



Estácio

FACULDADE ESTÁCIO

CÂMPUS VOLTA REDONDA – RJ

DESENVOLVIMENTO FULL STACK

DISCIPLINA – VAMOS MANTER AS INFORMAÇÕES?

TURMA – 2023.2

SEMESTRE – 3

VOLTA REDONDA, JULHO 2024.

DESENVOLVIMENTO FULL STACK

DISCIPLINA – VAMOS MANTER AS INFORMAÇÕES?

TURMA – 2023.2

SEMESTRE – 3

ALUNO – BRUNO SAMPAIO BASTOS

TUTOR – JOSYANE SOUZA

GITHUB - <https://github.com/BrunoTI-Code?tab=repositories>

VOLTA REDONDA, JULHO 2024.

1 TÍTULO DA PRÁTICA

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

2 Objetivo da Prática;

2.1 Identificar os requisitos de um sistema e transformá-los no modelo adequado.

2.2 Utilizar ferramentas de modelagem para bases de dados relacionais.

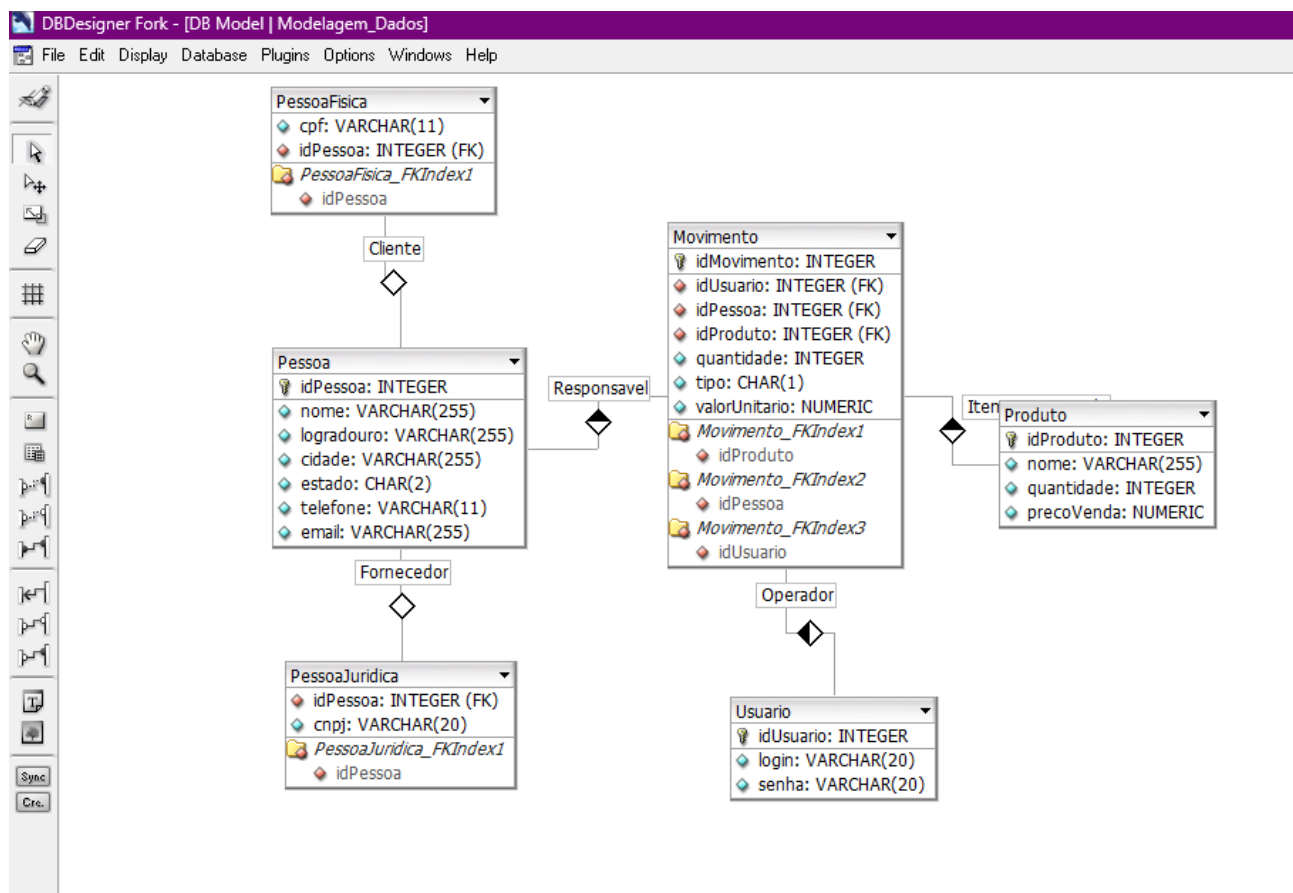
2.3 Explorar a sintaxe SQL na criação das estruturas do banco (DDL).

2.4 Explorar a sintaxe SQL na consulta e manipulação de dados (DML)

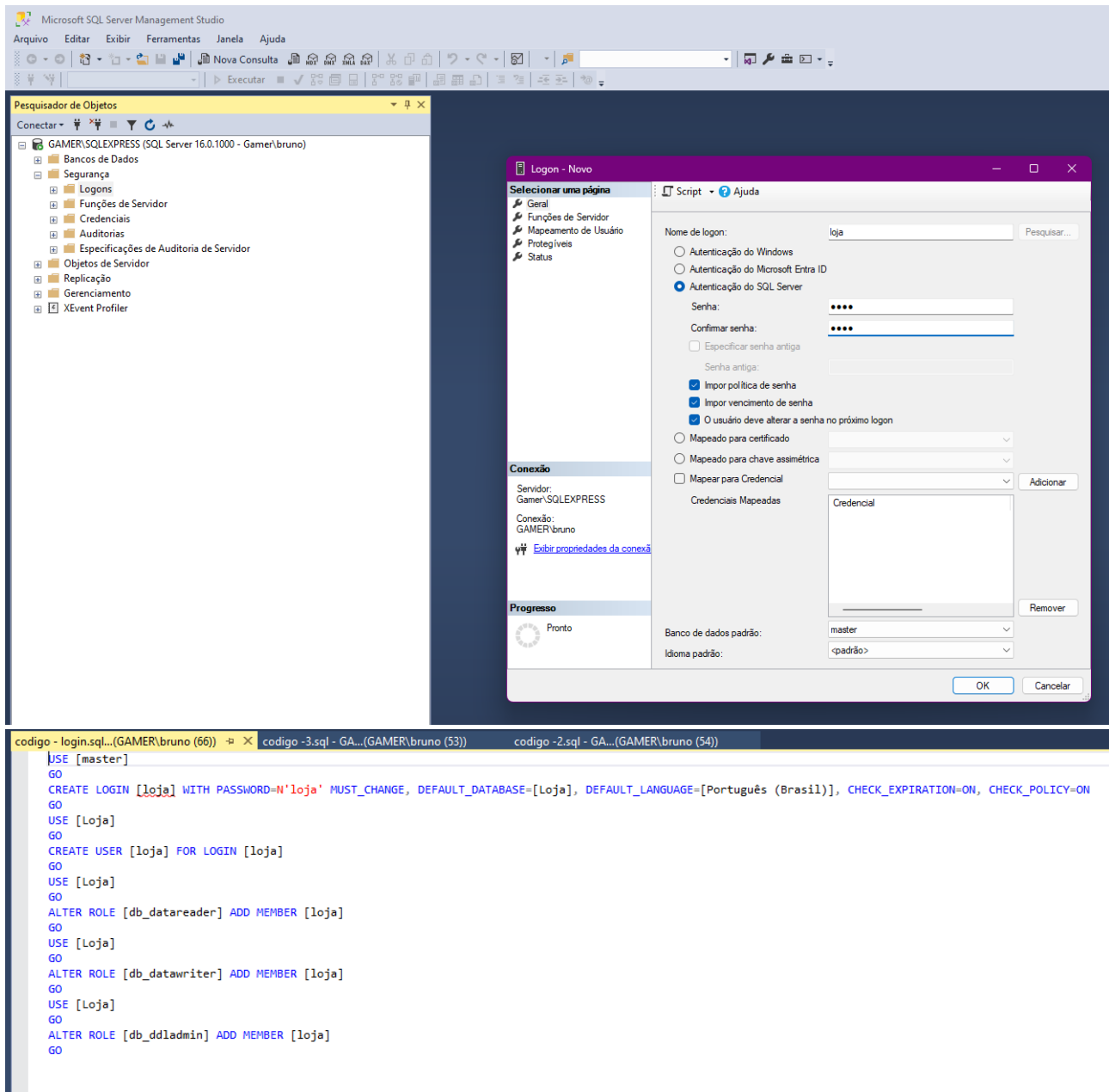
2.5 No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

3 TODOS OS CÓDIGOS SOLICITADOS NESTE ROTEIRO DE AULA;

3.1 DB-FORK



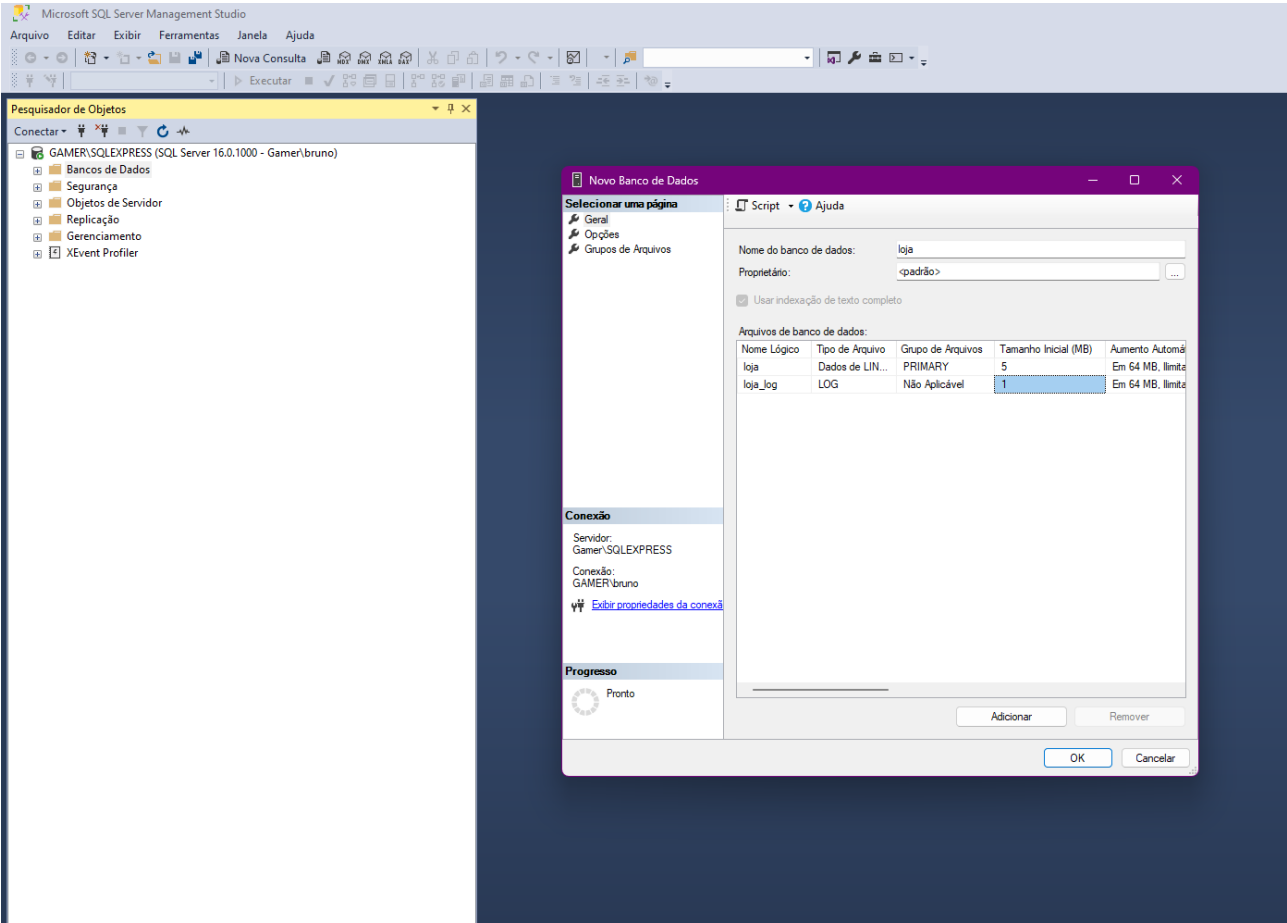
3.2 CRIANDO E CONFIGURANDO O LOGIN NO SQL



The screenshot displays the Microsoft SQL Server Management Studio interface. On the left, the 'Pesquisador de Objetos' (Object Explorer) shows the server structure for 'GAMER\SQLEXPRESS'. The 'Segurança' (Security) folder is expanded, showing 'Logons' (Logins). A 'Logon - Novo' (New Login) dialog box is open in the foreground. The 'Selecionar uma página' (Select a page) tab is active, showing the 'Geral' (General) page. The 'Nome de logon' (Login name) is set to 'loja'. The 'Autenticação do SQL Server' (SQL Server Authentication) radio button is selected. The 'Senha' (Password) and 'Confirmar senha' (Confirm password) fields are filled with masked characters. The 'Especificar senha antiga' (Specify old password) checkbox is unchecked. The 'Senha antiga' (Old password) field is empty. The 'Importar política de senha' (Import password policy), 'Importar vencimento de senha' (Import password expiration), and 'O usuário deve alterar a senha no próximo logon' (User must change password at next login) checkboxes are checked. The 'Mapeado para certificado' (Mapped to certificate) and 'Mapeado para chave assimétrica' (Mapped to asymmetric key) radio buttons are unchecked. The 'Mapear para Credencial' (Map to Credential) checkbox is unchecked. The 'Credenciais Mapeadas' (Mapped Credentials) list is empty. The 'Banco de dados padrão' (Default database) is set to 'master' and the 'Idioma padrão' (Default language) is set to '<padrão>'. The 'Progresso' (Progress) bar shows 'Pronto' (Ready). Below the dialog box, the SQL script in the query editor is visible:

```
USE [master]
GO
CREATE LOGIN [loja] WITH PASSWORD=N'loja' MUST_CHANGE, DEFAULT_DATABASE=[Loja], DEFAULT_LANGUAGE=[Português (Brasil)], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
GO
USE [Loja]
GO
CREATE USER [loja] FOR LOGIN [loja]
GO
USE [Loja]
GO
ALTER ROLE [db_datareader] ADD MEMBER [loja]
GO
USE [Loja]
GO
ALTER ROLE [db_datawriter] ADD MEMBER [loja]
GO
USE [Loja]
GO
ALTER ROLE [db_ddladmin] ADD MEMBER [loja]
GO
```

3.3 CRIANDO O BANCO DE DADOS COM O NOME “LOJA”




```
Solution1
SQLQuery1.sql - G... (GAMER\bruno (70)) - X

USE [master]
GO

/***** Object: Database [Loja]    Script Date: 31/07/2024 19:46:54 *****/
CREATE DATABASE [Loja]
    CONTAINMENT = NONE
    ON PRIMARY
    ( NAME = N'Loja', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Loja.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )
    LOG ON
    ( NAME = N'Loja_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Loja_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )
    WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
GO

ALTER DATABASE [Loja] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [Loja] SET ANSI_NULLS OFF
GO

ALTER DATABASE [Loja] SET ANSI_PADDING OFF
GO

ALTER DATABASE [Loja] SET ANSI_WARNINGS OFF
GO

ALTER DATABASE [Loja] SET ARITHABORT OFF
GO

ALTER DATABASE [Loja] SET AUTO_CLOSE ON
GO

ALTER DATABASE [Loja] SET AUTO_SHRINK OFF
GO

ALTER DATABASE [Loja] SET AUTO_UPDATE_STATISTICS ON
GO

ALTER DATABASE [Loja] SET CURSOR_CLOSE_ON_COMMIT OFF
GO

ALTER DATABASE [Loja] SET CURSOR_DEFAULT GLOBAL
GO

ALTER DATABASE [Loja] SET CONCAT_NULL_YIELDS_NULL OFF
GO

ALTER DATABASE [Loja] SET NUMERIC_ROUNDABORT OFF
GO

ALTER DATABASE [Loja] SET QUOTED_IDENTIFIER OFF
GO

ALTER DATABASE [Loja] SET RECURSIVE_TRIGGERS OFF
GO

ALTER DATABASE [Loja] SET RECURSIVE_TRIGGERS OFF
GO

ALTER DATABASE [Loja] SET ENABLE_BROKER
GO

ALTER DATABASE [Loja] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO

ALTER DATABASE [Loja] SET DATE_CORRELATION_OPTIMIZATION OFF
GO

ALTER DATABASE [Loja] SET TRUSTWORTHY OFF
GO

ALTER DATABASE [Loja] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO

ALTER DATABASE [Loja] SET PARAMETERIZATION SIMPLE
GO

ALTER DATABASE [Loja] SET READ_COMMITTED_SNAPSHOT OFF
GO

ALTER DATABASE [Loja] SET HONOR_BROKER_PRIORITY OFF
GO

ALTER DATABASE [Loja] SET RECOVERY SIMPLE
GO

ALTER DATABASE [Loja] SET MULTI_USER
GO

ALTER DATABASE [Loja] SET PAGE_VERIFY CHECKSUM
GO

ALTER DATABASE [Loja] SET DB_CHAINING OFF
GO

ALTER DATABASE [Loja] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO

ALTER DATABASE [Loja] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO

ALTER DATABASE [Loja] SET DELAYED_DURABILITY = DISABLED
GO

ALTER DATABASE [Loja] SET ACCELERATED_DATABASE_RECOVERY = OFF
GO

ALTER DATABASE [Loja] SET QUERY_STORE = ON
GO

ALTER DATABASE [Loja] SET QUERY_STORE ( OPERATION_MODE = READ_WRITE, CLEANUP_POLICY = ( STALE_QUERY_THRESHOLD_DAYS = 30 ), DATA_FLUSH_INTERVAL_SECONDS = 900, INTERVAL_LENGTH_MINUTES = 60, MAX_STORAGE_SIZE_MB = 1000, QUERY_CAPTURE_MODE = AUTO, SIZE_BASED_CLEANUP_MODE
```

ALTER DATABASE [Loja] SET READ_WRITE
GO

3.4 CÓDIGO COMPLETO DE CRIAÇÃO DO BANCO DE DADOS JUNTAMENTE COM AS TABELAS

```
Solution1
codigo -1.sql - GA...(GAMER\bruno (52)) -P X

CREATE DATABASE Loja;
GO

USE Loja;
GO

CREATE SEQUENCE ordemPessoaId
START WITH 1
INCREMENT BY 1;

CREATE TABLE Pessoa (
    idPessoa INTEGER NOT NULL CONSTRAINT PK_Pessoa PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    logradouro VARCHAR(255),
    cidade VARCHAR(255),
    estado CHAR(2) NOT NULL,
    telefone VARCHAR(11),
    email VARCHAR(255)
);
GO

CREATE TABLE PessoaFisica (
    idPessoa INTEGER NOT NULL CONSTRAINT PK_PessoaFisica PRIMARY KEY,
    cpf VARCHAR(11) NOT NULL,
    CONSTRAINT FK_PessoaFisica_Pessoa FOREIGN KEY (idPessoa) REFERENCES Pessoa (idPessoa)
);
GO

CREATE TABLE PessoaJuridica (
    idPessoa INTEGER NOT NULL CONSTRAINT PK_PessoaJuridica PRIMARY KEY,
    cnpj VARCHAR(14) NOT NULL,
    CONSTRAINT FK_PessoaJuridica_Pessoa FOREIGN KEY (idPessoa) REFERENCES Pessoa (idPessoa)
);
GO

CREATE TABLE Usuario (
    idUsuario INTEGER NOT NULL CONSTRAINT PK_Usuario PRIMARY KEY IDENTITY,
    login VARCHAR(20) NOT NULL,
    senha VARCHAR(20) NOT NULL
);
GO


CREATE TABLE Produto (
    idProduto INTEGER NOT NULL CONSTRAINT PK_Produto PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    quantidade INTEGER,
    precoVenda NUMERIC(5, 2)
);
GO


CREATE TABLE Movimento (
    idMovimento INTEGER NOT NULL CONSTRAINT PK_Movimento PRIMARY KEY,
    idUsuario INTEGER NOT NULL,
    idPessoa INTEGER NOT NULL,
    idProduto INTEGER,
    quantidade INTEGER,
    tipo CHAR(1),
    valorUnitario NUMERIC (5, 2),


```


Pesquisador de Objetos

Conectar ▾      


[-]  GAMER\SQLEXPRESS (SQL Server 16.0.1000 - Gamer\bruno)


[-]  Bancos de Dados


+  Bancos de Dados do Sistema

+  Instantâneos do Banco de Dados


[-]  Loja


+  Diagramas de Banco de Dados


[-]  Tabelas


+  Tabelas do Sistema


+  FileTables


+  Tabelas Externas


+  Tabelas de Grafo


+  dbo.Movimento


+  dbo.Pessoa


+  dbo.PessoaFisica


+  dbo.PessoaJuridica


+  dbo.Produto


+  dbo.Usuario


+  Exibições


+  Recursos Externos


+  Sinônimos


+  Programação


+  Repositório de Consultas


+  Service Broker


+  Armazenamento


+  Segurança

+  Segurança

+  Objetos de Servidor

+  Replicação

+  Gerenciamento

+  XEvent Profiler

MOVIMENTO

```
Solution1
SQLQuery15.sql -... (GAMER\bruno (65)) -> X

USE [Loja]
GO

/***** Object: Table [dbo].[Movimento]    Script Date: 31/07/2024 20:08:03 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Movimento](
    [idMovimento] [int] NOT NULL,
    [idUserario] [int] NOT NULL,
    [idPessoa] [int] NOT NULL,
    [idProduto] [int] NULL,
    [quantidade] [int] NULL,
    [tipo] [char](1) NULL,
    [valorUnitario] [numeric](5, 2) NULL,
    CONSTRAINT [PK_Movimento] PRIMARY KEY CLUSTERED
(
    [idMovimento] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Movimento] WITH CHECK ADD CONSTRAINT [FK_Movimento_Pessoa] FOREIGN KEY([idPessoa])
REFERENCES [dbo].[Pessoa] ([idPessoa])
GO

ALTER TABLE [dbo].[Movimento] CHECK CONSTRAINT [FK_Movimento_Pessoa]
GO

ALTER TABLE [dbo].[Movimento] WITH CHECK ADD CONSTRAINT [FK_Movimento_Produto] FOREIGN KEY([idProduto])
REFERENCES [dbo].[Produto] ([idProduto])
GO

ALTER TABLE [dbo].[Movimento] CHECK CONSTRAINT [FK_Movimento_Produto]
GO

ALTER TABLE [dbo].[Movimento] WITH CHECK ADD CONSTRAINT [FK_Movimento_Usuario] FOREIGN KEY([idUserario])
REFERENCES [dbo].[Usuario] ([idUserario])
GO

ALTER TABLE [dbo].[Movimento] CHECK CONSTRAINT [FK_Movimento_Usuario]
GO
```

PESSOA

```
SQLQuery16.sql -... (GAMER\bruno (62)) - X
USE [Loja]
GO

/***** Object: Table [dbo].[Pessoa]    Script Date: 31/07/2024 20:09:14 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Pessoa](
    [idPessoa] [int] NOT NULL,
    [nome] [varchar](255) NOT NULL,
    [logradouro] [varchar](255) NULL,
    [cidade] [varchar](255) NULL,
    [estado] [char](2) NOT NULL,
    [telefone] [varchar](11) NULL,
    [email] [varchar](255) NULL,
    CONSTRAINT [PK_Pessoa] PRIMARY KEY CLUSTERED
(
    [idPessoa] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

PESSOA FISICA

```
SQLQuery17.sql -... (GAMER\bruno (52)) - X
USE [Loja]
GO

/***** Object: Table [dbo].[PessoaFisica]    Script Date: 31/07/2024 20:10:23 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PessoaFisica](
    [idPessoa] [int] NOT NULL,
    [cpf] [varchar](11) NOT NULL,
    CONSTRAINT [PK_PessoaFisica] PRIMARY KEY CLUSTERED
(
    [idPessoa] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PessoaFisica] WITH CHECK ADD CONSTRAINT [FK_PessoaFisica_Pessoa] FOREIGN KEY([idPessoa])
REFERENCES [dbo].[Pessoa] ([idPessoa])
GO

ALTER TABLE [dbo].[PessoaFisica] CHECK CONSTRAINT [FK_PessoaFisica_Pessoa]
GO
```

PESSOA JURIDICA

```
SQLQuery18.sql -... (GAMER\bruno (62)) - X
USE [Loja]
GO

/***** Object: Table [dbo].[PessoaJuridica]    Script Date: 31/07/2024 20:11:16 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PessoaJuridica](
    [idPessoa] [int] NOT NULL,
    [cnpj] [varchar](14) NOT NULL,
    CONSTRAINT [PK_PessoaJuridica] PRIMARY KEY CLUSTERED
(
    [idPessoa] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PessoaJuridica] WITH CHECK ADD CONSTRAINT [FK_PessoaJuridica_Pessoa] FOREIGN KEY([idPessoa])
REFERENCES [dbo].[Pessoa] ([idPessoa])
GO

ALTER TABLE [dbo].[PessoaJuridica] CHECK CONSTRAINT [FK_PessoaJuridica_Pessoa]
GO
```

PRODUTO

SQLQuery21.sql - ... (GAMER\bruno (62))

```
USE [Loja]
GO

/***** Object: Table [dbo].[Produto]    Script Date: 31/07/2024 20:12:19 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Produto](
    [idProduto] [int] NOT NULL,
    [nome] [varchar](255) NOT NULL,
    [quantidade] [int] NULL,
    [precoVenda] [numeric](5, 2) NULL,
    CONSTRAINT [PK_Produto] PRIMARY KEY CLUSTERED
(
    [idProduto] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

USUARIO

SQLQuery22.sql - ... (GAMER\bruno (62))

```
USE [Loja]
GO

/***** Object: Table [dbo].[Usuario]    Script Date: 31/07/2024 20:12:54 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Usuario](
    [idUsuario] [int] IDENTITY(1,1) NOT NULL,
    [login] [varchar](20) NOT NULL,
    [senha] [varchar](20) NOT NULL,
    CONSTRAINT [PK_Usuario] PRIMARY KEY CLUSTERED
(
    [idUsuario] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

4 ANÁLISE E CONCLUSÃO:

4.1 COMO SÃO IMPLEMENTADAS AS DIFERENTES CARDINALIDADES, BASICAMENTE 1X1, 1XN OU NXN, EM UM BANCO DE DADOS RELACIONAL?

Em um banco de dados relacional, as cardinalidades são implementadas através de chaves primárias e chaves estrangeiras. Aqui está uma explicação básica de como cada tipo é tratado:

1x1 (Um para Um)

Descrição: Cada registro em uma tabela está associado a exatamente um registro em outra tabela. Implementação:

Chaves Primárias: Cada tabela possui uma chave primária única.

Chaves Estrangeiras: Uma das tabelas terá uma chave estrangeira que faz referência à chave primária da outra tabela e é definida como única. Isso garante que cada valor na chave estrangeira apareça uma única vez, mantendo o relacionamento um-para-um.

1xN (Um para Muitos)

Descrição: Um registro em uma tabela pode estar associado a múltiplos registros em outra tabela. Implementação:

Chaves Primárias: A tabela "um" possui uma chave primária única.

Chaves Estrangeiras: A tabela "muitos" possui uma chave estrangeira que faz referência à chave primária da tabela "um". Esta chave estrangeira não precisa ser única, permitindo que múltiplos registros na tabela "muitos" estejam associados a um único registro na tabela "um".

NxN (Muitos para Muitos)

Descrição: Múltiplos registros em uma tabela podem estar associados a múltiplos registros em outra tabela. Implementação:

Tabelas de Associação: Para implementar essa cardinalidade, é necessária uma tabela intermediária, conhecida como tabela de associação.

Chaves Primárias nas Tabelas Principais: Cada uma das tabelas principais possui uma chave primária única.

Chaves Estrangeiras na Tabela de Associação: A tabela de associação possui duas chaves estrangeiras, cada uma referenciando a chave primária de uma das tabelas principais. A combinação dessas duas chaves estrangeiras é única, garantindo que cada par de associações seja único e gerencie corretamente o relacionamento muitos-para-muitos.

4.2 QUE TIPO DE RELACIONAMENTO DEVE SER UTILIZADO PARA REPRESENTAR O USO DE HERANÇA EM BANCOS DE DADOS RELACIONAIS?

Para representar o uso de herança em bancos de dados relacionais, você pode utilizar dois principais tipos de relacionamento: a abordagem de "tabela única" e a abordagem de "tabelas separadas".

Tabela Única

Descrição: Todas as classes da hierarquia de herança são representadas em uma única tabela.

Implementação:

Uma tabela contém colunas para todos os campos de todas as classes na hierarquia.

Uma coluna adicional é usada para discriminar o tipo de cada registro (tipo de classe).

Pode resultar em muitas colunas nulas, pois nem todas as colunas serão relevantes para todas as classes.

Tabelas Separadas

Descrição: Cada classe na hierarquia de herança é representada por uma tabela separada.

Implementação:

Há uma tabela para a classe base e tabelas adicionais para cada subclasse.

A tabela da subclasse tem uma chave estrangeira que referência a chave primária da tabela da classe base.

4.3 COMO O SQL SERVER MANAGEMENT STUDIO PERMITE A MELHORIA DA PRODUTIVIDADE NAS TAREFAS RELACIONADAS AO GERENCIAMENTO DO BANCO DE DADOS?

O SQL Server Management Studio melhora a produtividade no gerenciamento de bancos de dados através de diversas funcionalidades:

Interface Gráfica Intuitiva: Facilita a navegação e a manipulação de objetos do banco de dados sem a necessidade de escrever código SQL manualmente.

Ferramentas de Administração e Monitoramento: Inclui SQL Server Profiler, Activity Monitor e Performance Dashboard para monitorar e administrar o desempenho do banco de dados.

IntelliSense e Autocomplete: Reduz erros de sintaxe e acelera a escrita de código SQL.

Suporte a Scripts e Automação: Permite a criação, execução e agendamento de scripts T-SQL, automatizando tarefas repetitivas.

Ferramentas de Design e Diagramas: Facilita o design e a documentação da estrutura do banco de dados.

Gerenciamento de Segurança: Simplifica a criação e gestão de logins, usuários e permissões.

Ferramentas de Backup e Restauração: Simplifica o processo de backup e restauração de bancos de dados.

Integração com Outras Ferramentas e Serviços: Suporte para integração com Azure SQL Database e outros serviços em nuvem.

Customização e Extensibilidade: Permite a personalização da interface e a instalação de extensões.

Documentação e Suporte: Oferece ampla documentação e suporte da comunidade para resolução de problemas e aprendizado.

5 2º PROCEDIMENTO | ALIMENTANDO A BASE

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

6 Objetivo da Prática 2;

2.1 Identificar os requisitos de um sistema e transformá-los no modelo adequado.

2.2 Utilizar ferramentas de modelagem para bases de dados relacionais.

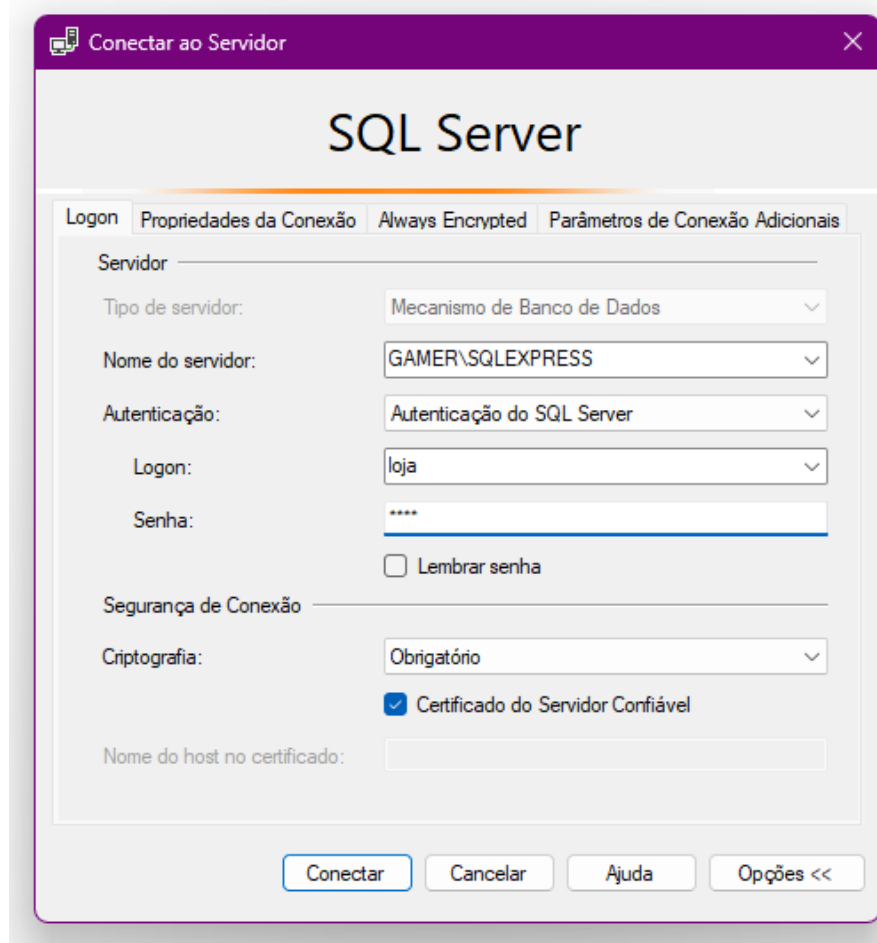
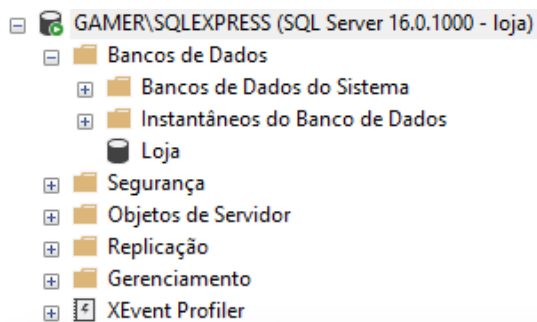
2.3 Explorar a sintaxe SQL na criação das estruturas do banco (DDL).

2.4 Explorar a sintaxe SQL na consulta e manipulação de dados (DML)

2.5 No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

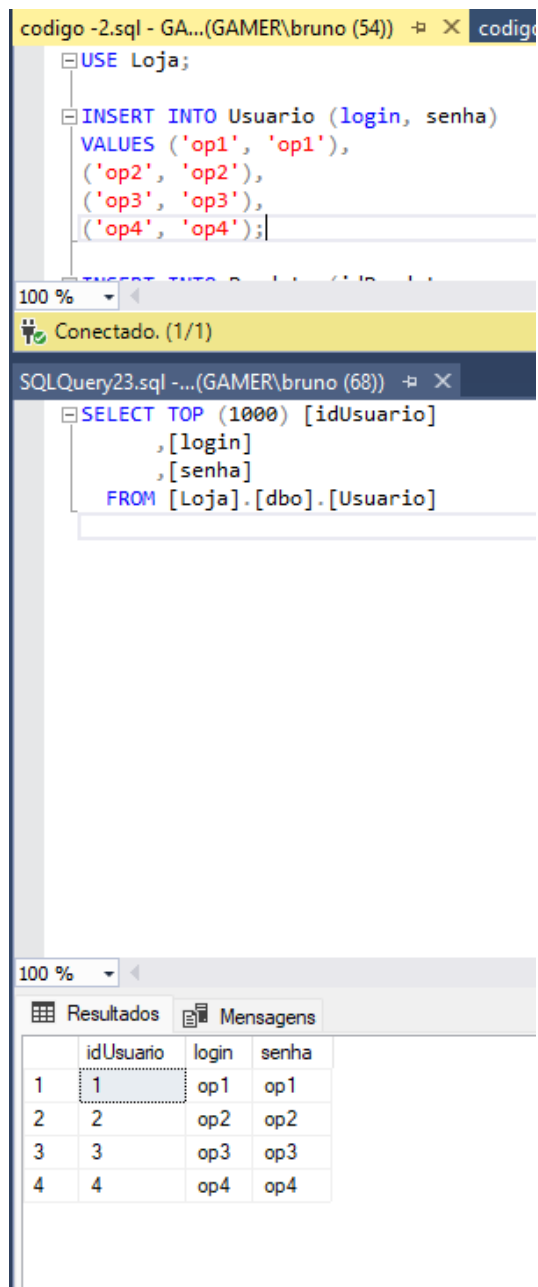
7 TODOS OS CÓDIGOS SOLICITADOS NESTE ROTEIRO DE AULA;

7.1 LOGAR. COMO USUÁRIO LOJA, SENHA LOJA.



7.2 UTILIZANDO O EDITOR DE SQL PARA INCLUIR DADOS NA TABELA , DE FORMA A
OBTER UM CONJUNTO COMO O APRESENTADO A SEGUIR:

Usuário



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL script with the following queries:

```
USE Loja;

INSERT INTO Usuario (login, senha)
VALUES ('op1', 'op1'),
('op2', 'op2'),
('op3', 'op3'),
('op4', 'op4');
```

The middle pane shows the status "Conectado. (1/1)" and the query window "SQLQuery23.sql - ... (GAMER\bruno (68))". The query in this window is:

```
SELECT TOP (1000) [idUsuario]
, [login]
, [senha]
FROM [Loja].[dbo].[Usuario]
```

The bottom pane shows the "Resultados" (Results) tab with a table containing 4 rows of data:

	idUsuario	login	senha
1	1	op1	op1
2	2	op2	op2
3	3	op3	op3
4	4	op4	op4

Produtos

codigo -2.sql - GA...(GAMER\bruno (54))* X codigo -3.sql - C

```
INSERT INTO Produto (idProduto, nome, quantidade, precoVenda)
VALUES ('1', 'Teclado', '30', '100.00'),
('3', 'Mouse', '100', '70.00'),
('4', 'Monitor', '10', '200.00');
```

100 %

Conectado. (1/1)

SQLQuery25.sql -...(GAMER\bruno (70)) X

```
SELECT TOP (1000) [idProduto]
, [nome]
, [quantidade]
, [precoVenda]
FROM [Loja].[dbo].[Produto]
```

100 %

Resultados Mensagens

	idProduto	nome	quantidade	precoVenda
1	1	Teclado	100	100.00
2	3	Mouse	100	70.00
3	4	Monitor	50	300.00

Usuário

codigo -2.sql - GA...(GAMER\bruno (62))

```
INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)
VALUES
(NEXT VALUE FOR ordemPessoaId, 'Natalia', 'Rua I, 50', 'Angra dos Reis', 'RJ', '1111-1111', 'Natalia@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Bruno', 'Rua P, 10', 'Volta Redonda', 'RJ', '2222-2222', 'Bruno@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Jonas', 'Rua 5, 80', 'Vitoria', 'ES', '3333-3333', 'Jonas@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Distribuidora Renova', 'Avenida J, 80', 'São Paulo', 'SP', '4444-4444', 'RenovaDist@gmail.com'),
(NEXT VALUE FOR ordemPessoaId, 'Empresa Jasper', 'Avenida H, 70', 'São Paulo', 'SP', '5555-5555', 'Jasper@gmail.com');
```

100 %

Conectado. (1/1)

SQLQuery9.sql - G...(GAMER\bruno (53))

```
SELECT TOP (1000) [idPessoa]
, [nome]
, [logradouro]
, [cidade]
, [estado]
, [telefone]
, [email]
FROM [Loja].[dbo].[Pessoa]
```

100 %

Resultados Mensagens

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	Natalia	Rua I, 50	Angra dos Reis	RJ	1111-1111	Natalia@gmail.com
2	2	Bruno	Rua P, 10	Volta Redonda	RJ	2222-2222	Bruno@gmail.com
3	3	Jonas	Rua 5, 80	Vitoria	ES	3333-3333	Jonas@gmail.com
4	4	Distribuidora Renova	Avenida J, 80	São Paulo	SP	4444-4444	RenovaDist@gmail.com
5	5	Empresa Jasper	Avenida H, 70	São Paulo	SP	5555-5555	Jasper@gmail.com

Pessoa física

codigo -2.sql - GA...(GAMER\bruno (61))

```
INSERT INTO PessoaFisica (idPessoa, cpf)
VALUES (1, '11111111111'),
(2, '22222222222'),
(3, '33333333333');
```

100 %

Conectado. (1/1)

SQLQuery12.sql -...(GAMER\bruno (66))

```
SELECT TOP (1000) [idPessoa]
, [cpf]
FROM [Loja].[dbo].[PessoaFisica]
```

100 %

Resultados Mensagens

	idPessoa	cpf
1	1	11111111111
2	2	22222222222
3	3	33333333333

Pessoa jurídica

codigo -2.sql - GA...(GAMER\bruno (61))

```
INSERT INTO PessoaJuridica (idPessoa, cnpj)
VALUES (4, '4444444444444444'),
(5, '5555555555555555');
```

100 %

Conectado. (1/1)

SQLQuery13.sql -...(GAMER\bruno (63))

```
SELECT TOP (1000) [idPessoa]
, [cnpj]
FROM [Loja].[dbo].[PessoaJuridica]
```

100 %

Resultados Mensagens

	idPessoa	cnpj
1	4	4444444444444444
2	5	5555555555555555

Movimento

codigo -2.sql - GA...(GAMER\bruno (51))

```
INSERT INTO Movimento (idMovimento, idUsuario, idPessoa, idProduto, quantidade, tipo, valorUnitario)
VALUES (1, 1, 5, 1, 40, 'E', 60.00),
(3, 2, 3, 3, 20, 'S', 80.00),
(5, 1, 4, 4, 60, 'E', 75.00),
(6, 2, 1, 1, 15, 'S', 60.00),
(7, 4, 2, 4, 25, 'S', 75.00),
(9, 3, 5, 3, 50, 'E', 80.00);
```

100 %

Conectado. (1/1)

SQLQuery2.sql - G...(GAMER\bruno (72))

```
SELECT TOP (1000) [idMovimento]
, [idUsuario]
, [idPessoa]
, [idProduto]
, [quantidade]
, [tipo]
, [valorUnitario]
FROM [Loja].[dbo].[Movimento]
```

100 %

Resultados Mensagens

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
1	1	1	5	1	40	E	60.00
2	3	2	3	3	20	S	80.00
3	5	1	4	4	60	E	75.00
4	6	2	1	1	15	S	60.00
5	7	4	2	4	25	S	75.00
6	9	3	5	3	50	E	80.00

8 EFETUAR AS SEGUINTE CONSULTAS SOBRE OS DADOS INSERIDOS:

Pessoa física e jurídica

codigo -3.sql - GA...(GAMER\bruno (53))

SELECT *

FROM PessoaFisica

INNER JOIN Pessoa ON PessoaFisica.idPessoa = Pessoa.idPessoa

SELECT *

FROM PessoaJuridica

INNER JOIN Pessoa ON PessoaJuridica.idPessoa = Pessoa.idPessoa

100 %

Resultados

Mensagens

	idPessoa	cpf	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	11111111111	1	Natalia	Rua I, 50	Angra dos Reis	RJ	1111-1111	Natalia@gmail.com
2	2	22222222222	2	Bruno	Rua P, 10	Volta Redonda	RJ	2222-2222	Bruno@gmail.com
3	3	33333333333	3	Jonas	Rua 5, 80	Vitoria	ES	3333-3333	Jonas@gmail.com

	idPessoa	cnpj	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	4	444444444444444	4	Distribuidora Renova	Avenida J, 80	São Paulo	SP	4444-4444	RenovaDist@gmail.com
2	5	555555555555555	5	Empresa Jasper	Avenida H, 70	São Paulo	SP	5555-5555	Jasper@gmail.com

Entrada/ saídas / valor total de material

codigo -3.sql - GA...(GAMER\bruno (53))

```
SELECT
    Produto.nome AS 'produto', Pessoa.nome AS 'fornecedor',
    Movimento.quantidade, Movimento.valorUnitario,
    Movimento.quantidade * Movimento.valorUnitario AS 'valorTotal'
FROM Movimento
INNER JOIN Produto ON Movimento.idProduto = Produto.idProduto
INNER JOIN Pessoa ON Movimento.idPessoa = Pessoa.idPessoa
WHERE Movimento.tipo = 'E';

SELECT
    Produto.nome AS 'produto', Pessoa.nome AS 'comprador',
    Movimento.quantidade, Movimento.valorUnitario,
    Movimento.quantidade * Movimento.valorUnitario AS 'valorTotal'
FROM Movimento
INNER JOIN Produto ON Movimento.idProduto = Produto.idProduto
INNER JOIN Pessoa ON Movimento.idPessoa = Pessoa.idPessoa
WHERE Movimento.tipo = 'S';

SELECT
    Produto.nome AS 'produto',
    SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalEntrada'
FROM Movimento
JOIN Produto ON Movimento.idProduto = Produto.idProduto
WHERE Movimento.tipo = 'E'
GROUP BY Produto.nome;

SELECT
    Produto.nome AS 'produto',
    SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalSaida'
FROM Movimento
JOIN Produto ON Movimento.idProduto = Produto.idProduto
WHERE Movimento.tipo = 'S'
GROUP BY Produto.nome;
```

85 %

Resultados Mensagens

	produto	fornecedor	quantidade	valorUnitario	valorTotal
1	Teclado	Empresa Jasper	40	60.00	2400.00
2	Monitor	Distribuidora Renova	60	75.00	4500.00
3	Mouse	Empresa Jasper	50	80.00	4000.00

	produto	comprador	quantidade	valorUnitario	valorTotal
1	Mouse	Jonas	20	80.00	1600.00
2	Teclado	Natalia	15	60.00	900.00
3	Monitor	Bruno	25	75.00	1875.00

	produto	valorTotalEntrada
1	Monitor	4500.00
2	Mouse	4000.00
3	Teclado	2400.00

	produto	valorTotalSaida
1	Monitor	1875.00
2	Mouse	1600.00
3	Teclado	900.00

Operadores que não efetuaram movimentações de entrada

codigo -3.sql - GA...(GAMER\bruno (53))

```
SELECT DISTINCT  
    Usuario.idUsuario, Usuario.login, Movimento.idMovimento  
FROM Usuario  
LEFT JOIN Movimento ON Usuario.idUsuario = Movimento.idUsuario AND Movimento.tipo = 'E'  
WHERE idMovimento IS NULL;
```

SELECT

100 %

Resultados Mensagens

	idUsuario	login	idMovimento
1	2	op2	NULL
2	4	op4	NULL

Valor total de entrada / saída por operador

codigo -3.sql - GA...(GAMER\bruno (53)) -p X

```
SELECT
  Usuario.login AS 'operador',
  SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalEntrada'
FROM Movimento
JOIN Usuario ON Movimento.idUsuario = Usuario.idUsuario
WHERE Movimento.tipo = 'E'
GROUP BY Usuario.login;

SELECT
  Usuario.login AS 'operador',
  SUM(Movimento.quantidade * Movimento.valorUnitario) AS 'valorTotalSaida'
FROM Movimento
JOIN Usuario ON Movimento.idUsuario = Usuario.idUsuario
WHERE Movimento.tipo = 'S'
GROUP BY Usuario.login;
```

100 %

Resultados Mensagens

	operador	valorTotalEntrada
1	op1	6900.00
2	op3	4000.00

	operador	valorTotalSaida
1	op2	2500.00
2	op4	1875.00

Valor médio de venda por produto

codigo -3.sql - GA...(GAMER\bruno (53))

```
SELECT
SUM(Movimento.valorUnitario * Movimento.quantidade) / SUM(Movimento.quantidade) AS 'médiaPonderada'
FROM Movimento
WHERE Movimento.tipo = 'S';
```

100 %

Resultados Mensagens

	médiaPonderada
1	72.916666

9 ANÁLISE E CONCLUSÃO:

9.1 QUAIS AS DIFERENÇAS NO USO DE SEQUENCE E IDENTITY?

IDENTITY

Definição: Propriedade de uma coluna em uma tabela.

Criação: Definido diretamente na coluna de uma tabela.

Incremento: Especifica um valor inicial e um incremento.

Escopo: Limitado à tabela onde está definido.

Controle: Menos flexível, controlado pela tabela.

Resequenciamento: Difícil de resequenciar sem scripts complexos.

SEQUENCE

Definição: Objeto de banco de dados independente.

Criação: Definido como um objeto separado no banco de dados.

Incremento: Pode definir valor inicial, incremento, mínimo, máximo e comportamento de ciclo.

Escopo: Global ao banco de dados, pode ser compartilhado entre tabelas.

Controle: Mais flexível, pode ser manipulado diretamente.

Resequenciamento: Fácil de reiniciar ou alterar sem modificar tabelas.

Resumo Comparativo

IDENTITY: Simples, embutido na tabela, para uso específico de uma tabela.

SEQUENCE: Flexível, configurável, reutilizável em múltiplas tabelas, requer configuração inicial.

Quando Usar

IDENTITY: Quando precisa de identificador único simples para uma tabela específica.

SEQUENCE: Quando precisa de maior controle, compartilhamento entre tabelas, ou configurações específicas de geração de valores.

10 QUAL A IMPORTÂNCIA DAS CHAVES ESTRANGERIAS PARA A CONSISTÊNCIA DO BANCO?

As chaves estrangeiras são essenciais para a consistência e integridade dos dados em bancos de dados relacionais, desempenhando as seguintes funções importantes:

Integridade Referencial: Garantem que relações entre tabelas sejam válidas, evitando referências a registros inexistentes.

Prevenção de Dados Órfãos: Impedem que registros em uma tabela filha fiquem sem correspondência em uma tabela pai.

Facilidade de Manutenção: Facilitam a manutenção dos dados, permitindo que alterações em registros pais sejam refletidas automaticamente nos registros filhos.

Validação de Dados: Servem como uma forma de validação, assegurando que apenas valores válidos sejam inseridos em colunas de chaves estrangeiras.

Facilitação de Consultas e Relacionamentos: Ajudam a definir claramente as relações entre tabelas, tornando consultas complexas mais eficientes.

Documentação Implícita: Fornecem uma documentação sobre como as tabelas estão relacionadas, facilitando a compreensão da estrutura do banco de dados.

Impacto na Performance: Embora possam introduzir alguma sobrecarga, também podem melhorar o desempenho de consultas que envolvem joins entre tabelas.

Conclusão

As chaves estrangeiras são fundamentais para garantir a consistência dos dados, prevenindo erros e facilitando tanto a manutenção quanto a consulta em bancos de dados relacionais.

11 QUAIS OPERADORES DO SQL PERTENCEM À ÁLGEBRA RELACIONAL E QUAIS SÃO DEFINIDOS NO CÁLCULO RELACIONAL?

A álgebra relacional e o cálculo relacional são dois formalismos teóricos fundamentais usados para manipular e consultar dados em bancos de dados relacionais. Aqui está um resumo dos operadores pertencentes a cada um:

Álgebra Relacional

Definição: Um conjunto de operadores que operam em relações (tabelas) e produzem novas relações.

Principais Operadores:

Seleção (σ): Filtra linhas que atendem a uma condição.

Projeção (π): Retorna colunas específicas, eliminando duplicatas.

União (\cup): Combina linhas de duas relações.

Interseção (\cap): Retorna linhas comuns a ambas as relações.

Diferença ($-$): Retorna linhas que estão em uma relação, mas não na outra.

Produto Cartesiano (\times): Combina todas as linhas de duas relações.

Junção (\bowtie): Combina linhas de duas relações com base em uma condição de correspondência.

Renomeação (ρ): Altera o nome de uma relação ou de suas colunas.

Cálculo Relacional

Definição: Uma abordagem declarativa que utiliza expressões lógicas para especificar consultas.

Principais Tipos:

Cálculo de Tupla: Especifica as tuplas que satisfazem uma condição lógica.

Exemplo: $\{t \mid t \in R \text{ e } \text{condição}(t)\}$.

Cálculo de Domínio: Especifica os valores dos atributos que satisfazem uma condição lógica.

Exemplo: $\{a_1, a_2, \dots, a_n \mid \exists t (t \in R \wedge \text{condição}(a_1, a_2, \dots, a_n))\}$.

12 COMO É FEITO O AGRUPAMENTO EM CONSULTAS, E QUAL REQUISITO É OBRIGATÓRIO?

O agrupamento em consultas SQL é realizado usando a cláusula GROUP BY, que permite agrupar linhas que têm valores iguais em uma ou mais colunas, facilitando a execução de funções de agregação, como SUM, COUNT, AVG, MAX, e MIN. O agrupamento é frequentemente utilizado para resumir dados e gerar relatórios.

Como Funciona:

Uso da Cláusula GROUP BY: Especifica as colunas para agrupar os dados.

Funções de Agregação: Utilizadas para calcular valores resumidos para cada grupo.

Exemplo:

```
SELECT coluna1, COUNT(*)  
FROM tabela  
GROUP BY coluna1;
```

Requisito Obrigatório:

Todas as colunas na cláusula SELECT que não são funções de agregação devem estar incluídas na cláusula GROUP BY. Caso contrário, ocorrerá um erro.

Exemplo de Requisito:

```
SELECT nome, AVG(salario)  
FROM funcionarios  
GROUP BY nome; -- Correto  
  
SELECT nome, departamento, AVG(salario)  
FROM funcionarios  
GROUP BY nome; -- Causará erro, pois 'departamento' não está agrupado.
```