Universidade Estácio de Sá
Campus Santa Cruz

Relatório Discente de Acompanhamento

Nome: Tamires de Souza Alves
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações?
Número da Turma: 9001
Semestre Letivo: 3º Semestre

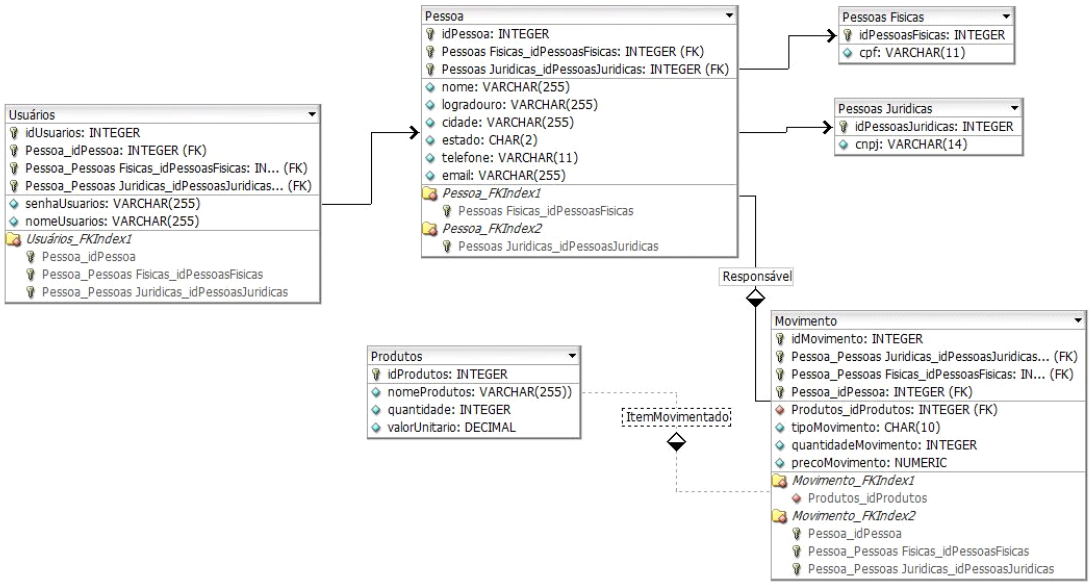
GitHub: https://github.com/tamiresalves024/MissaoPratica_Nivel2_Mundo3.git

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

Objetivo da Prática

Modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma SQL Server.

Procedimento 1 – Criando o banco de dados



Código

```
-- Criação da tabela de usuários
CREATE TABLE Usuarios (
    id_usuario SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    senha VARCHAR(100)
);

-- Criação da tabela de pessoas físicas
CREATE TABLE PessoaFisica (
    id_pf SERIAL PRIMARY KEY,
    id_usuario INTEGER UNIQUE REFERENCES Usuarios(id_usuario),
    cpf VARCHAR(14) UNIQUE,
    nome VARCHAR(100),
```

```
    endereco VARCHAR(200),
    telefone VARCHAR(20)
);

-- Criação da tabela de pessoas jurídicas
CREATE TABLE PessoaJuridica (
    id_pj SERIAL PRIMARY KEY,
    id_usuario INTEGER UNIQUE REFERENCES Usuarios(id_usuario),
    cnpj VARCHAR(18) UNIQUE,
    nome_empresa VARCHAR(100),
    endereco VARCHAR(200),
    telefone VARCHAR(20)
);

-- Criação da tabela de produtos
CREATE TABLE Produtos (
    id_produto SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    quantidade INTEGER,
    preco_venda DECIMAL(10, 2)
);

-- Criação da tabela de movimentos de compra
CREATE TABLE MovimentoCompra (
    id_movimento SERIAL PRIMARY KEY,
    id_usuario INTEGER REFERENCES Usuarios(id_usuario),
    id_produto INTEGER REFERENCES Produtos(id_produto),
    id_pessoa_juridica INTEGER REFERENCES PessoaJuridica(id_pj),
    quantidade INTEGER,
    preco_unitario DECIMAL(10, 2),
    data_movimento TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Criação da tabela de movimentos de venda
CREATE TABLE MovimentoVenda (
    id_movimento SERIAL PRIMARY KEY,
    id_usuario INTEGER REFERENCES Usuarios(id_usuario),
    id_produto INTEGER REFERENCES Produtos(id_produto),
    id_pessoa_fisica INTEGER REFERENCES PessoaFisica(id_pf),
    quantidade INTEGER,
    valor_total DECIMAL(10, 2),
    data_movimento TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Criação de sequência para geração de identificadores de pessoa
CREATE SEQUENCE pessoa_id_seq START 1;

-- Atribuição de identificadores para PessoaFisica e PessoaJuridica
```

```
ALTER TABLE PessoaFisica ALTER COLUMN id_pf SET DEFAULT
nextval('pessoa_id_seq');
ALTER TABLE PessoaJuridica ALTER COLUMN id_pj SET DEFAULT
nextval('pessoa_id_seq');
```

Análise e Conclusão

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

1 para 1 (1:1): Implementada utilizando uma chave estrangeira na tabela secundária que referencia a chave primária da tabela principal.

1 para N (1:N): Realizada através da inclusão de uma chave estrangeira na tabela "muitos" que referencia a chave primária da tabela "um".

N para N (N:N): É implementada por meio de uma tabela de associação (tabela intermediária) que contém as chaves estrangeiras de ambas as tabelas "N", estabelecendo assim uma relação indireta entre elas.

Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

Para representar o conceito de herança em bancos de dados relacionais, o tipo de relacionamento frequentemente utilizado é o "Mapeamento de Tabelas por Hierarquia de Classes" (ou "Table-Per-Hierarchy" - TPH). Nesse modelo de herança, uma única tabela é usada para representar múltiplos tipos de entidades ou classes.

Resumidamente, a TPH cria uma tabela que abrange todos os atributos de todas as classes envolvidas na hierarquia de herança. Um campo adicional (geralmente chamado de discriminador) é utilizado para identificar a qual classe pertence cada registro na tabela.

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) oferece várias ferramentas e recursos que melhoram a produtividade no gerenciamento de bancos de dados:

Interface Integrada: Oferece uma interface unificada para gerenciar bancos de dados, consultas, procedimentos armazenados, segurança, entre outros, facilitando a navegação e organização das tarefas.

Editor de Consultas Avançado: Possui um editor robusto que oferece realce de sintaxe, autocompletar de código, formatação automática e opções de depuração, agilizando o desenvolvimento e a execução de consultas SQL.

Ferramentas de Desempenho: Fornece ferramentas para análise de desempenho, permitindo identificar gargalos, otimizar consultas e monitorar o uso de recursos do servidor.

Administração de Segurança: Facilita a administração de permissões, login e usuários, possibilitando o controle fino de acesso aos dados.

Automatização de Tarefas: Permite a criação e execução de scripts para automatizar tarefas rotineiras, como backups, manutenção e implementação de alterações no banco de dados.

Integração com outras Ferramentas Microsoft: Integra-se com outras ferramentas da Microsoft, como o Visual Studio, Azure, PowerShell, entre outros, proporcionando um ecossistema mais amplo para o gerenciamento e desenvolvimento de bancos de dados.

Esses recursos combinados ajudam os administradores e desenvolvedores de banco de dados a serem mais eficientes na gestão, manutenção e desenvolvimento de sistemas utilizando o SQL Server.

Procedimento 2 – Alimentando a base

Códigos do Procedimento 2
Análise e Conclusão – Procedimento 2

```
--Inserindo dados
INSERT INTO Usuario (UsuarioID, Nome, Email, Senha)
VALUES
    (1, 'Primeiro Usuário', 'primeiro@email.com', 'senha1'),
    (2, 'Segundo Usuário', 'segundo@email.com', 'senha2');


INSERT INTO Produtos (ProdutosID, Nome, Quantidade, PrecoVenda)
VALUES
    (1, 'Produto1', 8, 21),
    (2, 'Produto2', 10, 17);

--Consultas
Select * from Usuario;
Select *from Produtos;
Select * from PessoaFisica;
Select * from PessoaJuridica;
Select * from Pessoa;
select * from MovimentoCompra;

--Sequence
create sequence seqID as numeric start with 100 increment by 1;

-- Preenchendo tabela Pessoa com alguns exemplos
```

```
insert into Pessoa (PessoaID, Nome, Endereco, Telefone)
values (next value for seqID, 'Alice', 'Rua das Flores', '0112345678');

insert into Pessoa (PessoaID, Nome, Endereco, Telefone)
values (next value for seqID, 'Bob', 'Avenida Principal', '0223456789');

-- Preenchendo tabela PessoaFisica e PessoaJuridica
insert into PessoaFisica (PessoaFisicaID, Pessoa_PessoaID, CPF)
values ('102', '102', '444777999');

insert into PessoaJuridica (PessoaJuridicaID, Pessoa_PessoaID, CNPJ)
values ('103', '103', '555888111');

-- Atualizando a tabela MovimentoCompra
UPDATE MovimentoCompra
SET Quantidade = Quantidade - 2
WHERE Produtos_ProdutosID = 3;

-- Consultando dados de pessoas físicas
select Pessoa.PessoaID, Pessoa.Nome, Pessoa.Endereco, Pessoa.Telefone,
PessoaFisica.CPF
from Pessoa inner join PessoaFisica on Pessoa.PessoaID =
PessoaFisica.Pessoa_PessoaID;

-- Consultando dados de pessoas jurídicas
select Pessoa.PessoaID, Pessoa.Nome, Pessoa.Endereco, Pessoa.Telefone,
PessoaJuridica.CNPJ
from Pessoa inner join PessoaJuridica on Pessoa.PessoaID =
PessoaJuridica.Pessoa_PessoaID;
```

Análise e Conclusão

Quais as diferenças no uso de sequence e identity?
A diferença principal entre IDENTITY e SEQUENCE é que o IDENTITY está mais associado a uma coluna específica e geralmente é usado para gerar valores exclusivos automaticamente para essa coluna em cada linha de uma tabela, enquanto o SEQUENCE no PostgreSQL oferece uma abordagem mais flexível, permitindo a geração de sequências de valores que não estão vinculadas diretamente a uma coluna específica e podem ser compartilhadas entre várias tabelas.

Qual a importância das chaves estrangerias para a consistência do banco?
As chaves estrangeiras são fundamentais para a consistência do banco de dados, pois garantem integridade referencial. Elas estabelecem relacionamentos entre tabelas, assegurando que os valores em uma tabela que se referem a outra tabela realmente existam. Isso ajuda a evitar inconsistências e garante que não haja referências a dados inexistentes, mantendo a integridade e a consistência dos dados no banco.

Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?
Na álgebra relacional, operadores comuns do SQL incluem:

Projeção (SELECT)
União (UNION)
Interseção (INTERSECT)
Diferença (EXCEPT ou MINUS)
Produto Cartesiano (CROSS JOIN)
Restrição (WHERE)
No cálculo relacional, não há uma correspondência direta com os operadores do SQL, pois o cálculo relacional expressa consultas de forma mais descritiva, não utilizando operadores específicos como na álgebra. Em vez disso, utiliza-se a notação para descrever as condições e restrições das consultas de maneira mais teórica e declarativa.

Como é feito o agrupamento em consultas, e qual requisito é obrigatório?
É feito usando a cláusula GROUP BY, onde as linhas são agrupadas com base nos valores de uma ou mais colunas. Ao usar GROUP BY, é obrigatório listar todas as colunas não agregadas na cláusula GROUP BY ou usar funções de agregação (como SUM, COUNT, AVG, etc.) para essas colunas na seleção. Isso garante que o resultado seja significativo e previsível, evitando ambiguidades nos dados agrupados.