Normalização de Dados

A **normalização** é o processo de organizar os dados em um banco de dados de forma a minimizar a redundância e evitar anomalias (de inserção, exclusão e atualização). Ela é feita através de formas normais (FN), sendo as mais comuns:

1. **Primeira Forma Normal (1FN):**
    - Elimina tabelas que contêm grupos repetidos. Cada célula deve conter um único valor.

    Exemplo:

Cliente	Telefone
João	9999-9999, 8888-8888

    Na 1FN, criamos uma tabela separada para telefones:

Cliente	Telefone
João	9999-9999
João	8888-8888

2. **Segunda Forma Normal (2FN):**
    - Remove dependências parciais. Ou seja, todos os atributos não-chave devem depender da chave primária inteira.

○ **Exemplo:** Suponha que temos uma tabela de pedidos:

PedidoID	Cliente	DataPedido	Produto	Preço
1	João	01/09/2023	Teclado	100
2	Maria	01/09/2023	Monitor	500

Na 2FN, os detalhes do produto (nome e preço) devem estar em uma tabela separada para evitar redundância de dados:

PedidoID	Cliente	DataPedido
1	João	01/09/2023
2	Maria	01/09/2023

Tabela de produtos:

ProdutoID	Produto	Preço
1	Teclado	100
2	Monitor	500

3. **Terceira Forma Normal (3FN):**
    - Remove dependências transitivas. Ou seja, atributos que não dependem diretamente da chave primária são eliminados.
    - **Exemplo:** Se na tabela de clientes tivermos uma coluna de cidade e uma coluna de código postal, isso pode causar redundância, pois o código postal depende da cidade e não diretamente do cliente.

Outro exemplo: Tabela Não Normalizada

Vamos supor que temos a seguinte tabela de um sistema de **vendas**:

PedidoID	Nome	Endereço	Produto	Qtde	R$ Unit.	R$ Total	Data
1	João Silva	Rua A, 100	Teclado	2	100	200	10/09/2023
2	Maria Souza	Rua B, 200	Monitor	1	500	500	11/09/2023
1	João Silva	Rua A, 100	Mouse	1	50	200	10/09/2023

Problemas nesta tabela:

- **Redundância de dados:** O nome e endereço do cliente aparecem repetidamente para o mesmo cliente (João Silva aparece duas vezes).
- **Anomalias de atualização:** Se o endereço do João Silva mudar, será necessário atualizar em todas as linhas onde ele aparece.

- **Anomalias de exclusão:** Se excluirmos o pedido 2 (de Maria Souza), perderemos todas as informações sobre ela, mesmo que queiramos apenas excluir o pedido.

1ª Forma Normal (1FN)

Na 1FN, eliminamos grupos repetidos e garantimos que cada coluna contenha valores atômicos (um único valor por célula). Isso já está no exemplo acima, pois cada célula contém apenas um valor. Porém, ainda podemos melhorar, dividindo a tabela em duas: uma para clientes e outra para pedidos.

- **Antes:** | PedidoID | ClienteNome | ClienteEndereço | Produto | Quantidade | PreçoUnitário | TotalPedido | DataPedido |

Depois da 1FN:
Criamos uma tabela para cliente

ClienteID	ClienteNome	ClienteEndereço
1	João Silva	Rua A, 100
2	Maria Souza	Rua B, 200

E outra tabela para os pedidos:

PedidoID	ClienteID	Produto	Quantidade	PreçoUnitário	TotalPedido	DataPedido
1	1	Teclado	2	100	200	10/09/2023
2	2	Monitor	1	500	500	11/09/2023
1	1	Mouse	1	50	200	10/09/2023

Agora temos uma tabela para **clientes** e outra para **pedidos**, com um relacionamento entre elas usando **ClienteID**.

2ª Forma Normal (2FN)

Na 2FN, removemos dependências parciais. Ou seja, todos os atributos devem depender da chave primária completa. No exemplo anterior, a chave primária da tabela de pedidos é o `PedidoID`. No entanto, o **PreçoUnitário** do produto não depende do pedido, mas sim do produto. Então, devemos mover as informações sobre o produto para uma tabela separada.

Depois da 2FN:
Criamos uma tabela separada para produtos:

ProdutoID	ProdutoNome	PreçoUnitário
1	Teclado	100
2	Monitor	500
3	Mouse	50

E a tabela de pedidos agora fica assim:

PedidoID	ClienteID	ProdutoID	Quantidade	DataPedido
1	1	1	2	10/09/2023
2	2	2	1	11/09/2023
1	1	3	1	10/09/2023

3ª Forma Normal (3FN)

Na 3FN, removemos **dependências transitivas**. Ou seja, se um atributo depende de outro que não seja a chave primária, ele deve ser movido para uma tabela separada. No nosso exemplo, não temos mais dependências transitivas. O preço unitário já foi removido da tabela de pedidos e colocado na tabela de produtos, o que significa que estamos na 3FN.

Resultado Final (Após Normalização até 3FN)

Tabela de Clientes**:**

ClienteID	ClienteNome	ClienteEndereço
1	João Silva	Rua A, 100
2	Maria Souza	Rua B, 200

Tabela de Produtos:

ProdutoID	ProdutoNome	PreçoUnitário
1	Teclado	100
2	Monitor	500
3	Mouse	50

Tabela de Pedidos:

PedidoID	ClienteID	ProdutoID	Quantidade	DataPedido

1	1	1	2	10/09/2023
2	2	2	1	11/09/2023
1	1	3	1	10/09/2023

Benefícios da Normalização:

- **Redução de Redundância:** O nome e endereço do cliente não são repetidos em várias linhas. Agora eles estão em uma tabela separada.
- **Melhor Manutenção:** Se um cliente mudar de endereço, basta atualizar em uma única linha.
- **Evita Anomalias:** Se quisermos excluir um pedido, isso não afeta os dados do cliente ou do produto.

A **normalização** ajuda a garantir que os dados sejam consistentes, fáceis de manter e sem redundâncias, ao mesmo tempo em que organiza de maneira eficiente as informações.

Exercício de fixação

ID_Aluno	Nome_Aluno	Curso	Professor	Sala
1	João	Matemática	Prof. Carlos	101
2	Maria	Matemática	Prof. Carlos	101
1	João	Física	Prof. Fernanda	102
3	Pedro	Química	Prof. Laura	103
2	Maria	Física	Prof. Fernanda	102