

Projetar Software é Difícil?

Explorando os Desafios e Complexidades do
Desenvolvimento de Software

Paulo Vinícius Moreira Dutra¹

¹IF Sudeste MG – Campus Muriaé

Engenharia de Software, 2024


Sumário

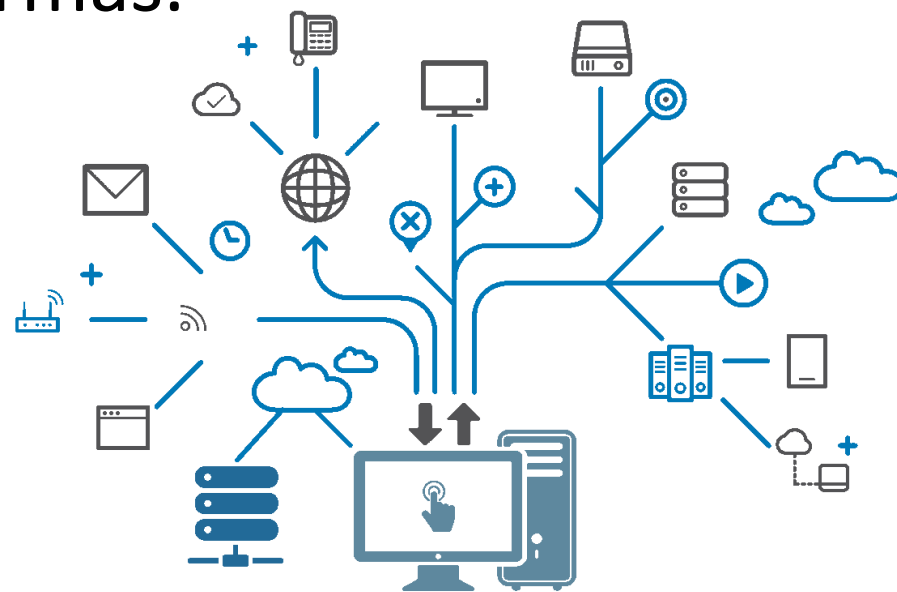
- **Parte I - Introdução**
 - Por que projetar software é considerado difícil?

Parte I

- Por que projetar software é considerado difícil?

Complexidade Técnica

- Diversidade de tecnologias e plataformas.
 - Integração com sistemas legados.
 - Gerenciamento de performance e escalabilidade.
- 
- Um diagrama de integração de sistemas. No centro, um ícone de globo terrestre está conectado por linhas azuis a vários outros ícones: um telefone, uma nuvem com um checkmark, um envelope, um relógio, um sinal de Wi-Fi, um servidor, um monitor com uma mão tocando a tela, um sinal de localização com um 'X', e um sinal de Wi-Fi com um 'X'. Há também ícones de uma nuvem, um sinal de Wi-Fi, e um sinal de localização com um checkmark. O diagrama ilustra a conexão entre diferentes tecnologias e plataformas.



Requisitos em Constante Mudança



Expectativas dos stakeholders.



Mudanças no mercado e nas necessidades do usuário.



Dificuldades em capturar todos os requisitos desde o início.

Erros Comuns de Iniciantes



Trabalho em equipe:
desenvolvedores, designers,
gerentes de projeto.



Alinhamento entre diferentes
disciplinas (design, backend,
frontend).



Ferramentas de colaboração e
suas limitações.

O Mundo real é diferente do
acadêmico. Ele é cruel.

Erros Comuns de Iniciantes

Erros Comuns de Iniciantes

- Subestimar a Complexidade do Projeto:
 - Iniciantes muitas vezes acreditam que o projeto será mais simples do que realmente é, ignorando detalhes críticos.
- Falta de Clareza nos Requisitos:
 - Falham em capturar e entender todos os requisitos necessários pode levar a problemas significativos mais tarde no desenvolvimento.

Erros Comuns de Iniciantes

- Negligenciar o Teste e Depuração:
 - Desenvolvedores iniciantes não dedicam tempo suficiente para testar e depurar o software, o que pode resultar em bugs persistentes.
- Não Planejar Mudanças Futuras:
 - Iniciantes tendem a criar soluções rígidas que não são fáceis de adaptar ou expandir conforme as necessidades do projeto mudam.

Erros Comuns de Iniciantes

- Foco Excessivo em Detalhes Técnicos:
 - Concentrar-se demais em aspectos técnicos específicos sem considerar a visão geral do projeto pode levar a um produto final desconexo e incoerente.

Erros Comuns de Iniciantes

- Falta de Conhecimento do Domínio do Problema:
 - Falta de compreensão do ambiente real em que o software será utilizado, como uma empresa de e-commerce.
 - Resultando em soluções que não atendem às necessidades práticas dos usuários ou que são difíceis de integrar nos fluxos de trabalho existentes.

A Importância de conhecer o domínio da aplicação



Importância de conhecer o domínio da aplicação

- Identificação de Necessidades Reais: Entender o domínio permite captar as necessidades específicas do negócio.
- Requisitos Precisos: Evita interpretações erradas e ajuda a definir requisitos que realmente atendem ao usuário.
- Adequação ao Contexto: O software desenvolvido será mais relevante e eficaz para os usuários finais.

Importância de conhecer o domínio da aplicação

- Redução de Erros: Conhecimento do domínio minimiza a chance de erros durante o desenvolvimento e na fase de testes.
- Resolução Problemas Específicos: Facilita a identificação e solução de problemas complexos que podem surgir no sistema.

Importância de conhecer o domínio da aplicação

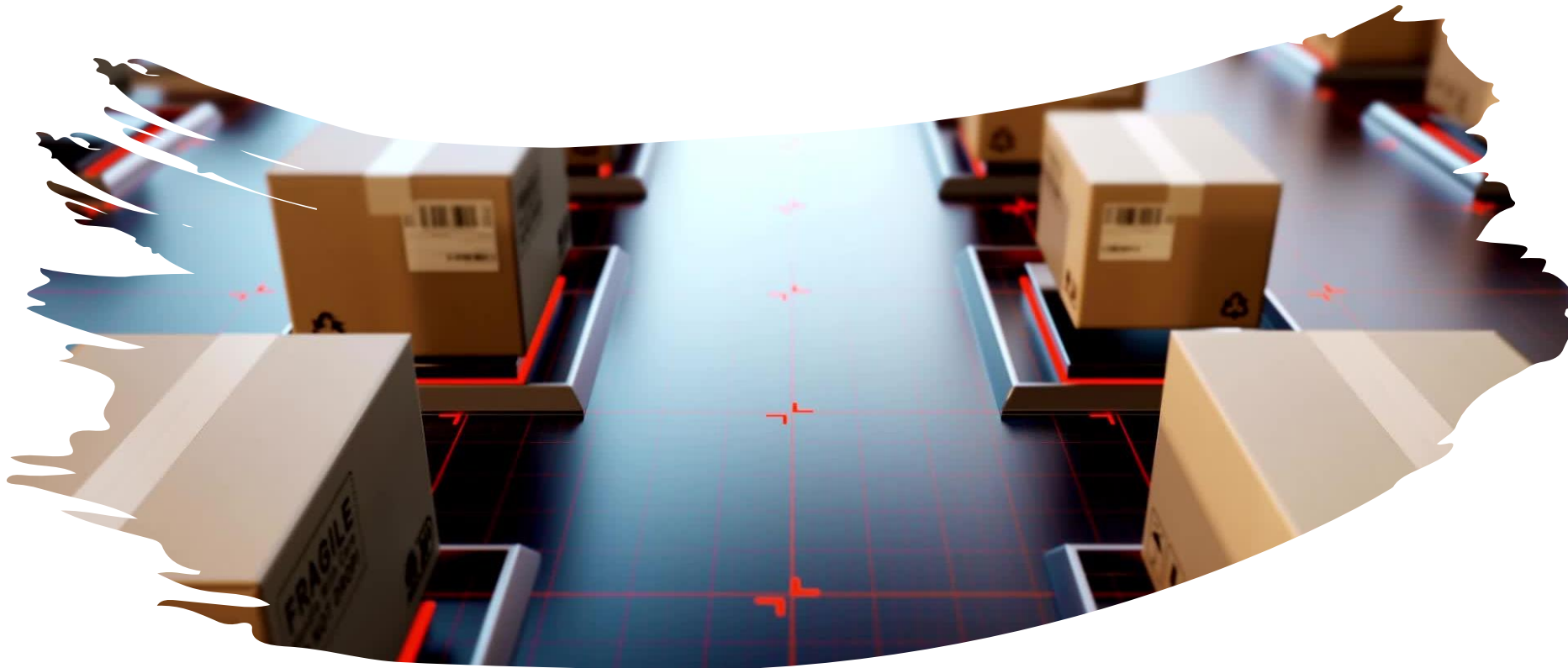
- Tomada de Decisões assertivas: Permite fazer escolhas técnicas que são mais adequadas ao contexto do negócio.
- Linguagem Comum: Facilita a comunicação entre desenvolvedores e especialistas do domínio.
- Expectativas Alinhadas: Reduz mal-entendidos e garante que o produto final corresponda às expectativas.

Importância de conhecer o domínio da aplicação

- Facilidade em Implementar Mudanças: Conhecimento do domínio permite ajustar o software rapidamente conforme o negócio evolui.
- Manutenibilidade a Longo Prazo: Construir uma base sólida para futuras expansões e manutenções do sistema.

Como
solucionar
um
problema?





O problema: Sistema de Cadastro com
Endereços de Cobrança e Entrega

Objetivo: Desenvolver um sistema de cadastro de clientes onde cada cliente possui um endereço de cobrança padrão único e pode ter vários endereços de entrega.



Antes vamos
relembrar a
normalização
de banco de
dados.



Porque a normalização foi criada?

- Normalização é uma técnica usada para organizar os dados em um banco de dados.
- Seu objetivo principal é reduzir a redundância e evitar inconsistências nos dados.
- Foi desenvolvida para garantir a integridade dos dados e melhorar a eficiência no armazenamento dos dados.

Tipos de normalização

- 1NF (Primeira Forma Normal): Elimina grupos repetidos e garante que os dados são atômicos (não divisíveis).
- 2NF (Segunda Forma Normal): Remove dependências parciais de uma chave composta.
- 3NF (Terceira Forma Normal): Elimina dependências transitivas entre colunas não-chave.
- 4NF (Quarta Forma Normal): Evita dependências multivaloradas.

Modelagem do banco de dados:

1. Entenda das regras de negócio do sistema;
2. Identifique as possíveis entidades e atributos;
3. Defina os relacionamentos entre as entidades
4. Normalize o modelo de entidade e relacionamento. Preciso realmente?

Modelagem do banco de dados: Relembrando normalização – 1NF

- Primeira forma normal (1NF): É o primeiro passo na normalização de um banco de dados e se concentra em organizar os dados de forma que eles estejam em um formato mais simples e eficiente.
- Definição: Uma tabela estará na 1NF, se, e somente se, todas as colunas tiverem apenas valores atômicos, ou seja, cada coluna só puder ter um valor para cada linha da tabela.

Modelagem do banco de dados: Relembrando normalização – 1NF

- Na 1NF não deve haver atributos compostos ou multivalorados.
- O campo **Endereço Cliente** é um atributo composto, neste caso, é necessário desmembrar esse campo.

ID Pedido	ID Produto	Data Pedido	Nome Produto	Quantidade	Preço Unitário	Nome Cliente	Endereço Cliente
1	101	01/01/2024	Notebook	2	3000	João Silva	Rua A, 123, Centro
1	102	01/01/2024	Mouse	1	100	João Silva	Rua A, 123, Centro
2	103	15/02/2024	Teclado	1	150	Maria Santos	Rua B, 456, Barra
2	104	15/02/2024	Monitor	2	700	Maria Santos	Rua B, 456, Barra

Modelagem do banco de dados: Relembrando normalização – 1NF

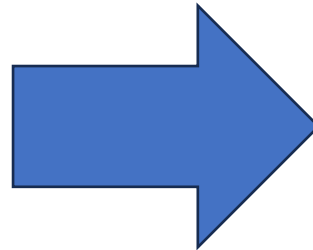
- Campo **Endereço Cliente** com os dados desmembrados:

ID Pedido	ID Produto	Data Pedido	Nome Produto	Quantidade	Preço Unitário	Nome Cliente	Rua	Número	Bairro
1	101	01/01/2024	Notebook	2	3000	João Silva	Rua A	123	Centro
1	102	01/01/2024	Mouse	1	100	João Silva	Rua A	123	Centro
2	103	15/02/2024	Teclado	1	150	Maria Santos	Rua B	456	Barra
2	104	15/02/2024	Monitor	2	700	Maria Santos	Rua B	456	Barra

Modelagem do banco de dados: Relembrando normalização – 1NF

- E para atributos multivalorados? Desmembre os dados em linhas separadas.

ID Cliente	Nome	Telefone
1	João	3722-5555, 3722-4444
2	Maria	3721-2222, 3721-3333

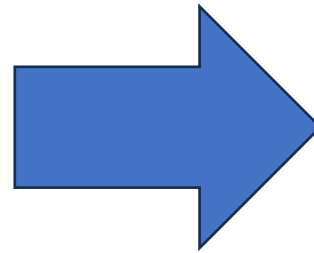


ID Cliente	Nome	Telefone
1	João	3722-5555
1	João	3722-4444
2	Maria	3721-2222
2	Maria	3721-3333

Modelagem do banco de dados: Relembrando normalização – 1NF

- Como próximo passo, elimine grupos repetidos criando tabelas separadas e relacionadas:

ID Cliente	Nome
1	João
2	Maria



ID Contato	ID Cliente	Telefone
1	1	3722-5555
2	1	3722-4444
3	2	3721-2222
4	2	3721-3333

Modelagem do banco de dados: Relembrando normalização – 2NF

- Segunda forma normal (2NF): É utilizada para eliminar dependências parciais em tabelas que possuam chave primária composta e não devem possuir chaves duplicadas.

ID Pedido	ID Produto	Data Pedido	Nome Produto	Quantidade	Preço Unitário	Nome Cliente	Rua	Número	Bairro
1	101	01/01/2024	Notebook	2	3000	João Silva	Rua A	123	Centro
1	102	01/01/2024	Mouse	1	100	João Silva	Rua A	123	Centro
2	103	15/02/2024	Teclado	1	150	Maria Santos	Rua B	456	Barra
2	104	15/02/2024	Monitor	2	700	Maria Santos	Rua B	456	Barra

Modelagem do banco de dados: Relembrando normalização – 2NF

- **ID Pedido** e **ID Produto** são as chaves primárias da tabela Pedido.
- Dependências parciais: Na tabela pedido, os atributos **Data Pedido**, **Nome do Cliente**, **Rua**, **Número** e **Bairro** dependem parcialmente do **ID Pedido** e não do **ID Produto**. Neste caso, temos uma dependência parcial que viola a 2NF.

Modelagem do banco de dados: Relembrando normalização – 2NF

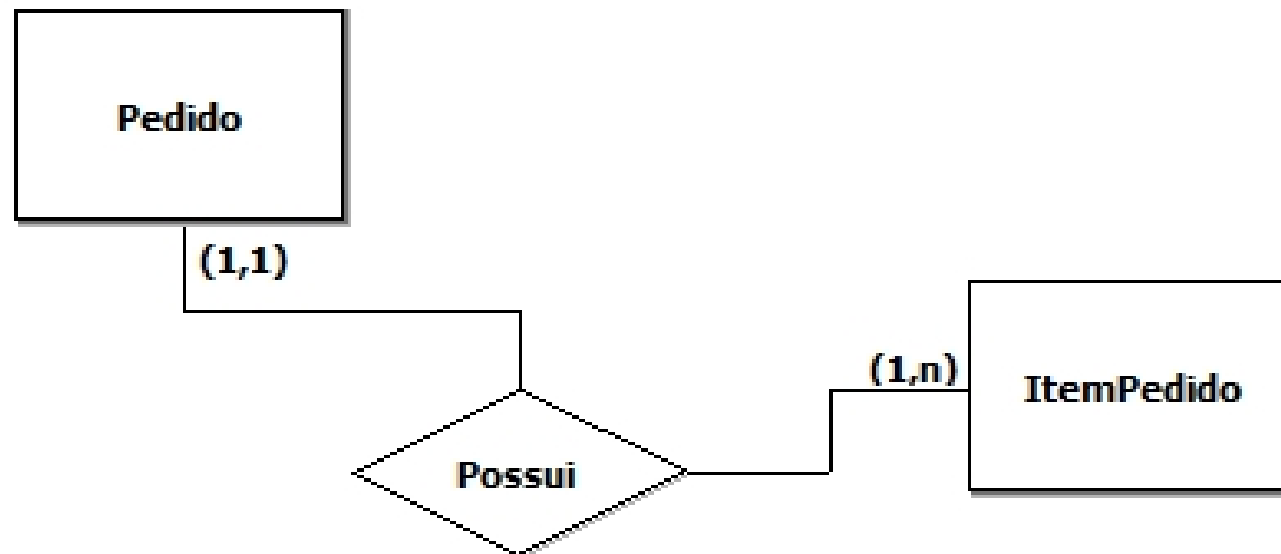
- Resolvendo: Crie tabelas separadas para os atributos com dependência parcial.

ID Pedido	Data Pedido	Nome Cliente	Rua	Número	Bairro
1	01/01/2024	João Silva	Rua A	123	Centro
2	15/02/2024	Maria Santos	Rua B	456	Barra

ID Pedido	ID Produto	Nome Produto	Quantidade	Preço Unitário
1	101	Notebook	2	3000
1	102	Mouse	1	100
2	103	Teclado	1	150
2	104	Monitor	2	700

Modelagem do banco de dados: Relembrando normalização – 2NF

- Nosso primeiro modelo de entidade e relacionamento:



Modelagem do banco de dados: Relembrando normalização – 2NF

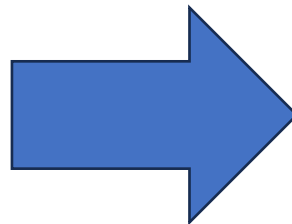
- Resolvendo: Na entidade **Itens Pedido**, ainda temos na mesma entidade o atributo **Nome Produto** que depende parcialmente de **ID Produto**.

ID Pedido	ID Produto	Nome Produto	Quantidade	Preço Unitário
1	101	Notebook	2	3000
1	102	Mouse	1	100
2	103	Teclado	1	150
2	104	Monitor	2	700

Modelagem do banco de dados: Relembrando normalização – 2NF

- Resolvendo: Separamos os atributos, **ID Produto**, **Nome Produto** e **Preço Unitário** em uma terceira tabela:

ID Produto	Nome Produto	Preço Unitário
101	Notebook	3000
102	Mouse	100
103	Teclado	150
104	Monitor	700

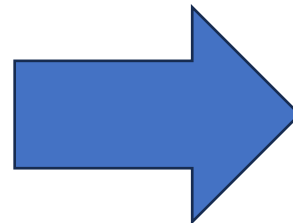


ID Pedido	ID Produto	Quantidade	Preço Unitário
1	101	2	3000
1	102	1	100
2	103	1	150
2	104	2	700

Modelagem do banco de dados: Relembrando normalização – 2NF

- Por que manter o **Preço Unitário** na tabela Item Pedido? Neste caso, fazemos isso com o objetivo de manter um histórico de preços ao longo da realização dos pedidos.

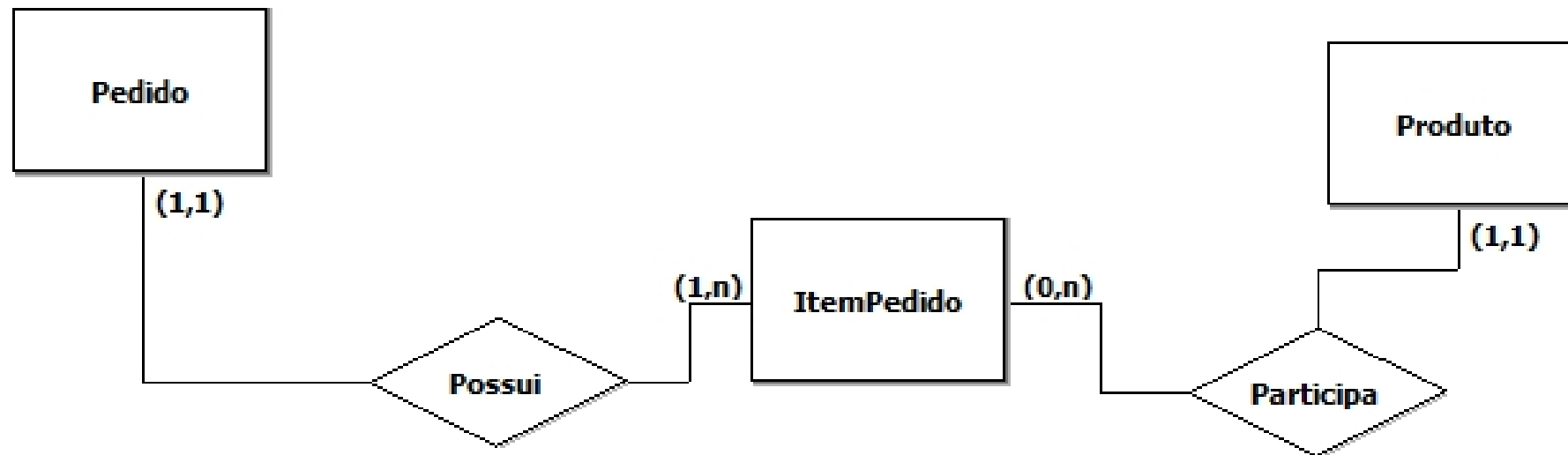
ID Produto	Nome Produto	Preço Unitário
101	Notebook	3000
102	Mouse	100
103	Teclado	150
104	Monitor	700



ID Pedido	ID Produto	Quantidade	Preço Unitário
1	101	2	3000
1	102	1	100
2	103	1	150
2	104	2	700

Modelagem do banco de dados: Relembrando normalização – 2NF

- Ao final, temos a nossa representação no modelo de entidade e relacionamento:



Modelagem do banco de dados: Relembrando normalização – 3NF

- Terceira forma normal (3NF): É aplicada para eliminar dependências transitivas. Uma tabela está na 3NF se ela já estiver na 2NF e todos os atributos não chave forem dependentes diretamente da chave primária, e não de outro atributo não chave.

Modelagem do banco de dados: Relembrando normalização – 3NF

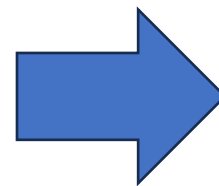
- Dependências Transitivas: Nesta tabela, **Rua**, **Número** e **Bairro**, dependem do **Nome Cliente**, que por sua vez depende de **ID Pedido**. Isso significa que temos uma dependência transitiva, o que viola a 3NF.

ID Pedido	Data Pedido	Nome Cliente	Rua	Número	Bairro
1	01/01/2024	João Silva	Rua A	123	Centro
2	15/02/2024	Maria Santos	Rua B	456	Barra

Modelagem do banco de dados: Relembrando normalização – 3NF

- Após identificar os atributos com dependência transitiva, separamos esses atributos em outra tabela.
- Como Nome Cliente não pode ser utilizado como chave primária, criamos um campo identificador para ele:

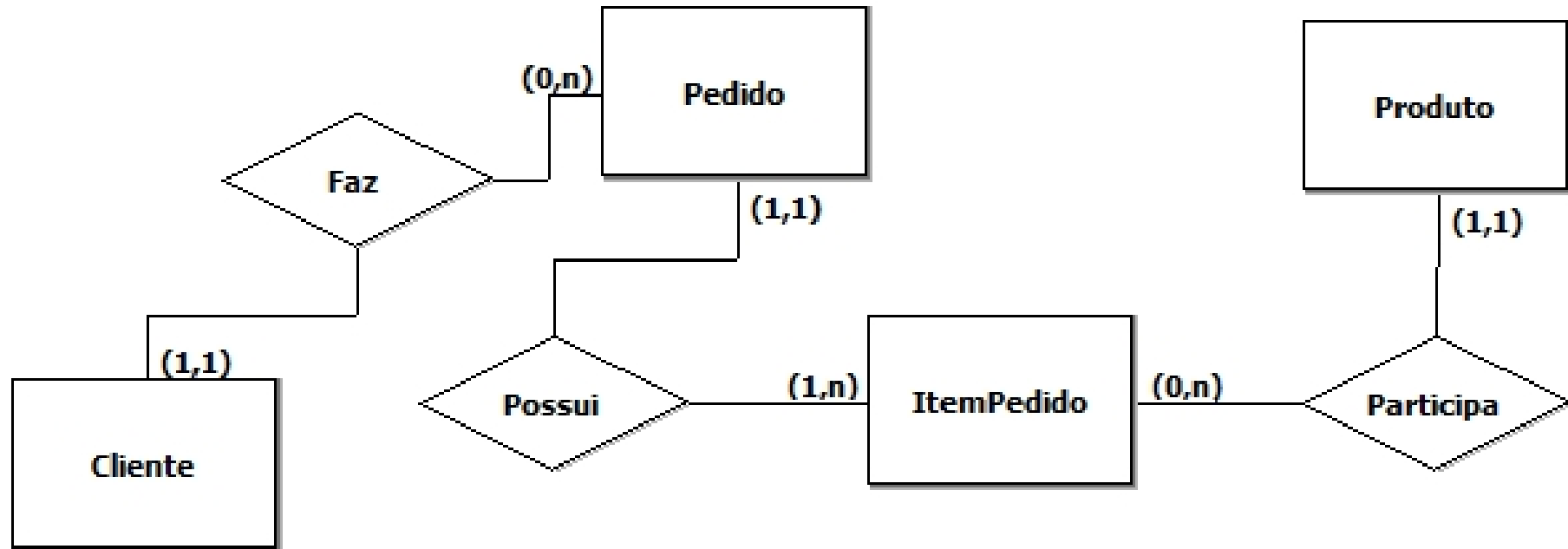
ID Cliente	Nome Cliente	Rua	Número	Bairro
1	João Silva	Rua A	123	Centro
2	Maria Santos	Rua B	456	Barra



ID Pedido	ID Cliente	Data Pedido.
1	1	01/01/2024
2	2	15/02/2024

Modelagem do banco de dados: Relembrando normalização – 3NF

- Representação do nosso modelo normalizado:



Retornando ao nosso problema:

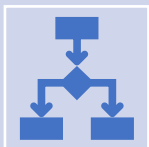
Desenvolver um sistema de cadastro de clientes onde cada cliente possui um endereço de cobrança padrão único e pode ter vários endereços de entrega.



Requisitos:



Cada cliente deve ter um único endereço de cobrança.



O cliente pode cadastrar múltiplos endereços de entrega.



O sistema deve permitir a seleção de um endereço de entrega durante a realização de uma compra.

Modelagem do Banco de Dados

- Como primeiro passo, identificamos os possíveis campos e definimos algumas informações para que possamos entender como armazenar os dados.

ID Cliente	Nome	Email	Telefone	Endereço de Cobrança	Endereço de Entrega
1	Paulo	paulo@gmail.com	3721-5555	Rua A, 123, Barra	Rua B, 456, Centro; Rua C, 789, Barra
2	João da Silva	joao@gmail.com	3722-2222	Rua B, 23, Centro	Rua C, 150, União; Rua B, 50, Centro

Modelagem do Banco de Dados

- Segundo passo: Separar os campos com valores compostos e multivalorados. Criar a chave primária em cada tabela.

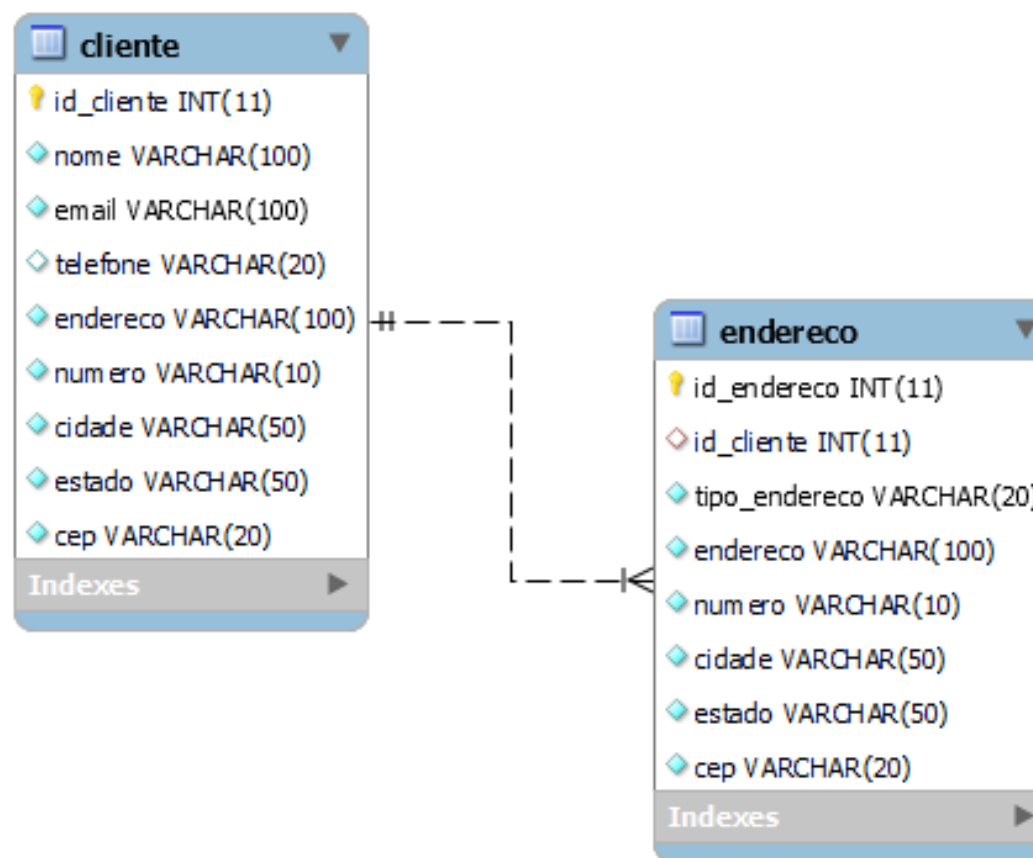
ID Cliente	Nome	Email	Telefone	Endereço	Número	Bairro	Cidade	Estado	CEP
1	Paulo	paulo@gmail.com	3721-5555	Rua A	123	Barra	Muriaé	MG	36880-002
2	João da Silva	joao@gmail.com	3722-2222	Rua B	23	Centro	Muriaé	MG	36880-000



ID Endereço	ID Cliente	Endereço	Número	Bairro	Cidade	Estado	CEP
1	1	Rua B	456	Centro	Muriaé	MG	36880-000
2	1	Rua C	789	Barra	Muriaé	MG	36880-001

Modelagem do Banco de Dados

- Ao final teremos o seguinte modelo de entidade e relacionamento:



Modelagem do Banco de Dados

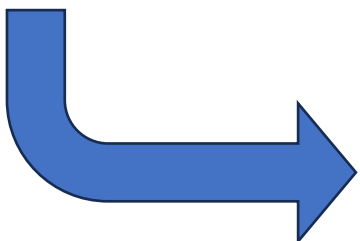
- Se quisermos diminuir as redundâncias em nosso modelo?
Como o endereço na tabela Cliente.

ID Cliente	Nome	Email	Telefone	Endereço	Número	Bairro	Cidade	Estado	CEP
1	Paulo	paulo@gmail.com	3721-5555	Rua A	123	Barra	Muriaé	MG	36880-002
2	João da Silva	joao@gmail.com	3722-2222	Rua B	23	Centro	Muriaé	MG	36880-000

Modelagem do Banco de Dados

- Neste caso, devemos utilizar a normalização na 3NF. Devemos remover a chave estrangeira da tabela Endereço e definir uma chave estrangeira na tabela Cliente

ID Endereço	Endereço	Número	Bairro	Cidade	Estado	CEP
1	Rua B	456	Centro	Muriaé	MG	36880-000
2	Rua C	789	Barra	Muriaé	MG	36880-001
3	Rua A	123	Barra	Muriaé	MG	36880-002
4	Rua B	23	Centro	Muriaé	MG	36880-000

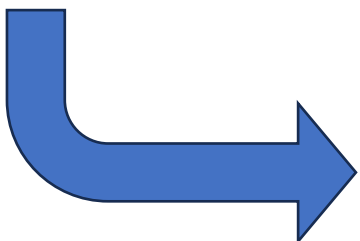


ID Cliente	Nome	Email	Telefone	ID Endereço
1	Paulo	paulo@gmail.com	3721-5555	3
2	João da Silva	joao@gmail.com	3722-2222	4

Modelagem do Banco de Dados

- Neste ponto, estamos definindo o endereço padrão de cobrança do cliente. Entretanto, ainda temos os endereços de entrega.

ID Endereço	Endereço	Número	Bairro	Cidade	Estado	CEP
1	Rua B	456	Centro	Muriaé	MG	36880-000
2	Rua C	789	Barra	Muriaé	MG	36880-001
3	Rua A	123	Barra	Muriaé	MG	36880-002
4	Rua B	23	Centro	Muriaé	MG	36880-000

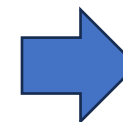


ID Cliente	Nome	Email	Telefone	ID Endereço
1	Paulo	paulo@gmail.com	3721-5555	3
2	João da Silva	joao@gmail.com	3722-2222	4

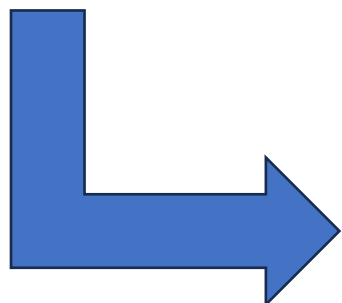
Modelagem do Banco de Dados

- Criamos uma tabela chamada Endereços de Entrega:

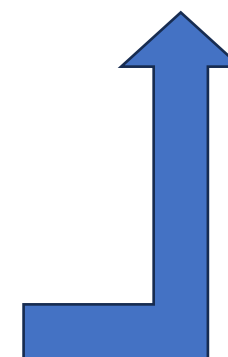
ID Endereço	Endereço	Número	Bairro	Cidade	Estado	CEP
1	Rua B	456	Centro	Muriaé	MG	36880-000
2	Rua C	789	Barra	Muriaé	MG	36880-001
3	Rua A	123	Barra	Muriaé	MG	36880-002
4	Rua B	23	Centro	Muriaé	MG	36880-000



ID Cliente	ID Endereço
1	1
1	2

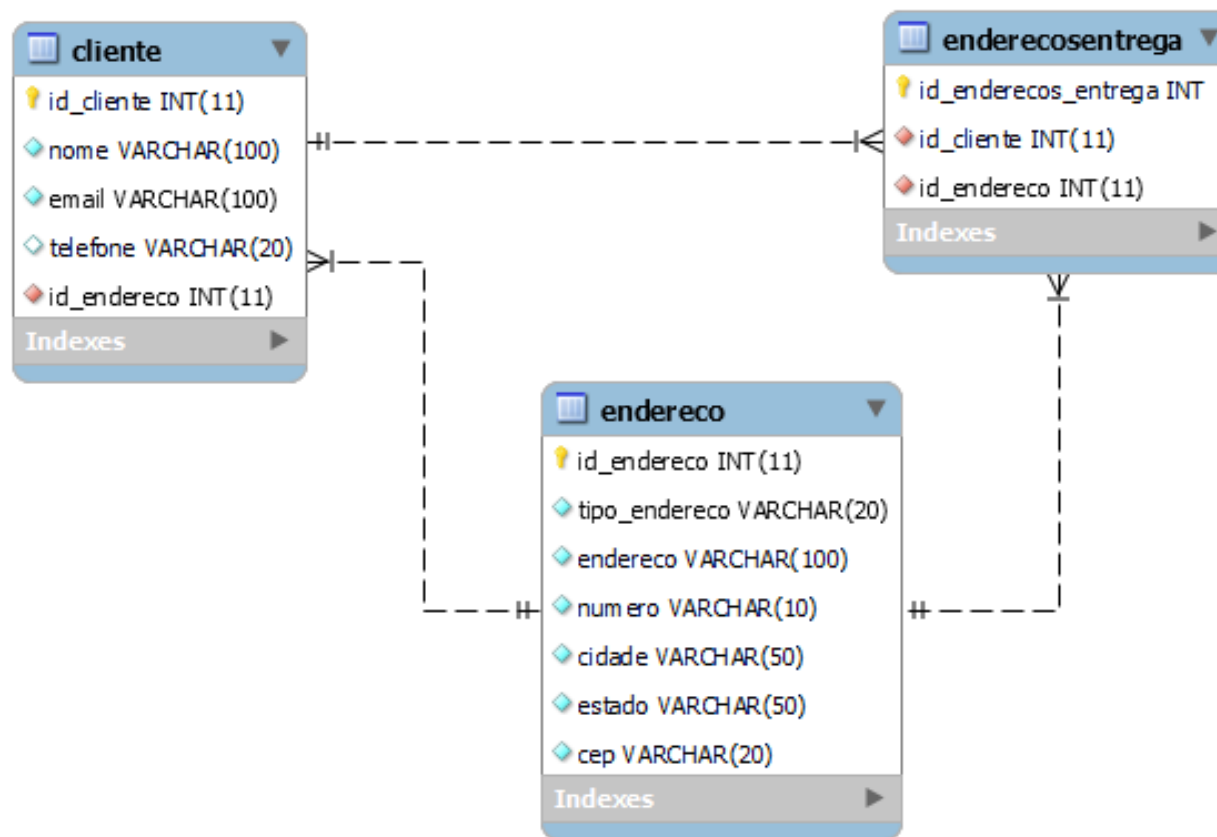


ID Cliente	Nome	Email	Telefone	ID Endereço
1	Paulo	paulo@gmail.com	3721-5555	3
2	João da Silva	joao@gmail.com	3722-2222	4



Modelagem do Banco de Dados

- Após normalizar na 3NF temos:



Podemos
normalizar o
banco de
dados ainda
mais?



Modelagem do Banco de Dados

- Normalização aplicada até a 3NF:
 - Cada cliente tem um endereço de cobrança único referenciado pela tabela de endereços.
 - Cada cliente pode ter vários endereços de entrega.
 - Cada endereço possui suas próprias informações atômicas, eliminando redundâncias.

Modelagem do Banco de Dados – 4NF

- Expandindo a normalização para 4NF: Visa garantir que não existam dependências multivaloradas, evitando a repetição de combinações de dados que não estão diretamente relacionadas. Ela é usada em sistemas onde múltiplos atributos podem depender de uma chave primária, mas são independentes entre si.

Modelagem do Banco de Dados – 4NF

- Como muitos endereços compartilham o mesmo bairro, cidade e estado, podemos criar novas entidades para armazenar essas informações.

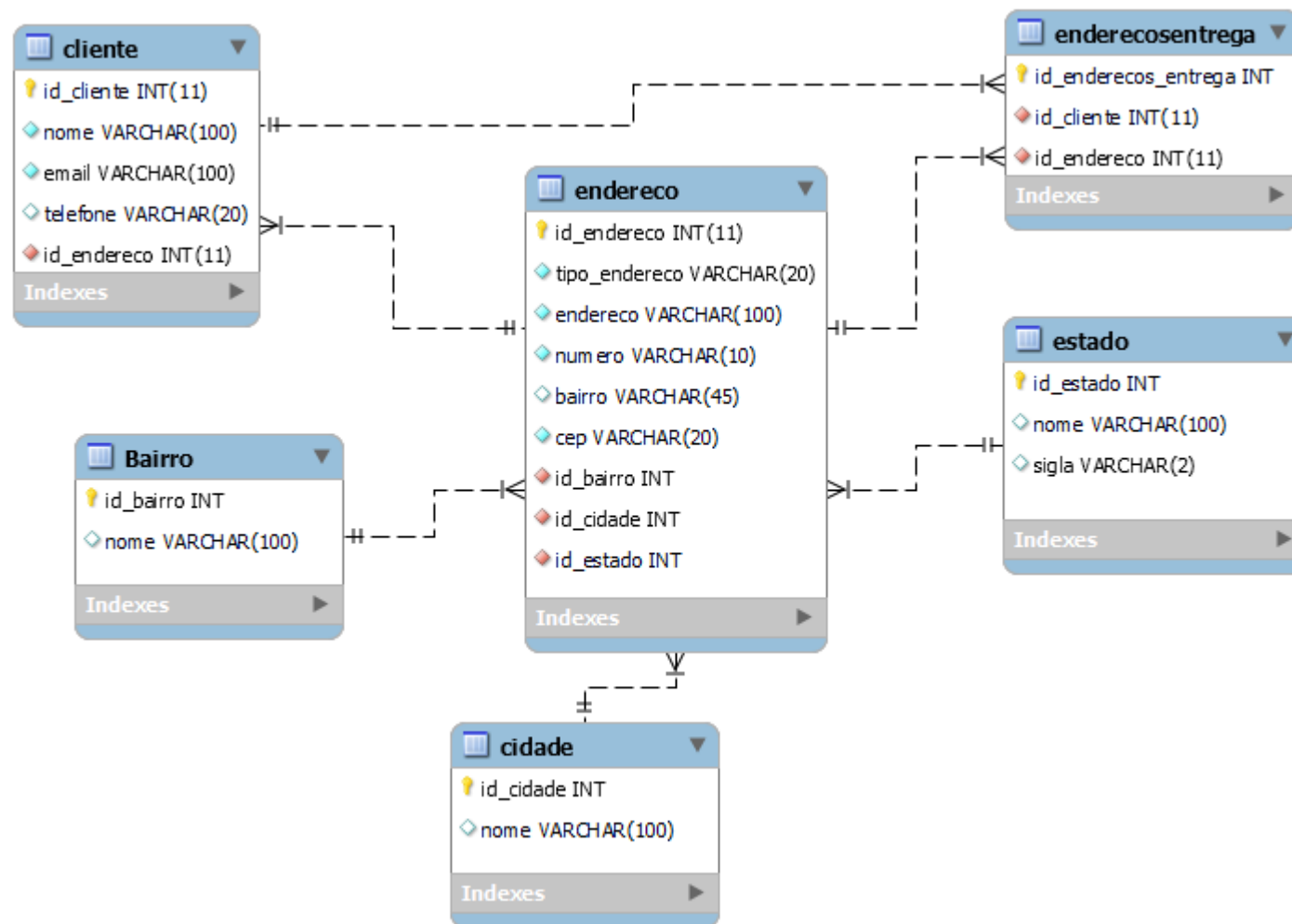
ID Bairro	Nome
1	Barra
2	Centro
3	João XIII

ID Cidade	Nome
1	Muriaé
2	Cataguases

ID Estado	Nome	Sigla
1	Minas Gerais	MG
2	Rio de Janeiro	RJ

Modelagem do Banco de Dados

- Modelo de entidade e relacionamento normalizado na 4NF



Vantagens e Desvantagens da 4NF

- **Vantagens:**

- **Eliminação de Redundância:** Informações repetitivas, como o nome do estado ou cidade, estão centralizadas em uma única tabela, evitando redundâncias.
- **Manutenção:** Mudanças no nome de um estado, cidade ou bairro precisam ser feitas apenas uma vez.
- **Escalabilidade:** Conforme o sistema cresce, o banco de dados permanece eficiente, pois cada entidade é armazenada uma única vez.

Vantagens e Desvantagens da 4NF

- **Desvantagens:**

- Complexidade: A quantidade de tabelas aumenta, tornando o modelo mais complexo e aumentando a quantidade de cadastros no sistema.
- Join Performance: O uso de múltiplos JOINS em consultas complexas pode diminuir a performance em sistemas muito grandes, exigindo otimizações ou uso de cache.

Quando a Normalização Adicional é Necessária?

Quando a normalização na 4NF é necessária?

- **Grande Volume de Dados Repetidos:**
 - Se houver muitos endereços com os mesmos valores para **estado**, **cidade** ou **bairro**, a normalização pode economizar espaço e reduzir a redundância. Isso evita a repetição de dados em cada linha de endereço.
- **Exemplo:** Um sistema com milhares de usuários no mesmo estado/cidade/bairro.

Quando a normalização na 4NF é necessária?

- **Manutenção Simplificada:**

- Se o nome de uma cidade, estado ou bairro mudar, basta alterar em um único lugar (na tabela de cidades ou estados). Sem a normalização, você teria que atualizar cada registro que contém esses valores.
- **Exemplo:** Mudança do nome de uma cidade como "Bandeirantes" para "Bandeirantes do Sul".

Quando a normalização na 4NF é necessária?

- **Relatórios Detalhados e Consultas Otimizadas:**
 - Em sistemas que exigem consultas frequentes ou relatórios detalhados, como "Quantos clientes temos em cada estado?", a normalização facilita essas operações e evita erros de consistência.
- **Exemplo:** Consultas que somam os clientes em cidades específicas.



Quando a Normalização do
Adicional NÃO É Necessária?



Quando a normalização na 4NF não é necessária?

- **Simplicidade do Sistema:**

- Se o sistema não tiver um volume enorme de registros e o foco for mais na **simplicidade** do desenvolvimento e da manutenção, deixar essas informações de estado, cidade e bairro na própria tabela de **endereço** pode ser suficiente.
- **Exemplo:** Pequenos negócios ou sistemas onde a escalabilidade não é uma preocupação imediata.

Quando a normalização na 4NF não é necessária?


- **Performance:**

- Sistemas de banco de dados muito normalizados podem sofrer perda de performance quando várias tabelas precisam ser unidas (JOINS) para recuperar informações simples.
- **Exemplo:** Um sistema com poucos registros de usuários, onde o impacto de armazenar os dados repetidos é pequeno.



Quando a normalização na 4NF não é necessária?

- **Custo de Manutenção:**

- Cada nova tabela exige manutenção e entendimento, aumentando a complexidade. Para um sistema pequeno, essa complexidade pode não valer o esforço adicional.
- **Exemplo:** Sistemas internos de uso limitado, onde a simplicidade é mais importante que a escalabilidade.



DEVO OU NÃO UTILIZAR A
NORMALIZAÇÃO MÁXIMA?



Conclusão

Não é sempre necessário normalizar até o nível de separar estados, cidades e bairros em tabelas para o nosso problema apresentado. A decisão deve levar em conta o **tamanho** do sistema, a **quantidade de dados repetidos** e a **complexidade** da manutenção.



Conclusão

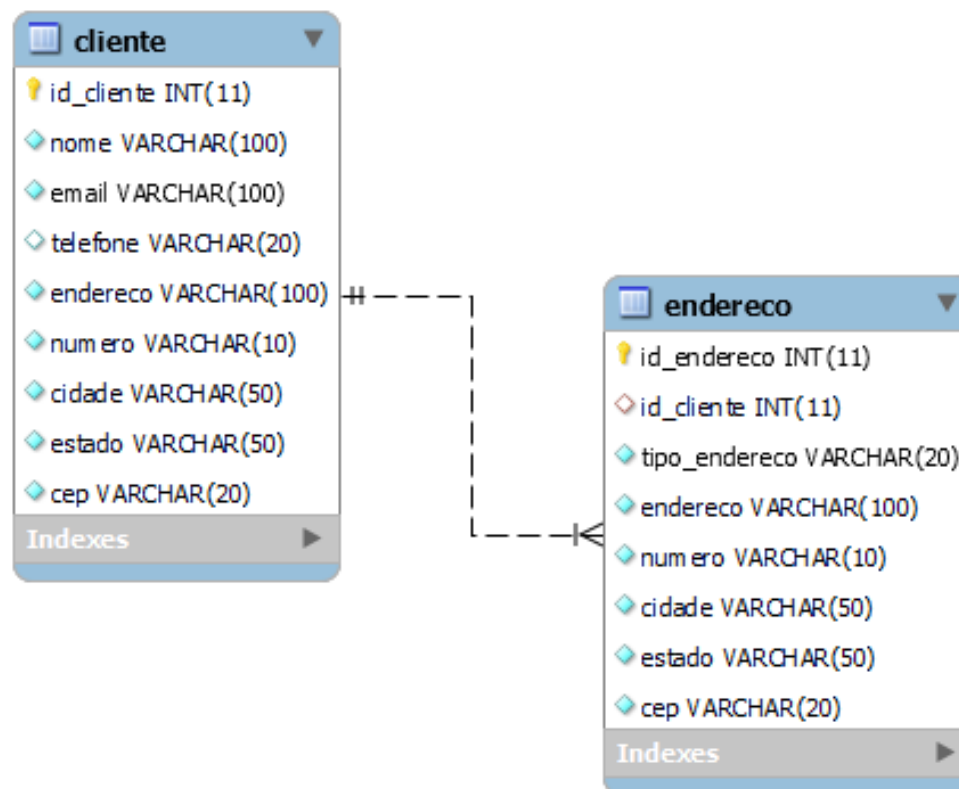
- **Normalização adicional é recomendada se:**
 - O sistema cresce rapidamente em termos de registros e endereços.
 - Há necessidade de atualizações frequentes em estados, cidades ou bairros.
 - É importante evitar a repetição de dados e manter a integridade referencial.

Conclusão

- **Normalização adicional pode ser desnecessária se:**
 - O volume de dados é gerenciável.
 - O sistema precisa ser simples e rápido de desenvolver.
 - O custo de manutenção das tabelas adicionais não compensa os benefícios.

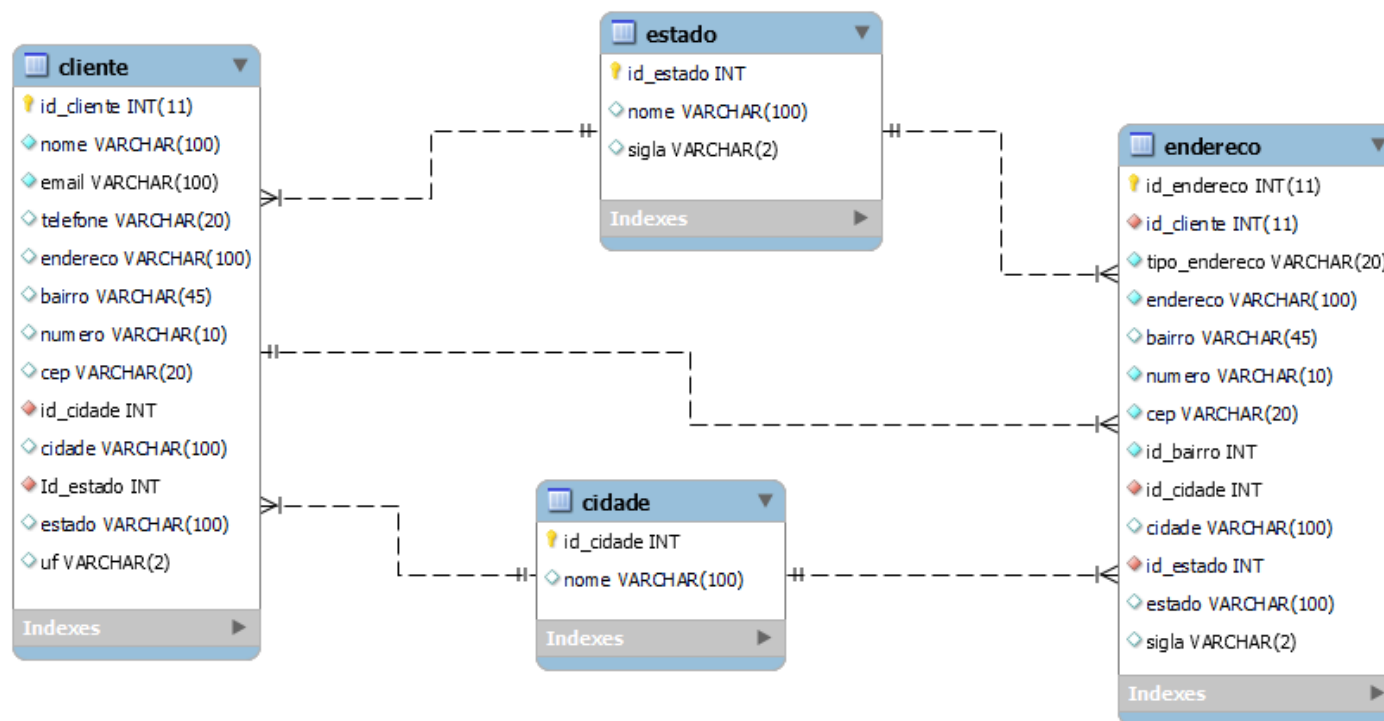
Conclusão

- Para o nosso problema, podemos apenas armazenar o cliente e o endereço de entrega com um modelo simples e eficiente.



Conclusão

- Entretanto, a medida que o sistema cresce pode ser necessário acrescentar as tabelas adicionais. Podemos também modelar algo como:



Conclusão

- Para evitar problemas do usuário digitar algum dado errado no endereço, podemos utilizar de APIs para buscar os dados a partir do CEP.

