

N_BANC DAD II4-a - Texto de apoio

Site: [EAD Mackenzie](#)

Tema: BANCO DE DADOS {TURMAS 03B} 2023/1

Livro: N_BANC DAD II4-a - Texto de apoio

Impresso por: FELIPE BALDIM GUERRA .

Data: quinta, 6 abr 2023, 23:33

Índice

1. NORMALIZAÇÃO
2. PRIMEIRA FORMA NORMAL – 1FN
3. DEPENDÊNCIA FUNCIONAL
4. SEGUNDA FORMA NORMAL – 1FN
5. TERCEIRA FORMA NORMAL – 3FN
6. REFERÊNCIAS

1. NORMALIZAÇÃO

Introdução

Você sabe o que é normalização?

Para que o modelo de seu banco de dados esteja bem modelado, ou seja, não tenha redundância, existem algumas diretrizes que você deve utilizar. A normalização possui “regras” para eliminar redundâncias.

O objetivo da normalização é reagrupar informações para eliminar redundâncias.

A normalização é utilizada tanto no processo de modelagem de um banco de dados quanto no processo de engenharia reversa.

O termo engenharia reversa vem do fato de usar-se como ponto de partida do processo um produto implementado para obter sua especificação (modelo conceitual). Nos exemplos a seguir, será utilizada a engenharia reversa para explicar e exemplificar as seguintes formas normais: 1FN (1ª forma normal), 2FN (2ª forma normal) e 3FN (3ª forma normal).

Mas o que é uma forma normal?

Uma forma normal é uma regra que deve ser obedecida por uma tabela para que esta seja “bem projetada”.

Existem diversas formas normais, e cada uma delas elimina um tipo de redundância.

Um banco de dados normalizado até a 3FN é o mais utilizado na prática.

Exemplo: considere o conjunto de dados da Figura 1, com muita redundância, no qual temos as seguintes informações:

- para cada projeto, são informados o código, a descrição e o tipo do projeto, bem como os empregados que atuam no projeto;
- para cada empregado, são informados seu código, nome, categoria funcional, salário de acordo com sua categoria funcional, data em que o empregado foi alocado ao projeto e o tempo (em meses) pelo qual o empregado foi alocado ao projeto.

Atenção: observe, na Figura 1, que temos dados duplicados referentes a um mesmo projeto, uma vez que vários empregados podem trabalhar no mesmo projeto e, também, a um mesmo empregador, pois um empregado pode trabalhar em mais de um projeto.

Figura 1 – Conjunto de dados representado na forma de uma tabela não normalizada

Proj			Emp					
Cod_proj	Tipo	Descrição	Cod_emp	Nome	Categoria	Salário	Data_ini	Tempo
LSC001	Novo Desenv.	Sist. Estoque	2146	João	A1	4	1/11/01	24
LSC001	Novo Desenv.	Sist. Estoque	3145	Sílvio	A2	4	2/10/01	24
LSC001	Novo Desenv.	Sist. Estoque	6126	José	B1	9	3/10/02	18
LSC001	Novo Desenv.	Sist. Estoque	1214	Carlos	A2	4	4/10/02	18
LSC001	Novo Desenv.	Sist. Estoque	8191	Mário	A1	4	1/11/02	12
PAG02	Manutenção	Sist. RH	8191	Mário	A1	4	1/05/03	12
PAG02	Manutenção	Sist. RH	2146	João	A1	4	4/01/01	24
PAG02	Manutenção	Sist. RH	6126	José	B1	9	1/11/02	12

Adaptada de: Heuser (2009).

Uma representação da extensão do modelo relacional (um pouco modificada) para representar tabelas aninhadas seria:

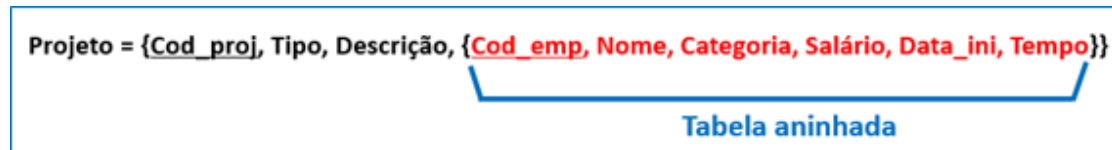
Projeto = {Cod_proj, Tipo, Descrição, {Cod_emp, Nome, Categoria, Salário, Data_ini, Tempo}}

Com o modelo relacional correspondente ao conjunto de dados, passa-se para o processo de normalização.

2. PRIMEIRA FORMA NORMAL – 1FN

Com o Uma tabela encontra-se na 1FN quando NÃO contém tabelas aninhadas. Ou seja, o domínio de um atributo deve incluir apenas valores atômicos (simples e indivisíveis) e o valor de qualquer atributo em uma tupla (linha) deve ser um único valor.

Se você observar, temos uma tabela aninhada, ou seja, uma tabela dentro de uma outra, como mostrado a seguir:



Portanto, a passagem para a 1FN consta da eliminação das tabelas aninhadas. E, para transformar uma tabela não normalizada em um esquema que obedeça a regra da 1FN, cria-se uma tabela referente à própria tabela e uma tabela para cada tabela aninhada.

Exemplo da 1FN: no caso da Figura 1 cujo conjunto de dados contém muita redundância, as **relações resultantes da 1FN** seriam:

Proj = {Cod_proj, Tipo, Descrição}

ProjEmp = {Cod_proj, Cod_emp, Nome, Categoria, Salário, Data_ini, Tempo}

A decomposição das tabelas para a 1FN é feita nos seguintes passos:

- é criada uma tabela na 1FN referente à tabela não normalizada e que contém apenas os atributos com valores atômicos, isto é, sem as tabelas aninhadas. A chave primária da tabela na 1FN é idêntica à chave da tabela não normalizada;
- para cada tabela aninhada, é criada uma tabela na 1FN composta pelos seguintes atributos:
 - a chave primária de cada uma das tabelas na qual a tabela em questão está aninhada;
 - os atributos da própria tabela aninhada;

- a chave primária da tabela aninhada será a chave primária dela mesma caso seja suficiente e, caso contrário, deve-se determinar quais os demais atributos necessários para identificar as linhas da tabela na 1FN, compondo, assim, a chave primária na 1FN.

O conteúdo das tabelas na 1FN, considerando os dados que tínhamos, inicialmente, na Figura 1, ficaria da seguinte forma (Figura 2):

Figura 2 – Tabelas (com dados) referentes à 1FN

Proj			Proj_Emp						
Cod_proj	Tipo	Descrição	Cod_proj	Cod_emp	Nome	Categoria	Salário	Data_ini	Tempo
LSC001	Novo Desenv.	Sist. Estoque	LSC001	2146	João	A1	4	1/11/01	24
PAG02	Manutenção	Sist. RH	LSC001	3145	Silvio	A2	4	2/10/01	24
			LSC001	6126	José	B1	9	3/10/02	18
			LSC001	1214	Carlos	A2	4	4/10/02	18
			LSC001	8191	Mário	A1	4	1/11/02	12
			PAG02	8191	Mário	A1	4	1/05/03	12
			PAG02	2146	João	A1	4	4/01/01	24
			PAG02	6126	José	B1	9	1/11/02	12

Adaptada de: Heuser (2009).

O que é importante você observar nos dados da Figura 2:

- a **tabela Proj** na 1FN **eliminou um tipo de redundância**, ou seja, temos uma linha para cada projeto diferente;
- a **tabela ProjEmp foi criada** e sua chave primária é formada pelos atributos Cod_proj e Cod_emp, pois um empregado pode trabalhar em mais de um projeto (se um empregado pudesse trabalhar em um único projeto, a chave primária seria composta apenas pelo Cod_emp);
- na **tabela ProjEmp, ainda temos redundância**, pois os dados dos empregados estão duplicados no caso deles participaram de mais de um projeto.

3. DEPENDÊNCIA FUNCIONAL

Para você compreender a 2FN e a 3FN, é necessário conhecer o conceito de **dependência funcional**, ou seja:

em uma tabela relacional, diz-se que um atributo C2 depende funcionalmente de um atributo C1 (ou vice-versa) quando, em todas as linhas da tabela, para cada valor de C1 que aparece na tabela, aparece o mesmo valor de C2.

O conceito fica mais fácil com o exemplo destes dados, a seguir:

...	Código	...	Salário	...
	E1		10	
	E3		9	
	E1		10	
	E2		5	
	E3		9	
	E2		5	
	E1		10	

Podemos dizer que o **atributo Salário depende funcionalmente de um atributo Código** pelo fato de cada valor de Código estar associado sempre ao mesmo valor de Salário, ou seja:

- sempre que aparece o valor “E1” do atributo Código, o valor do atributo Salário é “10”;
- sempre que aparece o valor “E2” do atributo Código, o valor do atributo Salário é “5”;
- sempre que aparece o valor “E3” do atributo Código, o valor do atributo Salário é “9”.

De forma geral, o determinante de uma dependência funcional pode ser um conjunto de atributos, e não somente um atributo.

4. SEGUNDA FORMA NORMAL – 1FN

Uma tabela encontra-se na 2FN quando:

- além de encontrar-se na 1FN, cada atributo que não faz parte da chave primária depende da chave primária completa.

Uma tabela que não se encontra na 2FN contém dependências funcionais parciais, ou seja, contém atributos não chave que dependem apenas de uma parte da chave primária. Obviamente, uma tabela que está na 1FN e cuja chave primária é formada por um único atributo não contém dependências parciais. Isso porque, nesta tabela, é impossível um atributo depender de uma parte da chave primária, visto que a chave primária é composta por um único atributo (atômico e indivisível).

Importante: toda tabela que está na 1FN e que possui apenas um atributo como chave primária já está na 2FN. O mesmo conceito aplica-se para uma tabela que contenha apenas atributos que fazem parte da chave primária.

Para transformarmos um banco de dados para a 2FN, pegamos as relações na 1FN e aplicamos a regra da 2FN.

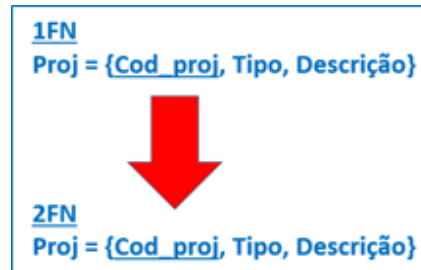
Tabelas na 1FN (que já vimos):

Proj = {Cod_proj, Tipo, Descrição}

ProjEmp = {Cod_proj, Cod_emp, Nome, Categoria, Salário, Data_ini, Tempo}

O processo de passagem da 1FN para a 2FN é o seguinte:

- copiar para a 2FN cada tabela que tenha chave primária simples ou que não tenha atributos além da chave primária. No caso do nosso exemplo, é o que acontece com a tabela Proj;




- para cada tabela com chave primária composta e com, pelo menos, uma coluna não chave (no nosso exemplo, a tabela ProjEmp), fazer a seguinte pergunta para cada atributo não chave: **“o atributo depende de toda a chave ou de apenas parte dela?”**

1. Caso o **atributo dependa de toda a chave**, criar o atributo correspondente na tabela com a chave completa na 2FN (os atributos Data_ini e Tempo da tabela ProjEmp na 2FN). Ou seja, os **atributos Data_ini e Tempo dependem da chave primária completa**, pois, para determinar a data em que um empregado começou a trabalhar em um projeto, bem como para determinar o tempo pelo qual ele foi alocado ao projeto, é necessário conhecer tanto o código do projeto quanto o código do empregado;

Proj Emp						
Cod_proj	Cod_emp	Nome	Categoria	Salário	Data_ini	Tempo
LSC001	2146	João	A1	4	1/11/01	24
LSC001	3145	Silvio	A2	4	2/10/01	24
LSC001	6126	José	B1	9	3/10/02	18
LSC001	1214	Carlos	A2	4	4/10/02	18
LSC001	8191	Mirio	A1	4	1/11/02	12
PAG02	8191	Mirio	A1	4	1/05/03	12
PAG02	2146	João	A1	4	4/01/01	24
PAG02	6126	José	B1	9	1/11/02	12

2. caso o **atributo dependa apenas de parte da chave**, deve ser criada (caso ainda não exista) uma tabela na 2FN que tenha como chave primária a parte da chave que é determinante do atributo em questão (a tabela Emp na 2FN). Ou seja, os **atributos Nome, Categoria e Salário dependem, cada um, apenas do atributo Cod_emp**, já que esses atributos são determinados tão somente pelo código do empregado (os atributos Nome, Salário e Categoria da tabela Emp na 2FN).



Cod_proj	Cod_emp	Nome	Categoria	Salário	Data_ini	Tempo
LSC001	2146	João	A1	4	1/11/01	24
LSC001	3145	Silvio	A2	4	2/10/01	24
LSC001	6126	José	B1	9	3/10/02	18
LSC001	1214	Carlos	A2	4	4/10/02	18
LSC001	8191	Mário	A1	4	1/11/02	12
PAG02	8191	Mário	A1	4	1/05/03	12
PAG02	2146	João	A1	4	4/01/01	24
PAG02	6126	José	B1	9	1/11/02	12

Assim, o **modelo relacional correspondente à 2FN** é o seguinte:

Proj = {Cod_proj, Tipo, Descrição}

Emp = {Cod_emp, Nome, Categoria, Salário}

ProjEmp = {Cod_proj, Cod_emp, Data_ini, Tempo}

O conteúdo das tabelas na 2FN, considerando os dados que tínhamos na 1FN – Figura 2, ficaria da seguinte forma:

Figura 3 – Tabelas (com dados) referentes à 2FN

Proj		
Cod_proj	Tipo	Descrição
LSC001	Novo Desenv.	Sist. Estoque
PAG02	Mantenção	Sist. RH

Proj Emp			
Cod_proj	Cod_emp	Data_ini	Tempo
LSC001	2146	1/11/01	24
LSC001	3145	2/10/01	24
LSC001	6126	3/10/02	18
LSC001	1214	4/10/02	18
LSC001	8191	1/11/02	12
PAG02	8191	1/05/03	12
PAG02	4112	4/01/01	24
PAG02	6126	1/11/02	12

Emp			
Cod_emp	Nome	Categoria	Salário
2146	João	A1	4
3145	Silvio	A2	4
6126	José	B1	9
1214	Carlos	A2	4
8191	Mário	A1	4

Adaptada de: Heuser (2009).

O que é importante você observar nos dados da Figura 3:

- a tabela **Proj** não foi alterada;
- a tabela **ProjEmp** agora **não tem mais redundância**;
- a tabela **Emp** tem ainda um outro tipo de redundância, já que o atributo Salario depende da Categoria do empregado (ou seja, todos da categoria "A1" tem o salário "4", "B1" é "9" e "A2" é "4").

5. TERCEIRA FORMA NORMAL – 3FN

Uma tabela encontra-se na 3FN quando:

- além de estar na 2FN, todo atributo não chave depende diretamente da chave primária, isto é, quando não há dependências funcionais transitivas ou indiretas.

Uma **dependência funcional transitiva** ou **indireta** ocorre quando um **atributo**, além de depender da chave primária completa, **depende de outro atributo ou combinação de atributos não chaves**.

Para transformarmos um banco de dados para a 3FN, pegamos as relações na 2FN e aplicamos a regra da 3FN.

Tabelas na 2FN (que já vimos):

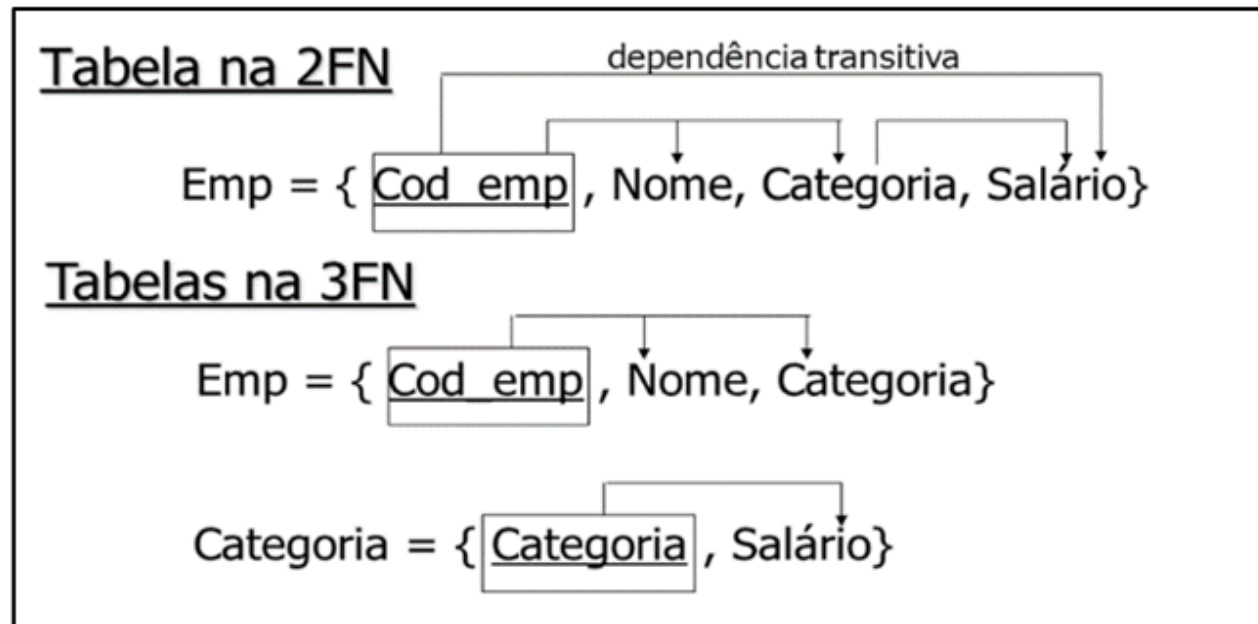
Proj = {Cod_proj, Tipo, Descrição}

Emp = {Cod_emp, Nome, Categoria, Salário}

ProjEmp = {Cod_proj, Cod_emp, Data_ini, Tempo}

O processo de passagem da 2FN para a 3FN é o seguinte:

- copiar para a 3FN cada tabela que tenha menos que dois atributos não chave, pois, nesse caso, não há como ter dependências transitivas;
- para tabelas com dois ou mais atributos não chave, criar uma tabela na 3FN com a chave primária da tabela em questão e, para cada atributo não chave, fazer a seguinte pergunta: **“o atributo depende de algum outro atributo não chave?”**



1. Caso o atributo dependa apenas da chave, copie o atributo para a tabela na 3FN;
2. caso o **atributo dependa de outro atributo**, criar (caso ainda não exista) uma tabela na 3FN que tenha como chave primária o atributo do qual há a dependência transitiva, copiar o atributo dependente para a tabela criada e o atributo determinante deve permanecer também na tabela original.

Assim, o **modelo relacional correspondente à 3FN** é o seguinte:

Proj = { Cod_proj, Tipo, Descrição }

Categoria = { Categoria, Salário }

Emp = { Cod_emp, Nome, Categoria }

ProjEmp = { Cod_proj, Cod_emp, Data_ini, Tempo }

O conteúdo das tabelas na 3FN, considerando os dados que tínhamos na 2FN – Figura 3, ficaria da seguinte forma:

Figura 4 – Tabelas (com dados) referentes à 3FN

Proj			Categoria	
Cod_proj	Tipo	Descrição	Categoria	Salário
LSC001	Novo Desenv.	Sist. Estoque	A1	4
PAG02	Manutenção	Sist. RH	A2	4
			B1	9

Proj_Emp				Emp		
Cod_proj	Cod_emp	Data_ini	Tempo	Cod_emp	Nome	Categoria
LSC001	2146	1/11/01	24	2146	João	A1
LSC001	3145	2/10/01	24	3145	Silvio	A2
LSC001	6126	3/10/02	18	6126	José	B1
LSC001	1214	4/10/02	18	1214	Carlos	A2
LSC001	8191	1/11/02	12	8191	Mário	A1
PAG02	8191	1/05/03	12			
PAG02	4112	4/01/01	24			
PAG02	6126	1/11/02	12			

Adaptada de: Heuser (2009).

O que é importante você observar nos dados da Figura 4:

- as tabelas **Proj** e **ProjEmp** não foram alteradas;
- a tabela **Emp** não tem redundância, pois, sabendo-se a categoria que o empregado pertence, é possível obter seu salário por meio da tabela Categoria (que foi criada na 3FN).

Atenção: não deixe de assistir à videoaula “Normalização – 1FN, 2FN e 3FN”, com a professora Elisângela Botelho Gracias, com uma explicação sobre a 1FN, 2FN e 3FN.

Na Figura 5, temos um resumo da 1FN, 2FN e 3FN.

Figura 5 – Resumo das formas normais 1FN, 2FN e 3FN

Forma normal	Teste	Solução (normalização)
Primeira (1FN)	Relação não deve ter atributos multivalorados ou relações aninhadas.	Formar novas relações para cada atributo multivalorado ou relação aninhada.
Segunda (2FN)	Para relações em que a chave primária contém múltiplos atributos, nenhum atributo não chave deverá ser funcionalmente dependente de uma parte da chave primária.	Decompor e montar uma nova relação para cada chave parcial com seu(s) atributo(s) dependente(s). Certificar-se de manter uma relação com a chave primária original e quaisquer atributos que sejam total e funcionalmente dependentes dela.
Terceira (3FN)	A relação não deve ter um atributo não chave determinado funcionalmente por outro atributo não chave (ou por um conjunto de atributos não chave). Ou seja, não deve haver dependência transitiva de um atributo não chave sobre a chave primária.	Decompor e montar uma relação que inclua o(s) atributo(s) não chave que determina(m) funcionalmente outro(s) atributo(s) não chave.

Fonte: Elmasri; Navathe (2018).

6. REFERÊNCIAS

ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. 7. ed. São Paulo: Pearson, 2018.

HEUSER, A. C. *Projeto de banco de dados*. 6. ed. Porto Alegre: Bookman, 2009.