PROF RAFAEL ASSUNÇÃO

BD I
Normalização

Objetivo da Aula

Compreender a importância da normalização

Estudar as 3 formas normais

Aplicar as formas normais com exemplos práticos

1. O que é Normalização?

Definição:

A normalização é o processo de organizar os dados em um banco de dados relacional de forma a reduzir a redundância e dependência. O principal objetivo é evitar problemas como anomalias de **inserção**, **atualização** e **deleção**.

Por que Normalizar?

- Redução de redundância: Dados duplicados podem ser evitados.
- Melhoria na integridade: Evita inconsistências.
- Eficiência nas consultas: Dados bem estruturados facilitam consultas rápidas.
- A normalização é dividida em formas normais, que são um conjunto de regras para garantir que o banco de dados seja organizado de maneira eficiente.

Anomalia de Inserção

A anomalia de inserção ocorre quando não é possível inserir dados sem ter que inserir dados desnecessários ou redundantes.

Exemplo:

Imagine uma tabela Empregado que armazena informações sobre o empregado e o departamento onde ele trabalha:

Empregado_ID	Nome	Departamento	Chefe
1	João	Vendas	Carlos Silva
2	Maria	TI	João Costa
3	Pedro	Vendas	Carlos Silva

Se quisermos inserir um novo departamento, por exemplo, o departamento **"Marketing"**, mas ainda não temos nenhum empregado associado a ele, seríamos obrigados a inserir dados repetidos do chefe e outros campos desnecessários, como:

Empregado_ID	Nome	Departamento	Chefe
NULL	NULL	Marketing	Carlos Silva

Anomalia de Atualização

A anomalia de atualização ocorre quando, ao atualizar um dado, há a necessidade de atualizar várias linhas em várias tabelas para garantir **consistência**.

Exemplo:

Na tabela de Empregado que já mostrei anteriormente, se o nome do chefe mudar, precisamos atualizar em todas as linhas do departamento de um chefe específico. Se houver 10, 20 ou mais empregados no mesmo departamento, teríamos que atualizar todas essas entradas.

Empregado_ID	Nome	Departamento	Chefe
1	João	Vendas	Carlos Silva
2	Maria	TI	João Costa
3	Pedro	Vendas	Carlos Silva

Problema (Anomalia de Atualização):

Se Carlos Silva deixar de ser o chefe e for substituído por alguém, precisaríamos alterar a informação do chefe em cada linha onde ele é mencionado. Isso pode gerar inconsistências se esquecermos de atualizar alguma linha.

Anomalia de Deleção

A anomalia de deleção ocorre quando a exclusão de um dado leva à perda de informações importantes ou à inconsistência de dados.

Exemplo:

Considerando a tabela original Empregado e Departamento:

Empregado_ID	Nome	Departamento	Chefe
1	João	Vendas	Carlos Silva
2	Maria	TI	João Costa
3	Pedro	Vendas	Carlos Silva

Problema (Anomalia de Deleção):

Se decidirmos excluir o **empregado João** (por exemplo), a informação do chefe **Carlos Silva** também é perdida, mesmo que ele seja o chefe de outros departamentos ou empregados. Isso pode causar perda de dados importantes.

Primeira Forma Normal (1FN)

Definição:

A tabela está em 1FN quando todos os atributos (colunas) contêm apenas valores atômicos (indivisíveis) e não há grupos repetitivos de dados.

Requisitos para 1FN:

- Nenhuma coluna pode conter múltiplos valores (listas, arrays, etc.).
- Todos os dados devem ser atômicos, ou seja, cada valor deve ser único e indivisível.

Exemplo:

(não está na 1FN)

idCliente	Nome	Telefones
1	João da Silva	1111-1111, 2222-2222
2	Maria Magalhães	3333-3333,4444-4444

Problema:

A coluna "Telefones" contém múltiplos valores, o que não é permitido na 1FN e o nome está como um atributo composto, contendo nome e sobrenome.

Para normalizar a tabela é necessário criar um atributo para sobrenome, para transformarmos nome em um atributo simples e criar uma tabela para Telefones, afim de criar atributos atômicos.

idCliente	nomeCliente	sobrenomeCliente
1	João	da Silva
2	Maria	Magalhães

idTel	Tel	idCliente
1	1111-1111	1
2	2222-2222	1
3	3333-3333	2
4	4444-4444	2

Segunda Forma Normal (2FN)

Definição:

A tabela está em 2FN quando já está em 1FN e, além disso, todos os atributos não-chave são totalmente dependentes da chave primária.

Requisitos para 2FN:

- A tabela já deve estar em 1FN.
- Nenhum atributo não-chave pode depender apenas de uma parte da chave primária (se a chave for composta).

Exemplo:

(Tabela em 1FN)

idCliente	Nome	Produto	Preço
1	João	Chocolate ao Leite	R$ 7,50
1	João	Chocolate Branco	R$ 7,50
2	Maria	Chocolate Meio Amargo	R$ 12,50

Problema:

O nome do cliente depende de idCliente, mas o produto e o preço dependem de idCliente e do produto. Isso viola a 2FN.

Outro problema é haver redundância de tuplas, duplicando o idCliente e o nome do mesmo.

idCliente	Nome
1	João
2	Maria

idProduto	Produto	Preço
1	Chocolate ao Leite	R$ 7,50
2	Chocolate Branco	R$ 7,50
3	Chocolate Meio Amargo	R$ 12,50

Agora, a dependência funcional está completamente sobre a chave primária em ambas as tabelas.

Terceira Forma Normal (3FN)

Definição:

A tabela está em 3FN quando já está em 2FN e não há dependências transitivas entre os atributos não-chave.

Requisitos para 3FN:

- A tabela já deve estar em 2FN.
- Nenhuma dependência transitiva (A -> B -> C) pode existir entre os atributos.

Exemplo:

(Tabela em 2FN)

idCliente	nome	cidade	estado
1	João	São Paulo	SP
2	Maria	Rio de Janeiro	RJ

Problema:

A cidade e o estado dependem do cliente, mas também estão relacionadas entre si (cidade -> estado). Isso cria uma dependência transitiva.

idCliente	nome	idCidade
1	João	1
2	Maria	2

idCidade	cidade	estado
1	São Paulo	SP
2	Rio de Janeiro	RJ

Agora, a tabela está livre de dependências transitivas.

Passos para Normalizar até a 3FN

1. **Ter uma tabela não normalizada;**
2. **Remover atributos multivalorados e compostos;**
3. **Remover dependências parciais;**
4. **Remover Dependências Transitivas.**