



**Estácio**

# Universidade Estácio de Sá

3º período em Desenvolvimento Full stack

Matrícula: 202302891292

Aluno: Luiz Fabrício Mello Ferreira

**Vamos manter as informações**

# 1º Procedimento | Criando o Banco de Dados

## Objetivo da Prática

Este projeto foi desenvolvido com o objetivo de aplicar, na prática, os conhecimentos teóricos adquiridos sobre modelagem de dados, utilizando um cenário fictício de uma loja. A estrutura abrange clientes, fornecedores, compradores e as operações de compra e venda, organizadas em tabelas com dados inseridos, acompanhadas pela elaboração de consultas SQL. A prática da modelagem e implementação de dados permitiu uma compreensão mais profunda dos conceitos abordados em sala de aula.

## Etapas do Projeto

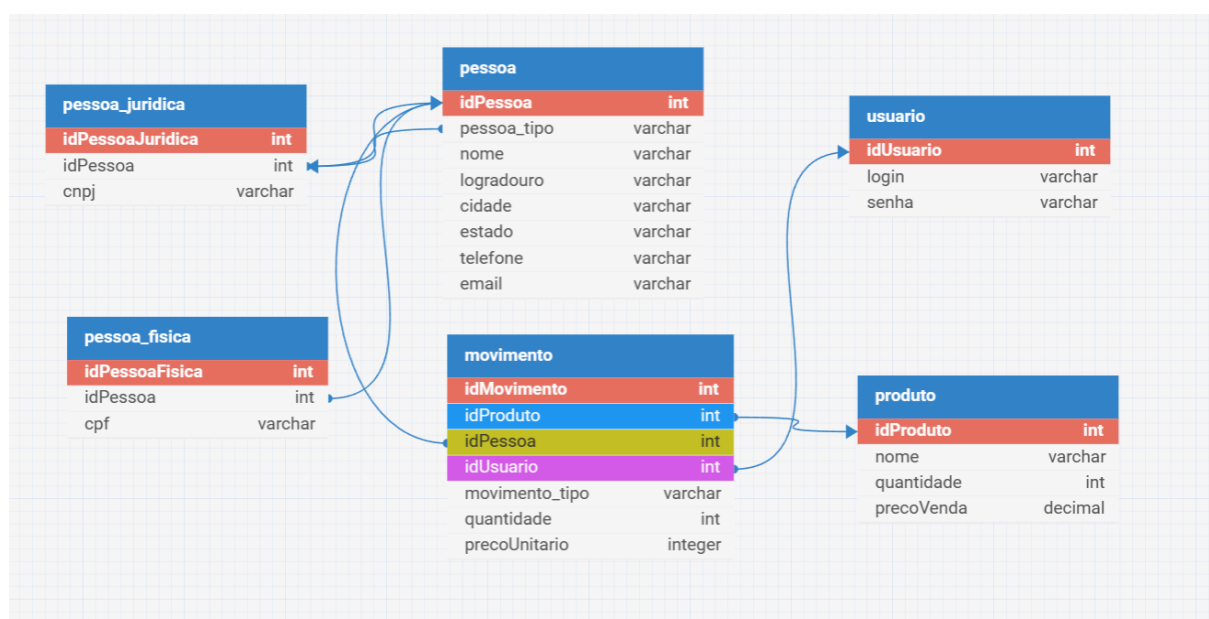
1. **Entendimento dos Requisitos:** O primeiro passo foi compreender as necessidades do sistema, incluindo os requisitos de dados e as expectativas dos usuários. Esse entendimento inicial foi fundamental para guiar a modelagem e o desenvolvimento do banco de dados.
2. **Modelagem e Estruturação do Banco de Dados:** Foi projetada a estrutura do banco, definindo tabelas, campos, chaves primárias e estrangeiras, e relacionamentos entre entidades como clientes, fornecedores, produtos e transações. Para isso, foi utilizada a ferramenta DB Designer Online, que permitiu uma modelagem visual das entidades, atributos e seus relacionamentos.
3. **Geração e Implementação do Esquema SQL:** Após a modelagem visual, a ferramenta gerou automaticamente o código SQL do esquema do banco, que foi adaptado e implementado no SQL Server Management Studio. Essa implementação incluiu a criação de tabelas e a inserção de dados fictícios, simulando um ambiente de operação real.
4. **Consultas SQL:** Foram elaboradas consultas SQL para extrair e analisar informações do banco de dados, desde dados específicos de clientes até movimentações de produtos. Essas consultas auxiliaram na validação do funcionamento do banco e na extração de insights relevantes.

## Metodologia

A metodologia utilizada visou à eficiência e à clareza no processo de criação e implementação do banco de dados. O uso do DB Designer Online para modelagem visual facilitou a validação do design conceitual antes da implementação. O SQL Server Management Studio foi então utilizado para adaptar o esquema gerado e realizar a implementação no servidor SQL.

## Considerações Finais

A modelagem de dados é uma etapa crucial no desenvolvimento de sistemas de banco de dados, pois garante que o design atenda aos requisitos do usuário, além de ser escalável e fácil de manter. A combinação de modelagem visual e implementação prática proporcionou uma transição coesa entre conceito e execução, resultando em um banco de dados funcional para o projeto fictício da loja.



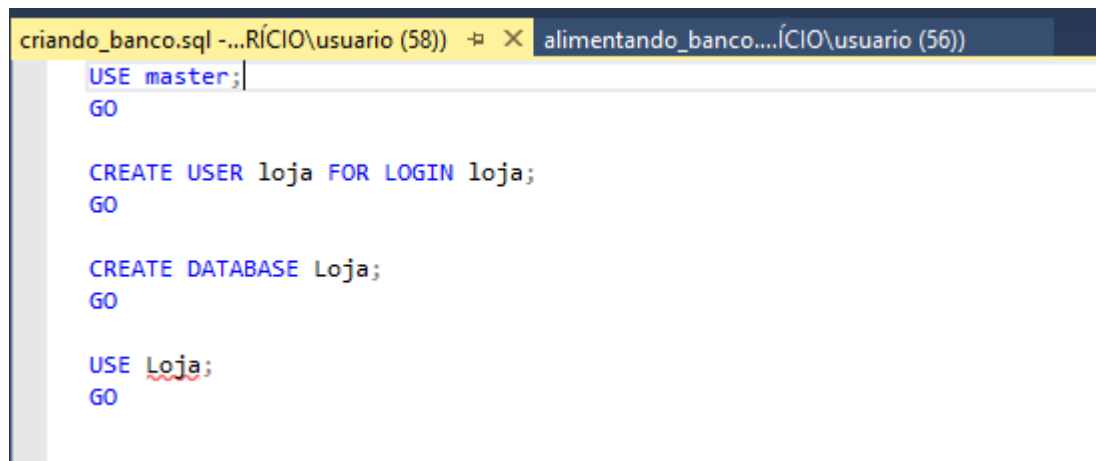
## Imagem 01 - Modelagem de dados

A imagem acima apresenta uma visão geral da modelagem de dados, destacando as entidades e seus principais atributos. Ela também ilustra as interligações entre os atributos das entidades, permitindo observar as dependências entre as tabelas.

Com base nessa modelagem, podemos identificar a presença de herança entre algumas tabelas — um conceito abordado na disciplina anterior. Por exemplo, as

tabelas **pessoa\_fisica** e **pessoa\_juridica** compartilham o atributo **idPessoa**, que também está presente na tabela usuário, representando a herança de informações entre essas entidades.

A modelagem evidencia as relações entre as tabelas, bem como as chaves primárias e estrangeiras. Com essas informações em mãos, damos início ao processo de criação das tabelas no banco de dados.



```
criando_banco.sql -...RÍCIO\usuario (58))  X  alimentando_banco....ÍCIO\usuario (56))
USE master;
GO

CREATE USER loja FOR LOGIN loja;
GO

CREATE DATABASE Loja;
GO

USE Loja;
GO
```

## Imagem 02 – Criação de usuário e banco de dados

Após a criação do banco de dados, iniciou-se a criação das tabelas **pessoa**, **pessoa\_fisica**, **pessoa\_juridica**, movimento, produto e usuário. A tabela **pessoa** armazena e herda as informações das tabelas **pessoa\_fisica** e **pessoa\_juridica**, consolidando os dados dessas duas entidades.

A tabela **produto** é responsável por registrar todas as informações dos produtos inseridos pelos usuários, enquanto a tabela **movimento** monitora todas as transações realizadas, como entradas e saídas de produtos.

```
CREATE TABLE pessoa(  
    idPessoa INT PRIMARY KEY IDENTITY(1,1) NOT NULL,  
    pessoa_tipo VARCHAR(100),  
    nome VARCHAR(255) NOT NULL,  
    logradouro VARCHAR(255) NOT NULL,  
    cidade VARCHAR(255) NOT NULL,  
    estado CHAR(2) NOT NULL,  
    telefone VARCHAR(11) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    CONSTRAINT check_tipo_pessoa CHECK (pessoa_tipo IN ('fisica', 'juridica'))  
)  
GO  
  
CREATE TABLE pessoa_fisica(  
    idPessoaFisica INT PRIMARY KEY IDENTITY(1,1) NOT NULL,  
    idPessoa INT NULL,  
    cpf VARCHAR(11) NOT NULL,  
    FOREIGN KEY (idPessoa) REFERENCES pessoa(idPessoa)  
)  
  
CREATE TABLE pessoa_juridica (  
    idPessoaJuridica INT PRIMARY KEY IDENTITY(1,1) NOT NULL,  
    idPessoa INT NULL,  
    cnpj VARCHAR(14) NOT NULL  
    FOREIGN KEY (idPessoa) REFERENCES pessoa(idPessoa)  
)
```

**Imagem 03 - Tabelas pessoa, pessoa\_fisica e pessoa\_juridica**

```

CREATE TABLE produto (
  idProduto INT PRIMARY KEY NOT NULL,
  nome VARCHAR(255) NOT NULL,
  quantidade INT NOT NULL,
  precoVenda DECIMAL(10,2) NOT NULL
)

CREATE TABLE movimento (
  idMovimento INT PRIMARY KEY NOT NULL,
  idProduto INT NOT NULL,
  idPessoa INT NOT NULL,
  idUsuario INT NOT NULL,
  movimento_tipo VARCHAR(100), -- 'E' para entrada, 'S' para saída
  quantidade INT NOT NULL,
  precoUnitario DECIMAL(10,2) NOT NULL,
  CONSTRAINT check_tipo_movimento CHECK (movimento_tipo IN ( 'entrada', 'saida')),
  FOREIGN KEY (idProduto) REFERENCES produto(idProduto),
  FOREIGN KEY (idPessoa) REFERENCES pessoa(idPessoa),
  FOREIGN KEY (idUsuario) REFERENCES usuario(idUsuario)
)

CREATE TABLE usuario (
  idUsuario INT UNIQUE NOT NULL,
  login VARCHAR(255) NOT NULL,
  senha VARCHAR(255) NOT NULL
)

```

#### Imagem 04 - Tabelas produto, movimento e usuario

Após a criação das tabelas, chegou o momento de inserir dados fictícios para preencher e dar vida ao banco de dados. Essa etapa simula o funcionamento de um sistema de loja, com seus usuários, produtos, fornecedores e compradores, criando um ambiente realista para testes. Abaixo estão os códigos utilizados para inserir os dados nas tabelas.

```

INSERT INTO [usuario](idUsuario, login, senha)
VALUES
(1, 'paulasantos', 'paula1234'),
(2, 'clebson', 'clebe123'),
(3, 'paloma', 'palo198'),
(4, 'cremilda', 'cre1234')

INSERT INTO [produto](idProduto, nome, quantidade, precoVenda)
VALUES
(1, 'Abacate', 100, 5.00),
(2, 'Laranja', 500, 2.00),
(3, 'Manga', 800, 4.00),
(4, 'Pera', 450, 9.00)

--Inserir pessoas--
INSERT INTO [pessoa] ( pessoa_tipo, nome, logradouro, cidade, estado, telefone, email)
VALUES ( 'fisica', 'Paula Gonçalves Dias', 'Rua Clara Castro', 'Rio de Janeiro', 'RJ', '11982377465', 'paula@email.com');

INSERT INTO [pessoa_fisica] (idPessoa, cpf) VALUES (1, 37570270726);

INSERT INTO [pessoa] ( pessoa_tipo, nome, logradouro, cidade, estado, telefone, email)
VALUES ( 'fisica', 'Clebson Albuquerque', 'Avenida Julho Maia 4983', 'Rio de Janeiro', 'RJ', '21934753764', 'clebson@email.com');

INSERT INTO [pessoa_fisica] (idPessoa, cpf) VALUES (2, 09384752932);

INSERT INTO [pessoa] ( pessoa_tipo, nome, logradouro, cidade, estado, telefone, email)
VALUES ( 'juridica', 'Paloma Costa', 'Rua Primeiro de Janeiro', 'Belo Horizonte', 'MG', '31976756383', 'paloma@email.com');

INSERT INTO [pessoa_juridica] (idPessoa, cnpj) VALUES (3, 85665739393404);

INSERT INTO [pessoa] (pessoa_tipo, nome, logradouro, cidade, estado, telefone, email)
VALUES ( 'juridica', 'Cremilda do Carmo', 'Rua Fonte da Arapuca', 'Porto Alegre', 'RS', '51975730803', 'cremilda@email.com');

INSERT INTO [pessoa_juridica] (idPessoa, cnpj) VALUES (4, 96783757389230);

-- Inserir movimentação fornecedor
INSERT INTO [movimento] (idMovimento, idProduto, idPessoa, idUsuario, movimento_tipo, quantidade, precoUnitario)
VALUES
(1, 1, 1, 1, 'entrada', 20, 4.5),
(2, 2, 2, 2, 'entrada', 15, 3.00),
(3, 1, 1, 1, 'saida', 10, 3.35),
(4, 1, 1, 1, 'entrada', 25, 4.50),
(5, 1, 1, 1, 'saida', 5, 2.50)

--Dados completos de pessoas físicas
SELECT pessoa.idPessoa, pessoa.nome, pessoa.logradouro, pessoa.cidade, pessoa.estado, pessoa.telefone, pessoa.email, pessoa_fisica.cpf
FROM pessoa INNER JOIN pessoa_fisica ON pessoa.idPessoa = pessoa_fisica.idPessoaFisica

--Dados completos de pessoas jurídica
SELECT pessoa.idPessoa, pessoa.nome, pessoa.logradouro, pessoa.cidade, pessoa.estado, pessoa.telefone, pessoa.email, pessoa_juridica.cnpj
FROM pessoa INNER JOIN pessoa_juridica ON pessoa.idPessoa = pessoa_juridica.idPessoaJuridica

--Movimentação de entrada
SELECT
P.nome AS Fornecedor, Prod.nome AS Produto, Mov.quantidade, Mov.precoUnitario AS PrecoUnitario,
Mov.quantidade * Mov.precoUnitario AS ValorTotal
FROM movimento Mov
JOIN Produto Prod ON Mov.idProduto = Prod.idProduto
JOIN Pessoa P ON Mov.idPessoa = P.idPessoa
WHERE movimento_tipo = 'entrada';

--Movimentação de saída
SELECT
P.nome AS Comprador, Prod.nome AS Produto, Mov.quantidade, Mov.precoUnitario AS PrecoUnitario,
Mov.quantidade * Mov.precoUnitario AS ValorTotal FROM Movimento Mov
JOIN Produto Prod ON Mov.idProduto = Prod.idProduto
JOIN Pessoa P ON Mov.idPessoa = P.idPessoa
WHERE movimento_tipo = 'saida';

```

```

--Valor total das entradas agrupadas por produto
SELECT Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) AS TotalEntradas
FROM Movimento Mov
WHERE movimento_tipo = 'entrada'
GROUP BY Mov.idProduto;

--Valor total das saidas agrupadas por produto
SELECT Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) AS TotalSaidas
FROM Movimento Mov
WHERE movimento_tipo = 'saida'
GROUP BY Mov.idProduto;

--Operadores que não efetuaram movimentações de entrada
SELECT DISTINCT U.*
FROM usuario U
LEFT JOIN Movimento M ON U.idUsuario = M.idUsuario AND movimento_tipo = 'entrada'
WHERE M.idUsuario IS NULL;

--Valor total de entrada, agrupado por operador
SELECT M.idUsuario, SUM(M.quantidade * M.precoUnitario) AS TotalEntradas FROM Movimento M
WHERE movimento_tipo = 'entrada'
GROUP BY M.idUsuario;

--Valor total de saída, agrupado por operador
SELECT M.idUsuario, SUM(M.quantidade * M.precoUnitario) AS TotalSaidas FROM Movimento M
WHERE movimento_tipo = 'saida'
GROUP BY M.idUsuario;

--Valor médio de venda por produto, utilizando média ponderada
SELECT
Prod.nome AS Produto, Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) / SUM(Mov.quantidade) AS MediaPonderada
FROM Movimento Mov
JOIN Produto Prod ON Mov.idProduto = Prod.idProduto
WHERE movimento_tipo = 'saida'
GROUP BY Mov.idProduto, Prod.nome;

```

**Imagem 04, 05 e 06 – A imagem mostra como inserir os dados fictícios nas tabelas e seus relacionamentos.**

Nessa etapa, utilizamos o comando **USE Loja** para garantir que estamos operando no banco de dados correto. Em seguida, começamos a inserir os dados dos usuários, incluindo seus IDs, logins e senhas, permitindo o acesso ao sistema da loja.

Depois, inserimos os produtos, detalhando as informações necessárias para que fornecedores e compradores possam realizar transações de compra e venda.

Em seguida, cadastramos as pessoas, classificando-as como pessoa física ou pessoa jurídica. Os compradores são registrados como pessoa jurídica, enquanto os vendedores são cadastrados como pessoa física.

Por fim, na tabela de movimentação, armazenamos todas as transações realizadas, registrando o relacionamento entre produtos, usuários e pessoas, bem como a quantidade de produtos e o preço unitário de cada transação.



```

--Dados completos de pessoas físicas
SELECT pessoa.idPessoa, pessoa.nome, pessoa.logradouro, pessoa.cidade, pessoa.estado, pessoa.telefone, pessoa.email, pessoa_fisica.cpf
FROM pessoa INNER JOIN pessoa_fisica ON pessoa.idPessoa = pessoa_fisica.idPessoaFisica

--Dados completos de pessoas jurídicas
SELECT pessoa.idPessoa, pessoa.nome, pessoa.logradouro, pessoa.cidade, pessoa.estado, pessoa.telefone, pessoa.email, pessoa_juridica.cnpj
FROM pessoa INNER JOIN pessoa_juridica ON pessoa.idPessoa = pessoa_juridica.idPessoaJuridica

--Movimentação de entrada
SELECT
P.nome AS Fornecedor, Prod.nome AS Produto, Mov.quantidade, Mov.precoUnitario AS PrecoUnitario,
Mov.quantidade * Mov.precoUnitario AS ValorTotal
FROM movimento Mov
JOIN Produto Prod ON Mov.idProduto = Prod.idProduto
JOIN Pessoa P ON Mov.idPessoa = P.idPessoa
WHERE movimento_tipo = 'entrada';

--Movimentação de saída
SELECT
P.nome AS Comprador, Prod.nome AS Produto, Mov.quantidade, Mov.precoUnitario AS PrecoUnitario,
Mov.quantidade * Mov.precoUnitario AS ValorTotal FROM Movimento Mov
JOIN Produto Prod ON Mov.idProduto = Prod.idProduto
JOIN Pessoa P ON Mov.idPessoa = P.idPessoa
WHERE movimento_tipo = 'saida';

--Valor total das entradas agrupadas por produto
SELECT Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) AS TotalEntradas
FROM Movimento Mov
WHERE movimento_tipo = 'entrada'
GROUP BY Mov.idProduto;

--Valor total das saídas agrupadas por produto
SELECT Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) AS TotalSaídas
FROM Movimento Mov
WHERE movimento_tipo = 'saida'
GROUP BY Mov.idProduto;

--Operadores que não efetuaram movimentações de entrada
SELECT DISTINCT U.[
FROM usuario U
LEFT JOIN Movimento M ON U.idUsuario = M.idUsuario AND movimento_tipo = 'entrada'
WHERE M.idUsuario IS NULL;

--Valor total de entrada, agrupado por operador
SELECT M.idUsuario, SUM(M.quantidade * M.precoUnitario) AS TotalEntradas FROM Movimento M
WHERE movimento_tipo = 'entrada'
GROUP BY M.idUsuario;

--Valor total de saída, agrupado por operador
SELECT M.idUsuario, SUM(M.quantidade * M.precoUnitario) AS TotalSaídas FROM Movimento M
WHERE movimento_tipo = 'saida'
GROUP BY M.idUsuario;

--Valor médio de venda por produto, utilizando média ponderada
SELECT
Prod.nome AS Produto, Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) / SUM(Mov.quantidade) AS MediaPonderada
FROM Movimento Mov
JOIN Produto Prod ON Mov.idProduto = Prod.idProduto
WHERE movimento_tipo = 'saida'
GROUP BY Mov.idProduto, Prod.nome;

```

Ativar o Windows

## Imagem 07 - Consultas das tabelas

Resultados		Mensagens						
	idPessoa	nome	logradouro	cidade	estado	telefone	email	cpf
1	1	Silvia Santos Conceição	Rua Jubiramba Castro	São Paulo	SP	11982377465	silvia@email.com	37570270726
2	2	Judiciara Albuquerque	Avenida Julho Maia ...	Rio de J...	RJ	21934753764	judiciara@emai...	9384752932
	idPessoa	nome	logradouro	cidade	estado	telefone	email	cnpj
1	1	Silvia Santos Conceição	Rua Jubiramba Castro	São Paulo	SP	11982377465	silvia@email.com	85665739393404
2	2	Judiciara Albuquerque	Avenida Julho Maia 4983	Rio de Janeiro	RJ	21934753764	judiciara@email.com	96783757389230
	Fornecedor	Produto	quantidade	PrecoUnitario	ValorTotal			
1	Silvia Santos Conceição	Banana	20	4.50	90.00			
2	Judiciara Albuquerque	Laranja	15	3.00	45.00			
3	Silvia Santos Conceição	Banana	25	4.50	112.50			
	Comprador	Produto	quantidade	PrecoUnitario	ValorTotal			
1	Silvia Santos Conceição	Banana	10	3.35	33.50			
2	Silvia Santos Conceição	Banana	5	2.50	12.50			
	idProduto	TotalEntradas						
1	1	202.50						
2	2	45.00						
	idProduto	TotalSaidas						
1	1	46.00						
	idUsuario	login	senha					
1	3	luciana242	lulu2345					
2	4	rosenilda44678	rose3124					
	idUsuario	TotalEntradas						
1	1	202.50						
2	2	45.00						
	idUsuario	TotalSaidas						
1	1	46.00						
	Produto	idProduto	MediaPonderada					
1	Banana	1	3.066666					

## Imagem 08 - Resultado das consultas

### 1. Como são implementadas as diferentes cardinalidades, como 1X1, 1XN ou NxN, em um banco de dados relacional?

a) 1 para 1 (1X1): Geralmente, isso é feito utilizando uma chave estrangeira em uma das tabelas para apontar para a chave primária da outra tabela. Por exemplo, uma tabela de "Pessoa" pode ter uma chave primária única, e a tabela "Endereço" pode ter uma chave estrangeira que referencia essa chave primária.

b) 1 para N (1XN): É implementado de forma semelhante ao 1X1, mas várias linhas da tabela relacionada podem se referir à mesma linha da tabela principal. Por exemplo, uma tabela de "Departamento" pode ter várias linhas na tabela "Funcionário", todas apontando para o mesmo departamento. c) N para N (NXN): Para esse tipo de relação, é criada uma tabela intermediária que faz a associação entre as chaves primárias das duas tabelas envolvidas. Um exemplo seria o relacionamento entre "Estudante" e "Curso", onde uma tabela de junção indicaria quais alunos estão inscritos em quais cursos.

## **2. Que tipo de relacionamento deve ser usado para representar herança em bancos de dados relacionais?**

Uma abordagem comum para representar herança em bancos de dados relacionais é criar uma tabela para cada classe concreta e uma tabela separada para a classe base. Cada tabela de classe concreta teria uma chave estrangeira referenciando a tabela da classe base. Isso é conhecido como modelagem de herança por tabela. Por exemplo, em uma hierarquia de classes onde "Veículo" é a classe base e "Carro" e "Moto" são classes derivadas, haveria uma tabela "Veículo" e tabelas específicas para "Carro" e "Moto", cada uma referenciando a tabela "Veículo".

## **3. Como o SQL Server Management Studio (SSMS) ajuda a melhorar a produtividade nas tarefas relacionadas ao gerenciamento de banco de dados?**

O SSMS oferece diversos recursos que otimizam o trabalho com bancos de dados SQL Server:

- a) Interface amigável e fácil de usar para administração de banco de dados.
- b) Ferramentas para editar esquemas de banco de dados, como criação de tabelas, índices e procedimentos armazenados, seja via GUI ou através de scripts.
- c) Ferramentas avançadas de consulta SQL, com destaque de sintaxe, sugestões de código, execução de consultas e exibição dos resultados.
- d) Capacidades para monitorar e otimizar o desempenho do banco de dados por meio de ferramentas de diagnóstico e perfil.
- e) Funcionalidades de segurança que permitem o gerenciamento de usuários, permissões e auditoria.
- f) Integração com outras ferramentas da Microsoft, como Azure e Power BI.

## **4. Quais são as diferenças entre sequence e identity?**

- a) Sequence: Um sequence é um objeto que gera uma sequência de números exclusivos de acordo com parâmetros definidos pelo usuário. Ele pode ser utilizado

em várias tabelas e em diferentes colunas de uma tabela. No entanto, o controle de acesso e a manutenção do valor atual da sequência são responsabilidade do usuário.

b) Identity: O identity é uma propriedade de uma coluna que gera automaticamente valores exclusivos. É mais simples de utilizar que um sequence, pois o banco de dados cuida da geração e do controle dos valores automaticamente.

## **5. Qual a importância das chaves estrangeiras para a consistência do banco de dados?**

As chaves estrangeiras são essenciais para garantir a integridade referencial e a consistência dos dados em um banco de dados relacional. Elas estabelecem relacionamentos entre tabelas, garantindo que cada valor em uma coluna de chave estrangeira corresponda a um valor existente na coluna de chave primária de outra tabela. Isso previne referências a dados inexistentes e assegura a integridade durante inserções, atualizações e exclusões.

## **6. Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?**

a) Álgebra Relacional: Alguns dos operadores da álgebra relacional incluem projeção, seleção, união, interseção, diferença, produto cartesiano e junção.

b) Cálculo Relacional: No cálculo relacional, os operadores são geralmente descritos como predicados ou expressões que definem conjuntos de tuplas. Operadores comuns incluem seleção, projeção, união, interseção e diferença.

## **7. Como funciona o agrupamento em consultas e qual é o requisito obrigatório?**

O agrupamento em consultas é realizado através da cláusula GROUP BY, que agrupa as linhas com base nos valores de uma ou mais colunas. O requisito obrigatório ao usar o GROUP BY é que todas as colunas selecionadas que não fazem parte de uma função de agregação precisam estar incluídas na cláusula

GROUP BY. Isso assegura que o SQL agrupe corretamente as linhas e calcule funções de agregação como SUM, COUNT, AVG, entre outras.

Abaixo, segue o link do repositório no Github de todo projeto: