

Dependência funcional em Banco de Dados

Definição de Dependência Funcional

1

Chave Primária

A chave primária é um conjunto de atributos que identificam unicamente cada registro em uma tabela.

2

Determinante

Um determinante é um conjunto de atributos que determina o valor de outros atributos em uma tupla.

3

Dependência Funcional

Uma dependência funcional ocorre quando o valor de um atributo (ou conjunto de atributos) determina univocamente o valor de outro atributo.

Determinante

- Determinante em banco de dados é um atributo ou conjunto de atributos que determina funcionalmente o valor de outro atributo. Em outras palavras, quando conhecemos o valor do determinante, podemos identificar com precisão o valor do atributo determinado.
- Exemplo: Em uma tabela de funcionários, o atributo CPF determina o Nome do funcionário.
- Conhecendo o CPF, sabemos exatamente qual é o nome associado.

Chave Primária

- Uma chave primária é um atributo ou conjunto de atributos que identifica de forma única cada registro em uma tabela. Ela não pode conter valores nulos e deve ser única para cada registro.
- Exemplo: Em uma tabela de alunos, o número de matrícula pode ser usado como chave primária, pois cada aluno possui um número único.

Dependência Funcional

Uma dependência funcional ocorre quando o valor de um atributo (ou conjunto de atributos) determina de maneira única o valor de outro atributo. Representamos isso como $X \rightarrow Y$, onde X determina Y .

- **Exemplo:** Se tivermos uma tabela de produtos onde cada Código de Produto determina um único Nome de Produto, podemos representar isso como: $\text{Código_Produto} \rightarrow \text{Nome_Produto}$.

Tipos de Dependência Funcional

Dependência Trivial

Uma dependência funcional trivial ocorre quando um atributo é determinado por sua própria chave primária.

Dependência Parcial

Uma dependência parcial ocorre quando um atributo não-chave é determinado por apenas parte da chave primária.

Dependência Transitiva

Uma dependência transitiva ocorre quando um atributo não-chave é determinado por outro atributo não-chave.

Dependência Trivial

Uma dependência trivial ocorre quando um atributo depende de um conjunto de atributos que o contém. Em outras palavras, Y é dependente de X de forma trivial se Y for um subconjunto de X .

- **Exemplo didático:** Se tivermos os atributos {CPF, Nome, Idade} e dissermos que $\{CPF, Nome\} \rightarrow Nome$, isso é uma dependência trivial porque Nome está contido no conjunto {CPF, Nome}.

CPF	Nome	Idade
123.456.789-00	Ana	25
987.654.321-00	Carlos	30
111.222.333-44	Beatriz	28

- Neste caso, $\{\text{CPF}, \text{Nome}\} \rightarrow \text{Nome}$ é trivial porque o lado direito (Nome) já está contido no lado esquerdo.

Dependência Parcial

- Uma dependência parcial ocorre quando um atributo não-chave depende apenas de parte de uma chave primária composta (formada por múltiplos atributos).

Imagine uma tabela de matrículas em cursos:

ID_Aluno	ID_Curso	Nome_Aluno	Nome_Curso
1	101	Ana	Banco de Dados
1	102	Ana	Programação
2	101	Bruno	Banco de Dados

Nesta tabela, a chave primária é composta pelos atributos {ID_Aluno, ID_Curso}. No entanto:

- Nome_Aluno depende apenas de ID_Aluno: $ID_Aluno \rightarrow Nome_Aluno$
- Nome_Curso depende apenas de ID_Curso: $ID_Curso \rightarrow Nome_Curso$

Essas são dependências parciais, pois os atributos não-chave dependem apenas de parte da chave primária composta.

Dependência Transitiva

- Uma dependência transitiva ocorre quando um atributo não-chave depende de outro atributo não-chave, que por sua vez depende da chave primária.

Considere uma tabela de funcionários:

CPF (PK)	Depto_ID	Nome_Depto
123.456.789-00	10	TI
987.654.321-00	20	Marketing
111.222.333-44	10	TI

Nesta tabela temos:

- CPF → Depto_ID (CPF determina o departamento do funcionário)
- Depto_ID → Nome_Depto (ID do departamento determina o nome do departamento)
- Logo, CPF → Nome_Depto por transitividade

Esta é uma dependência transitiva, pois Nome_Depto depende de CPF indiretamente, através de Depto_ID.

Importância da Dependência Funcional

Modelagem Eficiente

Entender as dependências funcionais é crucial para projetar um modelo de dados relacional eficiente e evitar redundâncias.

Normalização

As dependências funcionais guiam o processo de normalização, que visa eliminar anomalias e garantir a integridade dos dados.

Consultas Otimizadas

O conhecimento das dependências funcionais ajuda a formular consultas SQL mais eficientes e rápidas.

Integridade de Dados

As dependências funcionais asseguram que os dados mantêm sua consistência e integridade dentro do banco de dados.

Identificação de Dependências Funcionais

1

Análise Conceitual

Compreender o domínio da aplicação e as regras de negócio é fundamental para identificar as dependências funcionais.

2

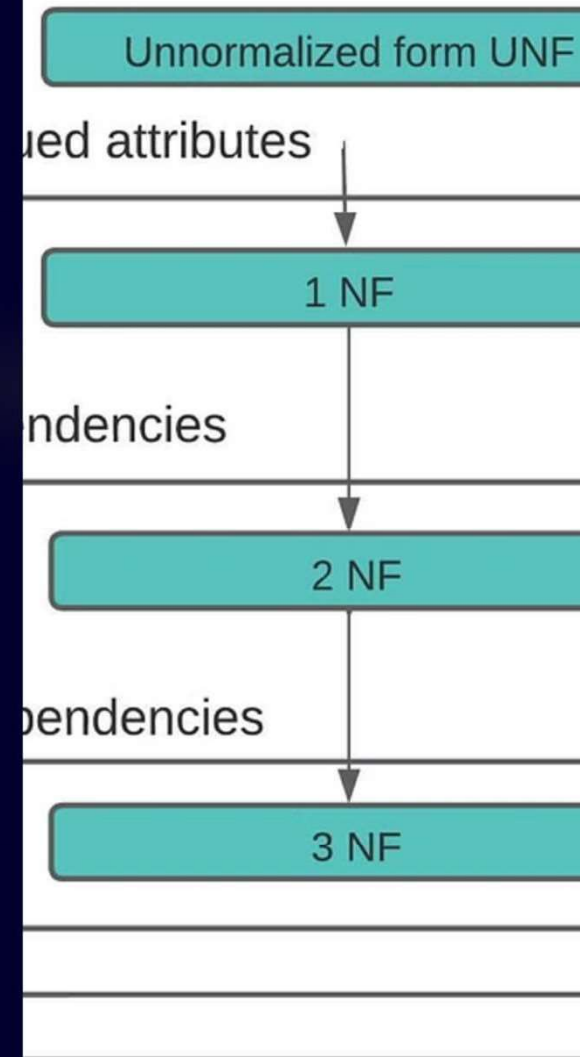
Estudo dos Dados

Examinar cuidadosamente os atributos e seus valores ajuda a revelar as relações de dependência entre eles.

3

Testes e Validação

Aplicar técnicas como testes de inserção, atualização e exclusão para validar as dependências funcionais identificadas.



Decomposição de Relações

1

Identificação de Dependências

Analisar as dependências funcionais presentes em uma relação para determinar sua decomposição.

2

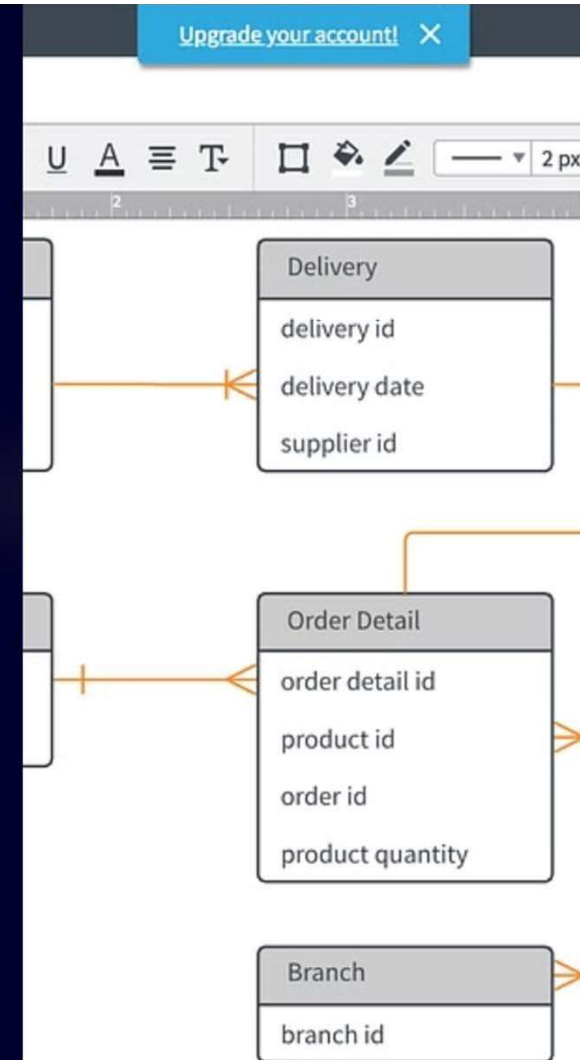
Criação de Novas Relações

Dividir a relação original em novas relações, cada uma contendo um conjunto de atributos determinados por suas próprias chaves primárias.

3

Preservação da Integridade

Garantir que a decomposição preserve as dependências funcionais originais e não introduza anomalias de dados.



Normalização de Bancos de Dados



1FN

Primeira Forma Normal: Elimina atributos compostos e multivalorados.



2FN

Segunda Forma Normal: Elimina dependências funcionais parciais.



3FN

Terceira Forma Normal: Elimina dependências funcionais transitivas.



FNBC

Forma Normal de Boyce-Codd: Elimina todas as dependências funcionais.

Aplicações da Dependência Funcional

Modelagem de Dados

As dependências funcionais guiam a criação de esquemas de banco de dados robustos e eficientes.

Otimização de Consultas


O conhecimento das dependências ajuda a formular consultas SQL mais rápidas e otimizadas.

Garantia de Integridade

As dependências funcionais asseguram a consistência e integridade dos dados no banco de dados.

Manutenção de Dados

As dependências funcionais facilitam a realização de operações de atualização, inserção e exclusão de dados.



Introdução à Normalização em Banco de Dados

A normalização de banco de dados é uma técnica fundamental para garantir a integridade e eficiência dos seus dados. Ela envolve a estruturação do seu banco de dados em múltiplas tabelas, estabelecendo relações entre elas e eliminando redundâncias. Esse processo ajuda a evitar problemas comuns, como atualizações anômalas, inserções e exclusões incorretas, além de facilitar a manutenção e a expansão do seu sistema no futuro.

Database Development Phases

O que é Normalização?

Identificação de Entidades

O primeiro passo na normalização é identificar as entidades, ou seja, os objetos do mundo real que serão representados no banco de dados. Isso envolve definir os principais tipos de informação que serão armazenados.

1

Criação de Tabelas

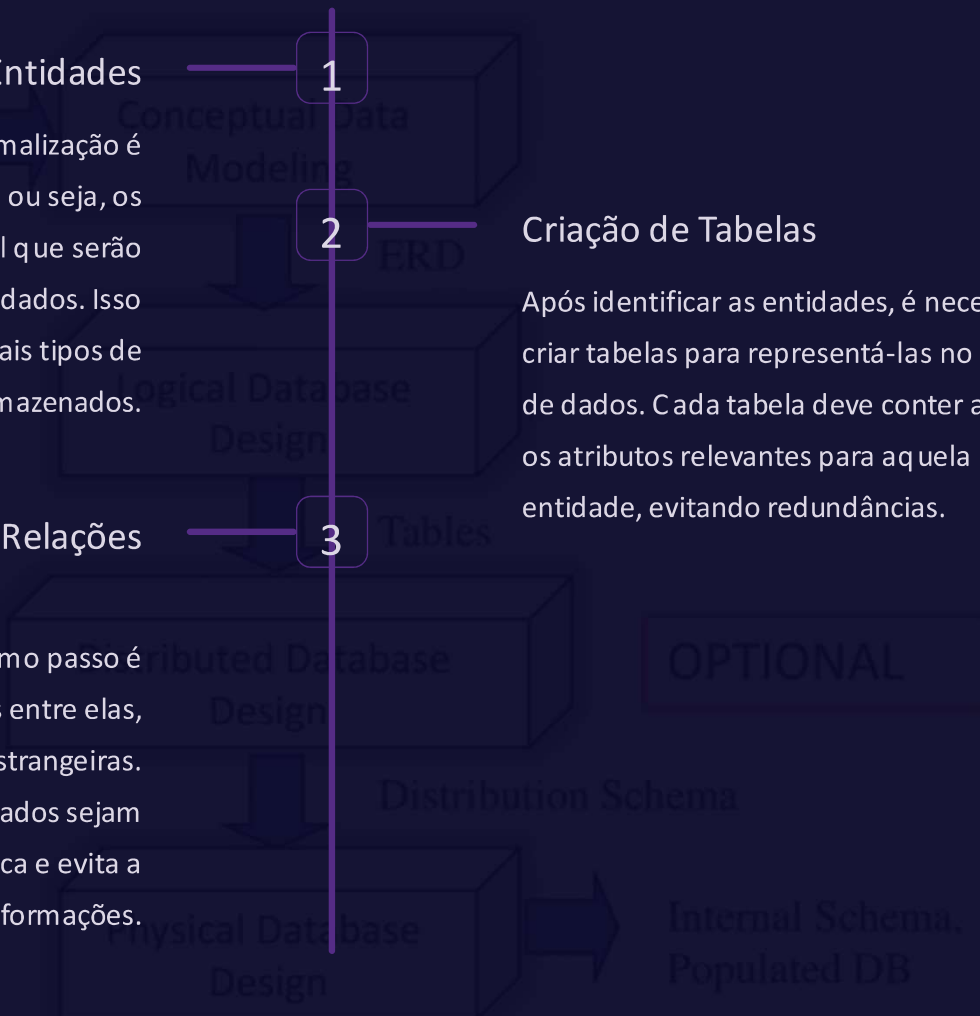
Após identificar as entidades, é necessário criar tabelas para representá-las no banco de dados. Cada tabela deve conter apenas os atributos relevantes para aquela entidade, evitando redundâncias.

2

Estabelecimento de Relações

Com as tabelas criadas, o próximo passo é estabelecer as relações entre elas, utilizando chaves primárias e estrangeiras. Isso permite que os dados sejam organizados de forma lógica e evita a duplicação de informações.

3



Primeira Forma Normal (1FN)

1

Eliminar Valores Repetidos

A primeira forma normal exige que cada célula da tabela contenha apenas um valor atômico, ou seja, um único valor indivisível. Isso significa que não podem haver campos com múltiplos valores ou listas de valores.

2

Criar Chave Primária

Cada tabela deve ter uma chave primária única que identifique cada registro de forma exclusiva. Essa chave pode ser composta por um ou mais atributos.

3

Evitar Dependências Parciais

Todas as colunas da tabela devem depender apenas da chave primária, evitando dependências parciais que possam gerar anomalias.

Tabela não normalizada:

ID_Cliente	Nome_Cliente	Telefones
1	João	99887766, 88776655
2	Maria	77665544

Esta tabela não está na 1FN porque o atributo "Telefones" não é atômico.

Tabela na 1FN:

ID_Cliente	Nome_Cliente	Telefone
1	João	99887766
1	João	88776655
2	Maria	77665544

Segunda Forma Normal (2FN)

Dependência Funcional Total

A segunda forma normal exige que todos os atributos não-chave dependam de forma completa da chave primária da tabela. Isso significa que nenhum atributo pode depender apenas de uma parte da chave primária composta.

Eliminação de Dependências Parciais

Caso existam dependências parciais, elas devem ser removidas, criando-se novas tabelas para armazenar esses dados de forma correta.

Melhoria na Integridade

Ao atingir a 2FN, o banco de dados se torna mais íntegro, pois reduz a redundância de dados e previne atualizações anômalas.

Tabela na 1FN, mas não na 2FN

ID_Aluno	ID_Curso	Nome_Aluno	Nome_Curso
1	101	Ana	Banco de Dados
1	102	Ana	Programação
2	101	Bruno	Banco de Dados

Problema: Nome_Aluno depende apenas de ID_Aluno e Nome_Curso depende apenas de ID_Curso, sendo dependências parciais.

Solução (tabelas na 2FN):

- Tabela Matrícula:

ID_Aluno	ID_Curso
1	101
1	102
2	101

Tabela Aluno:

ID_Aluno	Nome_Aluno
1	Ana
2	Bruno

Tabela Curso:

ID_Curso	Nome_Curso
101	Banco de Dados
102	Programação

Terceira Forma Normal (3FN)

Transitividade

A terceira forma normal exige que não haja dependências transitivas entre os atributos não-chave. Isso significa que nenhum atributo não-chave deve depender de outro atributo não-chave.

Eliminação de Dependências Transitivas

Caso existam dependências transitivas, elas devem ser removidas, criando-se novas tabelas para armazenar esses dados de forma independente.

Redução de Redundância

Ao atingir a 3FN, o banco de dados fica ainda mais livre de redundâncias, melhorando a eficiência do armazenamento e reduzindo a chance de erros.

Melhoria da Manutenção

A terceira forma normal facilita a manutenção do banco de dados, pois as alterações em um atributo não afetam outros atributos que não dependem dele.

Tabela na 2FN, mas não na 3FN:

CPF (PK)	Nome	Depto_ID	Nome_Depto
123.456.789-00	Ana	10	TI
987.654.321-00	Bruno	20	Marketing
111.222.333-44	Carolina	10	TI

Problema: Nome_Depto depende de Depto_ID, que por sua vez depende de CPF (dependência transitiva).

Solução (tabelas na 3FN):

Tabela Funcionário:

CPF (PK)	Nome	Depto_ID
123.456.789-00	Ana	10
987.654.321-00	Bruno	20
111.222.333-44	Carolina	10

Tabela Departamento:

Depto_ID (PK)	Nome_Depto
10	TI
20	Marketing

Terceira Forma Normal (3FN)

A terceira forma normal (3NF) é uma forma normal do modelo relacional de banco de dados que visa eliminar redundâncias de dados e inconsistências, garantindo assim a integridade dos dados. Para exemplificar o uso da terceira forma normal, considere uma tabela de informações de produtos em uma loja, onde temos os seguintes atributos: SKU (Stock Keeping Unit - unidade de controle de estoque), Nome do Produto, Categoria do Produto e Descrição da Categoria.

SKU	Nome do Produto	Categoria	Descrição da Categoria
001	Camiseta	Roupas	Roupas Masculinas
002	Calça	Roupas	Roupas Masculinas
003	Vestido	Roupas	Roupas Femininas
004	Celular	Eletrônicos	Smartphones
005	Notebook	Eletrônicos	Computadores Portáteis

Esta tabela não está na 3NF, pois apresenta redundância de dados na coluna "Descrição da Categoria". A redundância ocorre porque a descrição da categoria está sendo repetida para cada produto com a mesma categoria. Para normalizar essa tabela para a 3NF, devemos dividir a tabela em duas, uma para os produtos e outra para as categorias, estabelecendo uma relação entre elas.

A Forma Normal de Boyce-Codd (BCNF)

É uma forma normal mais restritiva do que a Terceira Forma Normal (3NF). Ela garante que não haja dependências funcionais parciais, onde um atributo não chave depende parcialmente de uma chave candidata. Para ilustrar a BCNF, considere uma tabela de informações de funcionários, onde temos os seguintes atributos: Número de Identificação do Funcionário (ID), Nome do Funcionário, Departamento e Nome do Gerente.

ID	Nome do Funcionário	Departamento	Nome do Gerente
001	João Silva	Vendas	Maria Oliveira
002	Ana Santos	Vendas	Maria Oliveira
003	Carlos Pereira	TI	Marcos Costa
004	Maria Oliveira	RH	João Silva

Nesta tabela, temos uma dependência funcional parcial entre os atributos "Nome do Gerente" e "Departamento". Ou seja, o nome do gerente está determinando o departamento em que o funcionário trabalha. Para normalizar essa tabela para a BCNF, devemos dividir a tabela em duas, uma para os funcionários e outra para os departamentos, estabelecendo uma relação entre elas.