



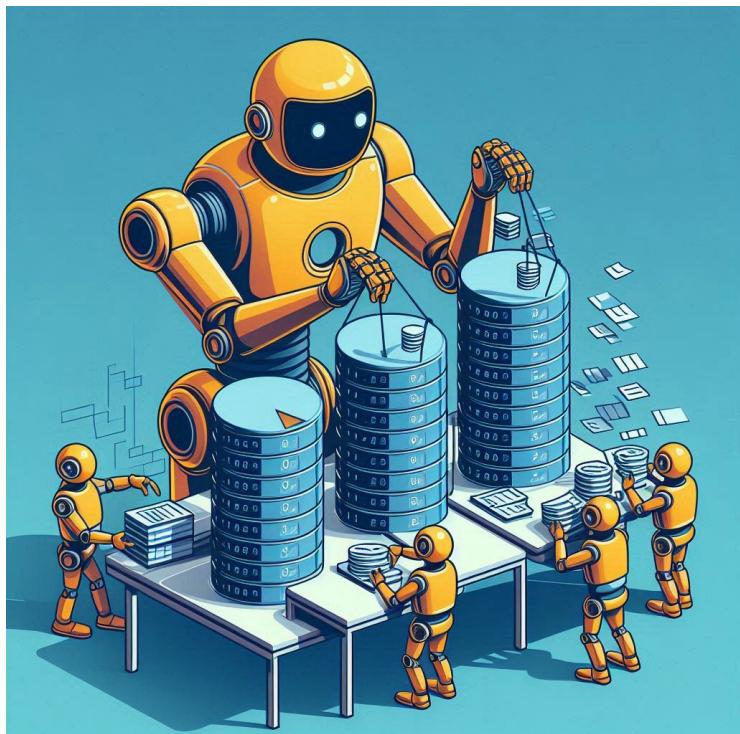
Capítulo 4 - Normalização de Dados

"Se depurar é o processo de remover bugs de software, então programar deve ser o processo de colocá-los."

Edsger Dijkstra

Neste capítulo, exploraremos os fundamentos essenciais da normalização de dados, um processo crucial para projetar bancos de dados relacionais eficientes e livres de redundâncias. A normalização visa organizar dados em tabelas para minimizar a redundância e a dependência, garantindo consistência e integridade dos dados. Abordaremos os seguintes tópicos:

- Definição de Normalização: Explicação sobre o que é normalização de dados e sua importância no projeto de bancos de dados relacionais.
- Objetivos da Normalização: Discutiremos os principais objetivos da normalização, como redução de redundância, melhoria na consistência e facilitação da manutenção dos dados.



Formas Normais

- Primeira Forma Normal (1FN): Descrição dos critérios necessários para que uma tabela esteja na 1FN, incluindo a eliminação de valores repetidos e a identificação de atributos atômicos.
- Segunda Forma Normal (2FN): Explicação sobre os requisitos para alcançar a 2FN, focando na eliminação de dependências parciais em relação à chave primária.
- Terceira Forma Normal (3FN): Detalhamento dos critérios para atingir a 3FN, que inclui a remoção de dependências transitivas, garantindo que cada atributo não dependa apenas da chave primária.

Dependência Funcional

- Definição de Dependência Funcional: Conceito de dependência funcional entre atributos e como ela influencia o processo de normalização.

- Aplicações da Dependência Funcional na Normalização: Exemplos práticos de como identificar dependências funcionais e aplicá-las para normalizar esquemas de banco de dados.

Neste capítulo, você aprenderá os princípios fundamentais da normalização de dados, desde os conceitos básicos até a aplicação prática das formas normais. Compreender esses conceitos é essencial para projetar bancos de dados eficientes e robustos, garantindo a integridade e o desempenho dos sistemas que os utilizam.

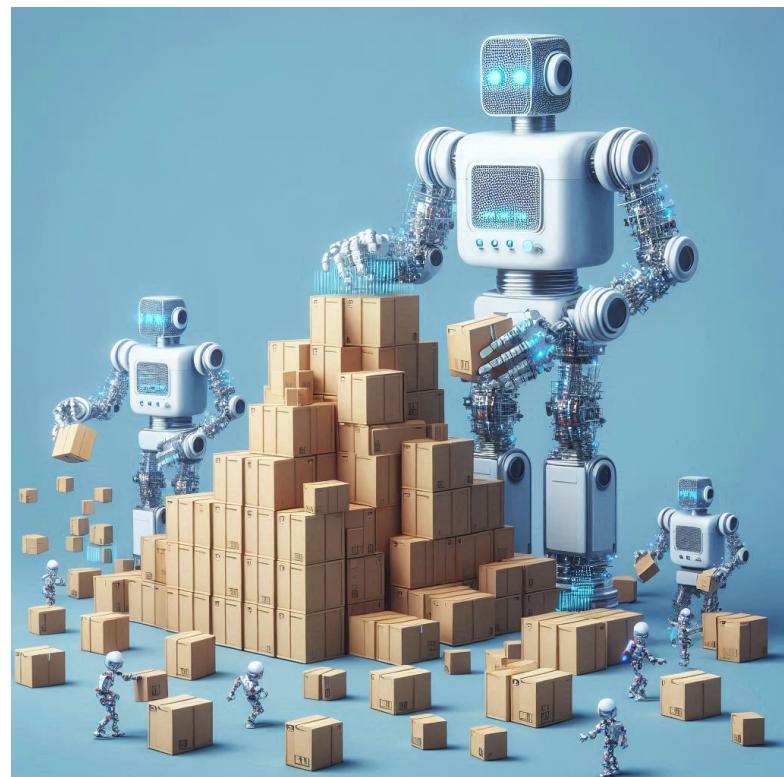
Seção 4.1: Conceitos Básicos de Normalização

A normalização de dados é um processo fundamental no projeto de bancos de dados relacionais. Ela se refere à técnica de organizar os dados em tabelas de forma estruturada e eficiente. O objetivo principal da normalização é eliminar a redundância e inconsistência nos dados, garantindo que cada pedaço de informação seja armazenado apenas uma vez e de maneira correta. Isso facilita a manutenção do banco de dados e reduz o risco de inconsistências ou erros.

A redundância de dados ocorre quando uma mesma informação é armazenada mais de uma vez no banco de dados. Isso pode acontecer em diferentes registros ou tabelas, levando a uma duplicação desnecessária de informações. Por exemplo, imagine uma tabela de clientes onde cada cliente tem suas informações, incluindo endereço. Se um cliente realizar várias compras, e seu endereço estiver armazenado em cada registro de compra, isso é redundância.

Exemplo:

Suponha que temos a seguinte tabela de Clientes:



ID	Nome	Endereço
1	João	Rua A, São Paulo
2	Maria	Av. B, Rio de Janeiro
3	Pedro	Rua C, São Paulo

E uma tabela de Pedidos:

ID Pedido	ID Cliente	Data	Total	Endereço
1	1	2024-06-15	100	Rua A, São Paulo
2	1	2024-06-16	150	Rua A, São Paulo
3	2	2024-06-15	200	Av. B, Rio de Janeiro
4	1	2024-06-17	120	Rua A, São Paulo

Note que o endereço do cliente está duplicado para o João em cada pedido que ele fez. Isso é redundância de dados.

A inconsistência de dados ocorre quando informações contraditórias ou conflitantes são armazenadas no banco de dados. Isso pode acontecer devido à falta de controle sobre as atualizações ou alterações nos dados. Por exemplo, se um cliente altera seu endereço em um pedido, mas essa atualização não é refletida em todos os lugares onde o endereço é armazenado, podemos ter inconsistências.

Exemplo:

Continuando com os exemplos acima, se o cliente João alterar seu endereço para "Rua D, São Paulo" apenas no último pedido, mas o endereço anterior ainda está registrado nos pedidos anteriores, teremos uma inconsistência de dados.

ID Pedido	ID Cliente	Data	Total	Endereço
1	1	2024-06-15	100	Rua A, São Paulo
2	1	2024-06-16	150	Rua A, São Paulo
3	2	2024-06-15	200	Av. B, Rio de Janeiro

4	1	2024-06-17	120	Rua D, São Paulo
---	---	------------	-----	------------------

Neste caso, o endereço de João não está consistente em todos os pedidos.

Para ilustrar melhor esses conceitos, vamos utilizar mais um exemplo. Imagine que você está criando um banco de dados para uma escola. Você precisa armazenar informações sobre os alunos, como nome, idade, turma e notas. Em vez de ter uma única tabela enorme com todas essas informações misturadas, a normalização sugere dividir essas informações em tabelas separadas com relações claras entre elas. Por exemplo:

Tabela Alunos:

ID Aluno	Nome	Idade	Turma
1	João	15	A
2	Maria	14	B
3	Pedro	16	A

Tabela Notas:

ID Nota	ID Aluno	Disciplina	Nota
1	1	Matemática	8.5
2	1	História	7.0
3	2	Matemática	9.0
4	2	História	8.5
5	3	Matemática	7.5
6	3	História	6.5

Neste exemplo, temos uma tabela separada para os alunos e outra para as notas. Cada tabela tem uma chave primária única (ID Aluno e ID Nota) para identificar cada registro de forma exclusiva.

Para ilustrar um exemplo negativo de como não estruturar um banco de dados, vamos consolidar todas as informações sobre alunos, turmas e notas em uma única tabela sem normalização:

Tabela Não Normalizada:

ID	Nome	Idade	Turma	Disciplina	Nota
1	João	15	A	Matemática	8.5
2	Maria	14	B	História	7.0
3	Pedro	16	A	Matemática	9.0
4	João	15	A	História	8.5
5	Pedro	16	A	Matemática	7.5
6	Maria	14	B	História	6.5

Nesse caso, se um aluno mudar de turma, com as tabelas divididas você só precisa atualizar a informação na tabela Alunos, e não em todas as ocorrências onde o aluno aparece, como seria se tudo estivesse em uma única tabela. A normalização resolve esses problemas ao dividir as informações em tabelas separadas, como Alunos e Notas, com relacionamentos claros entre elas. Isso não apenas economiza espaço de armazenamento, mas também facilita a manutenção dos dados, reduzindo o risco de inconsistências e melhorando o desempenho das consultas no banco de dados.

A normalização, portanto, não apenas organiza os dados de maneira mais eficiente, mas também ajuda a garantir que o banco de dados seja mais fácil de manter e menos propenso a erros. Esses princípios são fundamentais para qualquer projeto de banco de dados, independentemente da sua complexidade ou escala.

Objetivos da Normalização

1. Redução de Redundância: Evitar a repetição desnecessária dos dados. Com a normalização, as informações são armazenadas apenas uma vez, economizando espaço e melhorando a eficiência.
2. Melhoria na Consistência: Garantir que os dados sejam precisos e consistentes em todo o banco de dados. Isso ajuda a evitar discrepâncias ou informações contraditórias.
3. Facilitação da Manutenção dos Dados: Simplificar as operações de atualização e manutenção do banco de dados. Mudanças em uma parte do banco de dados não precisam ser replicadas em várias tabelas, tornando o sistema mais fácil de gerenciar e menos propenso a erros.

Problemas com a Não Normalização:

- Redundância de Dados: Os dados de alunos (nome, idade, turma) são repetidos para cada disciplina em que o aluno está matriculado. Por exemplo, João e Pedro aparecem várias vezes na tabela, resultando em desperdício de espaço de armazenamento.
- Inconsistência dos Dados: Se um aluno alterar sua idade, turma ou nome, todas as ocorrências desses dados na tabela terão que ser atualizadas individualmente. Isso aumenta a probabilidade de inconsistências nos dados se alguma atualização for esquecida.
- Dificuldade de Manutenção: Adicionar novas disciplinas ou alunos implica em inserir múltiplas linhas na tabela, o que pode ser propenso a erros e dificulta a manutenção do banco de dados.
- Desempenho Reduzido: Consultas que precisam agrupar ou filtrar informações específicas (como todas as notas de um aluno) podem ser mais lentas devido à estrutura ampla e redundante da tabela.

Vamos utilizar outro exemplo para ilustrar ainda mais esses problemas. Imagine que estamos projetando um banco de dados para uma loja de roupas que precisa gerenciar informações sobre clientes, produtos e pedidos.

Para exemplificar como não fazer um projeto lógico de banco de dados, vamos consolidar as tabelas de Clientes, Produtos, Pedidos e Itens do Pedido em tabelas maiores que poderiam ter valores repetidos. Vamos criar uma estrutura simplificada, não normalizada, para ilustrar:

Tabela Unificada (Não Normalizada):

ID Cliente	Nome Cliente	Telefone	Email	ID Produto	Nome Produto	Categoria	Preço	ID Pedido	Data Pedido	ID Item	Quantidade
1	João Silv a	(11) 9876 5-43 21	joao@email.com	1	Camisa Branca	Roupa s	R\$ 39, 90	1	2024-06-15	1	2
1	João Silv a	(11) 9876 5-43 21	joao@email.com	2	Calça Jeans	Roupa s	R\$ 89, 90	1	2024-06-15	2	1

2	Maria Souza	(11) 9987 6-5432	maria@email.com	3	Tênis Espor tivo	Calça dos	R\$ 129, 90	2	202 4-0 6-16	3	1
3	Pedro Santos	(11) 9876 5-1234	pedro@email.com	1	Cami seta Branc a	Roupa s	R\$ 39, 90	3	202 4-0 6-17	4	3
3	Pedro Santos	(11) 9876 5-1234	pedro@email.com	2	Calça Jeans	Roupa s	R\$ 89, 90	3	202 4-0 6-17	5	1

Justificativa para Não Normalização:

- Redundância de Dados: Cada vez que um cliente faz um pedido, suas informações pessoais (nome, telefone, e-mail) são repetidas para cada item do pedido. Isso aumenta o armazenamento necessário e dificulta a atualização de informações pessoais do cliente sem alterar múltiplos registros.
- Inconsistência dos Dados: Se um cliente atualizar seu telefone ou e-mail, seria necessário atualizar todas as ocorrências na tabela unificada, o que pode resultar em erros de inconsistência se alguma atualização for esquecida.
- Dificuldade de Manutenção: Adicionar ou remover produtos de um pedido implica em modificar múltiplos registros, aumentando a complexidade da operação e a chance de erros.
- Desempenho Reduzido: Consultas que precisam agrupar ou filtrar informações específicas (como todos os pedidos de um cliente) podem ser mais lentas devido à estrutura ampla e redundante da tabela.

Esta abordagem não normalizada resultaria em um banco de dados menos eficiente, mais difícil de manter e com maior risco de inconsistências nos dados à medida que a loja de roupas cresce e aumenta seu volume de vendas.

A normalização nos ajuda a organizar esses dados da seguinte maneira:

Tabela Clientes:

ID Cliente	Nome	Endereço	Telefone
------------	------	----------	----------

1	Maria Silva	Rua A, 123	(11) 98765-4321
2	João Santos	Av. Principal, 456	(11) 99999-8888
3	Ana Costa	Travessa B, 789	(11) 87654-3210

Tabela Produtos:

ID Produto	Nome	Categoria	Preço
1	Camiseta Branca	Roupas	R\$ 39,90
2	Calça Jeans	Roupas	R\$ 89,90
3	Tênis Esportivo	Calçados	R\$ 129,90

Tabela Pedidos:

ID Pedido	ID Cliente	Data Pedido	Total
1	1	2024-06-01	R\$ 129,80
2	2	2024-06-02	R\$ 219,80
3	3	2024-06-03	R\$ 169,90

Tabela Itens do Pedido:

ID Item	ID Pedido	ID Produto	Quantidade
1	1	1	2
2	1	3	1
3	2	1	3
4	3	3	1

Neste exemplo, temos quatro tabelas separadas: Clientes, Produtos, Pedidos e Itens do Pedido. Cada tabela possui uma chave primária única (ID Cliente, ID Produto, ID Pedido, ID Item) para identificar cada registro de forma exclusiva. Isso permite que as informações sejam organizadas de maneira eficiente, facilitando a gestão de clientes, produtos e pedidos na loja de roupas.

Seção 4.2: Primeira Forma Normal (1FN)

A Primeira Forma Normal (1FN) é um conceito fundamental no projeto de bancos de dados relacionais, que estabelece os critérios básicos para estruturar tabelas de forma organizada e

sem redundâncias. Ela se originou no contexto dos estudos sobre modelagem de dados e foi formalizada como parte da teoria de normalização por Edgar F. Codd nos anos 1970.



A 1FN surgiu da necessidade de organizar os dados de maneira que cada atributo (ou campo) de uma tabela contenha apenas valores atômicos, ou seja, valores indivisíveis e simples. Antes da normalização, era comum armazenar múltiplos valores em uma única célula da tabela, o que dificultava a consulta e manipulação dos dados.

Para que uma tabela esteja na Primeira Forma Normal, ela deve atender aos seguintes critérios:

1. Valores Atômicos: Cada célula da tabela deve conter apenas um valor simples e indivisível. Isso significa que um campo não deve conter múltiplos valores, listas ou estruturas complexas.
2. Identificação da Chave Primária: Cada tabela deve ter uma chave primária que identifique exclusivamente cada registro. A chave primária não deve repetir-se em nenhum registro da tabela.

A Primeira Forma Normal é importante por diversos motivos:

- Redução de Redundância: Ao organizar os dados em valores atômicos, evita-se a repetição desnecessária de informações na tabela, o que economiza espaço de armazenamento e melhora a eficiência.
- Melhoria na Consistência: Garante que os dados sejam precisos e consistentes, uma vez que cada valor é armazenado de forma única e clara.
- Facilitação da Manutenção: Simplifica as operações de atualização, inserção e exclusão de dados. Mudanças em um campo específico afetam apenas aquele campo, sem afetar outras partes da tabela.

Vamos utilizar um Exemplo Negativo (Não Normalizado - Escola):

ID	Nome	Idade	Turma	Disciplina	Nota 1	Nota 2
1	João	15	A	Matemática, História	8.5	7.0
2	Maria	14	B	Matemática, História	9.0	8.5
3	Pedro	16	A	Matemática, História	7.5	6.5

- Problemas: A coluna Disciplina contém múltiplos valores, violando a regra de valores atômicos. Além disso, as notas estão repetidas em colunas separadas, o que pode levar a inconsistências e dificuldades de atualização.

Normalização para 1FN (Escola):

ID	Nome	Idade	Turma	Disciplina	Nota
1	João	15	A	Matemática	8.5
1	João	15	A	História	7.0
2	Maria	14	B	Matemática	9.0
2	Maria	14	B	História	8.5
3	Pedro	16	A	Matemática	7.5
3	Pedro	16	A	História	6.5

- Melhoria: Agora cada linha contém apenas um valor para Disciplina e Nota, eliminando a redundância e respeitando a atomicidade dos dados. Lembrando que essa tabela ainda não está 100% normalizada, e sim normalizada na forma N1.

Vamos ver outro Exemplo Negativo (Não Normalizado - Loja de Roupas):

ID	Cliente	Produto	Categoria	Preço
1	João	Camiseta, Calça	Vestuário	50, 80
2	Maria	Tênis, Camiseta	Calçados	120, 60
3	Pedro	Casco, Calça	Vestuário	150, 80

- Problemas: A coluna Produto contém múltiplos valores, violando a regra de valores atômicos. Além disso, os preços estão repetidos em colunas separadas para cada produto.

Normalização para 1FN (Loja de Roupas):

ID	Cliente	Produto	Categoria	Preço
1	João	Camiseta	Vestuário	50
1	João	Calça	Vestuário	80
2	Maria	Tênis	Calçados	120
2	Maria	Camiseta	Vestuário	60
3	Pedro	Casaco	Vestuário	150
3	Pedro	Calça	Vestuário	80

- Melhoria: Cada linha agora representa apenas um produto com seu respectivo preço, eliminando a repetição de dados na coluna Produto e respeitando a atomicidade dos valores. Porém, isso ainda pode ser melhorado, como veremos a seguir.

Seção 4.3: Segunda Forma Normal (2FN)

A Segunda Forma Normal (2FN) é um conceito avançado na teoria de normalização de bancos de dados relacionais, desenvolvido por Edgar F. Codd nos anos 1970. Ela surgiu como uma extensão da Primeira Forma Normal (1FN), visando eliminar dependências parciais em relação à chave primária, proporcionando uma estrutura de dados mais organizada e eficiente.

A 2FN teve origem na necessidade de eliminar redundâncias e dependências parciais que poderiam surgir em tabelas que já estavam na 1FN. Codd percebeu que, mesmo após a normalização para a 1FN, ainda era possível que alguns campos dependessem apenas de uma parte da chave primária, o que poderia comprometer a integridade e consistência dos dados.

Para que uma tabela esteja na Segunda Forma Normal, ela deve atender aos seguintes critérios:

1. Estar na 1FN: A tabela já deve estar na Primeira Forma Normal, ou seja, todos os campos devem conter valores atômicos e cada registro deve ser único e identificado por uma chave primária.
2. Eliminar Dependências Parciais: Todos os atributos não chave (ou seja, que não fazem parte da chave primária) devem depender completamente da chave primária. Isso

significa que nenhum campo deve depender de apenas uma parte da chave primária, mas sim da chave primária como um todo.

Quando dizemos que um atributo depende parcialmente da chave primária, significa que parte do seu valor pode ser determinada por apenas uma parte da chave primária, em vez de depender dela como um todo. Isso pode levar a problemas se não for corrigido, pois pode resultar em dados inconsistentes ou difíceis de atualizar. Imagine uma tabela de Pedidos com os seguintes atributos:

- ID Pedido (Chave Primária)
- ID Cliente (Parte da Chave Primária)
- Nome Cliente (Depende do ID Cliente)
- Endereço Cliente (Depende do ID Cliente)
- Data do Pedido (Depende do ID Pedido)
- Total do Pedido (Depende do ID Pedido)

Na situação acima, a tabela de Pedidos pode estar em conformidade com a Primeira Forma Normal (1FN), onde todos os atributos são atômicos e não há repetição desnecessária de dados. No entanto, para estar em conformidade com a Segunda Forma Normal (2FN), precisamos garantir que nenhum atributo dependa parcialmente da chave primária.

Por exemplo, se o atributo "Nome Cliente" depender apenas do "ID Cliente", e não do "ID Pedido" junto com o "ID Cliente", isso seria uma dependência parcial. Na 2FN, o "Nome Cliente" deve depender do "ID Cliente" e "ID Pedido" juntos, não apenas do "ID Cliente" isoladamente. Isso garante que todas as informações do pedido estejam corretamente associadas ao pedido específico, evitando inconsistências se o mesmo cliente fizer pedidos diferentes ao longo do tempo.

Assim, a 2FN busca eliminar dependências parciais para garantir que cada atributo dependa da chave primária como um todo, mantendo a integridade e consistência dos dados no banco de dados relacional.

A Segunda Forma Normal é importante por diversos motivos:

- Eliminação de Redundâncias: Ao eliminar dependências parciais, reduz-se a redundância de dados no banco, melhorando a eficiência de armazenamento e consulta.

- Melhoria na Integridade dos Dados: Garante que os dados sejam consistentes e precisos, uma vez que cada campo está corretamente associado à chave primária completa.
- Facilitação da Manutenção: Simplifica operações de atualização, inserção e exclusão de dados, pois mudanças na chave primária refletem-se corretamente em todos os campos associados.

Exemplo (Escola):

Para exemplificar uma tabela que está na Primeira Forma Normal (1FN) mas não na Segunda Forma Normal (2FN), vamos considerar uma tabela fictícia para armazenar informações sobre alunos e suas disciplinas matriculadas em uma escola.

Tabela Alunos_Displiplinas:

ID Aluno	Nome Aluno	Disciplina	Professor	Nota
1	João	Matemática	Prof. Silva	8.5
1	João	História	Prof. Santos	7.0
2	Maria	Matemática	Prof. Silva	9.0
2	Maria	História	Prof. Santos	8.5
3	Pedro	Matemática	Prof. Silva	7.5
3	Pedro	História	Prof. Santos	6.5

Explicação:

Nesta tabela, temos as seguintes colunas:

- ID Aluno: Identificador único do aluno.
- Nome Aluno: Nome do aluno.
- Disciplina: Disciplinas em que os alunos estão matriculados.
- Professor: Nome do professor da disciplina.
- Nota: Nota obtida pelo aluno na disciplina.

Primeira Forma Normal (1FN):

- Todos os valores são atômicos, ou seja, cada célula contém apenas um único valor.

- Não há repetição de grupos de colunas.

A tabela acima está na 1FN porque atende a esses critérios. No entanto, ela não está na Segunda Forma Normal (2FN) devido às dependências parciais em relação à chave primária.

Problema de Dependência Parcial (2FN):

Na 2FN, nenhum atributo não chave deve depender de apenas uma parte da chave primária. No exemplo dado, o atributo Professor depende apenas do ID Aluno e Disciplina juntos, o que é uma dependência parcial. Isso significa que o mesmo professor pode ser associado a diferentes combinações de aluno e disciplina, o que não é ideal.

Para corrigir isso e tornar a tabela compatível com a 2FN, precisamos dividir a tabela em duas, de modo que cada tabela tenha uma chave primária única e as dependências sejam mais diretas e completas.

Exemplo de Divisão:

Tabela Alunos:

ID Aluno	Nome Aluno
1	João
2	Maria
3	Pedro

Tabela Disciplinas_Alunos:

ID Aluno	Disciplina	Professor	Nota
1	Matemática	Prof. Silva	8.5
1	História	Prof. Santos	7.0
2	Matemática	Prof. Silva	9.0
2	História	Prof. Santos	8.5
3	Matemática	Prof. Silva	7.5
3	História	Prof. Santos	6.5

Explicação da Solução:

Agora, na nova estrutura:

- Tabela Alunos: Contém apenas informações sobre os alunos, com o ID Aluno como chave primária.
- Tabela Disciplinas_Alunos: Contém as informações sobre as disciplinas em que os alunos estão matriculados, com ID Aluno e Disciplina juntos como chave primária. O Professor e a Nota dependem diretamente dessa chave primária composta, eliminando a dependência parcial.

Dessa forma, dividindo a tabela original em duas, garantimos que cada tabela esteja na 1FN e na 2FN, seguindo os princípios de normalização para melhorar a estrutura e integridade dos dados no banco de dados relacional.

Seção 4.4: Terceira Forma Normal (3FN)

A Terceira Forma Normal (3FN) é um conceito avançado na teoria de normalização de bancos de dados relacionais, também desenvolvido por Edgar F. Codd nos anos 1970. Ela surgiu como uma evolução da Segunda Forma Normal (2FN), focando na eliminação de dependências transitivas entre os atributos não chave, garantindo assim uma estrutura de dados ainda mais organizada e eficiente.

A 3FN foi proposta para resolver uma limitação da 2FN, que ainda permitia dependências transitivas entre os atributos não chave. Codd percebeu que, mesmo após a normalização para a 2FN, era possível que um atributo não chave dependesse de outro atributo não chave, criando uma cadeia de dependências que poderia levar a redundâncias e inconsistências nos dados.

Critérios da 3FN:

Para que uma tabela esteja na Terceira Forma Normal, ela deve atender aos seguintes critérios:

1. Estar na 2FN: A tabela já deve estar na Segunda Forma Normal, ou seja, deve eliminar dependências parciais em relação à chave primária.
2. Eliminar Dependências Transitivas: Todos os atributos não chave devem depender exclusivamente da chave primária e não de outros atributos não chave. Isso significa que cada atributo não chave deve ser diretamente dependente da chave primária e não de outros atributos não chave.

Imagine que você tem um conjunto de atributos em uma tabela de banco de dados. A dependência transitiva ocorre quando um atributo não chave depende de outro atributo que, por sua vez, depende da chave primária. Isso cria uma cadeia de dependências indiretas que pode

levar a redundâncias e problemas de integridade nos dados. Por exemplo, suponha que temos uma tabela com informações sobre estudantes em uma escola. Cada aluno tem um número de identificação único (ID Aluno) como chave primária. Além disso, a tabela possui atributos como Nome do Aluno, Turma e Endereço.

Se adicionarmos o atributo Cidade ao lado do atributo Endereço, a dependência do atributo Cidade passa a ser transitiva. Isso ocorre porque o atributo Cidade depende diretamente do atributo Endereço, e o Endereço, por sua vez, depende do ID Aluno.

A Terceira Forma Normal é importante por diversos motivos:

- Redução de Redundâncias e Anomalias: Ao eliminar dependências transitivas, reduz-se a redundância de dados e evita-se anomalias de atualização, inserção e exclusão.
- Melhoria na Estruturação dos Dados: Proporciona uma estrutura de dados mais clara e organizada, facilitando a compreensão e manutenção do banco de dados.
- Consistência e Integridade: Garante que os dados sejam consistentes e que todas as informações sejam precisas e corretas, melhorando a qualidade dos dados armazenados.

Para exemplificar a situação onde a tabela está na Primeira e Segunda Forma Normal (1FN e 2FN), mas não está na Terceira Forma Normal (3FN), vou descrever a estrutura das tabelas Alunos e Disciplinas_Alunos.

Tabela Alunos:

- ID Aluno (Chave Primária)
- Nome Aluno

Exemplo de dados:

ID Aluno	Nome Aluno
1	João
2	Maria
3	Pedro

Tabela Disciplinas_Alunos:

- ID Aluno (Chave Estrangeira referenciando ID Aluno na tabela Alunos)
- Disciplina

- Professor
- Nota

Exemplo de dados:

ID Aluno	Disciplina	Professor	Nota
1	Matemática	Prof. Silva	8.5
1	História	Prof. Costa	7.0
2	Matemática	Prof. Silva	9.0
2	História	Prof. Costa	8.5
3	Matemática	Prof. Silva	7.5
3	História	Prof. Costa	6.5

Análise para a 3FN:

Na estrutura atual, a tabela Disciplinas_Alunos não está na Terceira Forma Normal (3FN) devido à presença de dependências transitivas. Vamos identificar o problema:

1. Dependência Transitiva:

- O atributo Professor depende funcionalmente da combinação de ID Aluno e Disciplina.
- Isso significa que o valor do Professor não está diretamente dependente da chave primária da tabela, que é o ID Aluno.

2. Solução para 3FN:

Para colocar a tabela Disciplinas_Alunos na Terceira Forma Normal, devemos remover essa dependência transitiva. Uma abordagem seria criar uma nova tabela, por exemplo, chamada Disciplinas, que contenha informações sobre as disciplinas e seus respectivos professores. Assim, a tabela Disciplinas_Alunos faria referência apenas à chave estrangeira da tabela Disciplinas, e não teria mais essa dependência transitiva.

Exemplo de Estrutura Proposta:

Tabela Disciplinas:

- ID Disciplina (Chave Primária)
- Disciplina
- Professor

Exemplo de dados:

ID Disciplina	Disciplina	Professor
1	Matemática	Prof. Silva
2	História	Prof. Costa

Tabela Disciplinas_Alunos (após a normalização):

- ID Aluno (Chave Estrangeira referenciando ID Aluno na tabela Alunos)
- ID Disciplina (Chave Estrangeira referenciando ID Disciplina na tabela Disciplinas)
- Nota

Exemplo de dados:

ID Aluno	ID Disciplina	Nota
1	1	8.5
1	2	7.0
2	1	9.0
2	2	8.5
3	1	7.5
3	2	6.5

Ao separar a informação sobre as disciplinas e seus professores em uma tabela separada (Disciplinas), eliminamos a dependência transitiva que existia na tabela Disciplinas_Alunos. Agora, cada atributo não chave (como Professor) depende diretamente da chave primária da tabela à qual pertence (ID Disciplina), tornando a estrutura do banco de dados mais normalizada e compatível com a Terceira Forma Normal (3FN).

Para elucidar ainda mais esses conceitos. Vamos resolver mais um problema de Normalização de Dados até a Terceira Forma Normal (3FN) utilizando um Exemplo de uma Pequena Empresa

Para ilustrar esse processo de normalização até a Terceira Forma Normal (3FN), vamos considerar um exemplo simples de uma pequena empresa que armazena informações dos funcionários em uma única tabela sem normalização. Vamos seguir o processo passo a passo para alcançar cada forma normal.

Passo 1: Primeira Forma Normal (1FN)

Suponha que temos uma tabela única da empresa com as seguintes informações misturadas:

ID	Nome Funcionário	Departamento	Localização Departamento	Salário	Cargo
1	João	Vendas	São Paulo	5000	Vendedor
2	Maria	Financeiro	Rio de Janeiro	6000	Analista
3	Pedro	Vendas	São Paulo	4500	Vendedor

Esta tabela está em uma forma não normalizada, pois mistura atributos de diferentes entidades (funcionários, departamentos e detalhes de departamento) em uma única estrutura. Para aplicar a 1FN, precisamos dividir essa tabela em estruturas mais simples, garantindo que cada coluna contenha apenas valores atômicos e que não haja repetições de grupos de valores.

Tabela Funcionários:

ID	Nome Funcionário	Salário	Cargo
1	João	5000	Vendedor
2	Maria	6000	Analista
3	Pedro	4500	Vendedor

Tabela Departamentos:

ID	Departamento	Localização
1	Vendas	São Paulo
2	Financeiro	Rio de Janeiro

Após a primeira normalização, dividimos a tabela original em duas: uma para Funcionários e outra para Departamentos. Ainda não estamos na 2FN, pois a tabela de Funcionários ainda possui dependências parciais em relação à chave primária (ID).

Passo 2: Segunda Forma Normal (2FN)

Para alcançar a 2FN, precisamos garantir que todos os atributos não chave dependam completamente da chave primária da tabela. Neste caso, precisamos remover a dependência parcial do departamento em relação ao ID do funcionário.

Tabela Funcionários:

ID	Nome Funcionário	Salário	Cargo	ID Departamento
1	João	5000	Vendedor	1
2	Maria	6000	Analista	2
3	Pedro	4500	Vendedor	1

Tabela Departamentos:

ID	Departamento	Localização
1	Vendas	São Paulo
2	Financeiro	Rio de Janeiro

Agora, a tabela Funcionários possui uma chave estrangeira (ID Departamento) que referencia a tabela Departamentos. Removemos a dependência parcial do Departamento em relação ao ID do Funcionário.

Passo 3: Terceira Forma Normal (3FN)

Para alcançar a 3FN, devemos garantir que não haja dependências transitivas entre os atributos não chave. Vamos analisar se há dependências transitivas na tabela Funcionários:

Para garantir que a estrutura das tabelas esteja na Terceira Forma Normal (3FN), devemos analisar se há dependências transitivas entre os atributos não chave. Vamos revisar os atributos da tabela Funcionários:

Tabela Funcionários:

ID	Nome Funcionário	Salário	Cargo	ID Departamento
1	João	5000	Vendedor	1
2	Maria	6000	Analista	2

3	Pedro	4500	Vendedor	1
---	-------	------	----------	---

Nesta tabela, identificamos os seguintes atributos não chave:

- Salário
- Cargo

Para aplicar a Terceira Forma Normal (3FN), devemos garantir que não haja dependências transitivas entre esses atributos não chave e a chave primária (ID).

Análise de Dependências Transitivas:

1. Salário depende diretamente do ID (chave primária), pois cada funcionário tem um salário único associado diretamente ao seu ID.
2. Cargo também depende diretamente do ID, pois cada funcionário tem um cargo específico associado diretamente ao seu ID.

Mesmo estando na norma 3FN podemos melhorar ainda mais essa relação. Poderíamos considerar a criação de uma nova tabela para o Cargo, se cada cargo tiver atributos adicionais que não sejam diretamente dependentes da tabela Funcionários. Isso ajudaria a evitar redundância e a garantir a Terceira Forma Normal (3FN).

Exemplo de Tabela Cargo:

ID Cargo	Nome Cargo	Descrição
1	Vendedor	Vende produtos aos clientes
2	Analista	Analisa dados financeiros

Na tabela Funcionários, substituiríamos o atributo Cargo pelo ID Cargo como chave estrangeira referenciando a tabela Cargo:

Tabela Funcionários (após modificação):

ID	Nome Funcionário	Salário	ID Cargo	ID Departamento
1	João	5000	1	1
2	Maria	6000	2	2
3	Pedro	4500	1	1

Tabela Cargo:

ID Cargo	Nome Cargo	Descrição
1	Vendedor	Vende produtos aos clientes
2	Analista	Analisa dados financeiros

Justificação da Melhoria:

- Redução de Redundância: Ao separar o cargo em uma tabela própria, evitamos repetições do mesmo texto (nome do cargo) em várias linhas da tabela Funcionários.
- Consistência e Normalização: A chave estrangeira ID Cargo na tabela Funcionários agora referencia diretamente a tabela Cargo, o que facilita a manutenção e garante a integridade referencial.

Portanto, ao criar a tabela Cargo e referenciar o ID Cargo na tabela Funcionários, estamos promovendo uma estrutura mais normalizada que atende aos critérios da Terceira Forma Normal (3FN), eliminando dependências transitivas e melhorando a organização e eficiência do banco de dados.

PROMPTS PARA APRENDER MAIS COM O CHATGPT

Nesta seção, exploraremos como você pode utilizar o ChatGPT para auxiliar no processo de normalização de suas tabelas de banco de dados. A normalização é crucial para organizar dados de maneira eficiente, reduzindo redundâncias e garantindo consistência. Aqui estão os passos e exemplos práticos de como o ChatGPT pode ajudar nesse processo:

Passo 1: Definição do Problema e Consulta ao ChatGPT

Antes de iniciar o processo de normalização, é importante entender a estrutura atual dos seus dados e identificar possíveis problemas de redundância ou dependências não necessárias. Aqui estão alguns prompts que você pode usar para iniciar uma conversa com o ChatGPT:

1. Prompt para Identificar Estrutura Atual: "Olá ChatGPT, estou trabalhando em um projeto de banco de dados e tenho uma tabela que parece ter muitos atributos misturados. Posso te fornecer a estrutura atual da tabela para obter sugestões sobre como normalizá-la?"
2. Prompt para Consultar sobre Dependências Funcionais: "Olá ChatGPT, estou revisando minha tabela de funcionários e quero garantir que todos os atributos sejam funcionalmente dependentes da chave primária. Você pode me ajudar a identificar quais atributos são dependentes de quais?"

Exemplo Prático com o ChatGPT:

Contexto: Você está trabalhando em um sistema de gerenciamento de vendas para uma loja e tem uma tabela que contém informações sobre pedidos e produtos. No entanto, os dados estão misturados e você deseja normalizá-los para melhorar a estrutura do banco de dados.

Consulta ao ChatGPT: Você decide usar o ChatGPT para orientá-lo no processo de normalização da tabela de pedidos e produtos.

Prompt Inicial: "Olá ChatGPT, estou desenvolvendo um banco de dados para um sistema de gerenciamento de vendas. Atualmente, tenho uma tabela que contém informações misturadas sobre pedidos e produtos. Posso te fornecer a estrutura atual da tabela para obter orientações sobre como normalizá-la?"

Resposta do ChatGPT: "Claro! Por favor, forneça a estrutura atual da tabela de pedidos e produtos."

Estrutura Atual da Tabela:

- ID Pedido
- Data Pedido
- ID Cliente
- Nome Cliente
- Endereço Cliente
- ID Produto
- Nome Produto
- Categoria Produto
- Preço Produto
- Quantidade

Continuação da Conversa: "Com base na estrutura atual da tabela, você pode me ajudar a separar os dados relacionados aos pedidos, clientes e produtos em tabelas separadas? Além disso, como posso garantir que cada atributo seja funcionalmente dependente da chave primária correspondente?"

ChatGPT: "Para normalizar essa tabela, você pode começar dividindo-a em três tabelas: Pedidos, Clientes e Produtos. Na tabela Pedidos, você teria ID Pedido, Data Pedido, ID Cliente e ID Produto como chaves estrangeiras. Na tabela Clientes, você pode ter ID Cliente, Nome Cliente e Endereço Cliente como atributos. E na tabela Produtos, ID Produto, Nome Produto, Categoria Produto e Preço Produto. Certifique-se de que cada atributo depende diretamente da chave primária correspondente, como ID Pedido, ID Cliente e ID Produto."

Ao usar o ChatGPT, você pode obter orientações valiosas sobre como normalizar suas tabelas de banco de dados, garantindo uma estrutura mais eficiente e organizada. O ChatGPT pode ajudar a identificar dependências funcionais, sugerir separações de tabelas e fornecer insights sobre as melhores práticas de design de banco de dados.

O ChatGPT pode oferecer ainda valiosas visões adicionais, exemplos práticos e orientações detalhadas sobre cada aspecto da normalização. Aqui estão alguns prompts específicos que podem ajudá-lo a aprofundar seu conhecimento:

1. Conceitos Básicos de Normalização:

- "Olá ChatGPT, posso aprender mais sobre os conceitos básicos de normalização de dados? Quais são os benefícios de normalizar um banco de dados?"
- "Como a normalização ajuda a reduzir a redundância e melhorar a consistência dos dados?"

2. Formas Normais (1FN, 2FN, 3FN):

- "ChatGPT, estou estudando as formas normais. Você pode explicar mais detalhadamente o que é a Primeira Forma Normal (1FN) e por que é importante?"
- "Quais são os critérios para alcançar a Segunda Forma Normal (2FN)? Como identificar dependências parciais em uma tabela?"
- "Explique a Terceira Forma Normal (3FN). Como identificar e eliminar dependências transitivas usando exemplos práticos?"

3. Dependência Funcional:

- "Posso aprender mais sobre dependência funcional entre atributos? Como identificar dependências funcionais em um esquema de banco de dados?"
- "Qual é o papel da dependência funcional no processo de normalização? Como ela influencia o design do banco de dados?"

4. Aplicações da Dependência Funcional na Normalização:

- "ChatGPT, gostaria de exemplos práticos de como identificar e aplicar dependências funcionais para normalizar um banco de dados."
- "Como as dependências funcionais são usadas para dividir uma tabela grande em estruturas mais eficientes?"

Exemplo de Conversa:

Prompt para Aprender sobre 1FN: "Olá ChatGPT, estou aprendendo sobre normalização de dados e gostaria de entender melhor o conceito de Primeira Forma Normal (1FN). Você pode explicar mais sobre isso?"

Resposta do ChatGPT: "Claro! A Primeira Forma Normal (1FN) é o primeiro passo na normalização de dados, onde cada coluna em uma tabela contém apenas valores atômicos (indivisíveis). Isso significa que não deve haver grupos repetidos de valores e cada campo deve conter apenas um valor. Por exemplo, em uma tabela de clientes, cada cliente deve ter suas informações únicas, sem repetição de dados. Posso te ajudar a aplicar esse conceito em um exemplo específico?"

Prompt para Aplicar 2FN: "ChatGPT, estou revisando uma tabela de pedidos e produtos. Como posso garantir que ela esteja na Segunda Forma Normal (2FN)?"

Resposta do ChatGPT: "Para estar na 2FN, é necessário eliminar dependências parciais da chave primária. Isso significa que cada atributo não chave deve depender totalmente da chave primária e não de apenas parte dela. Por exemplo, se você tem um campo 'Preço do Produto' dependendo apenas do 'ID do Produto', isso pode indicar uma dependência parcial. Posso ajudá-lo a identificar e corrigir isso na sua tabela?"

Utilizando o ChatGPT, você pode explorar ainda mais os conceitos e aplicações da normalização de dados. Os prompts fornecidos acima podem orientá-lo na aprendizagem contínua e na aplicação prática dos princípios de normalização, garantindo que seus projetos de banco de dados sejam eficientes, organizados e livres de inconsistências.

EXERCÍCIOS DE FIXAÇÃO

Abaixo estão as questões sobre normalização reformuladas com exemplos em formato de tabela, prontas para serem copiadas para o Word:

Questão 1: Primeira Forma Normal (1FN)

Considere a seguinte tabela que armazena informações sobre alunos e suas disciplinas:

Tabela Alunos:

ID_Aluno	Nome_Aluno	Disciplinas
1	João	Matemática, História
2	Maria	Inglês, Ciências

Esta tabela está na Primeira Forma Normal (1FN)? Justifique sua resposta e sugira uma forma de reorganizá-la, se necessário.

Questão 2: Dependências Parciais (2FN)

Analise a seguinte tabela que registra informações sobre pedidos e produtos:

Tabela Pedidos_Produtos:

ID_Pedido	Nome_Cliente	Produto	Preço_Produto
1	João	Camisa	R\$ 50,00
1	João	Calça	R\$ 80,00
2	Maria	Camiseta	R\$ 30,00

Identifique se há dependências parciais em relação à chave primária desta tabela. Explique como você poderia reestruturar esta tabela para estar na Segunda Forma Normal (2FN).

Questão 3: Terceira Forma Normal (3FN)

Considere a tabela a seguir que armazena informações sobre produtos e suas categorias:

Tabela Produtos_Categorias:

ID_Produto	Nome_Produto	Categoria	Descrição_Categoria
1	Camisa	Vestuário	Roupas
2	Calça	Vestuário	Roupas
3	Tênis	Calçados	Sapatos

Esta tabela está na Terceira Forma Normal (3FN)? Justifique sua resposta e sugira possíveis melhorias, se necessário.

Questão 4: Normalização de um Sistema de Vendas Online

Considere um sistema de vendas online que mantém registros de pedidos e produtos. A estrutura inicial das tabelas mistura informações dos clientes, produtos e pedidos da seguinte forma:

Tabela Pedidos_Produtos:

ID_Pedido	Nome_Cliente	Produto	Categoria_Produto	Preço_Produto	Endereço_Entrega
1	João	Camisa	Vestuário	R\$ 50,00	Rua A, São Paulo
1	João	Calça	Vestuário	R\$ 80,00	Rua A, São Paulo
2	Maria	Camiseta	Vestuário	R\$ 30,00	Rua B, Rio de Janeiro

Identifique os problemas de redundância e dependência parcial nesta estrutura. Proponha uma reorganização das tabelas para atender à Segunda Forma Normal (2FN) e à Terceira Forma Normal (3FN). Explique como você dividiria as informações em tabelas separadas, identificando as chaves primárias e estrangeiras necessárias.

Questão 5: Normalização de um Sistema de Gestão de Eventos

Imagine um sistema de gestão de eventos que registra informações sobre eventos, participantes e localizações em uma única tabela:

Tabela Eventos:

ID_Evento	Nome_Evento	Data_Evento	Nome_Participante	Email_Participante	Local_Evento	Capacidade_Local
1	Conferência	2023-07-15	João	joao@email.com	Auditório A	200
1	Conferência	2023-07-15	Maria	maria@email.com	Auditório A	200
2	Workshop	2023-08-20	Pedro	pedro@email.com	Sala B	50

Analise as dependências funcionais presentes nesta tabela. Proponha uma estrutura de tabelas normalizadas que atendam à Segunda Forma Normal (2FN) e à Terceira Forma Normal (3FN). Considere como você poderia dividir as informações de forma eficiente, evitando redundâncias e mantendo a integridade dos dados.

Questão 6: Normalização de um Sistema de Gestão Escolar

Considere um sistema de gestão escolar que mantém informações sobre alunos, turmas, disciplinas e notas em uma única tabela:

Tabela Escola:

ID_Aluno	Nome_Aluno	Turma	Disciplina	Nota
1	João	A	Matemática	8.5
1	João	A	História	7.0
2	Maria	B	Matemática	9.0
2	Maria	B	História	8.5
3	Pedro	A	Matemática	7.5
3	Pedro	A	História	6.5

Identifique as dependências transitivas e parciais nesta estrutura. Proponha uma divisão em tabelas que atendam à Terceira Forma Normal (3FN), explicando como você organizaria as tabelas de Alunos, Turmas, Disciplinas e Notas. Quais seriam as chaves primárias e estrangeiras necessárias?

Essas questões foram elaboradas para ajudar na compreensão e aplicação dos conceitos de normalização em cenários práticos de banco de dados.