

N

ormalização

Técnicas e Conceitos

Bráulio Ferreira de Carvalho

Na minha experiência profissional tenho observado, ao longo dos anos, como a normalização é tratada com pouca atenção. Já encontrei muitas tabelas sem chave primária e com redundância de informações, além de projetos cheios de anomalias de atualização.

O pior é que muitas vezes, ao averiguar, me deparei com “justificativas técnicas” – na maior parte dos casos infundadas. Há pouco me surpreendi com o comentário de um aluno - um analista havia lhe “ensinado” que a normalização está desaparecendo! Só se sumirem juntos os bancos de dados relacionais e as boas práticas de desenvolvimento.

A normalização tem dois objetivos principais: i) garantir a integridade dos dados, evitando que informações sem sentido sejam inseridas; ii) organizar e dividir as tabelas da forma mais eficiente possível, diminuindo a redundância e permitindo a evolução do banco de dados com o mínimo de efeito colateral. Esses objetivos são atingidos através da implementação de regras conhecidas como **formas normais**.

São seis as formas normais mais conhecidas:

1FN (1^a Forma Normal)

2FN (2^a Forma Normal)

3FN (3^a Forma Normal)

FNBC (Forma Normal de Boyce e Codd)

4FN (4^a Forma Normal)

5FN (5^a Forma Normal)

Uma forma normal engloba todas as anteriores, ou seja, para que uma tabela esteja na 2FN, ela obrigatoriamente deve estar na 1FN e assim por diante (**figura 1**).

Vale ressaltar que as três primeiras formas normais atendem à maioria dos casos de normalização. Neste artigo seguiremos passo a passo até a terceira forma normal - a quarta, quinta e outras formas normais pouco conhecidas serão apresentadas em outra edição.

Conceitos Úteis

Entender o significado dos conceitos a seguir pode ajudá-lo na leitura de textos e publicações acadêmicos, que muitas vezes são as melhores fontes de pesquisa para assuntos como normalização, modelagem e projeto de banco de dados.

Variáveis de relação

Os termos “variável de relação” e “relação”, de um modo simplista, podem ser interpretados como o equivalente matemático para “tabela”.

Chaves

Chave candidata – É um atributo ou conjunto de atributos que são únicos para cada registro. Para cada tabela podemos ter uma ou várias chaves desse tipo. Por exemplo, uma tabela de clientes pode possuir duas chaves candidatas: *Código* e *CPF*.

Chave primária – Entre as chaves candidatas, escolhemos uma para ser o identificador principal da tabela – o atributo escolhido passa então a ser chamado de chave primária. Em alguns bancos de dados a chave primária é referenciada como PK (do inglês, Primary Key).

Chaves alternativas – São as chaves candidatas que não foram definidas como chave primária.

Chave estrangeira - É o atributo ou conjunto de atributos que faz a ligação com uma chave candidata de outra tabela. Muitas vezes a chave estrangeira é referenciada como FK (do inglês, Foreign Key).

Dependência Funcional (DF)

Sempre que um atributo X identifica um atributo Y, dizemos que entre eles há uma **dependência funcional**. Temos, portanto, que X é o **determinante** e que Y é o **dependente**. A representação é: $X \rightarrow Y$ (lê-se X determina Y ou Y é dependente de X). Veja um exemplo:

Dada uma determinada cidade (não considerando cidades homônimas) sabemos o seu estado e com o estado temos o país. Isso



é representado da seguinte forma:

cidade → estado
estado → país

Em outras palavras, estado é funcionalmente dependente de cidade e país é funcionalmente dependente de estado. Ou ainda, estado é dependente de cidade e país é dependente de estado. E por último, cidade determina estado e estado determina país.

Trivialidade

A **dependência funcional trivial** indica que um determinante com mais de um atributo pode determinar seus próprios membros quando isolados. Veja um exemplo:

{banco, agência} → banco

(DF trivial, pois banco é parte do determinante)

{banco, agência} → agência

(DF trivial, pois agência é parte do determinante)

Quando um determinante identifica outro atributo qualquer, temos uma **dependência funcional não trivial** (essa DF é a que nos interessa no processo de normalização):

{banco, agência} → cidade

(DF não trivial, pois cidade não faz parte do determinante)

Transitividade

Se um atributo X determina Y e se Y determina Z, podemos dizer que X determina

Z de forma transitiva. Outra leitura é: existe uma dependência funcional transitiva de X para Z. Veja um exemplo:

cidade → estado

estado → país

cidade → país (cidade determina país **de forma transitiva**)

DF Irreduzível à esquerda

Dizemos que o lado esquerdo de uma dependência funcional é irreduzível quando o determinante está em sua forma mínima. Temos a forma mínima quando não é possível reduzir a quantidade de atributos determinantes sem perder a dependência funcional. Exemplo:

{cidade, estado} → país

Esta DF não está na forma irreduzível à esquerda, pois podemos ter somente o estado como determinante:

estado → país

Nota: Nem sempre estar na forma irreduzível à esquerda significa possuir um determinante com apenas uma coluna.

Anomalias de Atualização

São os problemas causados quando efetuamos uma inserção, atualização ou deleção em uma base de dados não normalizada. Como exemplo, observe a tabela da figura 2 e as considerações a seguir:

■ A tabela mistura dados do fornecedor,

dados do fornecimento e dados de localização do fornecedor;

- Não existem tabelas auxiliares – todas as informações estão armazenadas nessa tabela;

- Nenhum campo pode ser nulo;

- A chave primária é composta pelos campos CódFornecedor e CódPeça.

Com base nisso, podemos identificar anomalias para cada operação DML:

Anomalias no comando INSERT - Não podemos inserir a localização de um fornecedor até que ele forneça pelo menos uma peça. De fato, a tabela não mostra que o fornecedor F5 está localizado em Atenas (ele existe na vida real, mas não pode ser representado).

Ainda temos outro problema: sempre que cadastrarmos o fornecimento de uma peça teremos que repetir os dados do fornecedor (campos Nome, Status e Cidade).

Anomalias no comando UPDATE - O valor do campo Cidade aparece repetido várias vezes na tabela. Essa redundância aumenta a probabilidade de erros durante uma atualização. Por exemplo, se o fornecedor F1 se mudar de Londres para Amsterdã, teremos que pesquisar todos os registros correspondentes e atualizá-los, tomando o cuidado de não deixar os dados inconsistentes (uma ocorrência de F1 pode receber Amsterdã e outra Londres). Observe que o mesmo problema se aplica aos campos Nome, Status e Preço.

• CódFornecedor	Nome	Status	Cidade	• CódPeça	Preço	Qtde	Valor	F2
F1	Fornecedor 1	20	Londres	P1	R\$ 10,00	300	R\$ 3.000,00	
F1	Fornecedor 1	20	Londres	P2	R\$ 20,00	200	R\$ 4.000,00	
F1	Fornecedor 1	20	Londres	P3	R\$ 15,00	400	R\$ 6.000,00	
F1	Fornecedor 1	20	Londres	P4	R\$ 25,00	200	R\$ 5.000,00	
F1	Fornecedor 1	20	Londres	P5	R\$ 12,00	100	R\$ 1.200,00	
F1	Fornecedor 1	20	Londres	P6	R\$ 5,00	100	R\$ 500,00	
F2	Fornecedor 2	10	Paris	P1	R\$ 10,00	300	R\$ 3.000,00	
F2	Fornecedor 2	10	Paris	P2	R\$ 20,00	400	R\$ 8.000,00	
F3	Fornecedor 3	10	Paris	P2	R\$ 20,00	200	R\$ 4.000,00	
F4	Fornecedor 4	20	Londres	P2	R\$ 20,00	200	R\$ 4.000,00	
F4	Fornecedor 4	20	Londres	P4	R\$ 25,00	300	R\$ 7.500,00	
F4	Fornecedor 4	20	Londres	P5	R\$ 12,00	400	R\$ 4.800,00	

Tabela Original

Funcionário	Sexo	Salário
Carlos	M	5.000,00
Marcos	M	4.500,00

O campo *valor* representa outra anomalia: ele é calculado pela multiplicação de *preço* e *qtde*. Se alterarmos o preço unitário ou a quantidade, teremos que atualizar também o campo *valor* - um passo a mais que pode ser esquecido.

Anomalias no comando DELETE - Se eliminarmos todos os registros de um fornecedor, apagaremos não apenas a ligação com as peças, mas também a informação de que o fornecedor está localizado em uma determinada cidade (os problemas de INSERT e DELETE são na realidade duas faces da mesma moeda).

Decomposição sem perdas

O processo de normalização envolve a quebra ou decomposição de uma tabela em partes menores. Essa decomposição tem que ser reversível, de modo que nenhuma informação seja perdida no processo. O fato de uma decomposição ser ou não sem perdas está intimamente ligado ao conceito de dependência funcional visto anteriormente. Veja as considerações abaixo, com base na figura 3:

- Na decomposição da letra **a** é possível restaurar a tabela original, pois com o nome do funcionário obtemos o sexo e o salário correspondentes (decomposição sem perdas);
 - Na letra **b** não é possível obter a tabela

F4

Tabelas na Decomposição

	Funcionário	Sexo	Funcionário	Salário
a)	Carlos	M	Carlos	5.000,00
	Funcionário	Sexo	Funcionário	Salário
b)	Carlos	M	M	5.000,00
	Marcos	M	M	4.500,00

original, pois para o funcionário “Carlos”, de sexo “M”, encontraremos dois salários na segunda tabela.

Junção (Join)

Junção é o processo inverso da decomposição sem perdas - veja letra a na figura 3. Como temos um atributo comum (*Funcionario*), podemos obter a tabela original fazendo a junção das duas tabelas decompostas.

Projeções

Significa considerar apenas alguns campos de uma tabela. Na **figura 4** vemos uma projeção que inclui os campos *a,b,d* e *f*.

Importância da Normalização

Os benefícios mais importantes obtidos com a normalização de uma base de dados são:

Tabelas flexíveis e de fácil manutenção - Uma base sem as anomalias de atualização e com uma estrutura de armazenamento eficaz é mais simples de atualizar e evoluir (embora o processo de consulta se torne mais complexo, pois a normalização geralmente aumenta o número de tabelas do banco).

Eliminação de redundâncias – Sem redundâncias, as tabelas ficam menores, mais consistentes e menos sujeitas a discrepâncias. Sempre que puder, evite o que o mercado chama de “redundância controlada” – como o nome já diz, ela cria a necessidade de controle adicional e aumenta a complexidade.

Passo a passo em normalização

Para efeitos didáticos, vamos trabalhar com o seguinte exemplo: em uma determinada empresa, os produtos recebidos de um fornecedor são registrados em um formulário próprio, que pode ser visualizado na figura 5. Vamos informatizar esse processo

F3

criando uma base de dados para armazenar as informações deste formulário.

A primeira versão da base pode ser vista na figura 6. Essa estrutura ainda não pode ser carregada em um banco de dados relacional, pois possui campos multivvalorados (*CodItem*, *CodProd*, *Produto*, *Qtde*, *Preço*, *Total Item*) e não se apresenta no formato de tabela.

1^a Forma Normal (1FN)

O primeiro passo para normalizar a estrutura da **figura 6** é aplicar as regras da primeira forma normal. Podemos dizer que uma entidade está na primeira forma normal quando cada atributo contém somente um valor, em somente um lugar. Essa exigência também é conhecida como **atomicidade de dados**. As regras gerais para obtenção da 1FN são:

- Não podemos ter atributos multivaleados. Nesse caso, colocamos cada valor do atributo em uma linha diferente e repetimos os dados de todas as outras colunas;
 - Não podemos ter atributos repetidos, como Telefone1, Telefone2 etc. A solução é semelhante ao item anterior;
 - Todos os registros têm que ser diferentes;
 - A entidade não pode ter mais de duas dimensões;
 - Cada atributo deve ter somente um tipo de dado. Uma violação comum dessa regra, por exemplo, é a criação de um campo para armazenar o CPF e o CNPJ, alternadamente. Esse cenário deve ser evitado pois cria complicações para a evolução da regra de negócio.

Observe na figura 7 a tabela gerada após a aplicacão da 1FN.

Chave Primária

Para trabalhar com a 2FN precisamos definir uma chave primária. Na tabela recém-criada, a chave escolhida é formada pelos campos *Nro_nota* e *CodItem* (pintados de cinza), pois através deles determinamos todos os outros campos. Por exemplo:

Com o número da nota, determinamos os campos *Data*, *CodForn*, *Nome*, *Telefone* e *Endereço*:

NroNota → {Data, CodForn, Nome, Telefone, Endereço}

Com o número da nota e o código do item determinamos os demais campos:

{**NroNota, CodItem**} → {CodProd, Produto, Quantidade, Preço, TotalItem}

2ª Forma Normal (2FN)

Essa forma normal visa a diminuição da redundância e o desagrupamento de informações. Com a 2FN, uma tabela passa a representar uma quantidade menor de entidades (o ideal é que cada entidade seja armazenada em apenas uma tabela) - observe que a tabela da figura 7 agrupa as entidades *Nota Fiscal*, *Item da Nota*, *Fornecedor* e *Produto*.

A definição da segunda forma normal é: uma tabela está em 2FN se estiver em 1FN e todo atributo não-chave for determinado por **todos** os campos da chave primária. Em outras palavras, é necessário eliminar as dependências funcionais parciais.

A tabela do exemplo viola a 2FN pois os campos *Data*, *CodForn*, *Nome*, *Telefone* e *Endereço* não são determinados pela chave primária **completa** (o campo *CodItem* não é necessário para identificar essas informações):

NroNota → {Data, CodForn, Nome, Telefone, Endereço}

Como regra geral, a 2FN deve ser aplicada através dos passos:

1) Eleger a chave primária da tabela;

2) Verificar as dependências funcionais parciais;

3) Mover os campos não enquadrados na 2FN para uma nova tabela, fazendo a decomposição sem perdas;

4) Na tabela criada, repetir os passos 1 à 4 até eliminar a DF parcial.

O resultado pode ser visualizado na figura 8. Observe que todos os campos, nas duas tabelas, são agora determinados por suas chaves primárias **completas** – garantindo a 2FN. Note também que o resultado da aplicação desta forma normal são tabelas mais simples, que representam as entidades com mais proximidade.

Nota: Se a chave primária possui apenas um

Nota de Fornecimento de Mercadoria					
Data		15/06/03		Nº da nota	
Fornecedor					
CódForm	Nome		Telefone	Endereço	
22	Empresa Excalibur		031 3335-5255	Rua Itamaracá, nº 15	
Item					
CodItem	CodProd	Produto	Quantidade	Preço	Total Item
1	CA	Chapa de aço	35	15,00	525,00
2	BB	Bobina	20	15,00	300,00
3	TC	Tábuas Corridas	50	20,00	1.000,00
Nota de Fornecimento de Mercadoria					
Data		20/07/03		Nº da nota	
Fornecedor					
CódForm	Nome		Telefone	Endereço	
17	Cia Silva		031 3334 4787	Rua Cardoso, nº 145	
Item					
CodItem	CodProd	Produto	Quantidade	Preço	Total Item
1	TC	Tábuas Corridas	50	20,00	1.000,00
2	CM	Compensado	30	15,00	450,00
3	RM	Ripas de Madeira	300	2,00	600,00
Nota de Fornecimento de Mercadoria					
Data		22/08/03		Nº da nota	
Fornecedor					
CódForm	Nome		Telefone	Endereço	
22	Empresa Excalibur		031 3335 5255	Rua Itamaracá, nº 15	
Item					
CodItem	CodProd	Produto	Quantidade	Preço	Total Item
1	CA	Chapa de aço	50	15,00	750,00
2	BB	Bobina	20	15,00	300,00

campo ou é composta por todos os campos, a tabela já está automaticamente na 2FN.

3ª Forma Normal (3FN)

A 3FN dá continuidade ao objetivo da 2FN: reduzir as redundâncias, desagrupando as tabelas de forma que cada uma represente apenas uma entidade. Na figura 8, a tabela 1 agrupa informações sobre as

entidades *Nota* e *Fornecedor* e a tabela 2 agrupa informações sobre as entidades *Item Nota* e *Produto*.

A técnica utilizada pela 3FN é a identificação e eliminação da **transitividade**. Dizemos que uma tabela está na 3FN se também estiver na 2FN e todo atributo não chave for determinado de forma não transitiva pela chave primária. Em outra leitura, dizemos

NroNota	Data	CodForn	Nome	Telefone	Endereço	CodItem	CodProd	Produto	Qtde	Preço	TotalItem
1	15/6/2003	22	Empresa Excalibur	3335-5255	Rua Itamaraca 15	1	CA	Chapa de Aço	35	15	525
						2	BB	Bobina	20	15	300
						3	TC	Tábuas Corridas	50	20	1000
2	20/7/2003	17	Cia Silva	3334-4787	Rua Cardoso, 145	1	TC	Tabuas Corridas	50	20	1000
						2	CM	Compensado	30	15	450
						3	RM	Ripas de Madeira	300	2	600
3	22/8/2003	22	Empresa Excalibur	3335-5255	Rua Itamaracá, 15	1	CA	Chapa de Aço	50	15	750
						2	BB	Bobina	20	15	300

NroNota	Data	CodForn	Nome	Telefone	Endereço	CodItem	CodProd	Produto	Qtde	Preço	TotalItem
1	15/6/2003	22	Empresa Excalibur	3335-5255	Rua Itamaraca 15	1	CA	Chapa de Aço	35	15	525
1	15/6/2003	22	Empresa Excalibur	3335-5255	Rua Itamaraca 15	2	BB	Bobina	20	15	300
1	15/6/2003	22	Empresa Excalibur	3335-5255	Rua Itamaraca 15	3	TC	Tábuas Corridas	50	20	1000
2	20/7/2003	17	Cia Silva	3334-4787	Rua Cardoso, 145	1	TC	Tabuas Corridas	50	20	1000
2	20/7/2003	17	Cia Silva	3334-4787	Rua Cardoso, 145	2	CM	Compensado	30	15	450
2	20/7/2003	17	Cia Silva	3334-4787	Rua Cardoso, 145	3	RM	Ripas de Madeira	300	2	600
3	22/8/2003	22	Empresa Excalibur	3335-5255	Rua Itamaracá, 15	1	CA	Chapa de Aço	50	15	750
3	22/8/2003	22	Empresa Excalibur	3335-5255	Rua Itamaracá, 15	2	BB	Bobina	20	15	300

Tabela Nova

NroNota	Data	CodForn	Nome	Telefone	Endereço
1	15/6/2003	22	Empresa Excalibur	3335-5255	Rua Itamaraca 15
2	20/7/2003	17	Cia Silva	3334-4787	Rua Cardoso, 145
3	22/8/2003	22	Empresa Excalibur	3335-5255	Rua Itamaraca 15

Tabela 2

NroNota	CodItem	CodProd	Produto	Qtde	Preço	TotalItem
1	1	CA	Chapa de Aço	35	15	525
1	2	BB	Bobina	20	15	300
1	3	TC	Tábuas Corridas	50	20	1000
2	1	TC	Tabuas Corridas	50	20	1000
2	2	CM	Compensado	30	15	450
2	3	RM	Ripas de Madeira	300	2	600
3	1	CA	Chapa de Aço	50	15	750
3	2	BB	Bobina	20	15	300

que todo atributo não chave deve ser determinado **somente** pela chave primária.

Vamos analisar a tabela criada na 2FN. Observe que os campos *Nome*, *Telefone* e *Endereço* podem ser determinados tanto pela chave primária quanto pelo campo *CodForn*:

$$\{ \text{NroNota} \} \rightarrow \{ \text{Data}, \text{CodForn}, \text{Nome}, \text{Telefone}, \text{Endereço} \}$$

$$\{ \text{CodForn} \} \rightarrow \{ \text{Nome}, \text{Telefone}, \text{Endereço} \}$$

Assim, esses campos possuem **dependência funcional transitiva** com a chave primária:

$$\{ \text{NroNota} \} \rightarrow \text{CodForn} \rightarrow \{ \text{Nome}, \text{Telefone}, \text{Endereço} \}$$

Na segunda tabela também temos transitividade:

$$\{ \text{NroNota} \} \rightarrow \text{CodProd} \rightarrow \{ \text{Produto}, \text{Preço} \}$$

Outro tipo de violação da 3FN são os campos calculados, que também possuem transitividade. Na 3FN, todos os campos calculados são removidos da base de dados.

Veja a representação:

$$\{ \text{NroNota}, \text{CodItem} \} \xrightarrow{\text{Preço, Quantidade}} \{ \text{TotalItem} \}$$

Para adequar as tabelas à 3FN, seguimos um roteiro semelhante ao usado na 2FN:

1) Mover os campos com transitividade para uma nova tabela;

2) Criar uma chave primária na tabela nova com o(s) campo(s) da tabela original que determinava(m) diretamente os campos movidos.

3) Na nova tabela, repetir os passos 1, 2 e 3, até eliminar totalmente a transitividade.

Observe na figura 9 o formato final das tabelas, conseguido após a aplicação da 3FN. Nesse ponto temos a organização ideal para a base de dados, pelos motivos a seguir:

1) A decomposição foi feita sem perdas – através de junções, podemos recuperar a tabela da figura 7;

2) As quatro entidades (*Nota*, *Item Nota*, *Fornecedor* e *Produto*) possuem tabelas exclusivas, eliminando o agrupamento de informações e a redundância;

3) As tabelas foram separadas de tal forma que as anomalias de atualização não poderão ocorrer;

4) As tabelas são fáceis de evoluir e manter. Por exemplo, se quisermos incluir os dados

de um produto que ainda não tenha sido fornecido, podemos inserir sua descrição na tabela *Produtos*. Observe que isso não era possível até a aplicação da 3FN;

5) Do ponto de vista relacional, os dados estão armazenados e distribuídos de forma eficiente.

Integridade

As formas normais visam a consistência da base de dados sob o aspecto da eliminação de redundâncias, mas não garantem que as informações certas serão inseridas no **lugar certo** – um ponto crucial para que o banco se mantenha íntegro ao longo do tempo. Para que as tabelas obtidas com a aplicação das formas normais façam sentido, é preciso existir **regras de integridade**. Veja alguns exemplos:

1) Não podemos inserir um produto na tabela *Item Nota* que não tenha sido cadastrado na tabela *Produto*;

2) Não podemos apagar um registro na tabela *Nota* sem apagar o conjunto de registros correspondentes da tabela *Item Nota*;

3) Não podemos alterar o código de um fornecedor sem alterar todos os registros da tabela *Nota*, que o referenciam.

Basicamente, existem quatro tipos de regras de integridade: de entidade, de domínio, referencial e definida pelo usuário – elas serão detalhadas nos próximos números da SQL Magazine.

Desnormalização

A normalização não se preocupa com a performance de obtenção dos dados e sim com a melhor forma de organizá-los. Como a normalização geralmente aumenta o número de tabelas, há uma tendência de queda de performance nas consultas – que passam a necessitar de mais *joins* para que sejam efetuadas.

A desnormalização é o processo inverso, onde o administrador do banco abre mão de algumas regras das formas normais com o objetivo de otimizar as consultas. É importante notar que isso não é o mesmo que ter uma base de dados não-normalizada: a desnormalização acontece sempre *depois* da aplicação das formas normais.

A desnormalização é, na maioria dos casos, desaconselhável e deve ser aplicada somente quando o administrador tiver certeza de que ela trará benefícios **reais**. Alguns profissionais fazem suposições sobre a perda de performance e já projetam desnormalizando, sem fazer nenhum *benchmark*. O problema disso é que em muitos casos o ganho de velocidade não surge efetivamente ou não paga o custo das anomalias de atualização geradas pela desnormalização. Vale notar que nem sempre os *joins* degradam performance, pois a maioria dos SGBDs mantém o conteúdo de tabelas pequenas em *cache* e disponibilizam diversos outros recursos para a otimização de consultas.

Como exemplo, imagine que nossa aplicação acesse massivamente um relatório contendo o campo calculado *TotalItem* (eliminado na 3FN). Para evitar o cálculo do total a todo momento, podemos abrir mão da 3FN e criar o campo fisicamente na tabela de itens de nota, arcando com a responsabilidade de mantê-lo devidamente

atualizado. Observe que a idéia e a justificativa são questionáveis – a desnormalização deve ser aplicada somente se o administrador tiver **provas** de que o ganho de performance será real e trará benefícios.

Recomendo a leitura do artigo publicado por Craig Mullins em <http://www.tdan.com/i001fe02.htm> - o especialista mostra diversos casos comuns no uso da desnormalização.

Algumas dúvidas comuns sobre a desnormalização:

O que dizer quando a desnormalização é utilizada para fins históricos? No exemplo citado, se o preço de um produto for alterado, todos os dados de fornecimento serão modificados. Repetir o campo preço na tabela *Item_Nota*, para manter o histórico, é uma boa prática?

Este caso não se trata de desnormalização. O que temos são dois atributos com sutil diferença: *Preço atual* e *Preço histórico*.

F9

Tabela Fornecedor

o-- CodForn	Nome	Telefone	Endereço
22	Empresa Excalibur	3335-5255	Rua Itamaraca, 15
17	Cia Silva	3334-4787	Rua Cardoso, 145

Tabela Produto

o-- CodProd	Produto	Preço
CA	Chapa de Aço	15,00
BB	Bobina	15,00
TC	Tábuas Corridas	20,00
CM	Compensado	15,00
RM	Ripas de Madeira	2,00

Tabela Nota

o-- NroNota	CodForn
1	22
2	17

Tabela Item Nota

o-- NroNota	o-- CodItem	CodProd	Qtde
1	1	CA	35
1	2	BB	20
1	3	TC	50
2	1	TC	50
2	2	CM	30
2	3	RM	300

Atributos Multivalorados

O objetivo principal da 1FN é transformar estruturas não padronizadas em tabelas, para que possam ser armazenadas em um banco de dados relacional. De fato, podemos dizer que toda tabela já está na 1FN; caso contrário, ela não seria uma tabela.

O caso mais comum de violação da 1FN são estruturas que possuem atributos multivalorados, repetidos ou aninhados. Na 1FN, esses atributos devem ser “planificados”, fazendo com que cada célula armazene somente um valor. Existem duas formas de enquadrar os atributos multivalorados na 1FN:

1) Expandir o atributo multivalorado em vários registros, repetindo o valor dos demais campos – essa foi a solução escolhida neste artigo e é a que mais se aproxima da definição criada por Codd, o pai da normalização.

2) Criar uma tabela auxiliar para conter o atributo multivalorado – essa opção é adotada por alguns autores pela praticidade. No entanto, em alguns casos, essa abordagem pode induzir a erros. Por exemplo, observe o aninhamento de três tabelas na figura 10 (vários gostos dentro de dependentes e vários dependentes dentro de pai).

Utilizando a primeira abordagem chegamos facilmente à estrutura da figura 11. Agora, imagine a dificuldade em criar diretamente as tabelas auxiliares. Teremos que verificar quem é a chave, se a tabela auxiliar terá outra chave, se precisará de uma tabela “auxiliar da auxiliar” etc. Em suma, será necessário utilizar os conceitos de chave primária, dependência funcional, dependência funcional irredutível à esquerda e decomposição sem perdas, que são utilizados na 2FN. Certamente, nesse caso, projetistas iniciantes podem cometer erros com muito mais facilidade.

Devemos sempre normalizar até o fim? Por exemplo, geralmente os campos *cidade*, *bairro*, *UF* e *CEP* apresentam repetição – é correto criarmos tabelas auxiliares para cada um?

Depende. Se a aplicação realizar pesquisas nestes campos, sim – imagine se quisermos saber a quantidade de pessoas em um bairro e este estiver cadastrado de forma diferente em mais de um registro (ex.: Sto Antônio em um registro e "Santo Antonio" em outro). Tal como no site dos Correios, hoje vemos vários sistemas que a partir do CEP trazem todas as demais informações do endereço, deixando claro que várias tabelas foram utilizadas.

Conclusão

A normalização, apesar de extremamente útil no projeto de um banco de dados, não é remédio para todos os males, já que nem todas as redundâncias ou anomalias de atualização podem ser eliminadas através das formas normais. Além disso, a normalização não deve substituir uma boa análise do negócio da aplicação – ela é apenas uma ferramenta de apoio para o projetista. ■

Pai			F10			
código	nome	telefone	Dependente			idade
			nome	gosto	idade	
1	José	345-5456	Pedro	futebol	10	45
			Lucas	futebol	8	
				luta		
			Maria	flores	6	
				colorir		
				dança		
2	Antônio	434-4554	Marcos	games	9	37
				xadrez		
			Marta	brincos	7	
				colares		
				chocolate		
				brinquedos		

código pai	nome pai	telefone pai	nome dependente	gosto dependente	idade dependente	idade pai
1	José	345-5456	Pedro	futebol	10	45
1	José	345-5457	Lucas	futebol	8	45
1	José	345-5458	Lucas	luta	8	45
1	José	345-5459	Maria	flores	6	45
1	José	345-5460	Maria	colorir	6	45
1	José	345-5461	Maria	dança	6	45
2	Antônio	434-4554	Marcos	games	9	37
2	Antônio	434-4555	Marcos	xadrez	9	37
2	Antônio	434-4556	Marta	brincos	7	37
2	Antônio	434-4557	Marta	colares	7	37
2	Antônio	434-4558	Marta	chocolate	7	37
2	Antônio	434-4559	Marta	brinquedos	7	37

R C.J.DATE, **Introdução a Sistemas de Banco de Dados** – 7ª edição americana

Henry F. Korth e Abraham Silberschatz, **Sistema de banco de Dados** – 2ª edição

Carlos Alberto Heuser, **Projeto de Banco de Dados** – 4ª edição

Michael J. Hernandez, **Aprenda a Projetar seu Próprio Banco de Dados**

A O Prof. Bráulio Ferreira de Carvalho (braulio@cic.unb.br ou brauliof@stf.gov.br) ministra as disciplinas “Banco de Dados” e “Organização de Arquivos” para a graduação em Licenciatura da Computação da Universidade de Brasília - UNB. Atua também na área de Administração de Dados e Metodologia de Sistemas na Secretaria de Informática do STF - Supremo Tribunal Federal, além de ministrar treinamentos e consultorias na área de desenvolvimento de sistemas de modo geral.

ormas

normais superiores

Bráulio Ferreira de Carvalho

Na edição anterior, analisamos alguns conceitos importantes à compreensão do processo de normalização e exploramos as três primeiras formas normais (FNs). Neste artigo, daremos continuidade ao assunto e abordaremos os seguintes temas:

FNBC (Forma Normal de Boyce e Codd):

4FN (4ª Forma Normal);
5FN (5ª Forma Normal).

Existem ainda duas outras formas normais, mas, por serem pouco práticas, falaremos brevemente sobre elas e indicaremos uma bibliografia para consulta posterior.

A exemplo do artigo anterior, antes de entrarmos propriamente nas formas normais, será necessário analisar alguns novos conceitos.

CONCEITOS ÚTEIS

Mais sobre chaves

- **Chaves simples:** são aquelas que contêm somente um atributo;
- **Chaves compostas:** são aquelas que contêm mais de um atributo;
- **Chaves superpostas:** dizemos que duas chaves são superpostas quando pelo menos uma delas é composta e entre elas existe pelo menos um atributo em comum;
- **Chaves disjuntas:** dizemos que duas chaves são disjuntas se entre elas não existe superposição.

Exemplo: considere a tabela da **Figura 1**. Nela, temos várias chaves candidatas: A - CodGeral; B - Semestre+CodDisc; C - Semestre+Disciplina. Observe os tipos de chave desta tabela:

- **Chaves simples:** somente **A**;
- **Chaves compostas:** **B** e **C**;
- **Chaves superpostas:** **B** é sobreposta a **C**, pois o campo *semestre* está nas duas chaves;
- **Chaves disjuntas:** **A** é disjunta a **B** e **A** é disjunta a **C**.

Relacionamentos binários

- Conceito utilizado na etapa de projeto de banco de dados. Indica que uma entidade se relaciona de alguma forma com outra entidade. Representa, portanto, a forma como **duas** entidades se relacionam.

Relacionamentos ternários

- Indica que uma entidade se relaciona de alguma forma com duas outras entidades. Representa, portanto, a forma como **três** entidades se relacionam.

Nota: as formas normais a seguir se aplicam somente a tabelas com chaves compostas. Portanto, se ao normalizar até a 3FN você obtiver tabelas com apenas chave simples, elas já estarão na FNBC, 4FN e 5FN.

Forma Normal de Boyce e Codd (FNBC)

No processo de normalização, essa forma normal deve ser aplicada às tabelas em 3FN que possuam mais de uma chave candidata (lembre-se de que a chave primária é também uma chave candidata), onde pelo menos uma delas seja composta e onde haja superposição entre elas. Mais adiante, veremos como essa FN pode ser utilizada para substituir as formas normais anteriores.

Nota: a 3FN e a FNBC são muito próximas e, normalmente, analisamos essas duas etapas de uma só vez.

Para simplificar, definimos que uma tabela está em FNBC se e somente se todos os determinantes são chaves candidatas.

Ocupação das salas para o horário de 18:00 às 20:40				
CodGeral	Semestre	CodDisc	Disciplina	Sala
1	1º 2003	6543	Banco de Dados	A
2	1º 2003	5722	Computação Básica	B
3	1º 2003	3544	Organização de Arquivos	C
4	2º 2003	6543	Banco de Dados	C
5	2º 2003	5722	Computação Básica	A
6	2º 2003	3544	Organização de Arquivos	B

Ou seja, se houver algum atributo que seja determinado por outro(s) atributo(s) que não é (sejam) uma chave candidata, não estamos na FNBC. A solução é levar esses atributos para outra tabela, utilizando o conceito de decomposição sem perdas.

Nota: tabelas na 3FN que não possuem superposição de chaves já estão na FNBC.

Vejamos um exemplo. Suponha que em um processo de normalização você chegue na tabela da **Figura 2**, que contém os dados de cliente, agência e gerente. As colunas da chave primária estão indicadas.

Observe que ela está na 3FN pois: i) todos os atributos são atômicos (1FN); ii) todos os atributos não-chave dependem totalmente da chave primária (2FN); iii) não existe transitividade em relação à chave primária (3FN).

Observe, entretanto, que existe anomalia de atualização, pois a agência está sendo repetida para o mesmo gerente. Essa anomalia só desaparecerá com a FNBC (observe que existe outra chave candidata e que ela pode ser sobreposta à chave primária):

Chave primária {Cliente, Agência} → {Gerente}	Há sobreposição pois o campo <i>Cliente</i> encontra-se nas duas chaves.
Chave candidata {Cliente, Gerente} → {Agência}	

Observamos também que existe um determinante que não é chave candidata:

~~Chave candidata~~ {Gerente} → {Agência}

Cliente	Agência	Gerente
Maria	Planalto	Pedro
Carlos	Planalto	Zélia
Estevão	Planalto	Pedro
Luis	Planalto	Zélia
Clara	Centro	Frances
Lucas	Centro	Marcos

F2

Cliente	Gerente
Maria	Pedro
Carlos	Zélia
Estevão	Pedro
Luis	Zélia
Clara	Frances
Lucas	Marcos

F3

CPF	Produto	Qtde	FormaPagto	TipoPagto
123456	Banana	10	Dinheiro	Espécie
123456	Alface	20	Cheque	Compensado
101010	Banana	5	Dinheiro	Espécie

F4

CPF	Produto	Qtde	FormaPagto	FormaPagto	TipoPagto
123456	Banana	10	Dinheiro	Cheque	Compensado
123456	Alface	20	Cheque	Dinheiro	Espécie
101010	Banana	5	Dinheiro		

F5

Para estarmos na FNBC, decomponemos a tabela conforme a **Figura 3** (o atributo determinado por um campo que não é chave candidata vai para outra tabela):

A FNBC substitui as FNs anteriores

Ao contrário das outras formas normais, a FNBC não exige que a tabela já esteja na forma normal anterior (3FN) para que seja aplicada. Ou seja, podemos ir de uma tabela não normalizada diretamente para a FNBC. Vamos constatar isso observando a tabela da **Figura 4**, que a princípio se encontra na 1FN:

Fazendo um levantamento das chaves candidatas, temos:

- {CPF, Produto} → {Qtde, FormaPagto, TipoPagto}

Fazendo um levantamento das dependências funcionais, temos:

- {CPF, Produto} → {Qtde}
- {CPF, Produto} → {FormaPagto}
- {CPF, Produto} → {TipoPagto}
- {FormaPagto} → {TipoPagto}

Nas etapas normais da normalização, veríamos que ela já está na 2FN e levaríamos, pela 3FN, *TipoPagto* para outra tabela, pois teríamos transitividade. Nesse caso, obteríamos as tabelas da **Figura 5**.

Como não temos mais de uma chave em nenhuma tabela, já estamos automaticamente na FNBC. Observe agora que, se aplicássemos a regra da FNBC diretamente na tabela ainda em 1FN, obteríamos o mesmo resultado, pois *FormaPagto* é determinante, mas não é uma chave candidata.

O interessante é que a regra da FNBC se aplica a todas as tabelas, independente de elas já estarem na 3FN ou 2FN. Ou seja, ela serve como um “atalho” para as formas normais anteriores.

Você pode se perguntar: “por que a 1FN, 2FN e 3FN, se podemos ir diretamente para a FNBC?” Veja algumas explicações para essa questão:

- Pela historicidade dos fatos (a FNBC surgiu depois);
- As três primeiras formas normais existem independentemente da FNBC (podemos desnormalizar, lembra?);
- As três primeiras formas normais são mais difundidas por serem mais fáceis de compreender;

F ormas Normais Superiores

Apartamento	Motorista		Veículo		
	Nome	Idade	Placa	Carro	Ano
101	Sr. Juca	55	GBD-2541	Siena	2003
	D. Marilda	50	GHJ-5488	Parati	2002
	Pedro	22			
102	Sr. Paulo	57	GDA-7677	Fusca	1990
	D. Clara	56			

F6

Apartamento	Nome	Idade	Placa	Carro	Ano
101	Sr. Juca	55	GBD-2541	Siena	2003
101	Sr. Juca	55	GHJ-5468	Parati	2002
101	D. Marilda	50	GBD-2541	Siena	2003
101	D. Marilda	50	GHJ-5468	Parati	2002
101	Pedro	22	GBD-2541	Siena	2003
101	Pedro	22	GHJ-5468	Parati	2002
102	Sr. Paulo	57	GDA-7677	Fusca	1990
102	D. Clara	56	GDA-7677	Fusca	1990

F7

Observe que, neste exemplo, as duas DMVs estão relacionadas entre si (*nome dependente* e *idade dependente* estão conectados). Entretanto, poderíamos ter em uma mesma tabela as seguintes DMVs:

DF: {CPF} $\rightarrow\!\!\!\rightarrow$ {Carro}

Pois temos vários carros para cada pessoa

DMV: {CPF} $\rightarrow\!\!\!\rightarrow$ {Nome Dependente}

Pois temos vários dependentes para cada pessoa

Nesse caso, as DMVs não possuem relação direta (*dependente* e *carro* não estão conectados) e são tidas como **DMVs independentes**.

4ª Forma Normal

Neste nível de normalização, lidaremos com o conceito de dependência multivalorada, que também pode trazer anomalias de atualização.

Nota: por causa da DMV, alguns autores confundem a 4FN com relacionamentos binários do tipo “muitos para muitos”. A diferença é que a DMV ocorre “dentro” de uma tabela, ao contrário dos relacionamentos.

A 4FN só é necessária se a tabela contiver:

- Um multideterminante que aponte para **mais de um** multidependente;
- Independência entre esses multidependentes. É como se uma tabela contivesse duas outras tabelas (ou mais) que não possuíssem relação entre si.

De uma forma simples, dizemos que uma tabela está em 4FN quando ela está em FNBC e não possui duas ou mais DMVs independentes (não considerando as DFs que, como vimos, também são DMVs) de um mesmo multideterminante.

Por exemplo, imagine um condomínio que deseja cadastrar os veículos e os motoristas que os dirigem (para cada apartamento).

Apartamento	Nome	Placa	Placa	Carro	Ano
101	Sr. Juca	GBD-2541			
101	Sr. Juca		GHJ-5468		
101	D. Marilda	GBD-2541			
101	D. Marilda		GHJ-5468		
101	Pedro	GBD-2541			
101	Pedro		GHJ-5468		
102	Sr. Paulo	GDA-7677			
102	D. Clara	GDA-7677			

Nome	Idade
Sr. Juca	55
D. Marilda	50
Pedro	22
Sr. Paulo	57
D. Clara	56

F8

4ª Forma Normal (4FN)

Para compreender a 4FN, precisamos do conceito de Dependência Multivalorada.

Dependência Multivalorada (DMV)

A Dependência Multivalorada é, na verdade, uma ampliação da Dependência Funcional. Na DF, o valor de um atributo determina o valor de outro atributo; na DMV, o valor de um atributo determina um conjunto de valores de outro atributo. Sendo assim, toda DF é uma DMV de apenas uma ocorrência; a recíproca não é verdadeira.

Assim como a DF é representada por $X \rightarrow Y$, a DMV é representada por $X \rightarrow\!\!\!\rightarrow Y$. Lemos que X multidetermina Y ou que Y é multidependente de X.

Veja os exemplos:

DF: {CPF} \rightarrow {Nome}

Pois temos somente um nome para cada CPF

DMV: {CPF} $\rightarrow\!\!\!\rightarrow$ {Nome Dependente} Pois temos vários dependentes para cada pessoa

{CPF} $\rightarrow\!\!\!\rightarrow$ {Idade Dependente} Pois temos vários dependentes para cada pessoa

Apartamento	Nome
101	Sr. Juca
101	D. Marilda
101	Pedro
102	Sr. Paulo
102	D. Clara

Apartamento	Placa
101	GBD-2541
101	GHJ-5468
102	GDA-7677

Nome	Idade
Sr. Juca	55
D. Marilda	50
Pedro	22
Sr. Paulo	57
D. Clara	56

Placa	Carro	Ano
GBD-2541	Siena	2003
GHJ-5468	Parati	2002
GDA-7677	Fusca	1990

Surgiram das DMVs

Considere que qualquer motorista do apartamento pode dirigir qualquer carro daquela moradia (isso cria uma independência entre veículos e motoristas). Um exemplo de dados seria o mostrado na **Figura 6**. Aplicando a 1FN, teríamos a tabela da **Figura 7**.

Ao normalizarmos até a Forma Normal de Boyce Codd, teremos as tabelas da **Figura 8**.

Apesar de já estar na FNBC, a primeira tabela continua com anomalias de atualização. Se o apartamento 101 comprar mais um carro, digamos de placa GAA-3433 (que deve ser inserido na terceira tabela), precisaremos inserir mais 3 linhas na primeira tabela, já que ele poderá ser dirigido tanto pelo Sr. Juca como por Pedro ou por D. Marilda. Se o apartamento 102 vender o carro, perderemos a informação de quem mora no referido apartamento. Como exercício, verifique as outras anomalias.

Estas anomalias existem nesta tabela pela presença de duas DMV independentes:

{Apartamento} $\rightarrow\!\!\!\rightarrow$ {Nome}
{Apartamento} $\rightarrow\!\!\!\rightarrow$ {Placa}

Ou ainda, representadas de uma forma muito usual em algumas bibliografias:

{Apartamento} $\rightarrow\!\!\!\rightarrow$ {Nome} | {Placa}

Observe que, neste exemplo, podemos ter vários motoristas em um apartamento e vários veículos para o mesmo apartamento. Como o motorista de um apartamento pode dirigir qualquer carro desse mesmo apartamento, temos uma independência entre carro e veículos (o que não seria verdade se cada um tivesse seu próprio carro). No entanto, observe que um motorista não pode dirigir o veículo de outro apartamento e, logicamente, um veículo não pode ser dirigido pelo motorista de outro apartamento. Isso evidencia a multidependência de motorista/veículo em relação a apartamento:



Para deixarmos a tabela na 4FN, devemos efetuar a decomposição sem perdas e levar cada DMV para uma tabela diferente.

Observe na **Figura 9** que já não temos mais as anomalias de atualização e que podemos voltar à tabela original se fizermos a junção de todas as tabelas.

Nota: se não partissemos do princípio que cada pessoa pudesse dirigir qualquer carro de sua residência, não teríamos DMVs independentes, pois o veículo teria relação direta com motorista. Essa relação direta nos impediria de decompor a tabela, pois causaria erros quando efetuássemos a junção.

Observação: note que a primeira tabela pode ser unida à terceira tabela e que a segunda tabela também pode ser unida à quarta tabela, sem problemas (gerando assim tabelas mais adequadas para o contexto do problema). Ficaríamos então com as tabelas da **Figura 10**.

Ou seja, a normalização nem sempre leva ao formato ideal, exigindo sempre uma análise do seu resultado.

5ª Forma Normal (5FN)

Para compreender a 5FN, precisamos analisar o conceito de Dependência de junção.

Dependência de Junção (DJ)

A DJ utiliza os conceitos de *decomposição sem perdas* e de *junção*. Na edição anterior, aprendemos a decompor uma tabela em duas outras, sem perda de informação. Aqui, ampliaremos

Apartamento	Nome	Idade
101	Sr. Juca	55
101	D. Marilda	50
101	Pedro	22
102	Sr. Paulo	57
102	D. Clara	56

Apartamento	Placa	Carro	Ano
101	GBD-2541	Siena	2003
101	GHJ-5468	Parati	2002
102	GDA-7677	Fusca	1990

Formas Normais Superiores

F11

Autorizada	Montadora	Espécie
JSV	FORD	TRATOR
JSV	FORD	CAMINHÃO
JSV	FORD	CARRO
JSV	FIAT	CARRO
JSV	FIAT	CAMINHÃO
GTC	FORD	CARRO



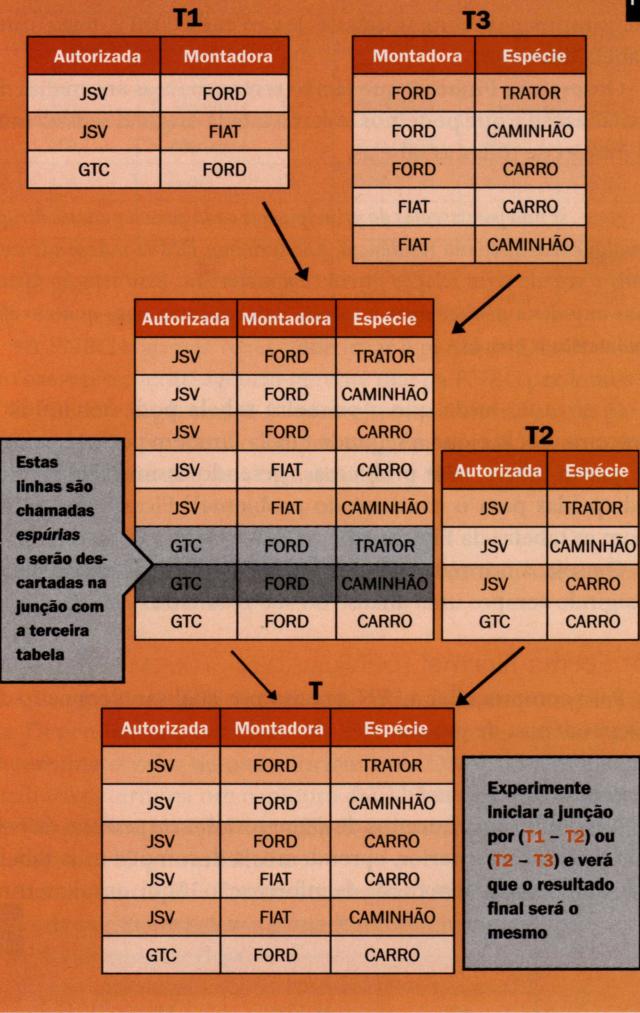
Autorizada	Montadora
JSV	FORD
JSV	FIAT
GTC	FORD

Autorizada	Espécie
JSV	TRATOR
JSV	CAMINHÃO
JSV	CARRO
GTC	CARRO

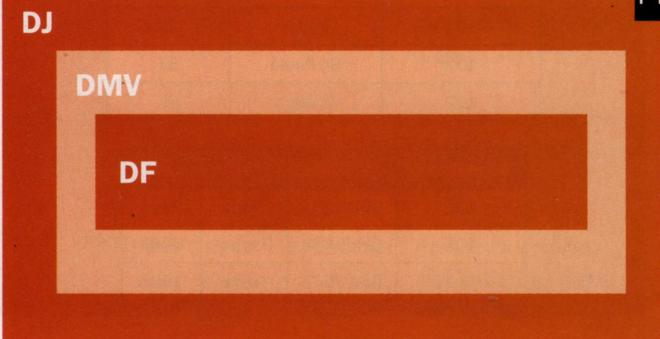
Montadora	Espécie
FORD	TRATOR
FORD	CAMINHÃO
FORD	CARRO
FIAT	CARRO
FIAT	CAMINHÃO

Se fizermos a junção destas 3 projeções de **T**
obteremos a Tabela Original **T**

F12



F13



esse conceito para tabelas que possam ser decompostas em mais de duas, sem perda de informação (denominado **tabela n-decomponível**, onde "n" é maior que dois). Claramente, este conceito só é aplicado em tabelas com 3 ou mais atributos.

Este conceito determina o seguinte:

- 1) Se uma tabela **T** possui três atributos {a1, a2, a3};
- 2) Teremos três projeções possíveis para ela: **T1**: {a1, a2}, **T2**: {a1, a3}, **T3**: {a2, a3};

3) Dizemos que há uma **dependência de junção** se todas as linhas em **T** puderem ser formadas a partir da junção dessas três projeções simultaneamente. Ou seja, se a tabela **T** original puder ser decomposta em 3 tabelas menores **T1**, **T2** e **T3** (3-decomponível), as quais sejam originadas de suas projeções, teremos na tabela **T** o que chamamos *dependência de junção*.

Para representar que **T** tem dependência de junção com suas projeções, usamos esta forma: **T***(T1, T2, T3).

Para exemplificar, utilizaremos o trio Autorizada-Montadora-Espécie:

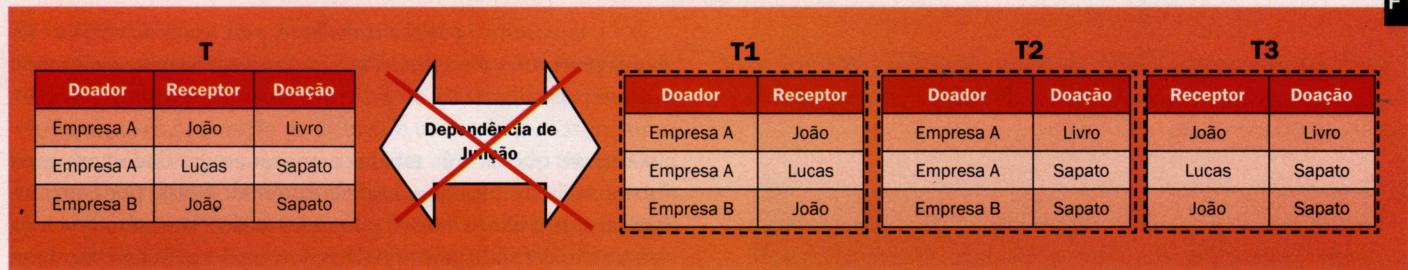
▪ Vamos considerar o seguinte: se uma autorizada vende determinada espécie de veículo e representa uma montadora que fabrica essa espécie, então ela comercializará tal espécie para a montadora.

▪ Quando encontrar situações semelhantes a essa, verifique se existe a chamada **restrição cíclica**, que pode ser representada nas quatro observações abaixo:

- ♦ Se a autorizada **A** vende a espécie de veículo **V** (projeção **T2**: {a1,a3});
- ♦ Se a autorizada **A** representa a montadora **M** (projeção **T1**: {a1,a2});
- ♦ Se a montadora **M** fabrica a espécie de veículo **V** (projeção **T3**: {a2,a3});
- ♦ Significa que a autorizada **A** comercializará a espécie de veículo **V** para a montadora **M** (**Tabela T**).

Portanto se considerarmos essa restrição verdadeira, teremos a dependência de junção, como demonstra a **Figura 11**.

Pelo fato de a tabela **T** possuir dependência de junção, ela contém anomalias de atualização. Por exemplo: não será possível armazenar a informação de que a montadora Honda produz moto enquanto não houver uma autorizada designada para ela. Ou ainda, se a FORD deixar de produzir CARRO, teremos de eliminar duas linhas (a 3^a e a 6^a) e ainda perderemos a informação de que a GTC representa a FORD.



A Figura 12 nos mostra que, se fizermos a junção das tabelas **T1**, **T2** e **T3**, chegaremos na tabela **T** (indicando assim a dependência de junção). Iniciamos com a junção de **T1** e **T3**; na tabela resultante, fazemos a junção com **T2**.

Da mesma forma que toda DF é uma DMV (conforme mencionamos anteriormente), aqui também a DJ é uma generalização de DMV. Porém, essa demonstração é mais complexa e, por isso, não a incluiremos aqui. Entretanto, o leitor que quiser comprová-la pode consultar as páginas 342 a 346 do livro “Introdução a Sistemas de Bancos de Dados”, de C.J.Date. Essa generalização é mostrada graficamente na **Figura 13**.

▪ **Resumindo:** quando tivermos uma tabela que possa ser representada de uma forma mais simples por suas projeções (e isso só será possível se tivermos a restrição cíclica), teremos uma DJ nessa tabela.

Para facilitar ainda mais o entendimento, vejamos outro exemplo em que não teremos a dependência de junção. Suponha que nós tenhamos empresas que façam doações de alguns objetos para algumas pessoas, como na tabela original **T** da **Figura 14**.

Doador	Receptor	Doação
Empresa A	João	Livro
Empresa A	Lucas	Sapato
Empresa B	João	Sapato

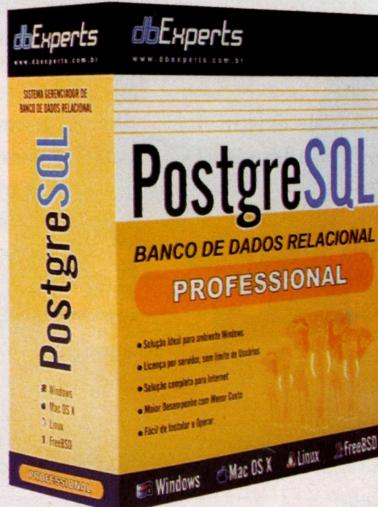
Observe que aqui não temos a DJ, pois se fizermos a junção das projeções de **T** teremos a tabela da **Figura 15** (que não é igual a **T**).

Ou seja, a dependência de junção não existe pois não temos a restrição cíclica:

- Se o doador **D** doou para o receptor **R**;
- Se o doador **D** doou o objeto **O**;
- Se o receptor **R** recebeu o objeto **O**;
- Não significa que o doador **D** doou o objeto **O** para o receptor **R**.

dbExperts

- Documentação totalmente em português
- Instalação super facilitada
- Suporte gratuito de 30 dias para instalação e configuração
- 1 chamada de suporte gratuita para desenvolvimento
- Total segurança dos dados
- Licenciamento em Windows apenas por servidor sem limite de usuários
- Diversas ferramentas de manipulação
- Total integridade com linguagens VB, Clarion, ASP, PHP, Delphi, Java, etc.
- Novas funções



PostgreSQL

Treinamentos

- SQL Básico
- PostgreSQL Módulo I
- PostgreSQL Módulo II
- PostgreSQL + PHP
- PostgreSQL + Delphi
- Treinamentos especiais para turmas fechadas

Suporte

Consultoria

Migração

RECURSOS E DESEMPENHO EQUIVALENTES AO ORACLE E SQL SERVER!

www.dbexperts.com.br

(011) 5506-1991

De fato, observe que a empresa A doou para João. A empresa A doou sapato. João recebeu sapato. Porém, a empresa A **não** doou sapato para João.

5ª Forma Normal

Esta forma ocorre muito raramente e exige uma atenção maior para que se perceba a sua necessidade. Esse fato colabora para que nem todos os autores a citem em seus artigos.

Precisamos aqui estar bem familiarizados com os processos de decomposição sem perdas e de junção e com o conceito de dependência de junção. Esta forma normal é também chamada **forma normal de projeção-junção** (FN/PJ).

Uma tabela está em 5FN se estiver em 4FN e não contiver DJs que não sejam determinadas por chaves candidatas.

Em outras palavras, se for possível n -decompor a tabela, então ela deverá ser substituída por suas n projeções para estar na 5FN.

Nota: alguns autores confundem a 5FN com relacionamentos ternários. Ao contrário dos relacionamentos, a DJ acontece **internamente** na tabela.

Como exemplo, utilizaremos o trio Piloto - Avião - Trajeto. Para tal, considere o seguinte:

- 1) Se o Piloto **P** pilota determinado avião **A**;
- 2) Se o Piloto **P** pilota pelo trajeto **T**;
- 3) Se o avião **A** faz o trajeto **T**;
- 4) Então o piloto **P** pilota o avião **A** pelo trajeto **T**.

Temos aqui a restrição cíclica. Observe que a terceira consideração não é derivada das duas considerações anteriores, como pode parecer à primeira vista. Repare que o piloto P poderia pilotar o avião A em um trajeto diferente de T, e o avião A poderia fazer o trajeto T com outro piloto que não o P. Veja o gráfico da **Figura 16**.

Observe que **T** encontra-se em 4FN, pois, apesar de existir a relação **piloto →→ avião | trajeto**, não temos **independência** entre avião e trajeto.

Como projeções de **T**, temos: **T1: {piloto, avião}**, **T2: {piloto, trajeto}**, **T3: {avião, trajeto}**. Há uma dependência de junção, pois todas as linhas em **T** podem ser formadas pela junção das três projeções simultaneamente. Portanto, na 5FN teríamos três tabelas no lugar da tabela original.

Nota: o processo da 5FN é trabalhoso, pois temos de verificar todas as projeções e o resultado de suas junções (imagine uma tabela 5-decomponível).

Observe que as três tabelas armazenam os dados em uma forma mais simples que a tabela original. Nas, podemos informar que o piloto "0030" pilotará o avião "103" sem que haja um trajeto definido para aquele avião ou piloto. Repare também que, se o piloto "0010" passar a fazer o trajeto "Rio-Spa", bastará inserir essa linha na tabela **T2**.

Se o piloto "0020" não pilotar mais o avião "105", ao excluirmos essa linha da tabela original **T**, perdemos a informação de que o avião "105" faz o trajeto "Rec-Rio". Já no trio de tabelas, essa operação poderia ser feita facilmente na tabela **T1**.

Normalização passo-a-passo

1. Faça um levantamento dos dados a serem armazenados. Procure organizá-los de forma a retirar todas as multivalorações, deixando cada linha com apenas um valor por coluna. Com isso obtemos tabelas na 1FN;

2. Para cada tabela na 1FN, elimine as dependências funcionais parciais (movendo para outra tabela os atributos que dependem de parte da chave primária). Este passo produzirá várias tabelas na 2FN;

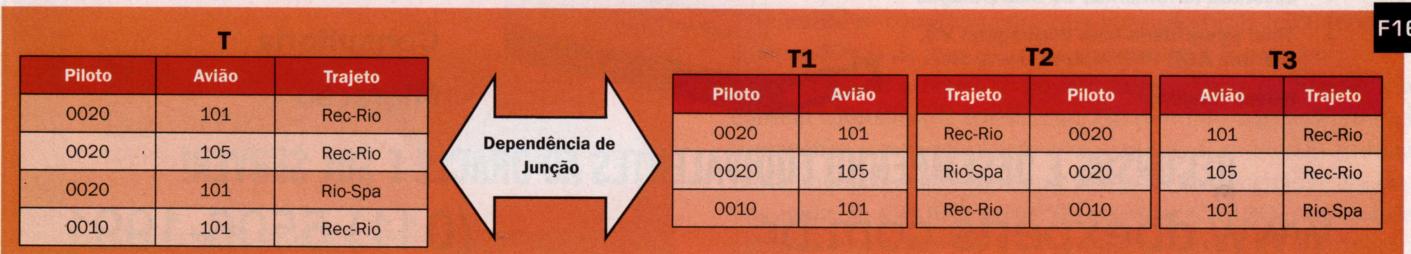
3. Para cada tabela na 2FN, elimine as dependências funcionais transitivas (movendo para outra tabela os atributos que não dependem somente da chave primária). Este passo produzirá tabelas na 3FN;

4. Para cada tabela na 3FN que possua mais de uma chave candidata com sobreposição, elimine as dependências funcionais em que o determinante não seja uma chave candidata (movendo os atributos dependentes para outra tabela).

Este passo produzirá tabelas na FNBC. As regras de 2 a 4 podem ser resumidas em uma única orientação: "Obtenha projeções das tabelas originais até eliminar todas as DFs em que o determinante não seja uma chave candidata";

5. Para cada tabela na FNBC, verifique se existe mais de uma DMV (que não seja também DF) de um mesmo multi-determinante e, se existir, se elas são independentes. Nesse caso, eliminate-as, migrando essas colunas para outra tabela. Este passo produzirá tabelas na 4FN.;

6. Para cada tabela na 4FN, verifique e eliminate quaisquer DJs que não sejam determinadas por chaves candidatas – embora talvez devamos acrescentar: "se você conseguir encontrá-las". Este passo produzirá uma coleção de tabelas em 5FN.



CONCLUSÃO

Existem outras formas normais, como **Forma Normal de Chave Domínio** e **Forma Normal de Restrição-União**. Elas são baseadas na restrição de linhas de uma tabela. Assim como a projeção visa obter apenas algumas colunas de uma tabela (vertical), a restrição visa buscar apenas algumas linhas de uma tabela (horizontal). Essas restrições podem ser recompostas por meio de *união*, em contraposição à projeção de colunas vistas neste e no outro artigo, que podem ser recompostas por meio de *junção*.

Como essas formas normais não possuem aplicabilidade prática para os SGBD atuais (a união não apresenta as melhores performances), não descreveremos esse assunto em mais detalhes aqui. Contudo, se tiver interesse, você encontrará informações no livro "Introdução a Sistemas de Bancos de Dados", de C.J.Date (capítulo 12, páginas 354 e 355).

Ressaltamos novamente que, na maioria dos projetos de banco de dados, o projetista (principalmente iniciante) precisará normalizar até a 3FN. Espero ter contribuído para o seu aprendizado e até a próxima. ■

R

C.J.DATE , **Introdução a Sistemas de Banco de Dados** – 7^a edição americana
Henry F. Korth e Abraham Silberschatz, **Sistema de Banco de Dados** – 2^a edição
Carlos Alberto Heuser , **Projeto de Banco de Dados** – 4^a edição
Michael J. Hernandez , **Aprenda a Projetar seu Próprio Banco de Dados**
Prof. Fábio – UFSC / Fernando Fonseca e Ana Carolina, **Alguns exemplos retirados de transparências na WEB**

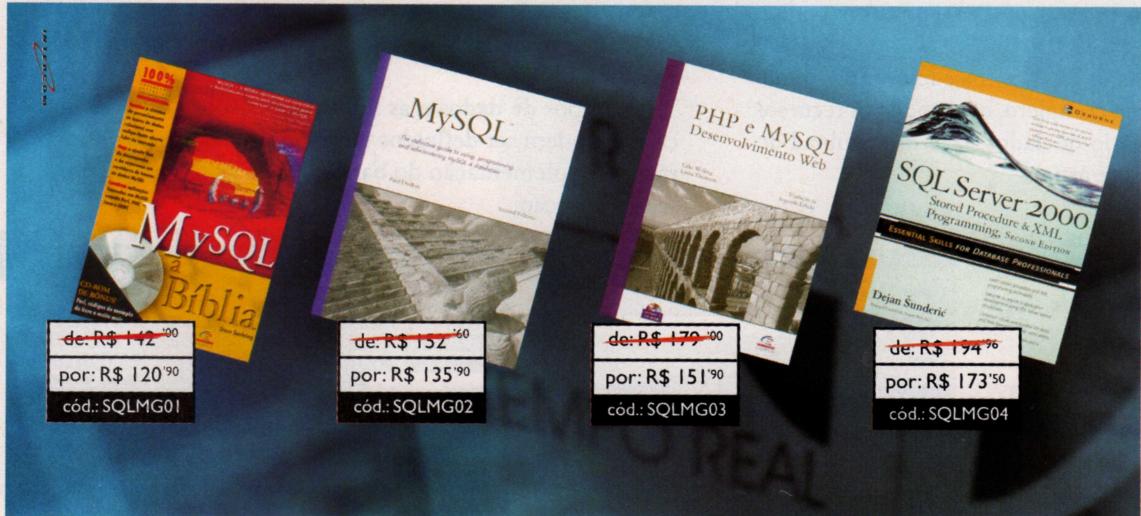
A

O Prof. Bráulio Ferreira de Carvalho (braulio@cic.unb.br ou brauliof@stf.gov.br) ministra as disciplinas "Banco de Dados" e "Organização de Arquivos" na graduação em Licenciatura da Computação da Universidade de Brasília – UNB. Atua também na área de Administração de Dados e Metodologia de Sistemas, na Secretaria de Informática do STF (Supremo Tribunal Federal). Além disso, aplica treinamentos e dá consultorias na área de desenvolvimento de sistemas de modo geral.

S

Comente essa matéria em: www.sqlmagazine.com.br/sql7

Pensou Base de Dados, pensou Tempo Real



A livraria **Tempo Real** possui a maior variedade de livros do mercado! São mais de 15 mil títulos, nacionais e importados, disponíveis em nossas lojas ou em nosso site. Confira ao lado os títulos que selecionamos para você!!

Os valores promocionais serão válidos somente para compras efetuadas no site e utilizando os códigos dos cupons. Promoção válida até dia 30.II.2003

ACESSE

www.tempo-real.com.br

OU VISITE NOSSAS LOJAS

loja São Paulo: Al. Santos, 1202 | São Paulo, SP | CEP 01418-100 | Fone: (11) 3266.2988
loja Porto Alegre: Av. Augusto Meyer, 167 | Porto Alegre, RS | CEP 90550-110 | Fone: (51) 3029.0044

A livraria do profissional de informática



Prática | Completa | Especializada