

**Disciplina: nível 2 - Vamos manter as informações**

**Turma: 9001      Semestre letivo: terceiro**

**Nome: David Lins do Amaral**

## **Missão prática nível 2**

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

### **Objetivos da prática**

1. Identificar os requisitos de um sistema e
2. transformá-los no modelo adequado.
3. Utilizar ferramentas de modelagem para bases
4. de dados relacionais.
5. Explorar a sintaxe SQL na criação das estruturas
6. do banco (DDL).
7. Explorar a sintaxe SQL na consulta e
8. manipulação de dados (DML)
9. No final do exercício, o aluno terá vivenciado a
10. experiência de modelar a base de dados para um
11. sistema simples, além de implementá-la, através
12. da sintaxe SQL, na plataforma do SQL Server.

# CRIAÇÃO DAS TABELAS SOLICITADAS E INCLUSÃO DE REFERENCIAS:

SQLQuery19.sql - R...DEX1000\linsd (55)\* X SQLQuery18.sql - R...DEX1000\linsd (72)\*

```
CREATE SEQUENCE seq_pessoa_id
START WITH 1
INCREMENT BY 1;

CREATE TABLE pessoa (
    id INT PRIMARY KEY DEFAULT NEXT VALUE FOR seq_pessoa_id,
    nome VARCHAR(255) NOT NULL
);

CREATE TABLE pessoa_fisica (
    id_pessoa INT PRIMARY KEY,
    cpf VARCHAR(11) UNIQUE NOT NULL,
    data_nascimento DATE,
    FOREIGN KEY (id_pessoa) REFERENCES pessoa(id)
);

CREATE TABLE pessoa_juridica (
    id_pessoa INT PRIMARY KEY,
    cnpj VARCHAR(14) UNIQUE NOT NULL,
    razao_social VARCHAR(255) NOT NULL,
    FOREIGN KEY (id_pessoa) REFERENCES pessoa(id)
);

CREATE TABLE produtos (
    id INT IDENTITY(1,1) PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    quantidade INT NOT NULL DEFAULT 0,
    preco_venda DECIMAL(10, 2) NOT NULL
);

CREATE TABLE usuarios (
    id INT IDENTITY(1,1) PRIMARY KEY,
    nome NVARCHAR(100) NOT NULL,
    email NVARCHAR(100) NULL,
    senha NVARCHAR(255) NOT NULL
);

CREATE TABLE movimento (
    idMovimento INT IDENTITY(1,1) PRIMARY KEY,
    idUsuario INT NOT NULL,
    idPessoa INT NOT NULL,
    idProduto INT NOT NULL,
    quantidade INT NOT NULL,
    tipo CHAR(1) CHECK (tipo IN ('E', 'S')) NOT NULL,
    valorUnitario DECIMAL(10, 2) NOT NULL,
    dataMovimento DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (idUsuario) REFERENCES usuarios(id),
    FOREIGN KEY (idPessoa) REFERENCES pessoa(id),
    FOREIGN KEY (idProduto) REFERENCES produtos(id),
    CHECK (quantidade > 0 AND valorUnitario > 0)
);
```

83 %

Mensagens

Comandos concluídos com êxito.

Horário de conclusão: 2025-04-25T05:42:12.4346029-03:00

RYDEX1000\SQLEXPRESS (SQL Server 16.0.1000 - RYDEX1000\linsd)

Bancos de Dados

Bancos de Dados do Sistema

Instantâneos do Banco de Dados

loja

Diagramas de Banco de Dados

Tabelas

Tabelas do Sistema

FileTables

Tabelas Externas

Tabelas de Grafo

dbo.movimento

dbo.pessoa

dbo.pessoa\_fisica

dbo.pessoa\_juridica

dbo.produtos

dbo.usuarios

Exibições

Recursos Externos

Sinônimos

Programação

Repositório de Consultas

Service Broker

Armazenamento

Segurança

Segurança

Logons

##MS\_PolicyEventProcessingLogin##

##MS\_PolicyTsqlExecutionLogin##

AUTORIDADE NT\SISTEMA

BUILTIN\Usuários

loja

NT Service\MSSQL\$SQLEXPRESS

NT SERVICE\SQLTELEMETRY\$SQLEXPRESS

NT SERVICE\SQLWriter

NT SERVICE\Winmgmt

RYDEX1000\linsd

sa

Funções de Servidor

Credenciais

Auditorias

Especificações de Auditoria de Servidor

Objetos de Servidor

Replicação

Gerenciamento

XEvent Profiler

# Inserindo dados:

## *Tabela usuário dados:*

```
SQLQuery20.sql - R...DEX1000\linsd (81))* X SQLQuery19.sql - R...DEX1000\linsd (81))* X
INSERT INTO usuarios (nome, senha)
VALUES ('op1', 'op1');

INSERT INTO usuarios (nome, senha)
VALUES ('op2', 'op2');
```

83 %

Mensagens

(1 linha afetada)

(1 linha afetada)

Horário de conclusão: 2025-04-25T05:44:17.7261582-03:00

## *Tabela produto dados:*

```
SQLQuery21.sql - R...DEX1000\linsd (54))* X SQLQuery20.sql - R...DEX1000\linsd (54))* X
INSERT INTO produtos (nome, quantidade, preco_venda)
VALUES ('Banana', 100, 5.00);

INSERT INTO produtos (nome, quantidade, preco_venda)
VALUES ('Laranja', 500, 2.00);

INSERT INTO produtos (nome, quantidade, preco_venda)
VALUES ('Manga', 800, 4.00);
```

83 %

Mensagens

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

Horário de conclusão: 2025-04-25T05:45:45.2087217-03:00

## *Tabelas pessoa, pessoa\_fisica, pessoa\_juridica (dados):*

```
SQLQuery22.sql - R...DEX1000\linsd (51))* X SQLQuery21.sql - R...DEX1000\linsd (54))* X SQLQuery20.sql - R...DEX1000\linsd (54))* X
INSERT INTO pessoa (nome) VALUES ('Joao');
DECLARE @joao_id INT;
SELECT @joao_id = id FROM pessoa WHERE nome = 'Joao';
INSERT INTO pessoa_fisica (id_pessoa, cpf) VALUES (@joao_id, '11111111111');

INSERT INTO pessoa (nome) VALUES ('JJC');
DECLARE @jjc_id INT;
SELECT @jjc_id = id FROM pessoa WHERE nome = 'JJC';
INSERT INTO pessoa_juridica (id_pessoa, cnpj, razao_social) VALUES (@jjc_id, '222222222222', 'JJC');
```

83 %

Mensagens

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

Horário de conclusão: 2025-04-25T05:55:41.4108490-03:00

## Tabela movimento dados:

SQLQuery23.sql - R...DEX1000\linsd (74))\* SQLQuery22.sql - R...DEX1000\linsd (51))\* SQLQuery21.sql - R...DEX1000\linsd (54))\* SQLQuery20.sql - R...DEX1000\linsd (81))\*

```

INSERT INTO movimento (idUserio, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES (2, (SELECT id FROM pessoa WHERE nome = 'Joao'), 1, 5, 'S', 6.50);
INSERT INTO movimento (idUserio, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES (2, (SELECT id FROM pessoa WHERE nome = 'JJC'), 2, 10, 'E', 1.80);
INSERT INTO movimento (idUserio, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES (1, (SELECT id FROM pessoa WHERE nome = 'Joao'), 3, 2, 'S', 4.20);
INSERT INTO movimento (idUserio, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES (1, (SELECT id FROM pessoa WHERE nome = 'JJC'), 1, 7, 'E', 5.90);
INSERT INTO movimento (idUserio, idPessoa, idProduto, quantidade, tipo, valorUnitario) VALUES (2, (SELECT id FROM pessoa WHERE nome = 'Joao'), 4, 3, 'S', 3.10);

```

83 %

Mensagens

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

(1 linha afetada)

Horário de conclusão: 2025-04-25T06:06:58.4275010-02:00

# CONSULTAS:

Query24.sql - R...DEX1000\linsd (69))\* SQLQuery23.sql - R...DEX1000\linsd (74))\* SQLQuery22.sql - R...DEX1000\linsd (51))\* SQLQuery21.sql - R...DEX1000\linsd (54))\* SQLQuery20.sql - R...DEX1000\linsd (81))\* SQLQuery19.sql - R...

```

SELECT pf.*, p.nome FROM pessoa_fisica pf JOIN pessoa p ON pf.id_pessoa = p.id;
SELECT pj.*, p.nome FROM pessoa_juridica pj JOIN pessoa p ON pj.id_pessoa = p.id;
SELECT m.quantidade, m.valorUnitario, m.quantidade * m.valorUnitario AS valorTotal, pr.nome AS produto, pe.nome AS fornecedor FROM movimento m JOIN produtos pr ON m.idProduto = pr.id JOIN pessoa pe ON m.idPessoa = pe.id WHERE m.tipo = 'E';
SELECT m.quantidade, m.valorUnitario, m.quantidade * m.valorUnitario AS valorTotal, pr.nome AS produto, pe.nome AS comprador FROM movimento m JOIN produtos pr ON m.idProduto = pr.id JOIN pessoa pe ON m.idPessoa = pe.id WHERE m.tipo = 'S';
SELECT pr.nome AS produto, SUM(m.quantidade * m.valorUnitario) AS valorTotalEntrada FROM movimento m JOIN produtos pr ON m.idProduto = pr.id WHERE m.tipo = 'E' GROUP BY pr.nome;
SELECT pr.nome AS produto, SUM(m.quantidade * m.valorUnitario) AS valorTotalSaida FROM movimento m JOIN produtos pr ON m.idProduto = pr.id WHERE m.tipo = 'S' GROUP BY pr.nome;
SELECT u.nome FROM usuarios u LEFT JOIN movimento m ON u.id = m.idUsuario AND m.tipo = 'E' WHERE m.idMovimento IS NULL;
SELECT u.nome AS operador, SUM(m.quantidade * m.valorUnitario) AS valorTotalEntrada FROM movimento m JOIN usuarios u ON m.idUsuario = u.id WHERE m.tipo = 'E' GROUP BY u.nome;
SELECT u.nome AS operador, SUM(m.quantidade * m.valorUnitario) AS valorTotalSaida FROM movimento m JOIN usuarios u ON m.idUsuario = u.id WHERE m.tipo = 'S' GROUP BY u.nome;
SELECT pr.nome AS produto, SUM(m.quantidade * m.valorUnitario) / SUM(m.quantidade) AS valorMedioVenda FROM movimento m JOIN produtos pr ON m.idProduto = pr.id WHERE m.tipo = 'S' GROUP BY pr.nome;

```

Resultados Mensagens

	id_pessoa	cpf	data_nascimento	nome	
1	4	11111111111	NULL	Joao	

	id_pessoa	cnpj	razao_social	nome	
1	5	222222222222	JJC	JJC	

	quantidade	valorUnitario	valorTotal	produto	fornecedor	
1	10	1.80	18.00	Laranja	JJC	
2	7	5.90	41.30	Banana	JJC	

	quantidade	valorUnitario	valorTotal	produto	comprador	
1	5	6.50	32.50	Banana	Joao	
2	2	4.20	8.40	Manga	Joao	
3	3	3.10	9.30	Produto 4	Joao	

	produto	valorTotalEntrada	
1	Banana	41.30	
2	Laranja	18.00	

	produto	valorTotalSaida	
1	Banana	32.50	
2	Manga	8.40	
3	Produ...	9.30	

nome

	operador	valorTotalEntrada	
1	op1	41.30	
2	op2	18.00	

	operador	valorTotalSaida	
1	op1	40.90	
2	op2	9.30	

	produto	valorMedioVenda	
1	Banana	6.500000	
2	Manga	4.200000	
3	Produ...	3.100000	

Consulta executada com êxito.

Consulta executada com êxito.

## ANÁLISE DE CONCLUSÃO:

### Quais as diferenças no uso de sequence e identity?

R: Percebi que IDENTIFY é como ter um contador automático embutido diretamente na coluna da tabela, ele parece funcionar bem para gerar ids únicos quando adiciono novas linhas. Já o SEQUENCE é como ter um contador separado que se pode usar em várias tabelas ou até fora delas.

### Qual a importância das chaves estrangeiras para a consistência do banco?

R: As chaves estrangeiras são importantes porque garantem a integridade dos dados no banco, evitando que sejam inseridas ou mantidas informações que não fazem sentido, como por exemplo registrar uma venda para um cliente que não existe. Elas também impedem que registros importantes sejam apagados sem antes tratar os dados relacionados, ajudando a manter o banco organizado e confiável.

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

R: Os operadores da álgebra relacional são usados para manipular os dados nas tabelas e incluem: seleção, projeção, união, diferença, produto cartesiano, junção e renomeação. Já o cálculo relacional não usa operadores, mas sim expressões lógicas para descrever o que se quer buscar no banco. Ele pode ser feito por cálculo de tuplas ou de domínios. Enquanto a álgebra relacional mostra como os dados serão obtidos, o cálculo relacional foca em dizer apenas quais dados queremos.

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

R: O agrupamento em consultas é feito com o GROUP BY, que serve pra juntar os dados que têm valores iguais em uma ou mais colunas. A gente usa isso quando quer fazer contas, tipo somar ou contar registros por grupo. O que é obrigatório é que todas as colunas que aparecem no SELECT e não estão dentro de funções como SUM ou COUNT têm que estar no GROUP BY, senão o banco dá erro.