POLO OLARIA - RIO DE JANEIRO - RJ

DESENVOLVIMENTO FULL STACK

Nivel 2: Vamos Manter as Informações? | Turma 9001

Erik Bastos de Moraes

Missão Prática | Nível 2 | Mundo 3

2º Procedimento | Alimentando a Base

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).

5- No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

SQL Server Management Studio

```
CREATE SEQUENCE seq_pessoa
START WITH 1
INCREMENT BY 1;
GO

CREATE TABLE Usuario (
    id_usuario INT IDENTITY(1,1) PRIMARY KEY,
    nome VARCHAR(100),
    login VARCHAR(50) UNIQUE,
    senha VARCHAR(50)
);

CREATE TABLE Pessoa (
    id_pessoa INT PRIMARY KEY DEFAULT NEXT VALUE FOR seq_pessoa,
    nome VARCHAR(100),
    endereco VARCHAR(200),
    telefone VARCHAR(20),
    email VARCHAR(100),
    cpf VARCHAR(14),
    cnpj VARCHAR(18)
);

CREATE TABLE Produto (
    id_produto INT IDENTITY(1,1) PRIMARY KEY,
    nome VARCHAR(100),
    quantidade INT,
    preco_venda DECIMAL(10,2)
```

```
);

CREATE TABLE Compra (
    id_compra INT IDENTITY(1,1) PRIMARY KEY,
    id_usuario INT,
    id_produto INT,
    id_fornecedor INT,
    quantidade INT,
    preco_unitario DECIMAL(10,2),
    data DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario),
    FOREIGN KEY (id_produto) REFERENCES Produto(id_produto),
    FOREIGN KEY (id_fornecedor) REFERENCES Pessoa(id_pessoa)
);

CREATE TABLE Venda (
    id_venda INT IDENTITY(1,1) PRIMARY KEY,
    id_usuario INT,
    id_produto INT,
    id_cliente INT,
    quantidade INT,
    data DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario),
    FOREIGN KEY (id_produto) REFERENCES Produto(id_produto),
    FOREIGN KEY (id_cliente) REFERENCES Pessoa(id_pessoa)
);

INSERT INTO Usuario (nome, login, senha) VALUES
('1', 'op1', 'op1'),
('2', 'op2', 'op2');
```

```
INSERT INTO Produto (nome, quantidade, preco_venda) VALUES
('Mouse', 100, 50.00),
('Teclado', 200, 80.00),
('Monitor', 50, 600.00);

DECLARE @idPF1 INT = NEXT VALUE FOR seq_pessoa;
INSERT INTO Pessoa (id_pessoa, nome, endereco, telefone, email, cpf) VALUES
(@idPF1, 'Roland', 'Distrito 7, 100', '99999999999', 'roland@gmail.com', '123.456.789-00');

DECLARE @idPJ1 INT = NEXT VALUE FOR seq_pessoa;
INSERT INTO Pessoa (id_pessoa, nome, endereco, telefone, email, cnpj) VALUES
(@idPJ1, 'TimeTrack Corporation', 'Distrito 20', '1133334444', 'tcorp@wing.com', '12.345.678/0001-99');

INSERT INTO Compra (id_usuario, id_produto, id_fornecedor, quantidade, preco_unitario)
VALUES (1, 1, @idPJ1, 10, 40.00);

INSERT INTO Venda (id_usuario, id_produto, id_cliente, quantidade)
VALUES (2, 1, @idPF1, 5);

SELECT * FROM Pessoa WHERE cpf IS NOT NULL;

SELECT * FROM Pessoa WHERE cnpj IS NOT NULL;

SELECT c.id_compra, p.nome AS produto, f.nome AS fornecedor, c.quantidade,
    c.preco_unitario, (c.quantidade * c.preco_unitario) AS valor_total
FROM Compra c
JOIN Produto p ON c.id_produto = p.id_produto
```

```
JOIN Pessoa f ON c.id_fornecedor = f.id_pessoa;

SELECT v.id_venda, p.nome AS produto, c.nome AS comprador, v.quantidade,
       p.preco_venda AS preco_unitario, (v.quantidade * p.preco_venda) AS valor_total
FROM Venda v
JOIN Produto p ON v.id_produto = p.id_produto
JOIN Pessoa c ON v.id_cliente = c.id_pessoa;

SELECT p.nome, SUM(c.quantidade * c.preco_unitario) AS total_entrada
FROM Compra c
JOIN Produto p ON c.id_produto = p.id_produto
GROUP BY p.nome;

SELECT p.nome, SUM(v.quantidade * p.preco_venda) AS total_saida
FROM Venda v
JOIN Produto p ON v.id_produto = p.id_produto
GROUP BY p.nome;

SELECT u.id_usuario, u.nome
FROM Usuario u
WHERE u.id_usuario NOT IN (SELECT DISTINCT id_usuario FROM Compra);

SELECT u.nome, SUM(c.quantidade * c.preco_unitario) AS total_comprado
FROM Compra c
JOIN Usuario u ON c.id_usuario = u.id_usuario
GROUP BY u.nome;

SELECT u.nome, SUM(v.quantidade * p.preco_venda) AS total_vendido
FROM Venda v
JOIN Usuario u ON v.id_usuario = u.id_usuario
```

```
JOIN Produto p ON v.id_produto = p.id_produto

GROUP BY u.nome;


SELECT p.nome,

      CAST(SUM(v.quantidade * p.preco_venda) AS FLOAT) / SUM(v.quantidade) AS
media_ponderada

FROM Venda v

JOIN Produto p ON v.id_produto = p.id_produto

GROUP BY p.nome;
```

Análise e Conclusão

A. Quais as diferenças no uso de sequence e identity?

R: O identity é mais direto e simples, porém com menos flexibilidade, já o sequence é mais reutilizavel, e possui maior controle e flexibilidade.

B. Qual a importância das chaves estrangerias para a consistência do banco?

R: Elas garantem que as relações entre tabelas sejam mantidas de forma coerente e sem erros.

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

R: Os que pertencem à algebra relacional são alguns como: SELECT, JOIN, UNION, INTERSECT, EXCEPT e PROJECT, já no cálculo relacional, seriam: AND, Or, NOT, =, !=, <, >, <=, >=, entre outros.

D. Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

R: O agrupamento em consultas SQL é feito utilizando o operador GROUP BY, que permite agrupar os resultados de uma consulta com base em uma ou mais colunas, após agrupar, é possível realizar agregações (como COUNT, SUM, AVG, MIN, MAX) sobre cada grupo.

---

https://github.com/ErikBM2661/Loja.git