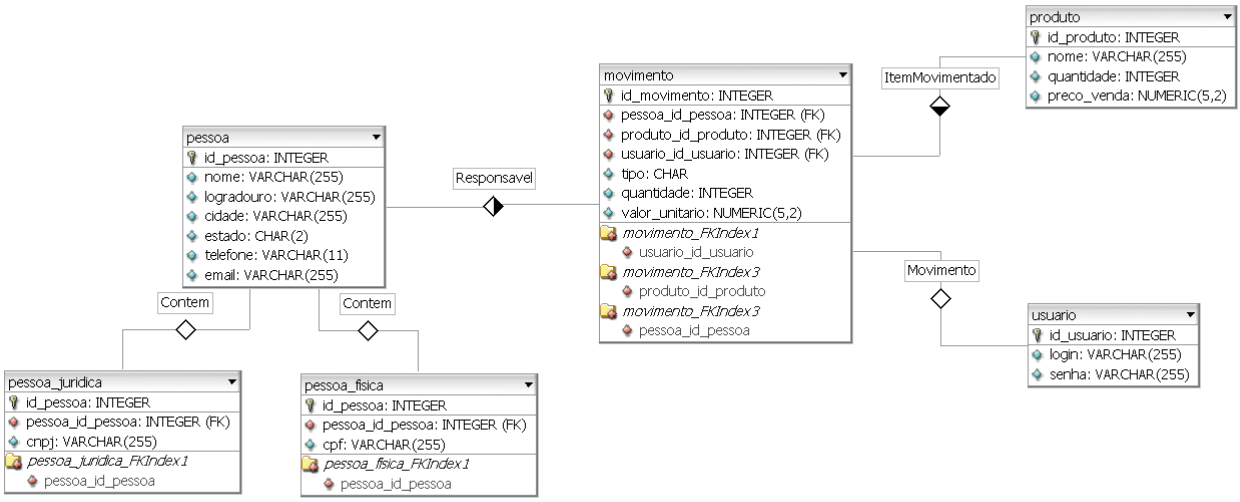
Nome do campus: Polo Pirituba
Nome do curso: Desenvolvimento Full Stack
Nome da disciplina: Nível 2: Vamos manter as informações
Número da turma: EAD - 9001
Semestre letivo: 2025.1 3 semestre
Nome do integrante da prática: Guilherme Wissenbach Ferreira
Repositório Github: https://github.com/guilherme-wissenbach/college/tree/main/Terceiro%20semestre/Nivel%202

Título da prática: Missão Prática | Nível 2 | Mundo 3

Objetivo da prática:

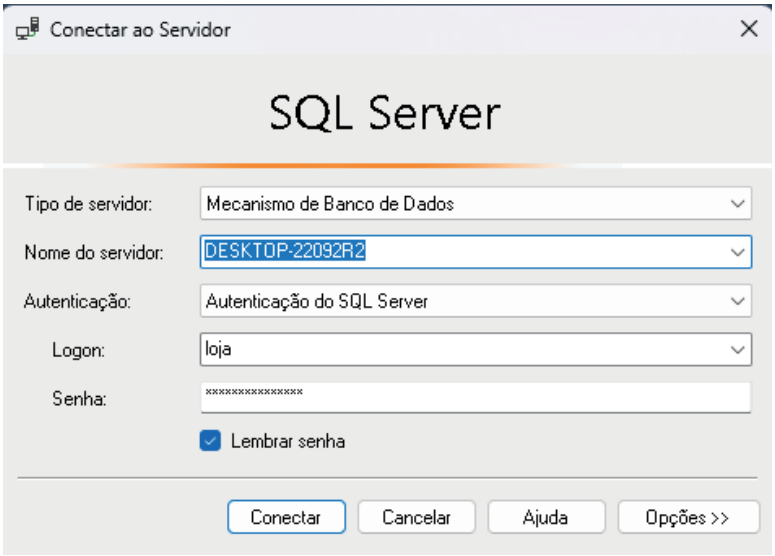
1 - Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2 - Utilizar ferramentas de modelagem para bases de dados relacionais.
3 - Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4 - Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
5 - No final 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.

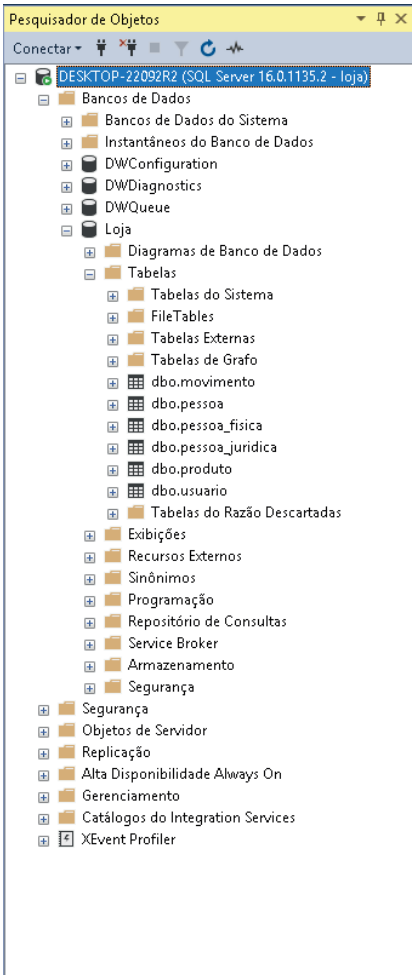
Baixar e executar a ferramenta de modelagem:
Definir o modelo de dados para um sistema com as características apresentadas nos tópicos seguintes:

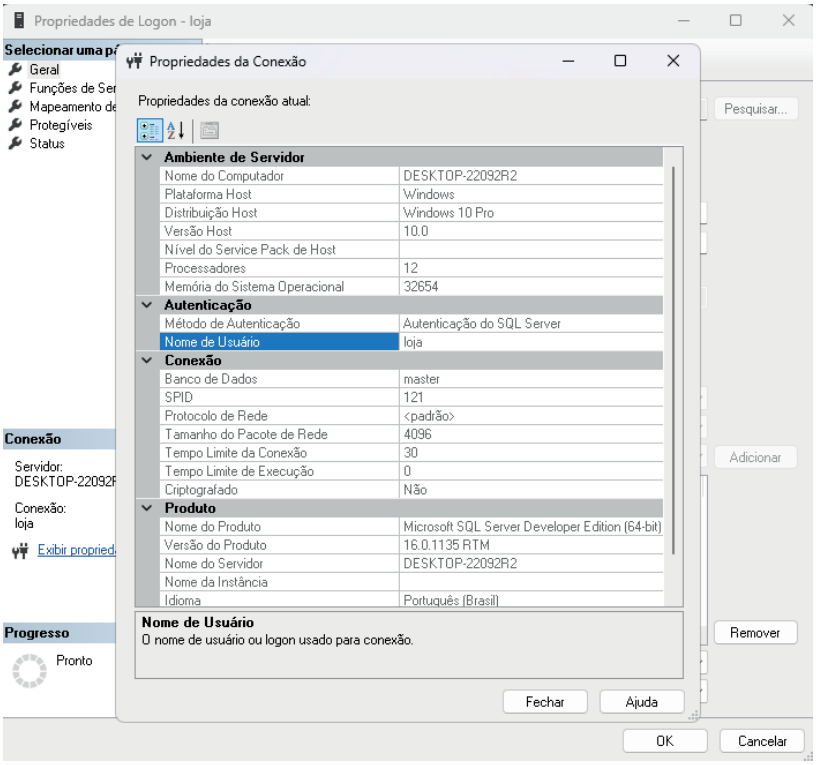


Estácio

Utilizar o SQL Server Management Studio para criar a base de dados modelada no tópico anterior:







Estácio

Procedimento 01

```
create database Loja;

use Loja;

create table pessoa(
  id_pessoa int NOT NULL ,
  nome varchar(255) NOT NULL  ,
  logradouro varchar(255) NOT  NULL  ,
  cidade varchar(255)NOT  NULL  ,
  estado char(2)NOT  NULL  ,
  telefone varchar(11)NOT  NULL  ,
  email varchar(255)NOT  NULL    ,
primary key(id_pessoa));

create table pessoa_fisica (
  id_pessoa int NOT NULL,
  cpf varchar(255)  NOT NULL,
  primary key (id_pessoa),
  foreign key (id_pessoa) references pessoa(id_pessoa));

create table pessoa_juridica (
  id_pessoa int NOT NULL,
  cnpj varchar(255)  NOT NULL,
  primary key (id_pessoa),
  foreign key (id_pessoa) references pessoa(id_pessoa));

create table produto (
  id_produto int NOT NULL  ,
  nome varchar(255)  NOT NULL  ,
  quantidade varchar(255)  NOT NULL  ,
  preco_venda numeric(5,2)  NOT NULL    ,
primary key(id_produto));

create table usuario (
  id_usuario int NOT NULL ,
  login varchar(255)   NOT NULL  ,
  senha varchar(255)   NOT NULL    ,
primary key(id_usuario));
```

```
create table movimento (
  id_movimento int NOT NULL,
  id_usuario int  NOT NULL  ,
  id_pessoa int  NOT NULL  ,
  id_produto int  NOT NULL  ,
  quantidade int  NOT NULL  ,
  tipo char  NOT NULL  ,
  valor_unitario numeric(5,2)  NOT NULL    ,
primary key(id_movimento),
foreign key (id_usuario) references usuario(id_usuario),
foreign key (id_produto) references produto(id_produto),
foreign key (id_pessoa) references pessoa(id_pessoa));

create sequence seq_pessoa
        as numeric
        start with 1
        increment by 1
        no cycle;
```

Análise e Conclusão:

Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?
Atraves dos graus de relação que entidades ou tabelas têm entre si.

Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?
A forma mais comum e recomendada para representar herança em bancos relacionais é através de relacionamentos do tipo 1x1.

Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?
Atraves de inferface gráfica intuitiva, editor de código com recursos inteligentes, ferramentas de análise e monitoramento, automatização de tarefas e facilidade na gestão de segurança.

Procedimento 02

Usuários

Use Loja;

select * from usuario;

Resultados | Mensagens

	id_usuario	login	senha
1	1	op1	op1
2	2	op2	op2

Produtos

select * from produto;

Resultados | Mensagens

	id_produto	nome	quantidade	preco_venda
1	1	Banana	100	5.00
2	3	Laranja	500	2.00
3	4	Manga	800	4.00

Movimentos

select * from movimento;

Resultados | Mensagens

	id_movimento	id_usuario	id_pessoa	id_produto	quantidade	tipo	valor_unitario
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

Procedimento 02

Dados completos de pessoas físicas e juridicas

```
-- Dados completos de pessoas físicas.
SELECT *
FROM pessoa, pessoa_fisica
WHERE pessoa.id_pessoa = pessoa_fisica.id_pessoa;

--Dados completos de pessoas jurídicas.
SELECT *
FROM pessoa, pessoa_juridica
WHERE pessoa.id_pessoa = pessoa_juridica.id_pessoa;
```

Resultados | Mensagens

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

Resultados | Mensagens

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

Procedimento 02

```
use Loja;

insert into usuario
values
   (1, 'op1', 'op1'),
   (2, 'op2', 'op2');

insert into produto
values
   (1, 'Banana', 100, 5.00),
   (3, 'Laranja', 500, 2.00),
   (4, 'Manga', 800, 4.00);

insert into pessoa
values
   (NEXT VALUE FOR seq_Pessoa, 'Joao', 'Rua 12, cas 3, Quitanda', 'Riacho do Sul', 'PA',
'1111-1111','joao@riacho.com');

insert into pessoa
values
   (NEXT VALUE FOR seq_Pessoa, 'JJC', 'Rua 11, Centro','Riacho do Norte', 'PA', '1212-1212','jjc@riacho.com');
```

```
insert into pessoa_fisica
values
   (1,'11111111111');

insert into pessoa_juridica
values
   (2,'22222222222222');

insert into movimento
values
   (1,1,1,1,20,'S',4.00),
   (4,1,1,3,15,'S',2.00),
   (5,2,1,3,10,'S',3.00),
   (7,1,2,3,15,'E',5),
   (8,1,2,4,20,'E',4.00);

-- Dados completos de pessoas físicas.
SELECT *
FROM pessoa, pessoa_fisica
WHERE pessoa.id_pessoa = pessoa_fisica.id_pessoa;

--Dados completos de pessoas jurídicas.
SELECT *
FROM pessoa, pessoa_juridica
WHERE pessoa.id_pessoa = pessoa_juridica.id_pessoa;

--Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.
SELECT id_movimento, movimento.id_produto, produto.nome as 'Produto',pessoa.id_pessoa,
pessoa.nome as 'Fornecedor', movimento.quantidade, valor_unitario,
(movimento.quantidade *  valor_unitario) as valor_total
FROM movimento
JOIN pessoa
   ON movimento.id_pessoa = pessoa.id_pessoa
JOIN produto
   ON movimento.id_produto = produto.id_produto
WHERE movimento.tipo = 'E';

--Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total
SELECT
   id_movimento,
   movimento.id_produto,
   produto.nome as 'Produto',
   movimento.id_pessoa,
   pessoa.nome as 'Comprador',
   movimento.quantidade,
   valor_unitario,
   (movimento.quantidade * valor_unitario) as valor_total
FROM movimento
```

```
JOIN pessoa
   ON movimento.id_pessoa = pessoa.id_pessoa
JOIN produto
   ON movimento.id_produto = produto.id_produto
WHERE movimento.tipo = 'S';

--Valor total das entradas agrupadas por produto.
SELECT
   produto.nome,
  SUM(movimento.quantidade * movimento.valor_unitario) AS 'VALOR TOTAL ENTRADAS'
FROM movimento
JOIN produto
   ON produto.id_produto = movimento.id_produto
WHERE movimento.tipo = 'E'
GROUP BY produto.nome;

--Valor total das saídas agrupadas por produto.
SELECT produto.nome, SUM (movimento.quantidade * movimento.valor_unitario) AS 'VALOR TOTAL
SAIDAS'
FROM movimento
JOIN produto
   ON produto.id_produto = movimento.id_produto
WHERE movimento.tipo = 'S'
GROUP BY produto.nome;

--Operadores que não efetuaram movimentações de entrada (compra).
SELECT movimento.id_usuario AS 'ID DO OPERADOR'
FROM movimento
except
SELECT movimento.id_usuario
FROM movimento
WHERE movimento.tipo = 'E';

--Valor total de entrada, agrupado por operador.
SELECT usuario.login AS OPERADOR, SUM (movimento.quantidade * movimento.valor_unitario) AS 'VALOR
TOTAL ENTRADAS'
FROM movimento
JOIN usuario
   ON usuario.id_usuario = movimento.id_usuario
WHERE movimento.tipo = 'E'
GROUP BY usuario.login;

--Valor total de saída, agrupado por operador.
SELECT usuario.login AS OPERADOR, SUM (movimento.quantidade * movimento.valor_unitario) AS 'VALOR
TOTAL SAIDAS'
FROM movimento
JOIN usuario
```

```
  ON usuario.id_usuario = movimento.id_usuario
WHERE movimento.tipo = 'S'
GROUP BY usuario.login;

--Valor médio de venda por produto, utilizando média ponderada.
SELECT produto.nome, SUM (movimento.quantidade * movimento.valor_unitario) /
SUM(movimento.quantidade) as 'Valor médio de venda'
FROM movimento
JOIN produto
  ON produto.id_produto = movimento.id_produto
WHERE movimento.tipo = 'S'
GROUP BY produto.nome;
```

Análise e Conclusão:

Quais as diferenças no uso de sequence e identity?

No SQL, tanto sequence quanto identity são utilizados para gerar valores numéricos automáticos, normalmente aplicados a colunas que funcionam como chaves primárias. O identity tem uma vantagem de ser mais simples e não exige objetos separados nem chamadas adicionais, mas ele acaba sendo limitado ao escopo da tabela, ja o sequence ele permite gerar valores numéricos de maneira mais flexível e podendo ser utilizado por várias tabelas e permite configurações avançadas.

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

Ela evita dados órfãos, mantém a integridade dos relacinamentos, facilita a manutenção e a organização dos dados e melhora a confiabilidade dos dados.

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

Operadores do SQL que pertencem à álgebra relacional:
SELEÇÃO, PROJEÇÃO, UNIÃO, DIFERENÇA, PRODUTO CARTESIANO, JUNÇÃO, DIVISÃO, RENOMEAÇÃO, ATRIBUIÇÃO;

Operadores do SQL que pertencem ao cálculo relacional:
PREDICADOS LÓGICOS, QUANTIFICADORES E EXPRESSÕES BOOLEANAS.

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

O agrupamento em consultas é feito utilizando o “GROUP BY”, que organiza os dados em grupos com base em uma ou mais colunas.