Estácio

Campus: Polo Centro - Jequié (BA)
Curso: Desenvolvimento fullstack

Disciplina: Nível 1 Iniciando caminho pelo Java
turma: 2023.1
Semestre: 2024.1

Aluno: Israel dos Santos Hamdan D’Araujo

Título da pratica: Vamos manter informações
Objetivos da prática

1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
7. SQL, na plataforma do SQL Server.

Códigos usados para criação do Banco de Dados:

1. Tabela pessoa

```
CREATE TABLE Pessoas(
  idPessoa INT  NOT NULL,
  Nome VARCHAR(255) NOT NULL,
  Logradouro VARCHAR(255),
  Cidade VARCHAR(255),
  Estado CHAR(2)  NOT NULL,
  Telefone VARCHAR(11)  NOT NULL,
  Email VARCHAR(255)  NOT NULL
  PRIMARY KEY([idPessoa])
  CONSTRAINT DF_Pessoa_idPessoa DEFAULT (NEXT VALUE FOR seq_pessoa)
);
```

A tabela pessoa armazena os dados das pessoas físicas e jurídicas, mas ela armazena os dados gerais, não constando CPF e CNPJ

2. Tabela de pessoas físicas

```
CREATE TABLE PessoasFisicas (
  idPessoaFisica INT PRIMARY KEY NOT NULL,
  CPF VARCHAR(11) UNIQUE NOT NULL,
  CONSTRAINT FK_PessoaFisica_Pessoa FOREIGN KEY (idPessoaFisica) REFERENCES Pessoas(idPessoa)
);
```

   Esta tabela relaciona o id da pessoa física com a tabela pessoa e armazena o CPF da pessoa física, é sua característica principal, o Constriant serve para dizer que o id da pessoa física referência o id da pessoa

3. Tabela de pessoas jurídicas

```
CREATE TABLE PessoasJuridicas (
  idPJ INT PRIMARY KEY  NOT NULL,
  CNPJ VARCHAR(11) UNIQUE NOT NULL,
  CONSTRAINT FK_PessoaJuridica_Pessoa FOREIGN KEY (idPJ) REFERENCES Pessoas(idPessoa)
);
```

   Esta tabela relaciona o id da pessoa jurídica com a tabela pessoa e armazena o CNPJ da pessoa jurídica, é sua característica principal, o Constriant serve para dizer que o id da pessoa jurídica referência o id da pessoa

4. Tabela de usuários

```
CREATE TABLE Usuarios (
  idUser INT PRIMARY KEY  NOT NULL,
  loginUser VARCHAR(20) UNIQUE NOT NULL,
  Senha VARCHAR(30) NOT NULL,
  CONSTRAINT C_User Foreign key (idUser) REFERENCES Pessoas(idPessoa)
);
```

A tabela de usuários serve somente para armazenar os operadores do sistema, o id dele é uma referência ao id pessoa

5. Tabela de produtos

```
CREATE TABLE Produtos (
  idProduto INT PRIMARY KEY  NOT NULL,
  nomeProduto VARCHAR(255) NOT NULL,
  quantidade INT NOT NULL,
  preco NUMERIC(10,2) NOT NULL
);
```

A tabela de produtos serve para guardar os produtos do nosso estoque

6. Tabela de movimentos

A tabla de movimentos serve para registrar as operações de entrada e saída do sistema

```
CREATE TABLE Movimentos (
  idMovimento INT PRIMARY KEY NOT NULL,
  tipo CHAR(1) NOT NULL,
  quantidade INT NOT NULL,
  precoUnitario MONEY NOT NULL,
  FK_idUsuario INT,
  FK_idPessoa INT,
  FK_idProduto INT,
  CONSTRAINT C_fk_idUsuario FOREIGN KEY (FK_idUsuario) REFERENCES Usuarios(idUser),
  CONSTRAINT C_fk_idPessoa FOREIGN KEY (FK_idPessoa) REFERENCES Pessoas(idPessoa),
  CONSTRAINT C_fk_idProduto FOREIGN KEY (FK_idProduto) REFERENCES Produtos(idProduto)
);
```

E faz referência aos ids do usuário, da pessoa e do produto, afinal sempre vai ter que ter uma pessoa comprando, outra vendendo

7. Alimentando a tabela de pessoas

```
INSERT INTO Pessoas (idPessoa, Nome, Logradouro, Cidade,  Estado, Telefone, Email)
VALUES
  (1, 'João Silva','Rua', 'São Luis', 'Ma', '11999999999', 'joaosilva@email.com'),
  (2, 'Maria Souza', 'Avenida', 'Petropoles', 'RJ', '21999999999', 'mariasouza@email.com'),
  (3, 'Pedro Oliveira','Condominio', 'Belo Horizonte', 'MG', '31999999999', 'pedrooliveira@email.com'),
  (4, 'Budwiser','Rua', 'São Luis', 'Ma', '11999999999', 'budwiser@email.com'),
  (5, 'Ambev', 'Avenida', 'Petropoles', 'RJ', '21999999999', 'ambev@email.com'),
  (6, 'Heiniken','Parque', 'Belo Horizonte', 'MG', '31999999999', 'Heiniken@email.com'),
  (7, 'Corona','Parque', 'Belo Horizonte', 'MG', '31999999999', 'Corona@email.com'),
  (8, 'Coca-cola','Fabrica', 'Uberlandia ', 'MG', '31336541325', 'cocacolabr@gmail.com');
```

Isso faz com que a tabela de pessoas seja populada por pessoas físicas e jurídicas

8. Alimentando as tabelas de pessoas jurídicas com CNPJ e pessoas físicas com CPF fazendo a relação dos ids de cada uma

```
INSERT INTO PessoasFisicas (idPessoaFisica , CPF)
VALUES
  (1, '41412651561'),
  (2, '51651151516'),
  (3,'7784116516');


 --inserindo pessoas juridicas
INSERT INTO PessoasJuridicas (idPJ, CNPJ)

VALUES
  (4, '2151321531'),
  (5, '1232151546'),
  (6, '1516541351'),
  (7, '5531546535'),
  (8, '58796413561');
```

9. Alimentando a tabela de usuários

```
INSERT INTO Usuarios (idUser, loginUser, Senha)

Values
    (1, 'Op1', 'op1'),
    (2, 'Op2', 'op2')
```

10. Alimentando a tabela de produtos

```
--inserindo os dados do produto
INSERT INTO Produtos (idProduto, nomeProduto, quantidade, preco)


Values
    (1, 'Cerveja Budwiser', 800, 5.00),
    (2, 'Cerveja Heiniken', 800,7.80),
    (3, 'Cerveja Heiniken zero álcool', 800, 7.80),
    (4, 'Cerveja Corona', 800, 8.00),
```

No nosso caso é uma distribuidora de bebidas

11. Alimentando a tabela de Movimentos

```
INSERT INTO Movimentos (idMovimento, tipo, quantidade, precoUnitario, FK_idUsuario, FK_idPessoa, FK_idProduto)
VALUES
    (1, 'S', 10, 5.00, 1, 1, 1),
    (2, 'E', 100, 2.00, 1, 7, 4);
```

Análise e conclusão.

1. **Como são implementadas as diferentes cardinalidades, basicamente 1x1, 1xN ou NxN em um banco de dados relacional?**
    a. Quando vamos relacionar dados em um banco precisamos nos atentar a cardinalidade que desejada
        i. 1x1 = É uma relação direta de um dado com um outro dado, por exemplo, eu empresto um livro para você, o livro é meu, mas no momento ele está em sua posse, esse relacionamento também é chamado de **um para um**
        ii. 1xN = É quando dado está relacionado com vários outros dados, esse relacionamento também é chamado de **um para muitos.** Por exemplo, é como se eu emprestasse um livro que é meu para um grupo de amigos próximos
        iii. NxN = É quando muitos dados estão relacionados a muitos dados, esse relacionamento também é chamado de **muitos para muitos**. Um exemplo disso é uma biblioteca, onde há uma tabela de livros e uma tabela de tomadores, onde um livro está vinculado a vários tomadores
2. **Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?**
    a. Para representar a herança em bancos de dados tem 3 estratégias principais:
        i. *Tabela única*: As classes filhas compartilham a mesma tabela no banco de dados com o campo indicando onde acontece o registro
        ii. *Tabelas separadas:* Cada classe e tabela e a conexão é feita por chaves estrangeiras (*Forigen key -FK)*

iii. Tabela de junção: Cada classe tem sua própria tabela no banco e o relacionamento é feito por uma tabela de junção para mapear os relacionamentos entre elas

3. Vantagens do SQL Server Management Studio?
    a. Ter uma ferramenta própria para trabalhar com Bancos de dados facilita muito o dia a dia do profissional de banco de dados, o chamado DBA (Data Base Administrator), pois fornece opções de monitoramento e otimização de desempenho do banco de dados, ter interface gráfica fácil e usar e além de poder escrever as consultas SQL direto no próprio banco, sem necessidade de uso de terminais ou outra coisa