas entre parenteses e a palavra “**Values**” com os registros entre parenteses também seguindo a ordem dos atributos que você escolheu quando selecionou a Tabela.

1	INSERT INTO	Pessoa	(PessoaId, Nome, Sobrenome, Genero, DataNascimento)
2	VALUES		
3	(5, 'Luiz', 'Lima', 1, 19850323),		
4	(6, 'Luiza', 'Luz', 0, 19980817),		
5	(7, 'Marcos', 'Santos', 1, 20000101),		
6	(8, 'Maria', 'Mariana', 0, 19991212);		
7			

Execução finalizada sem erros.			
Resultado: consulta executada com sucesso. Levou 1ms, 4 linhas afetadas			
Na linha 1:			
INSERT INTO Pessoa (PessoaId, Nome, Sobrenome, Genero, DataNascimento)			
VALUES			
(5, 'Luiz', 'Lima', 1, 19850323),			
(6, 'Luiza', 'Luz', 0, 19980817),			
(7, 'Marcos', 'Santos', 1, 20000101),			
(8, 'Maria', 'Mariana', 0, 19991212);			

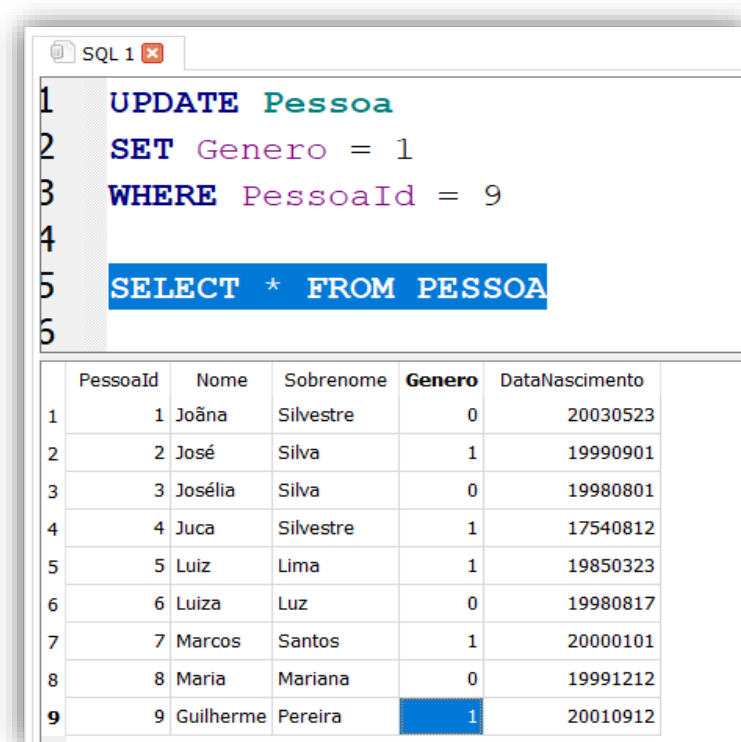
Figura 24: Instrução *Select* em maiúsculo com a instrução *Where*, *In Order By* e *Desc*.

Insira na tabela Pessoas (PessoaId, Nome, Sobrenome, Genero, DataNascimento), Passamos entre parenteses, as colunas da nossa tabela.

Valores (Id, Nome, Sobrenome, Genero, DataNascimento), Passamos entre parenteses, os valores correspondentes aos valores da tabela, separados por virgula e o ultimo valor com ponto e virgula.

Palavra Reservada UPDATE e SET

Caso algum valor da tabela esteja errado, é possível atualizar somente aquele valor com instruções próprias da linguagem SQL. para isso, usamos a instrução **UPDATE**, adicionei mais um valor a minha tabela com o genero feminino, porem seu nome é 'Guilherme', claramente, esse é um erro de input de dados, que o proprio usuário pode ter selecionado errado ou o banco de dados armazenou errado, ou ele é do gênero feminino, mas descartemos essa última opção para fins acadêmicos.



The screenshot shows a SQL IDE window titled 'SQL 1'. The SQL editor contains the following code:

```
1 UPDATE Pessoa
2 SET Genero = 1
3 WHERE PessoaId = 9
4
5 SELECT * FROM PESSOA
6
```

Below the editor, a table view displays the data from the 'PESSOA' table. The table has columns: PessoaId, Nome, Sobrenome, Genero, and DataNascimento. The row for PessoaId 9 is highlighted in blue, showing the update of the 'Genero' column to 1.

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	0	20030523
2	2	José	Silva	1	19990901
3	3	Josélia	Silva	0	19980801
4	4	Juca	Silvestre	1	17540812
5	5	Luiz	Lima	1	19850323
6	6	Luiza	Luz	0	19980817
7	7	Marcos	Santos	1	20000101
8	8	Maria	Mariana	0	19991212
9	9	Guilherme	Pereira	1	20010912

Figura 25: Instruções Seleccionadas.

Reparem que para executar apenas determinadas instruções em SQL, basta selecionar a instrução e apertar F5.

Usamos duas palavras reservadas, a palavra *UPDATE* e a palavra *SET*. para atualizar a tabela pessoa, para o gênero igual a um, onde o id da pessoa é igual a 9, podia utilizar outro valor para representar no lugar do Id, mas o Id é confiável e único, podia estar manipulando outro

valor, caso utilizasse o nome como referência, e o nome Guilherme estivesse sido registrado mais de uma vez, as instruções seriam aplicadas a ambos os registros.

Caso não especifique o **WHERE**, a instrução de update será aplicada a todos os registros da minha tabela, por isso essa instrução está até em negrito quando foi citada pela primeira vez.

Palavra Reservada DELETE

A instrução delete, como o nome já sugere, serve para deletar os registros da minha tabela, parecido com a função *update*, requer uma condição com o **WHERE** para não deletar todos os registros da minha base de dados.

A instrução é bem simples, vou deletar o valor que tínhamos atualizado com a função *update*, o 'Guilherme'.

- O SQLite já faz o *commit*, ou seja, não precisamos especificar o *commit* da consulta, o *commit* nada mais é que uma instrução que envia nossas alterações.



The screenshot shows a SQLite SQL editor window titled 'SQL 1'. The SQL code entered is:

```
1 DELETE FROM Pessoa
2 WHERE PessoaId = 9
3
4 SELECT * FROM Pessoa
```

Below the SQL editor, the result of the SELECT query is displayed as a table with 8 rows and 5 columns: PessoaId, Nome, Sobrenome, Genero, and DataNascimento.

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	0	20030523
2	2	José	Silva	1	19990901
3	3	Josélia	Silva	0	19980801
4	4	Juca	Silvestre	1	17540812
5	5	Luiz	Lima	1	19850323
6	6	Luiza	Luz	0	19980817
7	7	Marcos	Santos	1	20000101
8	8	Maria	Mariana	0	19991212

Figura 26: Realizando todas as consultas.

Palavra Reservada DROP

Ultimo Conceito básico de SQL, a instrução *DROP TABLE*, como todos as instruções de SQL, servem para deletar os registros, diferente da *DELETE*, essa aqui deleta a tabela inteira.

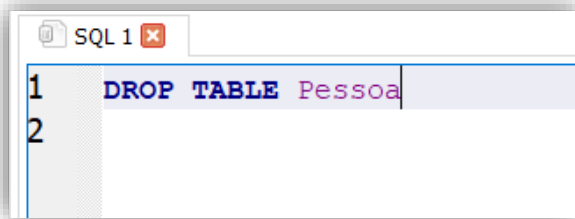


Figura 27: Sim, esse é o comando, após executar, repare que a Tabela pessoa até mudou de cor, pois ela não existe mais em sua base de dados.

Relacionando Tabelas

Antes de prosseguir para o SQL Server, tem mais um conceito essencial quando estiver trabalhando com tabelas, é o conceito de relacionamento de tabelas, onde já foi citado uma introdução sobre as chaves primárias, conceitos de normalização, etc.

Vai ser utilizado as mesmas tabelas do diagrama da Figura 3, que é o diagrama de relacionamento de três tabelas, a tabela Mercado, Venda e Produto que foi desenvolvida no final da modelagem de dados para ser desenvolvida com a linguagem SQL e o SQLite.

Com o seu SQLite aberto, prossiga clicando em “**Abrir banco de dados**”, ou “**Novo banco de dados**” para desenvolver essa prática, depois clicar em Executar SQL para escrever a tabela, segue as seguintes instruções para criar as tabelas sem ser manualmente.

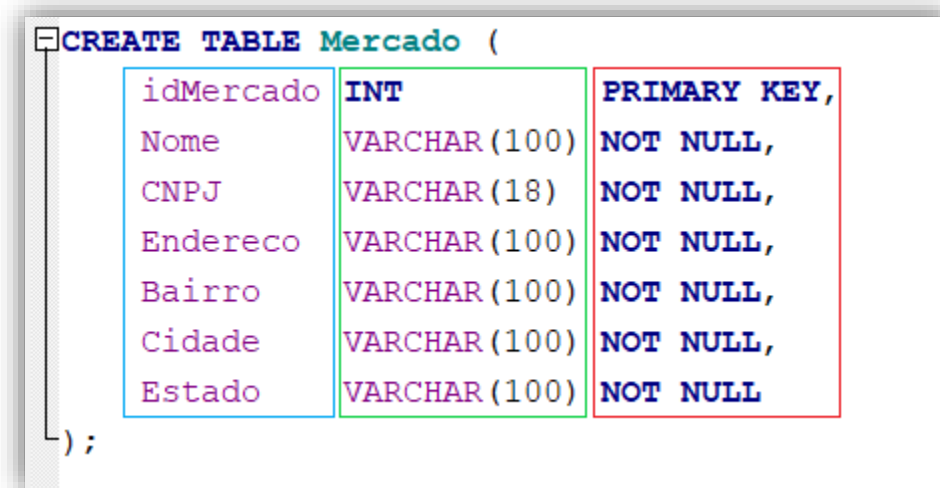


Figura 28: Criação de uma tabela com SQL.

Usa-se a palavra reservada *CREATE TABLE* e o nome da tabela, seguido de parênteses com as devidas colunas, como o próprio SQL escreve durante a criação manual das tabelas, os dados escritos dentro do retângulo representativo azul marinho corresponde à os nomes das colunas da tabela, seguido pelo tipo de dado que essa coluna vai aceitar representado pelas informações representadas dentro do retângulo verde e por fim, as *constraints*, que nada mais são que as restrições da coluna especifica na tabela representada também pelo retângulo em vermelho, após especificar o nome da coluna, seu respectivo tipo de dado e as restrições, usa-

se o separador virgula para a próxima coluna exatamente como esta na imagem, os espaços extra são representativos e para fins didáticos.

Seguido das outras tabelas como na imagem da figura 3.

```
CREATE TABLE Produto (
    idProduto INT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Tipo VARCHAR(100) NOT NULL,
    Preco INT NOT NULL,
    Fornecedor VARCHAR(100) NOT NULL,
    Estoque VARCHAR(45),
    idMercado INT NOT NULL,
    FOREIGN KEY (idMercado) REFERENCES Mercado(idMercado)
);

CREATE TABLE Venda (
    idVenda INT PRIMARY KEY,
    DataVenda VARCHAR(50) NOT NULL,
    Preco INT NOT NULL,
    idMercado INT NOT NULL,
    idProduto INT NOT NULL,
    FOREIGN KEY (idMercado) REFERENCES Mercado(idMercado),
    FOREIGN KEY (idProduto) REFERENCES Produto(idProduto)
);
```

Figura 29: Criação de duas tabelas com chave estrangeira tabela com SQL.

Criação de mais duas tabelas seguindo a mesma lógica anterior da primeira tabela e também seguindo o diagrama físico desenvolvido na modelagem de dados pelo analista, onde existem mais algumas palavras reservadas sendo elas a *FOREIGN KEY*, e *REFERENCES*.

A chave estrangeira nada mais é que a chave primária de outra tabela onde relaciono as tabelas pelas chaves estrangeiras, mas antes de referenciar a tabela já tem que estar criada a outra tabela, não seria possível referenciar a tabela Mercado na tabela Produto sem antes ter criado a tabela Mercado.

Mas para referenciar a tabela em outra tabela é preciso seguir algumas instruções, por exemplo na figura 29 está representado dentro de um retângulo vermelho a coluna chamada idMercado com o formato inteiro e não nulo, mesmo nome e tipo de dados da chave primária da tabela Mercado da figura 28, separado com virgula inicia-se a referência pela instrução “FOREIGN KEY (Coluna da tabela atual) REFERENCES Tabela a ser referenciada(Coluna referenciada)”, ou seja, chave estrangeira idMercado referencia a tabela mercado a coluna idMercado.

Consequentemente no retângulo em verde, está sendo referenciada duas tabelas, ambas criadas já anteriormente, primeiramente cria as colunas e seu tipo de dado, idMercado inteiro e não nulo, e idProduto não nulo e inteiro, seguido das devidas referencias, chave estrangeira

vai ser a coluna idMercado e referência a tabela Mercado a coluna idMercado, mesma coisa a idProduto.

```
INSERT INTO Mercado
```

```
VALUES
```

```
(1, "MercadoNovo", "01.100.100/1001-00", "Rua Jeorge, n 40", "Um Bairro", "Florianopolis", "SC"),  
(2, "MercadoSuper", "02.020.020/0201-00", "Rua Cloves, n 20", "Outro Bairro", "Caçador", "SC"),  
(3, "MercadoUltra", "03.003.003/0031-00", "Rua Jonas, n 30", "Mais Um Bairro", "Videira", "SC");
```

```
INSERT INTO Produto
```

```
VALUES
```

```
(1, "Dona Benta 1Kg", "Farinha de Trigo", 35.60, "Jonas S, Paraná", NULL, 1),  
(2, "Nordeste 1Kg", "Farinha de Trigo", 25.75, "Jonas S, Paraná", 10, 2),  
(3, "Sabonete Protex U", "Higiene Pessoal", 2.50, "Santos K, Rio Grande do Sul", NULL, 3),  
(4, "Sabonete Phebo U", "Higiene Pessoal", 2.75, "Santos K, Rio Grande do Sul", 15, 3),  
(5, "Alface U", "Verdura", 15.00, "Fazenda Feliz, Paraná", 33, 2),  
(6, "Repolho U", "Verdura", 10.20, "Fazenda Feliz, Paraná", 44, 1);
```

```
INSERT INTO Venda
```

```
VALUES
```

```
(1, "10/10/2021", 25.75, 1, 2),  
(2, "10/10/2021", 30.00, 1, 2),  
(3, "10/10/2021", 35.60, 1, 1),  
(4, "10/10/2021", 10.20, 1, 1),
```

Recomendo copiar as instruções acima e colar em Executar SQL, se seu banco estiver devidamente correto seguindo os passos, vai ser executados as devidas inserções sem erro.

Para realizar as primeiras consultas com tabelas relacionadas usa-se o conceito de *joins*, que vai ser um dos últimos assunto desse documento, mas uma introdução a ele vai ser usada para o SQLite e as tabelas que foram criadas.

```
SELECT * FROM Produto
INNER JOIN Mercado
ON Mercado.idMercado = Produto.idMercado
```

Figura 30: Simples consulta com o Inner Join.

A instrução “**Inner Join**” retorna todos os valores que estejam em ambas as tabelas registradas ligadas pelo id e a chave estrangeira, ou seja, selecione todos os valores da tabela Produto, com uma junção interna com a tabela Mercado ligando pelo id da tabela mercado correspondente com a chave estrangeira da tabela Mercado na tabela Produto.

Ou seja, para realizar as junções são necessárias duas tabelas e pelo menos uma chave estrangeira em uma das tabelas para realizar a junção com o id correspondente da chave primária pelo id correspondente da chave estrangeira, caso o conceito não ficou muito claro, volte algumas páginas principalmente na modelagem dos dados, é o mesmo exemplo usado aqui tem as devidas chaves primárias e estrangeiras para o relacionamento das tabelas.

O conceito de *inner joins* e de todos as junções (*joins*) é o mesmo dos conjuntos na matemática, onde você desenhava as bolinhas e ligava os correspondentes números com os devidos símbolos matemáticos como o pressente e o contido, etc.

```
51 SELECT Mercado.Nome, Produto.Nome FROM Produto
52 INNER JOIN Mercado
53 ON Mercado.idMercado = Produto.idMercado
54
```

	Nome	Nome
1	MercadoNovo	Dona Benta 1Kg
2	MercadoSuper	Nordeste 1Kg
3	MercadoUltra	Sabonete Protex U
4	MercadoUltra	Sabonete Phebo U
5	MercadoSuper	Alface U
6	MercadoNovo	Repolho U

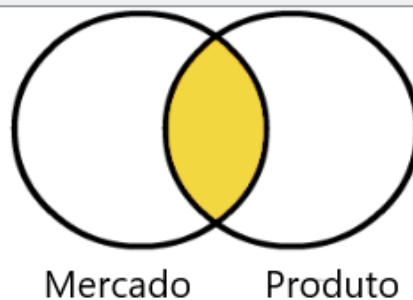


Figura 31: Nomes em tabelas iguais.

Repare a mesma consulta anterior, agora retornando somente o nome do mercado com o nome do produto, usa-se o operador ponto, para selecionar a coluna da tabela, “Tabela.Coluna”. Porém no resultado da consulta ambas as tabelas tem o mesmo Nome, para deixar a consulta mais adequada, usa-se apelidos.

<pre>SELECT Mercado.Nome as "Mercado", Produto.Nome as "Produto" FROM Produto INNER JOIN Mercado ON Mercado.idMercado = Produto.idMercado</pre>	
Mercado	Produto
MercadoNovo	Dona Benta 1Kg

Figura 32: Usando apelidos.

Usa-se a palavra reservada “as” para renomear as tabelas, entraremos em mais detalhes ao decorrer do documento sobre junções.

<pre>SELECT M.Nome, P.Nome FROM Produto P INNER JOIN Mercado M ON M.idMercado = P.idMercado</pre>	
Nome	Nome
MercadoNovo	Dona Benta 1Kg
MercadoSuper	Nordeste 1Kg

Figura 33: Usando apelidos.

Como podemos apelidar o resultado da consulta com a sintaxe anterior, é possível apelidar uma tabela apenas usando uma letra depois do nome da tabela, assim agiliza na hora da consulta e na seleção das colunas, o resultado é o mesmo da consulta anterior, apenas rearrumei a consulta para melhor entendimento.

Ao decorrer do documento vai ser comentado mais sobre as junções e seus outros tipos, esse foi apenas uma introdução as junções no *SQLite*, recomendo que continue fazendo outras junções com as outras tabelas, por exemplo uma junção com as três tabelas.

Imagine-se que seu líder vem até você e faz a seguinte pergunta, gostaria de saber quais são as vendas em quais mercados e qual o produto vendido.

<pre> SELECT V.DataVenda, M.Nome as "Mercado", P.Nome as "Produto" FROM Venda V INNER JOIN Mercado M ON M.idMercado = V.idMercado INNER JOIN Produto P ON P.idProduto = V.idProduto </pre>			
DataVenda	Mercado	Produto	
10/10/2021	MercadoNovo	Nordeste 1Kg	
10/10/2021	MercadoNovo	Nordeste 1Kg	
10/10/2021	MercadoNovo	Dona Benta 1Kg	
10/10/2021	MercadoNovo	Dona Benta 1Kg	
10/10/2021	MercadoNovo	Sabonete Protex U	
10/10/2021	MercadoNovo	Sabonete Protex U	

Figura 34: Usando apelidos.

Agradeço por chegar até aqui, agora tem uma listinha básica de 5 exercícios para prosseguirmos ao capítulo seguinte.

As atividades estão no final do capítulo, é o penúltimo link nas referências bibliográficas, você já pode testar seus conhecimentos no Quis também.

SQL Server Funcionalidades

Antes de começar os conceitos mais avançadinhos em SQL, recomendo fazer o download do SQL Server, para aprender mais uma ferramenta.

Para realizar o download dessa ferramenta, está em um segundo documento na lista de documentos.

Após ter realizado o download o SQL Server e de seu gerenciador de banco de dados, basta abrir o seu Microsoft SQL Server Management Studio 18, digitando *ssms*, na barra de pesquisa do Windows (Caso estiver usando o Windows) e clicar em conectar.

Contexto de Negócio em Mercado

Sim, eu tinha falado anteriormente que não iria comentar mais sobre isso, mas é interessante você entender o que iremos fazer a partir daqui.

Os irmãos Jonas e Eduardo, dois sócios e amigos de empreendimento de sucesso e estão planejando entrar no mercado, mais precisamente no varejo, com apenas alguns produtos, porem mesmo com os produtos de entrada, a ideia inicial e a audiência na região selecionada para começar o desenvolvimento do empreendimento, eles não têm uma expertise para desenvolver uma base de dados básica para começar a essa ideia, então ambos contrataram uma equipe de consultoria para conseguir realizar esse desafio.

Para a equipe de consultoria, fizeram as seguintes perguntas.

- Preciso de uma base de dados para armazenar meus produtos.
- Preciso de várias tabelas, como Empregados, Produtos, Provedores
- É necessário popular com alguns dados que possuo e testar.
- Estávamos pensando sobre uma categoria externa para futuros produtos ou uma categoria já no produto?

Vou deixar sua criatividade livre para escolher o nome do mercado.

Lembram-se, realmente entenda o que eles querem que você faça, quebre o problema em partes e anote detalhadamente o projeto a ser concluído, assim você pode assinar o contrato e vai desenvolver exatamente o que foi proposto nas reuniões, etc.

Como não estamos inseridos no negócio, tente imaginar-se como você faria isso, você deve ir ao mercado certo, então, nos mercados possui o pessoal que trabalha, chamados de empregados ou colaboradores, possui categorias para os produtos, pois você não vai ver uma pasta de dentes junto com um pedaço de carne no freezer por exemplo.

Possui também os provedores ou fornecedores dos alimentos, fornecedores de equipamento geral, de produtos higiênicos, frutas, entre outros, todos esses exemplos vão ser as tabelas.

Fundamentos das Tabelas

Apesar de já ter sido criado algumas tabelas no capítulo passado, vamos criar mais uma, porém, antes, é necessário criar um novo banco de dados no SQL Server.

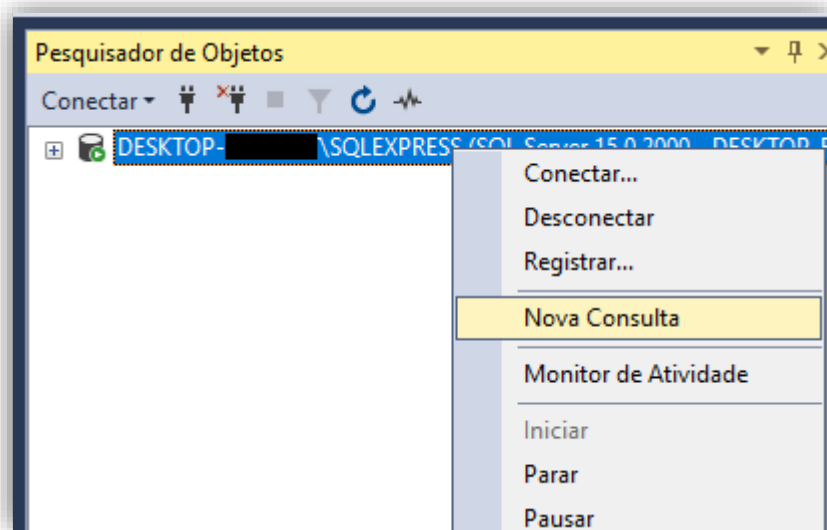


Figura 35: Iniciar uma consulta no SQL Server.

Para isso, vai ser desenvolvido tudo com instruções SQL, para começar, clique com o botão direito na sua base de dados do seu computador local e, em nova consulta.

```
CREATE DATABASE Market
```

Vou utilizar nomes do banco de dados e das tabelas em inglês, mas você pode escrever os nomes em português.

- Para deletar uma base de dados é *DROP DATABASE Nome*, caso queira deletar a 'Market', basta clicar com o botão direito e em nova consulta e deletar.

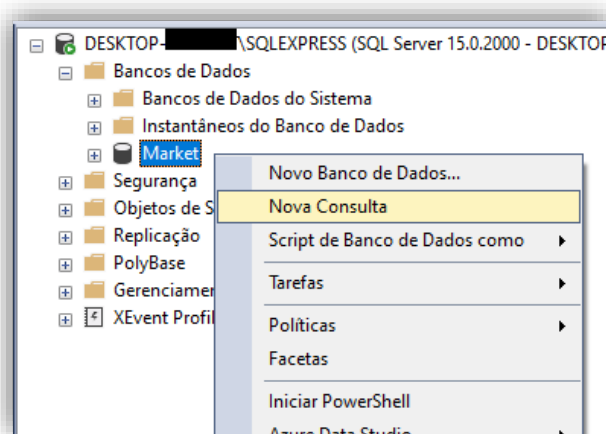


Figura 36: Realizar consultas em uma base de dados específica.

Seu banco de dados vai estar armazenado na pasta como mesmo nome, agora vamos acessar para criar nossas tabelas.

Palavras Reservadas para Tabelas

```
CREATE TABLE Category (  
    CategoryId INT PRIMARY KEY NOT NULL,  
    CategoryName VARCHAR(150) NOT NULL UNIQUE,  
);  
  
CREATE TABLE Supplier (  
    SupplierId INT PRIMARY KEY NOT NULL,  
    CompanyName NVARCHAR(100) NOT NULL UNIQUE,  
    ContactName NVARCHAR(50) NOT NULL,  
    City NVARCHAR(50) NOT NULL,  
    Address NVARCHAR(100) NOT NULL,  
    Phone NVARCHAR(20) NOT NULL
```

Figura 37: Criando uma tabela no SQL Server.

Tabela Categoria, passando dois parâmetros. *CategoryId* é um valor inteiro chave primária não nulo. *CategoryName* é uma coluna do tipo *varchar* até 150 caracteres, não nulo e **Único**.

A palavra reservada *Unique*. Aplicada a coluna, essa coluna não pode ter registros duplicados, ou seja, no nosso exemplo de mercado, não queremos categorias de produtos que sejam iguais.

A tabela *Supplier*, é fornecedor em inglês, essa tabela possui mais colunas que a tabela anterior, possui uma chave primária, e o único diferencial é o *CompanyName*, que é único.

```

CREATE TABLE Product (
    ProductId INT PRIMARY KEY NOT NULL,
    ProductName VARCHAR(150) NOT NULL UNIQUE,
    CategoryId INT FOREIGN KEY REFERENCES Category(CategoryId),
    Supplier INT FOREIGN KEY REFERENCES Supplier(SupplierId),
    UnitPrice MONEY NOT NULL,
);

CREATE TABLE Employee (
    EmployeeId INT PRIMARY KEY NOT NULL,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Title NVARCHAR(20) NOT NULL,
    Gender NCHAR(2) CHECK (Gender in ('M', 'F')),
    BirthDate DATETIME NOT NULL DEFAULT GETDATE(),
    HireDate DATETIME NOT NULL DEFAULT GETDATE(),
    State NVARCHAR(50) DEFAULT 'Ohio',
    Country NVARCHAR(10) DEFAULT 'USA',
    City NVARCHAR(100) DEFAULT 'Toleto',
    Photo IMAGE,
    Notes NTEXT,
);

```

Figura 38: Criando mais tabelas para a base de dados Market.

A tabela *Product*, que inglês significa produto, tem os mesmos requisitos da tabela anterior, contudo, a *CategoryId*, *Supplier*, são referências de outras tabelas, ou seja, são as chaves estrangeiras e a *UnitPrice*, preço unitário, seu formato é de uma palavra reservada para representar o dinheiro.

Tem inclusive um erro de digitação, pois faltou o Id, esse erro vamos corrigir ao decorrer do guia com instruções do próprio SQL.

Sobre a tabela funcionários, essa tabela contém várias informações que já foram citadas, porem a coluna *Gender*, está com mais uma palavra reservada, a palavra *check*, é uma **constraint**, uma limitação que podemos impor para que, nessa coluna só seja aceito os registros que sejam entre 'M' e 'F'. reparem que eu guardo esses registros em parênteses.

Na coluna *BirthDate* e *HireDate*, seus registros possuem três palavras reservadas novas, a instrução **Datetime**, **Default** e **Getdate()**, sobre o datetime já foi citado e explicado nos fundamentos de SQL.

A palavra reservada **Default**, caso nenhum valor seja registrado nas colunas com essa instrução, os valores vão ser aplicados a *função* ao lado, que é a *Getdate()*, essa função pega a data atual durante o registro.

Essa mesma lógica do default foi selecionada e aplicada nas próximas colunas, para obter valores *default*.

A coluna foto, o valor de seu registro é binário, porém no SQL Server, possui um nome reservado especial, o **Image**. Já a coluna 'Note', é um tipo de texto normal.

Essa próxima coluna, é na verdade um desafio especial.

O seu desafio é fazer uma listinha da tabela, comentando a respeito das suas colunas, por exemplo, 'Na tabela *Employee*, a primeira coluna é a *EmployeeId*, essa coluna significa que é a chave primária, a coluna que representa essa tabela, o tipo de dado aceito por essa coluna é do tipo inteiro e essa coluna não aceita valores nulos. Aliás, você sabia que essa *constraint* possui embutido as funcionalidades de *Unique* e *Not Null*, sendo assim eu não preciso especificar fora'.

Pois, Jonas e Eduardo querem saber algumas coisas a mais, caso você tenha fechado o escopo como citei anteriormente, pode prosseguir caso quebrou o problema em partes e realmente conversou e entendeu as causas do problema, ou seja, fechando assim o escopo do projeto, ou assinando a consultoria.

```
CREATE TABLE Customer (
    CustomerId INT PRIMARY KEY NOT NULL,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    State NVARCHAR(50) DEFAULT 'Ohio',
    Country NVARCHAR(10) DEFAULT 'USA',
);

CREATE TABLE Orders (
    OrdersId INT PRIMARY KEY NOT NULL,
    CustomerId INT FOREIGN KEY REFERENCES Customer(CustomerId),
    EmployeeId INT FOREIGN KEY REFERENCES Employee(EmployeeId),
    OrderDate DATETIME NOT NULL DEFAULT GETDATE(),
    ShippingDate DATETIME NOT NULL DEFAULT GETDATE()
);

CREATE TABLE OrderDetail (
    OrdersId INT PRIMARY KEY NOT NULL,
    ProductId INT FOREIGN KEY REFERENCES Orders(OrdersId),
    Quantity SMALLINT NOT NULL,
    UnitPrice MONEY NOT NULL,
);
```

Figura 39: Mais algumas tabelas com seus devidos tipos de dados.

Alterar e Modificar Colunas

Deletar Determinadas Colunas

Essa instrução é muito poderosa, ela que altera a coluna, deleta a coluna e modifica a mesma, essa é uma das instruções que não pode sair da sua caixinha de instruções da linguagem SQL.

Como tinha citado, ela possui várias funcionalidades, a primeira e mais simples, é a instrução para deletar uma tabela, vamos usar as palavras reservadas, **Alter**, **Table**, **Drop** e **Constraint**.

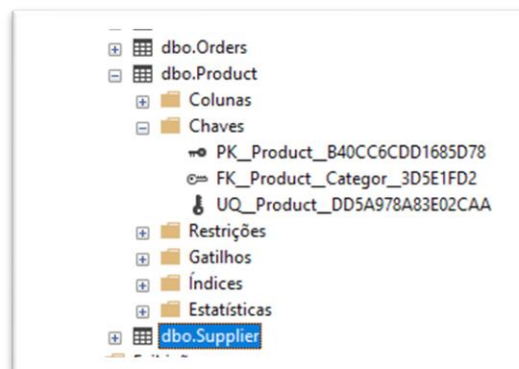
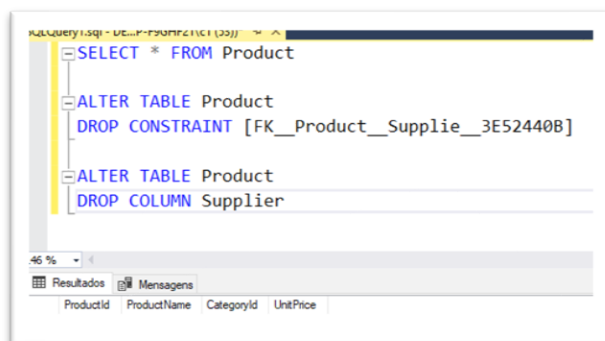


Figura 40: Deletar uma chave estrangeira de uma tabela.

Figura 41: Pasta chaves, onde está todas as informações de chaves estrangeiras e primárias da tabela.

Para remover uma chave estrangeira no SQL Server, antes é preciso desanexar a chave da tabela, para isso é usado o *drop constraint* e a localização da chave.

- Você pode arrastar a chave e colocar ela na parte da sua instrução para não ter que digitar todo o código.

Depois de desanexar a chave, é preciso dropar a tabela com a instrução *drop column* e passando a tabela em específico para realizar a deleção. provavelmente vai dar um erro, mas é só atualizar a base de dados e as colunas já vão ter sido deletadas corretamente, apenas um bug do SQL Server.

Geralmente realizar instruções com o SQL Server em chaves primárias e chaves estrangeiras é, em algumas vezes complicada para iniciantes, por isso, crie com cuidado suas tabelas.

Adicionar Novas Colunas

Para adicionar coluna no SQL é bem simples, basta usar o mesmo comando abordado no tópico acima, alterando a instrução de *DROP* para *ADD*, porem, temos que especificar o tipo de dados que essa coluna vai aceitar, no exemplo, usei o tipo inteiro.

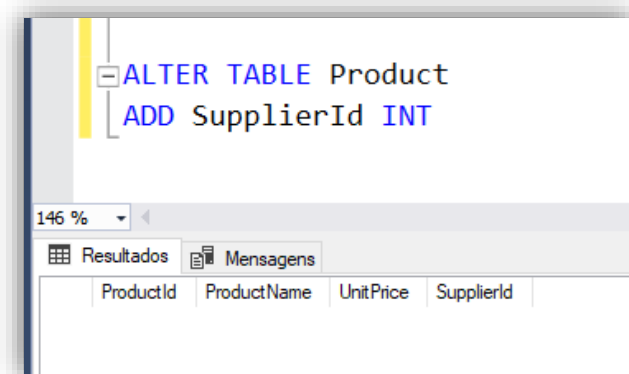


Figura 42: Adicionando coluna na tabela.

Sem muitos comentários, essa instrução é bem fácil de entender e aplicar.

Estou alterando a tabela *Product* e adicionando uma coluna nova chamada *SupplierId* do tipo INT.

Modificar Tabelas

Essa instrução, como a anterior, é bem simples, vamos passar a coluna da nossa tabela e modificar o seu atual tipo de dado, as vezes não é possível alterar devido a já ter dados na coluna, quando acontece isso, usamos outras técnicas.

```
ALTER TABLE Product  
ALTER COLUMN SupplierId BIT
```

Figura 43: Alterando uma coluna especifica da tabela.

Para modificar e colocar para chave estrangeira a coluna *SupplierId*, uma das formas é criar ela já com as configurações de uma chave estrangeira.

```
ALTER TABLE Product  
ADD SupplierId INT FOREIGN KEY REFERENCES Supplier(SupplierId)
```

Resultados Mensagens

ProductId	ProductName	UnitPrice	CategoryId	SupplierId
-----------	-------------	-----------	------------	------------

Figura 44: Adicionando chave estrangeira em uma coluna.

A outra forma é fazer uma referência já em uma coluna de uma tabela criada, ou seja, alterar o tipo da coluna para o tipo de chave estrangeira.

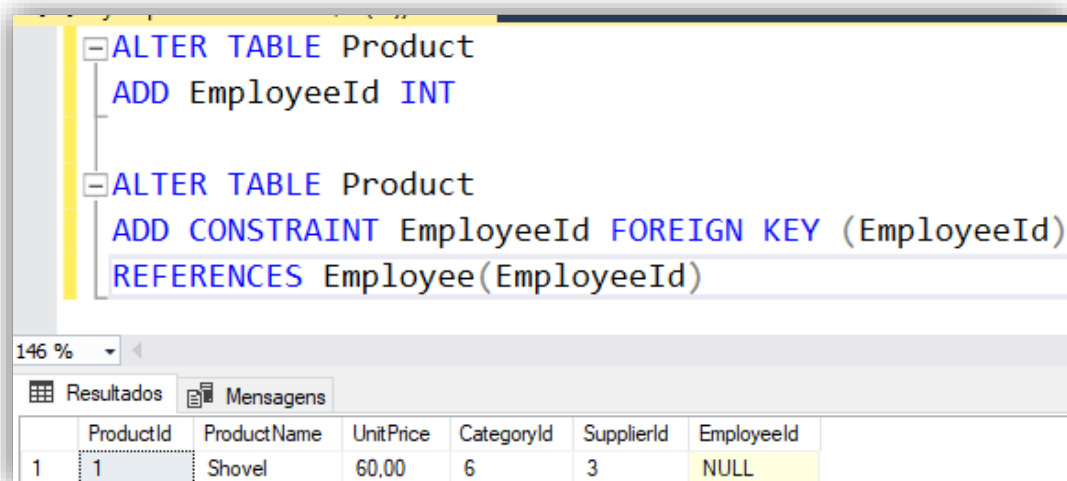


Figura 45: Adicionando mais chaves estrangeiras.

Alterar tabela Produto,

Adicionar *constraint* a coluna *EmployeeId* do tipo chave estrangeira (*EmployeeId*)

Referência a tabela *Employee* (*EmployeeId*).

Caso já tiver registros, a coluna vai assumir o valor nulo.

Funções e Apelidos

Como já foi citado anteriormente, podemos fazer inúmeras contas matemáticas, mas existe algumas funções que a própria linguagem SQL que nos oferece.

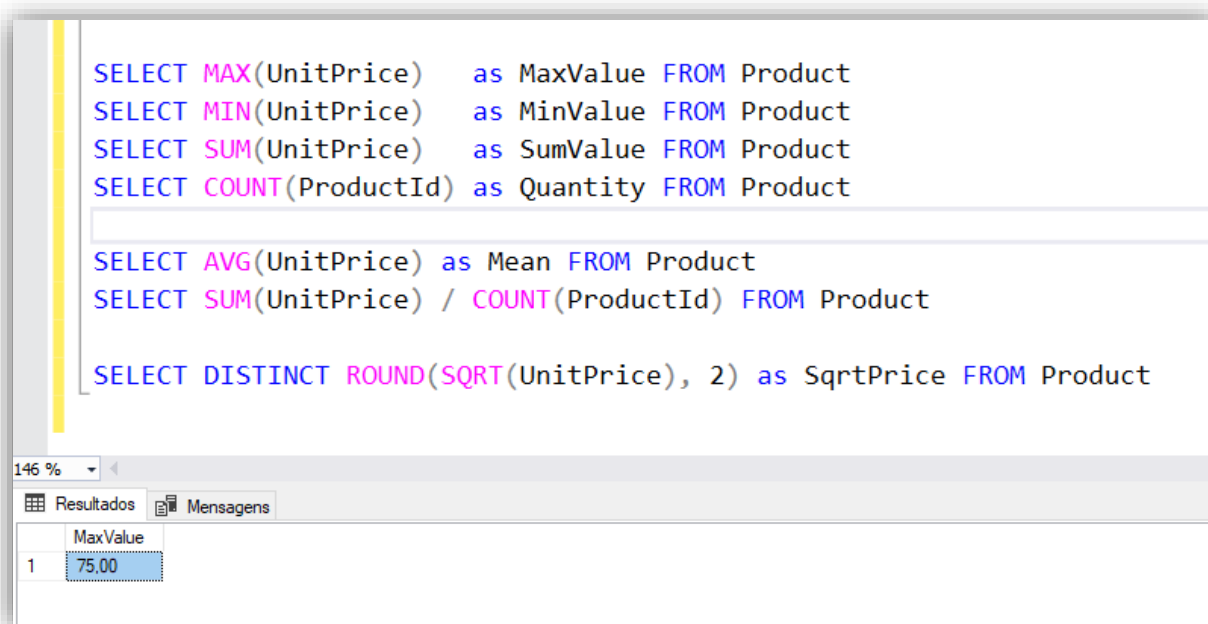


Figura 46: Algumas funções em SQL Server.

As funções são palavras reservadas, que recebem como parâmetro as colunas que você quer realizar as instruções.

A função **MAX**, retorna o maior valor registrado na coluna que foi indicada entre parênteses, a função **MIN**, retorna o menor valor registrado, a função **SUM** soma todos os valores registrados na coluna indicada, e a função **COUNT**, conta a quantidade de valores registrados.

A função **AVG**, retorna a média de valores da coluna especificada, e também é possível aplicar os operadores matemáticos nos resultados das funções, o resultado da instrução após a instrução da média, é a própria média escrita de outra forma, a soma de valores dividido pela quantidade de valores, ambos as instruções retornaram à média.

Pode-se criar algumas funções aninhadas, funções dentro de outras funções, a última instrução vai retornar os valores *únicos*, onde aplica-se a função **SQRT**, que retorna a raiz quadrada, porém o resultado dessa função está em outra função, a **ROUND**, essa instrução, reduz os caracteres depois da vírgula, como muitos valores da raiz quadrada vem com dígitos após a vírgula, foi aplicado uma função de arredondamento, passando o valor para ser arredondado e a quantidade de casas após a vírgula, que no exemplo da imagem, foram duas casas.

Reparem que após a função, existe outra palavra reservada, a palavra **AS**, essa instrução indica que a instrução após, vai ser um apelido para a essa coluna, entendemos isso como o nome da coluna, pois, após aplicar alguma função, a coluna não vai possuir nome, assim podemos dar um nome a essa nova coluna criada, reparem que no resultado da consulta, a coluna está com o nome de MaxValue. Futuramente usaremos muitas essas funções e os apelidos.

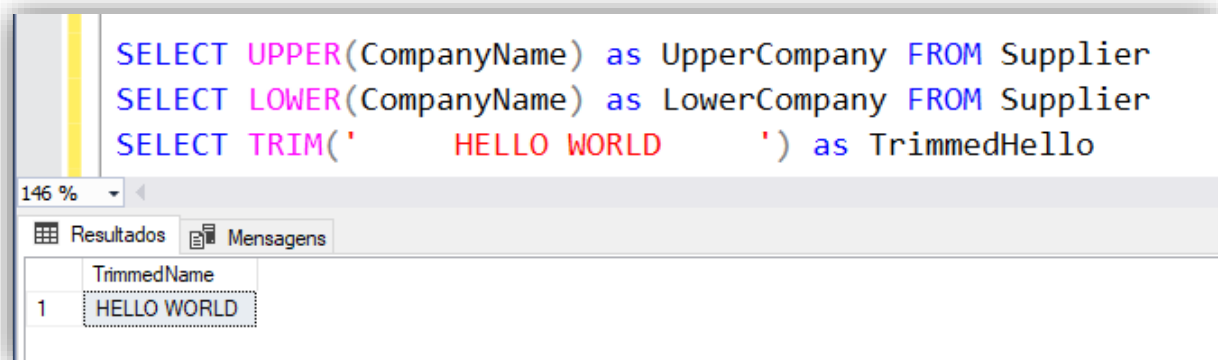


Figura 47 Mais algumas funções.

Similarmente, temos funções para registros no formato *string*, existem muitas outras funções, mas esse é um guia básico.

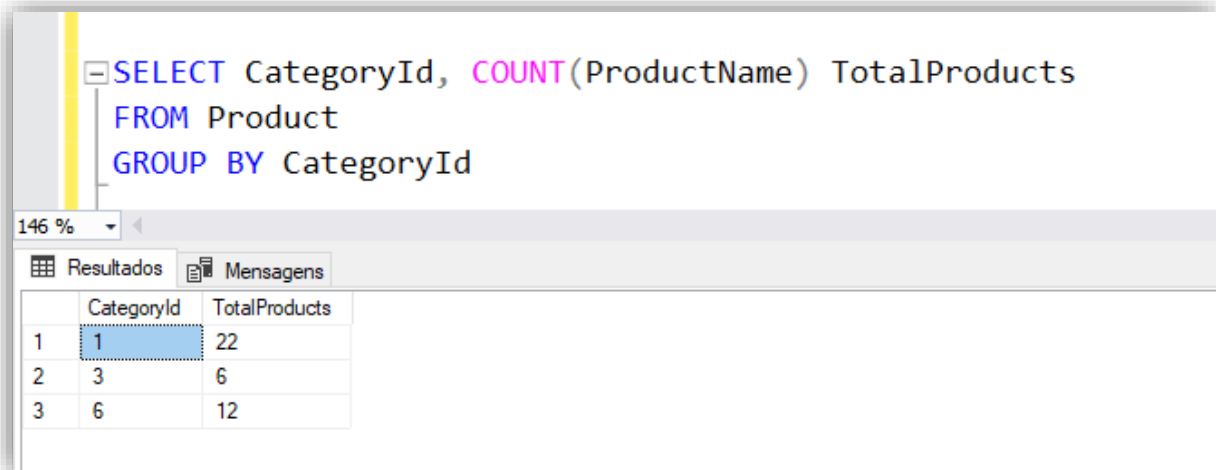
A função **UPPER**, converte todos os registros passados como parâmetros para maiúsculo.

A função **LOWER**, converte todos os registros para minúsculo.

A função **TRIM**, é a mesma da estatística, sua tradução é 'aparar', o registro passado como parâmetro, caso o usuário tiver digitado algum espaço a mais na digitação de seu formulário, a função vai remover esse espaço.

Instruções de Agrupamento

Uma das formas de agrupar valores no SQL é com a instrução **GROUP BY**, essa instrução é fácil de compreender, basicamente, vamos agrupar valores aplicando alguma dessas funções citadas a cima a alguma coluna com registros em alguma tabela da nossa base de dados.



The screenshot shows a SQL query window with the following text:

```
SELECT CategoryId, COUNT(ProductName) TotalProducts
FROM Product
GROUP BY CategoryId
```

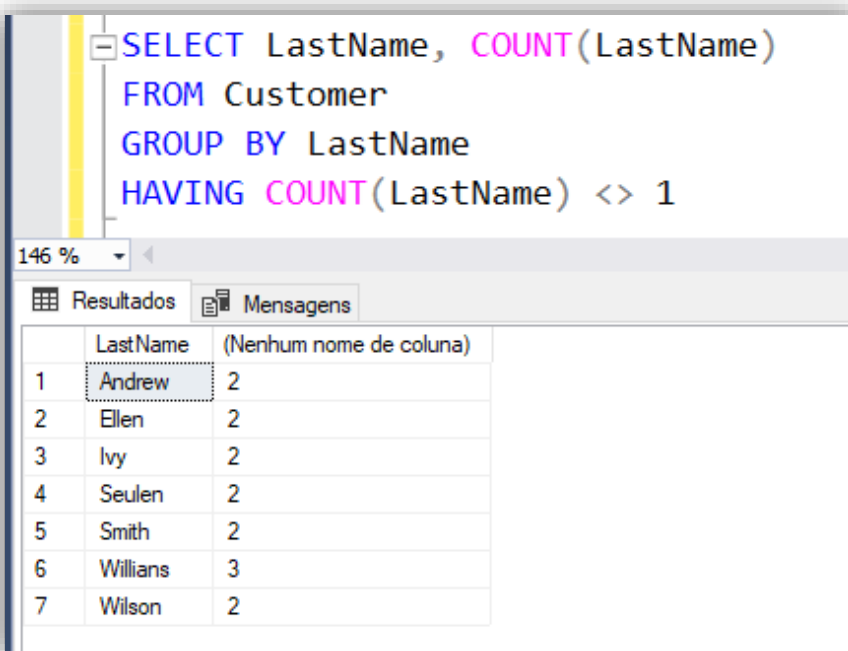
Below the query, the results are displayed in a table with two columns: **CategoryId** and **TotalProducts**. The table has three rows of data.

	CategoryId	TotalProducts
1	1	22
2	3	6
3	6	12

Figura 48: Simples agrupamentos de valores.

Nessa instrução, está sendo agrupado com a função **COUNT**.

Selecionar a *CategoryId*, contar a quantidade de produtos e apelidar de '*TotalProducts*' da tabela *Products*, e agrupar pelas *CategoryId*.



The screenshot shows a SQL query window with the following text:

```
SELECT LastName, COUNT(LastName)
FROM Customer
GROUP BY LastName
HAVING COUNT(LastName) <> 1
```

Below the query, the results are displayed in a table with two columns: **LastName** and **(Nenhum nome de coluna)**. The table has seven rows of data.

	LastName	(Nenhum nome de coluna)
1	Andrew	2
2	Ellen	2
3	Ivy	2
4	Seulen	2
5	Smith	2
6	Willians	3
7	Wilson	2

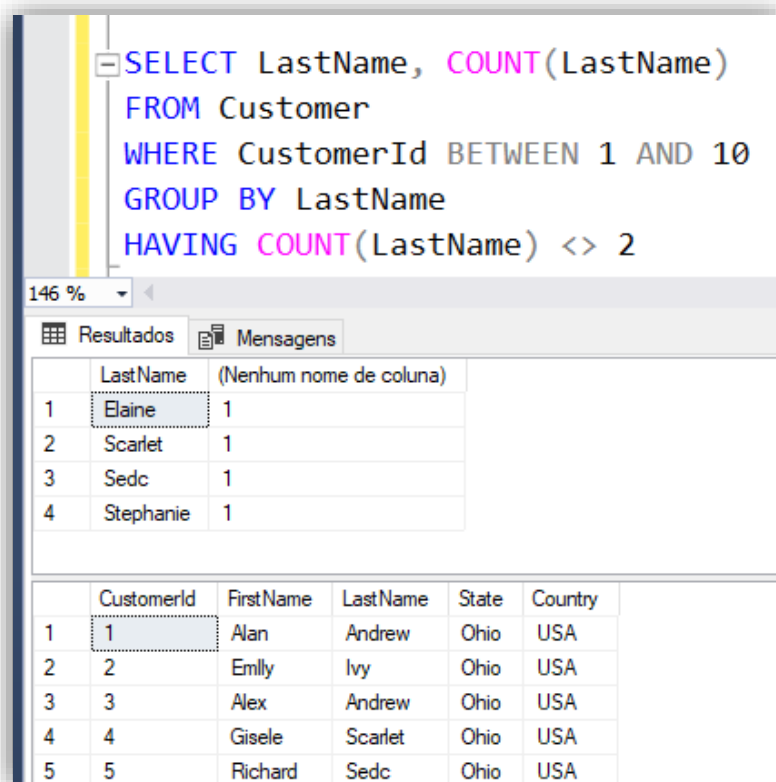
Figura 49: Instrução *Having*.

É possível aplicar uma instrução parecida com o *Where*, que é específica para agrupamentos, a instrução **HAVING**, essa instrução só funciona após o agrupamento, juntamente com uma função.

Estou selecionando, todos os clientes, que possuem o Sobrenome diferente do registro com valor um.

Outro ponto legal para compartilhar nesse guia, é que na minha instrução, não apliquei o apelido, sendo assim, essa nova coluna gerada a partir da minha consulta, não possui nome.

Caso queiram dicas para popular essa base de dados, no penúltimo capítulo desse guia, tem a lista de referências, e lá tem um dos sites que obtive algumas referências para colocar na base de dados.



```
SELECT LastName, COUNT(LastName)
FROM Customer
WHERE CustomerId BETWEEN 1 AND 10
GROUP BY LastName
HAVING COUNT(LastName) <> 2
```

146 %

Resultados Mensagens

	LastName	(Nenhum nome de coluna)
1	Elaine	1
2	Scarlet	1
3	Sedc	1
4	Stephanie	1

	CustomerId	FirstName	LastName	State	Country
1	1	Alan	Andrew	Ohio	USA
2	2	Emilly	Ivy	Ohio	USA
3	3	Alex	Andrew	Ohio	USA
4	4	Gisele	Scarlet	Ohio	USA
5	5	Richard	Sedc	Ohio	USA

Figura 50: Instrução *Having* com *GroupBy*.

Podemos sem problema utilizar instrução *Where*, para realizar consultar mais sigilosas. Nesse caso, ele fica antes go *Group by*. O resultado abaixo, é parte da seleção total.

Sub Consulta

A Sub consulta ou consulta ‘aninhada’, ou em inglês, **SubQuery**, é uma consulta dentro de uma instrução **WHERE**, nessa sub consulta, as colunas dentro da sub consulta, devem corresponder as consultas externas e não pode conter ordenamento em uma sub consulta.

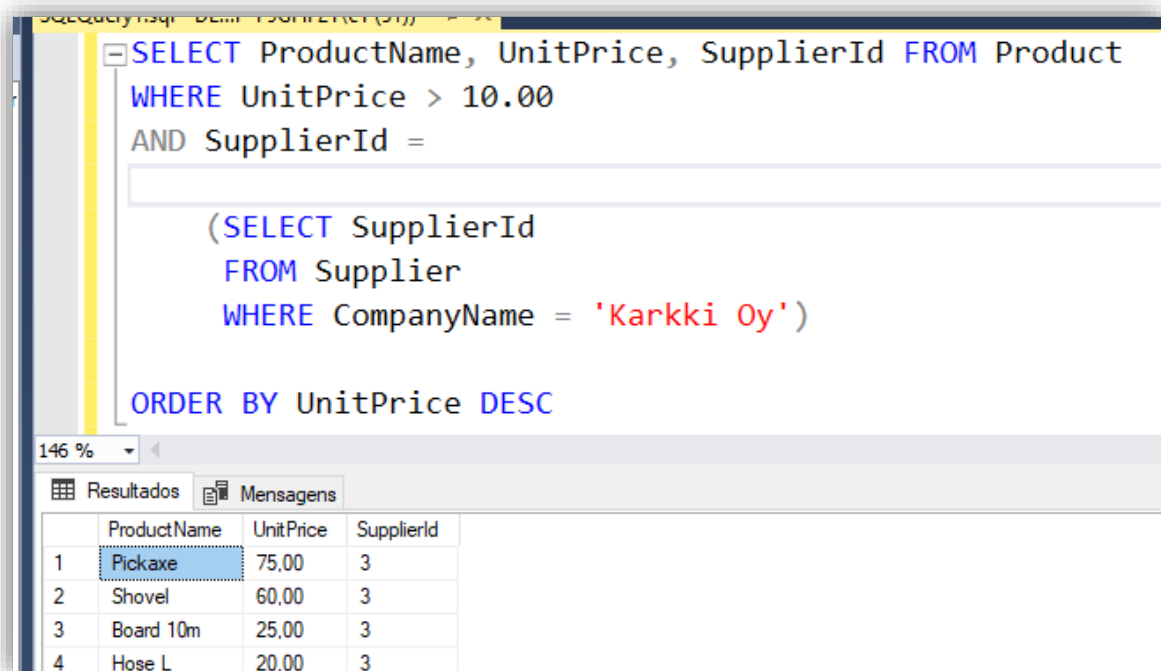


Figura 51: Primeira *SubQuery*.

A consulta na imagem, utiliza-se duas tabelas diferentes para obter os registros, um espaço para fins diádicos, apenas para ficar mais fácil de visualizar.

A instrução *Where* é a responsável pela instrução da sub consulta, está sendo buscando da tabela *Supplier*, todos os *SupplierId*, cuja *CompanyName* é igual a '*Karkki Oy*', esse resultado, vai ser voltado ao *Where*, para concluir a instrução, retornando os devidos registros da consulta.

- Uma sub consulta também suporta outra sub consulta.

Junções

As junções, ou em inglês **Joins**, a junção mais simples que existe é o **SELF JOIN**. Pois, apenas trabalhemos com mais de uma tabela na sub consulta, então, vamos agora utilizar, mais de uma tabela para realizar consultas ainda melhores.

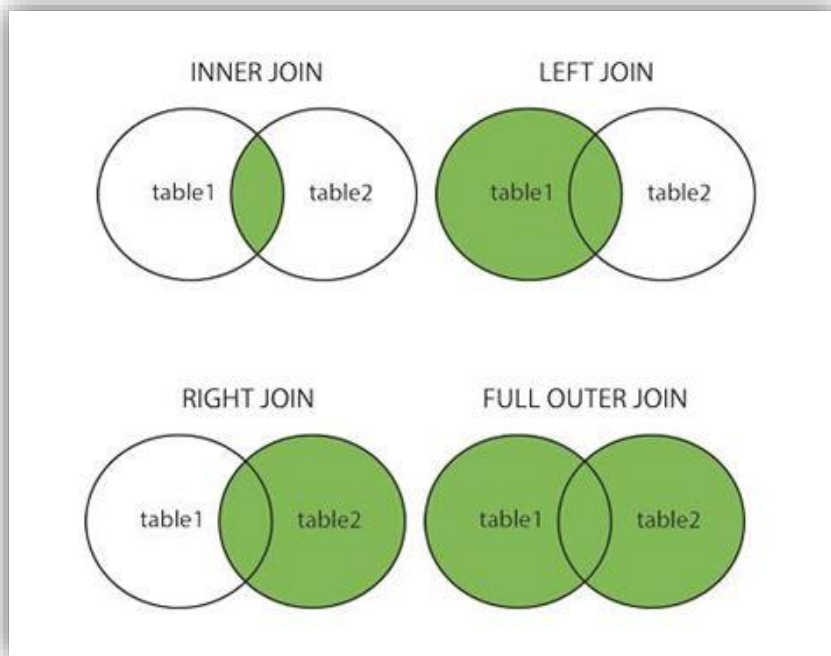


Figura 52: Gráfico de Junções.

Repare na imagem ao lado, é uma imagem já bem utilizada, porém, representa bem o que as junções fazem, são divididos em cinco categorias, que ao decorrer desse tópico, vai ser abordado e utilizado esses tipos de instruções.

A junção mais fácil de compreender, é a Self Join, que não está desenhada da imagem.

Self Join

Onde a tabela se junta a ela mesmo. Sim, é isso mesmo, nessa consulta, entra alguns conceitos de correlação, chamamos a mesma tabela duas vezes, porem com apelidos diferentes.

O conceito de correlação é bastante extenso, não vou abordar esse método estatístico.

`SELECT * FROM Product p, Product p2`

	ProductId	ProductName	UnitPrice	CategoryId	SupplierId	ProductId	ProductName	UnitPrice	CategoryId	SupplierId
1	1	Shovel	60,00	6	3	1	Shovel	60,00	6	3
2	2	Pickaxe	75,00	6	3	1	Shovel	60,00	6	3

Figura 53: Selecionando duas vezes da mesma coluna.

A priori é meio confuso, mas entenda agora que, o sgbd, fez todas as possíveis combinações de ambas as tabelas, para ela mesmo, minhas simples quarenta linhas de registros, se tornaram mil e seiscentos registros.



Podemos aplicar filtros também, assim organizando melhor as consultas de correlação entre os registros, que no caso da tabela são os produtos.

Nessa próxima consulta, vai ser colocar em prática, selecionando apenas quatro colunas das duas tabelas, e comparando o valor de preço iguais para diferentes produtos registrados em ambas as tabelas da base de dados.

```

SELECT
    p.ProductId, p.ProductName, p.UnitPrice,
    p2.ProductId, p2.ProductName, p2.UnitPrice
FROM Product p, Product p2
WHERE p.UnitPrice = p2.UnitPrice
ORDER BY p.UnitPrice DESC

```

	ProductId	ProductName	UnitPrice	ProductId	ProductName	UnitPrice
1	2	Pickaxe	75,00	2	Pickaxe	75,00
2	1	Shovel	60,00	1	Shovel	60,00
3	5	Board 10m	25,00	5	Board 10m	25,00
4	12	Hose L	20,00	12	Hose L	20,00
5	17	Bean 20C	20,00	12	Hose L	20,00
6	18	Bean 30C	20,00	12	Hose L	20,00
7	12	Hose L	20,00	17	Bean 20C	20,00
8	17	Bean 20C	20,00	17	Bean 20C	20,00
9	18	Bean 30C	20,00	17	Bean 20C	20,00
10	12	Hose L	20,00	18	Bean 30C	20,00
11	17	Bean 20C	20,00	18	Bean 30C	20,00
12	18	Bean 30C	20,00	18	Bean 30C	20,00
13	4	Board 05m	15,00	11	Hose M	15,00
14	11	Hose M	15,00	11	Hose M	15,00
15	24	Soap Ub Large	15,00	11	Hose M	15,00

Figura 54: Selecionando duas vezes da mesma coluna com mais instruções.

Inner Join

Diferente do *Self Join*, o **Inner Join**, ele irá retornar valores, que estejam, de alguma forma ligados pelas chaves, no caso, a *foreign key*, ou chave estrangeira, que é a chave primária de uma tabela que esta sendo referenciada por outra tabela.

	ProductId	ProductName	UnitPrice	CategoryId	SupplierId
1	1	Shovel	60,00	6	3
2	2	Pickaxe	75,00	6	3

Lembra da tabela *Product*, ela possui duas chaves estrangeiras.

A chave para a categoria em qual ela pertence e para o seu devido fornecedor, porém, para alterar o id e colocar o novo respectivo, usamos o **Join**.

Esse tipo de *join*, vai retornar valores que estejam tanto na tabela da *esquerda*, quanto na Tabela referenciada. Entraremos mais em detalhes sobre as tabelas ao decorrer do documento.

The screenshot shows a SQL query editor with the following query:

```
SELECT TOP 10
    p.ProductId, p.ProductName, p.UnitPrice,
    p.SupplierId,
    c.CategoryName
FROM Product p
INNER JOIN Category c
ON c.CategoryId = p.CategoryId
ORDER BY p.UnitPrice
```

Below the query, the results are displayed in a table with 6 columns: ProductId, ProductName, UnitPrice, SupplierId, and CategoryName. The results are ordered by UnitPrice.

	ProductId	ProductName	UnitPrice	SupplierId	CategoryName
1	9	Wrench K	2,00	3	Geral Products
2	19	Soap Small	2,00	2	Hygienic Products
3	22	Soap Ub Small	2,00	2	Hygienic Products
4	30	Toothpaste Balkan	2,00	2	Hygienic Products
5	31	Toothpaste Island	3,00	2	Hygienic Products
6	21	Soap Sand Small	3,00	2	Hygienic Products

Figura 55: Consultas com *Inner Join*.

Similarmente ao *self join*, usa-se as palavras para apelidar as tabelas, após selecionar a tabela principal, vai ser utilizado a palavra reservada **INNER JOIN**, e digitar a tabela a qual vai ser referenciada, também outra palavra reservada, a **ON**, que resumidamente como já foi abordado vai fazer a junção das tabelas, partindo das chaves que estão presentes nas duas tabelas.

Contudo, a tabela *SupplierId*, ainda está sem a devida identificação, podemos usar a mesma junção, e a instrução vai ficar assim.

```

SELECT TOP 10
    p.ProductId, p.ProductName, p.UnitPrice,
    s.CompanyName,
    c.CategoryName
FROM Product p

INNER JOIN Category c
ON c.CategoryId = p.CategoryId
INNER JOIN Supplier s
ON s.SupplierId = p.SupplierId

ORDER BY p.UnitPrice DESC

```

	ProductId	ProductName	UnitPrice	CompanyName	CategoryName
1	2	Pickaxe	75.00	Karkki Oy	Geral Products
2	1	Shovel	60.00	Karkki Oy	Geral Products
3	5	Board 10m	25.00	Karkki Oy	Geral Products
4	12	Hose L	20.00	Karkki Oy	Geral Products
5	17	Bean 20C	20.00	Gai pâturage	Cereals
6	18	Bean 30C	20.00	Gai pâturage	Cereals
7	11	Hose M	15.00	Karkki Oy	Geral Products
8	4	Board 05m	15.00	Karkki Oy	Geral Products
9	24	Soap Ub Large	15.00	Zaanse Snoepfabriek	Hygienic Products
10	16	Rice 30C	13.00	Gai pâturage	Cereals

Figura 56: Consultas com mais de um *Inner Join*.

O Inner Join, ficou logo a baixo do outro Inner Join, sempre, sendo referenciado pela tabela mãe, que no caso é a tabela *Product*. A instrução é a mesma que a anterior, só que, dessa vez, deixando a nossa consulta mais bonita e ordenando em forma decrescente.

Full Join

Esse *join*, vai juntar todos os registros da tabela principal e da tabela referenciada, ou seja, a tabela principal é a tabela da esquerda, e a referenciada é a tabela da direita.

```

SELECT
    A.ProductName , A.UnitPrice,
    C.CategoryName
FROM Product A

FULL JOIN Category C
ON C.CategoryId = A.CategoryId

ORDER BY UnitPrice ASC

```

	ProductName	UnitPrice	CategoryName
1	NULL	NULL	Alcoholic Drinks
2	NULL	NULL	Bottled Juices
3	NULL	NULL	Canned Food
4	NULL	NULL	Fruits and Vegetables
5	NULL	NULL	Meat and Meat Products
6	NULL	NULL	Refrigerated Products
7	Wrench K	2,00	Geral Products
8	Soap Small	2,00	Hygienic Products

Figura 57: Consultas com mais de um *Inner Join*.

Esse *join*, retornou todos os registros, independentemente se já foram atribuídos a algum outro registro, ou seja, da tabela *Category*, existem registros de algumas categorias que não foram associadas a nenhum produto.

Seguindo essa lógica, não existe nenhum valor atribuído a essa categoria, então esse *Join*, atribuiu todos os valores dessa consulta para nulos, lembrando que nas tabelas foi aplicado a restrição de não aceitar registros nulos.

Porém, o *full outer join*, ou *full join*, faz essa associação quando, nenhum registro foi atribuído a sua referência, lembrando, eu poderia fazer a junção pela categoria referenciando a tabela *product*.

Left e Right Join

Quando juntamos as tabelas, temos dois lados, esquerda e direita, o lado esquerdo, é a tabela que você usa na instrução *from*, e a da direita, é a referenciada pela *foreign key*.

Left Screenshot (Right Join):

```

SELECT
    A.ProductName , A.UnitPrice,
    C.CategoryName
FROM Category C

RIGHT JOIN Product A
ON C.CategoryId = A.CategoryId

ORDER BY UnitPrice ASC

```

	ProductName	UnitPrice	CategoryName
1	Wrench K	2,00	Geral Products
2	Soap Small	2,00	Hygienic Products
3	Soap Ub Small	2,00	Hygienic Products
4	Toothpaste Balkan	2,00	Hygienic Products
5	Toothpaste Island	3,00	Hygienic Products
6	Soap Sand Small	3,00	Hygienic Products

Right Screenshot (Left Join):

```

SELECT
    A.ProductName , A.UnitPrice,
    C.CategoryName
FROM Category C

LEFT JOIN Product A
ON C.CategoryId = A.CategoryId

ORDER BY UnitPrice ASC

```

	ProductName	UnitPrice	CategoryName
1	NULL	NULL	Meat and Meat Products
2	NULL	NULL	Canned Food
3	NULL	NULL	Fruits and Vegetables
4	NULL	NULL	Bottled Juices
5	NULL	NULL	Alcoholic Drinks
6	NULL	NULL	Refrigerated Products
7	Wrench K	2,00	Geral Products
8	Soap Small	2,00	Hygienic Products

Figura 58: Consultas com *Right Join* e *Left Join*.

Inverti as instruções, consultando agora pela categoria, juntando pelo produto, reparem que, na primeira imagem, usei a coluna da direita, que são os produtos, assim, essa consulta vai retornar os registros que estejam em alguma categoria registrada, resumindo, nesse contexto, teve o mesmo resultado de uma consulta com instruções de *inner join*.

Já a segunda imagem, foi agrupado as categorias com os produtos, porém, os produtos sem registros também foram retornados.

Finalizando assim nosso primeiro ciclo de estudo de SQL, existem mais coisas legais para estudar como por exemplo chaves primarias compostas, mais informações sobre normas, entre muitas outras coisas, inclusive NoSQL e BigData.

Restaurando uma base de dados

Já acabou a lista de atividades da pasta SQL Server Fundamentos, está livre para começar!

Pois bem, vou mostrar nesse último tópico, como fazer uma restauração de uma base dedados real, basta fazer o download do arquivo .BAK.

<https://github.com/Microsoft/sql-server-samples/releases/download/adventureworks/AdventureWorksLT2019.bak>

Para realizar o download, basta copiar o link.

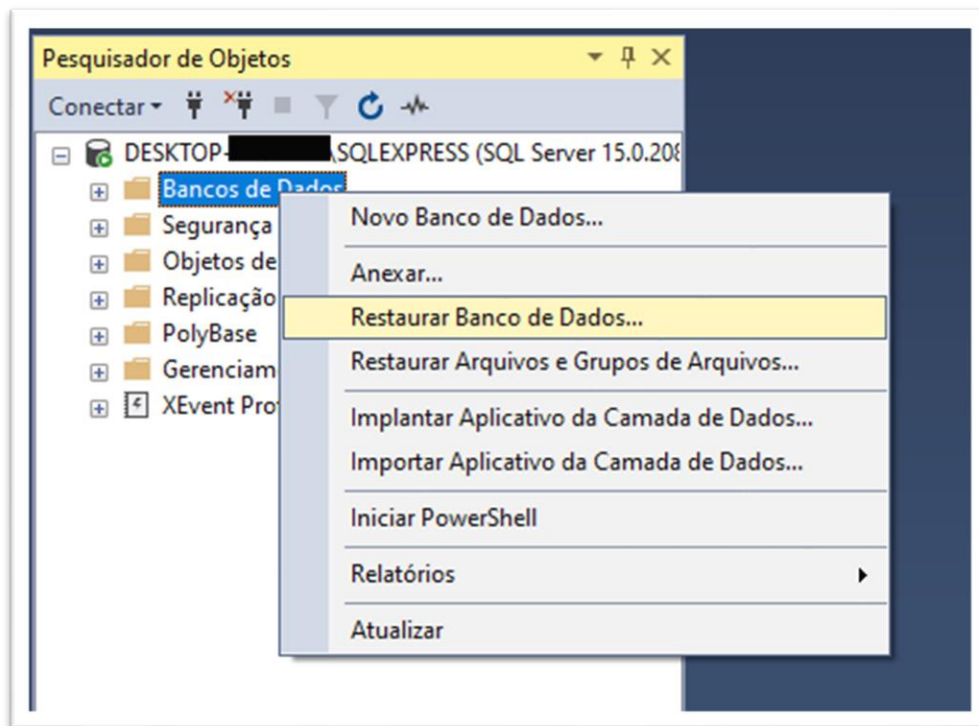


Figura 59: Restaurar uma base de dados.

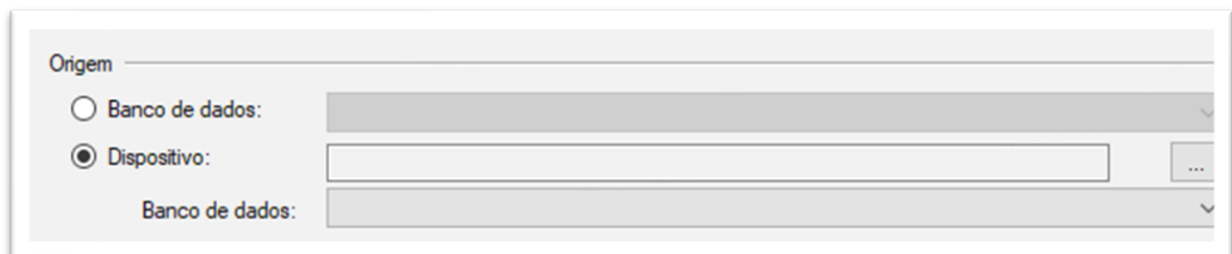


Figura 60: Informações da localização da base de dados.

Clique em Dispositivo, após abrir o *popup* de informações da restauração da base de dados.

Observação, coloca-se o download do arquivo *BAK*, de fácil acesso para você. depois que achar o devido diretório onde você salvou o arquivo, basta clicar no arquivo a direita e clicar no botão adicionar.

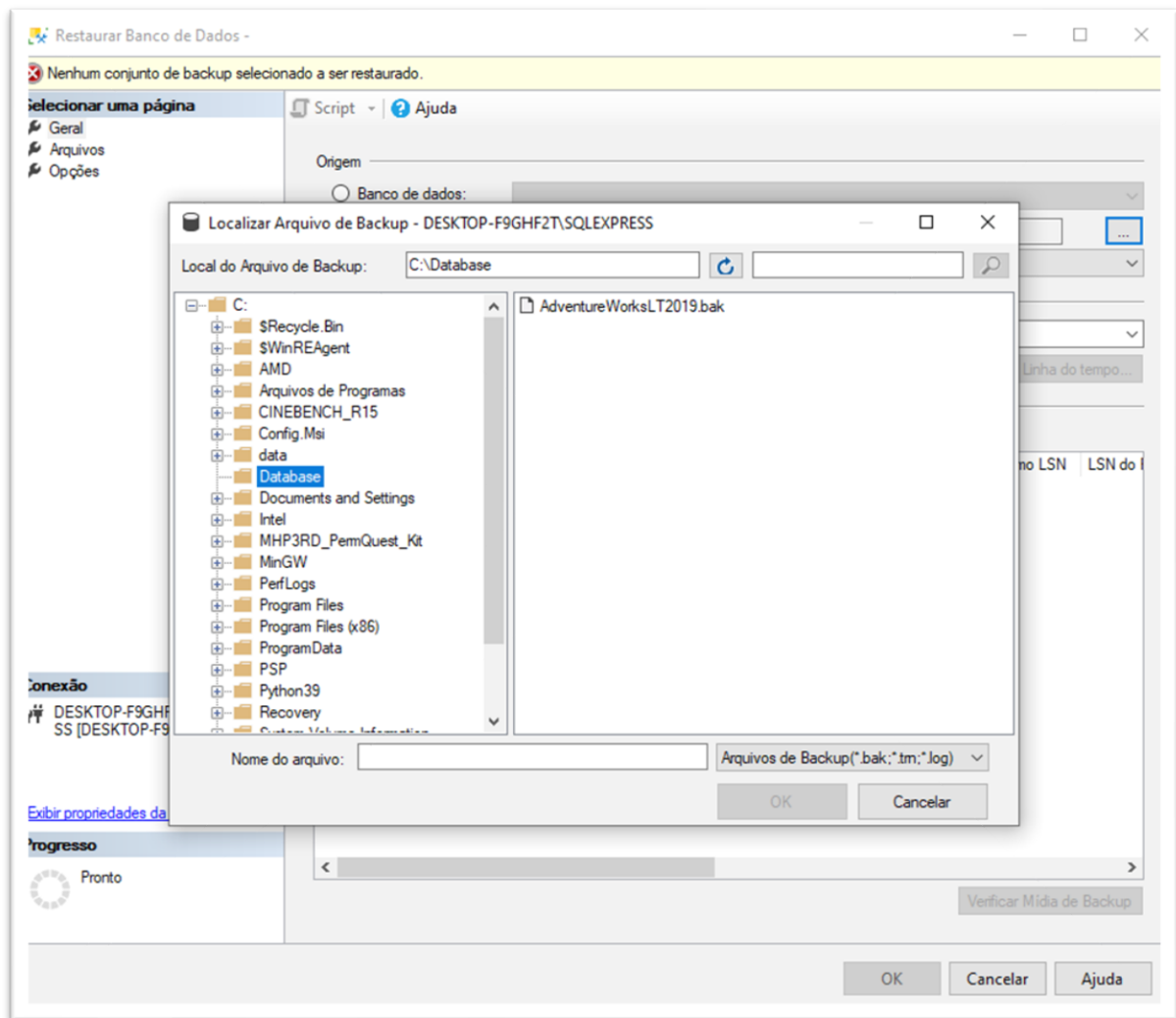


Figura 61: Selecionando o arquivo para restauração.

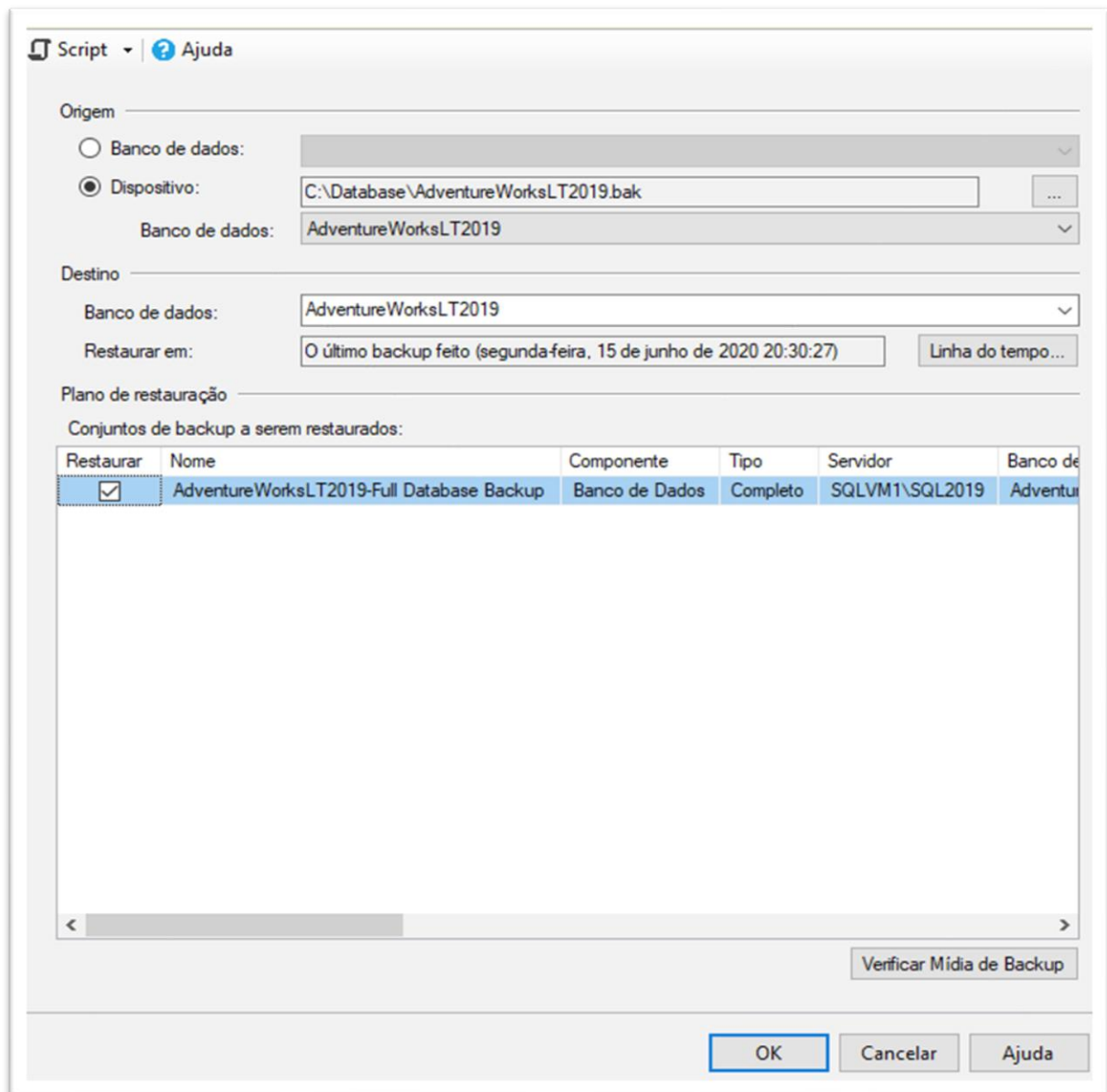


Figura 62: Preparando a restauração.

Basta marcar a *checkbox* Restaurar e clicar em Ok.

Agora você tem uma base de dados novinha em folha para poder realizar suas consultas.

Glossário

Atributos: Lembra da função filtro do Excel, onde você aplicava essa função nas colunas, pois bem, os atributos são as colunas.

Banco de Dados Relacional: Imagine uma tabela no Excel, um banco de dados é isso. O conceito de banco de dados relacional, está, devido a relação dessas tabelas por chaves, veremos mais à frente esse conceito, mas, por agora, entendemos que são tabelas organizadas em linhas e colunas

Chave Estrangeira (FK): Quando uma PK está em na linha de outra tabela.

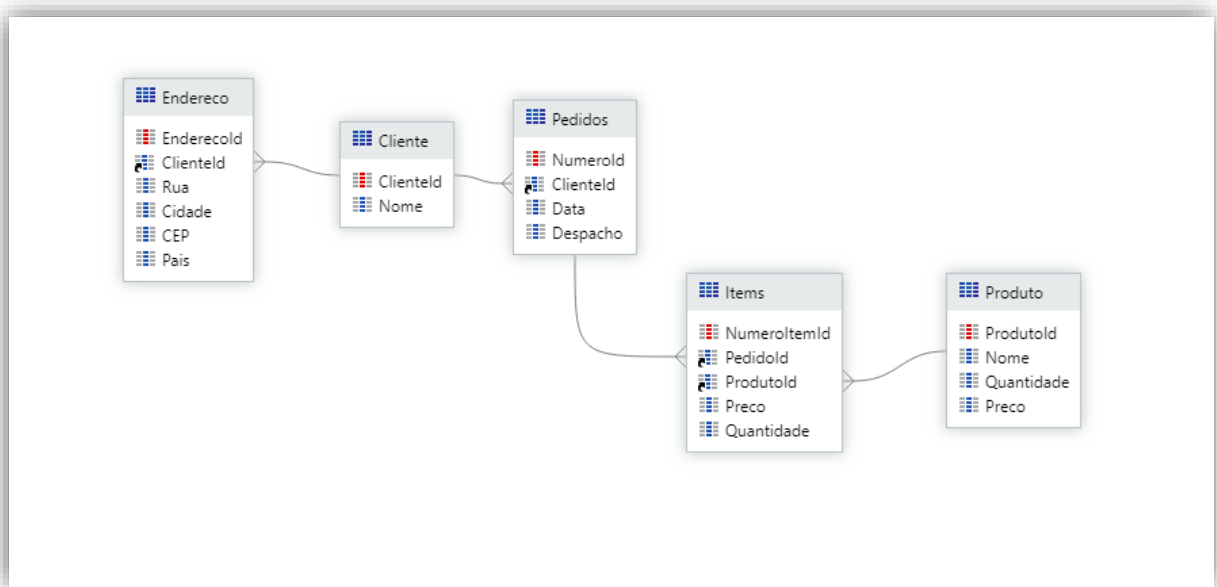


Figura 63: Esquema de Base de dados básica.

Chave Primária (PK): Chave principal, usada para identificar e diferenciar registros, busca a unicidade das linhas.

- É possível somente uma chave primária por tabela.
- Uma tabela pode conter a chave primária de outra tabela.

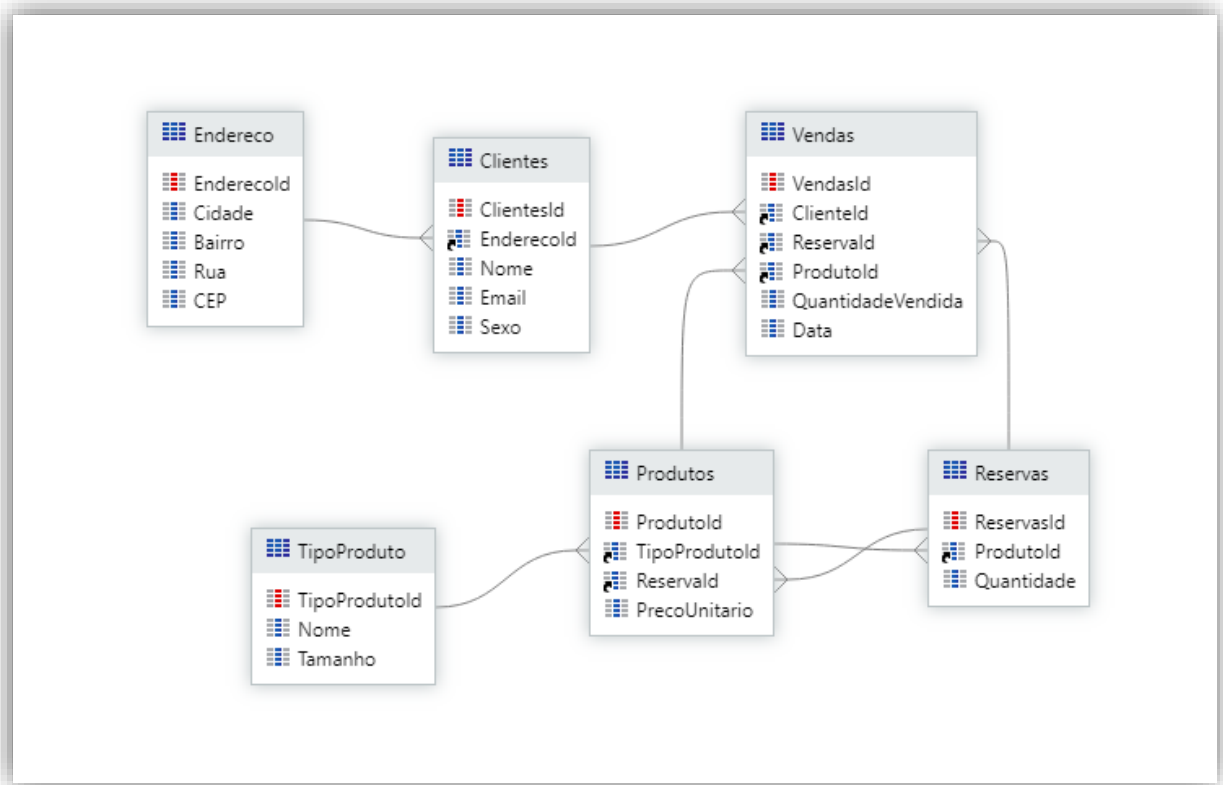


Figura 64: Outro Esquema de Base de dados básica.

Checkbox: São caixas selecionáveis, geralmente vemos essas caixinhas quando estamos respondendo algum formulário online ou validando as clássicas caixas de ‘Eu não sou um Robô’.

Domínio: Conjunto de valores que uma coluna pode possuir, se eu tiver valores numéricos, só poderei ter valores numéricos.

Entidade: Nada mais é que uma tabela.

Índice: É uma lista de valores ordenados que apontam a posição das linhas, muito útil para agilizar a consulta.

Integridade Referencial: Conceito de relação das tabelas, ou seja, a partir dos dados, eu não consigo fazer determinada ação devido a essa relação, exemplo, se houver registros relacionados eu não consigo remover esse determinado registro, como uma escada, não consigo subir estando já em cima.

Normalização: Redução das tabelas em tabelas menores como na última foto apresentada.

RDBMS / SGBD (Relational Database Management System): É uma GUI, um software que manipula nosso Banco de Dados Relacional.

Registro: São as linhas das tabelas, também chamada de Tupla, que podem conter valores nulos.

Scripts: Entenda como um arquivo originado de uma linguagem de programação que possui uma extensão, exemplo, arquivos de C++, possuem a extensão “nome.cpp”, ou arquivos Java Script, possuem a extensão “nome.js”.

Tabela: Simples estruturas de linhas e colunas que contém uma Pk.

Variável: Em programação de computadores, a variável é um espaço na memória para armazenar determinado tipo de dado.

Bibliografia

Documentação SQL (Microsoft) -> <https://docs.microsoft.com/pt-br/sql/?view=sql-server-ver15>

Visão Geral SQL (Microsoft) -> <https://docs.microsoft.com/pt-br/sql/relational-databases/databases/databases?view=sql-server-ver15>

Tipos de Dados (W3School) -> https://www.w3schools.com/sql/sql_datatypes.asp

Cadeia de Caracteres (Wikipedia) -> https://pt.wikipedia.org/wiki/Cadeia_de_caracteres

Guia de Referência (W3Schoolss) -> https://www.w3schools.com/sql/sql_quickref.asp

Palavras Reservadas (W3Schools) -> https://www.w3schools.com/sql/sql_ref_keywords.asp

Funções (W3School) -> https://www.w3schools.com/sql/sql_ref_sqlserver.asp

Base de Dados Ideia -> https://www.researchgate.net/figure/Product-categories-and-the-number-of-products-in-each-category_tbl1_24068280

Nomes em Inglês -> <https://www.ssa.gov/oact/babynames/decades/century.html>

Lucid Chart -> <https://www.lucidchart.com/pages/>

Base de Dados da Microsoft -> <https://docs.microsoft.com/pt-br/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms>

Modelagem de Dados (DevMedia)-> <https://www.devmedia.com.br/modelagem-de-dados-tutorial/20398>

Lista de Atividades SQLite -> <https://github.com/xGabrielR/SQL-Fundamentals/tree/main/Atividades/SQL%20%26%20SQLite%20Fundamentos>

Lista de Atividades SQL Server -> <https://github.com/xGabrielR/SQL-Fundamentals/tree/main/Atividades/SQL%20Server%20Fundamentos>

Quis SQL -> <https://www.opinionstage.com/c-t/sql-quiz>

Informações Adicionais

Lista de Atualizações:

Gabriel Richter. - *30/07/2021*

Gabriel Richter. - *09/08/2021*

Gabriel Richter. - *11/08/2021*

Gabriel Richter. - *12/08/2021*

Gabriel Richter. - *28/10/2021*

Gabriel Richter. - *19/11/2021*

Gabriel Richter. - *21/11/2021*