UNIVERSIDADE TUIUTÍ DO PARANÁ

ALEXANDRE JUNIOR GARCIA DOS SANTOS
ÍCARO DE ASSIS HOSTERT
LEONARDO DE PAULA TEIXEIRA
VITOR ROBERTO BATISTA SCHIRMER

PROGRAMAÇÃO EM BANCO DE DADOS - ESTUDO DIRIGIDO

CURITIBA
2024

ALEXANDRE

ICARO

LEONARDO

VITOR ROBERTO BATISTA SCHIRMER

PROGRAMAÇÃO EM BANCO DE DADOS - ESTUDO DIRIGIDO

Trabalho apresentado para o curso de Análise e Desenvolvimento de Sistemas, da Universidade Tuiutí do Paraná como requisito avaliativo do primeiro Bimestre da disciplina de Programação de Banco de Dados

Professor: Msc. Sérgio Luiz Marques Filho

CURITIBA

2023

SUMÁRIO

1. INTRODUÇÃO

Foi desenvolvido scripts usando SQL SERVER 2008 simulando uma *blockchain* de criptomoedas. Para melhor entendimento, o conteúdo do

Estrutura-se um banco de dados simulando uma *blockchain* de criptomoedas, usando SQL SERVER 2008. Os scripts desenvolvidos têm a função de criar o esquema, as tabelas, inserir dados e criar visões. Para um melhor entendimento, três scripts foram criados: *setup.sql*, *views.sql*, *insert.sql*

2. SCRIPTS

2.1 setup.sql

Esse script irá criar o esquema e as tabelas.

```
CREATE DATABASE CRYPTO_DATABASE;
GO

USE CRYPTO_DATABASE;
GO

-- Create tables and indexes
CREATE TABLE Moeda (
    CodigoMoeda VARCHAR(3) PRIMARY KEY,
    Nome VARCHAR(255)
);
GO

CREATE INDEX IDX_Moeda_CodigoMoeda ON Moeda(CodigoMoeda);
GO

CREATE TABLE ParesMoedas (
    CodigoMoedaBase VARCHAR(3),
    CodigoMoedaCotacao VARCHAR(3),
    Valor FLOAT,
    PRIMARY KEY (CodigoMoedaBase, CodigoMoedaCotacao),
    FOREIGN KEY (CodigoMoedaBase) REFERENCES
Moeda(CodigoMoeda),
    FOREIGN KEY (CodigoMoedaCotacao) REFERENCES
Moeda(CodigoMoeda)
);
GO

CREATE INDEX IDX_ParesMoedas_CodigoMoedaBase_CodigoMoedaCotacao
ON ParesMoedas(CodigoMoedaBase, CodigoMoedaCotacao);
GO

CREATE TABLE Corretora (
    CodigoCorretora INT PRIMARY KEY,
    Nome VARCHAR(255)
);
GO

CREATE INDEX IDX_Corretora_CodigoCorretora ON
Corretora(CodigoCorretora);
GO

CREATE TABLE Cliente (
    CodigoCliente INT PRIMARY KEY,
    Nome VARCHAR(255),
    Email VARCHAR(255),
```

```
    Celular VARCHAR(20),
    PassHash VARCHAR(255),
    MoedaPrincipal VARCHAR(3),
    FOREIGN KEY (MoedaPrincipal) REFERENCES Moeda(CodigoMoeda)
);
GO

CREATE INDEX IDX_Cliente_CodigoCliente ON
Cliente(CodigoCliente);
GO

CREATE TABLE Carteira (
    Endereco VARCHAR(255) PRIMARY KEY,
    CodigoCorretora INT,
    CodigoCliente INT,
    FOREIGN KEY (CodigoCorretora) REFERENCES
Corretora(CodigoCorretora),
    FOREIGN KEY (CodigoCliente) REFERENCES
Cliente(CodigoCliente)
);
GO

CREATE INDEX IDX_Carteira_Endereco ON Carteira(Endereco);
GO

CREATE TABLE ItemCarteira (
    CodigoItemCarteira INT PRIMARY KEY,
    Endereco VARCHAR(255),
    CodigoMoeda VARCHAR(3),
    Quantidade FLOAT,
    FOREIGN KEY (Endereco) REFERENCES Carteira(Endereco),
    FOREIGN KEY (CodigoMoeda) REFERENCES Moeda(CodigoMoeda)
);
GO

CREATE INDEX IDX_ItemCarteira_CodigoItemCarteira ON
ItemCarteira(CodigoItemCarteira);
GO
```

2.2. insert.sql

Esse script irá popular as tabelas. Assim como foi pedido, usa-se o comando *begin transaction*, permitindo fazer rollback se necessário, mas para funcionar não se deve executar o *commit*.

```
USE CRYPTO_DATABASE;
GO

-- INSERE DADOS NAS TABELAS
BEGIN TRANSACTION;

INSERT INTO Moeda (CodigoMoeda, Nome) VALUES ('BTC', 'Bitcoin');
INSERT INTO Moeda (CodigoMoeda, Nome) VALUES ('ETH', 'Ethereum');
INSERT INTO Moeda (CodigoMoeda, Nome) VALUES ('BRL', 'Brazilian Real');

INSERT INTO ParesMoedas (CodigoMoedaBase, CodigoMoedaCotacao, Valor) VALUES
('BTC', 'BRL', 322156.60);
INSERT INTO ParesMoedas (CodigoMoedaBase, CodigoMoedaCotacao, Valor) VALUES
('ETH', 'BRL', 13616.22);

INSERT INTO Corretora (CodigoCorretora, Nome) VALUES (1, 'Nuinvest');
INSERT INTO Corretora (CodigoCorretora, Nome) VALUES (2, 'Binance');
INSERT INTO Corretora (CodigoCorretora, Nome) VALUES (3, 'XP Investimentos');

INSERT INTO Cliente (CodigoCliente, Nome, Email, Celular, PassHash, MoedaPrincipal)
VALUES (1, 'Vitor', 'vitor@example.com', '1234567890', HASHBYTES('MD5', 'senha123'),
'BTC');
INSERT INTO Cliente (CodigoCliente, Nome, Email, Celular, PassHash, MoedaPrincipal)
VALUES (2, 'Icaro', 'icaro@example.com', '0987654321',  HASHBYTES('MD5', 'senha456'),
'ETH');
INSERT INTO Cliente (CodigoCliente, Nome, Email, Celular, PassHash, MoedaPrincipal)
VALUES (3, 'Leonador', 'leo@example.com', '5555555555',  HASHBYTES('MD5', 'senha789'),
'BTC');
INSERT INTO Cliente (CodigoCliente, Nome, Email, Celular, PassHash, MoedaPrincipal)
VALUES (4, 'Alexandre', 'alexandre@example.com', '5555555555',  HASHBYTES('MD5',
'senha789'), 'BTC');

INSERT INTO Carteira (Endereco, CodigoCorretora, CodigoCliente)
VALUES ('vitor_carteira', 1, 1);
INSERT INTO Carteira (Endereco, CodigoCorretora, CodigoCliente)
VALUES ('icaro_carteira', 2, 2);
INSERT INTO Carteira (Endereco, CodigoCorretora, CodigoCliente)
VALUES ('leo_carteira', 3, 3);
INSERT INTO Carteira (Endereco, CodigoCorretora, CodigoCliente)
VALUES ('alexandre_carteira', 3, 4);

INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (1, 'vitor_carteira', 'BTC', 1.5);
INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (2, 'vitor_carteira', 'ETH', 0.5);
INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (3, 'vitor_carteira', 'BTC', 2);
INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (4, 'icaro_carteira', 'BTC', 1.0);
```

```
INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (5, 'icaro_carteira', 'ETH', 10.0);
INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (6, 'leo_carteira', 'BTC', 0.5);
INSERT INTO ItemCarteira (CodigoItemCarteira, Endereco, CodigoMoeda, Quantidade)
VALUES (7, 'alexandre_carteira', 'BTC', 2.0);

-- Confirme as alterações usando COMMIT ou as desfaça usando ROLLBACK
COMMIT;
-- ROLLBACK;
GO
```

2.3 views.sql

Esse script irá gerar as visões de três consultas. Para a criação correta, deve-se criá-las individualmente como foi ressaltado no comentário dentro do script.

A gente criou três consultas:

1. Consulta que lista as informações da carteira de cada cliente.
2. Quantidade de criptomoedas que cada cliente possui (saldo atual)
3. Total em reais que cada cliente possui, usando a tabela *ParesMoedas* para fazer o cálculo.

```
USE CRYPTO_DATABASE;
GO

-- Seleciona dados de todas as tabelas
SELECT * FROM Moeda;
SELECT * FROM ParesMoedas;
SELECT * FROM Corretora;
SELECT * FROM Cliente;
SELECT * FROM Carteira;
SELECT * FROM ItemCarteira;
GO

-- Carteira de cada cliente, mostrando o endereço da carteira e sua
corretora
CREATE VIEW CarteirasDosClientes AS
SELECT cli.Nome AS ClienteNome, cli.Email, car.Endereco, cor.Nome
AS CorretoraNome
FROM Cliente AS cli
LEFT JOIN Carteira AS car
    ON car.CodigoCliente = cli.CodigoCliente
LEFT JOIN Corretora AS cor
```

```
    ON cor.CodigoCorretora = car.CodigoCorretora;
GO

-- Quantidade de crypto que cada cliente possui (saldo atual)
CREATE VIEW QuantidadeDeMoedasDosClientes AS
SELECT cli.Nome, icar.CodigoMoeda, SUM(icar.Quantidade) AS
qtd_cyptomoedas, SUM(icar.Quantidade * pr.Valor) AS valor_em_reais
FROM Cliente AS cli
LEFT JOIN Carteira AS car
    ON car.CodigoCliente = cli.CodigoCliente
LEFT JOIN ItemCarteira icar
    ON icar.Endereco = car.Endereco
LEFT JOIN ParesMoedas AS pr
    ON pr.CodigoMoedaBase = icar.CodigoMoeda
GROUP BY cli.Nome, icar.CodigoMoeda;
GO

-- Total em reais que cada cliente possui
CREATE VIEW ValorTotalNaCarteira AS
SELECT cli.Nome, SUM(icar.Quantidade * pr.Valor) AS total
FROM Cliente AS cli
LEFT JOIN Carteira AS car
    ON car.CodigoCliente = cli.CodigoCliente
LEFT JOIN ItemCarteira icar
    ON icar.Endereco = car.Endereco
LEFT JOIN ParesMoedas AS pr
    ON pr.CodigoMoedaBase = icar.CodigoMoeda
GROUP BY cli.Nome;
GO

-- Seleciona dados das views criadas
SELECT * FROM CarteirasDosClientes;
GO

SELECT * FROM QuantidadeDeMoedasDosClientes;
GO

SELECT * FROM ValorTotalNaCarteira;
GO

-- Para deletar as views, use o seguinte comando:
-- DROP VIEW CarteirasDosClientes, QuantidadeDeMoedasDosClientes,
ValorTotalNaCarteira;
```