UNIVERSIDADE ESTÁCIO DE SÁ

POLO JEREISSATI 1 - MARACANAÚ/CE

TECNOLOGIA EM DESENVOLVIMENTO FULL STACK

RPG0015 - VAMOS MANTER AS INFORMAÇÕES?

EMANUEL ROSEIRA GUEDES

RELATÓRIO DA MISSÃO PRÁTICA

MARACANAÚ

2024

EMANUEL ROSEIRA GUEDES

RELATÓRIO DA MISSÃO PRÁTICA

Relatório apresentado à disciplina RPG0015 - Vamos manter as informações? do curso de Tecnologia em Desenvolvimento Full Stack da Universidade Estácio de Sá.

Tutora: Profa. Ma. Simone Ingrid Monteiro Gama

MARACANAÚ

2024

1 INTRODUÇÃO

A gestão eficiente de dados é essencial para o desenvolvimento de sistemas de informação robustos e eficazes. No contexto do Nível 2 do Mundo 3 do curso de Tecnologia em Desenvolvimento Full Stack, nível intitulado “Vamos Manter as Informações?”, a prática de modelagem e implementação desempenha um papel crucial na aplicação dos conhecimentos teóricos adquiridos. Este trabalho relata a experiência de criação de um banco de dados relacional para um projeto fictício de uma loja, envolvendo desde a definição da estrutura das tabelas até a inserção de dados e a elaboração de consultas SQL. A implementação prática proporcionou uma compreensão mais profunda dos conceitos abordados em aula, destacando a importância da modelagem adequada para garantir a integridade e eficiência dos dados.

1.1 Objetivos da prática

O objetivo principal desta prática é aplicar os conceitos teóricos de modelagem de banco de dados, utilizando o Microsoft SQL Server como plataforma. Através da criação de um banco de dados para um projeto fictício de uma loja, busca-se consolidar o entendimento sobre a estruturação de dados em ambientes relacionais.

Na sequência, são apresentados os objetivos específicos que orientaram a prática:

a) Realizar a criação do banco de dados “Loja” contemplando tabelas para armazenar informações sobre pessoas, usuários, produtos e movimentações.
b) Inserir dados fictícios nas tabelas criadas, abrangendo aspectos como pessoas físicas e jurídicas, usuários, produtos, e movimentações de entrada e saída.

Estes objetivos específicos visam não apenas a aplicação prática dos conceitos, mas também a análise crítica dos resultados obtidos, permitindo uma avaliação abrangente do processo de modelagem e implementação de banco de dados.

2 METODOLOGIA

A metodologia empregada nesta prática foi projetada para guiar a criação e implementação eficiente do banco de dados, levando em consideração a utilização da ferramenta DB Designer Online para a modelagem visual.

A primeira etapa consistiu na definição da estrutura do banco de dados por meio de uma modelagem visual. Utilizou-se a plataforma online DB Designer para criar esquemas conceituais, identificando entidades, atributos e os relacionamentos entre elas. Este ambiente visual permitiu uma abordagem interativa e intuitiva na elaboração do design conceitual.

A ferramenta DB Designer Online foi escolhida devido à sua acessibilidade e facilidade de uso. Durante essa fase, foram elaborados esboços visuais representando tabelas, seus atributos e as relações entre entidades. A visualização facilitou a validação conceitual da estrutura do banco de dados antes da implementação.

Após a conclusão da modelagem no DB Designer Online, a ferramenta permitiu a geração automática de códigos SQL correspondentes ao esquema criado. Esses códigos foram utilizados como base para a implementação no SQL Server Management Studio, garantindo consistência entre a modelagem conceitual e a implementação prática.

O ambiente SQL Server Management Studio foi utilizado para a implementação efetiva da estrutura do banco de dados no servidor SQL Server. Os códigos SQL gerados pelo DB Designer Online foram adaptados conforme necessário, considerando características específicas do SQL Server.

Para simular um ambiente real, foram inseridos dados fictícios nas tabelas do banco. Utilizando *scripts* SQL no SQL Server Management Studio, seguindo as diretrizes definidas para cada tabela, proporcionando uma carga inicial de informações no banco de dados.

A fase final abrangeu a elaboração e execução de consultas SQL para analisar e extrair informações do banco de dados. Utilizando o SQL Server Management Studio, formulou-se consultas que abordaram desde dados específicos de pessoas físicas e jurídicas até análises de movimentações, entradas e saídas de produtos.

Essa metodologia integrada, combinando a modelagem visual no DB

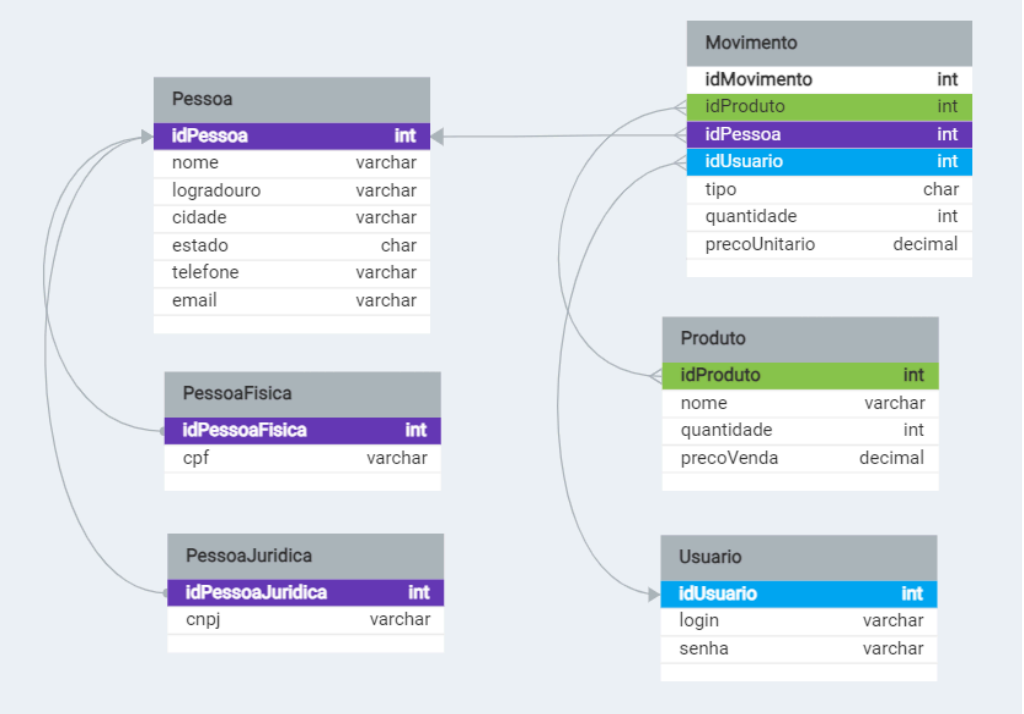
Designer Online e a implementação prática no SQL Server Management Studio, proporcionou uma abordagem abrangente e coesa para o desenvolvimento do banco de dados. Essa combinação de ferramentas facilitou a visualização conceitual, garantindo uma transição suave para a implementação efetiva no ambiente de servidor.

3 RESULTADOS

Com isso, apresentam-se os resultados da prática de modelagem e implementação do banco de dados para o projeto fictício da loja. Apresentando a modelagem do banco de dados utilizando a ferramenta DB Designer Online, criação da estrutura do banco no SQL Server Management Studio, alimentação do banco com dados fictícios e execução de consultas SQL.

O início deste projeto foi dedicada à modelagem conceitual do banco de dados, crucial para definir a estrutura que sustentaria o sistema da loja fictícia. Optou-se por utilizar a ferramenta DB Designer Online, uma plataforma de modelagem visual intuitiva e acessível, sendo criada a modelagem mostrada na Imagem 1, a seguir.

Imagem 1 - Modelagem do banco de dados.



A imagem acima representa uma visão geral da modelagem, evidenciando as entidades principais e seus atributos. A entidade Pessoa, por exemplo, engloba informações como nome, logradouro, cidade, estado, telefone e email. A relação entre Pessoa e Usuario foi estabelecida para incorporar dados de login e senha às informações das pessoas.

Uma abordagem de herança foi aplicada para diferenciar entre Pessoas Físicas e Jurídicas. As entidades PessoaFisica e PessoaJuridica herdam características da entidade base Pessoa, enquanto suas particularidades, como CPF e CNPJ, são representadas em tabelas específicas.

A modelagem visual permitiu a definição clara de chaves primárias, estrangeiras e índices, garantindo a integridade referencial do banco de dados. A relação entre as entidades, como indicado na imagem, estabelece a base para a construção de consultas complexas e a recuperação eficiente de dados durante a fase de implementação.

Essa representação gráfica foi fundamental para validar conceitualmente a estrutura do banco antes da implementação no SQL Server Management Studio. A visualização prévia proporcionou uma compreensão abrangente da interconexão entre as entidades e seus relacionamentos, tornando a transição da modelagem para a implementação mais fluida e consistente.

A fase de criação e estruturação do banco de dados foi realizada para definir a arquitetura necessária para suportar as operações da loja fictícia. O processo abrange desde a criação do banco no SQL Server até a definição das tabelas, chaves primárias e estrangeiras, estabelecendo relações coesas entre os dados. A seguir, detalha-se cada componente dessa etapa.

Para iniciar a implementação do banco de dados, foram executados *scripts* SQL para criar um ambiente seguro no SQL Server. Isso incluiu a criação de um login (loja) e usuário correspondente, garantindo a autenticação necessária para manipulação do banco de dados.

Código 1 - Criação de usuário e banco de dados.

```
USE master;
CREATE LOGIN loja WITH PASSWORD = 'loja', CHECK_POLICY = OFF;
CREATE USER loja FOR LOGIN loja;
CREATE DATABASE Loja;
USE Loja;
```

Em seguida, as tabelas foram concebidas para representar entidades essenciais no contexto da loja. A tabela Pessoa armazena informações gerais sobre clientes, a Usuario gerencia os usuários do sistema, PessoaFisica e PessoaJuridica registram dados específicos de cada tipo de pessoa, Produto guarda informações sobre os produtos disponíveis, e Movimento registra as transações realizadas.

Código 2 - Criação da estrutura de tabelas, campos e chaves.

```
CREATE TABLE [Pessoa] (
       idPessoa int NOT NULL UNIQUE,
       nome varchar(50) NOT NULL,
       logradouro varchar(50) NOT NULL,
       cidade varchar(50) NOT NULL,
       estado char(2) NOT NULL,
       telefone varchar(11) NOT NULL,
       email varchar(50) NOT NULL,
  CONSTRAINT [PK_PESSOA] PRIMARY KEY CLUSTERED
  (
  [idPessoa] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
CREATE TABLE [Usuario] (
       idUsuario int NOT NULL UNIQUE,
       login varchar(25) NOT NULL UNIQUE,
       senha varchar(16) NOT NULL,
  CONSTRAINT [PK_USUARIO] PRIMARY KEY CLUSTERED
  (
  [idUsuario] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
CREATE TABLE [PessoaFisica] (
       idPessoaFisica int NOT NULL UNIQUE,
       cpf varchar(11) NOT NULL UNIQUE,
  CONSTRAINT [PK_PESSOAFISICA] PRIMARY KEY CLUSTERED
  (
  [idPessoaFisica] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
CREATE TABLE [PessoaJuridica] (
       idPessoaJuridica int NOT NULL UNIQUE,
       cnpj varchar(14) NOT NULL UNIQUE,
  CONSTRAINT [PK_PESSOAJURIDICA] PRIMARY KEY CLUSTERED
  (
  [idPessoaJuridica] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
CREATE TABLE [Produto] (
       idProduto int NOT NULL UNIQUE,
       nome varchar(100) NOT NULL,
       quantidade int NOT NULL,
       precoVenda decimal (10, 2) NOT NULL,
```

```
  CONSTRAINT [PK_PRODUTO] PRIMARY KEY CLUSTERED
  (
  [idProduto] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
CREATE TABLE [Movimento] (
       idMovimento int NOT NULL UNIQUE,
       idProduto int NOT NULL,
       idPessoa int NOT NULL,
       idUsuario int NOT NULL,
       tipo char(1) NOT NULL,
       quantidade int NOT NULL,
       precoUnitario decimal(10, 2) NOT NULL,
  CONSTRAINT [PK_MOVIMENTO] PRIMARY KEY CLUSTERED
  (
  [idMovimento] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
```

As relações entre as tabelas foram cuidadosamente planejadas para manter a integridade referencial. Por exemplo, a tabela PessoaFisica possui uma chave estrangeira (PessoaFisica_fk0) que faz referência à chave primária (idPessoa) da tabela Pessoa. Semelhantemente, as tabelas PessoaJuridica e Movimento estabelecem relações com a tabela Pessoa.

Código 3 - Criação de relações/dependências referenciais e integrativas.

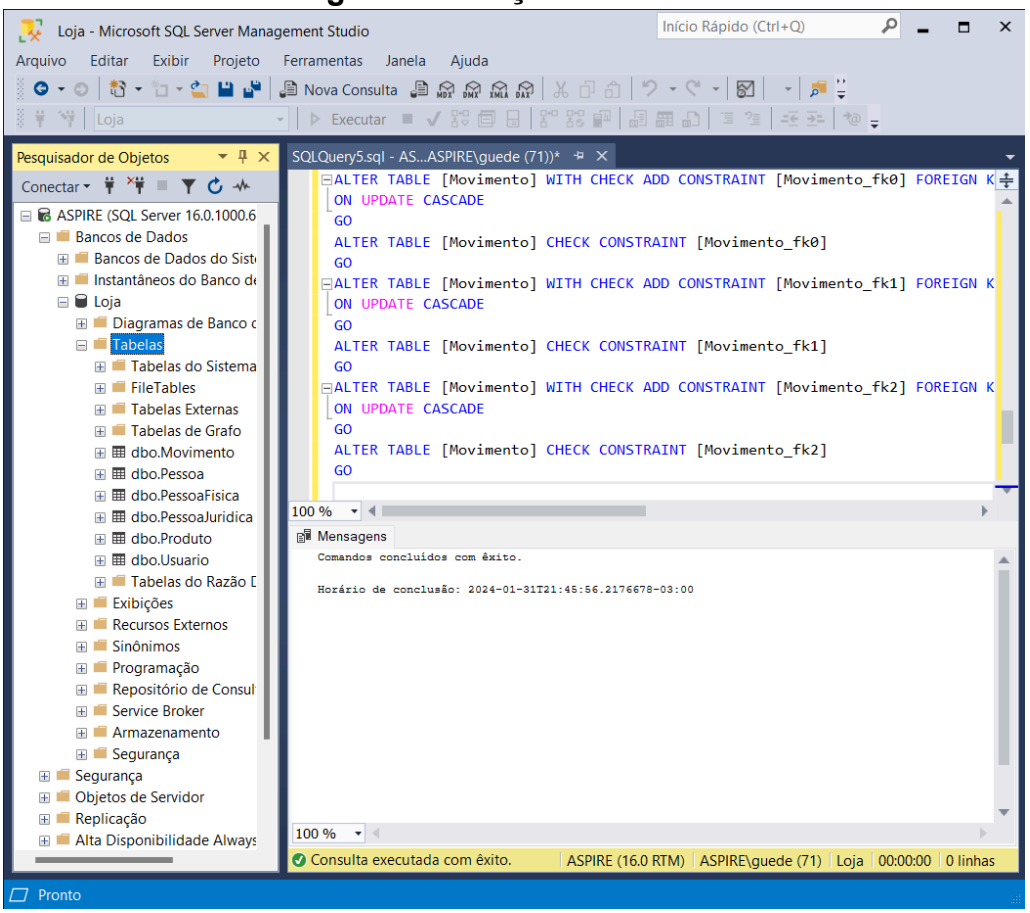
```
ALTER TABLE [PessoaFisica] WITH CHECK ADD CONSTRAINT [PessoaFisica_fk0] FOREIGN
KEY ([idPessoaFisica]) REFERENCES [Pessoa]([idPessoa])
ON UPDATE CASCADE
ALTER TABLE [PessoaFisica] CHECK CONSTRAINT [PessoaFisica_fk0]

ALTER TABLE [PessoaJuridica] WITH CHECK ADD CONSTRAINT [PessoaJuridica_fk0]
FOREIGN KEY ([idPessoaJuridica]) REFERENCES [Pessoa]([idPessoa])
ON UPDATE CASCADE
ALTER TABLE [PessoaJuridica] CHECK CONSTRAINT [PessoaJuridica_fk0]

ALTER TABLE [Movimento] WITH CHECK ADD CONSTRAINT [Movimento_fk0] FOREIGN KEY
([idProduto]) REFERENCES [Produto]([idProduto])
ON UPDATE CASCADE
ALTER TABLE [Movimento] CHECK CONSTRAINT [Movimento_fk0]
ALTER TABLE [Movimento] WITH CHECK ADD CONSTRAINT [Movimento_fk1] FOREIGN KEY
([idPessoa]) REFERENCES [Pessoa]([idPessoa])
ON UPDATE CASCADE
ALTER TABLE [Movimento] CHECK CONSTRAINT [Movimento_fk1]
ALTER TABLE [Movimento] WITH CHECK ADD CONSTRAINT [Movimento_fk2] FOREIGN KEY
([idUsuario]) REFERENCES [Usuario]([idUsuario])
ON UPDATE CASCADE
ALTER TABLE [Movimento] CHECK CONSTRAINT [Movimento_fk2]
```

As cardinalidades entre as tabelas foram estabelecidas de acordo com a

natureza dos dados. Por exemplo, a relação entre Pessoa e PessoaFisica é de 1 para 1, indicando que cada pessoa pode ter no máximo uma representação em PessoaFisica. Já a relação entre Movimento e Produto é de 1 para n, permitindo que um produto esteja associado a várias movimentações.

Os códigos foram devidamente testados apresentando o resultado esperado durante a modelagem do banco de dados como apresentado na Imagem 2, a seguir.

Imagem 2 - Criação das tabelas.



Essa abordagem estruturada na criação do banco de dados estabeleceu uma base sólida para as operações futuras da loja fictícia, assegurando a consistência e integridade dos dados. A próxima seção abordará a inserção de dados e consultas SQL, demonstrando a funcionalidade prática do banco implementado.

A fase de alimentação do banco de dados e execução de consultas SQL desempenha um papel crucial para verificar a eficácia da modelagem e garantir que as operações podem ser realizadas de maneira consistente. A seguir, detalha-se os

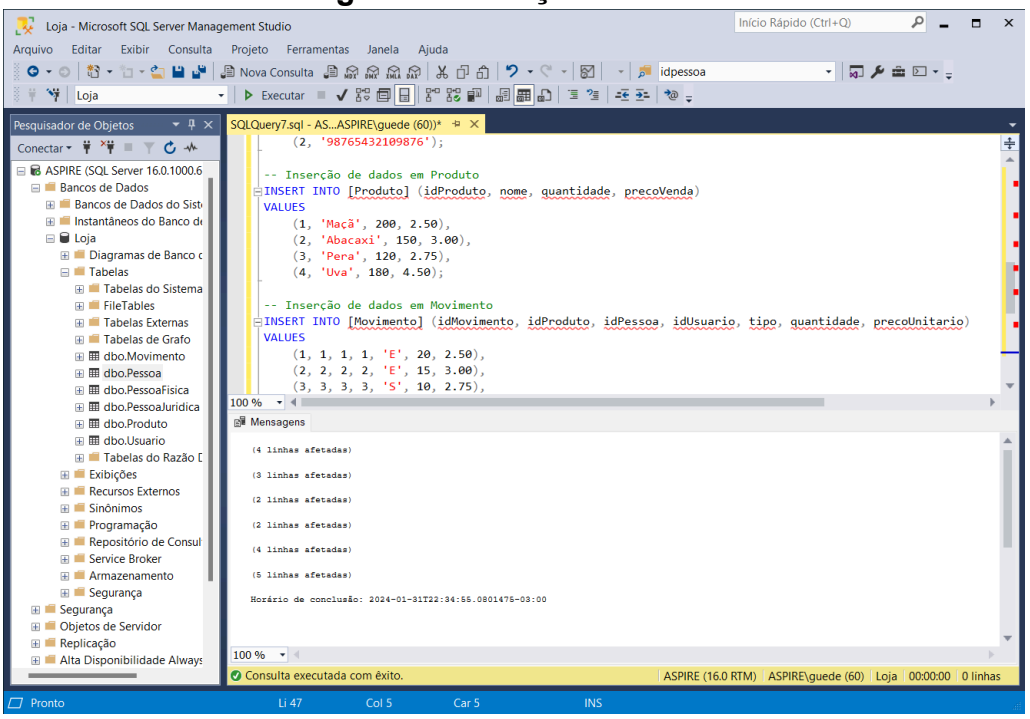
procedimentos de inserção de dados e apresenta-se as consultas SQL relevantes.

Os *scripts* de inserção de dados foram elaborados para preencher as tabelas com informações fictícias, simulando um ambiente operacional. Diversos dados foram adicionados, abrangendo clientes, usuários, produtos, e movimentações de entrada e saída, conforme o Código 4 com resultado apresentado na Imagem 3.

Código 4 - Inserção de dados nas tabelas.

```
INSERT INTO [Pessoa] (idPessoa, nome, logradouro, cidade, estado, telefone,
email)
VALUES
    (1, 'Ana Oliveira', 'Rua das Flores 123', 'São Paulo', 'SP', '11987654321',
'ana@email.com'),
    (2, 'Carlos Silva', 'Avenida Central 456', 'Rio de Janeiro', 'RJ',
'21987654321', 'carlos@email.com'),
    (3, 'Mariana Santos', 'Travessa Exemplo 789', 'Belo Horizonte', 'MG',
'31987654321', 'mariana@email.com'),
    (4, 'José Pereira', 'Rua Principal 321', 'Porto Alegre', 'RS',
'51987654321', 'jose@email.com');

INSERT INTO [Usuario] (idUsuario, login, senha)
VALUES
    (1, 'funcionario1', 'senha123'),
    (2, 'funcionario2', 'senha456'),
    (3, 'gerente1', 'senha789');

INSERT INTO [PessoaFisica] (idPessoaFisica, cpf)
VALUES
    (1, '11122233344'),
    (2, '55566677788');

INSERT INTO [PessoaJuridica] (idPessoaJuridica, cnpj)
VALUES
    (1, '12345678901234'),
    (2, '98765432109876');

INSERT INTO [Produto] (idProduto, nome, quantidade, precoVenda)
VALUES
    (1, 'Maçã', 200, 2.50),
    (2, 'Abacaxi', 150, 3.00),
    (3, 'Pera', 120, 2.75),
    (4, 'Uva', 180, 4.50);

INSERT INTO [Movimento] (idMovimento, idProduto, idPessoa, idUsuario, tipo,
quantidade, precoUnitario)
VALUES
    (1, 1, 1, 1, 'E', 20, 2.50),
    (2, 2, 2, 2, 'E', 15, 3.00),
    (3, 3, 3, 3, 'S', 10, 2.75),
    (4, 4, 4, 1, 'E', 25, 4.50),
    (5, 1, 2, 2, 'S', 5, 3.00);
```

Imagem 3 - Inserção de dados.



As consultas SQL foram desenvolvidas para extrair informações específicas do banco de dados, demonstrando a flexibilidade e potencial analítico do modelo implementado, apresentando o resultado das consultas considerando as seguintes requisições:

a) Dados completos de pessoas físicas.
b) Dados completos de pessoas jurídicas.
c) Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.
d) Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total.
e) Valor total das entradas agrupadas por produto.
f) Valor total das saídas agrupadas por produto.
g) Operadores que não efetuaram movimentações de entrada (compra).
h) Valor total de entrada, agrupado por operador.
i) Valor total de saída, agrupado por operador.
j) Valor médio de venda por produto, utilizando média ponderada.

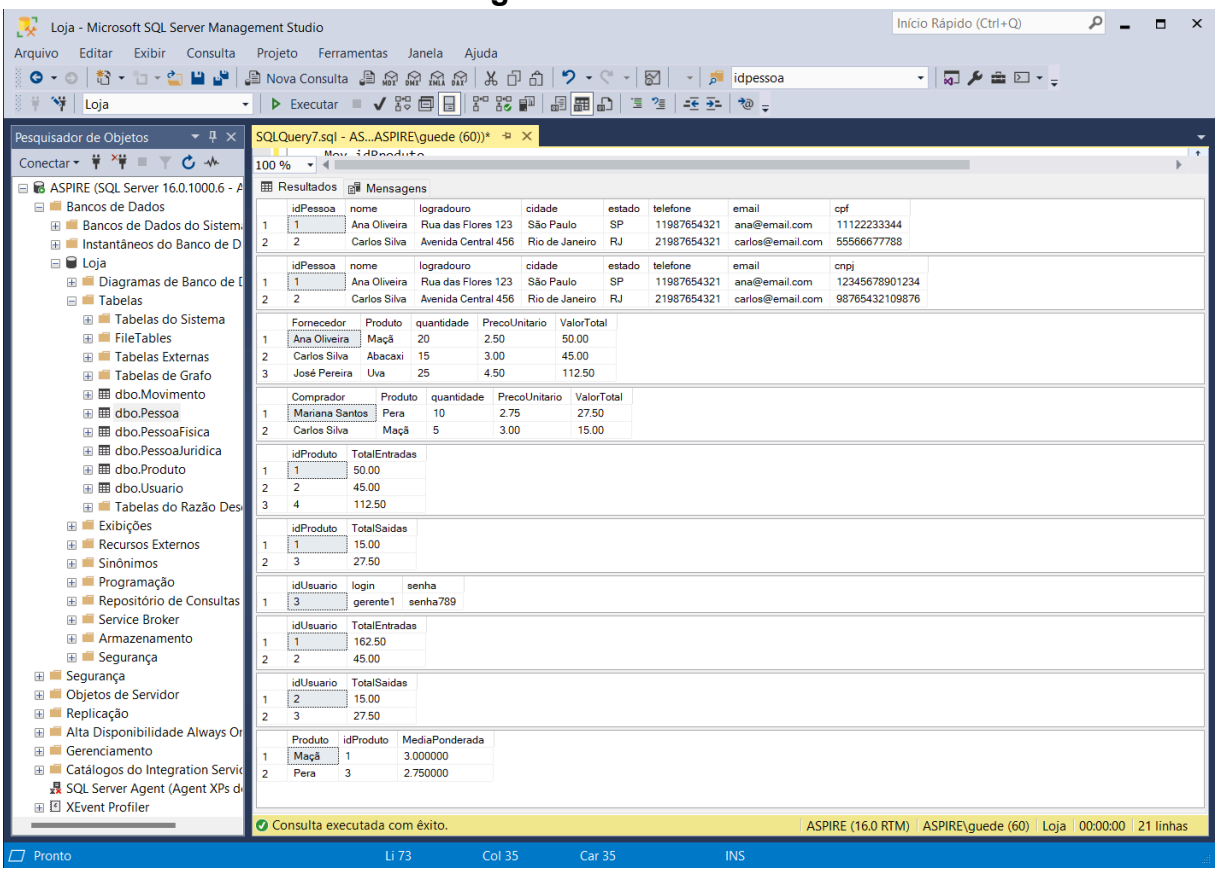
Para obter estas informações, utilizando as requisições contidas no Código 5, com os resultados das requisições SQL apresentados na Imagem 4.

Código 5 - Consultas.

```sql
SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade,
Pessoa.estado, Pessoa.telefone, Pessoa.email, PF.cpf
FROM Pessoa
JOIN PessoaFisica PF ON Pessoa.idPessoa = PF.idPessoaFisica;

SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade,
Pessoa.estado, Pessoa.telefone, Pessoa.email, PJ.cnpj
FROM Pessoa
JOIN PessoaJuridica PJ ON Pessoa.idPessoa = PJ.idPessoaJuridica;

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 Mov.tipo = 'E';

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 Mov.tipo = 'S';

SELECT Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) AS TotalEntradas
FROM Movimento Mov
WHERE Mov.tipo = 'E'
GROUP BY Mov.idProduto;

SELECT Mov.idProduto, SUM(Mov.quantidade * Mov.precoUnitario) AS TotalSaidas
FROM Movimento Mov
WHERE Mov.tipo = 'S'
GROUP BY Mov.idProduto;

SELECT DISTINCT U.*
FROM Usuario U
LEFT JOIN Movimento M ON U.idUsuario = M.idUsuario AND M.tipo = 'E'
WHERE M.idUsuario IS NULL;

SELECT M.idUsuario, SUM(M.quantidade * M.precoUnitario) AS TotalEntradas
FROM Movimento M
WHERE M.tipo = 'E'
GROUP BY M.idUsuario;

SELECT M.idUsuario, SUM(M.quantidade * M.precoUnitario) AS TotalSaidas
FROM Movimento M
WHERE M.tipo = 'S'
GROUP BY M.idUsuario;
```

```
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 Mov.tipo = 'S'
GROUP BY Mov.idProduto, Prod.nome;
```

Imagem 4 - Consultas.



4 DISCUSSÕES

A análise crítica das práticas e conceitos envolvidos na implementação do banco de dados para o projeto fictício da loja permite uma compreensão mais profunda sobre diversos aspectos. Abaixo, são apresentadas as respostas às perguntas-chave, discutindo a implementação prática e as relações com o projeto desenvolvido.

4.1 Como são implementadas as diferentes cardinalidades, basicamente 1x1, 1xN ou NxN, em um banco de dados relacional?

As diferentes cardinalidades em um banco de dados relacional são implementadas por meio dos relacionamentos entre tabelas. Esses relacionamentos são fundamentais para representar como as entidades estão associadas entre si. As principais formas de cardinalidade incluem:

a) 1x1 (um para um):
   i) Nesse cenário, uma linha em uma tabela está associada a no máximo uma linha em outra tabela e vice-versa.
   ii) É implementado quando há uma chave estrangeira em uma tabela referenciando a chave primária de outra tabela, e essa relação é única.

b) 1xN (um para muitos):
   i) Aqui, uma linha em uma tabela pode estar associada a várias linhas em outra tabela, mas cada linha nesta última tabela está associada a no máximo uma linha na primeira tabela.
   ii) É comumente implementado por meio de uma chave estrangeira em uma tabela que referencia a chave primária de outra tabela.

c) NxN (muitos para muitos):
   i) Em um relacionamento muitos para muitos, várias linhas em uma tabela podem estar associadas a várias linhas em outra tabela.
   ii) Essa cardinalidade é implementada por meio de tabelas de associação (tabelas intermediárias) que contêm chaves estrangeiras referenciando as tabelas principais, criando assim um relacionamento indireto.

A escolha adequada da cardinalidade depende dos requisitos específicos do sistema e das relações que estão sendo modeladas, visando sempre garantir a integridade referencial e a consistência dos dados.

Na prática do projeto, as cardinalidades foram representadas por meio dos relacionamentos entre as tabelas. A relação entre Pessoa e PessoaFisica/PessoaJuridica exemplifica 1X1, enquanto a relação entre Movimento e Produto representa 1XN. Essas escolhas influenciam diretamente na integridade e

estrutura do banco, permitindo a representação precisa das relações entre entidades.

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, é geralmente utilizado o conceito de modelagem conhecido como tabelas por tipo (ou tabelas por subclasse). Esse modelo de *design* permite representar hierarquias de herança, comum em programação orientada a objetos, de maneira eficiente em um banco de dados relacional.

Existem três abordagens principais para implementar herança em bancos de dados relacionais:

a) Tabela única:
   i) Todos os atributos das classes derivadas são armazenados em uma única tabela.
   ii) Um campo adicional, muitas vezes chamado de discriminador, é utilizado para identificar o tipo de cada registro.

b) Tabelas por subclasse:
   i) Cada classe concreta na hierarquia é representada por uma tabela separada.
   ii) A tabela da classe base contém os atributos comuns, enquanto as tabelas das classes derivadas contêm apenas os atributos específicos delas.
   iii) Relacionamentos de chave estrangeira são estabelecidos entre as tabelas.

c) Tabelas por tipo:
   i) Similar à abordagem de tabela única, mas com uma tabela para cada classe na hierarquia.
   ii) A tabela da classe base contém todos os atributos, enquanto as tabelas das classes derivadas contêm apenas os atributos específicos, mantendo um relacionamento 1:1 com a tabela da classe base.

Para representar herança, adotou-se a abordagem de Tabela Única. A

hierarquia entre Pessoa, PessoaFisica, e PessoaJuridica foi modelada em uma tabela única, simplificando a execução de consultas envolvendo todos os tipos de entidades e mantendo a consistência do banco.

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 (SSMS) impulsiona a produtividade nas tarefas de gerenciamento de bancos de dados, proporcionando uma interface gráfica intuitiva para navegação eficiente nos elementos do banco. Seu editor SQL avançado e recursos como realce de sintaxe e sugestões automáticas aceleram o desenvolvimento e execução de consultas. A integração com sistemas de controle de versão, como o Git, facilita o gerenciamento de alterações. O SSMS também oferece ferramentas de monitoramento de desempenho, automação de tarefas administrativas e relatórios visuais, contribuindo para uma gestão eficaz do banco de dados.

4.4 Quais as diferenças no uso de *sequence* e *identity*?

As diferenças entre o uso de *sequence* e *identity* no SQL Server estão principalmente relacionadas à sua implementação e funcionalidades. A *identity* é uma propriedade associada a uma coluna específica em uma tabela e é usada para gerar automaticamente valores numéricos incrementais quando novas linhas são inseridas. Já a *sequence* é um objeto de banco de dados independente que pode ser compartilhado por várias tabelas, proporcionando uma fonte centralizada de valores sequenciais. Além disso, as *sequences* oferecem maior flexibilidade, permitindo a pré-alocação de valores e a especificação de incrementos diferentes. A escolha entre *identity* e *sequence* depende das necessidades específicas de cada aplicação e do grau de controle desejado sobre a geração de valores sequenciais.

4.5 Qual a importância das chaves estrangeiras para a consistência do banco?

As chaves estrangeiras desempenham um papel crucial na garantia da consistência e integridade dos dados em um banco. Elas estabelecem relações

entre tabelas, vinculando registros de uma tabela aos registros correspondentes em outra. A principal importância das chaves estrangeiras reside na manutenção da integridade referencial, assegurando que os dados relacionados entre tabelas permaneçam consistentes. Quando uma chave estrangeira é definida, ela impõe restrições que evitam a criação de relações órfãs, onde um registro referenciado em uma tabela é excluído sem considerar as dependências em outras tabelas. Além disso, as chaves estrangeiras podem ser usadas para impor regras de atualização em cascata, garantindo que as alterações nos registros principais sejam refletidas nos registros dependentes. Isso contribui para a integridade global do banco de dados e ajuda a evitar inconsistências que poderiam surgir se não houvesse um mecanismo para gerenciar adequadamente as relações entre as tabelas.

No contexto do projeto da loja, as chaves estrangeiras desempenham um papel vital para manter a integridade dos dados, refletindo as relações entre diferentes entidades. Por exemplo, as chaves estrangeiras na tabela Movimento (como idProduto, idPessoa e idUsuario) estabelecem conexões com registros nas tabelas Produto, Pessoa e Usuario, respectivamente. Essas relações são essenciais para rastrear detalhes específicos de produtos, pessoas e usuários associados a cada movimentação. A imposição de restrições pela integridade referencial evita inconsistências, como movimentações associadas a produtos, pessoas ou usuários inexistentes. Além disso, ao utilizar regras de atualização em cascata, é possível garantir que alterações na estrutura das tabelas principais sejam refletidas corretamente nas movimentações, mantendo a coerência nos dados. Em suma, as chaves estrangeiras são fundamentais para a consistência do banco de dados na interconexão de informações entre tabelas, contribuindo para a confiabilidade e precisão dos dados no contexto do projeto da loja.

4.6 Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

Na álgebra relacional, alguns dos operadores fundamentais incluem projeção, seleção, união, interseção, diferença e junção. No projeto da loja, esses operadores têm aplicação direta ao realizar consultas SQL. A projeção é utilizada ao selecionar colunas específicas em consultas, como ao exibir apenas os nomes dos produtos ou pessoas. A seleção é aplicada para filtrar registros com base em

critérios específicos, por exemplo, ao recuperar apenas movimentações de entrada. A união e interseção são relevantes ao combinar ou encontrar a interseção de conjuntos de resultados, o que pode ser útil em análises de movimentações totais. A diferença pode ser aplicada para identificar registros exclusivos em conjuntos de dados. A junção é crucial para combinar informações de diferentes tabelas, como ao recuperar dados completos de pessoas físicas ou jurídicas.

Quanto ao cálculo relacional, ele não é diretamente implementado em SQL, mas os operadores mencionados são traduzidos para instruções SQL correspondentes. No contexto do projeto, ao formular consultas para analisar movimentações, calcular totais e apresentar informações detalhadas sobre pessoas físicas, jurídicas, produtos e usuários, esses operadores desempenham um papel significativo.

4.7 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 agrupa as linhas do resultado com base nos valores de uma ou mais colunas. No projeto da loja, o agrupamento é evidente nas consultas que envolvem a análise de movimentações por produto, calculando totais de entradas e saídas. Por exemplo, as consultas que buscam o valor total das entradas e saídas agrupadas por produto usam a cláusula "GROUP BY Mov.idProduto". Isso permite consolidar as informações e apresentar resultados agregados para cada produto individualmente.

Um requisito obrigatório ao usar a cláusula "GROUP BY" é que todas as colunas selecionadas na lista de projeção devem ser agregadas ou incluídas na cláusula "GROUP BY". No projeto, isso é evidenciado nas consultas que calculam totais, médias ponderadas e outras métricas relacionadas ao movimento de produtos. Cada coluna na lista de projeção é tratada de forma agregada ou incluída na cláusula "GROUP BY", garantindo a conformidade com o requisito fundamental dessa operação em consultas SQL.

5 CONSIDERAÇÕES FINAIS

A prática de modelagem e implementação do banco de dados para o projeto fictício da loja foi enriquecedora, proporcionando a aplicação prática de conceitos teóricos. Consolidou-se o conhecimento adquirido na disciplina de modelagem de banco de dados, utilizando a ferramenta DB Designer Online para uma abordagem visual e interativa na concepção do banco. A modelagem conceitual desempenhou um papel crucial, garantindo uma base sólida que reflete a interconexão entre entidades e seus relacionamentos.

A etapa de implementação no SQL Server Management Studio consolidou a estrutura do banco, evidenciando a importância da consistência entre a modelagem conceitual e a prática de implementação. A inserção de dados e a execução de consultas proporcionaram uma compreensão aprofundada da funcionalidade do banco, destacando a relevância das chaves estrangeiras para a integridade dos dados.

Explorar operadores SQL, como projeção, seleção, união e junção, em consultas representativas de cenários reais, enriqueceu a experiência. O uso de sequências, identidades e a compreensão das cardinalidades proporcionaram uma visão abrangente sobre os recursos disponíveis no SQL Server.

Durante a prática, alguns desafios foram encontrados, como a definição adequada das cardinalidades e a escolha entre tabelas por tipo, exigindo análises cuidadosas para garantir representações precisas das relações entre entidades. A inserção de dados fictícios demandou atenção aos detalhes para evitar inconsistências e assegurar a validade das consultas subsequentes.

A compreensão e aplicação correta dos operadores SQL, especialmente em consultas complexas envolvendo agrupamento, cálculos ponderados e a cláusula "GROUP BY", representaram desafios superados com uma abordagem sistemática para atender aos requisitos específicos.

Para aprimorar o projeto, deve-se considerar a implementação de procedimentos armazenados, visões e gatilhos para automatizar tarefas recorrentes e facilitar a manutenção do banco. A adoção de índices adequados poderia otimizar o desempenho em consultas, especialmente em bases de dados volumosas.

A integração de medidas de segurança, como controle de acesso e criptografia, pode fortalecer a proteção dos dados. Avaliar e implementar práticas

adicionais de normalização também poderia contribuir para uma estrutura mais eficiente e escalável.

Em síntese, a prática de modelagem e implementação do banco de dados foi fundamental para o desenvolvimento de habilidades práticas em gestão de dados. A aplicação efetiva dos conceitos teóricos, aliada ao uso de ferramentas como DB Designer Online e SQL Server Management Studio, proporcionou uma compreensão abrangente e coesa do processo.

A utilização de consultas SQL para analisar e extrair informações do banco, considerando diferentes cenários, demonstrou a versatilidade e poder dessa linguagem no contexto de desenvolvimento de sistemas de informação.

Dessa forma, a prática contribuiu significativamente para a formação técnica e prática no âmbito do curso de Tecnologia em Desenvolvimento Full Stack, proporcionando uma base sólida para abordagens futuras em modelagem e implementação de bancos de dados relacionais.