UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier

Repositório do Github:
https://github.com/Watthier09/RPG0015---Vamos-manter-as-informa-es-

1 - Título da prática:

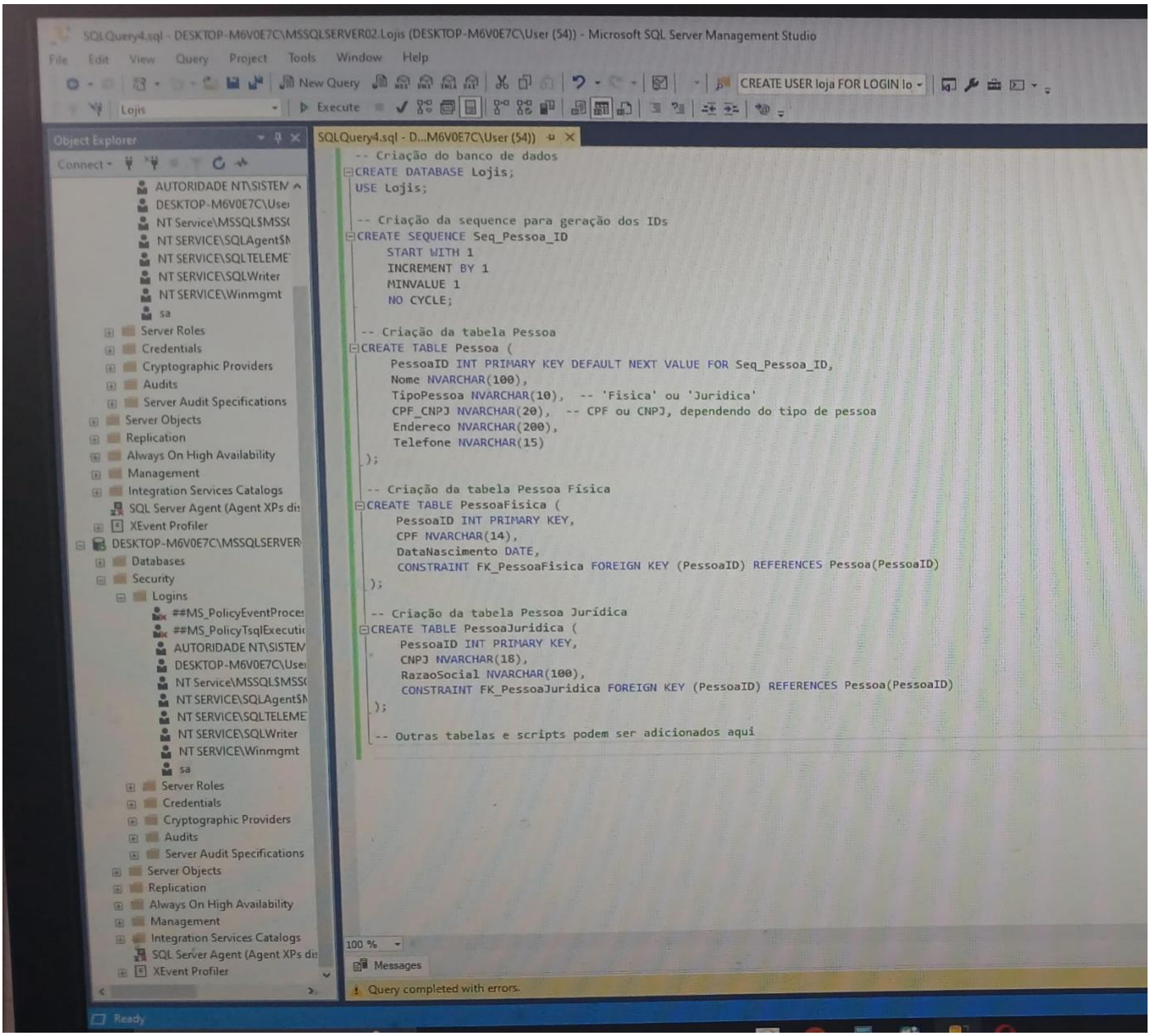
RPG0015 - Vamos manter as informações!
Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

2 - Objetivo da Prática:

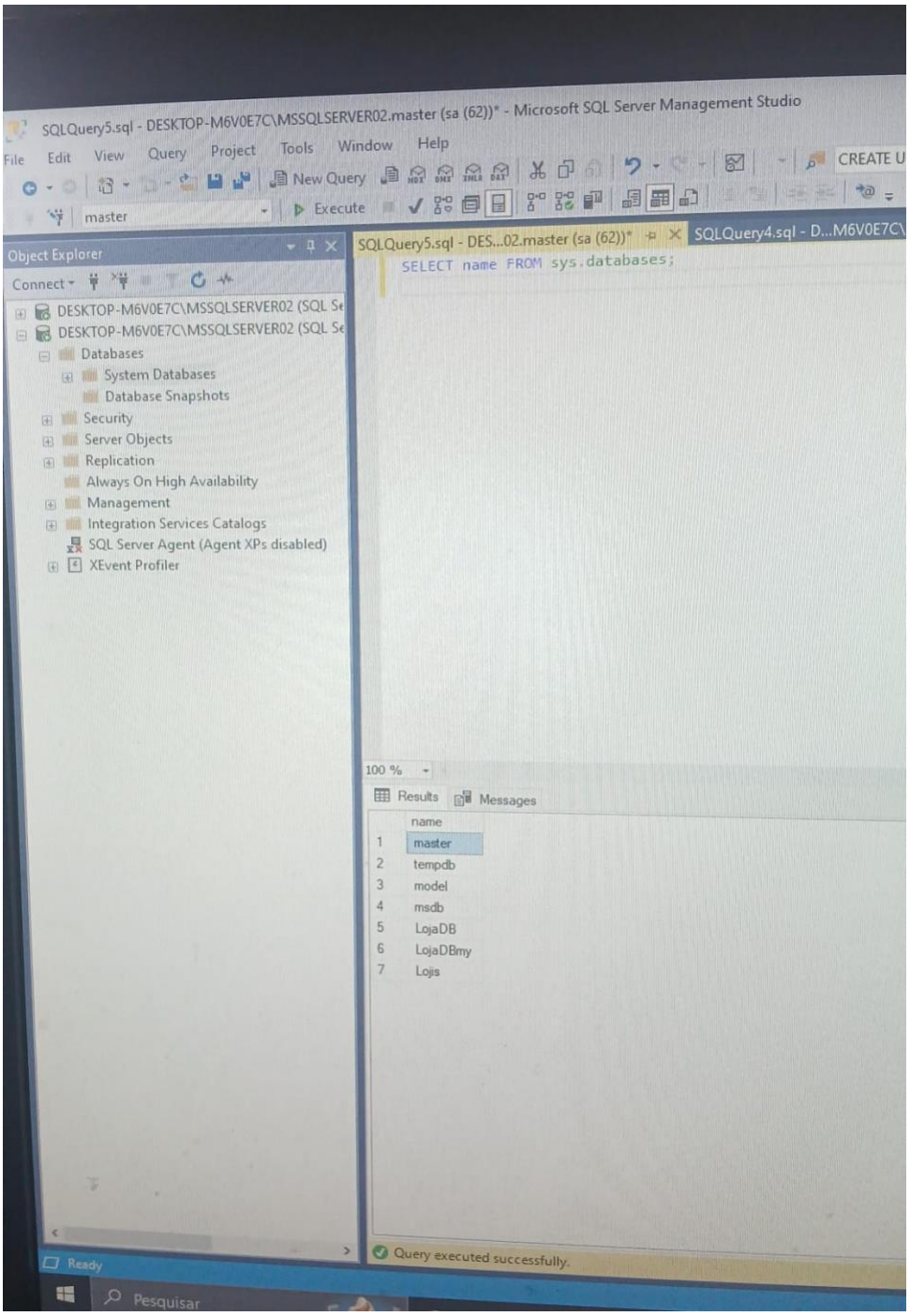
1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
5. No final do exercício,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

3 - Alguns Códigos registrados nesse desenvolvimento

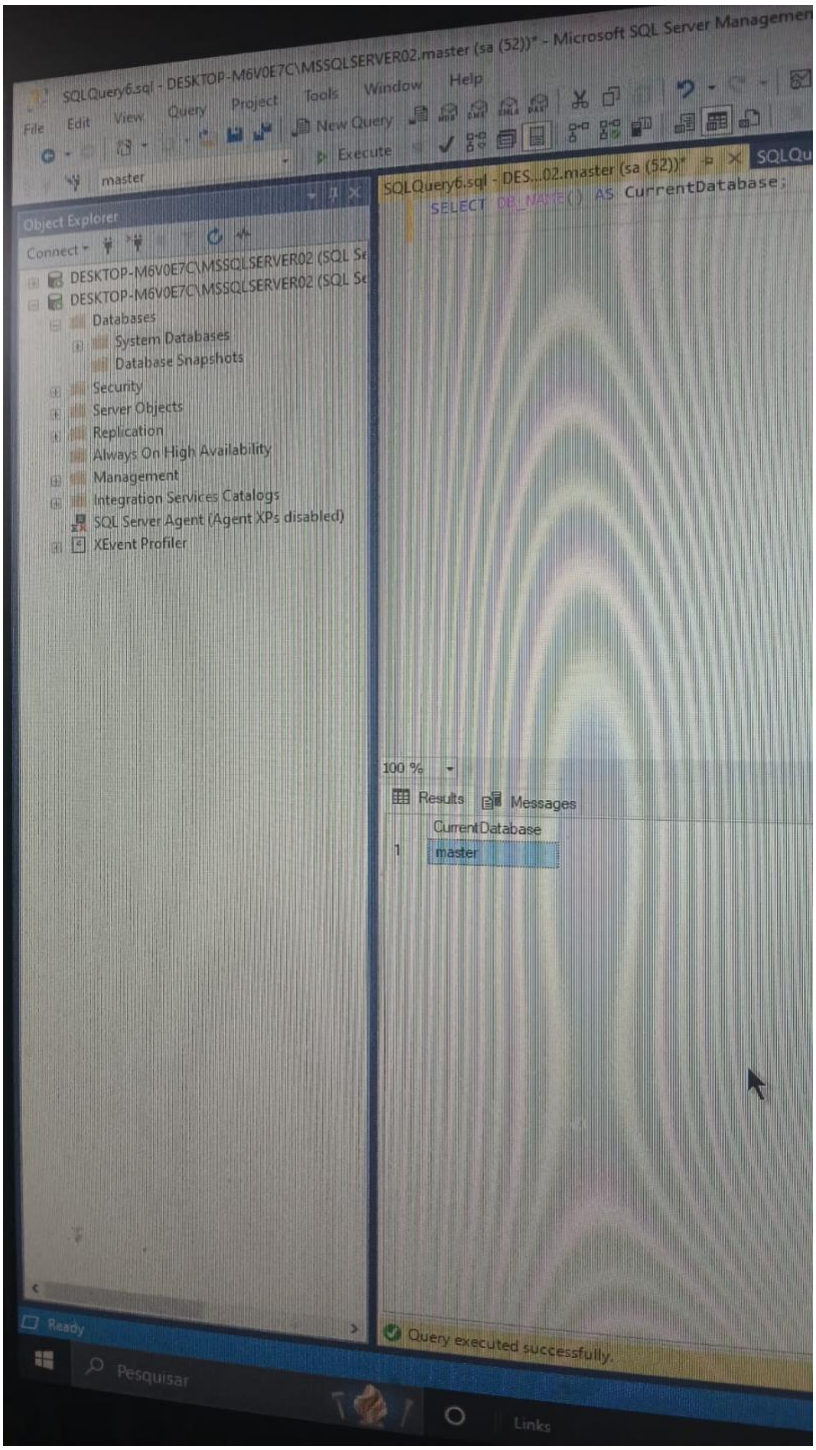
UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier



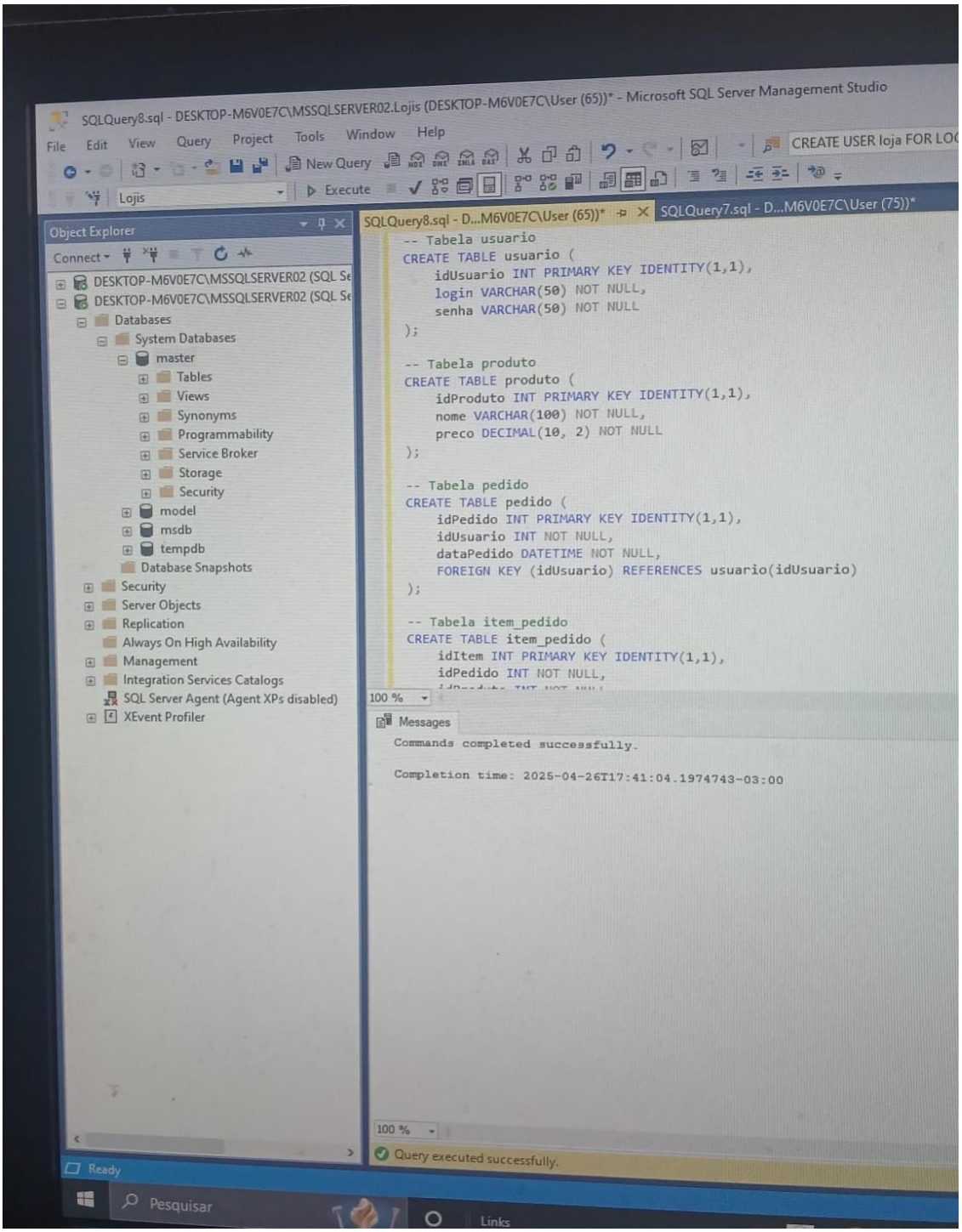
UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier



UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier



UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier



UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier

Análise e Conclusão:

a. Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

As cardinalidades em um banco de dados relacional representam a quantidade de entidades que podem estar relacionadas entre si. Cada tipo de relacionamento tem uma forma específica de implementação:

1x1 (Um para um):

Neste tipo de relacionamento, uma linha de uma tabela se relaciona com uma linha de outra tabela. Para implementar isso em um banco de dados relacional, é comum adicionar uma chave estrangeira (foreign key) na tabela "dependente", que referencia a tabela "principal". O relacionamento 1x1 pode ser mantido com uma restrição de unidade na chave estrangeira.

Exemplo: Em um sistema de gestão de usuários, um usuário pode ter um único perfil de login. Assim, as tabelas "Usuários" e "Perfis" podem ser relacionadas de maneira 1x1, com a tabela "Perfis" contendo uma chave estrangeira que referencia a tabela "Usuários".

1xN (Um para muitos):

Um relacionamento 1xN ocorre quando uma linha de uma tabela se relaciona com várias linhas de outra tabela. Para implementar isso, uma chave estrangeira é colocada na tabela "muitos" que referencia a tabela "um". Esse tipo de relacionamento é muito comum em bancos de dados relacionais.

Exemplo: Em um sistema de vendas, uma tabela "Clientes" pode se relacionar com várias linhas na tabela "Pedidos". Cada cliente pode ter múltiplos pedidos, mas cada pedido pertence a um único cliente. N x N (Muitos para muitos):

UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier

Para implementar um relacionamento N x N, é necessário criar uma tabela de junção, também conhecida como tabela intermediária. Essa tabela possui duas chaves estrangeiras, cada uma referenciando uma das tabelas relacionadas. A tabela de junção permite representar a relação entre as duas tabelas de forma eficiente.

Exemplo: Em um sistema educacional, uma tabela "Alunos" pode ter um relacionamento muitos para muitos com a tabela "Cursos", pois um aluno pode estar matriculado em vários cursos, e um curso pode ter vários alunos matriculados. Para representar isso, criamos uma tabela intermediária "Matriculas" que contém as chaves estrangeiras de "Alunos" e "Cursos".

b. Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

A herança, como um conceito de programação orientada a objetos, não é representada diretamente em bancos de dados relacionais, mas existem estratégias que podem ser utilizadas para representar esse conceito. As abordagens mais comuns são:
Tabela Única (Single Table Inheritance):

Nesse modelo, todos os tipos de entidades herdam de uma única tabela. Essa tabela contém todas as colunas necessárias para todas as subclasses, e uma coluna adicional para identificar o tipo de entidade. Embora seja simples, essa abordagem pode gerar muitas colunas nulas e um grande número de colunas na tabela, o que pode prejudicar o desempenho em grandes sistemas.

Exemplo: Se tivermos uma classe base "Pessoa" e subclasses "PessoaFisica" e "PessoaJuridica", podemos usar uma única tabela "Pessoas" para armazenar todos os dados, com uma coluna adicional que define o tipo de pessoa (Física ou Jurídica). Tabela por Tipo (Class Table Inheritance):

Nesse modelo, cada tipo de entidade tem sua própria tabela, e a tabela da superclasse (entidade base) contém os dados comuns, enquanto as tabelas das subclasses contêm os dados específicos. Cada tabela de subclasse tem uma chave primária que é também uma chave estrangeira referenciando a

UNIVERSIDADE ESTÁCIO DE SÁ
POLO CENTRO – IGREJINHA – RS
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Manter as Informações (RPG0015)
Turma: 2024.1 | 3º semestre
Nome: Emily Duarte Watthier

tabela da superclasse. Esse modelo é mais normalizado, mas pode gerar joins mais complexos entre tabelas.

Exemplo: A tabela "Pessoas" armazena os dados comuns, como nome e endereço, enquanto as tabelas "PessoaFisica" e "PessoaJuridica" armazenam os dados específicos, como CPF e CNPJ. Tabela Separada por Entidade (Concrete Table Inheritance):

Nesse modelo, cada tipo de entidade tem sua própria tabela, e não há tabela de superclasse. Ou seja, todas as tabelas têm suas próprias colunas, sem referência a uma tabela comum. Este modelo pode resultar em duplicação de dados, mas simplifica a consulta.

Exemplo: Tabelas "PessoaFisica" e "PessoaJuridica" podem ter todos os dados necessários de forma independente, sem a necessidade de uma tabela de "Pessoas" que armazene os dados comuns.