

Elmasri • Navathe

Sistemas de banco de dados

6ª edição



Elmasri • Navathe

Sistemas de banco de dados

6ª edição

Ramez Elmasri

*Departamento de Ciência da Computação e Engenharia
Universidade do Texas em Arlington*

Shamkant B. Navathe

*Faculdade de Computação
Georgia Institute of Technology*

Tradução:

Daniel Vieira

Revisão técnica:

Enzo Seraphim

Thatyana de Faria Piola Seraphim

Professores Doutores do Instituto de Engenharia de Sistemas e
Tecnologias da Informação — Universidade Federal de Itajubá

PEARSON

abdr 
ASSOCIAÇÃO
BRASILEIRA
DE DIREITOS
REPROGRÁFICOS
Respeite o direito autoral

© 2011 by Pearson Education do Brasil.
© 2011, 2007, 2004, 2000, 1994 e 1989 Pearson Education, Inc.

Tradução autorizada a partir da edição original, em inglês, *Fundamentals of Database Systems*, 6th edition, publicada pela Pearson Education, Inc., sob o selo Addison-Wesley.

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Pearson Education do Brasil.

Diretor editorial: Roger Trimer
Gerente editorial: Sabrina Cairo
Supervisor de produção editorial: Marcelo França
Editora plena: Thelma Babaoka
Editora de texto: Sabrina Levensteinas
Preparação: Paula Brandão Perez Mendes
Revisão: Elisa Andrade Buzzo
Capa: Thyago Santos sobre o projeto original de Lou Gibbs/Getty Images
Projeto gráfico e diagramação: Globaltec Artes Gráficas Ltda.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Elmasri, Ramez

Sistemas de banco de dados / Ramez Elmasri e Shamkant B. Navathe ; tradução Daniel Vieira ; revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim. -- 6. ed. -- São Paulo : Pearson Addison Wesley, 2011.

Título original: Fundamentals of database systems.

ISBN 978-85-4301-381-7

1. Banco de dados I. Navathe, Shamkant B..II. Título.


10-11462

CDD-005.75

Índices para catálogo sistemático:

1. Banco de dados : Sistemas : Processamento de dados 005.75
2. Banco de dados : Fundamentos : Processamento de dados 005.75

3ª reimpressão – Julho 2014
Direitos exclusivos para a língua portuguesa cedidos à
Pearson Education do Brasil Ltda.,
uma empresa do grupo Pearson Education
Rua Nelson Francisco, 26
CEP 02712-100 – São Paulo – SP – Brasil
Fone: (11) 2178-8686 – Fax: (11) 2178-8688
vendas@pearson.com



Fundamentos de dependências funcionais e normalização para bancos de dados relacionais

15

Nos capítulos 3 a 6, apresentamos diversos aspectos do modelo relacional e as linguagens associadas a ele. Cada *esquema de relação* consiste em uma série de atributos, e o *esquema de banco de dados relacional* consiste em uma série de esquemas de relação. Até aqui, assumimos que os atributos são agrupados para formar um esquema de relação usando o bom senso do projetista de banco de dados ou mapeando um projeto de esquema de banco de dados com base no modelo de dados conceitual, como o modelo de dados ER ou ER Estendido (EER). Esses modelos fazem o projetista identificar os tipos de entidade e de relacionamento e seus respectivos atributos, o que leva a um agrupamento natural e lógico dos atributos em relações quando os procedimentos de mapeamento discutidos no Capítulo 9 são seguidos. Porém, ainda precisamos de algum modo formal de análise porque um agrupamento de atributos em um esquema de relação pode ser melhor do que outro. Ao discutir o projeto de banco de dados nos capítulos 7 a 10, não desenvolvemos nenhuma medida de adequação ou *boas práticas* para medir a qualidade do projeto, além da intuição do projetista. Neste capítulo, vamos discutir parte da teoria que foi desenvolvida com o objetivo de avaliar esquemas relacionais para a qualidade do projeto — ou seja, para medir formalmente por que um conjunto de agrupamentos de atributos em esquemas de relação é melhor do que outro.

Existem dois níveis em que podemos discutir as boas práticas de esquemas de relação. O primeiro é o **nível lógico** (ou **conceitual**) — como os usuários interpretam os esquemas de relação e o significado de seus atributos. Ter bons esquemas de relação nesse nível permite que os usuários entendam claramente o significado dos dados nas relações, e daí formulem suas consultas corretamente. O segundo é o **nível de implementação** (ou **armazenamento físico**) — como as tuplas em uma relação da base são armazenadas e atualizadas. Esse nível se aplica apenas a esquemas das relações da base — que serão fisicamente armazenadas como arquivos —, enquanto no nível lógico estamos interessados em esquemas de relações da base e visões (relações virtuais). A teoria de projeto de banco de dados relacional desenvolvida neste capítulo se aplica principalmente a *relações da base*, embora alguns critérios de adequação também se apliquem a visões, como mostra a Seção 15.1.

Assim como em muitos problemas de projeto, o projeto de banco de dados pode ser realizado usando duas técnicas: de baixo para cima (bottom-up) ou de cima para baixo (top-down). Uma **metodologia de projeto de baixo para cima** (também chamada *projeto por síntese*) considera os relacionamentos básicos *entre atributos individuais* como ponto de partida e os utiliza para construir esquemas de relação. Essa técnica não é muito popular na prática,¹ pois sofre

¹ Uma exceção em que essa técnica é usada na prática é baseada em um modelo chamado *modelo relacional binário*. Um exemplo é a metodologia NIAM (Verheijen e VanBekum, 1982).

do problema de ter que coletar um grande número de relacionamentos binários entre atributos como ponto de partida. Para situações práticas, é quase impossível capturar relacionamentos binários entre todos esses pares de atributos. Ao contrário, uma **metodologia de projeto de cima para baixo** (também chamada *projeto por análise*) começa com uma série de agrupamentos de atributos em relações que existem naturalmente juntas, por exemplo, em uma fatura, formulário ou relatório. As relações são então analisadas individual e coletivamente, levando a mais decomposição, até que todas as propriedades desejáveis sejam atendidas. A teoria descrita neste capítulo se aplica às técnicas de projeto de cima para baixo e de baixo para cima, mas é mais apropriada quando usada com a técnica de cima para baixo.

O projeto de banco de dados relacional por fim produz um conjunto de relações. Os objetivos implícitos da atividade de projeto são *preservação da informação* e *redundância mínima*. A informação é muito difícil de se quantificar — logo, consideramos a preservação de informação em matéria de manutenção de todos os conceitos, incluindo tipos de atributo, tipos de entidade e tipos de relacionamento, bem como os relacionamentos de generalização/especialização, que são descritos usando um modelo como o EER. Assim, o projeto relacional precisa preservar todos esses conceitos, que são capturados originalmente no projeto conceitual após o mapeamento do projeto conceitual para lógico. A minimização da redundância implica diminuir o armazenamento redundante da mesma informação e reduzir a necessidade de múltiplas atualizações para manter a consistência entre diversas cópias da mesma informação, em resposta a eventos do mundo real que exijam fazer uma atualização.

Começamos este capítulo discutindo informalmente alguns critérios para esquemas de relação bons e ruins na Seção 15.1. Na Seção 15.2, definimos o conceito de *dependência funcional*, uma restrição formal entre os atributos que é a principal ferramenta para medir formalmente a adequação dos agrupamentos de atributo em esquemas de relação. Na Seção 15.3, vamos discutir as formas normais e o processo de normalização usando dependências funcionais. As formas normais sucessivas são definidas para atender a um conjunto de restrições desejáveis, expressas com dependências funcionais. O procedimento de normalização consiste em aplicar uma série de testes às relações para atender a esses requisitos cada vez mais rígidos e decompor as relações quando necessário. Na Seção 15.4, discutimos definições mais gerais das formas normais, que podem ser aplicadas diretamente a qualquer projeto dado e não exigem análise e normalização passo a passo. As seções 15.5 a 15.7 discutem outras formas normais, até a

quinta forma normal. Na Seção 15.6, apresentamos a dependência multivalorada (MVD), seguida pela dependência de junção (DJ) na Seção 15.7. No final há um resumo do capítulo.

O Capítulo 16 continuará o desenvolvimento da teoria relacionada ao projeto de bons esquemas relacionais. Discutimos as propriedades desejáveis da decomposição relacional — propriedade de junção não aditiva e propriedade de preservação da dependência funcional. Um algoritmo geral testa se uma decomposição tem ou não a propriedade de junção não aditiva (ou *sem perdas*) (o Algoritmo 16.3 também é apresentado). Depois, abordamos as propriedades das dependências funcionais e o conceito de uma cobertura mínima de dependências. Consideramos a técnica de baixo para cima para o projeto de banco de dados que consiste em um conjunto de algoritmos para projetar relações em uma forma normal desejada. Esses algoritmos consideram como entrada determinado conjunto de dependências funcionais e alcançam um projeto relacional em uma forma normal de destino, enquanto aderem às propriedades desejáveis acima. No Capítulo 16, também definimos outros tipos de dependências que melhoram ainda mais a avaliação das *boas práticas* de esquemas de relação.

Se o Capítulo 16 não for incluído no curso, recomendamos uma rápida introdução às propriedades desejáveis de decomposição e à discussão da Propriedade NJB da Seção 16.2.

15.1 Diretrizes de projeto informais para esquemas de relação

Antes de discutirmos a teoria formal do projeto de banco de dados relacional, vamos abordar quatro *diretrizes informais* que podem ser usadas como *medidas para determinar a qualidade* de projeto do esquema da relação:

- Garantir que a semântica dos atributos seja clara no esquema.
- Reduzir a informação redundante nas tuplas.
- Reduzir os valores NULL nas tuplas.
- Reprovar a possibilidade de gerar tuplas falsas.

Essas medidas nem sempre são independentes uma da outra, conforme veremos.

15.1.1 Comunicando uma semântica clara aos atributos nas relações

Sempre que agrupamos atributos para formar um esquema de relação, consideramos que aqueles atributos pertencentes a uma relação têm certo significado no mundo real e uma interpretação apro-

priada associada a eles. A **semântica** de uma relação refere-se a seu significado resultante da interpretação dos valores de atributo em uma tupla. No Capítulo 3, discutimos como uma relação pode ser interpretada como um conjunto de fatos. Se o projeto conceitual descrito nos capítulos 7 e 8 for feito cuidadosamente e o procedimento de mapeamento do Capítulo 9 for seguido de maneira sistemática, o projeto do esquema relacional deverá ter um significado claro.

Em geral, quanto mais fácil for explicar a semântica da relação, melhor será o projeto do esquema de relação. Para ilustrar isso, considere a Figura 15.1, uma versão simplificada do esquema de banco de dados relacional EMPRESA da Figura 3.5, e a Figura 15.2, que apresenta um exemplo de estados de relação preenchidos desse esquema. O significado do esquema de relação FUNCIONARIO é muito simples: cada tupla representa um funcionário, com valores para o nome do funcionário (Fnome), número do Cadastro de Pessoa Física (Cpf), data de nascimento (Datanasc), endereço (Endereco) e o número do departamento para o qual o funcionário trabalha (Dnumero). O atributo Dnumero é uma chave estrangeira que representa um *relacionamento implícito* entre FUNCIONARIO e DEPARTAMENTO. A semântica dos esquemas DEPARTAMENTO e PROJETO é muito simples: cada tupla DEPARTAMENTO

representa uma entidade de departamento, e cada tupla PROJETO representa uma entidade de projeto. O atributo Cpf_gerente de DEPARTAMENTO relaciona um departamento ao funcionário que é seu gerente, enquanto Dnum de PROJETO relaciona um projeto a seu departamento de controle; ambos são atributos de chave estrangeira. A facilidade com que o significado dos atributos de uma relação pode ser explicado é uma *medida informal* de quão bem a relação está projetada.

A semântica dos outros dois esquemas de relação da Figura 15.1 é ligeiramente mais complexa. Cada tupla em LOCALIZACAO_DEP gera um número de departamento (Dnumero) e *um dos* locais do departamento (Dlocalizacao). Cada tupla em TRABALHA_EM gera um número de Cadastro de Pessoa Física (Cpf), o número de projeto de *um dos* projetos em que o funcionário trabalha (Projnumero) e o número de horas por semana que o funcionário trabalha nesse projeto (Horas). Porém, os dois esquemas têm uma interpretação bem definida e não ambígua. O esquema LOCALIZACAO_DEP representa um atributo multivalorado de DEPARTAMENTO, enquanto TRABALHA_EM representa um relacionamento M:N entre FUNCIONARIO e PROJETO. Logo, todos os esquemas de relação na Figura 15.1 podem ser considerados fáceis de explicar e, portanto, bons do ponto de vista de ter uma semântica clara. Assim, podemos formular as seguintes diretrizes de projeto informal:

Diretriz 1

Projete um esquema de relação de modo que seja fácil explicar seu significado. Não combine atributos de vários tipos de entidade e de relacionamento em uma única relação. Intuitivamente, se um esquema de relação corresponde a um tipo de entidade ou um tipo de relacionamento, é simples interpretar e explicar seu significado. Caso contrário, se a relação corresponder a uma mistura de várias entidades e relacionamentos, haverá ambiguidades semânticas e a relação não poderá ser explicada com facilidade.

Exemplos de violação da diretriz 1. Os esquemas de relação das figuras 15.3(a) e 15.3(b) também têm semântica clara. (O leitor deve ignorar as linhas sob as relações por enquanto; elas são usadas para ilustrar a notação da dependência funcional, discutida na Seção 15.2.) Uma tupla no esquema de relação FUNC_DEP na Figura 15.3(a) representa um único funcionário, mas inclui informações adicionais — a saber, o nome (Dnome) do departamento para o qual o funcionário trabalha e o número do Cadastro de Pessoa Física (Cpf_gerente) do gerente de departamento. Para a relação FUNC_PROJ da Figura

FUNCIONARIO

Fnome	Cpf	Datanasc	Endereco	Dnumero
-------	-----	----------	----------	---------

ChE

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente
-------	---------	-------------

ChP

DEP_LOCALIZACAO

Dnumero	Dlocal
---------	--------

ChP

PROJETO

Projnome	Projnumero	Projlocal	Dnum
----------	------------	-----------	------

ChP

TRABALHA_EM

Cpf	Projnumero	Horas
-----	------------	-------

ChP

Figura 15.1

Um esquema de banco de dados relacional EMPRESA simplificado.

FUNCIONARIO

Fnome	Cpf	Datanasc	Endereco	Dnumero
Silva, Joao B.	12345678966	09-01-1965	Rua das Flores, 751, São Paulo, SP	5
Wong, Fernando T.	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	5
Zelaya, Alice J.	99988777767	19-01-1968	Rua Souza Lima, 35, Curitiba, PR	4
Souza, Jennifer S.	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	4
Lima, Ronaldo K.	66688444476	15-09-1962	Rua Rebouças, 65, Piracicaba, SP	5
Leite, Joice A.	45345345376	31-07-1972	Av. Lucas Obes, 74, São Paulo, SP	5
Pereira, André V.	98798798733	29-03-1969	Rua Timbira, 35, São Paulo, SP	4
Brito, Jorge E.	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	1

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente
Pesquisa	5	33344555587
Administração	4	98765432168
Matriz	1	88866555576

LOCALIZACAO_DEP

Dnumero	Dlocal
1	São Paulo
4	Mauá
5	Santo André
5	Itu
5	São Paulo

TRABALHA_EM

Cpf	Projnumero	Horas
12345678966	1	32,5
12345678966	2	7,5
66688444476	3	40,0
45345345376	1	20,0
45345345376	2	20,0
33344555587	2	10,0
33344555587	3	10,0
33344555587	10	10,0
33344555587	20	10,0
99988777767	30	30,0
99988777767	10	10,0
98798798733	10	35,0
98798798733	30	5,0
98765432168	30	20,0
98765432168	20	15,0
88866555576	20	NULL

PROJETO

Projnome	Projnumero	Projlocal	Dnum
ProdutoX	1	Santo André	5
ProdutoY	2	Itu	5
ProdutoZ	3	São Paulo	5
informatização	10	Mauá	4
Reorganização	20	São Paulo	1
Novosbenefícios	30	Mauá	4

Figura 15.2

Exemplo de estado de banco de dados para o esquema relacional da Figura 15.1.

15.3(b), cada tupla relaciona um funcionário a um projeto, mas também inclui o nome do funcionário (Fnome), nome do projeto (Projnome) e local do projeto (Projlocal). Embora não haja nada logicamente errado logicamente com essas duas relações, elas violam a Diretriz 1 ao misturar atributos de entidades distintas do mundo real: FUNC_DEP mistura atributos

dos funcionários e departamentos, e FUNC_PROJ mistura atributos de funcionários e projetos e o relacionamento TRABALHA_EM. Logo, elas se saem mal contra a medida de qualidade de projeto citado. Elas podem ser usadas como visões, mas causam problemas quando utilizadas como relações da base, conforme discutiremos na próxima seção.

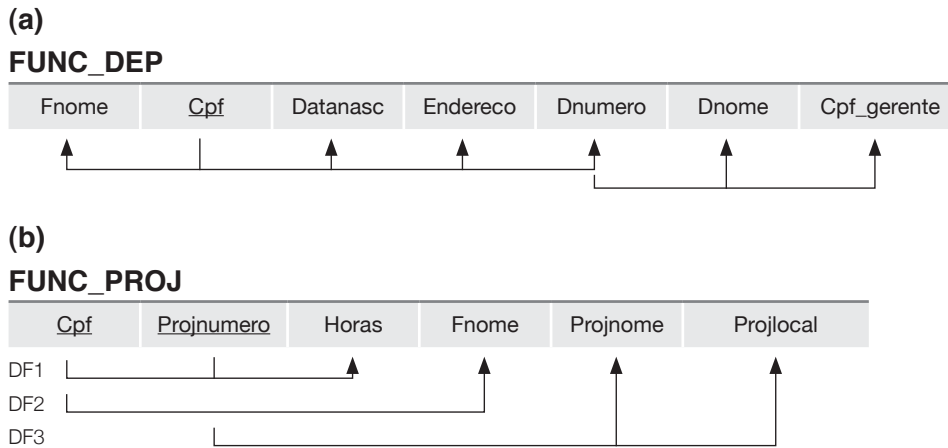


Figura 15.3

Dois esquemas de relação sofrendo de anomalias de atualização. (a) FUNC_DEP e (b) FUNC_PROJ.

15.1.2 Informação redundante nas tuplas e anomalias de atualização

Um objetivo do projeto de esquema é minimizar o espaço de armazenamento usado pelas relações (e, portanto, pelos arquivos correspondentes). O agrupamento de atributos em esquemas de relação tem um efeito significativo no espaço de armazenamento. Por exemplo, compare o espaço usado pelas duas relações da base FUNCIONARIO e DEPARTAMENTO da Figura 15.2 com o da relação da base FUNC_DEP da Figura 15.4, que é o resultado da aplicação da operação JUNCAO NATURAL em FUNCIONARIO e DEPARTAMENTO. Em FUNC_DEP, os valores de atributo pertencentes a determinado departamento (Dnumero, Dnome, Cpf_gerente) são repetidos para *cada funcionário que trabalha para esse departamento*. Ao contrário, a informação de cada departamento aparece apenas uma vez na relação DEPARTAMENTO da Figura 15.2. Somente o número do departamento (Dnumero) é repetido na relação FUNCIONARIO para cada funcionário que trabalha nesse departamento, como uma chave estrangeira. Comentários semelhantes se aplicam à relação FUNC_PROJ (ver Figura 15.4), que aumenta a relação TRABALHA_EM com atributos adicionais de FUNCIONARIO e PROJETO.

O armazenamento de junções naturais de relações da base leva a um problema adicional conhecido como **anomalias de atualização**. Estas podem ser classificadas em anomalias de inserção, anomalias de exclusão e anomalias de modificação.²

Anomalias de inserção. As anomalias de inserção podem ser diferenciadas em dois tipos, ilustrados pelos seguintes exemplos baseados na relação FUNC_DEP:

- Para inserir uma nova tupla de funcionário em FUNC_DEP, temos de incluir ou os valores de atributo do departamento para o qual o funcionário trabalha ou NULLs (se o funcionário ainda não trabalha para nenhum departamento). Por exemplo, para inserir uma nova tupla para um funcionário que trabalha no departamento 5, temos de inserir todos os valores de atributo do departamento 5 corretamente, de modo que eles sejam *coerentes* com os valores correspondentes para o departamento 5 em outras duplas de FUNC_DEP. No projeto da Figura 15.2, não temos de nos preocupar com esse problema de coerência, pois entramos apenas com o número do departamento na tupla do funcionário. Todos os outros valores de atributo do departamento 5 são registrados apenas uma vez no banco de dados, como uma única tupla na relação DEPARTAMENTO.
- É difícil inserir um novo departamento que ainda não tenha funcionários na relação FUNC_DEP. A única maneira de fazer isso é colocar valores NULL nos atributos para funcionário. Isso viola a integridade de entidade para FUNC_DEP, porque Cpf é sua chave primária. Além do mais, quando o primeiro funcionário é atribuído a esse departamento, não precisamos mais dessa tupla com valores NULL. Esse problema não ocorre no projeto da Figura 15.2, visto que um departamento é inserido na relação DEPARTAMENTO independentemente de haver ou não funcionários trabalhando para ele, e sempre que um funcionário é atribuído a esse departamento, uma tupla correspondente é inserida em FUNCIONARIO.

² Essas anomalias foram identificadas por Codd (1972a) para justificar a necessidade de normalização das relações, conforme discutiremos na Seção 15.3.

Redundância

FUNC_DEP						
Fnome	Cpf	Datanasc	Endereco	Dnumero	Dnome	Cpf_gerente
Silva, João B.	12345678966	09-01-1965	Rua das Flores, 751, São Paulo, SP	5	Pesquisa	33344555587
Wong, Fernando T.	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	5	Pesquisa	33344555587
Zelaya, Alice J.	99988777767	19-01-1968	Rua Souza Lima, 35, Curitiba, PR	4	Administração	98765432168
Souza, Jennifer S.	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	4	Administração	98765432168
Lima, Ronaldo K.	66688444476	15-09-1962	Rua Rebouças, 65, Piracicaba, SP	5	Pesquisa	33344555587
Leite, Joice A.	45345345376	31-07-1972	Av. Lucas Obes, 74, São Paulo, SP	5	Pesquisa	33344555587
Pereira, André V.	98798798733	29-03-1969	Rua Timbira, 35, São Paulo, SP	4	Administração	98765432168
Brito, Jorge E.	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	1	Matriz	88866555576

RedundânciaRedundância

Cpf	Projnumero	Horas	Fnome	Projnome	Projlocal
12345678966	1	32,5	Silva, João B.	ProdutoX	Santo André
12345678966	2	7,5	Silva, João B.	ProdutoY	Itu
66688444476	3	40,0	Lima, Ronaldo K.	ProdutoZ	São Paulo
45345345376	1	20,0	Leite, Joice A.	ProdutoX	Santo André
45345345376	2	20,0	Leite, Joice A.	ProdutoY	Itu
33344555587	2	10,0	Wong, Fernando T.	ProdutoY	Itu
33344555587	3	10,0	Wong, Fernando T.	ProdutoZ	São Paulo
33344555587	10	10,0	Wong, Fernando T.	Informatização	Mauá
33344555587	20	10,0	Wong, Fernando T.	Reorganização	São Paulo
99988777767	30	30,0	Zelaya, Alice J.	Novosbenefícios	Mauá
99988777767	10	10,0	Zelaya, Alice J.	Informatização	Mauá
98798798733	10	35,0	Pereira, André V.	Informatização	Mauá
98798798733	30	5,0	Pereira, André V.	Novosbenefícios	Mauá
98765432168	30	20,0	Souza, Jennifer S.	Novosbenefícios	Mauá
98765432168	20	15,0	Souza, Jennifer S.	Reorganização	São Paulo
88866555576	20	Null	Brito, Jorge E.	Reorganização	São Paulo

Figura 15.4
Exemplos de estados para FUNC_DEP e FUNC_PROJ resultando na aplicação da JUNÇÃO NATURAL às relações da Figura 15.2. Estas podem ser armazenadas como relações da base por questões de desempenho.

Anomalias de exclusão. O problema das anomalias de exclusão está relacionado à segunda situação de anomalia de inserção que acabamos de discutir. Se excluirmos de FUNC_DEP uma tupla de funcionário que represente o último funcionário trabalhando para determinado departamento, a informação referente a esse departamento se perde do banco de dados. Esse problema não ocorre no banco de dados da Figura 15.2, pois as tuplas de DEPARTAMENTO são armazenadas separadamente.

Anomalias de modificação. Em FUNC_DEP, se mudarmos o valor de um dos atributos de determinado

departamento — digamos, o gerente do departamento 5 —, temos de atualizar as tuplas de *todos* os funcionários que trabalham nesse departamento; caso contrário, o banco de dados ficará incoerente. Se deixarmos de atualizar algumas tuplas, o mesmo departamento mostrará dois valores diferentes para o gerente em diferentes tuplas de funcionário, o que seria errado.³

É fácil ver que essas três anomalias são indesejáveis e causam dificuldades para manter a coerência dos dados, bem como exigem atualizações desnecessárias que podem ser evitadas; logo, podemos declarar a próxima diretriz como se segue.

³ Este não é tão sério quanto os outros problemas, pois todas as tuplas podem ser atualizadas por uma única consulta SQL.

Diretriz 2

Projete os esquemas de relação da base de modo que nenhuma anomalia de inserção, exclusão ou modificação esteja presente nas relações. Se houver alguma anomalia,⁴ anote-as claramente e cuide para que os programas que atualizam o banco de dados operem corretamente.

A segunda diretriz é coerente com e, de certa forma, é uma reafirmação da primeira diretriz. Também podemos ver a necessidade de uma técnica mais formal para avaliar se um projeto atende a essas diretrizes. As seções 15.2 a 15.4 oferecem esses conceitos formais necessários. É importante observar que essas diretrizes às vezes *podem ter de ser violadas* a fim de *melhorar o desempenho* de certas consultas. Se FUNC_DEP for usado como uma relação armazenada (conhecida de outra forma como uma *visão materializada*) além das relações da base de FUNCIONARIO e DEPARTAMENTO, as anomalias em FUNC_DEP precisam ser observadas e consideradas (por exemplo, usando triggers ou procedimentos armazenados que fariam atualizações automáticas). Desse modo, sempre que a relação da base é atualizada, não ficamos com incoerências. Em geral, é aconselhável usar relações da base sem anomalias e especificar visões que incluem as junções para reunir os atributos frequentemente referenciados nas consultas importantes.

15.1.3 Valores NULL nas tuplas

Em alguns projetos de esquema, podemos agrupar muitos atributos em uma relação 'gorda'. Se muitos dos atributos não se aplicarem a todas as tuplas na relação, acabamos com muitos NULLs nessas tuplas. Isso pode desperdiçar espaço no nível de armazenamento e também ocasionar problemas com o conhecimento do significado dos atributos e com a especificação de operações JUNÇÃO no nível lógico.⁵ Outro problema com NULLs é como considerá-los quando operações de agregação como CONTA ou SOMA são aplicadas. Operações SELEÇÃO e JUNÇÃO envolvem comparações; se valores NULL estiverem presentes, os resultados podem se tornar imprevisíveis.⁶ Além do mais, os NULLs podem ter várias interpretações, como as seguintes:

- O atributo *não se aplica* a essa tupla. Por exemplo, Estado_visto pode não se aplicar a alunos do Brasil.

- O valor do atributo para essa tupla é *desconhecido*. Por exemplo, a Data_nascimento pode ser desconhecida para um funcionário.
- O valor é *conhecido, mas ausente*; ou seja, ele ainda não foi registrado. Por exemplo, o Numero_telefone_residencial para um funcionário pode existir, mas ainda não estar disponível e registrado.

Ter a mesma representação para todos os NULLs compromete os diferentes significados que eles podem ter. Portanto, podemos declarar outra diretriz.

Diretriz 3

Ao máximo possível, evite colocar atributos em uma relação da base cujos valores podem ser NULL com frequência. Se os NULLs forem inevitáveis, garanta que eles se apliquem apenas em casos excepcionais, e não à maioria das tuplas na relação.

Usar o espaço de modo eficaz e evitar junções com valores NULL são os dois critérios prioritários que determinam a inclusão das colunas que podem ter NULLs em uma relação ou que podem ter uma relação separada para essas colunas (com as colunas de chave apropriadas). Por exemplo, se apenas 15 por cento dos funcionários têm escritórios individuais, há pouca justificativa para incluir um atributo Numero_escritorio na relação FUNCIONARIO. Em vez disso, uma relação FUNC_ESCRITORIO (Fcpf, Numero_escritorio) pode ser criada para incluir tuplas apenas para funcionários com escritórios individuais.

15.1.4 Geração de tuplas falsas

Considere os esquemas de duas relações FUNC_LOCAL e FUNC_PROJ1 da Figura 15.5(a), que podem ser usados no lugar da única relação FUNC_PROJ da Figura 15.3(b). Uma tupla em FUNC_LOCAL significa que o funcionário cujo nome é Fnome trabalha em *algum projeto* cujo local é Projlocal. Uma tupla em FUNC_PROJ1 refere-se ao fato de o funcionário cujo número de Cadastro de Pessoa Física é Cpf trabalhar Horas por semana no projeto cujo nome, número e localização são Projnome, Projnumero e Projlocal. A Figura 15.5(b) mostra os estados da relação de FUNC_LOCAL e FUNC_PROJ1 correspondentes à relação FUNC_PROJ da Figura 15.4, que são obtidos aplicando as operações PROJETO (π) apropriadas a FUNC_PROJ (ignore as linhas tracejadas na Figura 15.5(b) por enquanto).

⁴ Outras considerações de aplicação podem determinar e tornar certas anomalias inevitáveis. Por exemplo, a relação FUNC_DEP pode corresponder a uma consulta ou a um relatório que é exigido com frequência.

⁵ Isso porque as junções interna e externa produzem diferentes resultados quando NULLs são envolvidos nas junções. Assim, os usuários precisam estar cientes dos diferentes significados dos vários tipos de junções. Embora isso seja razoável para usuários sofisticados, pode ser difícil para outros.

⁶ Na Seção 5.5.1, apresentamos comparações envolvendo valores NULL onde o resultado (na lógica de três valores) é TRUE, FALSE e UNKNOWN.

(a)

FUNC_LOCAL

Fnome	Projlocal
ChP	

(b)

FUNC_PROJ1

Cpf	Projnumero	Horas	Projnome	Projlocal
ChP				

(c)

FUNC_LOCAL

Fnome	Projlocal
Silva, João B.	Santo André
Silva, João B.	Itu
Lima, Ronaldo K.	São Paulo
Leite, Joice A.	Santo André
Leite, Joice A.	Itu
Wong, Fernando T.	Itu
Wong, Fernando T.	São Paulo
Wong, Fernando T.	Mauá
Zelaya, Alice J.	Mauá
Pereira, André V.	Mauá
Souza, Jennifer S.	Mauá
Souza, Jennifer S.	São Paulo
Brito, Jorge E.	São Paulo

FUNC_PROJ1

Cpf	Projnumero	Horas	Projnome	Projlocalizacao
12345678966	1	32,5	ProdutoX	Santo André
12345678966	2	7,5	ProdutoY	Itu
66688444476	3	40,0	ProdutoZ	São Paulo
45345345376	1	20,0	ProdutoX	Santo André
45345345376	2	20,0	ProdutoY	Itu
33344555587	2	10,0	ProdutoY	Itu
33344555587	3	10,0	ProdutoZ	São Paulo
33344555587	10	10,0	Computadorização	Mauá
33344555587	20	10,0	Reorganização	São Paulo
99988777767	30	30,0	Novosbenefícios	Mauá
99988777767	10	10,0	Computadorização	Mauá
98765432168	10	35,0	Computadorização	Mauá
98765432168	30	5,0	Novosbenefícios	Mauá
98765432168	30	20,0	Novosbenefícios	Mauá
98798798733	20	15,0	Reorganização	São Paulo
88866555576	20	NULL	Reorganização	São Paulo

Figura 15.5

Projeto particularmente fraco para a relação FUNC_PROJ da Figura 15.3(b). (a) Esquemas de duas relações FUNC_LOCAL e FUNC_PROJ1. (b) Resultado da projeção da extensão de FUNC_PROJ da Figura 15.4 para as relações FUNC_LOCAL e FUNC_PROJ1.

Suponha que usamos FUNC_PROJ1 e FUNC_LOCAL como relações da base em vez de FUNC_PROJ. Isso produz um projeto de esquema particularmente ruim, pois não podemos recuperar a informação que havia originalmente em FUNC_PROJ de FUNC_PROJ1 e FUNC_LOCAL. Se tentarmos uma operação JUNÇÃO NATURAL sobre FUNC_PROJ1 e FUNC_LOCAL, o resultado produz muito mais tuplas do que o conjunto original de tuplas em FUNC_PROJ. Na Figura 15.6, mostramos o resultado da aplicação da junção apenas às tuplas *acima* das linhas tracejadas da Figura 15.5(b) (para reduzir o tamanho da relação resultante). Tuplas adicionais, que não estavam em FUNC_PROJ, são chamadas de **tuplas falsas**, pois representam informação falsa, que não é válida. As tuplas falsas são marcadas com asteriscos (*) na Figura 15.6.

A decomposição de FUNC_PROJ em FUNC_LOCAL e FUNC_PROJ1 é indesejável porque, quando as juntamos (JUNÇÃO) de volta usando JUNÇÃO NATURAL, não obtemos a informação original correta. Isso porque, neste caso, Projlocal é o atributo que relaciona FUNC_LOCAL e FUNC_PROJ1, e Projlocal não é a chave primária nem uma chave estrangeira em FUNC_LOCAL ou FUNC_PROJ1. Agora podemos declarar informalmente outra diretriz de projeto.

Diretriz 4

Projete esquemas de relação de modo que possam ser unidos com condições de igualdade sobre os atributos que são pares relacionados corretamente (chave primária, chave estrangeira) de um modo que garanta que nenhuma tupla falsa será gerada. Evi-

Cpf	Projnumero	Horas	Projnome	Projlocal	Fnome
12345678966	1	32,5	ProdutoX	Santo André	Silva, João B.
*12345678966	1	32,5	ProdutoX	Santo André	Leite, Joice A.
12345678966	2	7,5	ProdutoY	Itu	Silva, João B.
*12345678966	2	7,5	ProdutoY	Itu	Leite, Joice A.
*12345678966	2	7,5	ProdutoY	Itu	Wong, Fernando T.
66688444476	3	40,0	ProdutoZ	São Paulo	Lima, Ronaldo K.
*66688444476	3	40,0	ProdutoZ	São Paulo	Wong, Fernando T.
*45345345376	1	20,0	ProdutoX	Santo André	Silva, João B.
45345345376	1	20,0	ProdutoX	Santo André	Leite, Joice A.
*45345345376	2	20,0	ProdutoY	Itu	Silva, João B.
45345345376	2	20,0	ProdutoY	Itu	Leite, Joice A.
*45345345376	2	20,0	ProdutoY	Itu	Wong, Fernando T.
*33344555587	2	10,0	ProdutoY	Itu	Silva, João B.
*33344555587	2	10,0	ProdutoY	Itu	Leite, Joice A.
33344555587	2	10,0	ProdutoY	Itu	Wong, Fernando T.
*33344555587	3	10,0	ProdutoZ	São Paulo	Lima, Ronaldo K.
33344555587	3	10,0	ProdutoZ	São Paulo	Wong, Fernando T.
33344555587	10	10,0	Computadorização	Mauá	Wong, Fernando T.
*33344555587	20	10,0	Reorganização	São Paulo	Lima, Ronaldo K.
33344555587	20	10,0	Reorganização	São Paulo	Wong, Fernando T.

*
*
*

Figura 15.6

Resultado da aplicação do NATURAL JOIN às tuplas acima das linhas tracejadas em FUNC_PROJ1 e FUNC_LOCAL da Figura 15.5. As tuplas falsas geradas são marcadas com asteriscos.

te relações com atributos correspondentes que não sejam combinações (chave estrangeira, chave primária), pois a junção sobre tais atributos pode produzir tuplas falsas.

Essa diretriz informal, obviamente, precisa ser declarada de maneira mais formal. Na Seção 16.2, discutiremos uma condição formal chamada propriedade de junção não aditiva (ou sem perda), que garante que certas junções não produzam tuplas falsas.

15.1.5 Resumo e discussão das diretrizes de projeto

Nas seções 15.1.1 a 15.1.4, discutimos informalmente situações que levam a esquemas de relação problemáticas e propusemos diretrizes informais para um bom projeto relacional. Os problemas que apontamos, que podem ser detectados sem ferramentas de análise adicionais, são os seguintes:

- Anomalias que causam trabalho redundante durante a inserção e modificação em uma relação, e que podem causar perda acidental

de informação durante a exclusão de uma relação.

- Desperdício de espaço de armazenamento devido a NULLs e a dificuldade de realizar seleções, operações de agregação e junções por causa de valores NULL.
- Geração de dados inválidos e falsos durante as junções em relações da base com atributos correspondentes que possam não representar um relacionamento apropriado (chave estrangeira, chave primária).

No restante deste capítulo, apresentamos os conceitos formais e a teoria que pode ser usada para definir os pontos *positivos* e *negativos* dos esquemas de relação *individuais* com mais precisão. Primeiro, discutimos a dependência funcional como uma ferramenta para análise. Depois, especificamos as três formas normais e a Forma Normal de Boyce-Codd (FNBC) para esquemas de relação. A estratégia para alcançar um bom projeto é decompor de maneira correta uma relação mal projetada. Também intro-

duzimos rapidamente formas normais adicionais que lidam com dependências adicionais. No Capítulo 16, discutimos as propriedades da decomposição com detalhes, e oferecemos algoritmos que projetam relações de baixo para cima, usando as dependências funcionais como ponto de partida.

15.2 Dependências funcionais

Até aqui, lidamos com as medidas informais do projeto de banco de dados. Agora, vamos introduzir uma ferramenta formal para a análise de esquemas relacionais, que nos permite detectar e descrever alguns dos problemas mencionados em termos precisos. O conceito isolado mais importante na teoria de projeto de esquema relacional é o de uma dependência funcional. Nesta seção, definimos formalmente o conceito e, na Seção 15.3, veremos como ele pode ser usado para definir formas normais para esquemas de relação.

15.2.1 Definição de dependência funcional

Uma dependência funcional é uma restrição entre dois conjuntos de atributos do banco de dados. Suponha que nosso esquema de banco de dados relacional tenha n atributos A_1, A_2, \dots, A_n . Vamos pensar no banco de dados inteiro sendo descrito por um único esquema de relação **universal** $R = \{A_1, A_2, \dots, A_n\}$.⁷ Não queremos dizer que realmente armazenaremos o banco de dados como uma única tabela universal — usamos esse conceito apenas no desenvolvimento da teoria formal das dependências de dados.⁸

Definição. Uma **dependência funcional**, indicada por $X \rightarrow Y$, entre dois conjuntos de atributos X e Y que são subconjuntos de R , especifica uma *restrição* sobre possíveis tuplas que podem formar um estado de relação r de R . A restrição é que, para quaisquer duas tuplas t_1 e t_2 em r que tenham $t_1[X] = t_2[X]$, elas também devem ter $t_1[Y] = t_2[Y]$.

Isso significa que os valores do componente Y de uma tupla em r dependem dos (ou são *determinados pelos*) valores do componente X ; como alternativa, os valores do componente X de uma tupla *determinam* exclusivamente (ou **funcionalmente**) os valores do componente Y . Também dizemos que existe uma dependência funcional de X para Y , ou que Y é **funcionalmente dependente** de X . A abreviação para a dependência funcional é DF ou d.f. O conjunto de atributos X é chamado de **lado esquerdo** da DF, e Y é chamado de **lado direito**.

Assim, a funcionalidade X determina Y em um esquema de relação R se, e somente se, sempre que duas tuplas de $r(R)$ combinarem sobre seu valor X , elas devem necessariamente combinar sobre seu valor Y . Observe o seguinte:

- Se uma restrição sobre R declarar que não pode haver mais de uma tupla com determinado valor X em qualquer instância de relação $r(R)$ — ou seja, X é uma **chave candidata** de R —, isso implica que $X \rightarrow Y$ para qualquer subconjunto de atributos Y de R (porque a restrição de chave implica que duas tuplas em qualquer estado válido $r(R)$ não terão o mesmo valor de X). Se X for uma chave candidata de R , então $X \rightarrow R$.
- Se $X \rightarrow Y$ em R , isso não quer dizer que $Y \rightarrow X$ em R .

Uma dependência funcional é uma propriedade da **semântica** ou **significado dos atributos**. Os projetistas de banco de dados usarão seu conhecimento da semântica dos atributos de R — ou seja, como eles se relacionam entre si — para especificar as dependências funcionais que devem ser mantidas em *todos* os estados de relação (extensões) r de R . Toda vez que a semântica de dois conjuntos de atributos em R indicar que uma dependência funcional deve ser mantida, especificamos a dependência como uma restrição. As extensões de relação $r(R)$ que satisfazem as restrições de dependência funcional são chamadas de **estados de relação válidos** (ou **extensões válidas**) de R . Logo, o uso principal das dependências funcionais é para descrever melhor um esquema de relação R ao especificar restrições sobre seus atributos que devem ser mantidas *o tempo todo*. Certas DFs podem ser especificadas sem que se refiram a uma relação específica, mas como uma propriedade desses atributos, dado seu significado comumente entendido. Por exemplo, $\{\text{Estado, Num_habilitacao}\} \rightarrow \text{Cpf}$ deve ser mantido para qualquer adulto no Brasil e, por isso, ser mantido sempre que esses atributos aparecerem em uma relação. Também é possível que certas dependências funcionais possam deixar de existir no mundo real se o relacionamento mudar. Por exemplo, a DF $\text{Cep} \rightarrow \text{Codigo_area}$ costumava existir como um relacionamento entre os códigos postais e os códigos de área de telefone no Brasil, mas, com a proliferação dos códigos de área telefônicos, isso não é mais verdadeiro.

Considere o esquema de relação FUNC_PROJ da Figura 15.3(b). Pela semântica dos atributos e da re-

⁷ Esse conceito de uma relação universal será importante quando discutirmos os algoritmos para o projeto de banco de dados relacional no Capítulo 16.

⁸ Esta suposição implica que cada atributo no banco de dados tenha um nome distinto. No Capítulo 3, iniciamos os nomes de atributo com nomes de relação, para obter exclusividade sempre que os atributos em relações distintas tinham o mesmo nome.

lação, sabemos que as seguintes dependências funcionais devem ser mantidas:

- $Cpf \rightarrow Fnome$
- $Projnumero \rightarrow \{Projnome, Projlocal\}$
- $\{Cpf, Projnumero\} \rightarrow Horas$

Essas dependências funcionais especificam que (a) o valor do número do Cadastro de Pessoa Física (Cpf) de um funcionário determina exclusivamente o nome do funcionário (Fnome), (b) o valor do número de um projeto (Projnumero) determina exclusivamente o nome do projeto (Projnome) e seu local (Projlocal) e (c) uma combinação de valores de Cpf e Projnumero determina exclusivamente o número de horas que o funcionário costuma trabalhar no projeto por semana (Horas). Como alternativa, dizemos que Fnome é determinado de maneira funcional por (ou dependente funcionalmente de) Cpf, ou *dado um valor de Cpf, sabemos o valor de Fnome*, e assim por diante.

Uma dependência funcional é uma *propriedade do esquema de relação R*, e não um estado de relação válido e específico *r* de *R*. Portanto, uma DF *não pode* ser deduzida automaticamente por determinada extensão de relação *r*, mas deve ser definida de maneira explícita por alguém que conhece a semântica dos atributos de *R*. Por exemplo, a Figura 15.7 mostra um estado em particular do esquema de relação ENSINA. Embora à primeira vista possamos pensar que $Texto \rightarrow Disciplina$, não podemos confirmar isso a menos que saibamos que é verdadeiro para *todos os estados legais possíveis* de ENSINA. É, no entanto, suficiente demonstrar *um único contraexemplo* para refutar uma dependência funcional. Por exemplo, como ‘Silva’ leciona tanto ‘Estruturas de Dados’ e ‘Gerenciamento de Dados’, podemos concluir que o Professor *não* determina funcionalmente a Disciplina.

Dada uma relação preenchida, não se podem determinar quais DFs são mantidas e quais não são, a

ENSINA

Professor	Disciplina	Texto
Silva	Estruturas de Dados	Bartram
Silva	Gerenciamento de Dados	Martin
Neto	Compiladores	Hoffman
Braga	Estruturas de Dados	Horowitz

Figura 15.7

Um estado de relação de ENSINA com uma possível dependência funcional $TEXTO \rightarrow DISCIPLINA$. Porém, $PROFESSOR \rightarrow DISCIPLINA$ está excluída.

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

Figura 15.8

Uma relação *R* (A, B, C, D) com sua extensão.

menos que o significado e os relacionamentos entre os atributos sejam conhecidos. Tudo o que se pode dizer é que certa DF *pode* existir se for mantida nessa extensão em particular. Não se pode garantir sua existência até que o significado dos atributos correspondentes seja claramente compreendido. Porém, pode-se afirmar de modo enfático que certa DF *não se mantém* se houver tuplas que mostrem a violação de tal DF. Veja a relação de exemplo ilustrativa na Figura 15.8. Nela, as DFs a seguir *podem ser mantidas* porque as quatro tuplas na extensão atual não têm violação dessas restrições: $B \rightarrow C$; $C \rightarrow B$; $\{A, B\} \rightarrow C$; $\{A, B\} \rightarrow D$; e $\{C, D\} \rightarrow B$. No entanto, as seguintes *não se mantêm* porque já temos violações delas na extensão dada: $A \rightarrow B$ (tuplas 1 e 2 violam essa restrição); $B \rightarrow A$ (tuplas 2 e 3 violam essa restrição); $D \rightarrow C$ (tuplas 3 e 4 a violam).

A Figura 15.3 apresenta uma **notação diagramática** para exibir DFs: cada DF aparece como uma linha horizontal. Os atributos do lado esquerdo da DF são conectados por linhas verticais à linha que representa a DF, enquanto os atributos do lado direito são conectados pelas linhas com setas que apontam para os atributos.

Indicamos com *F* o conjunto de dependências funcionais que são especificadas no esquema de relação *R*. Em geral, o projetista do esquema especifica as dependências funcionais que são *semanticamente óbvias*. Porém, diversas outras dependências funcionais se mantêm em *todas* as instâncias de relação válidas entre conjuntos de atributos que podem ser derivados das (e satisfazem as) dependências em *F*. Essas outras dependências podem ser *deduzidas* das DFs em *F*. Adiaremos os detalhes das regras de inferência e propriedades das dependências funcionais para o Capítulo 16.

15.3 Formas normais baseadas em chaves primárias

Após introduzir as dependências funcionais, agora estamos prontos para usá-las na especificação de alguns aspectos da semântica dos esquemas de relação. Consideramos que um conjunto de dependências funcionais é dado para cada relação, e que cada

relação tem uma chave primária designada. Essa informação combinada com os testes (condições) para formas normais controla o *processo de normalização* para o projeto do esquema relacional. A maioria dos projetos relacionais práticos assume uma das duas técnicas a seguir:

- Realiza um projeto de esquema conceitual usando um modelo conceitual como ER ou EER e mapeia o projeto conceitual para um conjunto de relações.
- Projeta as relações com base no conhecimento externo derivado de uma implementação existente de arquivos, formulários ou relatórios.

Ao seguir uma dessas técnicas, é útil avaliar a virtude de relações e decompô-las ainda mais, conforme a necessidade, para obter formas normais mais altas, usando a teoria de normalização apresentada neste capítulo e no seguinte. Nesta seção, focalizamos as três primeiras formas normais para esquemas de relação e a intuição por trás delas, e discutimos como elas foram desenvolvidas historicamente. Definições mais gerais dessas formas normais, que levam em conta todas as chaves candidatas de uma relação, em vez de apenas a chave primária, são adiadas para a Seção 15.4.

Começamos discutindo informalmente as formas normais e a motivação por trás de seu desenvolvimento, bem como revisando algumas definições do Capítulo 3, que são necessárias aqui. Depois, discutimos a primeira forma normal (1FN) na Seção 15.3.4, e apresentamos as definições da segunda forma normal (2FN) e terceira forma normal (3FN), que são baseadas em chaves primárias, nas seções 15.3.5 e 15.3.6, respectivamente.

15.3.1 Normalização de relações

O processo de normalização, proposto inicialmente por Codd (1972a), leva um esquema de relação por uma série de testes para *certificar* se ele satisfaz certa **forma normal**. O processo, que prossegue em um padrão de cima para baixo, avaliando cada relação em comparação com os critérios para as formas normais e decompondo as relações conforme a necessidade, pode assim ser considerado *projeto relacional por análise*. Inicialmente, Codd propôs três formas normais, que ele chamou de primeira, segunda e terceira forma normal. Uma definição mais forte da 3FN — chamada Forma Normal Boyce-Codd (FNBC) — foi proposta posteriormente por Boyce e Codd. Todas essas formas normais estão baseadas em uma única ferramenta analítica: as dependências funcionais entre os atributos de uma relação. Depois, uma quarta forma normal (4FN) e uma quinta forma

normal (5FN) foram propostas, com base nos conceitos de dependências multivaloradas e dependências de junção, respectivamente: estas serão discutidas rapidamente nas seções 15.6 e 15.7.

A **normalização de dados** pode ser considerada um processo de analisar os esquemas de relação dados com base em suas DFs e chaves primárias para conseguir as propriedades desejadas de (1) minimização da redundância e (2) minimização das anomalias de inserção, exclusão e atualização discutidas na Seção 15.1.2. Esse pode ser considerado um processo de 'filtragem' ou 'purificação' para fazer que o projeto tenha uma qualidade cada vez melhor. Esquemas de relação insatisfatórios, que não atendem a certas condições — os **testes de forma normal** —, são decompostos em esquemas de relação menores, que atendem aos testes e, portanto, possuem as propriedades desejáveis. Assim, o procedimento de normalização oferece aos projetistas de banco de dados o seguinte:

- Uma estrutura formal para analisar esquemas de relação com base em suas chaves e nas dependências funcionais entre seus atributos.
- Uma série de testes de forma normal que podem ser executados em esquemas de relação individuais, de modo que o banco de dados relacional possa ser **normalizado** para qualquer grau desejado.

Definição. A **forma normal** de uma relação refere-se à condição de forma normal mais alta a que ela atende e, portanto, indica o grau ao qual ela foi normalizada.

As formas normais, quando consideradas *isoladamente* de outros fatores, não garantem um bom projeto de banco de dados. Em geral, não é suficiente verificar em separado se cada esquema de relação no banco de dados está, digamos, na FNBC ou 3FN. Em vez disso, o processo de normalização pela decomposição também precisa confirmar a existência de propriedades adicionais que os esquemas relacionais, tomados juntos, devem possuir. Estas incluiriam duas propriedades:

A **propriedade de junção não aditiva** ou **junção sem perdas**, que garante que o problema de geração de tuplas falsas, discutido na Seção 15.1.4, não ocorra com relação aos esquemas de relação criados após a decomposição.

- A **propriedade de preservação de dependência**, que garante que cada dependência funcional seja representada em alguma relação individual resultante após a decomposição.

- A propriedade de junção não aditiva é extremamente crítica e **deve ser alcançada a todo custo**, ao passo que a propriedade de preservação de dependência, embora desejável, às vezes é sacrificada, conforme discutiremos na Seção 16.1.2. Adiaremos a apresentação dos conceitos e técnicas formais que garantem as duas propriedades citadas para o Capítulo 16.

15.3.2 Uso prático das formas normais

A maioria dos projetos práticos adquire projetos existentes de bancos de dados anteriores, projetos em modelos legados ou de arquivos existentes. A normalização é executada na prática, de modo que os projetos resultantes sejam de alta qualidade e atendam às propriedades desejáveis indicadas anteriormente. Embora várias formas normais mais altas tenham sido definidas, como a 4FN e a 5FN, que discutiremos nas seções 15.6 e 15.7, a utilidade prática dessas formas normais torna-se questionável quando as restrições sobre as quais elas estão baseadas são raras, e difíceis de entender ou detectar pelos projetistas e usuários de banco de dados que precisam descobrir essas restrições. Assim, o projeto de banco de dados praticado na indústria hoje presta atenção particular à normalização apenas até a 3FN, FNBC ou, no máximo, 4FN.

Outro ponto que merece ser observado é que os projetistas de banco de dados *não precisam* normalizar para a forma normal mais alta possível. As relações podem ser deixadas em um estado de normalização inferior, como 2FN, por questões de desempenho, como aquelas discutidas ao final da Seção 15.1.2. Fazer isso gera as penalidades correspondentes de lidar com as anomalias.

Definição. Desnormalização é o processo de armazenar a junção de relações na forma normal mais alta como uma relação da base, que está em uma forma normal mais baixa.

15.3.3 Definições de chaves e atributos participantes em chaves

Antes de prosseguirmos, vejamos novamente as definições de chaves de um esquema de relação, do Capítulo 3.

Definição. Uma **superchave** de um esquema de relação $R = \{A_1, A_2, \dots, A_n\}$ é um conjunto de atributos $S \subseteq R$ com a propriedade de que duas tuplas t_1 e t_2 em qualquer estado de relação válido r de R não terão $t_1[S] = t_2[S]$. Uma **chave** Ch é uma superchave com a propriedade adicional de que a remoção de qualquer atributo de Ch fará que Ch não seja mais uma superchave.

A diferença entre uma chave e uma superchave é que a primeira precisa ser *mínima*; ou seja, se tivermos uma chave $Ch = \{A_1, A_2, \dots, A_k\}$ de R , então $Ch - \{A_i\}$ não é uma chave de R para qualquer A_i , $1 \leq i \leq k$. Na Figura 15.1, $\{Cpf\}$ é uma chave para FUNCIONARIO, enquanto $\{Cpf\}$, $\{Cpf, Fnome\}$, $\{Cpf, Fnome, Datanasc\}$ e qualquer conjunto de atributos que inclua Cpf são todos superchaves.

Se um esquema de relação tiver mais de uma chave, cada uma é chamada de **chave candidata**. Uma das chaves candidatas é *arbitrariamente* designada para ser a **chave primária**, e as outras são chamadas de chaves secundárias. Em um banco de dados relacional prático, cada esquema de relação precisa ter uma chave primária. Se nenhuma chave candidata for conhecida para uma relação, a relação inteira pode ser tratada como uma superchave padrão. Na Figura 15.1, $\{Cpf\}$ é a única chave candidata para FUNCIONARIO, de modo que também é a chave primária.

Definição. Um atributo do esquema de relação R é chamado de **atributo principal** de R se ele for um membro de *alguma chave candidata* de R . Um atributo é chamado **não principal** se não for um atributo principal — ou seja, se não for um membro de qualquer chave candidata.

Na Figura 15.1, tanto Cpf quanto $Projnumero$ são atributos principais de TRABALHA_EM, ao passo que outros atributos de TRABALHA_EM são não principais.

Agora, vamos apresentar as três primeiras formas normais: 1FN, 2FN e 3FN. Elas foram propostas por Codd (1972a) como uma sequência para conseguir o estado desejável de relações 3FN ao prosseguir pelos estados intermediários de 1FN e 2FN, se necessário. Conforme veremos, 2FN e 3FN atacam diferentes problemas. Contudo, por motivos históricos, é comum segui-los nessa sequência. Logo, por definição, uma relação 3FN *já satisfaz* a 2FN.

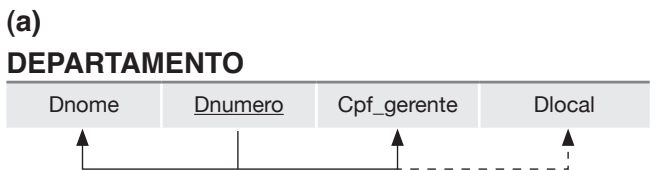
15.3.4 Primeira forma normal

A **primeira forma normal** (1FN) agora é considerada parte da definição formal de uma relação no modelo relacional básico (plano). Historicamente, ela foi definida para reprovar atributos multivalorados, atributos compostos e suas combinações. Ela afirma que o domínio de um atributo deve incluir apenas *valores atômicos* (simples, indivisíveis) e que o valor de qualquer atributo em uma tupla deve ser um *único valor* do domínio desse atributo. Logo, 1FN reprova ter um conjunto de valores, uma tupla de valores ou uma combinação de ambos como um valor de atributo para uma *única tupla*. Em outras palavras, a 1FN reprova *relações dentro de relações* ou *relações como*

valores de atributo dentro de tuplas. Os únicos valores de atributo permitidos pela 1FN são os **valores atômicos** (ou **indivisíveis**).

Considere o esquema de relação DEPARTAMENTO da Figura 15.1, cuja chave primária é Dnumero, e suponha que a estendamos ao incluir o atributo Dlocal, conforme mostra a Figura 15.9(a). Supomos que cada departamento pode ter *certo número de* locais. O esquema DEPARTAMENTO e um exemplo de estado de relação são mostrados na Figura 15.9. Como podemos ver, esta não está em 1FN porque Dlocal não é um atributo atômico, conforme ilustrado pela primeira tupla na Figura 15.9(b). Existem duas maneiras possíveis para examinar o atributo Dlocal:

- O domínio de Dlocal contém valores atômicos, mas algumas tuplas podem ter um conjunto desses valores. Nesse caso, Dlocal não é funcionalmente dependente da chave primária Dnumero.



(b)

DEPARTAMENTO

Dnome	<u>Dnumero</u>	Cpf_gerente	Dlocal
Pesquisa	5	33344555587	Santo André, Itu, São Paulo
Administração	4	98765432168	Mauá
Matriz	1	88866555576	São Paulo

(c)

DEPARTAMENTO

Dnome	<u>Dnumero</u>	Cpf_gerente	<u>Dlocal</u>
Pesquisa	5	33344555587	Santo André
Pesquisa	5	33344555587	Itu
Pesquisa	5	33344555587	São Paulo
Administração	4	98765432168	Mauá
Matriz	1	88866555576	São Paulo

Figura 15.9
Normalização na 1FN. (a) Um esquema de relação que não está em 1FN. (b) Exemplo de estado da relação DEPARTAMENTO. (c) Versão 1FN da mesma relação com redundância.

- O domínio de Dlocal contém conjuntos de valores e, portanto, é não atômico. Nesse caso, Dnumero → Dlocal, pois cada conjunto é considerado um único membro do domínio de atributo.⁹

De qualquer forma, a relação DEPARTAMENTO da Figura 15.9 não está na 1FN; de fato, ela nem sequer se qualifica como uma relação, de acordo com nossa definição na Seção 3.1. Existem três técnicas principais para conseguir a primeira forma normal para tal relação:

1. Remover o atributo Dlocal que viola a 1FN e colocá-lo em uma relação separada LOCALIZACAO_DEP, junto com a chave primária Dnumero de DEPARTAMENTO. A chave primária dessa relação é a combinação {Dnumero, Dlocal}, como mostra a Figura 15.2. Existe uma tupla distinta em LOCALIZACAO_DEP para *cada local* de um departamento. Isso decompõe a relação não 1FN em duas relações 1FN.
2. Expandir a chave de modo que haverá uma tupla separada na relação original DEPARTAMENTO para cada local de um DEPARTAMENTO, como mostra a Figura 15.9(c). Nesse caso, a chave primária torna-se a combinação {Dnumero, Dlocal}. Essa solução tem a desvantagem de introduzir a *redundância* na relação.
3. Se o *número máximo de valores* for conhecido para o atributo — por exemplo, se for conhecido que *no máximo três locais* poderão existir para um departamento —, substituir o atributo Dlocal pelos três atributos atômicos: Dlocal1, Dlocal2 e Dlocal3. Essa solução tem a desvantagem de introduzir *valores* NULL se a maioria dos departamentos tiver menos de três locais. Ela ainda introduz uma falsa semântica sobre a ordenação entre os valores de local, que não era intencionado originalmente. A consulta sobre esse atributo torna-se mais difícil. Por exemplo, considere como você escreveria a consulta: *Listar os departamentos que têm ‘Santo André’ como um de seus locais* nesse projeto.

Das três soluções anteriores, a primeira geralmente é considerada a melhor, pois não sofre de redundância e é completamente genérica, não tendo limite imposto sobre o número máximo de valores. De fato, se escolhermos a segunda solução, ela será

⁹ Nesse caso, podemos considerar o domínio de Dlocalizacoes como sendo o **conjunto de potência** do conjunto de locais isolados; ou seja, o domínio é composto por todos os subconjuntos possíveis do conjunto de locais isolados.

decomposta ainda mais durante as etapas de normalização subsequentes para a primeira solução.

A primeira forma normal também desaprova atributos multivalorados que por si só sejam compostos. Estes são chamados de **relações aninhadas**, pois cada tupla pode ter uma relação *dentro dela*. A Figura 15.10 mostra como a relação FUNC_PROJ poderia aparecer se o aninhamento for permitido. Cada tupla representa uma entidade de funcionário, e a relação PROJS(Projnumero, Horas) *dentro de cada*

(a)

FUNC_PROJ		Projs	
Cpf	Fnome	Projnumero	Horas

(b)

FUNC_PROJ			
Cpf	Fnome	Projnumero	Horas
12345678966	Silva, João B.	1	32,5
		2	7,5
66688444476	Lima, Ronaldo K.	3	40,0
45345345376	Leite, Joice A.	1	20,0
		2	20,0
33344555587	Wong, Fernando T.	2	10,0
		3	10,0
		10	10,0
		20	10,0
99988777767	Zelaya, Alice J.	30	30,0
		10	10,0
98798798733	Pereira, André V.	10	35,0
		30	5,0
98765432168	Souza, Jennifer S.	30	20,0
		20	15,0
88866555576	Brito, Jorge E.	20	NULL

(c)

FUNC_PROJ1

Cpf	Fnome
-----	-------

FUNC_PROJ2

Cpf	Projnumero	Horas
-----	------------	-------

Figura 15.10

Normalizando relações aninhadas para a 1FN. (a) Esquema da relação FUNC_PROJ com um atributo de relação aninhada PROJS. (b) Exemplo de extensão da relação FUNC_PROJ mostrando relações aninhadas dentro de cada tupla. (c) Decomposição de FUNC_PROJ nas relações FUNC_PROJ1 e FUNC_PROJ2 pela propagação da chave primária.

tupla representa os projetos do funcionário e o número de horas por semana que ele trabalha em cada projeto. O esquema dessa relação FUNC_PROJ pode ser representado da seguinte forma:

FUNC_PROJ(Cpf, Fnome, {PROJS(Projnumero, Horas)})

O conjunto de chaves { } identifica o atributo PROJS como multivalorado, e listamos os atributos componentes que formam o PROJS entre parênteses (). O interessante é que as tendências recentes para dar suporte a objetos complexos (ver Capítulo 11) e dados XML (ver Capítulo 12) tentam permitir e formalizar as relações aninhadas nos sistemas de bancos de dados relacionais, que eram reprovadas inicialmente pela 1FN.

Observe que Cpf é a chave primária da relação FUNC_PROJ nas figuras 15.10(a) e (b), enquanto Projnumero é a chave **parcial** da relação aninhada; ou seja, dentro de cada tupla, a relação aninhada precisa ter valores únicos de Projnumero. Para normalizar isso para a 1FN, removemos os atributos da relação aninhada para uma nova relação e *propagamos a chave primária* para ela. A chave primária da nova relação combinará a parcial com a chave primária da relação original. A decomposição e a propagação da chave primária resultam nos esquemas FUNC_PROJ1 e FUNC_PROJ2, como mostra a Figura 15.10(c).

Esse procedimento pode ser aplicado recursivamente a uma relação com aninhamento em nível múltiplo para **desaninhar** a relação para um conjunto de relações 1FN. Isso é útil na conversão de um esquema de relação não normalizado com muitos níveis de aninhamento em relações 1FN. A existência de mais de um atributo multivalorado em uma relação deve ser tratada com cuidado. Como um exemplo, considere a seguinte relação não 1FN:

PESSOA (Cpf, {Placa}, {Telefone})

Essa relação representa o fato de uma pessoa ter vários carros e vários telefones. Se a estratégia 2 acima for seguida, ela resulta em uma relação com todas as chaves:

PESSOA_NA_1FN (Cpf, Placa, Telefone)

Para evitar a introdução de qualquer relacionamento estranho entre Placa e Telefone, todas as combinações de valores possíveis são representadas para cada Cpf, fazendo surgir a redundância. Isso leva aos problemas tratados pelas dependências multivaloradas e 4FN, que discutiremos na Seção 15.6. O modo certo de lidar com os dois atributos multivalorados em PESSOA mostrados anteriormente é decompô-los em duas relações separadas, usando

a estratégia 1 já discutida: $P1(\underline{\text{Cpf}}, \text{Placa})$ e $P2(\underline{\text{Cpf}}, \text{Telefone})$.

15.3.5 Segunda forma normal

A **segunda forma normal** (2FN) é baseada no conceito de *dependência funcional total*. Uma dependência funcional $X \rightarrow Y$ é uma **dependência funcional total** se a remoção de qualquer atributo A de X significar que a dependência não se mantém mais; ou seja, para qualquer atributo $A \in X$, $(X - \{A\})$ não determina Y funcionalmente. Uma dependência funcional $X \rightarrow Y$ é uma **dependência parcial** se algum atributo $A \in X$ puder ser removido de X e a dependência ainda se mantiver; ou seja, para algum $A \in X$, $(X - \{A\}) \rightarrow Y$. Na Figura 15.3(b), $\{\text{Cpf}, \text{Projnumero}\} \rightarrow \text{Horas}$ é uma dependência total (nem $\text{Cpf} \rightarrow \text{Horas}$ nem $\text{Projnumero} \rightarrow \text{Horas}$ se mantêm). Contudo, a dependência $\{\text{Cpf}, \text{Projnumero}\} \rightarrow \text{Fnome}$ é parcial porque $\text{Cpf} \rightarrow \text{Fnome}$ se mantém.

Definição. Um esquema de relação R está em 2FN se cada atributo não principal A em R for *total e funcionalmente dependente* da chave primária de R .

O teste para a 2FN envolve testar as dependências funcionais cujos atributos do lado esquerdo fazem parte da chave primária. Se a chave primária tiver um único atributo, o teste não precisa ser aplicado. A relação FUNC_PROJ na Figura 15.3(b) está na 1FN, mas não está na 2FN. O atributo não principal Fnome viola a 2FN por causa da DF2, assim como os atributos não principais Projnome e Projlocal, por causa da DF3. As dependências funcionais DF2 e DF3 tornam Fnome, Projnome e Projlocal parcialmente dependentes da chave primária $\{\text{Cpf}, \text{Projnumero}\}$ de FUNC_PROJ, violando, assim, o teste da 2FN.

Se um esquema de relação não estiver na 2FN, ele pode ser *segundo normalizado* ou *normalizado pela 2FN* para uma série de relações 2FN em que os atributos não principais são associados com a parte da chave primária em que eles são total e funcionalmente dependentes. Portanto, as dependências funcionais DF1, DF2 e DF3 da Figura 15.3(b) levam à decomposição de FUNC_PROJ nos três esquemas de relação FP1, FP2 e FP3 mostrados na Figura 15.11(a), cada qual estando na 2FN.

15.3.6 Terceira forma normal

A **terceira forma normal** (3FN) é baseada no conceito de *dependência transitiva*. Uma dependên-

cia funcional $X \rightarrow Y$ em um esquema de relação R é uma **dependência transitiva** se houver um conjunto de atributos Z em R que nem sejam uma chave candidata nem um subconjunto de qualquer chave de R ,¹⁰ e tanto $X \rightarrow Z$ quanto $Z \rightarrow Y$ se mantiverem. A dependência $\text{Cpf} \rightarrow \text{Cpf_gerente}$ é transitiva por meio de Dnumero em FUNC_DEP na Figura 15.3(a), pois ambas as dependências $\text{Cpf} \rightarrow \text{Dnumero}$ e $\text{Dnumero} \rightarrow \text{Cpf_gerente}$ se mantêm e Dnumero não é nem uma chave por si só nem um subconjunto da chave de FUNC_DEP. Intuitivamente, podemos ver que a dependência de Cpf_gerente sobre Dnumero é indesejável em FUNC_DEP, pois Dnumero não é uma chave de FUNC_DEP.

Definição. De acordo com a definição original de Codd, um esquema de relação R está na 3FN se ele satisfizer a 2FN e nenhum atributo não principal de R for transitivamente dependente da chave primária.

O esquema de relação FUNC_DEP da Figura 15.3(a) está na 2FN, pois não existe dependência parcial sobre uma chave. Porém, FUNC_DEP não está na 3FN devido à dependência transitiva de Cpf_gerente (e também Dnome) em Cpf por meio de Dnumero. Podemos normalizar FUNC_DEP decompondo-o nos dois esquemas de relação 3FN DF1 e DF2 mostrados na Figura 15.11(b). Intuitivamente, vemos que DF1 e DF2 representam fatos de entidades independentes sobre funcionários e departamentos. Uma operação JUNTÃO NATURAL sobre DF1 e DF2 recuperará a relação original FUNC_DEP sem gerar tuplas falsas.

De maneira intuitiva, podemos ver que qualquer dependência funcional de que o lado esquerdo faz parte (é um subconjunto apropriado) da chave primária, ou qualquer dependência funcional de que o lado esquerdo é um atributo não chave, é uma DF *problemática*. A normalização 2FN e 3FN remove essas DFs problemáticas ao decompor a relação original em novas relações. Em relação ao processo de normalização, não é necessário remover as dependências parciais antes das dependências transitivas, porém, historicamente, a 3FN tem sido definida com a suposição de que uma relação é testada primeiro pela 2FN, antes de ser testada pela 3FN. A Tabela 15.1 resume informalmente as três formas normais com base nas chaves primárias, os testes usados em cada uma e a *solução* ou normalização realizada para alcançar a forma normal.

¹⁰ Essa é a definição geral de dependência transitiva. Como estamos preocupados apenas com as chaves primárias nesta seção, permitimos dependências transitivas onde X é a chave primária, mas Z pode ser (um subconjunto de) uma chave candidata.

(a)

FUNC_PROJ

	Cpf	Projnumero	Horas	Fnome	Projnome	Projlocal
DF1			↑	↑	↑	↑
DF2				↑	↑	↑
DF3					↑	↑

Normalizacao 2FN

FP1

	Cpf	Projnumero	Horas
DF1			↑

FP2

	Cpf	Fnome
DF2		↑

FP3

	Projnumero	Projnome	Projlocal
DF3		↑	↑

(b)

FUNC_DEP

	Fnome	Cpf	Datanasc	Endereco	Dnumero	Dnome	Cpf_gerente
			↑	↑	↑	↑	↑

Normalizacao 3FN

DF1

	Fnome	Cpf	Datanasc	Endereco	Dnumero
			↑	↑	↑

DF2

	Dnumero	Dnome	Cpf_gerente
		↑	↑

Figura 15.11

Normalizando para 2FN e 3FN. (a) Normalizando FUNC_PROJ em relações 2FN. (b) Normalizando FUNC_DEP em relações 3FN.

Tabela 15.1

Resumo das formas normais baseadas em chaves primárias e a normalização correspondente.

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

15.4 Definições gerais da segunda e terceira formas normais

Em geral, queremos projetar nossos esquemas de relação de modo que não tenham dependências

parciais nem transitivas, pois esses tipos de dependências causam as anomalias de atualização discutidas na Seção 15.1.2. As etapas para normalização para relações 3FN que discutimos até aqui desaprovam dependências parciais e transitivas na *chave*

primária. O procedimento de normalização descrito é útil para análise em situações práticas para determinado banco de dados, no qual as chaves primárias já foram definidas. Essas definições, entretanto, não levam em conta outras chaves candidatas de uma relação, se houver. Nesta seção, mostramos as definições mais gerais da 2FN e da 3FN que levam em conta *todas* as chaves candidatas de uma relação. Observe que isso não afeta a definição da 1FN, pois ela independe das chaves e dependências funcionais. Como uma definição geral de **atributo principal**, um atributo que faz parte de *qualquer chave candidata* será considerado principal. Dependências funcionais parciais e totais e dependências transitivas agora serão consideradas *com relação a todas as chaves candidatas* de uma relação.

15.4.1 Definição geral da segunda forma normal

Definição. Um esquema de relação R está na **segunda forma normal (2FN)** se cada atributo não principal A em R não for parcialmente dependente de *qualquer* chave de R .¹¹

O teste para 2FN envolve avaliar as dependências funcionais cujos atributos do lado esquerdo fazem *parte da* chave primária. Se a chave primária contiver um único atributo, o teste não precisa ser aplicado. Considere o esquema de relação LOTES mostrado na Figura 15.12(a), que descreve lotes de terreno à venda em diversas cidades de um estado. Suponha que existam duas chaves candidatas: Propriedade_num e {Nome_cidade, Num_lote}; ou seja, números de lote são únicos apenas dentro de cada cidade, mas números de Id_propriedade são únicos entre as cidades do estado inteiro.

Com base nas duas chaves candidatas Propriedade_num e {Nome_cidade, Num_lote}, as dependências funcionais DF1 e DF2 da Figura 15.12(a) se mantêm. Escolhemos Propriedade_num como a chave primária, por isso ela está sublinhada na Figura 15.12(a), mas nenhuma consideração especial será feita a essa chave sobre a outra chave candidata. Suponha que as duas outras dependências funcionais se mantenham em LOTES:

DF3: Nome_cidade \rightarrow Imposto

DF4: Area \rightarrow Preco

Em palavras, a dependência DF3 diz que Imposto é fixo para determinada cidade (não varia de lote para lote na mesma cidade), enquanto DF4 diz que o

preço de um lote é determinado por sua área, independentemente da cidade em que esteja. (Suponha que esse seja o preço do lote para fins de imposto.)

O esquema de relação LOTES viola a definição geral da 2FN porque Imposto é parcialmente dependente da chave candidata {Nome_cidade, Num_lote}, por causa da DF3. Para normalizar LOTES na 2FN, decompõe-o nas duas relações LOTES1 e LOTES2, mostradas na Figura 15.12(b). Construímos LOTES1 ao remover o atributo Imposto que viola a 2FN de LOTES e colocando-o com Nome_cidade (o lado esquerdo da DF3 que causa a dependência parcial) em outra relação LOTES2. Tanto LOTES1 quanto LOTES2 estão na 2FN. Observe que a DF4 não viola a 2FN e é transportada para LOTES1.

15.4.2 Definição geral da terceira forma normal

Definição. Um esquema de relação R está na **terceira forma normal** se toda vez que uma dependência funcional *não trivial* $X \rightarrow A$ se mantiver em R , ou (a) X for uma superchave de R ou (b) A for um atributo principal de R .

De acordo com essa definição, LOTES2 (Figura 15.12(b)) está na 3FN. No entanto, DF4 em LOTES1 viola a 3FN, pois Area não é uma superchave e Preco não é um atributo principal em LOTES1. Para normalizar LOTES1 para a 3FN, nós a decomparamos nos esquemas de relação LOTES1A e LOTES1B mostrados na Figura 15.12(c). Construímos LOTES1A removendo o atributo Preco que viola a 3FN de LOTES1 e colocando-o com Area (o lado esquerdo de DF4 que causa a dependência transitiva) em outra relação LOTES1B. Tanto LOTES1A quanto LOTES1B estão na 3FN.

Dois pontos precisam ser observados sobre esse exemplo e a definição geral da 3FN:

- LOTES1 viola a 3FN porque Preco é transitivamente dependente em cada uma das chaves candidatas de LOTES1 por meio do atributo não principal Area.
- Essa definição geral pode ser aplicada *diretamente* para testar se um esquema de relação está na 3FN (este *não* precisa passar pela 2FN primeiro). Se aplicarmos a definição da 3FN dada a LOTES com as dependências de DF1 a DF4, descobriremos que *ambas* violam a 3FN. Portanto, poderíamos decompor LOTES em LOTES1A, LOTES1B e LOTES2 diretamente. Logo, as dependências transitiva e

¹¹ Essa definição pode ser reformulada da seguinte forma: um esquema de relação R está na 2FN se cada atributo não principal A em R for total e funcionalmente dependente de *cada* chave de R .

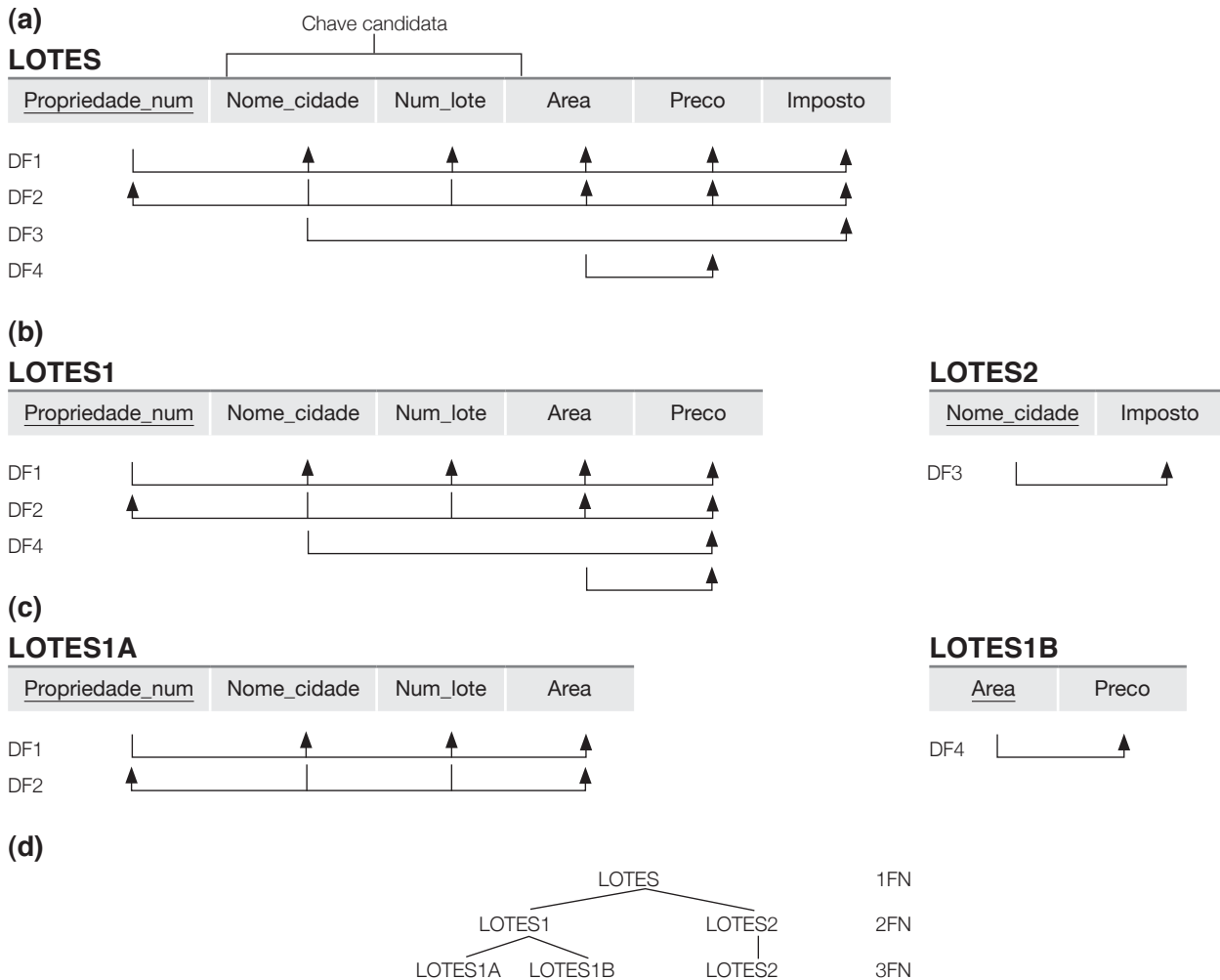


Figura 15.12

Normalização para 2FN e 3FN. (a) A relação LOTES com suas dependências funcionais de DF1 a DF4. (b) Decompondo para as relações 2FN LOTES1 e LOTES2. (c) Decompondo LOTES1 para as relações 3FN LOTES1A e LOTES1B. (d) Resumo da normalização progressiva de LOTES.

parcial que violam a 3FN podem ser removidas *em qualquer ordem*.

15.4.3 Interpretando a definição geral da terceira forma normal

Um esquema de relação R viola a definição geral da 3FN se uma dependência funcional $X \rightarrow A$, que se mantém em R , não atender a qualquer condição — significando que ela viola *ambas* as condições (a) e (b) da 3FN. Isso pode ocorrer devido a dois tipos de dependências funcionais problemáticas:

- Um atributo não principal determina outro atributo não principal. Aqui, em geral, temos uma dependência transitiva que viola a 3FN.
- Um subconjunto apropriado de uma chave de R determina funcionalmente um atributo

não principal. Aqui, temos uma dependência parcial que viola a 3FN (e também a 2FN).

Portanto, podemos indicar uma **definição alternativa geral da 3FN** da seguinte forma:

Definição alternativa. Um esquema de relação R está na 3FN se cada atributo não principal de R atender às duas condições a seguir:

- Ele é total e funcionalmente dependente de cada chave de R .
- Ele é dependente não transitivamente de cada chave de R .

15.5 Forma Normal de Boyce-Codd

A Forma Normal Boyce-Codd (FNBC) foi proposta como uma forma mais simples da 3FN, mas descobriu-se que ela era mais rigorosa. Ou seja, cada relação

em FNBC também está na 3FN. Porém, uma relação na 3FN *não necessariamente* está na FNBC. Intuitivamente, podemos ver a necessidade de uma forma normal mais forte que a 3FN ao voltar ao esquema de relação LOTES da Figura 15.12(a) com suas quatro dependências funcionais, de DF1 a DF4. Suponha que tenhamos milhares de lotes na relação, mas que eles sejam de apenas duas cidades: Ribeirão Preto e Analândia. Suponha também que os tamanhos de lote em Ribeirão Preto sejam de apenas 0,5, 0,6, 0,7, 0,8, 0,9 e 1,0 hectare, enquanto os tamanhos de lote em Analândia sejam restritos a 1,1, 1,2, ..., 1,9 e 2,0 hectares. Em tal situação, teríamos a dependência funcional adicional DF5: $\text{Area} \rightarrow \text{Nome_cidade}$. Se acrescentamos isso às outras dependências, o esquema de relação LOTES1A ainda estará na 3FN, pois Nome_cidade é um atributo principal.

A área de um lote que determina a cidade, conforme especificada pela DF5, pode ser representada por 16 tuplas em uma relação separada $R(\text{Area}, \text{Nome_cidade})$, pois existem apenas 16 valores de Area possíveis (ver Figura 15.13). Essa representação diminui a redundância de repetir a mesma informação em milhares de tuplas LOTES1A. A FNBC é uma *forma normal mais forte*, que reprovaria LOTES1A e sugeriria a necessidade de sua decomposição.

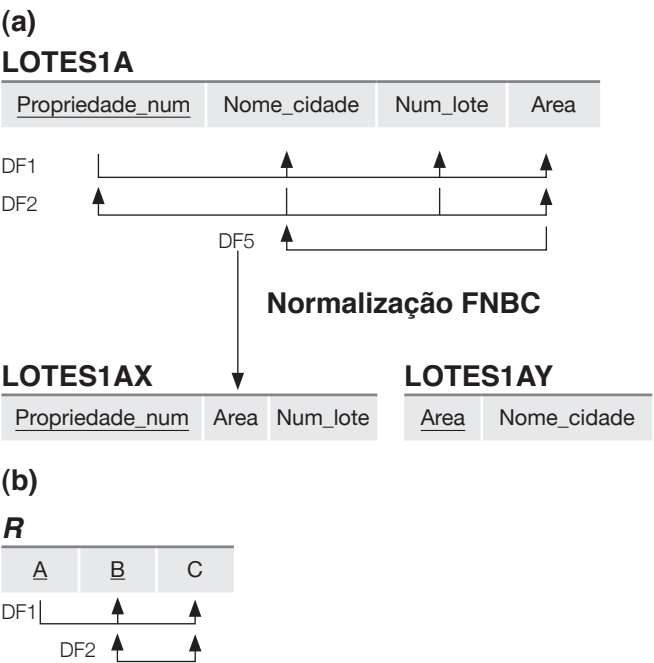


Figura 15.13
Forma normal de Boyce-Codd. (a) Normalização FNBC de LOTES1A com a dependência funcional DF2 sendo perdida na decomposição. (b) Uma relação esquemática com DFs; ela está na 3FN, mas não na FNBC.

Definição. Um esquema de relação R está na FNBC se toda vez que uma dependência funcional *não trivial* $X \rightarrow A$ se mantiver em R , então X é uma superchave de R .

A definição formal da FNBC difere da definição da 3FN porque a condição (b) da 3FN, que permite que A seja principal, está ausente da FNBC. Isso torna a FNBC uma forma normal mais forte em comparação com a 3FN. Em nosso exemplo, a DF5 viola a FNBC em LOTES1A porque AREA não é uma superchave de LOTES1A. Observe que DF5 satisfaz a 3FN em LOTES1A porque Nome_cidade é um atributo principal (condição b), mas essa condição não existe na definição da FNBC. Podemos decompor LOTES1A em duas relações FNBC, LOTES1AX e LOTES1AY, mostradas na Figura 15.13(a). Essa decomposição perde a dependência funcional DF2 porque seus atributos não coexistem mais na mesma relação após a decomposição.

Na prática, a maioria dos esquemas de relação que estão na 3FN também estão na FNBC. Somente se $X \rightarrow A$ se mantiver em um esquema de relação R com X não sendo uma superchave e A sendo um atributo principal é que R estará na 3FN, mas não na FNBC. O esquema de relação R mostrado na Figura 15.13(b) ilustra o caso geral de tal relação. O ideal é que o projeto de banco de dados relacional lute para alcançar FNBC ou 3FN para cada esquema de relação. Obter o *status* de normalização apenas de 1FN ou 2FN não é considerado adequado, visto que eles foram desenvolvidos historicamente como trampolins para a 3FN e a FNBC.

Como outro exemplo, considere a Figura 15.14, que mostra uma relação ENSINA com as seguintes dependências:

DF1: $\{\text{Aluno}, \text{Disciplina}\} \rightarrow \text{Professor}$
DF2:¹² $\text{Professor} \rightarrow \text{Disciplina}$

ENSINA

Aluno	Disciplina	Professor
Lima	Banco de dados	Marcos
Silva	Banco de dados	Navathe
Silva	Sistemas operacionais	Omar
Silva	Teoria	Charles
Souza	Banco de dados	Marcos
Souza	Sistemas operacionais	Antonio
Wong	Banco de dados	Gomes
Zelaya	Banco de dados	Navathe
Lima	Sistemas operacionais	Omar

Figura 15.14
Uma relação ENSINA que está na 3FN, mas não na FNBC.

¹² Essa dependência significa que *cada professor ensina uma disciplina* é uma restrição para essa aplicação.

Observe que $\{\text{Aluno}, \text{Disciplina}\}$ é uma chave candidata para essa relação e que as dependências mostradas seguem o padrão da Figura 15.13(b), com Aluno como A , Disciplina como B e Professor como C . Logo, essa relação está na 3FN, mas não na FNBC. A decomposição desse esquema de relação em dois esquemas não é direta, pois ele pode ser decomposto em um dos três pares possíveis a seguir:

1. $\{\text{Aluno}, \text{Professor}\}$ e $\{\text{Aluno}, \text{Disciplina}\}$.
2. $\{\text{Disciplina}, \text{Professor}\}$ e $\{\text{Disciplina}, \text{Aluno}\}$.
3. $\{\text{Professor}, \text{Disciplina}\}$ e $\{\text{Professor}, \text{Aluno}\}$.

Todas as três decomposições *perdem* a dependência funcional DF1. A *decomposição desejável* dessas que são mostradas é a 3 porque isso não gerará tuplas falsas após uma junção.

Um teste para determinar se uma decomposição é não aditiva (ou sem perdas) será discutido na Seção 16.2.4, sob a Propriedade NJB. Em geral, uma relação não na FNBC deve ser decomposta de modo a atender a esta propriedade.

Garantimos que atendemos a essa propriedade, pois a decomposição não aditiva é essencial durante a normalização. Possivelmente, podemos ter de abrir mão da preservação de todas as dependências funcionais nas relações decompostas, como acontece neste exemplo. O algoritmo 16.5 faz isso e poderia ser usado para dar a decomposição 3 para ENSINA, que produz duas relações em FNBC como:

$(\text{Professor}, \text{Disciplina})$ e $(\text{Professor}, \text{Aluno})$

Observe que, se designarmos $(\text{Aluno}, \text{Professor})$ como chave primária da relação ENSINA, a DF Professor \rightarrow Disciplina causa uma dependência parcial (não totalmente funcional) de Disciplina sobre uma parte dessa chave. Essa DF pode ser removida como uma parte da segunda normalização, produzindo exatamente as mesmas duas relações no resultado. Esse é um exemplo de caso em que podemos atingir o mesmo projeto FNBC definitivo por meio de caminhos de normalização alternativos.

15.6 Dependência multivalorada e quarta forma normal

Até aqui, discutimos o conceito de dependência funcional, que de longe é o tipo mais importante de dependência na teoria de projeto de banco de dados relacional, e formas normais baseadas nas dependências funcionais. Entretanto, em muitos casos, as

relações possuem restrições que não podem ser especificadas como dependências funcionais. Nesta seção, discutimos o conceito de dependência multivalorada (MVD — **Multivalued Dependency**) e definimos a *quarta forma normal*, que se baseia nessa dependência. Uma discussão mais formal das MVDs e suas propriedades ficará para o Capítulo 16. As dependências multivaloradas são uma consequência da primeira forma normal (1FN) (ver Seção 15.3.4), que desaprova um atributo em uma tupla para ter um *conjunto de valores*, e o processo correspondente de conversão de uma relação não normalizada para 1FN. Se tivermos dois ou mais atributos *independentes* multivalorados no mesmo esquema de relação, obtemos o problema de ter que repetir cada valor de um dos atributos com cada valor do outro atributo, a fim de manter o estado da relação coerente e as independências entre os atributos, envolvidos. Essa restrição é especificada por uma dependência multivalorada.

Por exemplo, considere a relação FUNC mostrada na Figura 15.15(a). Uma tupla nessa relação FUNC representa o fato de que um funcionário cujo nome é Fnome trabalha no projeto cujo nome é Projnome e tem um dependente cujo nome é Nome_dependente. Um funcionário pode atuar em vários projetos e ter vários dependentes, e seus projetos e dependentes são independentes um do outro.¹³ Para manter o estado da relação coerente, e para evitar quaisquer relacionamentos falsos entre os dois atributos independentes, devemos ter uma tupla separada para representar cada combinação de um dependente e de um projeto de um funcionário. Essa restrição é especificada como uma dependência multivalorada na relação FUNC, que definimos nesta seção. Informalmente, sempre que dois relacionamentos 1:N *independentes* $A:B$ e $A:C$ são misturados na mesma relação, $R(A, B, C)$, uma MVD pode surgir.¹⁴

15.6.1 Definição formal de dependência multivalorada

Definição. Uma dependência multivalorada $X \twoheadrightarrow Y$ especificada sobre o esquema de relação R , onde X e Y são subconjuntos de R , determina a seguinte restrição sobre qualquer estado de relação r de R : Se duas tuplas t_1 e t_2 existirem em r tais que $t_1[X] = t_2[X]$, então duas tuplas t_3 e t_4 também deverão existir em r com as seguintes propriedades,¹⁵ nas quais usamos Z para indicar $(R - (X \cup Y))$.¹⁶

¹³ Em um diagrama ER, cada um seria representado como um atributo multivalorado ou como um tipo de entidade fraca (ver Capítulo 7).

¹⁴ Essa MVD é indicada como $A \twoheadrightarrow B|C$.

¹⁵ As tuplas t_1 , t_2 , t_3 e t_4 não são necessariamente distintas.

¹⁶ Z é uma forma abreviada para os atributos em R após os atributos em $(X \cup Y)$ serem removidos de R .

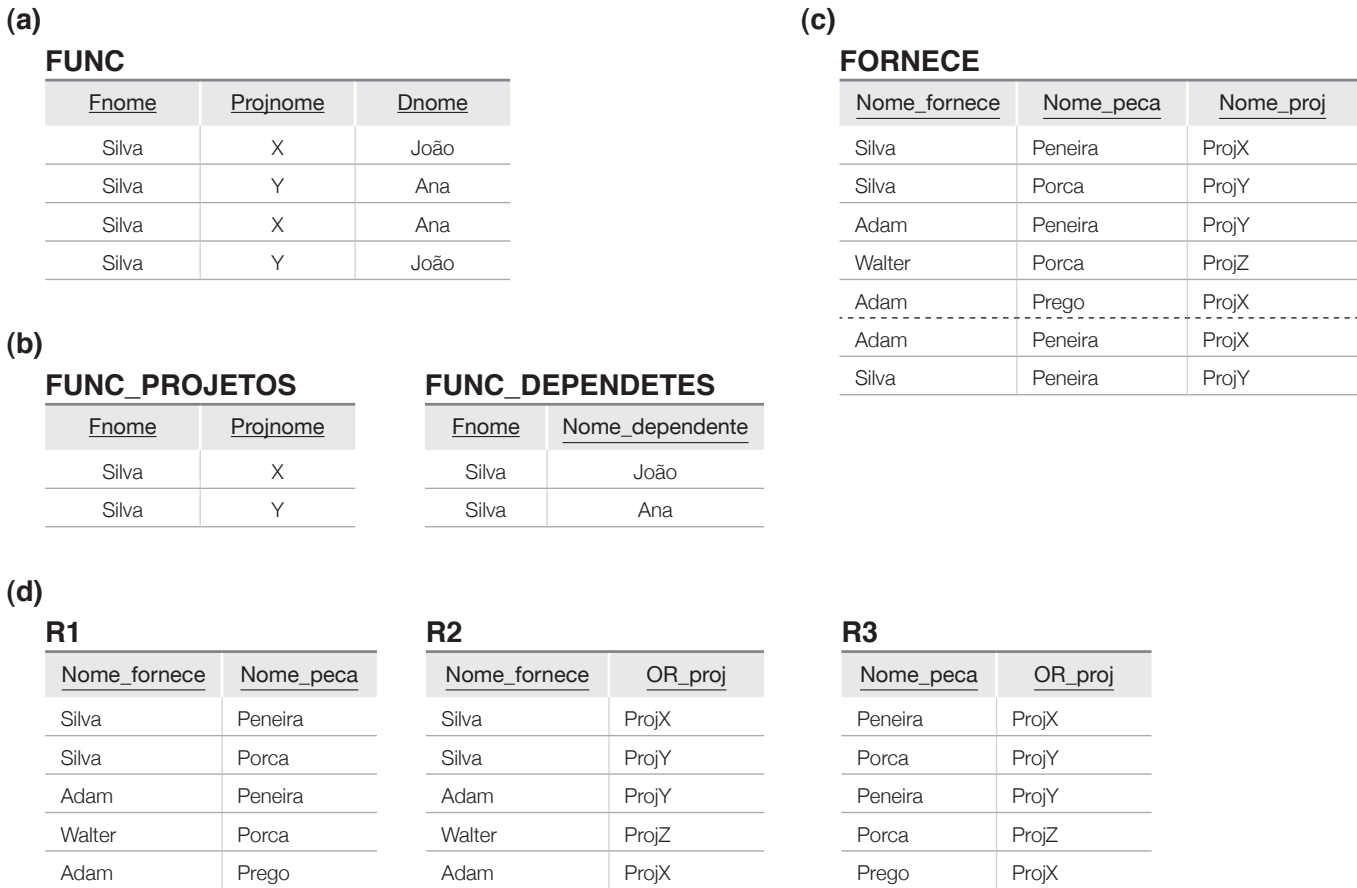


Figura 15.15

Quarta e quinta formas normais. (a) A relação FUNC com duas MVDs: $Fnome \twoheadrightarrow Projnome$ e $Fnome \twoheadrightarrow Dnome$. (b) Decompondo a relação FUNC em duas relações 4FN FUNC_PROJETOS e FUNC_DEPENDENTES. (c) A relação FORNECE sem MVDs está na 4FN, mas não na 5FN se tiver a DJ(R_1, R_2, R_3). (d) Decompondo a relação FORNECE nas relações 5FN R_1, R_2, R_3 .

■ $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.

■ $t_3[Y] = t_1[Y]$ e $t_4[Y] = t_2[Y]$.

■ $t_3[Z] = t_2[Z]$ e $t_4[Z] = t_1[Z]$.

Sempre que $X \twoheadrightarrow Y$ se mantiver, diremos que X **multidetermina** Y . Devido à simetria na definição, toda vez que $X \twoheadrightarrow Y$ for mantido em R , o mesmo acontecerá com $X \twoheadrightarrow Z$. Logo, $X \twoheadrightarrow Y$ implica $X \twoheadrightarrow Z$, e, portanto, às vezes é escrito como $X \twoheadrightarrow Y|Z$.

Uma MVD $X \twoheadrightarrow Y$ em R é chamada de **MVD trivial** se (a) Y for um subconjunto de X , ou (b) $X \cup Y = R$. Por exemplo, a relação FUNC_PROJETOS da Figura 15.15(b) tem a MVD trivial $Fnome \twoheadrightarrow Projnome$. Uma MVD que não satisfaz nem (a) nem (b) é chamada de **MVD não trivial**. Uma MVD trivial será mantida em *qualquer* estado de relação r de R . Ela é chamada dessa maneira porque não especifica qualquer restrição significativa sobre R .

Se tivermos uma *MVD não trivial* em uma relação, talvez precisemos repetir valores redundantemente nas tuplas. Na relação FUNC da Figura 15.15(a),

os valores ‘X’ e ‘Y’ de Projnome são repetidos com cada valor de Dnome (ou, por simetria, os valores ‘João’ e ‘Ana’ de Nome_dependente são repetidos com cada valor de Projnome). Essa redundância é claramente indesejável. Contudo, o esquema FUNC está na FNBC porque *nenhuma* dependência funcional se mantém em FUNC. Portanto, precisamos definir uma quarta forma normal que é mais forte que a FNBC e desaprova esquemas de relação como FUNC. Observe que as relações com MVDs não triviais tendem a ser **relações de todas as chaves** — ou seja, sua chave são todos os seus atributos juntos. Além do mais, é raro que essas relações de todas as chaves com uma ocorrência combinatória de valores repetidos sejam projetadas na prática. Porém, o reconhecimento das MVDs como uma dependência problemática em potencial é essencial no projeto relacional.

Agora, apresentamos a definição da **quarta forma normal (4FN)**, que é violada quando uma relação tem dependências multivaloradas indesejáveis e, portanto, pode ser usada para identificar e decompor essas relações.

Definição. Um esquema de relação R está na 4FN com relação a um conjunto de dependências F (que inclui dependências funcionais e dependências multivaloradas) se, para cada dependência multivalorada *não trivial* $X \twoheadrightarrow Y$ em F^+ ,¹⁷ X é uma superchave para R .

Podemos declarar os seguintes pontos:

- Uma relação de todas as chaves está sempre na FNBC, pois não tem DFs.
- Uma relação de todas as chaves, como a relação FUNC da Figura 15.15(a), que não tem DFs mas tem a MVD $F_{\text{nome}} \twoheadrightarrow \text{Projnome} \mid D_{\text{nome}}$, não está na 4FN.
- Uma relação que não está na 4FN devido a uma MVD não trivial precisa ser decomposta para convertê-la em um conjunto de relações na 4FN.
- A decomposição remove a redundância causada pela MVD.

O processo de normalização de uma relação envolvendo MVDs não triviais, que não está na 4FN, consiste em decompô-la de modo que cada MVD seja representada por uma relação separada, onde se torna uma MVD trivial. Considere a relação FUNC da Figura 15.15(a). FUNC não está na 4FN porque, nas MVDs não triviais, $F_{\text{nome}} \twoheadrightarrow \text{Projnome}$ e $F_{\text{nome}} \twoheadrightarrow \text{Nome_dependente}$, e F_{nome} não é uma superchave de FUNC. Decompomos FUNC em FUNC_PROJETOS e FUNC_DEPENDENTES, mostrados na Figura 15.15(b). Tanto FUNC_PROJETOS quanto FUNC_DEPENDENTES estão na 4FN, porque as MVDs $F_{\text{nome}} \twoheadrightarrow \text{Projnome}$ em FUNC_PROJETOS e $F_{\text{nome}} \twoheadrightarrow \text{Nome_dependentes}$ em FUNC_DEPENDENTES são MVDs triviais. Nenhuma outra MVD não trivial é mantida em FUNC_PROJETOS ou FUNC_DEPENDENTES. Também, nenhuma DF é mantida nesses esquemas de relação.

15.7 Dependências de junção e quinta forma normal

Em nossa discussão até aqui, indicamos as dependências funcionais problemáticas e mostramos como elas foram eliminadas por um processo de decomposição binária repetido para removê-las durante o processo de normalização, para obter 1FN, 2FN, 3FN e FNBC. Essas decomposições binárias precisam obedecer à propriedade NJB da Seção 16.2.4, que referenciamos ao discutir a decomposição para alcançar a FNBC. Obter a 4FN normalmente tam-

bém envolve eliminar as MVDs por decomposições binárias repetidas. Entretanto, em alguns casos, pode não haver decomposição de junção não aditiva de R em *dois* esquemas de relação, mas pode haver uma decomposição de junção não aditiva em *mais de dois* esquemas de relação. Além disso, pode não haver dependência funcional em R que viole qualquer forma normal até a FNBC, e também pode não haver MVD não trivial presente em R que viole a 4FN. Então, lançamos mão de outra dependência, chamada *dependência de junção* e, se estiver presente, executamos uma *decomposição multivias* para a quinta forma normal (5FN). É importante observar que tal dependência é uma restrição semântica bastante peculiar, que é muito difícil de detectar na prática. Portanto, a normalização para a 5FN raramente é feita nestes termos.

Definição. Uma *dependência de junção* (DJ), indicada por $DJ(R_1, R_2, \dots, R_n)$, especificada no esquema de relação R , determina uma restrição sobre os estados r de R . A restrição indica que cada estado válido r de R deve ter uma decomposição de junção não aditiva para R_1, R_2, \dots, R_n . Logo, para cada r desse tipo, temos

$$*(\pi_{R_1}(r), \pi_{R_2}(r), \dots, \pi_{R_n}(r)) = r$$

Observe que uma MVD é um caso especial de DJ em que $n = 2$. Ou seja, uma DJ indicada como $DJ(R_1, R_2)$ implica uma MVD $(R_1 \cap R_2) \twoheadrightarrow (R_1 - R_2)$ (ou, por simetria, $(R_1 \cap R_2) \twoheadrightarrow (R_2 - R_1)$). Uma dependência de junção $DJ(R_1, R_2, \dots, R_n)$, especificada sobre o esquema de relação R , é uma DJ *trivial* se um dos esquemas de relação R_i em $DJ(R_1, R_2, \dots, R_n)$ é igual a R . Tal dependência é chamada de trivial porque tem a propriedade de junção não aditiva para qualquer estado de relação r de R e, portanto, não especifica qualquer restrição sobre R . Agora, podemos definir a quinta forma normal, que também é chamada *forma normal projeção-junção*.

Definição. Um esquema de relação R está na quinta forma normal (5FN) (ou *forma normal projeção-junção* — FNPJ) com relação a um conjunto F de dependências funcionais, multivaloradas e de junção se, para cada dependência de junção não trivial $DJ(R_1, R_2, \dots, R_n)$ em F^+ (ou seja, implicada por F),¹⁸ cada R_i é uma superchave de R .

Para ver um exemplo de DJ, considere mais uma vez a relação de todas as chaves FORNECE da Figura 15.15(c). Suponha que a seguinte restrição adicional sempre seja mantida: toda vez que um fornecedor f

¹⁷ F^+ refere-se à cobertura das dependências funcionais F , ou todas as dependências que são implicadas por F . Isso será definido na Seção 16.1.

¹⁸ Novamente, F^+ refere-se à cobertura de dependências funcionais F , ou todas as dependências que são implicadas por F . Isso será definido na Seção 16.1.

fornece a peça p , e um projeto j usa a peça p , e o fornecedor f fornece *pelo menos uma* peça para o projeto j , então o fornecedor f também estará fornecendo a peça p ao projeto j . Essa restrição pode ser declarada de outras maneiras e especifica uma dependência de junção $DJ(R_1, R_2, R_3)$ entre as três projeções $R_1(\text{Nome_fornecedor}, \text{Nome_peça})$, $R_2(\text{Nome_fornecedor}, \text{Nome_proj})$ e $R_3(\text{Nome_peça}, \text{Nome_proj})$ de FORNECE. Se essa restrição for mantida, as tuplas abaixo da linha tracejada na Figura 15.15(c) devem existir em algum estado válido da relação FORNECE, que também contém as tuplas acima da linha tracejada. A Figura 15.15(d) mostra como a relação FORNECE com a dependência de junção é decomposta em três relações R_1 , R_2 e R_3 que estão, cada uma, na 5FN. Observe que a aplicação de uma junção natural a *duas quaisquer* dessas relações *produz tuplas falsas*, mas a aplicação de uma junção natural a *todas as três juntas* não as produz. O leitor deverá verificar isso no exemplo de relação da Figura 15.15(c) e suas projeções na Figura 15.15(d). Isso porque somente a DJ existe, mas nenhuma MVD é especificada. Observe, também, que a $DJ(R_1, R_2, R_3)$ é especificada em *todos* os estados de relação válidos, não apenas sobre aquele mostrado na Figura 15.15(c).

A descoberta de DJs em bancos de dados práticos com centenas de atributos é quase impossível. Isso só pode ser feito com um grande grau de intuição sobre os dados da parte do projetista. Portanto, a prática atual do projeto de banco de dados não presta muita atenção a elas.

Resumo

Neste capítulo, discutimos várias armadilhas no projeto de banco de dados relacional usando argumentos intuitivos. Identificamos informalmente algumas das medidas para indicar se um esquema de relação é *bom* ou *ruim*, e fornecemos diretrizes informais para um bom projeto. Essas diretrizes são baseadas na realização de um projeto conceitual cuidadoso no modelo ER e EER, seguindo o procedimento de mapeamento do Capítulo 9 corretamente, para mapear entidades e relacionamentos em relações. A imposição apropriada dessas diretrizes e a falta de redundância evitarão as anomalias de inserção/exclusão/atualização, e a geração de dados falsos. Recomendamos limitar os valores NULL, que causam problemas durante operações SELEÇÃO, JUNÇÃO e de agregação. Depois, apresentamos alguns conceitos formais que nos permitem realizar o projeto relacional de uma maneira de cima para baixo ao analisar as relações individualmente. Definimos esse processo de projeto pela análise e decomposição, introduzindo o processo de normalização.

Definimos o conceito de dependência funcional, que é a ferramenta básica para analisar esquemas relacionais, e discutimos algumas de suas propriedades. As dependências funcionais especificam as restrições semânticas

entre os atributos de um esquema de relação. Em seguida, descrevemos o processo de normalização para obter bons projetos ao testar relações para tipos indesejáveis de dependências funcionais *problemáticas*. Oferecemos um tratamento da normalização sucessiva com base em uma chave primária predefinida em cada relação, e depois relaxamos esse requisito e fornecemos definições mais gerais da segunda forma normal (2FN) e terceira forma normal (3FN), que levam em conta todas as chaves candidatas de uma relação. Apresentamos exemplos para ilustrar como, usando a definição geral da 3FN, determinada relação pode ser analisada e decomposta para eventualmente gerar um conjunto de relações na 3FN.

Apresentamos a Forma Normal Boyce-Codd (FNBC) e discutimos como ela é uma forma mais forte da 3FN. Também ilustramos como a decomposição de uma relação não FNBC deve ser feita considerando o requisito de decomposição não aditiva. Então, introduzimos a quarta forma normal com base em dependências multivaloradas, que normalmente surgem devido à mistura de atributos multivalorados independentes em uma única relação. Por fim, apresentamos a quinta forma normal, que é baseada na dependência de junção, e que identifica uma restrição peculiar que faz que uma relação seja decomposta em vários componentes, de modo que eles sempre produzam a relação original de volta, após uma junção. Na prática, a maioria dos projetos comerciais seguiu as formas normais até a FNBC. A necessidade de decomposição para a 5FN raramente surge na prática, e as dependências de junção são difíceis de detectar para a maioria das situações práticas, tornando a 5FN de valor mais teórico.

O Capítulo 16 apresentará a síntese e também a decomposição de algoritmos para o projeto de banco de dados relacional baseado em dependências funcionais. Em relação à decomposição, discutimos os conceitos de *junção não aditiva* (ou *sem perdas*) e *preservação de dependência*, que são impostos por alguns desses algoritmos. Outros tópicos no Capítulo 16 incluem um tratamento mais detalhado das dependências funcionais e multivaloradas, além de outros tipos de dependências.

Perguntas de revisão

- 15.1. Discuta a semântica de atributo como uma medida informal de boas práticas para um esquema de relação.
- 15.2. Discuta as anomalias de inserção, exclusão e modificação. Por que elas são consideradas ruins? Ilustre com exemplos.
- 15.3. Por que os NULLs em uma relação devem ser evitados ao máximo possível? Discuta o problema das tuplas falsas e como podemos impedi-lo.
- 15.4. Indique as diretrizes informais para o projeto de esquema de relação que discutimos. Ilustre como a violação dessas diretrizes pode ser prejudicial.
- 15.5. O que é uma dependência funcional? Quais são as possíveis fontes da informação que definem as dependências funcionais que se mantêm entre os atributos de um esquema de relação?

- 15.6. Por que não podemos deduzir uma dependência funcional automaticamente com base em um estado de relação em particular?
- 15.7. A que se refere o termo *relação não normalizada*? Como as formas normais se desenvolveram historicamente desde a primeira forma normal até a forma normal de Boyce-Codd?
- 15.8. Defina a primeira, segunda e terceira formas normais quando somente chaves primárias são consideradas. Como as definições gerais da 2FN e 3FN, que consideram todas as chaves de uma relação, diferem daquelas que consideram apenas chaves primárias?
- 15.9. Que dependências indesejáveis são evitadas quando uma relação está na 2FN?
- 15.10. Que dependências indesejáveis são evitadas quando uma relação está na 3FN?
- 15.11. De que maneira as definições generalizadas da 2FN e 3FN estendem as definições além das chaves primárias?
- 15.12. Defina a forma normal de Boyce-Codd. Como ela difere da 3FN? Por que ela é considerada uma forma mais forte de 3FN?
- 15.13. O que é dependência multivalorada? Quando ela surge?
- 15.14. Uma relação com duas ou mais colunas sempre tem uma MVD? Mostre com um exemplo.
- 15.15. Defina a quarta forma normal. Quando ela é violada? Quando ela costuma ser aplicada?
- 15.16. Defina a dependência de junção e a quinta forma normal.
- 15.17. Por que a 5FN também é chamada de forma normal projeção-junção (FNPJ)?
- 15.18. Por que os projetos de banco de dados práticos normalmente visam a FNBC, e não visam às formas normais mais altas?
- b. Cada departamento é descrito por um nome (Dnome), código de departamento (Dcodigo), número de escritório (Descritorio), telefone de escritório (Dtelefone) e faculdade (Dfaculdade). Tanto o nome quanto o código possuem valores únicos para cada departamento.
- c. Cada disciplina tem um nome (Dnome), descrição (Ddesc), número (Dnum), número de horas semestrais (Credito), nível (Nivel) e departamento de oferta (Num_dep). O número da disciplina é único para cada curso.
- d. Cada turma tem um professor (Unome), semestre (Semestre), ano (Ano), disciplina (Disciplina_turma) e número de turma (Num_turma). O número de turma distingue diferentes turmas da mesma disciplina que são lecionadas durante o mesmo semestre/ano; seus valores são 1, 2, 3, ..., até o número total de turmas lecionadas durante cada semestre.
- e. Um registro de nota refere-se a um aluno (Cpf), uma turma em particular e uma nota (Nota).
- Crie um esquema de banco de dados relacional para essa aplicação de banco de dados. Primeiro, mostre todas as dependências funcionais que devem ser mantidas entre os atributos. Depois, projete esquemas de relação para o banco de dados que estejam, cada uma, na 3FN ou na FNBC. Especifique os principais atributos de cada relação. Observe quaisquer requisitos não especificados e faça suposições apropriadas para tornar a especificação completa.
- 15.20. Que anomalias de atualização ocorrem nas relações FUNC_PROJ e FUNC_DEP das figuras 15.3 e 15.4?
- 15.21. Em que forma normal está o esquema de relação LOTES da Figura 15.12(a) com relação às interpretações restritivas da forma normal que levam em conta *apenas a chave primária*? Ela estaria na mesma forma normal se as definições gerais da forma normal fossem usadas?
- 15.22. Prove que qualquer esquema de relação com dois atributos está na FNBC.
- 15.23. Por que ocorrem tuplas falsas no resultado da junção das relações FUNC_PROJ1 e FUNC_LOCAL da Figura 15.5 (resultado mostrado na Figura 15.6)?
- 15.24. Considere a relação universal $R = \{A, B, C, D, E, F, G, H, I, J\}$ e o conjunto de dependências funcionais $F = \{ \{A, B\} \rightarrow \{C\}, \{A\} \rightarrow \{D, E\}, \{B\} \rightarrow \{F\}, \{F\} \rightarrow \{G, H\}, \{D\} \rightarrow \{I, J\} \}$. Qual é a chave para R ? Decomponha R em relações na 2FN e depois na 3FN.
- 15.25. Repita o Exercício 15.24 para o seguinte conjunto de dependências funcionais $G = \{ \{A, B\} \rightarrow \{C\}, \{B, D\} \rightarrow \{E, F\}, \{A, D\} \rightarrow \{G, H\}, \{A\} \rightarrow \{I\}, \{H\} \rightarrow \{J\} \}$.

Exercícios

- 15.19. Suponha que tenhamos os seguintes requisitos para um banco de dados de universidade, que é usado para registrar os históricos dos alunos:
- a. A universidade registra o nome de cada aluno (Anome), o número do aluno (Anum), o número do Cadastro de Pessoa Física (Cpf), o endereço moradia (Aendereco_mora) e o número do telefone (Atelefone_mora), endereço permanente (Aendereco_fixo) e telefone (Atelefone_fixo), data de nascimento (Datanasc), sexo (Sexo), tipo_aluno ('calouro', 'veterano', ..., 'graduado'), departamento principal (Dep_princ), departamento secundário (Dep_sec) (se houver) e programa de título (Titulacao) ('bacharel', 'mestrado', ..., 'doutorado'). Tanto Cpf quanto o número do aluno possuem valores únicos para cada um.

15.26. Considere a seguinte relação:

A	B	C	NUM_TUPLA
10	b1	c1	1
10	b2	c2	2
11	b4	c1	3
12	b3	c4	4
13	b1	c1	5
14	b3	c4	6

a. Dada a extensão (estado) anterior, qual das seguintes dependências *pode ser mantida* na relação acima? Se a dependência não puder ser mantida, explique por que, *especificando as tuplas que causam a violação*.

i. $A \rightarrow B$, ii. $B \rightarrow C$, iii. $C \rightarrow B$, iv. $B \rightarrow A$, v. $C \rightarrow A$

b. A relação acima tem uma chave candidata em potencial? Se tiver, qual é? Se não, por quê?

15.27. Considere uma relação $R(A, B, C, D, E)$ com as seguintes dependências:

$AB \rightarrow C$, $CD \rightarrow E$, $DE \rightarrow B$

AB é uma chave candidata dessa relação? Se não for, ABD é? Explique sua resposta.

15.28. Considere a relação R , que tem atributos que mantém horários de disciplinas e turmas em uma universidade; $R = \{Nr_disciplina, Nr_turma, Dep_oferece, Horas_credits, Nivel_disciplina, Cpf_professor, Semestre, Ano, Horas_dias, Nr_sala, Nr_de_alunos\}$. Suponha que as seguintes dependências funcionais sejam mantidas em R :

$\{Nr_disciplina\} \rightarrow \{Dep_oferece, Horas_credits, Nivel_disciplina\}$

$\{Nr_disciplina, Nr_turma, Semestre, Ano\} \rightarrow \{Horas_dias, Nr_sala, Nr_de_alunos, Cpf_professor\}$

$\{Nr_sala, Horas_dias, Semestre, Ano\} \rightarrow \{Cpf_professor, Nr_disciplina, Nr_turma\}$

Tente determinar quais conjuntos de atributos formam chaves de R . Como você normalizaria essa relação?

15.29. Considere as seguintes relações para um banco de dados de aplicação de processamento de pedido na ABC, Inc.

PEDIDO (Pnum, Pdata, Custo, Quantia_total)

ITEM_PEDIDO (Pnum, Inum, Qtd_pedida, Preco_total, Desconto_por)

Suponha que cada item tenha um desconto diferente. O Preco_total refere-se a um item, Pdata é a data em que o pedido foi feito e Quantia_total é o valor do pedido. Se aplicarmos uma junção natural nas relações ITEM_PEDIDO e PEDIDO nesse banco de dados, como será o esquema de relação resultante? Qual será sua chave? Mostre as DFs nessa relação resultante. Ela está na 2FN?

Está na 3FN? Por quê? (Indique as suposições, se você fizer alguma.)

15.30. Considere a seguinte relação:

VENDA_CARRO (Num_carro, Data_venda, Num_vendedor, Comissao_por, Desconto_tempo)

Suponha que um carro possa ser vendido por vários vendedores e, portanto, $\{Num_carro, Num_vendedor\}$ é a chave primária. Dependências adicionais são

$Data_venda \rightarrow Desconto_tempo$ e

$Num_vendedor \rightarrow Comissao_por$

Com base na chave primária dada, essa relação está na 1FN, 2FN ou 3FN? Por quê? Como você a normalizaria completamente com sucesso?

15.31. Considere a seguinte relação para livros publicados:

LIVRO (Titulo_livro, Nome_autor, Tipo_livro, Lista_preco, Afiliacao_autor, Editora)

Afiliacao_autor refere-se à afiliação do autor. Suponha que existam as seguintes dependências:

$Titulo_livro \rightarrow Editora, Tipo_livro$

$Tipo_livro \rightarrow Lista_preco$

$Nome_autor \rightarrow Afiliacao_autor$

a. Em que forma normal essa relação está? Explique sua resposta.

b. Aplique a normalização até não poder decompor mais a relação. Indique os motivos por trás de cada decomposição.

15.32. Este exercício lhe pede para converter declarações de negócios em dependências. Considere a relação DISCO_RIGIDO (Numero_de_serie, Fabricante, Modelo, Lote, Capacidade, Revendedor). Cada tupla na relação DISCO_RIGIDO contém informações sobre uma unidade de disco com um Numero_de_serie exclusivo, criado por um fabricante, com um número de modelo em particular, lançado em certo lote, que tem determinada capacidade de armazenamento e é vendido por certo revendedor. Por exemplo, a tupla Disco_rigido ('1978619', 'WesternDigital', 'A2235X', '765234', 500, 'CompUSA') especifica que a WesternDigital fabricou uma unidade de disco com número de série 1978619 e número de modelo A2235X, lançado no lote 765234; ele tem 500 GB e é vendido pela CompUSA.

Escreva cada uma das seguintes dependências como uma DF:

a. O fabricante e número de série identificam a unidade com exclusividade.

b. Um número de modelo é registrado por um fabricante e, portanto, não pode ser usado por outro fabricante.

- c. Todas as unidades de disco em determinado lote são do mesmo modelo.
- d. Todas as unidades de disco de certo modelo de um fabricante em particular possuem exatamente a mesma capacidade.

15.33. Considere a seguinte relação:

R (Num_medico, Num_paciente, Data, Diagnostico, Codigo_tratamento, Gasto)

Na relação acima, uma tupla descreve uma visita de um paciente a um médico junto com o código de tratamento e gasto diário. Suponha que o diagnóstico seja determinado (exclusivamente) para cada paciente por um médico. Suponha que cada código de tratamento tenha um custo fixo (independente do paciente). Essa relação está na 2FN? Justifique sua resposta e decomponha, se necessário. Depois, argumente se é necessária uma maior normalização para 3FN e, se preciso, realize-a.

15.34. Considere a seguinte relação:

VENDA_CARRO (Id_carro, Tipo_opcao, Opcao_listapreco, Data_venda, Opcao_descontopreco)

Essa relação se refere às opções instaladas nos carros (por exemplo, controle de navegação) que foram vendidas em um revendedor, e a lista de preços e descontos das opções.

Se $IDcarro \rightarrow Data_venda$ e $Tipo_opcao \rightarrow Opcao_listapreco$ e $IDcarro, Tipo_opcao \rightarrow Opcao_descontopreco$, argumente usando a definição generalizada da 3FN de que essa relação não está na 3FN. Depois, argumente com base em seu conhecimento da 2FN, por que ela sequer está na 2FN.

15.35. Considere a relação:

LIVRO (Nome_livro, Autor, Edicao, Ano)

com os dados:

Nome_livro	Autor	Edicao	Ano_copyright
Sistemas BD	Navathe	4	2004
Sistemas BD	Elmasri	4	2004
Sistemas BD	Elmasri	5	2007
Sistemas BD	Navathe	5	2007

- a. Com base em um conhecimento de senso comum dos dados acima, quais são as chaves candidatas possíveis dessa relação?
- b. Justifique que essa relação tem a MVD $\{ Livro \} \twoheadrightarrow \{ Autor \} \mid \{ Edicao, Ano \}$.
- c. Qual seria a decomposição dessa relação com base na MVD acima? Avalie cada relação resultante para a forma normal mais alta que ela possui.

15.36. Considere a seguinte relação:

VIAGEM (Id_viagem, Data_inicio, Cidades_visitadas, Cartoes_usados)

Essa relação refere-se a viagens de negócios feitas por vendedores da empresa. Suponha que VIAGEM tenha uma única Data_inicio, mas envolva muitas Cidades, e os vendedores podem usar múltiplos cartões de crédito na viagem. Crie uma população fictícia da tabela.

- a. Discuta quais DFs e/ou MVDs existem na relação.
- b. Mostre como você tratará de sua normalização.

Exercício de laboratório

Nota: o exercício a seguir usa o sistema DBD (*Data Base Designer*) que é descrito no manual do laboratório. O esquema relacional R e conjunto de dependências funcionais F precisam ser codificados como listas. Como exemplo, R e F para este problema são codificados como:

$R = [a, b, c, d, e, f, g, h, i, j]$

$F = [[a, b], [c]],$

$[[a], [d, e]],$

$[[b], [f]],$

$[[f], [g, h]],$

$[[d], [i, j]]]$

Como o DBD é implementado em Prolog, o uso de termos em maiúsculas é reservado para variáveis na linguagem e, portanto, constantes em minúsculas são usadas para codificar os atributos. Para obter mais detalhes sobre o sistema DBD, por favor, consulte o manual do laboratório.

15.37. Usando o sistema DBD, verifique suas respostas para os seguintes exercícios:

- a. 15.24 (3FN apenas)
- b. 15.25
- c. 15.27
- d. 15.28

Bibliografia selecionada

As dependências funcionais foram introduzidas originalmente por Codd (1970). As definições originais da primeira, segunda e terceira formas normais também foram definidas em Codd (1972a), onde pode ser encontrada uma discussão sobre anomalias de atualização. A Forma Normal de Boyce-Codd foi definida em Codd (1974). A definição alternativa da terceira forma normal é dada em Ullman (1988), assim como a definição da FNBC que mostramos aqui. Ullman (1988), Maier (1983) e Atzeni e De Antonellis (1993) contêm muitos dos teoremas e provas referentes a dependências funcionais.

Outras referências à teoria do projeto relacional serão dadas no Capítulo 16.