

O QUE SÃO FORMAS NORMAIS EM BANCOS DE DADOS?

Você sabe o que são formas normais? Saiba por que conhecer essas formas vai fazer você construir um excelente banco de dados!

As Formas Normais são uma série de procedimentos aplicados em um banco de dados para garantir que as suas tabelas estejam bem estruturadas e não contenham nenhum tipo de anomalia, seja de inclusão, atualização ou exclusão.

O que são anomalias? **Anomalias são inconsistências** em um banco de dados.

Damos a essa série de procedimentos o nome de Normalização.

Mas antes de dar início à apresentação das formas normais, vamos abordar 5 conceitos necessários para uma melhor compreensão. Os conceitos que falaremos agora serão:

- Dependência Funcional
- Dependência Funcional Parcial
- Dependência Funcional Transitiva
- Atributos Multivalorados
- Atributos Compostos

Conceito 1: Dependência Funcional:

Uma dependência funcional é um relacionamento entre dois ou mais atributos, de forma que o valor de um atributo identifica o valor do outro atributo, ou seja, um atributo está relacionado a outro, se nós sabemos qual o CPF de uma pessoa também vamos descobrir o nome, por exemplo.

CLIENTE	
<u>CPF</u>	Nome
111	Ana
222	Bruno
333	Carla
444	Diego
555	Ana

Conceito 2: Dependência Funcional Parcial:

Uma dependência funcional parcial ocorre quando os atributos não chave (não identificadores) não dependam de toda a chave primária quando ela for composta.

Chave primária: Consideramos como chave primária toda coluna da tabela que identifica de forma única cada linha da tabela.

No caso deste boletim temos três colunas que a sua composição de valores identifica o aluno de forma única. Essas colunas são Matrícula_Aluno, Cod_Disciplina e Período.

BOLETIM				
<u>Matrícula_Aluno</u>	<u>Cod_Disciplina</u>	<u>Período</u>	Nome_Disciplina	Nota
111	1	1	Cálculo I	8
111	2	1	Física I	9.5
111	3	1	Contabilidade I	10
111	4	1	Química	7.5
111	5	1	Intro. Eng.	8.5

Boletim

Perceba também que o nome da disciplina depende somente do código da disciplina, ou seja, depende de parte da chave primária (dependência funcional parcial) enquanto, a nota, depende da matrícula do aluno, código da disciplina e do período em que o aluno está matriculado.

Conceito 3: Dependência Funcional Transitiva:

Quando um ou mais campos de uma entidade não são dependentes diretamente da chave primária, ou de parte dela, mas sim dependente de outro campo da tabela (campo este que não a Chave Primária), temos uma dependência funcional transitiva.

Nesta tabela todos os funcionários são identificados de forma única pelo seu ID_Funcionario.

A coluna com Nome_Funcionários depende da coluna primária ID_Funcionario assim como a coluna com o ID_Cargo, porém, a coluna Nome_Cargo não depende da coluna ID_Funcionario.

A coluna Nome_Cargo depende somente da coluna ID_Cargo que não é nossa coluna primária, isso é o que chamamos de dependência transitiva.

Esse tipo de dependência pode ser corrigida na estrutura do banco de dados, assim, teremos uma coluna a menos nesta tabela e com isso menos informações repetidas.

FUNCIONÁRIO				
ID_Funcionario	Nome_Funcionario	ID_Cargo	Nome_Cargo	Salario
1	João	1	Analista	3500
2	Katia	1	Analista	3500
3	Luis	2	Técnico	4100
4	Maria	2	Técnico	4100
5	Neto	3	Assistente	2200

Funcionário

Conceito 4: Atributos Multivalorados:

Atributos multivalorados são atributos que podem conter mais de um valor para um mesmo registro. Nesta tabela, por exemplo, temos mais de um número de telefone para um mesmo funcionário.

Embora sejam números diferentes, ter mais de uma informação em uma mesma célula vai gerar problemas em um banco de dados.

FUNCIONÁRIO		
ID_Funcionario	Nome_Funcionario	Telefone
1	João	(21) 99999-0001 (21) 99999-0002 (21) 99999-0003
2	Katia	(21) 99999-0004
3	Luis	(21) 99999-0005 (21) 99999-0006
4	Maria	(21) 99999-0007
5	Neto	(21) 99999-0008

Funcionário

Conceito 5: Atributos Compostos:

Atributos compostos são atributos que poderiam ser subdivididos em vários atributos:

Neste caso o endereço tem as informações de bairro, estado e cidade todos na mesma célula.

ID_Funcionario	Nome_Funcionario	Endereço
1	João	Av. Mem de Sá, 100, apto 101 – Centro – Rio de Janeiro - RJ
2	Katia	Av. Portugal, 324, casa 01 – Urca – Rio de Janeiro – RJ
3	Luis	Av. Vieira Souto, 1300, apto 802, Leblon – Rio de Janeiro – RJ

Endereço completo

O ideal seria dividir esta única coluna em várias colunas, para eliminarmos o atributo composto:

ID_Funcionario	Nome_Funcionario	Endereço	Bairro	Cidade	UF
1	João	Av. Mem de Sá, 100, apto 101	Centro	Rio de Janeiro	RJ
2	Katia	Av. Portugal, 324, casa 01	Urca	Rio de Janeiro	RJ
3	Luis	Av. Vieira Souto, 1300, apto 802	Leblon	Rio de Janeiro	RJ

Colunas separadas

O que é a Normalização?

Podemos definir a Normalização como uma sequência de passos e verificações aplicadas a um banco de dados visando eliminar, ou pelo menos minimizar, as redundâncias e inconsistências no banco.

Tal procedimento é feito a partir da identificação de anomalias de inserção, exclusão e atualização em uma relação, decompondo-a em relações mais bem estruturadas e minimizando a redundância.

Eliminar a redundância nos dados possui algumas vantagens:

- Reduzir o espaço necessário para armazenar o banco de dados
- Melhorar a organização dos dados
- Reduzir o impacto de atualizações, inserções e exclusões nos dados dos bancos de dados

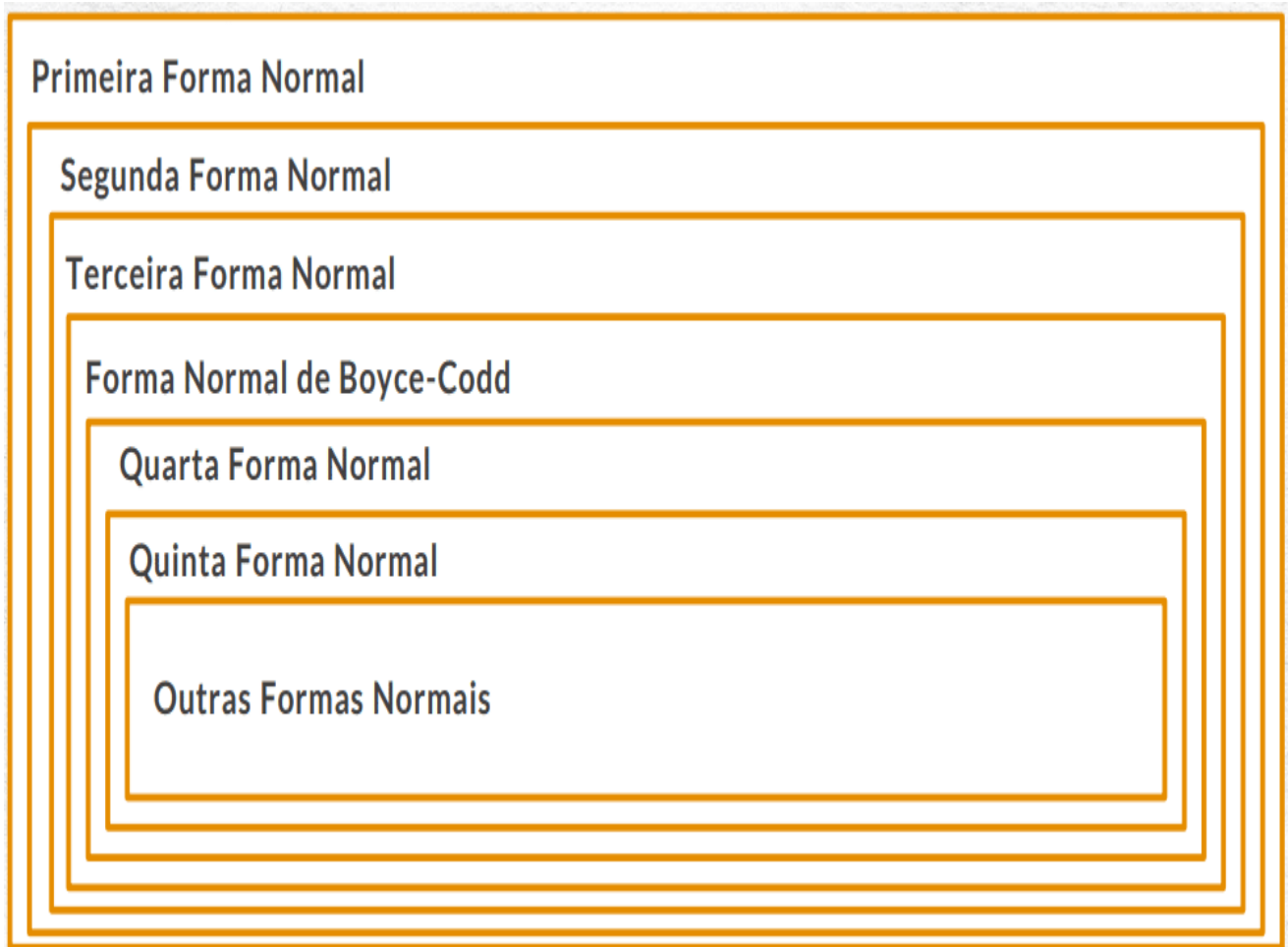
O processo de normalização é aplicado em etapas, conhecidas como **Formas Normais**, que vão garantir que o banco de dados fique bem estruturado.

Existem uma série de formas normais na literatura, são elas:

- Primeira Forma Normal
- Segunda Forma Normal

- Terceira Forma Normal
- Forma Normal de Boyce-Codd
- Quarta Forma Normal
- Quinta Forma Normal

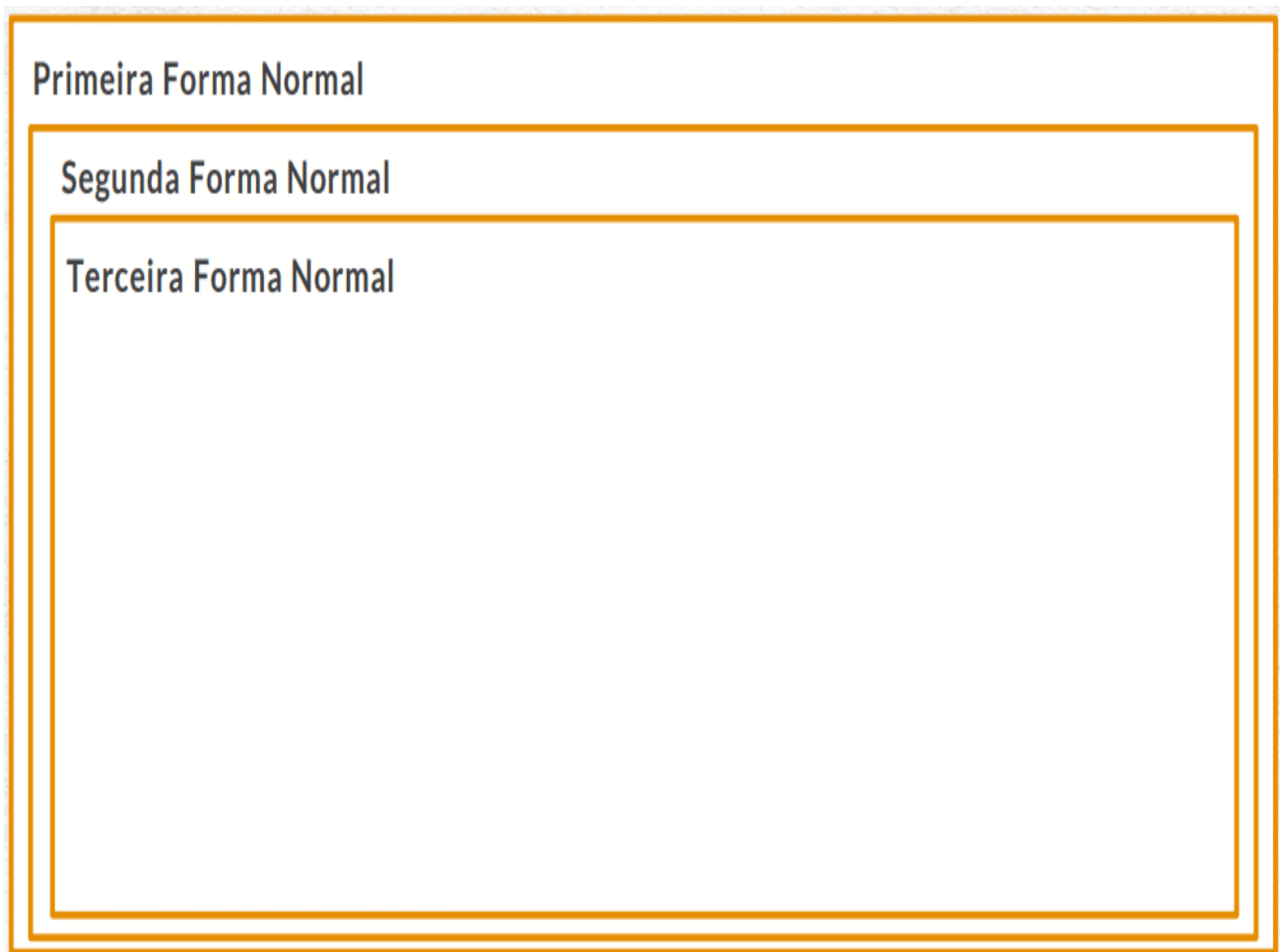
As Formas Normais são como se fossem fases de um jogo. Você só pode passar para a próxima se estiver adequado à Forma anterior:



Formas normais

Para muitos autores, a aplicação das três primeiras já é suficiente para garantir que o banco de dados não terá redundâncias e inconsistências.

Então, neste curso vamos focar nas três primeiras formas normais.



3 principais formas

Primeira Forma Normal (1FN):

A primeira forma normal visa eliminar atributos multivalorados e atributos compostos.

Passo a passo da 1FN:

Em resumo, para adequar uma tabela que não está na 1FN é necessário realizar os seguintes passos:

1. Identificar a existência de atributos multivalorados e atributos compostos.
2. Criar uma tabela para armazenar os dados do atributo multivalorado.

As colunas dessa nova tabela devem ser compostas por: (1) atributo multivalorado da tabela original e (2) chave primária da tabela original.

A chave primária da tabela original se transforma na chave estrangeira da nova tabela.

- Remover o atributo multivalorado da tabela original.
- Na tabela original, para cada atributo composto, criar uma coluna para cada informação a ser desmembrada.

Exemplo Prático: Tabela PESSOA

Vejamos um exemplo. Na tabela abaixo, temos que uma série de atributos sobre a entidade PESSOA.

Essa tabela não está na Primeira Forma Normal. Vamos fazer a adequação.

PESSOA				
CPF	Nome	Sexo	Localização	Telefone
111	Ana	F	Rio de Janeiro, RJ	999-444, 999-000
222	Bruno	M	São Paulo, SP	888-888, 444-333
333	Carla	F	Belo Horizonte, MG	555-777
444	Diego	M	Vitória, ES	999-999

Tabela pessoa.

1. Identificar a existência de atributos multivalorados (não atômicos) e atributos compostos.

Na tabela em questão, temos que o atributo Localização é um atributo composto, o que significa que ele poderia ser desmembrado em mais de uma informação: Cidade e Estado.

PESSOA				
CPF	Nome	Sexo	Localização	Telefone
111	Ana	F	Rio de Janeiro, RJ	999-444, 999-000
222	Bruno	M	São Paulo, SP	888-888, 444-333
333	Carla	F	Belo Horizonte, MG	555-777
444	Diego	M	Vitória, ES	999-999

Atributo Composto

Atributo composto

Por outro lado, o atributo Telefone é um atributo multivalorado (não atômico). Ou seja, um atributo que contém, na mesma célula, várias ocorrências daquele mesmo atributo.

PESSOA

CPF	Nome	Sexo	Localização	Telefone
111	Ana	F	Rio de Janeiro, RJ	999-444, 999-000
222	Bruno	M	São Paulo, SP	888-888, 444-333
333	Carla	F	Belo Horizonte, MG	555-777
444	Diego	M	Vitória, ES	999-999

Atributo
Multivalorado

Atributo multivalorado

- Criar uma tabela para armazenar os dados do atributo multivalorado.

As colunas dessa nova tabela devem ser compostas por: (1) atributo multivalorado da tabela original e (2) chave primária da tabela original.

A chave primária da tabela original se transforma na chave estrangeira da nova tabela.

Na tabela original, identificamos como chave primária o atributo CPF, e como atributo multivalorado a coluna Telefone.

A partir dessas duas colunas, criamos uma tabela nova.

PESSOA

CPF	Nome	Sexo	Localização	Telefone
111	Ana	F	Rio de Janeiro, RJ	999-444, 999-000
222	Bruno	M	São Paulo, SP	888-888, 444-333
333	Carla	F	Belo Horizonte, MG	555-777
444	Diego	M	Vitória, ES	999-999

CPF	Telefone
111	999-444
111	999-000
222	888-888
222	444-333
333	555-777
444	999-999

Criando uma tabela



Relacionando as tabelas



Liquidando atributo multivalorado

- Remover o atributo multivalorado da tabela original.

Agora que criamos uma tabela para armazenar as informações de telefones das pessoas, não precisamos mais da coluna Telefone na tabela PESSOA.

PESSOA					TELEFONE	
CPF	Nome	Sexo	Localização	Telefone	CPF	Telefone
111	Ana	F	Rio de Janeiro, RJ	999-444, 999-000	111	999-444
222	Bruno	M	São Paulo, SP	888-888, 444-333	111	999-000
333	Carla	F	Belo Horizonte, MG	555-777	222	888-888
444	Diego	M	Vitória, ES	999-999	222	444-333
					333	555-777
					444	999-999

Eliminando a coluna.

PESSOA				TELEFONE	
CPF	Nome	Sexo	Localização	CPF	Telefone
111	Ana	F	Rio de Janeiro, RJ	111	999-444
222	Bruno	M	São Paulo, SP	111	999-000
333	Carla	F	Belo Horizonte, MG	222	888-888
444	Diego	M	Vitória, ES	222	444-333
				333	555-777
				444	999-999

Resultado

- Na tabela original, para cada atributo composto, criar uma coluna para cada informação a ser desmembrada.

A coluna **Localização** é **um atributo composto**. Sabemos que podemos separá-la em mais de uma, com as informações de Cidade e Estado.

PESSOA

CPF	Nome	Sexo	Localização
111	Ana	F	Rio de Janeiro, RJ
222	Bruno	M	São Paulo, SP
333	Carla	F	Belo Horizonte, MG
444	Diego	M	Vitória, ES

PESSOA

CPF	Nome	Sexo	Cidade	Estado
111	Ana	F	Rio de Janeiro	RJ
222	Bruno	M	São Paulo	SP
333	Carla	F	Belo Horizonte	MG
444	Diego	M	Vitória	ES

A coluna Localização foi dividida em duas, contendo a informação de Cidade e Estado separadamente.

Desfazendo atributo composto.

Exemplo Prático: Tabela PESSOA:

E assim, corrigimos a tabela criando duas outras tabelas, fazendo com que no final a Primeira Forma Normal seja atingida.



Resultado

Segunda Forma Normal (2FN):

Uma tabela encontra-se na segunda forma normal se ela atende todos os requisitos da primeira forma normal e se os registros na tabela, que não são chaves, dependam da chave primária em sua totalidade e não apenas parte dela.

Aqui, vamos eliminar a dependência funcional parcial.

Passo a passo da 2FN:

Em resumo, para adequar uma tabela que não está na 2FN é necessário realizar os seguintes passos:

1. Identificar se existe na tabela uma chave primária composta.
2. Identificar cada atributo não chave que dependa de apenas parte da chave primária composta.

3. Criar uma tabela para cada conjunto de atributos não chave que dependam de parte da chave primária da tabela e adicionar estes atributos não chaves na tabela. As chaves primárias nas novas tabelas devem ter como base as chaves primárias da tabela original.

4. Remover os atributos não chave da tabela original que dependam de parte da chave primária.

Exemplo Prático: Tabela FUNCIONARIO_PROJETO:

Exemplo: a tabela abaixo contém duas colunas que formam uma chave primária: **id_func** e **id_proj**.

Repare agora, por exemplo, na coluna **nome_func**.

Essa coluna depende apenas de parte da chave primária, ou seja, ela depende apenas do **id_func**.

Da mesma forma que as colunas **nome_proj** e **local_proj** dependem apenas da coluna **id_proj**.

<u>id_func</u>	<u>id_proj</u>	horas_trabalhadas	nome_func	nome_proj	local_proj
1	A	10	Luis	Plan. Est.	RJ
2	B	20	Marta	Plano Neg.	SP
1	B	15	Luis	Plano Neg.	SP
3	C	30	Neide	Map. Processos	MG

Tabela exemplo

1. Identificar se existe uma chave primária composta na tabela.

A chave primária da tabela **funcionário_projeto** é composta por duas colunas: **id_func** e **id_proj**.

Chave Primária Composta

<u>id_func</u>	<u>id_proj</u>	horas_trabalhadas	nome_func	nome_proj	local_proj
1	A	10	Luis	Plan. Est.	RJ
2	B	20	Marta	Plano Neg.	SP
1	B	15	Luis	Plano Neg.	SP
3	C	30	Neide	Map. Processos	MG

Chave primária composta.

2. Identificar cada atributo não chave que dependa de apenas parte da chave primária composta.

O atributo não chave **nome_func** depende apenas da coluna **id_func** (parte da chave primária).

Chave Primária Composta

<u>id_func</u>	<u>id_proj</u>	horas_trabalhadas	nome_func	nome_proj	local_proj
1	A	10	Luis	Plan. Est.	RJ
2	B	20	Marta	Plano Neg.	SP
1	B	15	Luis	Plano Neg.	SP
3	C	30	Neide	Map. Processos	MG

O nome do funcionário depende apenas do id_func (e não de id_proj)

Chave primária composta

Já os atributos não chave **nome_proj** e **local_proj** dependem apenas da coluna **id_proj** (parte da chave primária).

Chave Primária Composta

<u>id_func</u>	<u>id_proj</u>	horas_trabalhadas	nome_func	nome_proj	local_proj
1	A	10	Luis	Plan. Est.	RJ
2	B	20	Marta	Plano Neg.	SP
1	B	15	Luis	Plano Neg.	SP
3	C	30	Neide	Map. Processos	MG

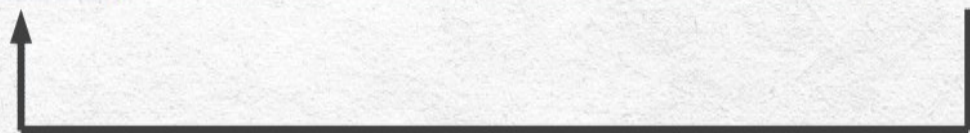


O nome do projeto depende apenas do id_proj (e não de id_func)

Chave primária composta

Chave Primária Composta

<u>id_func</u>	<u>id_proj</u>	horas_trabalhadas	nome_func	nome_proj	local_proj
1	A	10	Luis	Plan. Est.	RJ
2	B	20	Marta	Plano Neg.	SP
1	B	15	Luis	Plano Neg.	SP
3	C	30	Neide	Map. Processos	MG



O local do projeto depende apenas do id_proj (e não de id_func)

Chave primária composta

E o atributo não chave “horas_trabalhadas”?

<u>id_func</u>	<u>id_proj</u>	horas_trabalhadas	nome_func	nome_proj	local_proj
1	A	10	Luis	Plan. Est.	RJ
2	B	20	Marta	Plano Neg.	SP
1	B	15	Luis	Plano Neg.	SP
3	C	30	Neide	Map. Processos	MG

Atributo não chave.

Nesta tabela, este atributo depende da chave primária como um todo.

As horas trabalhadas estão associadas a um determinado funcionário alocado em um determinado projeto.

- Criar uma tabela para cada conjunto de atributos não chave que dependam de parte da chave primária da tabela e adicionar estes atributos não chaves na tabela.

As chaves primárias nas novas tabelas devem ter como base as chaves primárias da tabela original.

Temos, portanto, dois conjuntos de atributos não chave dependentes de parte da chave primária, são eles:

- nome_func → depende apenas de id_func
- nome_proj e local_proj → dependem apenas de id_proj

Portanto, criaremos mais duas tabelas, uma para cada conjunto de atributos não chaves não dependentes de toda a chave primária:

1. FUNCIONARIO

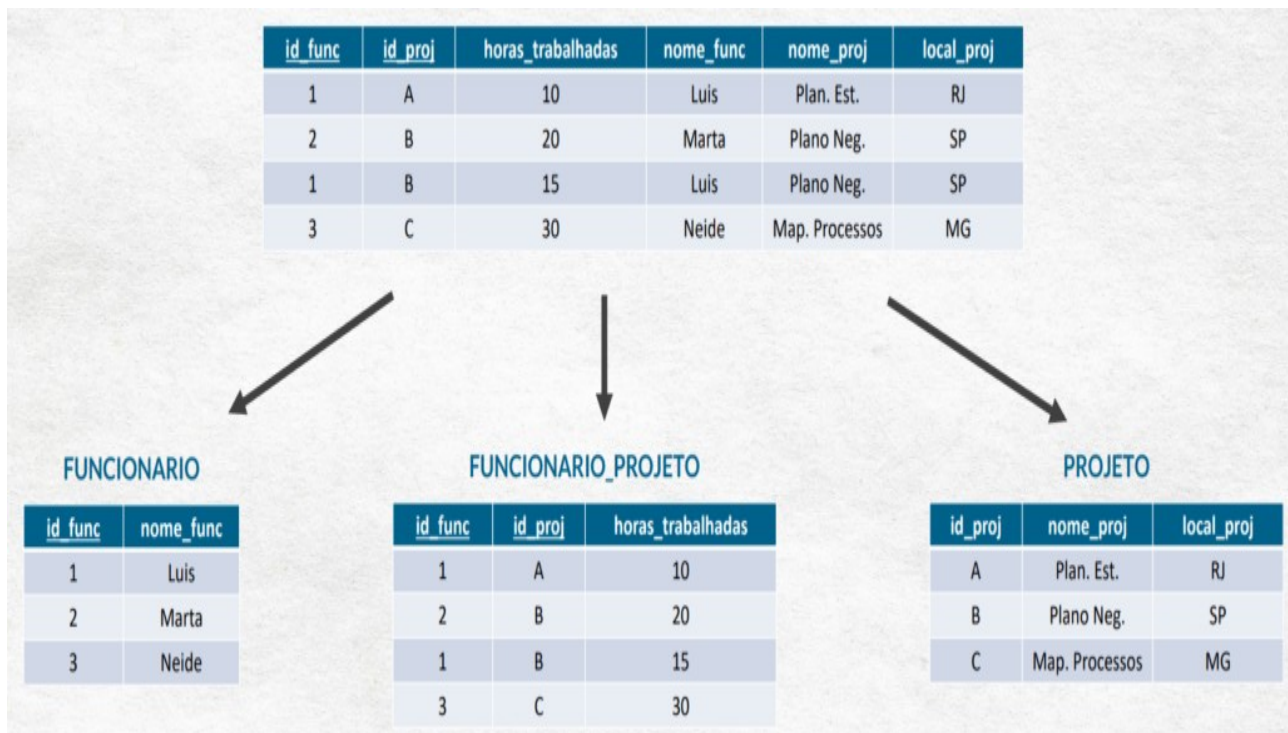
2. PROJETO

4) Remover os atributos não chave da tabela original que dependam de parte da chave primária.

Os mesmos atributos não-chave serão removidos da tabela original, e existirão apenas nas novas tabelas.

- nome_func → será removida por depender apenas de id_func
- nome_proj e local_proj → serão removidas por dependerem apenas de id_proj

Assim, podemos dividir nossa tabela inicial em três tabelas separadas, desta forma quando acrescentarmos informações de projetos ou funcionários não vamos precisar repetir valores.



Resultado

Terceira Forma Normal (3FN):

Se uma tabela está na primeira e segunda forma normal, mas ao analisarmos um registro encontramos um atributo não chave dependente de outro atributo não chave, precisaremos corrigir a tabela para a terceira forma normal.

Aqui basicamente corrigiremos a dependência funcional transitiva.

Passo a passo da 3FN:

Em resumo, para adequar uma tabela que não está na 3FN é necessário realizar os seguintes passos:

1. Identificar cada grupo de atributos não-chave que dependam de outros atributos não-chave.
2. Criar uma tabela para armazenar os atributos (ou conjunto de atributos) não-chave que não estão relacionados à chave primária da tabela original.

Definir como chave primária da tabela criada o atributo que é capaz de obter os dados não chave da tabela original e mover os atributos não chave que não são obtidos exclusivamente pela chave primária da tabela original para a nova tabela.

- Definir como chave estrangeira o atributo que é capaz de obter os dados não chaves da tabela original.

Exemplo Prático: Tabela FUNCIONÁRIO:

Vejamos um exemplo.

Na tabela abaixo, temos uma lista de funcionários e suas respectivas informações.

A chave primária dessa tabela é a coluna **id_func**.

<u>id_func</u>	nome_func	Sexo	id_dep	nome_dep	gerente_dep
1	Paula	F	100	Finanças	André
2	Rodrigo	M	101	RH	Bruna
3	Sandra	F	102	TI	Caio
4	Tiago	M	101	RH	Bruna

Tabela id_func.

Porém, existem atributos dessa tabela que não dependem da chave primária da tabela.

As colunas **nome_dep** e **gerente_dep** não são definidas pela chave primária **id_func**, mas sim pelo atributo **id_dep**.

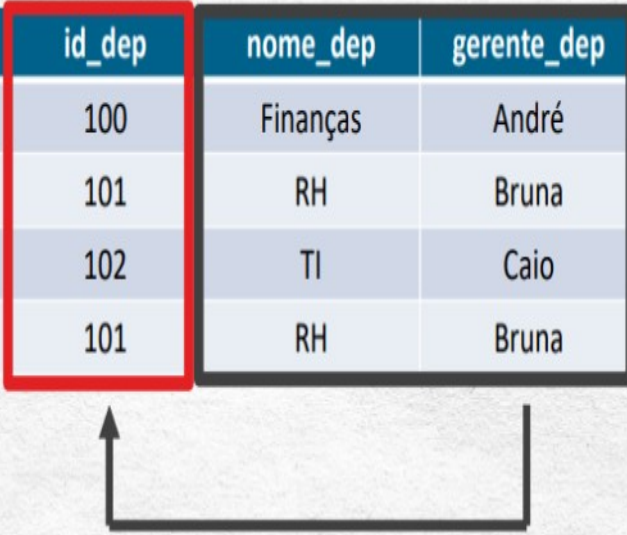
Ou seja, temos atributos não chave **nome_dep** e **gerente_dep** dependendo exclusivamente de outro atributo não chave **id_dep**.

1. Identificar cada grupo de atributos não-chave que dependam de outros atributos não-chave.

Existem atributos dessa tabela que não dependem da chave primária da tabela?

<u>id_func</u>	nome_func	Sexo	id_dep	nome_dep	gerente_dep
1	Paula	F	100	Finanças	André
2	Rodrigo	M	101	RH	Bruna
3	Sandra	F	102	TI	Caio
4	Tiago	M	101	RH	Bruna

Chave Primária



Exemplo.

- Criar uma tabela para armazenar os atributos (ou conjunto de atributos) não-chave que não estão relacionados à chave primária da tabela original.

Definir como chave primária da tabela criada o atributo que consegue obter os dados não chave da tabela original e mover os atributos não chave que não são obtidos exclusivamente pela chave primária da tabela original para a nova tabela.

Criamos uma tabela chamada DEPARTAMENTO, contendo os atributos não-chave que não dependem da chave primária da tabela original + atributo não-chave da tabela original que serve para identificar os outros atributos não-chave.

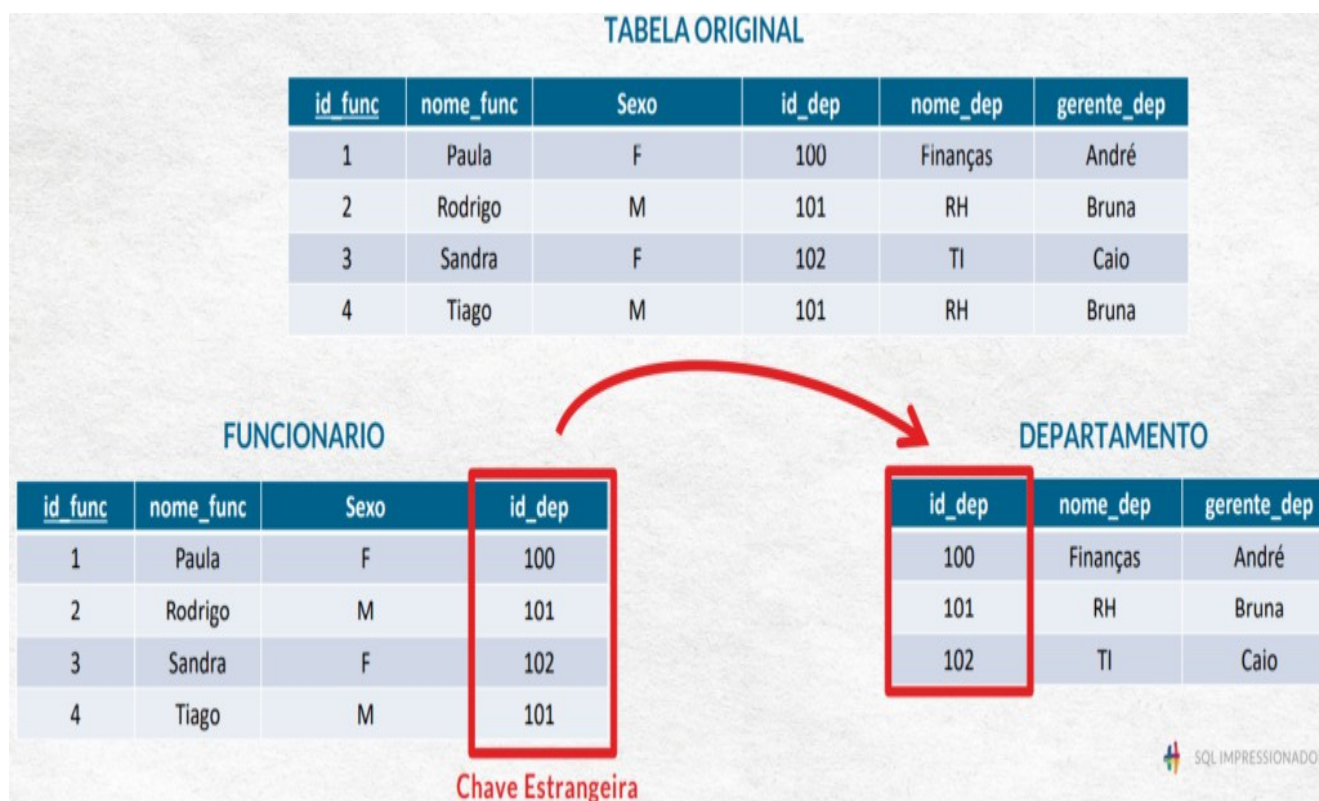
DEPARTAMENTO		
id_dep	nome_dep	gerente_dep
100	Finanças	André
101	RH	Bruna
102	TI	Caio

Departamento.

- Definir como chave estrangeira o atributo que é capaz de obter os dados não chaves da tabela original.

Desmembramos a nossa tabela original em duas.

Na tabela original mantemos apenas a chave primária e os atributos não chave associados a ela e transformamos **id_dep** em chave estrangeira, para se conectar com a tabela nova criada, de DEPARTAMENTO.



Chave estrangeira.

A chave estrangeira é responsável por conectar as informações com a nova tabela.

Resumo:

- Um banco de dados mal projetado pode apresentar anomalias de inserção, exclusão e atualização.
- Anomalias em bancos de dados geram redundâncias e inconsistências, o que prejudica o bom desempenho do banco de dados.
- Um banco de dados mal projetado pode apresentar anomalias de inserção, exclusão e atualização.
- Normalização é um processo de adequação do banco de dados por meio de Formas Normais, a fim de eliminar anomalias nos bancos de dados.
- Podemos considerar que um banco de dados está normalizado e livre de redundâncias e inconsistências se estiver adequado à Primeira, Segunda e Terceira Formas Normais.

Conclusão – O que são Formas Normais

Nesta aula ensinamos o processo de normalização, falamos sobre anomalias no banco de dados, dependência funcional, atributos compostos, 3 formas normais etc.

Nosso objetivo é fazer com que, antes de você **projetar um banco de dados**, já tenha essas etapas em mente.

Construir o banco de dados pautado nessas regras de normalização é muito melhor do que ter que voltar e corrigir o banco de dados sempre a cada atualização por não ter feito o melhor processo.

Esse é o nosso grande objetivo com essa aula! Que você entenda o que é a normalização e a sua importância, assim poderá se antecipar aos problemas criando seu banco de dados da melhor forma!

[Pesquisado em](#)