INSTITUTO DE ENSINO SUPERIOR ICEV

ENGENHARIA DE SOFTWARE

GABRIEL FEITOSA MELO COELHO

SAMUEL VICTOR LUZ MARTINS

PROJETO DE BANCO DE DADOS

TERESINA

2023

GABRIEL FEITOSA MELO COELHO
SAMUEL VICTOR LUZ MARTINS

PROJETO DE BANCO DE DADOS

Trabalho apresentado ao Instituto de Ensino Superior ICEV como requisito para conclusão da disciplina de Banco de dados do curso de Engenharia de Software.

TERESINA
2023

SUMÁRIO

1. INTRODUÇÃO

Os bancos de dados são uma parte fundamental da tecnologia moderna. Eles são usados para armazenar, gerenciar e recuperar dados de uma ampla variedade de aplicações, desde sistemas de comércio eletrônico a sistemas de saúde.

Na atualidade, os bancos de dados são ainda mais importantes do que nunca. Com o aumento da quantidade de dados gerados todos os dias, os bancos de dados são essenciais para garantir que esses dados sejam armazenados de forma segura e eficiente.

Este projeto final para a disciplina de banco de dados tem como objetivo desenvolver um projeto completo implementado em MySQL de banco de dados para ecommerce de eletrônicos. O projeto irá conter entidades como:

- Clientes: Finalidade de armazenar informações sobre os clientes do ecommerce, como nome, endereço, telefone e e-mail.
- Produtos: Finalidade de armazenar informações sobre os produtos vendidos pelo e-commerce, como nome, descrição, preço e estoque.
- Pedidos: Finalidade de armazenar informações sobre os pedidos feitos pelos clientes do e-commerce, incluindo os produtos encomendados, a quantidade encomendada e o valor total do pedido.

Para acessar os scripts e diagramas, aqui descritos, na íntegra ou com melhor qualidade é recomendado o acesso repositório deste projeto disponibilizado no GitHub no link a seguir: https://github.com/gabrielfmcoelho/Ecommerce-Database.git . Além disso, o repositório fornecido contém um container Docker que prepara configurações basicas, cria e popula o banco de dados automaticamente.

2. FERRAMENTAS

Para os scripts e o DER (Diagrama de Entidade e Relacionamento) foi usado a ferramenta de gestão e manipulação de banco de dados DATAGRIP da *Jetbrains*.

Para o desenvolvimento do MER (Modelo Entidade Relacionamento) foi usado a ferramenta *brModelo.*

3. MODELO CONCEITUAL

3.1. MINIMUNDO

Uma loja de venda de eletrônicos virtual, contratou um desenvolvedor de Software para desenvolver uma solução para o acompanhamento das suas vendas. O analista de sistema da empresa fabricante realizou uma reunião de definição de escopo com o dono do ecommerce para a realização do levantamento inicial dos requisitos, onde foram entrevistados os seguintes participantes (Dono da loja, Gerente de Entregas). Essa entrevista realizada identificou os seguintes requisitos. Identificou-se que deverá conter uma acompanhamento de produtos em estoque, Cadastro de categoria de produto, Cadastro de Cliente, Cadastro de Produto, informações de contato do usuário, uma tela de movimentação (Pedido). Tela para acompanhamento do histórico de pedidos. Como também um resumo do que foi entregue ou pendente. Para identificação dos relacionamentos entre as entidades foi mapeado da seguinte forma:

- Um usuário poderá montar apenas um carrinho e cada carrinho possuirá apenas um usuário.
- Cada pedido deverá conter um título de pagamento que descreva o pagamento e o valor final a ser pago.
- Um carrinho pode conter um ou mais produtos mas, um produto só pode estar contido em um carrinho.

3.2. DICIONÁRIO DE DADOS - DD

Usuario			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do usuário para armazenar a PK	int	chave primária
CPF	documento do usuário cadeia 11 caractere	varchar(11)	não aceita valor nulo
EMAIL	e mail do usuário	varchar(50)	não aceita valor nulo
SENHA	senha do usuario	varchar(20)	não aceita valor nulo
ID_INFO_PAGAMENTO	Código do InformacoesDePagamento para armazenar a FK	int	chave estrangeira
ID_INFO_CONTATO	Código do InformacoesDeContato para armazenar a FK	int	chave estrangeira
ID_INFO_HITORICO	Código do InformacoesDeHistorico para armazenar a FK	int	chave estrangeira

InformacoesDeContato			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
NOME	nome do usuário	varchar(50)	não aceita valor nulo
SOBRENOME	sobrenome do usuário	varchar(50)	não aceita valor nulo
ENDERECO	endereço do usuário	varchar(255)	não aceita valor nulo
CIDADE	cidade do usuário	varchar(100)	não aceita valor nulo
ESTADO	estado do usuário	varchar(20)	não aceita valor nulo
CEP	cep do usuario	varchar(8)	não aceita valor nulo
TELEFONE	telefone de contato do usuário	varchar(11)	não aceita valor nulo

InformacoesDePagamento			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
ID_USUARIO	Código do usuário para armazenar a FK	int	chave estrangeira
NUMERO_CARTAO	número do cartão do usuário	varchar(20)	não aceita valor nulo
VALIDADE	validade do cartão do usuário	varchar(5)	não aceita valor nulo
CVV	código de verificação do cartão do usuário	varchar(3)	não aceita valor nulo

HistoricoUsuario			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
ID_PEDIDO	Código do pedido para armazenar a FK	int	chave estrangeira
ID_USUARIO	Código do usuário para armazenar a FK	int	chave estrangeira

Carrinho			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
ID_USUARIO	Código do usuario para armazenar a FK	int	chave estrangeira
ID_PRODUTO	Código do produto para armazenar a FK	int	chave estrangeira
QUANTIDADE	quantidade do produto no carrinho	int	
ATIVO	item que declara se o carrinho está ativo ou não	char(1)	

Pedido			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
ID_CARRINHO	Código do carrinho para armazenar a FK	int	chave estrangeira
VALOR	valor do pedido	float	não aceita valor nulo
DATA	data do pedido	datetime	
ID_INFO_PAGAMENTO	Código do InformacoesDePagamento para armazenar a FK	int	chave estrangeira

TituloDePagamento			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do TituloDePagamento para armazenar a PK	int	chave primária
ID_PEDIDO	Código do pedido para armazenar a FK	int	chave estrangeira
DATA	data do pagamento	datetime	
DESCR_PAGAMENTO	descrição do meio de pagamento	varchar(10)	
VALOR	valor total a ser pago	float	não aceita valor nulo

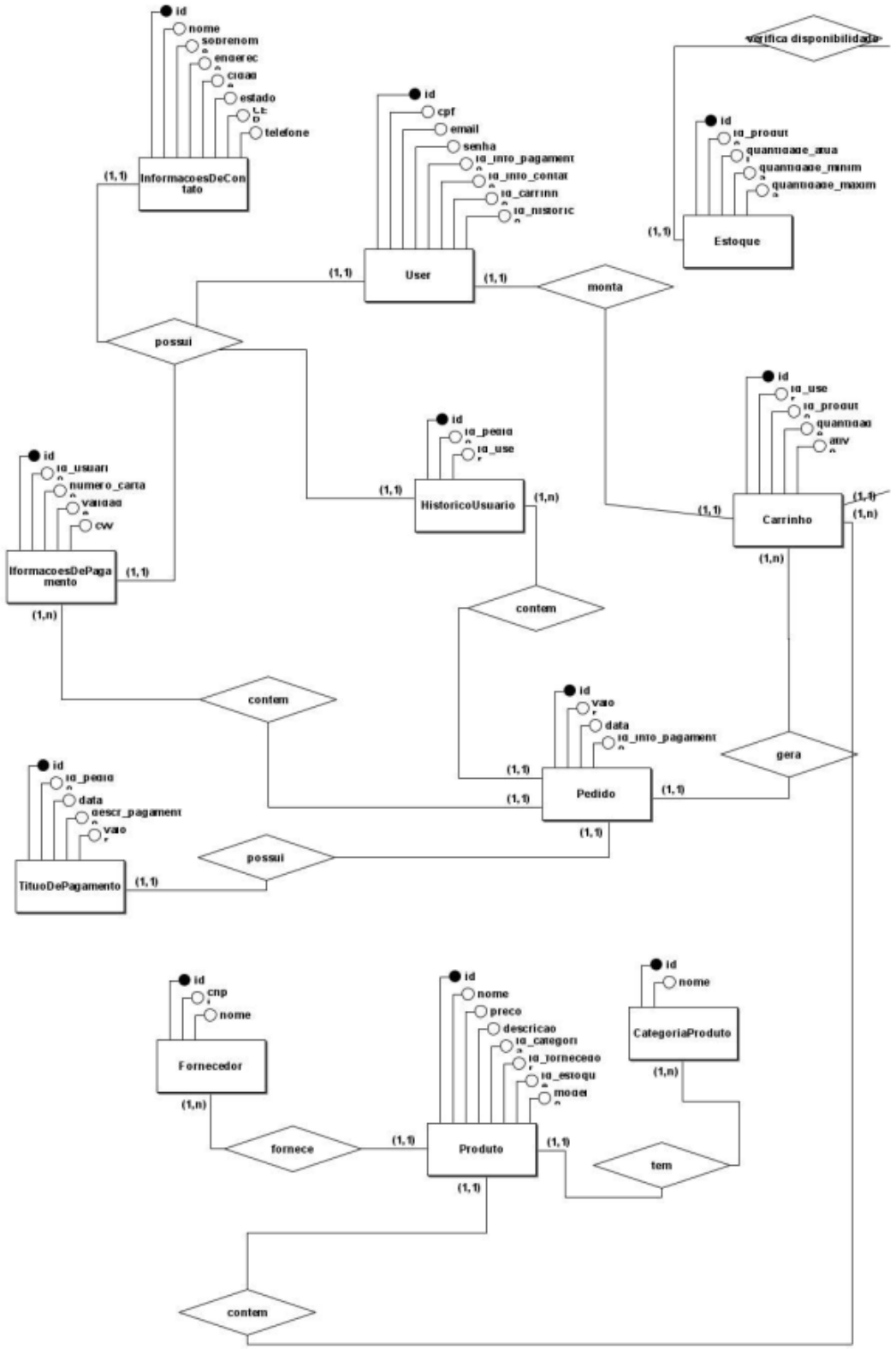
CategoriaProduto			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
NOME	categoria do produto	varchar(50)	não aceita valor nulo

Fornecedor			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
CNPJ	cnpj do fornecedor	varchar(14)	não aceita valor nulo
NOME	nome do fornecedor	varchar(50)	não aceita valor nulo

Produto			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
NOME	nome do produto	varchar(100)	não aceita valor nulo
PRECO	preço do produto	float	não aceita valor nulo
DESCRICAO	descrição do produto	text	
ID_CATEGORIA	Código do categoria para armazenar a FK	int	chave estrangeira
ID_FORNECEDOR	Código do fornecedor para armazenar a FK	int	chave estrangeira
ID_ESTOQUE	Código do estoque para armazenar a FK	int	chave estrangeira
MODELO	modelo do produto	varchar(20)	não aceita valor nulo

Estoque			
Atributo	Descrição	Domínio	Restrição do atributo
ID	Código do produto para armazenar a PK	int	chave primária
ID_PRODUTO	Código do produto para armazenar a FK	int	chave estrangeira
QUANTIDADE_ATUAL	quantidade atual do produto	int	não aceita valor nulo
QUANTIDADE_MINIMA	quantidade mínima do produto	int	não aceita valor nulo
QUANTIDADE_MAXIMA	quantidade máxima do produto	int	

3.3. ARTEFATO – MODELO DE ENTIDADE RELACIONAMENTO - MER



4. MODELO LÓGICO

4.1. NORMALIZAÇÃO DO BANCO

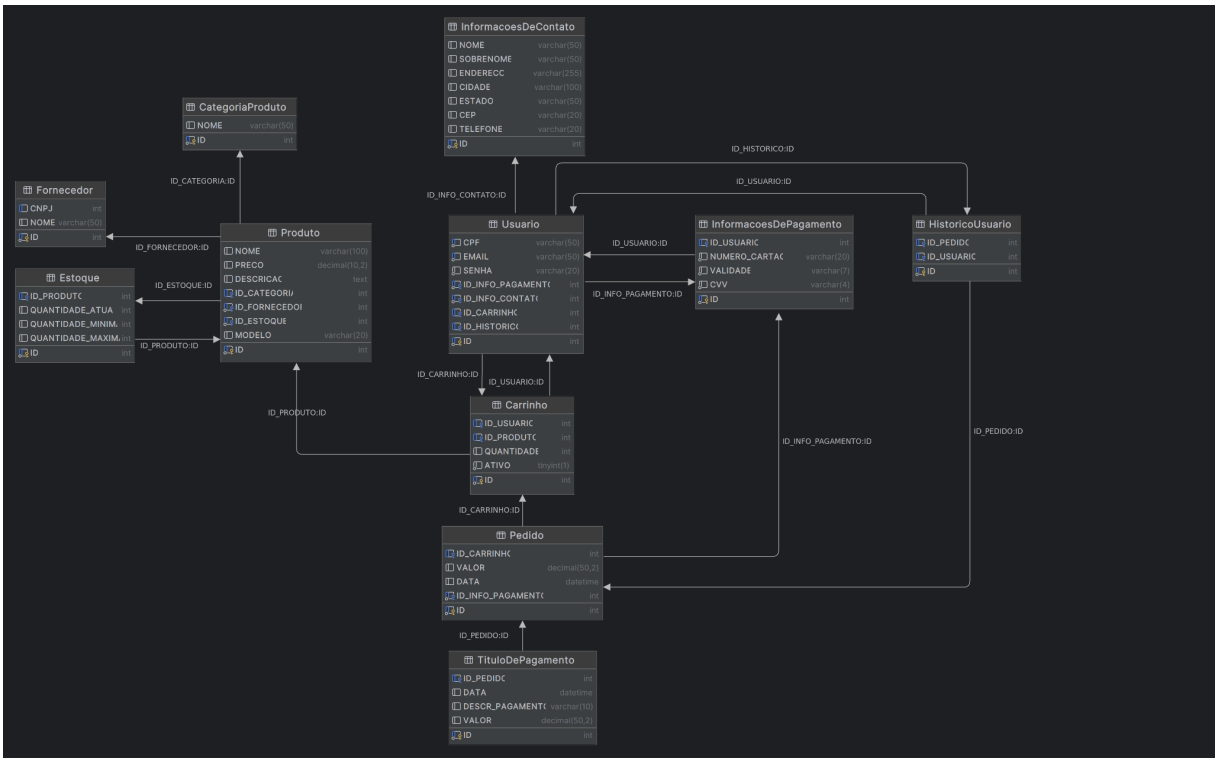
A normalização em bancos de dados é o processo que visa organizar as tabelas e relacionamentos de maneira a reduzir redundâncias e dependências, garantindo assim a integridade dos dados. A normalização segue regras específicas, conhecidas como formas normais. Dessa forma, nesta seção estão definidas as características de normalização deste banco de dados proposto.

- 1º Forma Normal
  Todas as tabelas possuem valores atômicos em suas células.
  Não há repetição de grupos de colunas.

- 2º Forma Normal
  Todas as tabelas possuem identificadores primários.
  As dependências parciais são eliminadas.

- 3º Forma Normal
  As dependências transitivas são eliminadas.

4.2. DIAGRAMA DE ENTIDADE RELACIONAL - DER



4.3. CRIAÇÃO DAS TABELAS

Nesta seção estão definidos os prints dos scripts de criação de cada uma das tabelas propostas nos diagramas iniciais, utilizando o comando DDL: CREATE.

```
create table Usuario
(
    ID                int         not null
        primary key,
    CPF               varchar(50) not null,
    EMAIL             varchar(50) not null,
    SENHA             varchar(20) not null,
    ID_INFO_PAGAMENTO int         not null,
    ID_INFO_CONTATO   int         not null,
    ID_CARRINHO       int         null,
    ID_HISTORICO      int         null,
    constraint CPF
        unique (CPF),
    constraint EMAIL
        unique (EMAIL),
    constraint Usuario_Carrinho_ID_fk
        foreign key (ID_CARRINHO) references Carrinho (ID),
    constraint Usuario_HistoricoUsuario_ID_fk
        foreign key (ID_HISTORICO) references HistoricoUsuario (ID),
    constraint Usuario_InformacoesDeContato_ID_fk
        foreign key (ID_INFO_CONTATO) references InformacoesDeContato (ID),
    constraint Usuario_InformacoesDePagamento_ID_fk
        foreign key (ID_INFO_PAGAMENTO) references InformacoesDePagamento (ID)
);
```

```
create table InformacoesDeContato
(
    ID         int          not null
        primary key,
    NOME       varchar(50)  null,
    SOBRENOME  varchar(50)  null,
    ENDERECO   varchar(255) null,
    CIDADE     varchar(100) null,
    ESTADO     varchar(50)  null,
    CEP        varchar(20)  null,
    TELEFONE   varchar(20)  null
);
```

```
create table InformacoesDePagamento
(
    ID           int         not null
        primary key,
    ID_USUARIO   int         null,
    NUMERO_CARTAO varchar(20) not null,
    VALIDADE     varchar(7)  not null,
    CVV          varchar(4)  not null,
    constraint InformacoesDePagamento_ibfk_1
        foreign key (ID_USUARIO) references Usuario (ID)
);

create index ID_USUARIO
    on InformacoesDePagamento (ID_USUARIO);
```

```sql
create table TituloDePagamento
(
    ID              int             not null
        primary key,
    ID_PEDIDO       int             null,
    DATA            datetime        null,
    DESCR_PAGAMENTO varchar(10)     null,
    VALOR           decimal(50, 2) null,
    constraint TituloDePagamento_ibfk_1
        foreign key (ID_PEDIDO) references Pedido (ID)
);

create index ID_PEDIDO
    on TituloDePagamento (ID_PEDIDO);
```

```sql
create table Produto
(
    ID            int            not null
        primary key,
    NOME          varchar(100)   null,
    PRECO         decimal(10, 2) null,
    DESCRICAO     text           null,
    ID_CATEGORIA  int            null,
    ID_FORNECEDOR int            not null,
    ID_ESTOQUE    int            not null,
    MODELO        varchar(20)    null,
    constraint Produto_Estoque_ID_fk
        foreign key (ID_ESTOQUE) references Estoque (ID),
    constraint Produto_Fornecedor_ID_fk
        foreign key (ID_FORNECEDOR) references Fornecedor (ID),
    constraint Produto_ibfk_1
        foreign key (ID_CATEGORIA) references CategoriaProduto (ID)
);

create index ID_CATEGORIA
    on Produto (ID_CATEGORIA);
```

```sql
create table Pedido
(
    ID              int             not null
        primary key,
    ID_CARRINHO     int             null,
    VALOR           decimal(50, 2) null,
    DATA            datetime        null,
    ID_INFO_PAGAMENTO int           not null,
    constraint Pedido_InformacoesDePagamento_ID_fk
        foreign key (ID_INFO_PAGAMENTO) references InformacoesDePagamento (ID),
    constraint Pedido_ibfk_1
        foreign key (ID_CARRINHO) references Carrinho (ID)
);

create index ID_CARRINHO
    on Pedido (ID_CARRINHO);
```

```sql
create table HistoricoUsuario
(
    ID         int not null
        primary key,
    ID_PEDIDO  int null,
    ID_USUARIO int null,
    constraint HistoricoUsuario_ibfk_1
        foreign key (ID_PEDIDO) references Pedido (ID),
    constraint HistoricoUsuario_ibfk_2
        foreign key (ID_USUARIO) references Usuario (ID)
);

create index ID_PEDIDO
    on HistoricoUsuario (ID_PEDIDO);

create index ID_USUARIO
    on HistoricoUsuario (ID_USUARIO);
```

```sql
create table Fornecedor
(
    ID   int          not null
        primary key,
    CNPJ int          null,
    NOME varchar(50) null,
    constraint CNPJ
        unique (CNPJ)
);
```

```sql
create table Estoque
(
    ID                int not null
        primary key,
    ID_PRODUTO        int null,
    QUANTIDADE_ATUAL  int null,
    QUANTIDADE_MINIMA int null,
    QUANTIDADE_MAXIMA int null,
    constraint Estoque_ibfk_1
        foreign key (ID_PRODUTO) references Produto (ID)
);

create index ID_PRODUTO
    on Estoque (ID_PRODUTO);
```

```sql
create table CategoriaProduto
(
    ID   int         not null
        primary key,
    NOME varchar(50) null
);
```

```sql
create table Carrinho
(
    ID         int        not null
        primary key,
    ID_USUARIO int        null,
    ID_PRODUTO int        null,
    QUANTIDADE int        null,
    ATIVO      tinyint(1) not null,
    constraint Carrinho_ibfk_1
        foreign key (ID_USUARIO) references Usuario (ID),
    constraint Carrinho_ibfk_2
        foreign key (ID_PRODUTO) references Produto (ID)
);

create index ID_PRODUTO
    on Carrinho (ID_PRODUTO);

create index ID_USUARIO
    on Carrinho (ID_USUARIO);
```

5. MODELO FÍSICO

5.1. POVOAMENTO

Nesta seção estão descritos os scripts de povoamento de cada tabela definida no capítulo anterior deste documento, para este objetivo foi utilizado o comando DML: INSERT.

1. Carrinho.sql
   INSERT INTO Carrinho (ID, ID_USUARIO, ID_PRODUTO, QUANTIDADE, ATIVO) VALUES
   (1, 1, 1, 2, 1),
   (2, 2, 3, 1, 1),
   (3, 3, 5, 3, 1),
   (4, 4, 2, 1, 1),
   (5, 5, 4, 2, 1);

2. CategoriaProduto.sql
   INSERT INTO CategoriaProduto (ID, NOME) VALUES
   (1, 'Smartphones'),
   (2, 'Laptops'),
   (3, 'Headphones'),
   (4, 'Câmeras'),
   (5, 'Tablets');

3. Estoque.sql
   INSERT INTO Estoque (ID, ID_PRODUTO, QUANTIDADE_ATUAL, QUANTIDADE_MINIMA, QUANTIDADE_MAXIMA) VALUES
   (1, 1, 50, 10, 100),
   (2, 2, 30, 5, 50),
   (3, 3, 20, 8, 80),
   (4, 4, 15, 3, 30),
   (5, 5, 40, 15, 150);

4. Fornecedor.sql
   INSERT INTO Fornecedor (ID, CNPJ, NOME) VALUES
   (1, 123456789, 'Fornecedor A'),
   (2, 987654321, 'Fornecedor B'),
   (3, 555555555, 'Fornecedor C'),
   (4, 111111111, 'Fornecedor D'),
   (5, 999999999, 'Fornecedor E');

5. HistoricoUsuario.sql
   INSERT INTO HistoricoUsuario (ID, ID_PEDIDO, ID_USUARIO) VALUES
   (1, 1, 1),
   (2, 2, 2),
   (3, 3, 3),
   (4, 4, 4),
   (5, 5, 5);

6. InformaçõesDeContato.sql
   INSERT INTO InformacoesDeContato (ID, NOME, SOBRENOME, ENDERECO, CIDADE, ESTADO, CEP, TELEFONE) VALUES
   (1, 'João', 'Silva', 'Rua A, 123', 'Cidade A', 'Estado A', '12345-678', '123-456-7890'),
   (2, 'Maria', 'Santos', 'Rua B, 456', 'Cidade B', 'Estado B', '98765-432', '987-654-3210'),
   (3, 'Pedro', 'Oliveira', 'Rua C, 789', 'Cidade C', 'Estado C', '54321-876', '543-210-9876'),
   (4, 'Ana', 'Souza', 'Rua D, 987', 'Cidade D', 'Estado D', '67890-123', '678-901-2345'),
   (5, 'Lucas', 'Pereira', 'Rua E, 654', 'Cidade E', 'Estado E', '87654-321', '876-543-2109');

7. InformaçõesDePagamento.sql
   INSERT INTO InformacoesDePagamento (ID, ID_USUARIO, NUMERO_CARTAO, VALIDADE, CVV) VALUES

```
(1, 1, '1234-5678-9012-3456', '12/24', '123'),
(2, 2, '2345-6789-0123-4567', '01/23', '456'),
(3, 3, '3456-7890-1234-5678', '06/25', '789'),
(4, 4, '4567-8901-2345-6789', '09/22', '012'),
(5, 5, '5678-9012-3456-7890', '03/24', '345');
```

8. Pedido.sql
```
INSERT INTO Pedido (ID, ID_CARRINHO, VALOR, DATA, ID_INFO_PAGAMENTO) VALUES
(1, 1, 1599.98, '2023-01-15 10:30:00', 1),
(2, 2, 99.99, '2023-02-20 15:45:00', 2),
(3, 3, 1499.97, '2023-03-10 08:00:00', 3),
(4, 4, 1299.99, '2023-04-05 12:15:00', 4),
(5, 5, 999.98, '2023-05-02 18:20:00', 5);
```

9. Produto.sql
```
INSERT INTO Produto (ID, NOME, PRECO, DESCRICAO, ID_CATEGORIA,
ID_FORNECEDOR, ID_ESTOQUE, MODELO) VALUES
(1, 'Smartphone X', 799.99, 'Tela 6", 128GB, Câmera 12MP', 1, 1, 1, 'X123'),
(2, 'Laptop Y', 1299.99, 'Intel i7, 16GB RAM, 512GB SSD', 2, 2, 2, 'Y456'),
(3, 'Headphone Z', 99.99, 'Cancelamento de ruído, Bluetooth', 3, 3, 3, 'Z789'),
(4, 'Câmera A', 499.99, 'Sensor Full Frame, 4K Video', 4, 4, 4, 'A321'),
(5, 'Tablet B', 499.99, 'Tela 10", 64GB, Caneta Inclusa', 5, 5, 5, 'B654');
```

10. TituloDePagamento.sql
```
INSERT INTO TituloDePagamento (ID, ID_PEDIDO, DATA, DESCR_PAGAMENTO, VALOR)
VALUES
(1, 1, '2023-01-16 10:30:00', 'Cartão de Crédito', 1599.98),
(2, 2, '2023-02-21 15:45:00', 'Cartão de Crédito', 99.99),
(3, 3, '2023-03-11 08:00:00', 'Cartão de Crédito', 1499.97),
(4, 4, '2023-04-06 12:15:00', 'Cartão de Crédito', 1299.99),
(5, 5, '2023-05-03 18:20:00', 'Cartão de Crédito', 999.98);
```

11. Usuario.sql
```
INSERT INTO Usuario (ID, CPF, EMAIL, SENHA, ID_INFO_PAGAMENTO,
ID_INFO_CONTATO, ID_CARRINHO, ID_HISTORICO) VALUES
(1, '123.456.789-01', 'joao@email.com', 'senha123', 1, 1, 1, 1),
(2, '234.567.890-12', 'maria@email.com', 'senha456', 2, 2, 2, 2),
(3, '345.678.901-23', 'pedro@email.com', 'senha789', 3, 3, 3, 3),
(4, '456.789.012-34', 'ana@email.com', 'senha012', 4, 4, 4, 4),
(5, '567.890.123-45', 'lucas@email.com', 'senha345', 5, 5, 5, 5);
```

5.2. CONSULTAS SQL E NOTAÇÕES ALGÉBRICAS

Nesta seção estão descritas 10 consultas SQL demonstrando a aplicação de diversos comandos DQL para a extração de insights. Para cada consulta é fornecido o código em SQL, assim como, a respectiva notação em álgebra relacional (em markdown para otimização da escrita em documentação).

1. Encontrar a quantidade atual de cada produto no estoque.sql
```
SELECT Produto.NOME, Estoque.QUANTIDADE_ATUAL
FROM Produto
JOIN Estoque ON Produto.ID_ESTOQUE = Estoque.ID;

-- \(\pi_{NOME, QUANTIDADE_ATUAL}(Produto \bowtie_{Produto.ID_ESTOQUE =
Estoque.ID} Estoque)\)
```

2. Encontrar o número de produtos em cada faixa de preço (por exemplo, 0-100, 100-200, etc.).sql
```
SELECT
   CASE
```

```
        WHEN Produto.PRECO BETWEEN 0 AND 100 THEN '0-100'
        WHEN Produto.PRECO BETWEEN 101 AND 200 THEN '101-200'
        WHEN Produto.PRECO BETWEEN 201 AND 300 THEN '201-300'
        ELSE 'Mais de 300'
    END AS FaixaDePreco,
    COUNT(Produto.ID) AS TotalProdutos
FROM Produto
GROUP BY FaixaDePreco;

-- \(\pi_{FaixaDePreco, TotalProdutos}(\gamma_{COUNT(ID)}(\sigma_{\top}(Produto)
\ltimes_{\text{agrupar por FaixaDePreco}}))\)
```

3. Encontrar o produto mais caro de cada categoria.sql

```
SELECT CategoriaProduto.NOME, MAX(Produto.PRECO) AS ProdutoMaisCaro
FROM CategoriaProduto
JOIN Produto ON CategoriaProduto.ID = Produto.ID_CATEGORIA
GROUP BY CategoriaProduto.NOME;

-- \(\pi_{NOME, ProdutoMaisCaro}(\gamma_{MAX(PRECO)}(Produto
\bowtie_{CategoriaProduto.ID = Produto.ID_CATEGORIA} CategoriaProduto))\)
```

4. Encontrar o total de produtos em cada categoria.sql

```
SELECT CategoriaProduto.NOME, COUNT(Produto.ID) AS TotalProdutos
FROM CategoriaProduto
LEFT JOIN Produto ON CategoriaProduto.ID = Produto.ID_CATEGORIA
GROUP BY CategoriaProduto.NOME;

-- \(\pi_{NOME, TotalProdutos}(\gamma_{COUNT(ID)}(\sigma_{\top}(CategoriaProduto)
\ltimes_{\text{agrupar por NOME}} Produto))\)
```

5. Encontrar o total gasto por cada usuário.sql

```
SELECT Usuario.ID, Usuario.EMAIL, SUM(Pedido.VALOR) AS TotalGasto
FROM Usuario
JOIN HistoricoUsuario ON Usuario.ID = HistoricoUsuario.ID_USUARIO
JOIN Pedido ON HistoricoUsuario.ID_PEDIDO = Pedido.ID
GROUP BY Usuario.ID, Usuario.EMAIL;
-- \(\pi_{ID, EMAIL, TotalGasto}(\rho_{ID_USUARIO \leftarrow ID}(Usuario \bowtie_{Usuario.ID
= HistoricoUsuario.ID_USUARIO} HistoricoUsuario \bowtie_{HistoricoUsuario.ID_PEDIDO =
Pedido.ID} Pedido))\)
```

6. Listar todos os pedidos feitos em uma data específica.sql

```
SELECT Pedido.ID, Usuario.EMAIL, Pedido.VALOR, Pedido.DATA
FROM Pedido
JOIN HistoricoUsuario ON Pedido.ID_CARRINHO = HistoricoUsuario.ID_PEDIDO
JOIN Usuario ON HistoricoUsuario.ID_USUARIO = Usuario.ID
WHERE Pedido.DATA = '2023-02-20 15:45:00';

-- \(\pi_{NOME, ProdutoMaisCaro}(\gamma_{MAX(PRECO)}(Produto
\bowtie_{CategoriaProduto.ID = Produto.ID_CATEGORIA} CategoriaProduto))\)
```

7. Listar todos os produtos com seus preços.sql

```
SELECT Produto.NOME, Produto.PRECO
FROM Produto;

-- \(\pi_{NOME, PRECO}(Produto)\)
```

8. Listar todos os produtos ordenados por preço, em ordem decrescente.sql

```
SELECT Produto.NOME, Produto.PRECO
FROM Produto
ORDER BY Produto.PRECO DESC;
```

-- \(\pi_{NOME, PRECO}(\sigma_{\top}(Produto) \ltimes_{\text{ordenar por PRECO DESC}})\)

9. Listar todos os usuários que ainda têm produtos no carrinho.sql
```
SELECT DISTINCT Usuario.ID, Usuario.EMAIL
FROM Usuario
JOIN Carrinho ON Usuario.ID = Carrinho.ID_USUARIO
WHERE Carrinho.ATIVO = 1;
```
-- \(\pi_{ID, EMAIL}(\sigma_{ATIVO=1}(Usuario \bowtie_{Usuario.ID = Carrinho.ID_USUARIO} Carrinho))\)

10. Listar todos os usuários que compraram produtos da categoria 'Laptops'.sql
```
SELECT DISTINCT Usuario.ID, Usuario.EMAIL
FROM Usuario
JOIN HistoricoUsuario ON Usuario.ID = HistoricoUsuario.ID_USUARIO
JOIN Pedido ON HistoricoUsuario.ID_PEDIDO = Pedido.ID
JOIN Carrinho ON Pedido.ID_CARRINHO = Carrinho.ID
JOIN Produto ON Carrinho.ID_PRODUTO = Produto.ID
JOIN CategoriaProduto ON Produto.ID_CATEGORIA = CategoriaProduto.ID
WHERE CategoriaProduto.NOME = 'Laptops';
```
-- \(\pi_{ID, EMAIL}(\sigma_{NOME='Laptops'}(Usuario \bowtie_{Usuario.ID = HistoricoUsuario.ID_USUARIO} HistoricoUsuario \bowtie_{HistoricoUsuario.ID_PEDIDO = Pedido.ID} Pedido \bowtie_{Pedido.ID_CARRINHO = Carrinho.ID} Carrinho \bowtie_{Carrinho.ID_PRODUTO = Produto.ID} Produto \bowtie_{Produto.ID_CATEGORIA = CategoriaProduto.ID} CategoriaProduto))\)

6.CONSIDERAÇÕES FINAIS

Ao longo do desenvolvimento deste projeto de banco de dados para uma empresa fictícia de e-commerce de eletrônicos, foram abordados diversos aspectos relacionados à modelagem, estrutura e funcionalidades do sistema proposto. O objetivo principal foi criar um banco de dados eficiente, que atenda às necessidades da suposta empresa, garantindo a integridade dos dados e facilitando a consulta e manipulação das informações.

Durante o processo de design, foram utilizados os princípios da normalização para organizar as tabelas, minimizar redundâncias e eliminar dependências indesejadas. A implementação seguiu as práticas recomendadas, como a definição de chaves primárias e estrangeiras, garantindo a consistência e a integridade referencial.

Pontos positivos do projeto incluem a conformidade com a Primeira e Segunda Formas Normais (1NF e 2NF), onde os valores estão atomizados, as dependências parciais foram eliminadas, identificadores primários estão presentes em todas as tabelas e as consultas SQL ricas em diversos exemplos de comandos com suas respectivas notações algébricas. Contudo, há alguns casos em que a Terceira Forma Normal (3NF) não foi completamente atingida ou poderia ser melhorada.

A documentação fornecida serve como guia para o entendimento da estrutura do banco de dados, suas relações, a lógica de armazenamento e consulta de informações.

Em conclusão, este projeto proporcionou uma experiência valiosa no design e implementação de um banco de dados para um cenário de e-commerce, destacando a importância da normalização e boas práticas de modelagem para garantir a qualidade e eficácia do sistema.