



Desenvolvimento de Sistemas

Normalização de dados: anomalias, conceito, motivação de uso, formas normais

Anomalias de inserção, remoção e atualização

Antes da apresentação sobre o assunto normalização, é de suma importância abordar as anomalias em banco de dados, ou seja, as alterações que podem gerar inconsistências em uma base de dados relacional, segundo Henrique Neto (2015).

As anomalias podem ser classificadas em três categorias: de inclusão, de exclusão e de atualização.

Anomalia de inclusão

A primeira anomalia é a de inclusão. Ela indica que não será possível cadastrar ou adicionar um determinado dado a não ser que outro dado esteja disponível.

Por exemplo, não é possível cadastrar um aluno em determinado curso se a instituição não tiver ofertado ou cadastrado o curso.

Anomalia de exclusão

A segunda anomalia é a de exclusão. Nela, ao excluir um certo registro em uma tabela, dados que estão intimamente ligados a esse registro em outra tabela serão ser excluídos.

Por exemplo, quando se exclui um cliente de supermercado, todas as compras realizadas por ele nesse estabelecimento serão automaticamente excluídas do sistema.

Anomalia de atualização

A terceira anomalia é a de atualização, na qual um determinado dado de uma tabela, ao ser alterado, deverá ter reflexo, alterando dados em outras tabelas que tenham uma ligação direta.

Por exemplo, ao alterar o código do aluno cadastrado em um curso técnico, as tabelas “CursosTecnicos” e “Instituição” deverão ser alteradas também, mantendo assim o relacionamento entre alunos, cursos e instituição.

A presença de tais inconsistências, que costumam aparecer em bancos de dados mal projetados ou, em sua maioria, não normalizados, pode causar dependência total, parcial ou transitiva.

Dependência total

A dependência total ocorre quando um atributo que não é chave depende de todas as chaves e não somente de um atributo ou parte dele. Observe a tabela 1, que simula uma situação de banco de dados em que se registram dados de tutoria de uma escola. Considerando que as colunas “idTutor” e “idConteudo” formam uma chave composta, a coluna “valorHora” depende das duas chaves.

idTutor	idConteudo	valorHora
1	0001	20,00
2	0002	45,00
3	0003	55,00

Tabela 1 – Tabela de banco de dados com dependência total



Dependência parcial

A dependência parcial ocorrerá quando o atributo não chave depender parcialmente das chaves compostas.

No exemplo mostrado na tabela 2, existe uma chave composta por “idTutor” e “idConteudo”. A coluna “conteudo” refere-se apenas à descrição do conteúdo que o tutor terá que ministrar. Pode-se inferir que a coluna “conteudo” não depende das duas chaves, somente da chave “idConteudo”, devendo, portanto, as duas colunas estarem juntas na mesma tabela. Por outro lado, a presença da coluna e chave “idTutor” representa, nesse caso, um erro de dependência parcial, pois não deveria estar na tabela em questão, devendo aludir a uma tabela de cadastro de tutores e não de conteúdo.

idTutor	idConteudo	conteudo
1	0001	Banco de dados
2	0002	Java
3	0003	Normalização

Tabela 2 – Tabela de banco de dados com dependência parcial

Dependência transitiva

A dependência transitiva ocorrerá quando um campo reconhecidamente não chave não depender da chave primária, mas depender de um atributo ou campo não chave.

No exemplo apresentado na tabela 3, considere “subTotal” uma coluna com o valor que o tutor receberia em horas pela disciplina. Esse cálculo é dependente do total de horas do conteúdo e do valor da hora. Assim, é possível perceber que a coluna “subTotal” depende somente do campo “totalHoras”, visto que “idTutor” e “idConteúdo” estão fazendo referência a tabelas diferentes (respectivamente cadastro de tutores e conteúdo, como descrito no exemplo anterior) e o total de horas influencia o valor do subtotal que o tutor receberá.


idTutor	idConteúdo	totalHoras	subTotal
1	0001	15	300,00
2	0002	20	900,00
3	0003	40	2.200,00

Tabela 3 – Tabela de banco de dados com dependência transitiva

Observar essas dependências e anomalias é importante, pois elas podem gerar problemas futuramente, quando as tabelas começam a ficar mais povoadas. Situações de duplicidade, por exemplo, podem ocorrer. Para isso, criaram-se “normas” para os dados, ou maneira de organizá-los para evitar esse tipo de situação. Daí vem a normalização de dados.

Normalização de dados

A técnica de normalização de banco de dados tem como objetivo proporcionar uma melhor reorganização dos dados, eliminando assim todo e qualquer tipo de redundância indesejável, presente na criação de um esquema de banco de dados (HENRIQUE NETO, 2015). Observe a tabela 4 e analise as informações que aparecem repetidas.



idTutor	idConteudo	conteúdo	totalHoras	subTotal
1	0001	Banco de dados	15	300,00
1	0002	Java	20	900,00
3	0003	Normalização	40	2.200,00
2	0002	Java	20	900,00
4	0003	Normalização	40	2.200,00

Tabela 4 – Tabela de banco de dados com redundância de dados

A repetição que você observou na tabela é mesmo necessária?

Por mais que os registros se refiram a aulas ou tutorias diferentes, com diferentes tutores, seria necessário repetir a descrição de conteúdo e a carga horária dele (“totalHoras”) tantas vezes, como acontece nas linhas 2 e 4, 3 e 5? Este portanto é um caso de redundância, que pode ser eliminado com normalização.

Ao se implementar um processo de normalização de dados em um banco de dados relacional, vislumbra-se ao final dessa prática **reduzir as duplicidades de dados (atributos multivalorados)**, além de qualquer tipo de dependência funcional e redundância de dados, os quais possam, de alguma forma, proporcionar a perda de informação.

O processo é composto por formas normais, que são regras para ajustar situações específicas e reestruturar as tabelas de banco de dados. Para que tal processo tenha êxito, deve-se aplicar continuamente as formas normais, em cascata, uma após a outra, ou seja, aplicando a primeira forma normal (1FN), a segunda forma normal (2FN), e assim por diante, totalizando 5FNs.

Na utilização do processo de normalização, considere sempre que, apesar de existirem 5FNs, usualmente em ambientes empresariais serão aplicadas somente 3FNs, ou seja, 1FN, 2FN e 3FN. Essas três regras resultarão em um esquema de

banco de dados classificado como satisfatório. As regras restantes serão utilizadas apenas em bases de dados que necessitem de tal aplicação.

É importante ressaltar que a normalização é um procedimento essencialmente lógico. Então, é preferível que ele aconteça no papel ou em diagrama entidade relacionamento (ERs), até que seja concluído para, a partir daí, ser aplicado em um possível banco de dados.

Primeira forma normal (1FN)

A primeira forma normal inicia-se por meio de uma tabela não normalizada de um banco de dados relacional, sendo modificada para a 1FN, isto é, uma tabela só estará normalizada na primeira forma quando não contiver mais tabelas aninhadas existentes, ou seja, quando essa tabela não puder ser desmembrada em outras tabelas.

A tabela 5 é uma tabela de banco de dados, que será chamada de “Alunos”. Suas colunas armazenam os alunos cadastrados em uma escola. Essa tabela não está normalizada na 1FN.

idAluno	nomeAluno	telefone1	telefone2	endereco
1	Galadriel	(51)XXXX-XXXX	(51)XXXX-XXXX	Rua 2, Teresópolis, Sergipe/SE - 85294-785
2	Elrond	(61)XXXX-XXXX	(61)XXXX-XXXX	Lot 5, Federativo, Rio de Janeiro/RJ - 25875-111
3	Gimli	(21)XXXX-XXXX	(21)XXXX-XXXX	Res 785, Costinha, São Paulo/SP - 74152-554
2	Elrond	(61)XXXX-XXXX	(61)XXXX-XXXX	Lot 5, Federativo, Rio de Janeiro/RJ - 25875-111
3	Gimli	(21)XXXX-XXXX	(21)XXXX-XXXX	Res 785, Costinha, São Paulo/SP - 74152-554

Tabela 5 – Tabela de banco de dados não normalizada

Perceba que, nas colunas “telefone1” e “telefone2”, poderiam existir mais do que dois telefones por aluno cadastrado. Deveria ser possível cadastrar mais telefones para contato, como, por exemplo, telefone comercial, residencial, celular e de contato.

Já a coluna “endereco” contém vários valores que podem e devem ser desassociados em colunas para contemplar a 1FN. É preciso dividir essa coluna até que ela se torne atômica (indivisível), ou seja, até que ela não possa mais ser desmembrada, de acordo com a ideia que o administrador do banco tem para esta tabela.

As tabelas a seguir mostram as alterações propostas para normalizar a tabela “Alunos” conforme a 1FN. Na tabela 6, vislumbra-se uma modificação na tabela de alunos para armazenar as informações do endereço do aluno de maneira atômica.

idAluno	nomeAluno	logradouro	bairro	cidade	estado	cep
1	Galadriel	Rua 2	Teresópolis	Porto Alegre	Porto Alegre	85294-785
2	Elrond	Lot 5	Leblon	Rio de Janeiro	Rio de Janeiro	25875-111
3	Gimli	Res 785	Costinha	São Paulo	São Paulo	74152-554

Tabela 6 – Itens da coluna “endereço” desassociados

Nota-se nessa tabela que as colunas “telefone1” e “telefone2” foram removidas da tabela original “Alunos”. Os telefones constarão em uma nova tabela, permitindo o cadastro de até mais que dois números para um aluno.

idTelefone	telefone	idAluno
1	(51)XXXX-XXXX	1
2	(61) XXXX-XXXX	2
3	(21) XXXX-XXXX	3
4	(51)XXXX-XXXX	1

Tabela 7 – Uma entidade nova criada a partir das colunas “telefone”

Para ajustar a entidade mostrada na tabela 6 à forma normal 1FN e à regra de negócio idealizada, foi criada mais uma entidade adicional de nome “telefone” (tabela 7) para armazenar os telefones, permitindo que o aluno cadastrado possa ter em seu cadastro mais de um número de telefone atrelado ao seu nome.

Um procedimento semelhante a esse poderia ter sido aplicado à coluna “endereço”, caso fosse desejado que o sistema permitisse cadastrar mais de um endereço por aluno. Se esse não for o caso, criar uma entidade “endereço” poderia levar a um relacionamento desnecessário, apenas prejudicando a *performance* da base de dados ao longo do tempo.

Segunda forma normal (2FN)

Para uma tabela estar em conformidade com a segunda forma normal 2FN, é preciso que ela esteja obrigatoriamente ajustada à primeira forma normal (1FN) e que todos os atributos não chave sejam dependentes da chave primária. Ao utilizar a regra 2FN, você estará evitando valores redundantes em sua base de dados, o que poderá causar uma baixa *performance* e o uso excessivo de recursos do banco de dados.

A tabela 8 é uma tabela de tutoria, que guarda informações do serviço de tutoria de cada tutor, apontando os conteúdos lecionados além do total de horas nessa tutoria. Essa tabela está normalizada conforme 1FN, mas não está normalizada conforme 2FN. Esse exemplo contém uma chave composta formada pelos atributos-chave “**idTutor**” e “**idConteudo**”. A coluna “valorHora” é relativa à informação de tutor – ou seja, registra o total de horas que o tutor fará com o conteúdo ministrado; a coluna do subtotal representa o valor total que o tutor receberá ao ministrar a disciplina. Assim, conclui-se que a coluna “idTutor” está fazendo alusão às colunas “valorHora” e “subTotal”.

Por outro lado, a coluna “totalHoras” é relativa à carga horária de cada conteúdo; assim, conclui-se que a coluna “idConteudo” está associada às colunas “conteudo” e “totalHoras”.

Esse comportamento não condiz com a regra 2FN, pois atributos não chave devem depender de uma única chave e não de somente parte dela, ou seja, os atributos que estiverem dentro da tabela e não são parte da chave

composta deverão depender da chave composta completa e não somente do “idTutor” ou do “idConteúdo”. Esse é um cuidado que se deve ter quando existe uma chave composta em alguma tabela do banco de dados.

idTutor	idConteúdo	conteúdo	totalHoras	valorHora	subTotal
1	0001	Banco de dados	15	20,00	300,00
1	0002	Java	20	45,00	900,00
3	0003	Normalização	40	55,00	2.200,00

Tabela 8 – Tabela que se encontra na 1FN, mas não na 2FN

Outro problema encontrado nesse exemplo é que a coluna “idConteúdo” faz referência às colunas “conteúdo” e “valorHora”, podendo, em algum momento, gerar uma duplicação de valores, pois, quando um novo tutor for cadastrado, ele receberá um conteúdo para lecionar com um valor de hora-aula estipulado (retorne à tabela 4 para compreender esse ponto).

Para não criar uma redundância em sua base de dados e ajustar uma melhor adequação à segunda forma normal, é preciso separar a tabela 8 em novas entidades. As colunas “idTutor” e suas dependentes formarão uma nova entidade, assim como as colunas “idConteúdo” e suas dependentes, estabelecendo assim um relacionamento por meio de chave estrangeira entre as tabelas. A tabela 9 é uma tabela de controle de horas e a tabela 10 representa o catálogo de conteúdo para este exemplo.

idTutor	valorHora	subTotal
1	15	20,00
2	20	45,00
3	40	55,00

Tabela 9 – Tabela apresentando uma entidade nova criada a partir da coluna “idTutor” e suas dependentes

idConteudo	conteudo	totalHoras
0001	Banco de dados	15
0002	Java	20
0003	Normalização	40

Tabela 10 – Tabela apresentando uma entidade nova criada a partir da coluna “idConteudo” e suas dependentes

Terceira forma normal (3FN)

Para que uma tabela possa estar ajustada na terceira forma normal (3FN), além de ser obrigatório obedecer à forma normal anterior (2FN), não pode existir nenhuma dependência transitiva, ou seja, nenhuma coluna não chave poderá depender de outra coluna que seja não chave.

Cumprindo as etapas anteriores, deve-se excluir ou remover as colunas que tenham valores obtidos da junção de outras colunas, seja por adição, multiplicação, subtração etc.

Na tabela 11, percebe-se que a coluna “subTotal” é a multiplicação das colunas “totalHoras” e “valorHora”. Então, para fazer a adequação à terceira forma normal, deve-se primeiro identificar todas as colunas que estão dependendo de outros atributos (que não são chave) e, a seguir, aplicar as técnicas indicadas para eliminar essa dependência. Para contornar esse problema, cria-se uma nova tabela contendo o “idTutor” como uma chave

estrangeira (como mostrado na tabela 12) e efetuam-se as operações algébricas diretamente no código, ou se cria uma *view* (contendo o resultado das operações algébricas) apenas para a consulta dessas informações (tabela 12).

idTutor	idConteudo	totalHoras	valorHora	subTotal
1	0001	15	20,00	300,00
2	0002	20	45,00	900,00
3	0003	40	55,00	2.200,00

Tabela 11 – Tabela que se encontra na 2FN, mas não na 3FN

idTutor	idConteudo	totalHoras	valorHora
1	0001	15	20,00
1	0002	20	45,00
3	0003	40	55,00

Tabela 12 – Normalizando conforme 3FN

valorTotalHoras	idTutor	subTotal
1	1	300,00
2	2	900,00
3	3	2.200,00

Tabela 13 – Normalizando conforme 3FN por meio de *view*

Forma normal de Boyce-Codd

Em alguns casos, pode ser usada a forma normal de Boyce-Codd (FNBC), que é uma variação ou aperfeiçoamento da 3FN (WELICH, 2018). Se for interessante o seu uso, após ajustar obrigatoriamente a tabela à norma 3FN, é aplicada a FNBC. Cabe salientar, entretanto, que a 3FN não precisa necessariamente estar conforme a FNBC.

Uma entidade está na FNBC quando todas as colunas determinantes forem chaves candidatas, ou seja, colunas as quais o desenvolvedor da entidade percebe que têm potencial para serem uma chave primária, mas não as utiliza como.


No exemplo a seguir será mostrada uma tabela não normalizada, mas com um potencial de normalização na FNBC (tabela 14).

idAluno	conteudo	tutor
1	Banco de dados	Galadriel
2	Java	Lea
3	Normalização	Legolas
4	Java	Ricardo
5	HTML	Gimli
6	Normalização	Thuor
7	Javascript	Claudio

Tabela 14 – Tabela não normalizada na FNBC

A tabela mostra que um determinado aluno aprenderá uma disciplina com um dado tutor e cada tutor ministrará uma disciplina, embora uma mesma disciplina possa ser lecionada por vários tutores.

Com essa disposição, são encontrados alguns problemas, como, por exemplo, a perda dos dados do tutor quando os dados de um aluno forem excluídos. Outro problema seria que, ao inserir um novo aluno ou tutor, poderá

se tornar obrigatório inserir dados que, de certa forma podem não estar  dispostos na entidade.

Para solucionar essa situação, mais uma vez, separa-se a tabela em duas, do seguinte modo (tabelas 15 e 16):

idAluno	conteudo
1	Banco de dados
2	Java
3	Normalização
4	Java
5	HTML
6	Normalização
7	Javascript

Tabela 15 – Normalização na FNBC: ID/conteúdo

conteudo	tutor
Banco de dados	Galadriel
Java	Lea
Normalização	Legolas
Java	Ricardo
HTML	Gimli
Normalização	Thuor
Javascript	Claudio

Tabela 16 – Normalização na FNBC: conteúdo/tutor

Com a geração dessas duas tabelas, é possível verificar que os problemas citados anteriormente foram solucionados por meio da aplicação da FNBC. Dessa forma, quando se inserir os dados de um aluno, caso seja um conteúdo diferente dos já inseridos, não será necessário omitir ou criar um nome de um tutor ao qual esse aluno não esteja vinculado.

Quarta forma normal (4FN)

Primeiramente, a 4FN segue o padrão das anteriores, que é de estar em acordo com sua forma normal anterior, ou seja, para aplicar a 4FN, é necessário ter aplicado a 3FN.

Uma tabela estará na quarta forma normal (4FN) quando não existirem nela dependências multivaloradas (mais de uma chave primária composta por atributos), diferentemente da chave composta, na qual se tem apenas duas chaves primárias.

A tabela 17 mostra dependências multivaloradas, ou seja, não se encontra ajustada conforme a 4FN, pois as três colunas são chaves primárias.

idTutor	idConteudo	codAluno
1	0001	1
2	0002	2
3	0003	3
4	0004	2
5	0005	1
6	0006	4

Tabela 17 – Tabela multivalorada

No exemplo da tabela 17, as colunas são chaves que representam conceitos diferentes, ou seja, realidades diferentes: “idTutor” refere-se a uma determinada tabela para tutores; “idConteúdo” refere-se ao conteúdo do curso; e “codAluno” refere-se a uma determinada tabela de alunos. Então, para realizar a transformação para 4FN, é preciso separar os seus atributos em duas entidades. Assim haverá duas entidades para realidades diferentes com seus atributos típicos (tabelas 18 e 19).

idTutor	idConteudo
1	0001
2	0002
3	0003
4	0004
5	0005
6	0006

Tabela 18 – Normalização conforme a 4FN

idTutor	codAluno
1	1
2	2
3	3
4	2
5	1
6	4

Tabela 19 – Normalização conforme a 4FN

Quinta forma normal (5FN)



A quinta forma normal será aplicada com pouca frequência, somente em casos particulares. Ela trata do conceito de junção, no qual se aplica a engenharia reversa em uma tabela. Esses casos particulares são os relacionamentos múltiplos, ternários, quaternários etc., nos quais o conteúdo de uma tabela não pode ser reconstruído a partir das entidades que foram decompostas para a normalização. Isso significa que, após uma tabela ser decomposta em outras menores, ela perderá o seu conteúdo inicial na tentativa recompô-la novamente. Portanto, uma tabela estará na 5FN quando o seu conteúdo não puder ser reconstruído a partir de entidades menores.

Na tabela 20 consta um relacionamento ternário, que associa três tabelas: conteúdo, aluno e matrícula. A linha destacada em vermelho representa o valor que está se repetindo com outro dado dessa mesma tabela.

idConteudo	codAluno	pedMatricula
0001	1	A2
0002	2	B3
0003	3	A2
0009	4	B3
0007	1	E4
0002	2	B3

Tabela 20 – Tabela não normalizada conforme 5FN

A seguir, veja a possibilidade de substituir esse relacionamento ternário por um binário para tentar enquadrar a tabela na 5FN.

Inicialmente, será retirado o dado que é repetido entre os dois relacionamentos: conteúdo e código do aluno (tabela 21).



idConteudo	codAluno
0001	1
0002	2
0003	3
0007	1
0009	4


Tabela 21 – Tabela conteúdo/código do aluno

Depois, deve-se criar uma relação do conteúdo com a matrícula (tabela 22).

idConteudo	pedMatricula
0001	A2
0002	B3
0003	A2
0007	E4
0009	B3

Tabela 22 – Tabela conteúdo/matricula

Em seguida, cria-se uma relação entre código do aluno e matrícula (tabela 23).



codAluno	pedMatricula
1	A2
2	B3
3	A2
1	E4
4	B3

Tabela 23 – Tabela código do aluno/matricula

Finalmente, apresenta-se a recomposição natural dos campos (tabela 24).

idConteudo	codAluno	pedMatricula
0001	1	A2
0002	2	B3
0003	3	A2
0007	1	E4
0009	4	B3

Tabela 24 – Normalizada na 5FN

Perceba que, quando aplicada a 5FN, os dados originais da tabela não permaneceram, o que quer dizer que não há uma dependência de junção entre elas.

Caso fosse possível retornar aos dados originais, representados na tabela 19, isso indicaria uma dependência de junção e a tabela não estaria na 5FN.

Vantagens e desvantagens em normalizar uma base de dados

Conclui-se, então, que normalizar um banco de dados pode parecer uma boa ideia, pois ele terá uma maior eficiência e rapidez. Estas bases, porém, só terão tais qualidades se forem bem desenhadas e estruturadas.

Uma base de dados funcionará melhor se for bem projetada e estruturada, de modo a armazenar e relacionar seus dados de logicamente.

Apesar de a normalização tornar os dados mais assertivos e intuitivos, ela também pode deixar brechas para que aconteçam erros, pois justamente essa divisão em inúmeras tabelas poderá deixar as consultas complexas e lentas, conforme seu uso. Portanto, quanto mais pessoas acessarem uma determinada aplicação, mais lenta será a sua resposta.

Observe agora algumas vantagens e desvantagens em normalizar uma base de dados, segundo o site Ptcomputador (2022).

Redução da duplicidade de dados

Normalizar uma base de dados reduz o tamanho dela e evita a duplicação de dados, garantindo que cada dado seja guardado apenas uma vez. As bases de dados podem ter uma quantidade significativa de informação, talvez milhões ou bilhões de fragmentos de dados, então é necessário reduzir essas informações redundantes e, com a normalização, esse problema é resolvido.

Organização de dados logicamente

As informações estão organizadas logicamente, do mesmo modo em que os objetos do mundo real que os dados representam são organizados. Essa organização, portanto, facilitará a projeção de um determinado banco de dados ou o

desenvolvimento de uma aplicação.



Os desenvolvedores de aplicações relatam que *softwares* que estão interligados com uma base de dados normalizada são mais fáceis de lidar do que aqueles interligados a uma base não normalizada.

Garantia da integridade referencial nos dados

A integridade referencial é a necessidade de ligações entre dados em tabelas unidas. Sem estas conexões entre as tabelas e os dados relacionados, poderá surgir uma inconsistência no banco de dados, fazendo com que as tabelas fiquem órfãs de conexão e consequentemente haja a perda de informações. Percebe-se então a necessidade de unir essas tabelas para evitar que informações sejam extintas.

Alta normalização, baixo desempenho

Uma base de dados demasiadamente normalizada, ou seja, com muitas tabelas e uniões entre elas, acabará se tornando mais lenta devido à alta normalização entre as tabelas. Quando uma quantidade maior de pessoas acessa esta aplicação, ela se torna mais lenta, havendo casos em que uma pequena “desnormalização” seja necessária, visando à obtenção de uma melhor *performance* da base de dados.

Necessidade de uma análise criteriosa para a normalização

Sempre que se pensa em uma base de dados, deve-se considerar uma análise altamente criteriosa, ou seja, normalizar uma base de dados poderá ser uma tarefa complexa e árdua. Deve-se ter em mente que, para criar bases de dados com

grande quantidade de informações, é preciso conhecer a regra de negócio, entender o que está acontecendo e o que pode acontecer e só então iniciar a criação do modelo lógico do banco.

Uma base de dados que não foi normalizada corretamente pode ter – e muito provavelmente terá – um baixo desempenho e armazenará as informações de modo ineficaz.

Prejuízos estruturais

Quando se eliminam dados que se repetem e suas dependências na entidade, o banco de dados tende a diminuir de tamanho, podendo incidir, porém, em outro erro, ou seja, pode-se agregar algum custo estrutural na base.

Um profissional deverá entender como normalizar uma entidade e conhecer suas técnicas de execução antes mesmo de começar essa tarefa. Ele poderá incorrer no erro de “supernormalizar” um banco de dados ao decompor suas tabelas. Se isso acontecer, pode-se criar consultas mais complexas ao associar estas entidades, criando assim um custo estrutural em sua base.

Que tal realizar um desafio?

Após um longo estudo sobre normalização, coloque em prática o que foi estudado, exercitando e aplicando o que foi aprendido.

Nesse desafio, você implementará, caso sejam necessárias, as três formas normais mais utilizadas (1FN, 2FN e 3FN), pois, como foi mencionado anteriormente, essas formas deixarão seu banco de dados já com uma classificação satisfatória.

Você foi contratado por uma determinada empresa de peças automotivas para resolver os problemas de *performance* e estruturação dos dados que estão ocorrendo.

Foi primeiramente diagnosticado que, por ter sido um sistema realizado às pressas para suprir as necessidades da empresa, o banco de dados não foi criado segundo os parâmetros necessários para que não ocorressem erros de inserção, remoção e duplicidade dos dados.

Dessa forma, será preciso normalizar as tabelas existentes no banco de dados. Aplique, então, inicialmente as três primeiras formas normais. Caso não seja necessário, aplique alguma das normalizações e justifique tal decisão.

codFunc	nomeFunc	telFunc	cargoFunc	salarioFunc	horasTrab
2545	Paolo	(xx)587469581 (xx)985858845	Estoquista	1.500,00	130
8921	Pedro	(xx)988431553 (xx)854145524	Mecânico	2.300,00	80
2545	Paolo	(xx)587469581 (xx)985858845	Estoquista	1.500,00	130
9782	Carla	(xx)789542359 (xx)687452325	Gerente	5.500,00	180
5648	Ana	(xx)965874541 (xx)741258963	Atendente	1.700,00	170

Tabela 25 – Tabela de desafio