Campus Polo Setor Central - Araguaína(TO)

Desenvolvimento Full-Stack

Nível 1: Iniciando o Caminho pelo Java

Turma 2023.1

Semestre Letivo 2024.1

Ana Clara Quidute Marinho Silva

Missão Prática | Nível 2 | Mundo 3

Objetivo 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.

1º Procedimento | Criando o Banco de Dados

Foi usado o DB Designer para a criação do modelo de dados.
Link do site: https://www.dbdesigner.net/

Seguindo todas as instruções do 1º procedimento, o modelo ficou assim:

Após a criação do usuário **loja**, foi feita a criação das estruturas do modelo e da `sequence`.

Script e resultados da criação das tabelas, das chaves estrangeiras e da `sequence`:

```
CREATE TABLE Usuarios (
	id_usuario int PRIMARY KEY NOT NULL,
	login_usuario varchar (15) NOT NULL,
	senha varchar (15) NOT NULL
);
```

Nome da Coluna	Tipo de Dados	Permitir Nul...
id_usuario	int	☐
login_usuario	varchar(15)	☐
senha	varchar(15)	☐
		☐

```
CREATE TABLE Pessoas (
	id_pessoa int default next value for seq_pessoa PRIMARY
KEY NOT NULL,
	nome varchar (200) NOT NULL,
	logradouro varchar (200) NOT NULL,
	cidade varchar (200) NOT NULL,
	estado char (2) NOT NULL,
	telefone varchar (11) NOT NULL,
	email varchar (200) NOT NULL
);
```

Nome da Coluna	Tipo de Dados	Permitir Nul...
id_pessoa	int	☐
nome	varchar(200)	☐
logradouro	varchar(200)	☐
cidade	varchar(200)	☐
estado	char(2)	☐
telefone	varchar(11)	☐
email	varchar(200)	☐
		☐

```
CREATE TABLE Pessoas_Fisicas (
	id_pFisica int PRIMARY KEY NOT NULL,
	id_pessoa int NOT NULL,
	cpf varchar(11) NOT NULL
);
```

Nome da Coluna	Tipo de Dados	Permitir Nul...
id_pFisica	int	☐
id_pessoa	int	☐
cpf	varchar(11)	☐
		☐

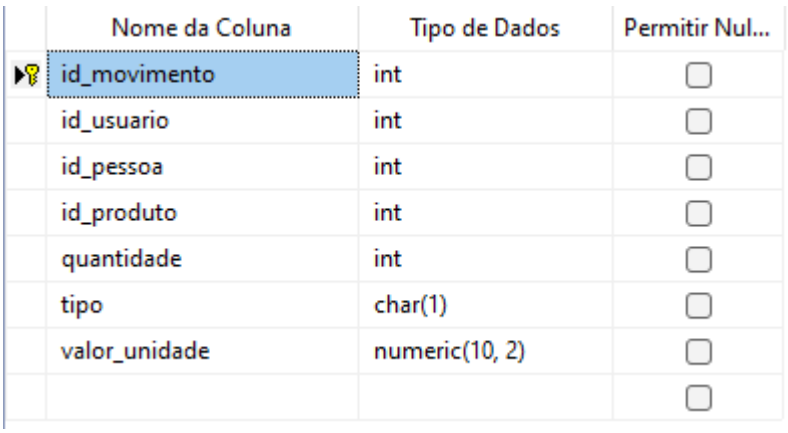
```
CREATE TABLE Pessoas_Juridicas (
	id_pJuridica int PRIMARY KEY NOT NULL,
	id_pessoa int NOT NULL,
	cnpj varchar(14) NOT NULL
);
```

Nome da Coluna	Tipo de Dados	Permitir Nul...
id_pJuridica	int	☐
id_pessoa	int	☐
cnpj	varchar(14)	☐
		☐

```
CREATE TABLE Produtos (
	id_produto int PRIMARY KEY NOT NULL,
	nome_produto varchar (200) NOT NULL,
	quantidade_produto int NOT NULL,
	valor_produto numeric(10,2) NOT NULL
);
```

Nome da Coluna	Tipo de Dados	Permitir Nul...
id_produto	int	☐
nome_produto	varchar(200)	☐
quantidade_produto	int	☐
valor_produto	numeric(10, 2)	☐
		☐

```
CREATE TABLE Movimentos (
	id_movimento int PRIMARY KEY NOT NULL,
	id_usuario int NOT NULL,
	id_pessoa int NOT NULL,
	id_produto int NOT NULL,
	quantidade int NOT NULL,
	tipo char (1) NOT NULL,
	valor_unidade numeric(10,2) NOT NULL
);
```



Nome da Coluna	Tipo de Dados	Permitir Nul...
id_movimento	int	☐
id_usuario	int	☐
id_pessoa	int	☐
id_produto	int	☐
quantidade	int	☐
tipo	char(1)	☐
valor_unidade	numeric(10, 2)	☐
		☐

```
-- Criação da SEQUENCE
CREATE SEQUENCE seq_pessoa AS INT
     START WITH 1
     INCREMENT BY 1;

-- Criação das FKs
-- (FK) P. Físicas x Pessoas
ALTER TABLE Pessoas_Fisicas
     ADD CONSTRAINT fk_PPFisica FOREIGN KEY (id_pessoa)
REFERENCES Pessoas (id_pessoa);

-- (FK) P. Jurídica x Pessoas
ALTER TABLE Pessoas_Juridicas
     ADD CONSTRAINT fk_PPJuridica FOREIGN KEY (id_pessoa)
REFERENCES Pessoas (id_pessoa);

-- (FK) Movimentos x Usuarios
ALTER TABLE Movimentos
     ADD CONSTRAINT fk_MovUsuario FOREIGN KEY (id_usuario)
REFERENCES Usuarios (id_usuario);

-- (FK) Movimentos x Pessoas
ALTER TABLE Movimentos
     ADD CONSTRAINT fk_MovPessoa FOREIGN KEY (id_pessoa)
REFERENCES Pessoas (id_pessoa);

-- (FK) Movimentos x Produto
ALTER TABLE Movimentos
     ADD CONSTRAINT fk_MovProduto FOREIGN KEY (id_produto)
REFERENCES Produtos (id_produto);
```

Análises e Conclusões

a. No banco de dados relacional, as cardinalidades vão definir a relação entre entidades, com quantas instâncias de uma entidade podem se relacionar com quantas instâncias da outra.
   As três cardinalidades que temos são as: **1:1**, onde uma instância de **uma** entidade só pode estar relacionada a no máximo **uma** instância na outra entidade, **1:N**, onde **uma** entidade pode se relacionar com **N** instâncias da outra entidade, e **N:N**, onde **N** entidades podem se relacionar com **N** instâncias da outra entidade.
b. Geralmente é utilizado o relacionamento Herança/Subtipo, onde uma tabela principal contém os atributos comuns a todos os subtipos e as tabela subtipos contém os atributos específicos de cada subtipo. Podemos ver essa relação da tabela Pessoas com as as tabelas `Pessoas_Fisicas` e `Pessoas_Juridicas`.
c. O SQL Server Management Studio é uma ferramenta de gerenciamento e pesquisa que facilita a manipulação do banco de dados. Com várias funções e opção de gerenciamento, uma das principais funções que permite melhorar a produtividade de quem o está usando é as opções que ele dá de auto complemento. E tem várias outras funções boas de serem usadas para melhorar a produtividade nas tarefas relacionadas ao gerenciamento do banco de dados.

2º Procedimento | Alimentando a Base

Logado como usuário Loja, foi inserido as informações no banco de dados.

Script e resultados da inserção de dados:

```
-- ALIMENTANDO A BASE

-- Tb. Usuarios
INSERT INTO Usuarios (id_usuario, login_usuario, senha)
     VALUES (1, 'op1', 'op1');
INSERT INTO Usuarios (id_usuario, login_usuario, senha)
     VALUES (2, 'op2', 'op2');
INSERT INTO Usuarios (id_usuario, login_usuario, senha)
     VALUES (3, 'op3', 'op3');
```

	id_usuario	login_usuario	senha
1	1	op1	op1
2	2	op2	op2
3	3	op3	op3

```
-- Tb. Produtos
INSERT INTO Produtos (id_produto, nome_produto,
quantidade_produto, valor_produto)
     VALUES (1, 'Banana', 100, 5)
INSERT INTO Produtos(id_produto, nome_produto,
quantidade_produto, valor_produto)
     VALUES (3, 'Laranja', 500, 2)
INSERT INTO Produtos(id_produto, nome_produto,
quantidade_produto, valor_produto)
     VALUES (4, 'Manga', 800, 4)
```

	id_produto	nome_produto	quantidade_produto	valor_produto
1	1	Banana	100	5.00
2	3	Laranja	500	2.00
3	4	Manga	800	4.00

```
-- Tb. Pessoas
INSERT INTO Pessoas (nome, logradouro, cidade, estado,
telefone, email)
     VALUES ('Joao', 'Rua 12, casa 3, Quitanda', 'Riacho do
Sul', 'PA', '1111-1111', 'joao@riacho.com')
INSERT INTO Pessoas (nome, logradouro, cidade, estado,
telefone, email)
     VALUES ('JCC', 'Rua 11, Centro', 'Riacho do Norte', 'PA',
'1212-1212', 'jcc@riacho.com')
```

	id_pessoa	nome	logradouro	cidade	estado	telefone	email
1	1	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com
2	2	JCC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jcc@riacho.com

```
-- Tb. Pessoas_Fisicas
INSERT INTO Pessoas_Fisicas (id_pFisica, id_pessoa, cpf)
     VALUES ('1','1','11111111111')
```

	id_pFisica	id_pessoa	cpf
1	1	1	11111111111

```
-- Tb. Pessoas_Juridicas
INSERT INTO Pessoas_Juridicas(id_pJuridica, id_pessoa, cnpj)
     VALUES ('1','2','22222222222222')
```

	id_pJuridica	id_pessoa	cnpj
1	1	2	22222222222222

```
-- Tb. Movimentos
INSERT INTO Movimentos (id_movimento, id_usuario, id_pessoa,
id_produto, quantidade, tipo, valor_unidade)
     VALUES (1,1,1,1,20,'S',4)
INSERT INTO Movimentos (id_movimento, id_usuario, id_pessoa,
id_produto, quantidade, tipo, valor_unidade)
     VALUES (4,1,1,3,15,'S',2)
INSERT INTO Movimentos (id_movimento, id_usuario, id_pessoa,
id_produto, quantidade, tipo, valor_unidade)
     VALUES (5,2,1,3,10,'S',3)
INSERT INTO Movimentos (id_movimento, id_usuario, id_pessoa,
id_produto, quantidade, tipo, valor_unidade)
     VALUES (7,1,2,3,15,'E',5)
INSERT INTO Movimentos (id_movimento, id_usuario, id_pessoa,
id_produto, quantidade, tipo, valor_unidade)
     VALUES (8,1,2,4,20,'E',4)
```

	id_movimento	id_usuario	id_pessoa	id_produto	quantidade	tipo	valor_unidade
1	1	1	1	1	20	S	4.00
2	4	1	1	3	15	S	2.00
3	5	2	1	3	10	S	3.00
4	7	1	2	3	15	E	5.00
5	8	1	2	4	20	E	4.00

Consultas sobre os dados inseridos e seus respectivos resultados:

```
-- A. Dados completos de pessoas físicas
SELECT Pessoas.*, Pessoas_Fisicas.cpf
FROM Pessoas
INNER JOIN Pessoas_Fisicas ON Pessoas.id_pessoa =
Pessoas_Fisicas.id_pessoa
```

	id_pessoa	nome	logradouro	cidade	estado	telefone	email	cpf
1	1	Joao	Rua 12, casa 3, Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	11111111111

```
-- B. Dados completos de pesssoas jurídicas
SELECT Pessoas.*, Pessoas_Juridicas.cnpj
FROM Pessoas
INNER JOIN Pessoas_Juridicas ON Pessoas.id_pessoa =
Pessoas_Juridicas.id_pessoa
```

	id_pessoa	nome	logradouro	cidade	estado	telefone	email	cnpj
1	2	JCC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jcc@riacho.com	22222222222222

```
-- C. Movimentações de entrada
SELECT Movimentos.*, Pessoas.nome, Produtos.nome_produto,
Movimentos.quantidade, Movimentos.valor_unidade,
(Movimentos.valor_unidade * Movimentos.quantidade) AS 'Valor
Total'
FROM Movimentos
INNER JOIN Pessoas ON Movimentos.id_pessoa = Pessoas.id_pessoa
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
WHERE Movimentos.tipo = 'E'
```

	id_movimento	id_usuario	id_pessoa	id_produto	quantidade	tipo	valor_unidade	nome	nome_produto	quantidade	valor_unidade	Valor Total
1	7	1	2	3	15	E	5.00	JCC	Laranja	15	5.00	75.00
2	8	1	2	4	20	E	4.00	JCC	Manga	20	4.00	80.00

```
-- D. Movimentações de saída
SELECT Movimentos.*, Pessoas.nome, Produtos.nome_produto,
Movimentos.quantidade, Movimentos.valor_unidade,
(Movimentos.valor_unidade * Movimentos.quantidade) AS 'Valor
Total'
FROM Movimentos
INNER JOIN Pessoas ON Movimentos.id_pessoa = Pessoas.id_pessoa
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
WHERE Movimentos.tipo = 'S'
```

	id_movimento	id_usuario	id_pessoa	id_produto	quantidade	tipo	valor_unidade	nome	nome_produto	quantidade	valor_unidade	Valor Total
1	1	1	1	1	20	S	4.00	Joao	Banana	20	4.00	80.00
2	4	1	1	3	15	S	2.00	Joao	Laranja	15	2.00	30.00
3	5	2	1	3	10	S	3.00	Joao	Laranja	10	3.00	30.00

```
-- E. Valor total das entradas agrupadas por produto
SELECT Produtos.nome_produto, SUM(Movimentos.valor_unidade *
Movimentos.quantidade) AS valor_total
FROM Movimentos
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
WHERE Movimentos.tipo = 'E'
GROUP BY Produtos.nome_produto
```

	nome_produto	valor_total
1	Laranja	75.00
2	Manga	80.00

```
-- F. Valor total das saídas agrupadas por produto
SELECT Produtos.nome_produto, SUM(Movimentos.valor_unidade *
Movimentos.quantidade) AS valor_total
FROM Movimentos
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
WHERE Movimentos.tipo = 'S'
GROUP BY Produtos.nome_produto
```

	nome_produto	valor_total
1	Banana	80.00
2	Laranja	60.00

```
-- G. Operadores (usuários) que não efetuaram compra
SELECT Usuarios.login_usuario
FROM Usuarios
LEFT JOIN Movimentos ON Usuarios.id_usuario =
Movimentos.id_usuario
WHERE Movimentos.id_movimento IS NULL
```

	login_usuario
1	op3

```
-- H. Valor total de entrada, agrupado por operadores
SELECT Movimentos.id_usuario, SUM(Movimentos.valor_unidade *
Movimentos.quantidade) AS valor_total
FROM Movimentos
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
WHERE Movimentos.tipo = 'E'
GROUP BY Movimentos.id_usuario
```

	id_usuario	valor_total
1	1	155.00

```
-- I. Valor total de saída, agrupado por operadores
SELECT Movimentos.id_usuario, SUM(Movimentos.valor_unidade *
Movimentos.quantidade) AS valor_total
FROM Movimentos
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
```

```
WHERE Movimentos.tipo = 'S'
GROUP BY Movimentos.id_usuario
```

	id_usuario	valor_total
1	1	110.00
2	2	30.00

```
-- J. Valor médio de venda por produto
SELECT Produtos.nome_produto AS 'Nome do Produto',
SUM(Movimentos.valor_unidade *
Movimentos.quantidade)/SUM(Movimentos.quantidade) AS 'Média
Poderada'
FROM Movimentos
INNER JOIN Produtos ON Movimentos.id_produto =
Produtos.id_produto
GROUP BY Produtos.nome_produto
```

	Nome do Produto	Média Poderada
1	Banana	4.000000
2	Laranja	3.375000
3	Manga	4.000000

Análise e conclusões:

a. A diferença entre usar `sequence` e `identity` é que o `sequence` pode ser colocado em mais de uma tabela, já o `identity` é exclusivo de uma tabela.
b. A chave estrangeira estabelece uma integridade entre as tabelas, garantindo que cada valor em uma coluna de referência tenha uma correspondente válida na tabela referenciada. Por exemplo, o `id_pessoa` na tabela `Pessoas_Fisicas` tem que referenciar um `id_pessoa` na tabela Pessoas pelo fato do `id_pessoa` ser uma chave estrangeira na tabela `Pessoas_Fisicas`.
c. Operadores de álgebra relacional: `SELECT`, `SELECT DISTINCT`, `UNION`, `INTERSECT`, `EXCEPT`, `CROSS JOIN`, `JOIN`; Operadores do cálculo relacional: `EXISTS`, `NOT EXIST`, `IN`, `ANY`, `SOME`, `ALL` e operadores de comparação (=, >, <, >=, <=, <>).
d. O agrupamento em consultas SQL é realizado usando a `GROUP BY`, onde ela é usada para agrupar os resultados de uma consulta com base nos valores de uma ou mais colunas específicas. O requisito obrigatório ao usar o `GROUP BY`, é que todas as cláusulas `SELECT` que não são usadas em funções de agregação devem estar presentes nela.