

Normalização

Introdução

No banco de dados, a organização e o projeto ditarão a facilidade, a rapidez e a agilidade para a manutenção e, principalmente, para a consulta das informações armazenadas, sem desperdiçar espaço do disco e tempo do usuário.

A fim de se chegar a um projeto formal, fundamentado em uma boa estrutura, para atingir o objetivo e mostrar eficiência, seguimos sempre o proposto. Isso quer dizer que, primeiro, é preciso se ater a uma boa modelagem, utilizando o MER – ferramenta que auxilia em uma proposta organizada com mais acurácia para seguir em frente. No segundo passo, que é o maior interesse, vamos ao modelo relacional, seguindo toda especificação do MER. Com o modelo relacional, e algumas regras determinadas, está tudo pronto para que os dados sejam inseridos, mas será que esse projeto está correto e não trará problemas futuros? Será que as tabelas foram bem projetadas? Não há espaço para redundâncias que poderiam gerar problemas?

Existe mais uma ferramenta para que o projeto possa ser melhorado. Trata-se de um procedimento denominado normalização, um processo matemático formal, fundamentado na teoria dos conjuntos, uma vez que o modelo relacional também segue a mesma estrutura. A normalização indicará o nível de qualidade de uma tabela. Esse processo trabalha com uma tabela de cada vez e, por meio de testes, afere-se se a tabela ou relação satisfaz à regra, ou, formalmente dizendo, satisfaz a uma forma normal.

O objetivo do processo é analisar os esquemas de tabela para que alcance determinadas propriedades desejáveis, como a minimização de redundâncias,

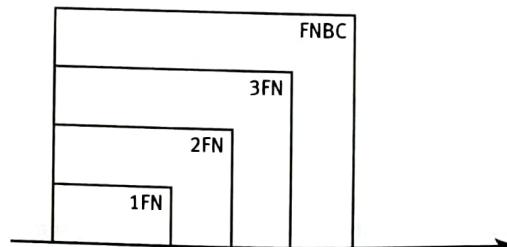
e meios para reduzir irregularidades de inserção, exclusão e atualização com a tabela populada. Porém tudo isso é possível se nos propusermos a satisfazer um conjunto de condições para normalizar essas tabelas. A normalização é usada para decompor ou dividir tabelas que não estejam bem projetadas em tabelas menores e mais eficientes, e assim ter projetos com alta qualidade.

No processo de normalização, as regras são chamadas de formas normais e têm como meta impedir redundâncias na organização das tabelas e evitar resultados indevidos na atualização de uma tabela, independentemente de outras. Além disso, elas garantem que projetos de bancos de dados apresentem as “propriedades desejáveis”, que se obtenha um esquema de banco de dados aplicável mais facilmente e inteligível, e que sejam evitadas inconsistências lógicas, resultantes de operações de atualizações em tabelas.

Para entender a normalização, é necessário saber que são impostas restrições à tabela em análise. Assim, o nível de restrições, ou forma normal, que a relação atingiu vai sendo sinalizado. Para que a tabela passe para um nível superior, ela precisa atender aos níveis inferiores primeiro, ou seja, é preciso seguir uma sequência, pois uma forma normal inferior é pré-requisito para o nível posterior. Quando a tabela atende à última regra, ou forma normal imposta, está implícito que ela atenderá também às formas normais anteriores à imposta. Na Figura 4.1 é mais fácil entender como se dá a classificação das formas normais. Como se observa na Figura 4.1, uma tabela só está na segunda forma normal (2FN) se atender à primeira forma normal (1FN) e às regras da 2FN e assim sucessivamente, com as outras formas normais. Nesse aspecto, diz-se que tabelas ou relações na forma normal Boyce Codd (FNBC) estão também na terceira forma normal (3FN). Relações na 3FN também estão na 2FN e relações na 2FN na 1FN. Por exemplo, dizemos que, se uma tabela atende à 3FN, ela atende à 2FN e, consequentemente, à 1FN.

Nesse processo de normalização, tabelas inadequadas são decompostas por meio da divisão de esquemas sem perder informação. Assim, teremos tabelas menores e mais apropriadas. Na decomposição, os produtos devem ser esquemas sem informações redundantes e sem perdas de dados e relacionamentos, mantendo a semântica original dos dados. É muito perigoso nesse processo perder informações relevantes.

FIGURA 4.1. Níveis de normalização



Na sequência deste capítulo explicaremos cada forma normal separada e detalhadamente.

A primeira forma normal

O processo de normalização se inicia pela primeira forma normal ou 1FN, que é a primeira de todas as regras. É uma forma, mais simples e abrangente exigindo apenas que todos os atributos tenham domínios atômicos, isto é, valores indivisíveis. Isso quer dizer que em cada tupla deve haver apenas um valor para cada atributo. É uma regra a que estamos acostumados, mas, por uma distração, pode-se colocar um atributo composto no MER e no mapeamento ter um atributo para aninhar valores. Se esse erro ocorrer no mapeamento, problemas sérios virão pela frente se a tabela não for normalizada. No exemplo da Figura 4.2 tem-se um esquema de relação. Esse esquema possui o atributo endereços, que indica valores não atômicos. Em outras palavras, sugere-se colocar mais de um valor em apenas um atributo. Se isso acontecer, há violação da 1FN. Ainda na Figura 4.2 temos a mesma tabela com dados e verifica-se que na primeira e na última tupla, o atributo endereços está com mais de um valor, não atendendo, assim, à 1FN.

FIGURA 4.2. Tabela Alunos – não atende à 1FN

ALUNOS

<u>matrícula</u>	nome	endereços
------------------	------	-----------

ALUNOS

<u>matrícula</u>	nome	endereços
1085123	José	Av. das Flores, 25, R. Francisco Moisés, 14
1078987	Antônio	R. 13 de maio, 345
1089771	Maria	R. Dunlop, 11
1067543	Rebeca	Av. Mokarzel, 165
1076233	Paulo	Av. Um, 89, R. 13 de maio, 341

Para normalizar a tabela, deve-se corrigir com as restrições impostas para a 1FN – é preciso colocar valores indivisíveis. A tabela normalizada, atendendo à 1FN, é mostrada na Figura 4.3, que foi corrigida acrescentando-se mais um atributo: endereço comercial. A tabela agora apresenta todos os valores atômicos

e indivisíveis. Uma opção seria mantê-la com os mesmos atributos e deixar o atributo endereço somente com um endereço para cada tupla, mas, nesse caso, alguns endereços seriam descartados. Essa informação poderia ser excluída? É preciso cuidado, pois as informações não podem ser descartadas. Ao normalizar uma tabela, a principal preocupação é tentar excluir o maior número possível de redundâncias sem perder informações. Com a tabela normalizada, como mostra a Figura 4.3, nota-se que, embora normalizada, possui algumas tuplas em branco, o que, em uma tabela com muitos dados, seria desperdício. Seria esta a solução ideal? Vamos analisar outra solução possível.

FIGURA 4.3. Tabela Alunos na 1FN – Opção 1

ALUNOS

<u>matrícula</u>	<u>nome</u>	<u>endereço</u>	<u>endereço comercial</u>
1085123	José	Av. das Flores, 25	R. Francisco Moisés, 14
1078987	Antônio	R. 13 de maio, 345	
1089771	Maria	R. Dunlop, 11	
1067543	Rebeca	Av. Mokarzel, 165	
1076233	Paulo	Av. Um, 89	R. 13 de maio, 341

Outro modo de normalizar a tabela Alunos para atender à 1FN é conservar os atributos matrícula e nome, e o atributo endereços passar para endereço, permitindo somente valores indivisíveis e replicando as informações em nome e matrícula, como mostra a Figura 4.4. Se admitirmos essa solução, a chave primária deve ser alterada, pois os dados no atributo matrícula foram duplicados. Assim, deve-se compor a chave primária com os atributos matrícula e endereço. A tabela resultante dessa normalização não apresenta perdas de informações, mas ainda possui algumas redundâncias.

Neste exemplo simples ainda é possível pensar em outra maneira de organizar a tabela para que ela atenda à 1FN. É possível decompô-la, mas este é um procedimento com o qual se deve ter muito cuidado para que não se percam informações. O atributo em questão é endereços, que deve sair e fazer parte da nova tabela. A chave primária é o atributo matrícula, que também deve fazer parte da nova tabela. Dessa forma, tem-se, então, duas tabelas. A primeira, a tabela Alunos, possui a matrícula como chave primária e o nome do aluno. À segunda tabela pertencem os atributos matrícula e endereço. Como nessa nova tabela Alunos-Endereços têm-se dados duplicados em matrícula e é possível ter em endereços, então a chave primária deve ser composta de matrícula e endereço como ilustrado na Figura 4.5.

FIGURA 4.4. Tabela Alunos na 1FN – Opção 2**ALUNOS**

<u>matrícula</u>	<u>nome</u>	<u>endereço</u>
1085123	José	Av. das Flores, 25
1085123	José	R. Francisco Moisés, 14
1078987	Antônio	R. 13 de maio, 345
1089771	Maria	R. Dunlop, 11
1067543	Rebeca	Av. Mokarzel, 165
1076233	Paulo	Av. Um, 89
1076233	Paulo	R. 13 de maio, 341

FIGURA 4.5. Tabela Alunos na 1FN – Opção 3**ALUNOS**

<u>matrícula</u>	<u>nome</u>
1085123	José
1078987	Antônio
1089771	Maria
1067543	Rebeca
1076233	Paulo

ALUNOS-ENDEREÇOS

<u>matrícula</u>	<u>endereço</u>
1085123	Av. das Flores, 25
1085123	R. Francisco Moisés, 14
1078987	R. 13 de maio, 345
1089771	R. Dunlop, 11
1067543	Av. Mokarzel, 165
1076233	Av. Um, 89
1076233	R. 13 de maio, 341

No exemplo mostrado há três soluções que podem ser adotadas. Para escolher uma dentre elas, deve-se verificar se não há redundâncias e se a solução adotada permite ter novas informações. Na última opção da tabela Alunos normalizada, temos duas tabelas, e ambas permitem a inclusão de dados sem redundâncias de informações.

A 1FN não admite tabelas com valores não atômicos para os atributos; eles devem ter valores únicos. Também não permite atributos multivalorados compostos, que são ditos como tabelas aninhadas. Para um melhor entendimento, pode-se pensar em uma tabela dentro de cada tupla, como mostra a Figura 4.6. Com esse problema de tabelas aninhadas, é possível também normalizar seguindo as opções anteriores ou decompondo a tabela.

Para a decomposição, retiram-se os atributos aninhados, colocando-os em outra tabela, e leva-se a chave primária. Agora a tabela Alunos é composta pela chave primária, matrícula e pelo atributo nome. A nova tabela possui os atributos rua, cidade e matrícula. Para que o atributo matrícula seja chave primária nesta tabela vemos, na Figura 4.6, que teremos valores repetidos. Assim, a chave deve ser composta com os atributos matrícula e rua.

FIGURA 4.6. Tabela Alunos (aninhada) na 1FN

ALUNOS

<u>matrícula</u>	<u>nome</u>	<u>endereço</u>	
		<u>rua</u>	<u>cidade</u>
1085123	José	Av. das Flores, 25 R. Francisco Moisés, 14	Campinas São Paulo
1078987	Antônio	R. 13 de maio, 345	Campinas
1089771	Maria	R. Dunlop, 11	Limeira
1067543	Rebeca	Av. Mokarzel, 165	Dourados
1076233	Paulo	Av. Um, 89 R. 13 de maio, 341	Salto Campinas

ALUNOS

<u>matrícula</u>	<u>nome</u>
1085123	José
1078987	Antônio
1089771	Maria
1067543	Rebeca
1076233	Paulo

ALUNOS-ENDEREÇOS

<u>matrícula</u>	<u>rua</u>	<u>cidade</u>
1085123	Av. das Flores, 25	Campinas
1085123	R. Francisco Moisés, 14	São Paulo
1078987	R. 13 de maio, 345	Campinas
1089771	R. Dunlop, 11	Limeira
1067543	Av. Mokarzel, 165	Dourados
1076233	Av. Um, 89	Salto
1076233	R. 13 de maio, 341	Campinas

Depois de ter estudado a 1FN, para que possamos continuar o processo de normalização, há algumas regras mais elaboradas a seguir, exigindo mais restrição. No próximo item, veremos alguns conceitos para que possamos entender e aplicar melhor as próximas regras impostas nesse processo.

Dependências funcionais

Para continuar o estudo sobre normalização, é preciso falar das restrições de integridade ou regras de integridade, denominadas também de dependências funcionais (DF).

Essas restrições ou DFs trabalham com os atributos de uma tabela, do inter-relacionamento de seus valores, mas considerando todas as tuplas da tabela e não somente um conjunto representativo de duas a três linhas. Podemos definir que há uma dependência funcional entre atributos se para cada valor do atributo A há somente um valor para o atributo B em um par de tuplas. Dizemos que um atributo depende funcionalmente de outro se para cada valor do atributo A sempre houver um valor único no atributo B. Assim, podemos representar com uma seta ($A \rightarrow B$). Nesse caso, diz-se que A realiza em B ou A seta B, ou A determina funcionalmente B, ou B é funcionalmente dependente de A, ou seja, para todo valor de A existe um único valor em B.

Na Figura 4.7 temos um exemplo de uma relação R com atributos A e B, para a qual o valor é x no atributo A, e para todo x existente em A trazemos o valor correspondente de 1 no atributo B para todas as tuplas da tabela R. Seguindo o mesmo raciocínio, quando o valor é y em A, tem-se somente o valor 7 em B, e para todo z tem-se 3. Concluindo, para todo valor no atributo A, há somente um único valor no atributo B, então dizemos que temos a DF $A \rightarrow B$.

FIGURA 4.7. Exemplo de dependência funcional

R	
A	B
x	1
y	7
z	3
z	3
x	1
y	7

$A \rightarrow B$

A Figura 4.8 mostra um exemplo da tabela Locais na qual há três atributos. Analisando, vê-se que para cada cidade há somente um endereço e para cada endereço um único CEP. Num primeiro momento, ao olhar rapidamente a tabela dizemos que há uma DF em que para cada cidade e endereço temos um CEP. Isso quer dizer que podemos representar a DF em cidade, endereço → CEP. É necessária uma análise, mas será que todas as tuplas inseridas nesta tabela seguirão a mesma regra? Se pensarmos em cada cidade, há várias ruas que não mantêm um único valor sempre, mas para cada rua há somente um CEP. Então podemos concluir que na tabela Locais não temos a DF cidade, endereço → CEP, e sim a DF endereço → CEP.

FIGURA 4.8. Exemplo de dependências funcionais

LOCais

cidade	endereço	CEP
São Paulo	Av. dos Ipes	54321-010
Curitiba	R. 13 de maio	25655-101
Curitiba	R. 13 de maio	25655-101
São Paulo	Av. dos Ipes	54321-010

Cidade, endereço → CEP X

endereço → CEP

No exemplo da tabela Alunos da Figura 4.9, temos que matrícula → nome, ou seja, para cada valor de matrícula haverá um único valor em nome, ou melhor, o nome depende funcionalmente da matrícula. O próprio conceito, depender funcionalmente, significa que para cada valor do atributo matrícula existe um valor único em nome. Assim, toda vez que aparecer nessa tabela o valor do atributo matrícula, haverá valor em nome e será o mesmo. O atributo que dita a DF é chamado de determinante, e pode ser mais de um. Para fixar melhor essa ideia, deve-se lembrar de que o atributo que está à esquerda da DF é dito determinante e o atributo à direita é dito atributo dependente ou dependente funcionalmente.

FIGURA 4.9. Exemplo de dependência funcional

ALUNOS

matrícula	nome
1085123	José Silva Reis
1085123	José Silva Reis
1078987	Antônio Camargo
1089771	Maria do Carmo Silva
1089771	Maria do Carmo Silva
1078987	Antônio Camargo

matrícula → nome

Podemos ainda encontrar os termos *dependência funcional total* e *dependência funcional parcial*. A dependência funcional é parcial quando, em uma tabela, tem-se uma chave primária composta e um de seus atributos depende funcionalmente de apenas parte da chave. Isso quer dizer que depende de apenas um ou mais atributos que compõem a chave, mas não de todos, e ainda ele pode ser removido e a DF continua existindo. A DF é total se o atributo em questão depende da totalidade da chave, ou seja, de todos os atributos que a compõem.

Continuando o processo de normalização utilizaremos o conceito de DFs e, assim, aprenderemos melhor a aplicação de uma DF.

A segunda forma normal

Entenderemos, agora, a segunda forma normal ou 2FN. A primeira regra para que uma tabela atenda à 2FN é que atenda também à 1FN. Na sequência verificaremos se a tabela possui chave primária composta, pois a forma normal só se aplica a tabelas que a possuam. Se isso não ocorrer, devemos desprezá-la e continuar o processo.

Se estivermos com uma tabela que atenda à 1FN e possua chave primária composta, precisamos verificar a presença de uma dependência funcional total. A tabela está na 2FN se cada linha possuir uma chave primária e se cada campo que não seja chave primária depender inteiramente da chave primária, e de todos os atributos que a compõem. A dependência funcional total ocorre quando temos uma chave primária composta e o atributo em questão depende de toda a chave e não apenas de uma parte dela.

No exemplo da Figura 4.10 temos uma tabela Alunos-Cursos com dados e verificamos que ela atende à 1FN. Agora a preocupação se dá para saber se a tabela

atende à 2FN. Para isso, ela precisa, antes, atender à 1FN, o que já é sabido. A segunda regra é mais restrita e diz que atributos que não pertençam à chave primária devem depender inteiramente da chave, ou seja, ter uma dependência funcional total da chave primária.

É necessário analisar os atributos separadamente, aqueles que não pertencem à chave primária. Na tabela Alunos-Cursos, tem-se que o atributo número-horas corresponde ao número de horas que o aluno deve cumprir no curso especificado. Nota-se que há um mesmo curso para alunos diferentes com número de horas diferente. Para analisar a dependência desse atributo em relação à chave primária, deve-se verificar sua dependência em relação às duas partes da chave, ou seja, o atributo número-horas depende diretamente dos alunos e também do curso.

Outro atributo é o local-curso, que está diretamente ligado ao curso, pois o local será determinado de acordo com ele. Sobre esse mesmo atributo, ele seria dependente do aluno? Em situações muito específicas sim, mas, para isso, a situação seria descrita pelo sistema. Como neste caso nada está especificado, conclui-se que não há uma ligação direta entre local-curso e aluno. Para uma melhor compreensão, devemos pensar que qualquer aluno convocado para o curso será indicado para determinado local sem restrição. Notamos, então, que a ligação ou dependência funcional existe apenas com o aluno e não com o curso. Dessa forma, a relação não atende à 2FN, como mostra a Figura 4.10. Na mesma figura verifica-se também a redundância das informações que são geradas no atributo local-curso.

FIGURA 4.10. Tabela Alunos-Cursos – não atende à 2FN

ALUNOS-CURSOS

<u>RA-aluno</u>	<u>número-curso</u>	<u>número-horas</u>	<u>local-curso</u>
101234245	234	40	LCC1
334970610	234	20	LCC1
225409877	564	30	LAB1

Para atingir a 2FN é preciso decompor a tabela Alunos-Cursos, e o mais importante é não perder nenhuma informação. Sabe-se que o atributo que viola a 2FN é o local-curso, assim deve-se retirá-lo. No entanto, essa informação é necessária e não se pode descartar um atributo. Vamos retirá-lo da tabela, mas mantê-lo em uma nova tabela. Se ficarmos atentos, verificaremos que o atributo está ligado ao atributo número-curso. Assim, teremos a solução. O número do curso repetirá e será a chave da nova tabela representada na Figura 4.11. Agora, vê-se que a tabela

Alunos-Cursos ficou normalizada, atendendo à 2FN, e a nova tabela Local-Cursos está com a chave primária número-curso e o atributo local-curso.

FIGURA 4.11. Tabelas na 2FN

ALUNOS-CURSOS

<u>RA-aluno</u>	<u>número-curso</u>	<u>número-horas</u>
10122234245	234	40
33498970610	234	20
22546509877	564	30

LOCAL-CURSOS

<u>número-curso</u>	<u>local-curso</u>
234	LCC1
564	LAB1

A terceira forma normal

Nesta etapa, para continuarmos o processo de normalização, temos de verificar as regras ditadas pela terceira forma normal (3FN), mais restrita que a 2FN. Como visto anteriormente o primeiro passo é atender à 1FN e à 2FN para depois voltarmos à 3FN. A 3FN exige que cada atributo não pertencente à chave primária dependa diretamente da chave primária e que os atributos sejam mutuamente independentes.

Nesse caso, fala-se outra vez em dependência funcional, ou seja, na 3FN não há uma dependência transitiva e sim uma dependência funcional total entre atributos e chave primária. Uma dependência é transitiva ou indireta quando um atributo depende de outro atributo que não seja chave da mesma tabela, apesar de depender da chave também. Para que a regra seja estabelecida é indicada a divisão da tabela com a eliminação das redundâncias tendo como referencial o atributo que possui uma dependência transitiva.

Vamos exemplificar com a tabela Alunos mostrada na Figura 4.12, que não atende à 3FN. Vamos analisá-la passo a passo e depois corrigi-la. Primeiro, verificamos e podemos afirmar que a tabela atende à 1FN, pois possui apenas valores atômicos. Ela também atende à 2FN, pois possui chave primária simples e atributos que dependem da chave.

Agora, vamos verificar os atributos para checar a dependência entre eles e a chave primária, analisando cada atributo separadamente. Começando pelo atributo nome, é fácil perceber que ele depende diretamente da matrícula, ou seja,

da chave primária. O atributo disciplina também depende da matrícula que está relacionada ao indivíduo.

Quanto ao atributo crédito, vemos que está diretamente ligado à disciplina, pois para cada disciplina temos um número estabelecido de crédito. O atributo crédito está ligado à disciplina para depois ocorrer a matrícula. Encontramos aqui uma dependência transitiva. Esse atributo está ligado à disciplina, que está ligada à matrícula. É notável que se trata de uma ligação indireta, sem dependência funcional total entre crédito e matrícula.

Assim, conclui-se que a relação não atende à 3FN e observam-se as redundâncias existentes nos atributos disciplina e créditos. Nem sempre conseguimos extrair todas as informações que se repetem, mas podemos amenizar essa ocorrência.

FIGURA 4.12. Tabela Alunos – não atende à 3FN

ALUNOS			
matrícula	nome	disciplina	crédito
1085123	José	redes	20
1078987	Antônio	banco de dados	25
1089771	Maria	50	15
1067543	Rebeca	redes	20
1076233	Paulo	50	15

Para que a tabela atenda à 3FN, devemos tomar providências como na tabela que normalizamos para a 2FN, ou seja, decompô-la. Nesse processo, é preciso cuidado, pois nenhuma informação ou relacionamento podem ser perdidos. Primeiro, tira-se o atributo em evidência e cria-se uma nova tabela. Então, faz-se o mesmo com o atributo crédito e o atributo determinante, disciplina, que está diretamente ligado a ele. O atributo disciplina deve ser repetido na nova tabela mas permanecer na tabela original, pois há uma dependência direta com a chave primária. Temos, agora, duas tabelas, a tabela Alunos normalizada na 3FN e a tabela Créditos com as respectivas disciplinas e créditos, conforme Figura 4.13.

Na maioria dos casos um projeto é normalizado até a 3FN, pois é suficiente para um bom projeto em mãos. Às vezes, é necessário um estudo de caso para observar a real necessidade de decomposição de tabelas a fim de atender às formas normais acima da 3FN, pois haver muitas tabelas pequenas em certos casos não é indicado. Na literatura encontramos a FNBC que veremos a seguir, mas também a 4FN e 5FN.

ALUNOS

<u>matrícula</u>	nome	disciplina
1085123	José	redes
1078987	Antônio	banco de dados
1089771	Maria	SO
1067543	Rebeca	redes
1076233	Paulo	SO

CRÉDITOS

disciplina	crédito
redes	20
banco de dados	25
SO	15