4

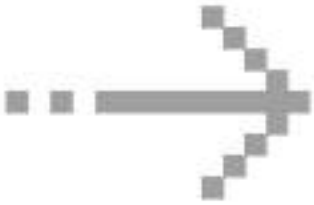
série livros didáticos informática ufrgs

projeto de banco de dados

6ª edição

carlos alberto heuser

bookman®



projeto de banco de dados

o autor

Carlos Alberto Heuser é engenheiro eletricista e mestre em Ciência da Computação pela UFRGS e doutor em Informática pela Universidade de Bonn, Alemanha. Atua na área da computação desde 1970, inicialmente como profissional de TI e, desde 1981, como docente do Instituto de Informática da UFRGS, onde é professor titular. É pesquisador na área de banco de dados, tendo publicado artigos em periódicos e congressos nacionais e internacionais, e atua nas áreas de sistemas Web e integração de informações.

COPIA NÃO AUTORIZADA É CRIME
ABDR
ASSOCIAÇÃO BRASILEIRA DE DIREITOS REPROGRÁFICOS
RESPEITE O DIREITO AUTORAL

H595p Heuser, Carlos Alberto.
Projeto de banco de dados [recurso eletrônico] / Carlos Alberto Heuser. – 6. ed. – Dados eletrônicos. – Porto Alegre : Bookman, 2009.

Editado também como livro impresso em 2009.
ISBN 978-85-7780-452-8

1. Banco de dados – Projeto. 2. Banco de dados – Modelo. I. Título.

CDU 004.6

Catalogação na publicação: Renata de Souza Borges CRB-10/Prov-021/08

carlos alberto heuser

projeto de banco de dados

6ª edição

Reimpressão 2010

2009

© Artmed Editora SA, 2009

Leitura final: *Monica Stefani*

Supervisão editorial: *Arysinha Jacques Affonso*

Capa e projeto gráfico: *Tatiana Sperhacke – TAT studio*

Imagem de capa: © *iStockphoto.com/Andrzej Thiel*

Editoração eletrônica: *Techbooks*

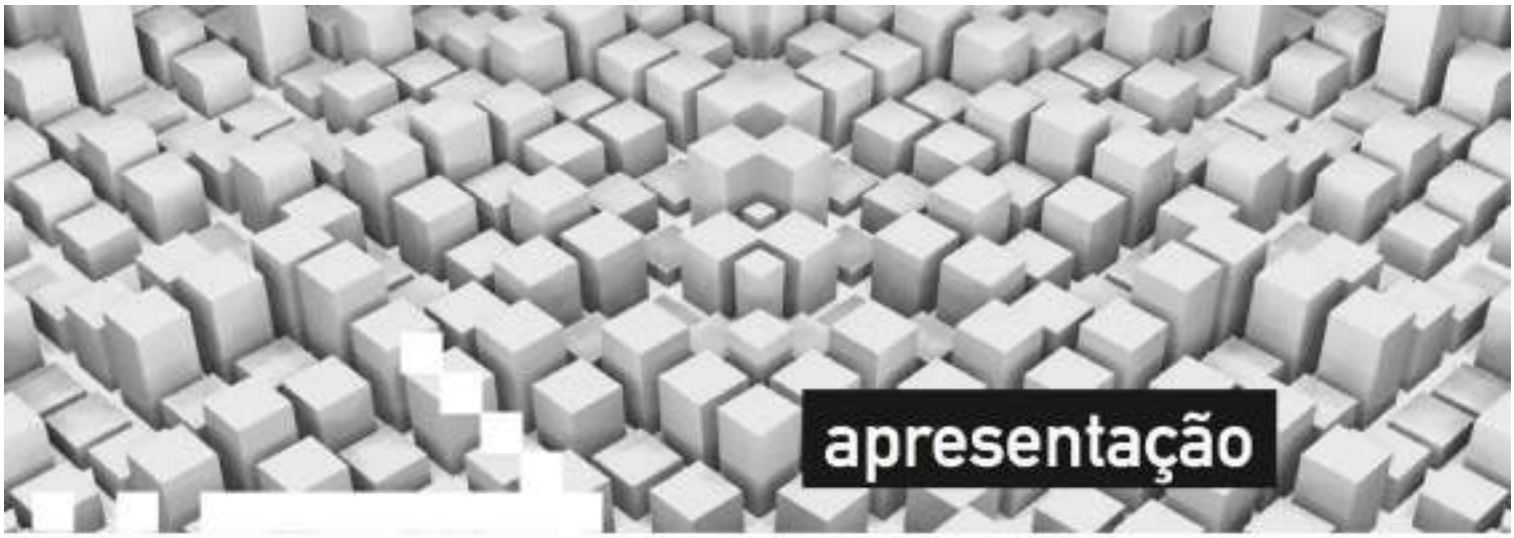
Reservados todos os direitos de publicação, em língua portuguesa, à
ARTMED® EDITORA S.A.
(BOOKMAN® COMPANHIA EDITORA é uma divisão da ARTMED® EDITORA S.A.)
Av. Jerônimo de Ornelas, 670 - Santana
90040-340 Porto Alegre RS
Fone (51) 3027-7000 Fax (51) 3027-7070

É proibida a duplicação ou reprodução deste volume, no todo ou em parte, sob quaisquer formas ou por quaisquer meios (eletrônico, mecânico, gravação, fotocópia, distribuição na Web e outros), sem permissão expressa da Editora.

SÃO PAULO
Av. Embaixador Macedo de Soares, 10735 – Galpão 5 – Cond. Espace Center
Vila Anastácio – 05095-035 São Paulo SP
Fone (11) 3665-1100 Fax (11) 3667-1333

SAC 0800 703-3444

IMPRESSO NO BRASIL
PRINTED IN BRAZIL



apresentação

A Série *Livros Didáticos* do Instituto de Informática da Universidade Federal do Rio Grande do Sul tem como objetivo a publicação de material didático para disciplinas ministradas em cursos de graduação em Computação e Informática, ou seja, para os cursos de Bacharelado em Ciência da Computação, de Bacharelado em Sistemas de Informação, de Engenharia da Computação e de Licenciatura em Informática. A série é desenvolvida tendo em vista as Diretrizes Curriculares do MEC e é resultante da experiência dos professores do Instituto de Informática e dos colaboradores externos no ensino e na pesquisa.

Os primeiros títulos, *Fundamentos da Matemática Intervalar* e *Programando em Pascal XSC* (esgotados), foram publicados em 1997 no âmbito do Projeto Aritmética Intervalar Paralela (ArInPar), financiados pelo ProTeM - CC CNPq/ Fase II. Essas primeiras experiências serviram de base para os volumes subseqüentes, os quais se caracterizam como livros-texto para disciplinas dos cursos de Computação e Informática.

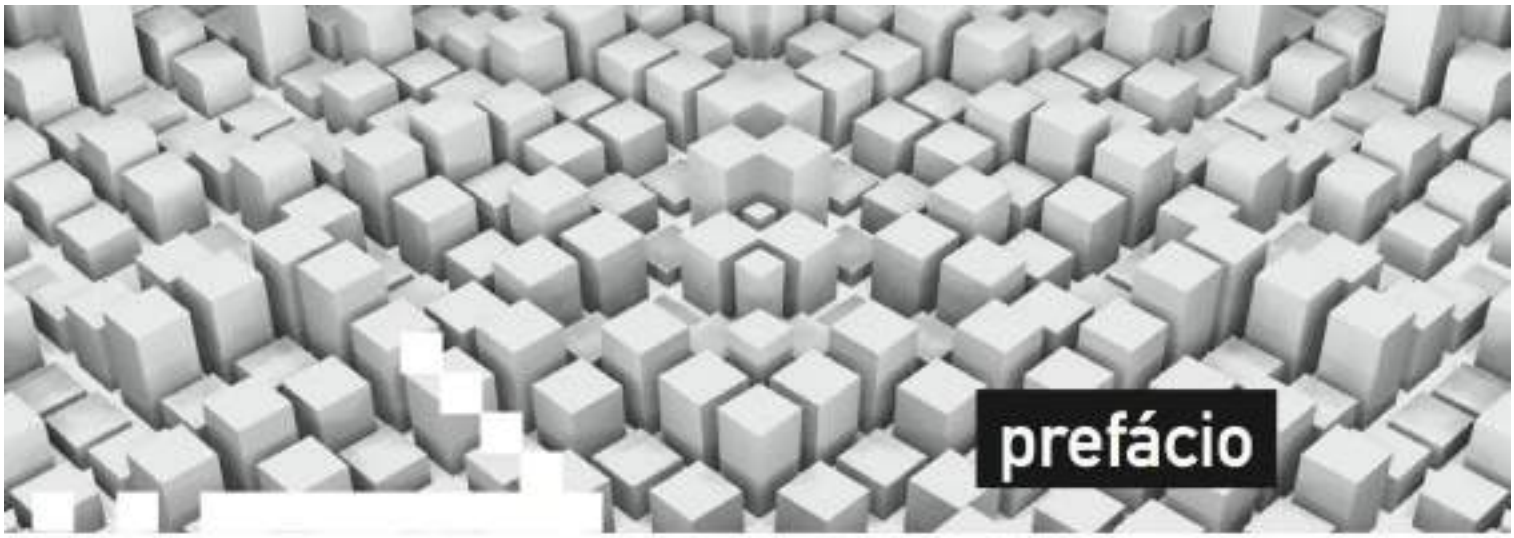
Em seus títulos mais recentes, a Série Livros Didáticos tem contado com a colaboração de professores externos que, em parceria com professores do Instituto, estão desenvolvendo livros de alta qualidade e valor didático. Hoje a série está aberta a qualquer autor de reconhecida capacidade.

O sucesso da experiência com esses livros, aliado à responsabilidade que cabe ao Instituto na formação de professores e pesquisadores em Computação e Informática, conduziu à ampliação da abrangência e à institucionalização da série.

Em 2008, um importante passo foi dado para a consolidação e ampliação de todo o trabalho: a publicação dos livros pela Artmed Editora S.A., por meio do selo Bookman. Hoje são 15 os títulos publicados – uma lista completa,

incluindo os próximos lançamentos, encontra-se nas orelhas desta obra – ampliando a oferta aos leitores da série. Sempre com a preocupação em manter nível compatível com a elevada qualidade do ensino e da pesquisa desenvolvidos no âmbito do Instituto de Informática da UFRGS e no Brasil.

Prof. Paulo Blauth Menezes
Comissão Editorial da Série Livros Didáticos
Instituto de Informática da UFRGS



prefácio

objetivos do livro

Os sistemas de gerência de banco de dados (SGBD) surgiram no início da década de 1970 com o objetivo de facilitar a programação de aplicações de banco de dados (BD). Os primeiros sistemas eram caros e difíceis de usar, requerendo especialistas treinados para usar o SGBD específico.

Nessa mesma época, houve um investimento considerável em pesquisa na área de banco de dados. Estas pesquisas resultaram em um tipo de SGBD, o *SGBD relacional*. A partir da década de 1980 e devido ao barateamento das plataformas de *hardware/software* necessárias à execução de um SGBD relacional, este tipo de SGBD passou a dominar o mercado, tendo se convertido em padrão internacional. Hoje, o desenvolvimento de sistemas de informação ocorre quase que exclusivamente sobre banco de dados, com uso de SGBD relacional.

Além do SGBD relacional, as pesquisas na área de BD resultaram também em um conjunto de técnicas, processos e notações para o *projeto de BD*.O projeto de BD, que inicialmente era feito com técnicas empíricas por alguns poucos especialistas no SGBD específico, hoje é executado com o auxílio de técnicas padronizadas e suportadas por um *software* específico (ferramentas CASE). Ao longo do tempo, formou-se um conjunto de conhecimentos sobre projeto de BD, que é amplamente aceito e deve ser dominado por qualquer profissional de informática. Estes conhecimentos são ministrados nas universidades, já em cursos de graduação, nas disciplinas de fundamentos de banco de dados ou mesmo em disciplinas específicas de projeto de banco de dados.

O projeto de um banco de dados ocorre usualmente em três etapas. A primeira etapa, a *modelagem conceitual*, captura formalmente os requisitos

de informação de um banco de dados. A segunda etapa, o *projeto lógico*, define, no nível do SGBD, as estruturas de dados que implementarão os requisitos identificados na modelagem conceitual. A terceira etapa, o *projeto físico*, estabelece parâmetros físicos de acesso ao BD, procurando otimizar o desempenho do sistema como um todo.

Este livro ensina o projeto de banco de dados, cobrindo as duas primeiras etapas do ciclo de vida de um banco de dados, a da modelagem conceitual e a do projeto lógico.

Na modelagem conceitual, o livro utiliza a *abordagem entidade-relacionamento (ER)* de Peter Chen, considerada hoje um padrão *de facto* de modelagem de dados. Além de apresentar os conceitos e as notações da abordagem ER, o livro apresenta regras e heurísticas para a construção de modelos.

Com referência ao *projeto lógico*, o livro cobre tanto o projeto propriamente dito (transformação de modelos ER em modelos relacionais) quanto a *engenharia reversa de BD* (extração de modelo conceitual a partir de modelo lógico relacional ou de modelos de arquivos convencionais).

→ público-alvo

O livro atende três públicos.

O primeiro é o de *alunos de graduação* de ciência da computação, de informática ou cursos semelhantes. O livro foi concebido para ser usado no ensino de uma primeira abordagem ao tema, o que normalmente ocorre em disciplinas de fundamentos de banco de dados ou de projeto de banco de dados. São disciplinas correspondentes à matéria obrigatória banco de dados das diretrizes curriculares do MEC para a área de computação, bem como à matéria obrigatória T3 do Currículo de Referência 96 da Sociedade Brasileira de Computação (SBC). O livro origina-se de notas de aula, que escrevi para suportar parte de uma disciplina, que ministro há vários anos no curso de Bacharelado em Ciência da Computação da UFRGS. Para cobrir todo o livro, necessita-se de pelo menos 20 horas/aula. Se forem executados alguns dos estudos de caso apresentados, este tempo deve ser estendido.

Outro público é o daqueles *profissionais de Informática* que, em sua formação, não tiveram contato com os modelos e as técnicas envolvidas no projeto

de banco de dados. Neste caso, o livro pode ser usado para auto-estudo ou para suporte a cursos de extensão ou de especialização em Projeto de Banco de Dados. Mesmo para profissionais que já conheçam modelagem de dados, o livro pode ser útil por apresentar um método para engenharia reversa de banco de dados. Este método é importante na atualidade, visto que muitas organizações contam com sistemas legados e estão envolvidas na tarefa de migrar estes sistemas para bancos de dados relacionais.

Um terceiro público é o de *usuários* de SGBD pessoais, que desejam sistematizar o projeto de seus bancos de dados. Para estes leitores, a parte referente à engenharia reversa provavelmente será demasiado avançada. Entretanto, o restante do livro é perfeitamente compreensível para aqueles que têm conhecimentos apenas introdutórios de informática.

→ organização

O livro está organizado de forma a não exigir conhecimentos prévios na área de banco de dados ou de engenharia de *software*.

O capítulo 1 apresenta os conceitos básicos de banco de dados necessários à compreensão do restante do texto. Ali são introduzidos conceitos como banco de dados, modelo de dados, sistema de gerência de banco de dados, modelo conceitual e modelo lógico. Se o leitor já dominar estes conceitos, poderá perfeitamente omitir este capítulo.

O capítulo 2 mostra a abordagem entidade-relacionamento, ensinando os conceitos básicos do modelo ER e a notação gráfica para a apresentação dos modelos. Como não há uma notação universalmente aceita para diagramas ER, neste capítulo preferi usar a notação original de Peter Chen. São apresentados tanto os conceitos básicos de entidade, atributo e relacionamento, quanto extensões do modelo ER em direção a modelos semânticos, como os conceitos de generalização/especialização e entidade associativa.

Enquanto o capítulo 2 visa à *compreensão* de modelos entidade-relacionamento, o capítulo 3 objetiva a *construção* de modelos ER. Além de apresentar o processo de modelagem, o capítulo inclui uma série de heurísticas que podem ser usadas na construção de modelos ER. Além disso, são discutidas as notações alternativas à de Peter Chen que aparecem mais freqüentemente na prática.

O capítulo 4 é uma introdução ao modelo relacional. Não se trata aqui de mostrar de forma completa o funcionamento de um SGBD relacional, mas apenas de transmitir o conhecimento mínimo necessário para a compreensão do restante do livro. Novamente, caso o leitor já conheça a abordagem relacional, poderá omitir este capítulo.

O capítulo 5 apresenta procedimentos para executar dois tipos de transformações entre modelos de dados. Uma transformação é o *projeto lógico* de banco de dados relacional, ou seja, a transformação de um modelo ER em um modelo relacional. A outra transformação é a *engenharia reversa de banco de dados relacional*, isto é, a transformação de um modelo lógico de banco de dados relacional em um modelo conceitual ER.

O capítulo 6 cobre a *engenharia reversa de arquivos*, apresentando regras para transformar a descrição de um conjunto de arquivos convencionais em um modelo de dados relacional. As regras baseiam-se na *normalização* de banco de dados relacional, que é apresentada no capítulo. Por tratar-se de um texto introdutório, são apresentadas apenas as quatro primeiras formas normais.

Finalmente, o capítulo 7 contém as soluções da maioria dos exercícios apresentados em cada um dos capítulos precedentes.

→ histórico das edições

Na quinta e na sexta edições, o livro sofreu sua mais profunda revisão desde seu lançamento. Houve alterações no projeto gráfico, na redação e no conteúdo.

O projeto gráfico do livro foi atualizado, com o objetivo fundamental de tornar a leitura mais fluída.

Na redação, erros de gramática e ortografia foram corrigidos e partes do texto foram reescritas, a fim de aumentar a legibilidade.

Já no conteúdo, as principais alterações são as relacionadas a seguir.

- Nas edições anteriores, procurei manter uma certa consistência entre a abordagem entidade-relacionamento e a abordagem orientada a objeto, tendo em vista que ambas as abordagens são importantes e coexistem na prática. O ponto em que isto se refletia mais fortemente era na apresentação da *generalização/especialização*. Para manter a consistência com a orientação a objeto, eu tratava somente da generalização/especialização *exclusiva*, que é aquela implementada na maioria das linguagens de programação OO.

  Nesta edição, há uma discussão mais completa da generalização/especialização, com a inclusão da generalização/especialização *compartilhada*, visto que esta continua aparecendo na maioria das variantes da abordagem ER e nas ferramentas CASE.

  A seção Generalização/especialização foi completamente reformulada, com a inclusão de alguns exercícios referentes ao assunto.
- A Seção Determinando construções, que discute critérios para determinar qual conceito da abordagem ER deve ser usado para representar um objeto de uma realidade modelada, foi reformulada. Foi incluída a Subseção Entidade relacionada *versus* especialização, que discute quando usar uma entidade relacionada e quando usar uma especialização. Também um exercício referente ao tema foi incluído.
- Com relação às variantes de notação diagramática de modelos ER (Seção Variantes da abordagem ER), incluí uma discussão da abordagem UML. Não se trata aqui de uma apresentação completa desta abordagem. Na verdade, procurei estabelecer uma relação entre os conceitos e a notação gráfica da UML com os conceitos e a notação gráfica da abordagem ER. Esta discussão me pareceu importante, dada a proliferação de ferramentas CASE que suportam a abordagem.
- Na discussão das alternativas que podem ser usadas para a implementação de relacionamentos em bancos de dados relacionais (Seção Implementação de relacionamentos e seguintes), nas edições anteriores, as alternativas eram classificadas nas categorias: "alternativa preferida", "pode ser usada" e "não usar". Entretanto, a classificação não era precisa e, na categoria "não usar", apareciam tanto alternativas que não fazem sentido do ponto de vista da abordagem relacional quanto alternativas de menor prioridade em relação às demais. Por esta razão, os critérios de classificação e o enquadramento das alternativas nestes critérios foram revistos.

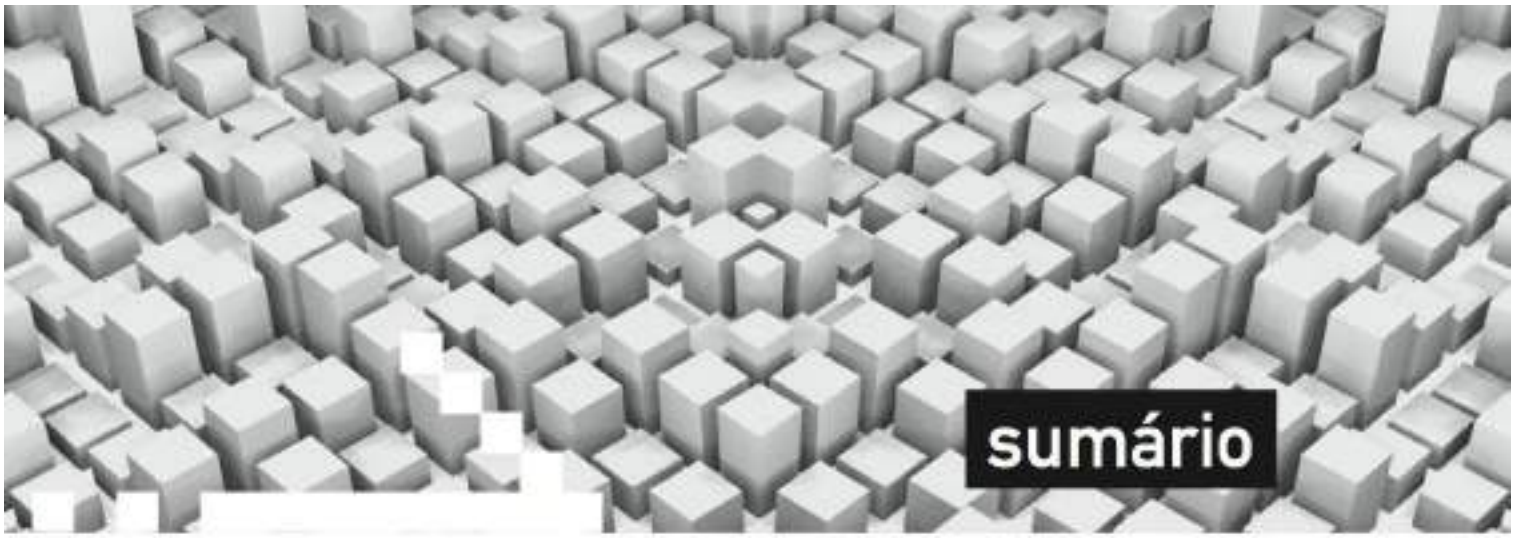
→ página www

A página http://www.inf.ufrgs.br/~heuser/livroProjBD/ é uma coletânea de informações relacionadas ao livro. Lá encontram-se erratas e informações sobre materiais didáticos e ferramentas CASE de demonstração que podem ser usadas no apoio ao texto.

→ agradecimentos

Versões preliminares deste livro foram usadas durante vários semestres, em cursos de graduação e de pós-graduação na UFRGS, e em instituições com as quais a UFRGS mantém convênio. Muitos alunos contribuíram com correções e sugestões. A versão preliminar também foi criteriosamente revisada pela querida e saudosa colega Lia G. Golendziner.

Após a publicação, vários leitores me passaram erratas do livro. Entre eles, Giuliano, que me passou uma extensa lista de erros de português e de quem infelizmente não fiquei com maiores referências, além de seu antigo e inoperante endereço de correio eletrônico ("robocapirai" em um provedor gratuito), bem como Aline Malanovicz, Marcelo Heinrich de Souza, Sérgio Dill e Zelindo Sílvio Petri.



sumário

1 → introdução 19

1.1 banco de dados **20**

1.1.1 compartilhamento de dados 20
1.1.2 sistema de gerência de banco de dados 23

1.2 modelos de banco de dados **24**

1.2.1 modelo conceitual 25
1.2.2 modelo lógico 26
1.2.3 modelo conceitual como modelo de organização 27
1.2.4 projeto de banco de dados 29

1.3 exercícios **30**

2 → abordagem entidade-relacionamento 33

2.1 entidade **34**

2.2 relacionamento **36**

2.2.1 conceituação 36
2.2.2 cardinalidade de relacionamentos 39
2.2.3 cardinalidade máxima 40
2.2.4 relacionamento ternário 43
2.2.5 cardinalidade mínima 45

2.3 exemplo de uso de entidades e relacionamentos 46

2.4 atributo 48

2.4.1 identificando entidades 50
2.4.2 identificando relacionamentos 53

2.5 generalização/especialização 54

2.6 entidade associativa 60

2.7 esquemas gráficos e textuais de modelos ER 62

2.8 exercícios 64

3 → construindo modelos ER 71

3.1 propriedades de modelos ER 72

3.1.1 um modelo ER é um modelo formal 72
3.1.2 modelos ER têm poder de expressão limitado 73
3.1.3 diferentes modelos podem ser equivalentes 75

3.2 determinando construções 77

3.2.1 atributo *versus* entidade relacionada 78
3.2.2 atributo *versus* especialização 79
3.2.3 entidade relacionada *versus* especialização 81
3.2.4 atributos opcionais e multivalorados 83

3.3 verificação do modelo 85

3.3.1 um modelo deve ser correto 85
3.3.2 um modelo deve ser completo 87
3.3.3 um modelo deve ser livre de redundância 87
3.3.4 um modelo deve refletir o aspecto temporal 89
3.3.5 entidade isolada e entidade sem atributos 94

3.4 estabelecimento de padrões 94

3.4.1 variantes da abordagem ER 94
3.4.2 uso de ferramentas de modelagem 100

3.5 estratégias de modelagem **100**

3.5.1 partindo de descrições de dados existentes 102
3.5.2 partindo do conhecimento de pessoas 103

3.6 exercícios **106**

4 abordagem relacional 119

4.1 composição de um banco de dados relacional **120**

4.1.1 tabela 120
4.1.2 chave 122
4.1.3 modelo de banco de dados relacional 128

4.2 consultas sobre o banco de dados **130**

4.3 exercícios **131**

5 transformações entre modelos 135

5.1 visão geral do projeto lógico **136**

5.2 transformação ER para relacional **137**

5.2.1 implementação inicial de entidades 140
5.2.2 implementação de relacionamentos 144
5.2.3 detalhes da implementação de relacionamentos 147
5.2.4 relacionamentos 1:1 147
5.2.5 relacionamentos 1:n 152
5.2.6 relacionamentos n:n 154
5.2.7 relacionamentos de grau maior que dois 154
5.2.8 implementação de generalização/especialização 156
5.2.9 refinamento do modelo relacional 162

5.3 engenharia reversa de modelos relacionais **169**

5.3.1 identificação da construção ER correspondente a cada tabela 170

5.3.2 definição de relacionamentos 1:n ou 1:1 173
5.3.3 definição de atributos 173
5.3.4 definição de identificadores de entidades..... 174

5.4 exercícios 176

6 → engenharia reversa de arquivos e normalização 183

6.1 introdução 184

6.2 visão geral do processo de engenharia reversa 185

6.3 documento exemplo.....186

6.4 representação na forma de tabela não-normalizada 187

6.5 normalização 190

6.5.1 passagem à primeira forma normal (1FN)..... 190
6.5.2 dependência funcional 195
6.5.3 passagem à segunda forma normal (2FN) 196
6.5.4 passagem à terceira forma normal (3FN)..... 200
6.5.5 passagem à quarta forma normal (4FN) 203
6.5.6 problemas da normalização 207

6.6 integração de modelos.....209

6.6.1 integração de tabelas com mesma chave 210
6.6.2 integração de tabelas com chaves contidas..... 211
6.6.3 volta à 2FN 213

6.7 finalização de modelo 214

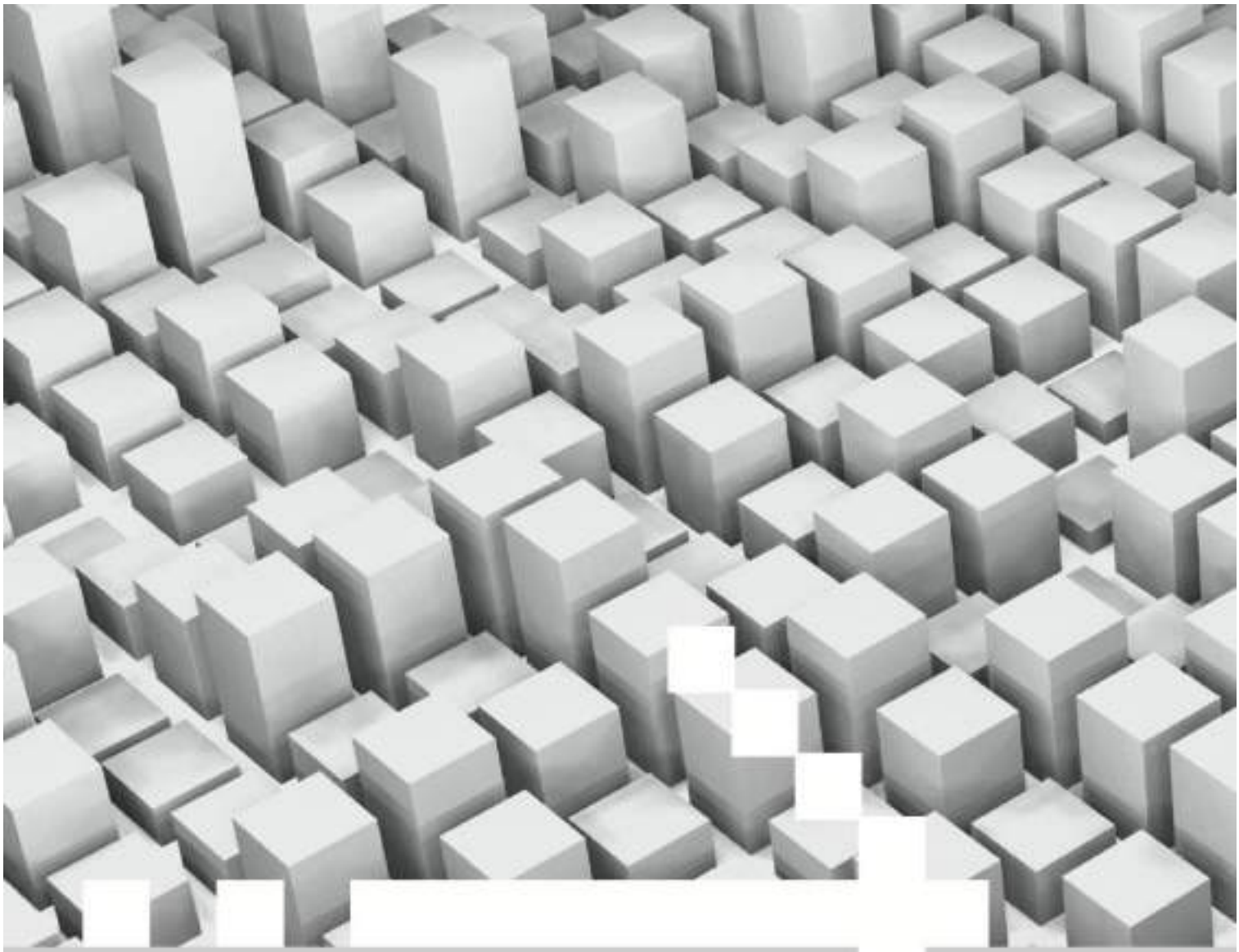
6.7.1 construção do modelo ER..... 214
6.7.2 verificação do modelo ER – limitações da normalização..... 214

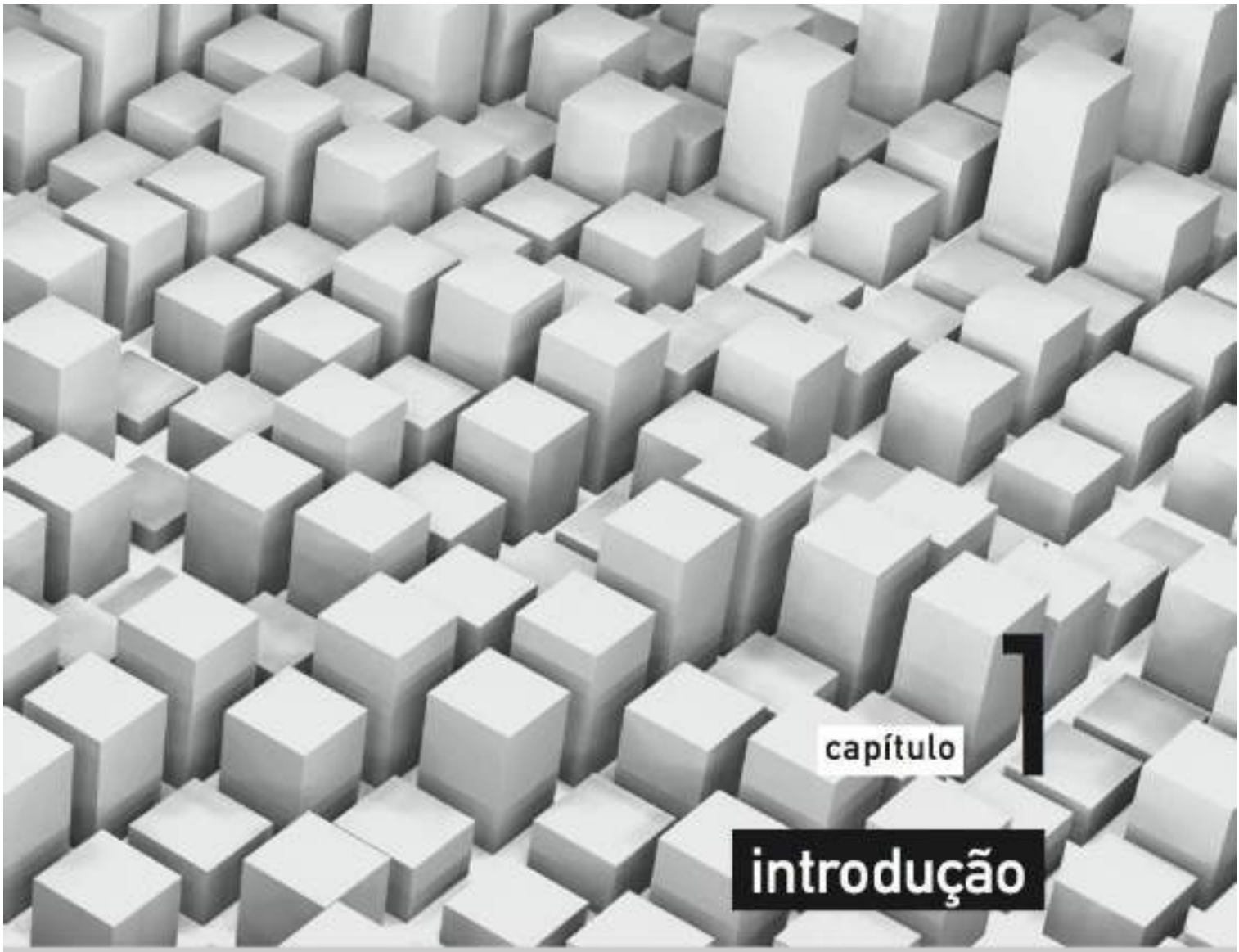
6.8 exercícios 214

soluções de exercícios selecionados 229

Referências 275

índice 279





capítulo 1

introdução

Este capítulo apresenta os conceitos da área de banco de dados necessários à compreensão do projeto de banco de dados. Além disso, fornece uma visão geral do processo do projeto de banco de dados. O leitor que já conhece os fundamentos da área provavelmente poderá passar ao próximo capítulo.

1.1 → banco de dados

1.1.1 compartilhamento de dados

Muitas vezes, a implantação da Informática em organizações ocorre de forma evolutiva e gradual. Inicialmente, apenas determinadas funções são automatizadas. Mais tarde, à medida que o uso da Informática vai se estabelecendo, novas funções vão sendo informatizadas.

Para exemplificar, vamos considerar uma indústria hipotética, na qual são executadas três funções:

- `vendas` – esta função concentra as atividades da indústria relativas ao contato com os clientes, como fornecimento de cotações de preços, vendas, e informações sobre disponibilidade de produtos.
- `produção` – esta função concentra as atividades da indústria relativas à produção, como planejamento da produção e controle do que foi produzido.
- `compras` – esta função concentra as atividades da indústria relativas à aquisição dos insumos necessários à produção, como cotações de preços junto a fornecedores, compras e acompanhamento do fornecimento.

No exemplo acima, os dados de um produto são usados em várias funções. Estes dados são necessários na função *Produção*, no planejamento da produção, pois para planejar o que vai ser produzido, é necessário conhecer como os produtos são estruturados (quais seus componentes) e como são produzidos. Os dados de produto também são necessários na função *Compras*, pois nesta função é necessário saber quais componentes devem ser adquiridos. Já na função *Vendas*, também é necessário conhecer dados de produtos, como seu preço, seu estoque atual, seu prazo de fabricação, etc.

Se cada uma das funções acima for informatizada de forma separada, sem considerar a informatização das demais funções, pode ocorrer que, para cada uma das funções, seja criado um arquivo separado de produtos (ver Figura 1.1).

Neste caso, surge o problema da *redundância de dados*. A redundância de dados ocorre quando uma determinada informação está representada no sistema em computador várias vezes. No exemplo, estão redundantes as infor-

Figura 1.1 Sistemas isolados.

mações referentes a um produto, que aparecem nos arquivos de produtos de cada um dos três sistemas.

Há dois tipos de redundância de dados, a redundância *controlada* de dados e a redundância *não controlada* de dados.

A redundância *controlada* de dados acontece quando o *software* tem conhecimento da múltipla representação da informação e garante a sincronia entre as diversas representações. Do ponto de vista do usuário externo ao sistema em computador, tudo acontece como se existisse uma única representação da informação. Essa forma de redundância é utilizada para melhorar a confiabilidade ou o desempenho global do sistema. Um exemplo é um sistema distribuído, onde uma mesma informação é armazenada em vários computadores, permitindo acesso rápido a partir de qualquer um deles.

A redundância *não controlada* de dados acontece quando a responsabilidade pela manutenção da sincronia entre as diversas representações de uma informação está com o usuário e não com o *software*. Este tipo de redundância deve ser evitado, pois traz consigo vários tipos de problemas:

- `entrada repetida da mesma informação` – a mesma informação é entrada várias vezes no sistema em computador. No exemplo da indústria, os dados de um produto são entrados no setor de vendas, no setor de produção e no setor de compras. Além de exigir trabalho desnecessário, a entrada repetida da mesma informação pode resultar em erros de transcrição de dados.
- `inconsistências de dados` – a responsabilidade por manter a sincronia entre as informações é do usuário. Por erro de operação, pode ocorrer que uma representação de uma informação seja modificada, sem que as de-

mais representações o sejam. Exemplificando, uma alteração na estrutura de um determinado produto pode ser informada através do sistema de produção e deixar de ser informada nos demais sistemas. A estrutura do produto passa a aparecer de forma diferente nos vários sistemas. O banco de dados passa a ter informações *inconsistentes*.

A solução para evitar a redundância não controlada de informações é o *compartilhamento de dados*. Nesta forma de processamento, cada informação é armazenada uma única vez, sendo acessada pelos vários sistemas que dela necessitam (Figura 1.2).

Ao conjunto de arquivos integrados que atendem a um conjunto de sistemas dá-se o nome de *banco de dados (BD)*[1].

banco de dados
=
conjunto de dados integrados que tem por objetivo atender a uma comunidade de usuários

O compartilhamento de dados tem reflexos na estrutura do software. A estrutura interna dos arquivos passa a ser mais complexa, pois estes devem ser construídos de forma a atender às necessidades dos diferentes sistemas. Para contornar este problema, usa-se um *sistema de gerência de banco de dados* (SGBD), conforme descrito na próxima seção.

Figura 1.2 Sistemas integrados com dados compartilhados.

[1] Em inglês, o termo utilizado hoje é *database* (base de dados). O termo *databank* (banco de dados) apareceu apenas nos primeiros trabalhos e caiu em desuso. Em português, ambos os termos podem ser usados. Neste texto, preferimos o termo *banco de dados*.

1.1.2 sistema de gerência de banco de dados

A programação de aplicações em computadores sofreu profundas modificações desde seus primórdios. No início, usando linguagens como COBOL, Basic, C e outras, os programadores incorporavam em um programa toda funcionalidade desejada. O programa continha as operações da interface de usuário, as transformações de dados e cálculos, as operações de armazenamento de dados, bem como as tarefas de comunicação com outros sistemas e programas.

Com o tempo, foram sendo identificadas funcionalidades comuns a muitos programas. Por exemplo, hoje, a grande maioria dos programas comunica-se com os usuários através de interfaces gráficas de janelas. Entretanto, normalmente, os programas não contêm todo o código referente à exibição dos dados na interface, mas utilizam *gerenciadores de interface de usuário*, conjuntos de rotinas que incluem as funcionalidades que um programador vai necessitar freqüentemente ao construir uma interface de usuário. Da mesma forma, para comunicar-se com processos remotos, os programas usam *gerenciadores de comunicação*. Para manter grandes repositórios compartilhados de dados, ou seja, para manter *bancos de dados*, são usados *sistemas de gerência de banco de dados*.

sistema de gerência de banco de dados (SGBD)
=
software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados

Essa modularização de programas tem várias vantagens. A *manutenção* de programas torna-se mais simples, pois uma separação clara de funções facilita a compreensão dos programas. A *produtividade* dos programadores também aumenta, já que os programas ficam menores, pois usam funções já construídas.

No mercado, há vários tipos de SGBD. Neste livro, nos concentramos em um tipo de SGBD, o *relacional*, que domina o mercado atualmente. Entretanto, muitas das idéias apresentadas nas seções referentes à modelagem de dados aplicam-se também a outros tipos, como os SGBDs orientados a objetos ou objeto-relacionais.

1.2 → modelos de banco de dados

Um *modelo de (banco de) dados* é uma descrição dos tipos de informações que estão armazenadas em um banco de dados. Por exemplo, no caso da indústria acima citado, o modelo de dados poderia informar que o banco de dados armazena informações sobre produtos e que, para cada produto, são armazenados seu código, preço e descrição. Observe que o modelo de dados não informa quais os produtos que estão armazenados no banco de dados, mas apenas que o banco de dados contém informações sobre produtos.

modelo de dados
=
descrição formal da estrutura de um banco de dados

Para construir um modelo de dados, usa-se uma *linguagem de modelagem de dados*. Linguagens de modelagem de dados podem ser classificadas de acordo com a forma de apresentar modelos, em linguagens *textuais* ou linguagens *gráficas*. Existem linguagens de modelagem para descrever modelos de dados em diferentes níveis de abstração e com diferentes objetivos. Cada representação de um modelo de dados através de uma linguagem de modelagem de dados recebe a denominação *esquema de banco de dados*.

De acordo com a intenção do modelador, um banco de dados pode ser modelado (descrito) em vários níveis de abstração. Um modelo de dados que servirá para explicar a um usuário leigo em informática qual é a organização de um banco de dados provavelmente não conterá detalhes sobre a representação em meio físico das informações. Já um modelo de dados usado por um técnico para otimizar a performance de acesso ao banco de dados conterá mais detalhes de como as informações estão organizadas internamente e, portanto, será menos abstrato.

No projeto de banco de dados, normalmente são considerados dois níveis de abstração de modelo de dados, o do *modelo conceitual* e o do *modelo lógico*.

Assim como é possível construir modelos de dados em vários níveis de abstração, também é possível usar diferentes técnicas, aplicando diferentes concei-

tos ao construir modelos. Ao conjunto de conceitos usados na construção de um modelo denominamos *abordagem de modelagem*[2]. Neste livro veremos diferentes abordagens, destinadas à modelagem de dados em diferentes níveis de abstração.

abordagem de modelagem
=
conjunto de conceitos usados para construir modelos

1.2.1 modelo conceitual

Um *modelo conceitual* é uma descrição do banco de dados de forma independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD.

modelo conceitual
=
modelo de dados abstrato, que descreve a estrutura de um banco de dados de forma independente de um SGBD particular

A técnica de modelagem conceitual mais difundida é a *abordagem entidade-relacionamento* (ER). Nesta técnica, um modelo conceitual é usualmente representado através de um diagrama, chamado *diagrama entidade-relacionamento* (DER). A Figura 1.3 apresenta um DER parcial para o problema da fábrica.

[2] Aqui cabe uma observação quanto à terminologia empregada no livro. Infelizmente, na Computação, mesmo em áreas bem estabelecidas, como a de banco de dados, a terminologia está longe de ser padronizada. Por exemplo, o termo "modelo de dados" aparece na literatura também com o significado de "abordagem", ou seja, de conjunto de conceitos para construir modelos. Autores que seguem esta linha usam, então, "modelo relacional" ou "modelo ER" para referir-se ao que, aqui, denominamos de "abordagem relacional" e "abordagem ER", respectivamente. Quando querem referir-se ao que aqui chamamos de "modelo", usam o termo "esquema".

A terminologia aqui empregada foi escolhida por ser de uso corrente e adotada por vários autores da área de banco de dados. Além disso, é coerente com a terminologia usada na Engenharia de *Software*, área correlata que também trata do problema de modelagem.

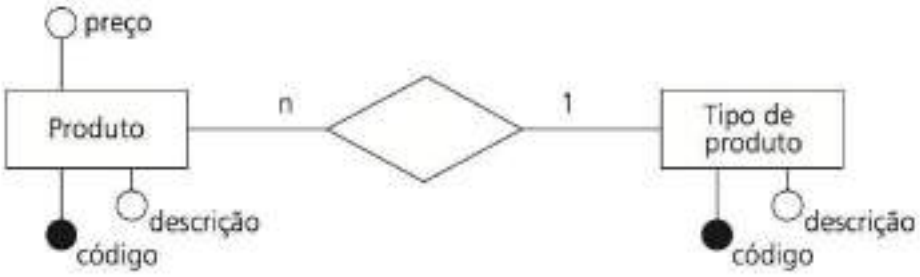


Figura 1.3 Exemplo de modelo conceitual.

Entre outras coisas, este modelo informa que o banco de dados contém dados sobre produtos e sobre tipos de produtos. Para cada produto, o banco de dados armazena o código, a descrição, o preço, bem como o tipo de produto ao qual está associado. Para cada tipo de produto, o banco de dados armazena o código, a descrição, bem como os produtos daquele tipo.

No capítulo 2, vamos aprender como modelos ER são compostos e, no capítulo 3, vamos ver como eles são construídos.

1.2.2 modelo lógico

Um *modelo lógico* é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo de SGBD que está sendo usado.

modelo lógico
=
modelo de dados que representa a estrutura de dados de um banco de dados conforme vista pelo usuário do SGBD

No presente livro, serão tratados apenas modelos lógicos referentes a SGBD relacional. Em um SGBD relacional, os dados estão organizados na forma de tabelas. A Figura 1.4 mostra um exemplo de BD relacional projetado a partir do modelo conceitual mostrado na Figura 1.3.

Um modelo lógico de um BD relacional deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas. O modelo lógico para o BD em questão é o seguinte:

```
TipoDeProduto (CodTipoProd,DescrTipoProd)
Produto (CodProd,DescrProd,PrecoProd,CodTipoProd)
         CodTipoProd referencia TipoDeProduto
```

Nos capítulos 4, 5 e 6, vamos aprender como bancos de dados relacionais são organizados e projetados.

O modelo lógico descreve a estrutura do banco de dados, conforme vista pelo usuário do SGBD. Detalhes de armazenamento interno de informações, que não têm influência sobre a programação de aplicações no SGBD, mas podem afetar o desempenho da aplicações (por exemplo, as estruturas de arquivos usadas no acesso às informações) não fazem parte do modelo lógico. Estes detalhes são representados no *modelo físico*. Modelos físicos não são tratados neste livro. Eles são usados apenas por profissionais que fazem *sintonia* de banco de dados, procurando otimizar o desempenho. As linguagens e notações para o modelo físico não são padronizadas e variam de SGBD a SGBD. A tendência em produtos mais modernos é esconder o modelo físico do usuário e transferir a tarefa de otimização ao próprio SGBD.

1.2.3 modelo conceitual como modelo de organização

Quando se observa um conjunto de arquivos em computador, sejam eles gerenciados por um SGBD, sejam eles arquivos convencionais, verifica-se que usualmente um arquivo contém informações sobre um conjunto de objetos

TipoDeProduto

CodTipoProd	DescrTipoProd
1	Computador
2	Impressora

Produto

CodProd	DescrProd	PrecoProd	CodTipoProd
1	PC desktop modelo X	2.500,00	1
2	PC notebook ABC	3.500,00	1
3	Impressora jato de tinta XX	500,00	2
4	Impressora laser XX	1.500,00	2

Figura 1.4 Exemplo de tabelas de BD relacional.

ou *entidades* da organização que é atendida pelo sistema em computador. Assim, no exemplo da indústria acima citado, um sistema em computador provavelmente conteria um arquivo para armazenar dados de produtos, outro para armazenar dados de vendas, outro para armazenar dados de ordens de compra e assim por diante. Cada registro de um destes arquivos é o representante, dentro do banco de dados, de um objeto (produto, venda, ordem de compra,...) da organização cujos dados estão armazenados no banco de dados.

Desta constatação surgiu uma das idéias fundamentais do projeto de banco de dados: a de que através da identificação das entidades que terão informações representadas no banco de dados é possível identificar os arquivos que comporão o banco de dados. Devido a esta relação um-para-um entre arquivos em computador e entidades da organização modelada, observou-se que um mesmo modelo conceitual pode ser interpretado de duas formas:

- como *modelo abstrato da organização*, que define as entidades da organização que tem informações armazenadas no banco de dados, e
- como *modelo abstrato do banco de dados*, que define que arquivos (tabelas) farão parte do banco de dados.

Exemplificando, se considerarmos o modelo da Figura 1.3 podemos interpretá-lo de duas formas.

Em uma interpretação, como modelo abstrato da organização, o diagrama nos informa que na organização há produtos e tipos de produtos, que associado a cada tipo de produto há um código do tipo e uma descrição e assim por diante.

Já na outra interpretação, como modelo abstrato de um banco de dados, o diagrama nos informa que o banco de dados contém um arquivo com dados de produtos e tipos de produtos, que para cada tipo de produto são armazenados seu código e sua descrição e assim por diante.

Na prática, convencionou-se iniciar o processo de construção de um novo banco de dados com a construção de um modelo dos objetos da organização que será atendida pelo banco de dados, ao invés de partir diretamente para o projeto do banco de dados.

Esta forma de proceder permite envolver o usuário na especificação do banco de dados, pois os detalhes compreensíveis por técnicos ainda não estão sen-

do definidos, estamos apenas descrevendo objetos de uma organização. Da prática da engenharia de *software*, sabe-se que o envolvimento do usuário na especificação do *software* aumenta a qualidade do *software* produzido. A idéia é que o usuário é aquele que melhor conhece a organização e, portanto, aquele que melhor conhece os requisitos que o *software* deve preencher. Modelos conceituais são modelos que descrevem a organização e, por isso, são mais simples de compreender por usuários leigos em Informática do que modelos que envolvem detalhes de implementação.

1.2.4 projeto de banco de dados

O projeto de um novo banco de dados dá-se em três fases, descritas a seguir.

1. `modelagem conceitual` – nesta primeira fase, é construído um modelo conceitual, na forma de um diagrama entidade-relacionamento. Este modelo captura as necessidades da organização em termos de armazenamento de dados independentemente de implementação.
2. `projeto lógico` – a etapa de projeto lógico objetiva transformar o modelo conceitual obtido na primeira fase em um modelo lógico. O modelo lógico define como o banco de dados será implementado em um SGBD específico.
3. `projeto físico` – na etapa de projeto físico, o modelo do banco de dados é enriquecido com detalhes que influenciam no desempenho do banco de dados, mas não interferem em sua funcionalidade. O modelo obtido neste passo é o modelo físico do banco de dados.

   Alterações neste modelo não afetam as aplicações que usam o banco de dados, já que o modelo não envolve aspectos funcionais do banco de dados. Na prática, o projeto físico é um processo contínuo, que ocorre mesmo depois de o banco de dados já estar implementado e em funcionamento. Este processo normalmente é chamado de *sintonia* (*tuning*) de banco de dados.

   Como esta etapa é dependente do SGBD específico, ela não será abordada neste livro.

O processo acima descrito é adequado para a construção de um novo banco de dados. Caso já exista um banco de dados ou um conjunto de arquivos convencionais, e se pretenda construir um novo banco de dados, o processo acima é modificado e incorpora uma etapa de *engenharia reversa*. A engenharia reversa é explicada nos capítulos 5 e 6.

1.3 → exercícios

exercício 1 Dê um exemplo, diferente do apresentado no início do capítulo, de redundância *não controlada* de dados.

exercício 2 Dê um exemplo, diferente do apresentado no início do capítulo, de redundância *controlada* de dados. Explique quais os benefícios que a redundância controlada tem neste caso específico.

exercício 3 Enumere as principais diferenças entre o desenvolvimento de software com arquivos convencionais e o desenvolvimento de software com SGBD.

exercício 4 Descreva alguns fatores que levam alguém a preferir o uso de arquivos convencionais ao uso de SGBD. Descreva alguns fatores que levam alguém a preferir o uso de SGBD ao uso de arquivos convencionais.

exercício 5 Defina, sem retornar ao capítulo acima, os seguintes conceitos: banco de dados, sistema de gerência de banco de dados, modelo de dados, esquema de dados, modelo conceitual, modelo lógico, modelagem conceitual e projeto lógico. Verifique a definição que você fez contra a apresentada no capítulo.

exercício 6 Um técnico em Informática juntamente com um futuro usuário definem formalmente que informações deverão estar armazenadas em um banco de dados a ser construído. O resultado deste processo é um modelo conceitual, um modelo lógico ou um modelo físico?

exercício 7 Um programador recebe um documento especificando precisamente a estrutura de um banco de dados. O programador deverá construir um *software* para acessar o banco de dados através de um SGBD conforme esta estrutura. Esse documento é um modelo conceitual, um modelo lógico ou um modelo físico?

exercício 8 UML (*Unified Modeling Language*) é um conjunto de conceitos usados para modelar um *software*, que, entre outras coisas, serve para modelar bases de dados no nível conceitual. UML é uma *abordagem* de modelagem de dados ou um *modelo* de dados?

exercício 9 A definição do fator de bloco de um arquivo faz parte do modelo conceitual, do modelo lógico ou do modelo físico?

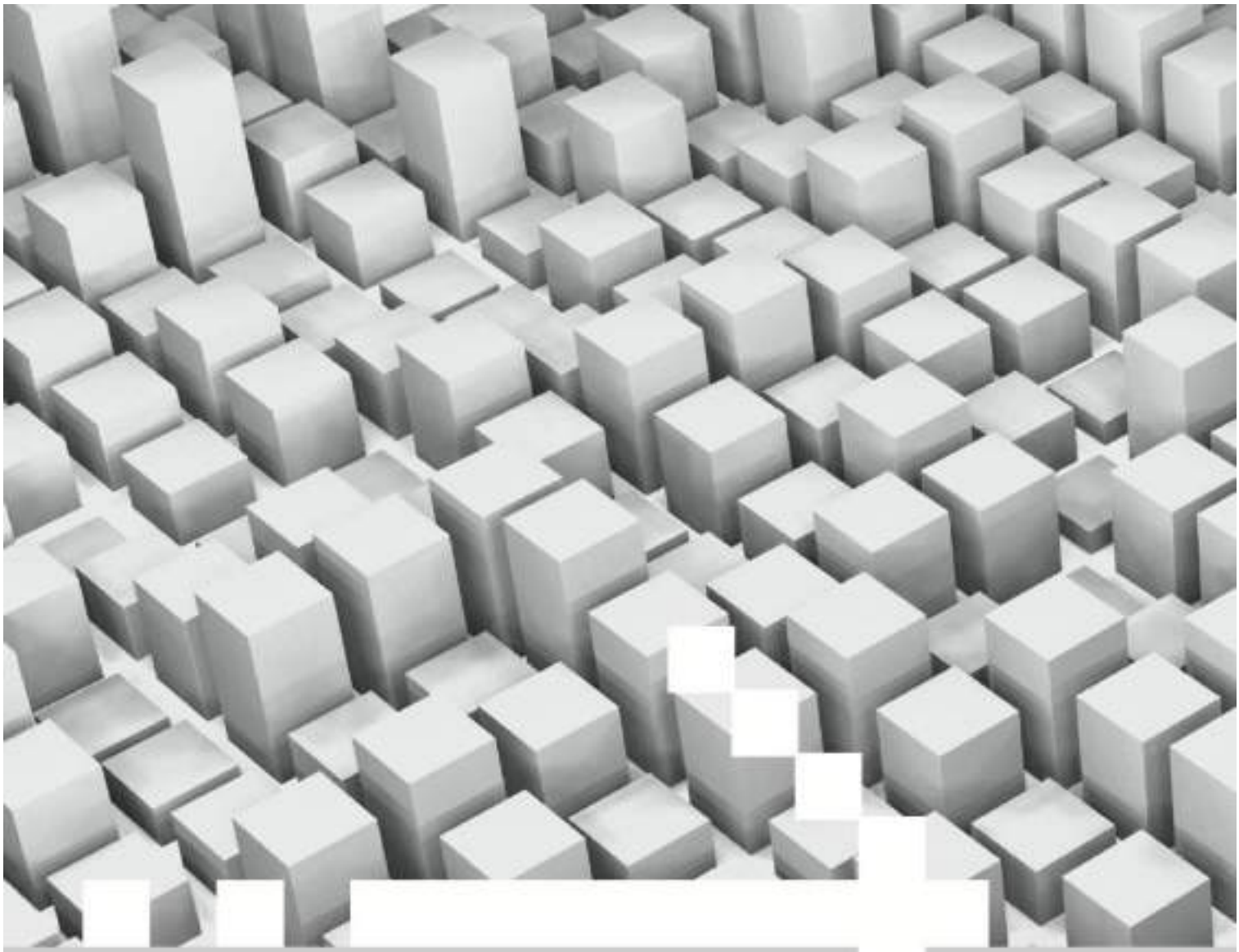
exercício 10 Dê um exemplo de aplicação de banco de dados. Defina quais seriam alguns arquivos que o banco de dados iria conter e quais os tipos de objetos da organização que neles estarão armazenados.

exercício 11 A definição do tipo de um dado (numérico, alfanumérico,...) faz parte do modelo conceitual, do modelo lógico ou do modelo físico?

exercício 12 Qual a diferença entre a redundância de dados controlada e a redundância de dados não controlada? Dê exemplos de cada uma delas.

1.4 → leituras recomendadas

Os conceitos apresentados neste capítulo aparecem em qualquer bom livro de fundamentos de banco de dados. Dois exemplos são os livros de ELMASRI; NAVATHE (2002) e de SILBERSCHATZ; KORTH; SUDARSHAN (1999), ambos com tradução para o português. Para o uso de modelos abstratos nas etapas iniciais do desenvolvimento de *software*, ver um livro introdutório à engenharia de *software*, como o de PRESSMAN (2002).



capítulo 2

abordagem entidade-relacionamento

Como vimos no capítulo 1, a primeira etapa do projeto de um banco de dados é a construção de um modelo conceitual, a chamada modelagem conceitual. O objetivo da modelagem conceitual é obter uma descrição abstrata, independente de implementação em computador, dos dados que serão armazenados no banco de dados. A técnica de modelagem mais utilizada é a abordagem entidade-relacionamento. Neste capítulo vamos apresentar os conceitos centrais da abordagem ER, acompanhados de uma notação gráfica para os respectivos diagramas.

A técnica de modelagem de dados mais difundida e utilizada é a *abordagem entidade-relacionamento(ER)*. Nesta técnica, o modelo de dados é representado através de um *modelo entidade-relacionamento (modelo ER)*. Geralmente, um modelo ER é representado graficamente através de um *diagrama entidade-relacionamento* (DER). A abordagem ER foi criada em 1976 por Peter Chen, podendo ser considerada como um padrão de fato para a modelagem conceitual. Mesmo as técnicas de modelagem orientada a objetos, que têm surgido nos últimos anos, como a UML, baseiam-se nos conceitos da abordagem ER.

Este capítulo apresenta os conceitos centrais da abordagem ER: *entidade*, *relacionamento*, *atributo*, *generalização/especialização* e *entidade associativa*. Junto com estes conceitos, é apresentada uma notação gráfica para diagramas ER. A notação gráfica usada no capítulo é a notação originalmente introduzida por Peter Chen. No capítulo 3, são discutidas as principais notações usadas na prática para construir diagramas ER.

2.1 → entidade

O conceito fundamental da abordagem ER é o conceito de *entidade*.

> entidade
> =
> conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações no banco de dados

Uma entidade[1] representa um conjunto de objetos da realidade modelada. Como o objetivo de um modelo ER é modelar de forma abstrata um BD, interessam-nos somente os objetos sobre os quais deseja-se manter informações. Vejamos alguns exemplos. No sistema de informações industrial que usamos no capítulo 1, alguns exemplos de entidades poderiam ser os produtos, os tipos de produtos, as vendas ou as compras. Já em um sistema de contas

[1] O termo "objeto" possui aqui a conotação que lhe é dada na linguagem natural, de "coisa, tudo que é perceptível ou manipulável". Não estamos falando aqui do termo "objeto" como usado na modelagem e na programação orientadas a objetos, onde o termo tem uma conotação mais precisa.

correntes, algumas entidades podem ser os clientes, as contas correntes, os cheques e as agências. Observe que uma entidade pode representar tanto objetos concretos da realidade (uma pessoa, um automóvel) quanto objetos abstratos (um departamento, um endereço[2]).

Em um DER, uma entidade é representada através de um retângulo que contém o nome da entidade. Alguns exemplos são mostrados na Figura 2.1.

Como dito acima, cada retângulo, cada entidade representa um *conjunto* de objetos sobre os quais deseja-se guardar informações. Assim, no exemplo da Figura 2.1, o primeiro retângulo designa o conjunto de todas as pessoas sobre as quais se deseja manter informações no banco de dados, enquanto o segundo retângulo designa o conjunto de todos os departamentos sobre os quais se deseja manter informações. Caso seja necessário referir um objeto particular (uma determinada pessoa ou um determinado departamento) fala-se em *ocorrência* de entidade. Mais recentemente, por influência da programação orientada a objetos, usa-se também o anglicismo *"instância"* de entidade.

Há autores que preferem usar o par de termos *conjunto de entidades* e *entidade* para designar respectivamente o conjunto de objetos e cada objeto individual. À primeira vista, esta terminologia é mais adequada, pois corresponde ao uso dos termos na linguagem natural, onde "entidade" é um indivíduo e não um coletivo. Entretanto, esta terminologia não é adequada no projeto de BD, no qual falamos com freqüência sobre conjuntos de objetos e raramente sobre indivíduos. Por esse motivo preferimos usar o par de termos *entidade* e *ocorrência de entidade*.

Da forma como está apresentado, o modelo da Figura 2.1 indica apenas quais os conjuntos de objetos sobre os quais deseja-se manter informações,

Figura 2.1 Representação gráfica de entidades.

[2] Neste ponto, aquele que já possui conhecimento sobre a modelagem ER poderá estar pensando: "Obviamente endereço é um atributo e não uma entidade!". Se você é um desses leitores, peço um pouco de paciência, pois esta questão será esclarecida mais adiante.

mas não quais as informações que devem ser mantidas para cada objeto. Estas informações são definidas pelas *propriedades* das entidades, dadas pelos *relacionamentos*, *atributos* e *generalizações/especializações*.

2.2 → relacionamento

2.2.1 conceituação

Uma das propriedades sobre as quais pode ser desejável manter informações é a *associação* entre objetos. Exemplificando, pode ser desejável saber quais pessoas estão associadas a quais departamentos em uma organização. A propriedade de entidade que especifica as associações entre objetos é o *relacionamento*.

relacionamento
=
conjunto de associações entre ocorrências de entidades

Em um DER, um relacionamento é representado através de um losango, ligado por linhas aos retângulos representativos das entidades que participam do relacionamento. A Figura 2.2 apresenta um DER contendo duas entidades, *PESSOA* e *DEPARTAMENTO*, e um relacionamento, *LOTAÇÃO*.

Este modelo expressa que o BD mantém informações sobre:

- um conjunto de objetos classificados como pessoas (entidade *PESSOA*),
- um conjunto de objetos classificados como departamentos (entidade *DEPARTAMENTO*) e
- um conjunto de associações, cada uma ligando um departamento a uma pessoa (relacionamento *LOTAÇÃO*).

Figura 2.2 Representação gráfica de relacionamento.

Da mesma forma que fizemos com entidades, quando quisermos nos referir a associações específicas dentro de um conjunto, vamos nos referir a *ocorrências* ou *instâncias* de relacionamentos. No caso do relacionamento *LOTAÇÃO*, uma ocorrência seria um par específico, formado por uma determinada ocorrência da entidade *PESSOA* e por uma determinada ocorrência da entidade *DEPARTAMENTO*.

Para fins didáticos, pode ser útil construir um *diagrama de ocorrências*, como o apresentado na Figura 2.3. Este diagrama refere-se ao modelo ER da Figura 2.2.

Em um diagrama de ocorrências, ocorrências de entidades são representadas por círculos brancos e ocorrências de relacionamentos por círculos negros. As ocorrências de entidade participantes de uma ocorrência de relacionamento são indicadas pelas linhas que ligam o círculo negro representativo da ocorrência de relacionamento aos círculos brancos representativos das ocorrências de entidades relacionadas. Assim, a Figura 2.3 representa que, entre outras, há uma ocorrência de *LOTAÇÃO* que liga a pessoa *p1* com o departamento *d1*. Observe-se que, na forma como está, o modelo da Figura 2.2 não informa quantas vezes uma entidade é associada através de um relacionamento (veremos como isso pode ser representado mais adiante). O modelo apresentado permite que uma ocorrência de entidade (por exemplo, a pessoa *p3*) não esteja associada a alguma ocorrência de entidade através

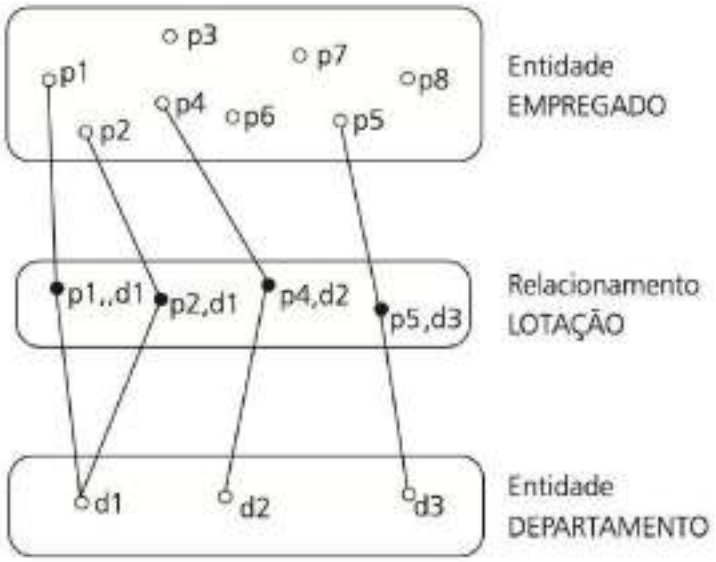


Figura 2.3 Diagrama de ocorrências.

do relacionamento, e que uma ocorrência de entidade (por exemplo, a pessoa *p1*) esteja associada a exatamente uma ocorrência de entidade através do relacionamento, ou ainda, que uma ocorrência de entidade (por exemplo, o departamento *d1*) esteja associada a mais de uma ocorrência de entidade através do relacionamento.

Não necessariamente um relacionamento associa entidades diferentes. A Figura 2.4 mostra um DER que contém um *auto-relacionamento*, isto é, um relacionamento entre ocorrências de uma mesma entidade. Neste caso, é necessário um conceito adicional, o de *papel* da entidade no relacionamento.

papel de entidade em relacionamento
=
função que uma instância da entidade cumpre dentro de uma instância do relacionamento

No caso do relacionamento de casamento, uma ocorrência de pessoa exerce o papel de marido e a outra ocorrência de pessoa exerce o papel de esposa. Papéis são anotados no DER como mostrado na Figura 2.4. No caso de relacionamentos entre entidades diferentes, como o de *LOTAÇÃO* mostrado acima, não é necessário indicar os papéis das entidades, já que eles são óbvios.

A Figura 2.5 apresenta um possível diagrama de ocorrências para o modelo ER da Figura 2.4. Os papéis (*marido e esposa*) das ocorrências de entidades

Figura 2.4 Auto-relacionamento com papéis.

em cada ocorrência de relacionamento foram anotados nas linhas que ligam os círculos representativos das ocorrências de entidade aos círculos representativos das ocorrências de relacionamentos.

2.2.2 cardinalidade de relacionamentos

Para fins de projeto de banco de dados, uma propriedade importante de um relacionamento é a de quantas ocorrências de uma entidade podem estar associadas a uma determinada ocorrência através do relacionamento. Esta propriedade é chamada de *cardinalidade* de uma entidade em um relacionamento. Há duas cardinalidades a considerar: a cardinalidade *máxima* e a cardinalidade *mínima*.

cardinalidade (mínima, máxima) de entidade em relacionamento
=
número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento

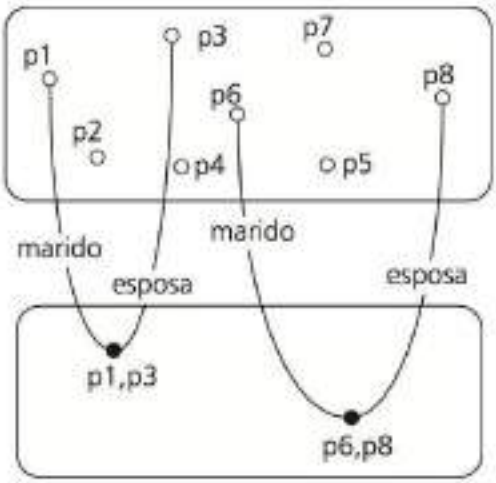


Figura 2.5 Diagrama de ocorrências para o relacionamento *CASAMENTO*.

2.2.3 cardinalidade máxima

Para exemplificar o conceito de cardinalidade, vamos retomar o exemplo da Figura 2.2. Vamos considerar as cardinalidades máximas descritas abaixo.

- Entidade *EMPREGADO* tem cardinalidade máxima **1** no relacionamento *LOTAÇÃO*.

  Isso significa que uma ocorrência de *EMPREGADO* pode estar associada a no máximo uma ocorrência de *DEPARTAMENTO* ou, em outros termos, que um empregado pode estar lotado em no máximo um departamento.
- Entidade *DEPARTAMENTO* tem cardinalidade máxima **120** no relacionamento *LOTAÇÃO*.

  Isso significa que uma ocorrência de *DEPARTAMENTO* pode estar associada a no máximo 120 ocorrências de *EMPREGADO* ou, em outros termos, que um departamento pode ter nele lotado no máximo 120 empregados.

Para o projeto de banco de dados, especialmente de bancos de dados relacionais, não é necessário distinguir entre diferentes cardinalidades máximas maiores que um. Por este motivo, apenas duas cardinalidades máximas são geralmente consideradas:

- a cardinalidade máxima um (**1**) e
- a cardinalidade máxima ilimitada, usualmente chamada de cardinalidade máxima "muitos" e referida pela letra **n.**

Assim, no exemplo acima, diz-se que a cardinalidade máxima da entidade *DEPARTAMENTO* no relacionamento *LOTAÇÃO* é **n**.

Em um DER, a cardinalidade máxima é representada conforme indicado na Figura 2.6. Observe a convenção usada. À primeira vista, ela pode parecer pouco natural, já que a cardinalidade vai anotada "do outro lado" do relacionamento ao qual se refere. Exemplificando, a cardinalidade máxima da entidade *EMPREGADO* no relacionamento *LOTAÇÃO* é anotada junto ao símbolo da entidade *DEPARTAMENTO*.

■ classificação de relacionamentos binários

A cardinalidade máxima pode ser usada para classificar relacionamentos binários. Um *relacionamento binário* é aquele cujas ocorrências contêm duas

Figura 2.6 Representação gráfica de cardinalidade máxima.

ocorrências de entidade, como todos os vistos até aqui. Podemos classificar os relacionamentos binários em **n:n**, **1:n** e **1:1**. As figuras 2.7, 2.8 e 2.9 apresentam exemplos de relacionamentos com cardinalidades máximas **1:1**, **1:n** e **n:n**, respectivamente. A seguir comentamos a interpretação de alguns relacionamentos apresentados nestas figuras.

Na Figura 2.7, no relacionamento *CASAMENTO*, as cardinalidades máximas expressam que uma pessoa pode possuir no máximo um marido e que uma

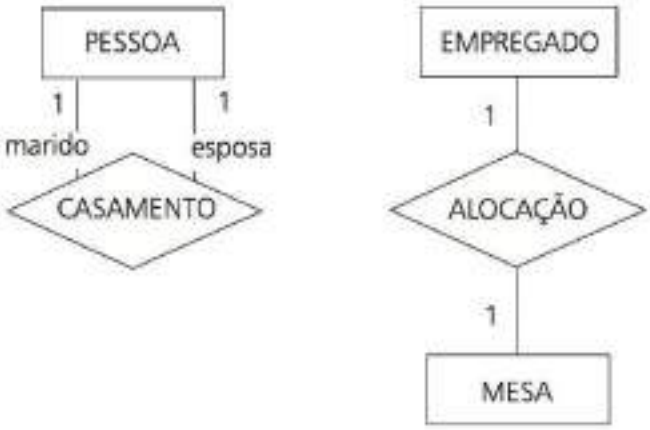


Figura 2.7 Relacionamentos **1:1**.

Figura 2.8 Relacionamentos **1:n**.

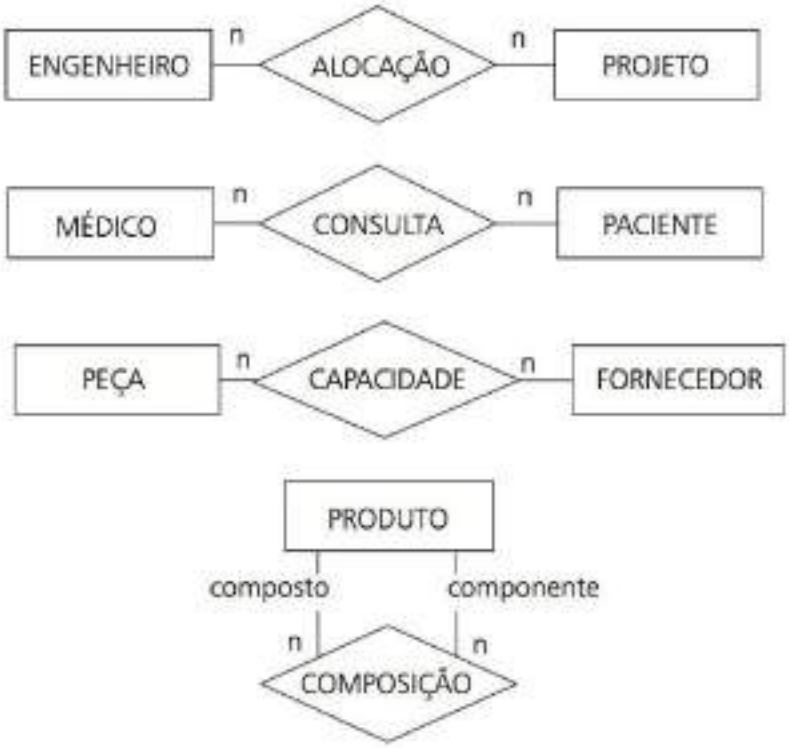


Figura 2.9 Relacionamentos **n:n**.

pessoa pode possuir no máximo uma esposa. Mais precisamente, as cardinalidades expressam que uma instância de pessoa pode estar associada via relacionamento a no máximo uma instância de pessoa no papel de esposa e vice-versa, uma instância de pessoa pode estar associada via relacionamento a no máximo uma instância de pessoa no papel de marido.

Observe que o relacionamento *CASAMENTO* (Figura 2.7) é também um relacionamento binário, apesar de envolver apenas uma entidade. O que determina o fato de o relacionamento ser binário é o número de ocorrências de entidade que participam de cada ocorrência do relacionamento. De cada ocorrência de *CASAMENTO* participam exatamente duas ocorrências da entidade *PESSOA* (um marido e uma esposa).

A Figura 2.8 mostra outros exemplos de relacionamentos **1:n**, além do relacionamento *LOTAÇÃO* que já havia sido visto acima. O relacionamento *INSCRIÇÃO* modela a inscrição de alunos em uma universidade pública, onde existe a restrição de um aluno estar inscrito em no máximo um curso.

O relacionamento entre as entidades *EMPREGADO* e *DEPENDENTE* (Figura 2.8) modela a associação entre um empregado e seus dependentes para fins de imposto de renda. Neste caso, um dependente pode estar associado a no máximo um empregado. Cabe observar que, no DER, não foi anotado o nome do relacionamento. No caso de no DER não constar o nome do relacionamento, este é denominado pela concatenação de nomes das entidades participantes. Assim, neste caso, o relacionamento é denominado *EMPREGADO-DEPENDENTE*.

O relacionamento *SUPERVISÃO* (Figura 2.8) é um exemplo de auto-relacionamento **1:n**. Ele modela a associação entre um empregado (supervisor) e seus supervisionados imediatos. A cardinalidade máxima expressa que um empregado pode possuir no máximo um supervisor, mas muitos supervisionados.

O tipo menos restrito de relacionamento é o de cardinalidade **n:n**. A Figura 2.9 apresenta alguns relacionamentos deste tipo, inclusive um auto-relacionamento.

2.2.4 relacionamento ternário

Todos os exemplos até aqui mostrados são de relacionamentos binários. A abordagem ER permite que sejam definidos relacionamentos de grau maior do que dois (relacionamentos ternários, quaternários,...). O DER da Figura 2.10 mostra um exemplo de um relacionamento ternário.

Cada ocorrência do relacionamento *DISTRIBUIÇÃO* associa três ocorrências de entidade: um produto a ser distribuído, uma cidade na qual é feita a distribuição e um distribuidor.

No caso de relacionamentos de grau maior que dois, o conceito de cardinalidade de relacionamento é uma extensão não trivial do conceito de cardinalidade em relacionamentos binários. Lembre-se de que, em um relacionamento binário R entre duas entidades A e B, a cardinalidade máxima de A em R indica quantas ocorrências de B podem estar associadas a cada ocorrência de A. No caso de um relacionamento ternário, a cardinalidade refere-se a *pares de entidades*. Em um relacionamento R entre três entidades A, B e C, a cardinalidade máxima de A e B dentro de R indica quantas ocorrências de C podem estar associadas a um par de ocorrências de A e B.

Exemplificando, na Figura 2.11, o **1** na linha que liga o retângulo representativo da entidade *DISTRIBUIDOR* ao losango representativo do relacionamento expressa que cada par de ocorrências (cidade, produto) está associado a no máximo um distribuidor. Significa que é concedida exclusividade de distribuição de um produto para um distribuidor em uma cidade.

Figura 2.10 Relacionamento ternário.

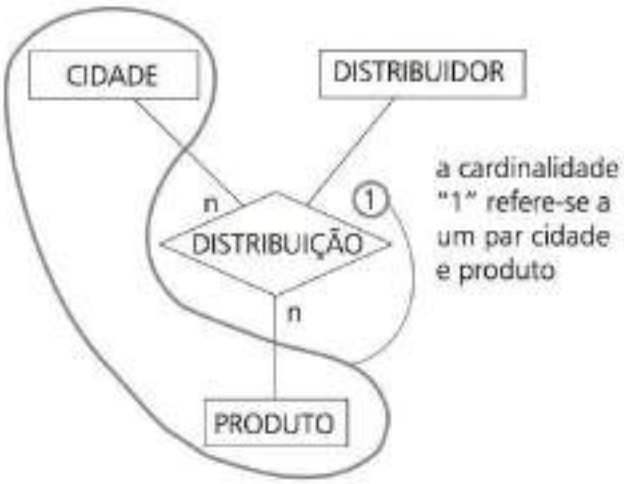


Figura 2.11 Cardinalidade em relacionamentos ternários.

Já os dois "n" expressam que:

- A um par (cidade, distribuidor) podem estar associados muitos produtos ou, em outros termos, um distribuidor pode distribuir em uma cidade muitos produtos.
- A um par (produto, distribuidor) podem estar associadas muitas cidades ou, em outros termos, um distribuidor pode distribuir um produto em muitas cidades.

2.2.5 cardinalidade mínima

Além da cardinalidade máxima, outra informação que pode ser representada por um modelo ER é o número *mínimo* de ocorrências de entidade associadas a uma ocorrência de uma entidade através de um relacionamento. Para fins de projeto de BD, consideram-se apenas duas cardinalidades mínimas: a cardinalidade mínima 0 e a cardinalidade mínima 1.

A cardinalidade mínima 1 também recebe a denominação de "associação *obrigatória*", já que ela indica que o relacionamento deve obrigatoriamente associar uma ocorrência de entidade a cada ocorrência da entidade em questão. Com base na mesma linha de raciocínio, a cardinalidade mínima 0 recebe a denominação "associação *opcional*".

A cardinalidade mínima é anotada no diagrama junto à cardinalidade máxima, conforme mostrado na Figura 2.12. Nesta figura, aparece novamente o

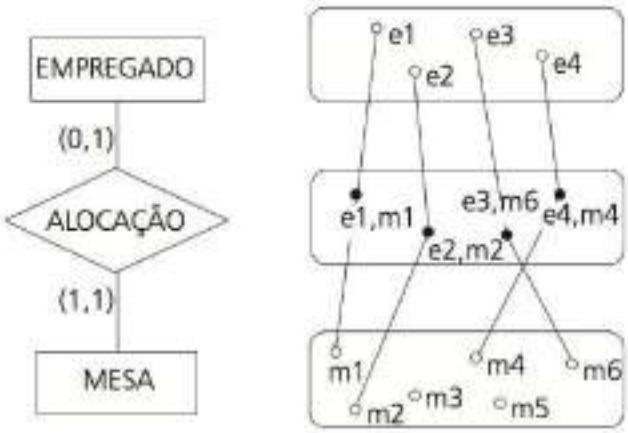


Figura 2.12 Cardinalidade mínima de relacionamento.

exemplo da alocação de empregados a mesas. Aqui, a cardinalidade mínima é usada para especificar que cada empregado deve ter a ele alocada obrigatoriamente uma mesa (cardinalidade mínima 1) e que uma mesa pode existir sem que a ela esteja alocado um empregado (cardinalidade mínima 0).

2.3 → exemplo de uso de entidades e relacionamentos

A Figura 2.13 apresenta um exemplo de um modelo ER mais abrangente que os anteriores, envolvendo diversas entidades e relacionamentos. Como se vê, um diagrama ER é apresentado na forma de um grafo. A distribuição dos símbolos de DER no papel é totalmente arbitrária e não tem maior significado do ponto de vista formal. Entretanto, para tornar o diagrama mais legível é comum evitar cruzamentos de linhas. Para isso, a recomendação geral é a de posicionar os retângulos representativos de entidades que participam de muitos relacionamentos no centro do diagrama.

O modelo da Figura 2.13 é uma parte do modelo de dados de um sistema de controle acadêmico de uma universidade fictícia. O modelo descreve o seguinte:

- Deseja-se manter informações sobre alunos, cursos, disciplinas e departamentos.
- Além disso, deseja-se manter informações sobre a associação de alunos a cursos, de disciplinas a cursos, de disciplinas a departamentos, bem como de disciplinas a suas disciplinas pré-requisito.

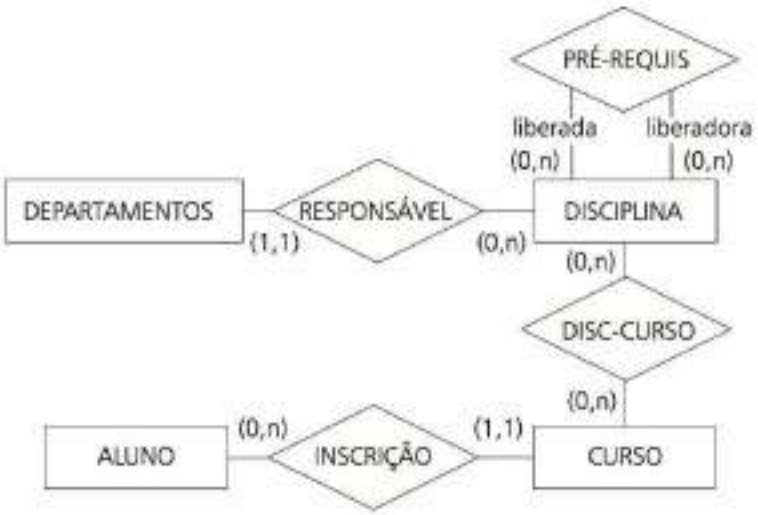


Figura 2.13 DER para o controle acadêmico de uma universidade.

- Através das cardinalidades expressa-se que:
  - Cada disciplina possui exatamente um departamento responsável, e um departamento é responsável por muitas disciplinas, inclusive por nenhuma. Note-se que, apesar de sabermos que os departamentos em uma universidade existem para ser responsáveis por disciplinas, especificamos a cardinalidade mínima de *DEPARTAMENTO* em *RESPONSÁVEL* como sendo **0**. Com isto admitimos a possibilidade de existirem departamentos vazios. Esta cardinalidade foi especificada considerando o estado do banco de dados imediatamente após a criação de um novo departamento, bem como o estado imediatamente após a eliminação da última disciplina de um departamento. Da forma como a restrição foi especificada, é possível incluir o departamento em uma transação para, depois, em transações subseqüentes, vinculá-lo às disciplinas sob sua responsabilidade. Se tivesse sido especificada a cardinalidade mínima **1**, ao menos uma disciplina teria que ser vinculada ao departamento já na própria transação de inclusão do departamento. Como observa-se da discussão acima, para especificar as cardinalidades mínimas é necessário possuir conhecimento sobre a ordem de execução das transações de inclusão e exclusão das entidades.
  - Uma disciplina pode possuir diversos pré-requisitos, inclusive nenhum. Uma disciplina pode ser pré-requisito de muitas outras disciplinas, inclusive de nenhuma.

- Uma disciplina pode aparecer no currículo de muitos cursos (inclusive de nenhum) e um curso pode possuir muitas disciplinas em seu currículo (inclusive nenhuma).
- Um aluno está inscrito em exatamente um curso e um curso pode ter nele inscritos muitos alunos (inclusive nenhum).

2.4 → atributo

Conforme mencionado no início da seção precedente, o modelo ER permite a especificação de propriedades de entidades. Uma propriedade é participar de um relacionamento. Outra propriedade é ter um atributo. O conceito de atributo serve para associar informações a ocorrências de entidades ou de relacionamentos.

atributo
=
dado que é associado a cada ocorrência de uma entidade
ou de um relacionamento

Atributos são representados graficamente conforme mostra a Figura 2.14. A figura expressa que cada ocorrência de *PROJETO* tem associado exatamente um nome, um código e um tipo.

Na prática, muitas vezes os atributos não são representados graficamente, para não sobrecarregar os diagramas, já que entidades podem possuir um grande número de atributos. Prefere-se usar uma representação textual que aparece separadamente do diagrama ER. Ao final deste capítulo, é fornecida uma possível sintaxe para uma representação textual dos atributos. No caso

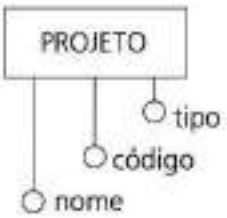


Figura 2.14 Atributos de uma entidade.

de ser usado um *software* para a construção de modelos ER, o próprio *software* encarrega-se do armazenamento da lista de atributos de cada entidade em um *dicionário de dados*.

O conjunto de valores que um determinado atributo pode assumir é chamado de *domínio do atributo*. Há várias linguagens para definir domínios de atributos, normalmente específicas de uma ferramenta CASE ou SGBD. Nestas definições são combinadas várias propriedades dos valores, como tamanho em caracteres, tipo de valores (número, data,...), enumeração de valores permitidos, etc. Raramente domínios de valores aparecem no diagrama ER. Como não há padrões para a especificação de domínios, ela não será discutida neste livro.

Um atributo pode possuir uma cardinalidade, de maneira análoga a uma entidade em um relacionamento. A cardinalidade de um atributo define quantos valores deste atributo podem estar associados a uma ocorrência da entidade/relacionamento a qual ele pertence. A representação diagramática da cardinalidade de atributos é derivada da representação da cardinalidade de entidades em relacionamentos, conforme mostra a Figura 2.15.

No caso de a cardinalidade ser **(1,1)** ela pode ser omitida do diagrama. Assim, o exemplo da Figura 2.15 expressa que nome e código são atributos *obrigatórios* (cardinalidade mínima **1** – cada entidade possui no mínimo um valor associado) e *monovalorados* (cardinalidade máxima **1** – cada entidade possui no máximo um valor associado). Já o atributo telefone, é um atributo *opcional* (cardinalidade mínima **0**) e *multivalorado* (cardinalidade máxima **n**).

Assim como entidades, também relacionamentos podem possuir atributos. A Figura 2.16 mostra um DER no qual um relacionamento, *ATUAÇÃO*, possui um atributo, a função que um engenheiro exerce dentro de um projeto.

Figura 2.15 Cardinalidade de atributos.

Figura 2.16 Atributo de relacionamento n:n.

Esta função não pode ser considerada atributo de *ENGENHEIRO*, já que um engenheiro pode atuar em diversos projetos, exercendo diferentes funções. Também, não é atributo de *PROJETO*, já que, em um projeto, podem atuar diversos engenheiros com funções diferentes.

Outro exemplo de atributo em relacionamento, agora em um relacionamento 1:n, é mostrado na Figura 2.17. Este diagrama modela vendas em uma organização comercial. Algumas vendas são à vista, outras a prazo. Vendas a prazo são relacionadas a uma financeira, através do relacionamento *FINANCIAMENTO*. Os atributos nº de parcelas e taxa de juros são atributos do relacionamento. Estes dois atributos poderiam ter sido incluídos na entidade *VENDA*. Neste caso, seriam atributos opcionais, já que nem toda venda é a prazo e possui estes atributos. Assim, preferiu-se usar o modelo da figura, exatamente para explicitar o fato de os atributos nº de parcelas e taxa de juros pertencerem somente a vendas a prazo.

2.4.1 identificando entidades

Cada entidade deve possuir um *identificador*.

> identificador de entidade
> =
> conjunto de um ou mais atributos e relacionamentos cujos valores servem para distinguir uma ocorrência da entidade das demais ocorrências da mesma entidade

O caso mais simples é o da entidade que possui um único atributo como identificador. No DER, atributos identificadores são representados por um círculo preto. No exemplo da Figura 2.18, o atributo *código* é identificador. Isso significa que cada pessoa possui um código diferente. Já os atributos *nome* e

Figura 2.17 Atributo de relacionamento 1:n.

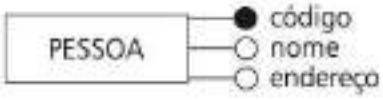


Figura 2.18 Identificador simples.

endereço não são identificadores – o mesmo nome (ou o mesmo endereço) pode ser associado a pessoas diferentes.

A Figura 2.19 mostra um exemplo no qual o identificador da entidade é composto por diversos atributos de vários atributos. Considera-se um almoxarifado de uma empresa de ferragens organizado como segue. Os produtos ficam armazenados em prateleiras. Estas prateleiras encontram-se em armários organizados em corredores. Os corredores são numerados seqüencialmente a partir de um e as prateleiras são numeradas seqüencialmente a partir de um, dentro de um corredor. Assim, para identificar uma prateleira é necessário conhecer seu número e o número do corredor em que se encontra. Para cada prateleira deseja-se saber sua capacidade em metros cúbicos.

Finalmente, há casos em que o identificador de uma entidade é composto não somente por atributos da própria entidade, mas também por relacionamentos dos quais a entidade participa (*relacionamento identificador*). Um exemplo deste caso é mostrado na Figura 2.20. Este modelo envolve empregados de

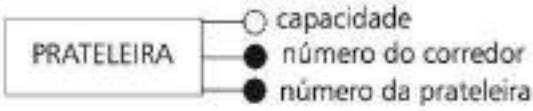


Figura 2.19 Identificador composto.

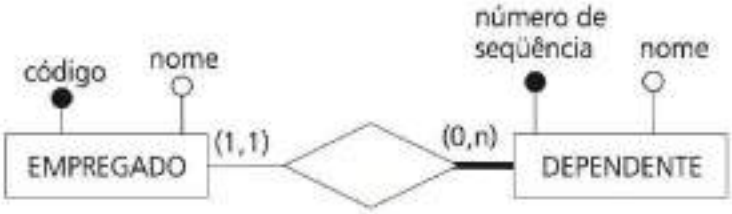


Figura 2.20 Relacionamento identificador.

uma organização, relacionados com os seus dependentes para fins de imposto de renda. Cada dependente está relacionado a exatamente um empregado. Um dependente é identificado pelo empregado ao qual ele está relacionado e por um número de seqüência que distingue os diferentes dependentes de um mesmo empregado. No DER, o relacionamento usado como identificador é indicado por uma linha mais densa, conforme mostra a Figura 2.20.

Nesse caso, alguns autores dizem que a entidade *DEPENDENTE* é uma entidade *fraca*. O termo "fraca" deriva do fato de a entidade somente existir quando relacionada a outra entidade e de usar, como parte de seu identificador, entidades relacionadas. Entretanto, os autores de livros mais recentes preferem não utilizar o conceito, já que ser "fraca" não é uma propriedade de uma entidade, mas sim de uma entidade em um relacionamento, visto que uma entidade pode ser "fraca" em um relacionamento e "forte" em outro.

Outro exemplo de relacionamento identificador é mostrado na Figura 2.21, que contém um fragmento de um DER sobre empresas. No exemplo, é representada a divisão de grupos de empresas em empresas e de empresas em filiais de empresas. Para identificar um grupo de empresas é usado um código. Já uma empresa é identificada pelo grupo ao qual está relacionada e por um número da empresa dentro do grupo. Finalmente, uma filial é identificada pela empresa a qual está vinculada e por um número de filial dentro da empresa.

O identificador de uma entidade, seja ele simples, composto por diversos atributos, ou composto por identificadores externos, deve obedecer a duas propriedades:

- O identificador deve ser *mínimo*. Isso significa que o identificador de uma entidade deve ser composto de tal forma que, retirando um dos atributos ou relacionamentos que o compõe, ele deixa de ser identificador.

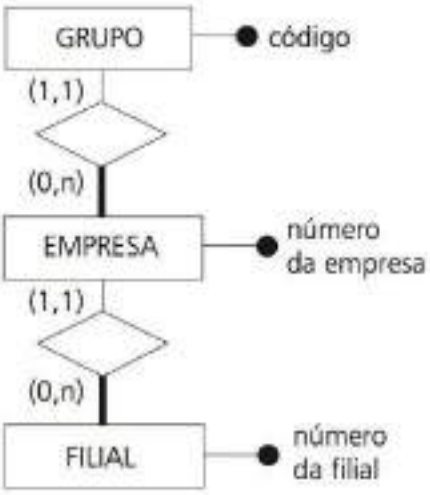


Figura 2.21 Entidades com relacionamentos identificadores.

Exemplificando, na entidade *PESSOA* na Figura 2.18, o par código e nome poderia ser usado para distinguir uma ocorrência de *PESSOA* das demais. Entretanto, estes atributos não formam um identificador mínimo, já que código é suficiente para distinguir as ocorrências de *PESSOA*.

- Para fins de projeto de BD relacional, cada entidade deve possuir um *único* identificador. Em alguns casos, diferentes conjuntos de atributos podem servir para distinguir as ocorrências da entidade. Exemplificando, a entidade *EMPREGADO* da Figura 2.22 poderia possuir como identificador tanto o atributo código, quanto o atributo CPF (identificador único do contribuinte junto à Receita Federal). Cabe ao modelador decidir qual dos dois atributos será usado como identificador da entidade. Alguns critérios para esta decisão aparecem mais adiante, no capítulo relativo ao projeto de BD a partir do modelo ER.

2.4.2 identificando relacionamentos

Em princípio, uma ocorrência de relacionamento diferencia-se das demais ocorrências do mesmo relacionamento pelas ocorrências de entidades que

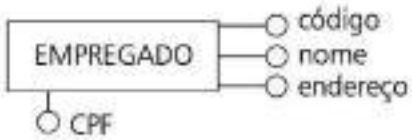


Figura 2.22 Identificadores alternativos.

dela participam. Exemplificando, uma ocorrência de *ALOCAÇÃO* (Figura 2.23) é identificada pela ocorrência de *ENGENHEIRO* e pela ocorrência de *PROJETO* que ela relaciona. Em outros termos, para cada par (engenheiro, projeto) há no máximo um relacionamento de alocação.

Entretanto, há casos nos quais entre as mesmas ocorrências de entidade podem existir diversas ocorrências de relacionamento. Um exemplo é o relacionamento *CONSULTA* entre entidades *MÉDICO* e *PACIENTE* (Figura 2.24). Para um determinado médico e um determinado paciente pode haver diversas consultas. Neste caso, é necessário algo que distinga uma consulta entre um médico e seu paciente das demais consultas entre este médico e este paciente. A diferenciação ocorre através de *atributos identificadores de relacionamento*. No caso do relacionamento *CONSULTA* (Figura 2.24) o atributo identificador do relacionamento pode ser data/hora.

Assim, um relacionamento é identificado pelas entidades dele participantes, bem como pelos atributos identificadores eventualmente definidos.

2.5 → generalização/especialização

Além de relacionamentos e atributos, propriedades podem ser atribuídas a entidades através do conceito de *generalização/especialização*. A partir deste conceito é possível atribuir propriedades particulares a um subconjunto das ocorrências (*especializadas*) de uma entidade *genérica*. No DER, o sím-

Figura 2.23 Relacionamento identificado por suas entidades.

Figura 2.24 Relacionamento identificado por suas entidades.

bolo para representar generalização/especialização é um triângulo isósceles, conforme mostra a Figura 2.25. A generalização/especialização mostrada nesta figura expressa que a entidade *CLIENTE* é dividida em dois subconjuntos, as entidades *PESSOA FÍSICA* e *PESSOA JURÍDICA*, cada uma com propriedades próprias.

Associada ao conceito de generalização/especialização está a idéia de *herança de propriedades*. Herdar propriedades significa que cada ocorrência da entidade especializada possui, além de suas próprias propriedades (atributos, relacionamentos e generalizações/especializações), também as propriedades da ocorrência da entidade genérica correspondente. Assim, segundo o DER da Figura 2.25, a entidade *PESSOA FÍSICA* possui, além de seus atributos particulares, *CPF* e *sexo*, também todas as propriedades da ocorrência da entidade *CLIENTE* correspondente, ou seja, os atributos *nome* e *código*, o seu identificador (atributo código), bem como o relacionamento com a entidade *FILIAL*. Resumindo, o diagrama expressa que toda pessoa física tem como atributos *nome*, *código*, *CPF* e *sexo*, é identificada pelo código e está obrigatoriamente relacionada a exatamente uma filial. Da mesma maneira, toda pessoa jurídica tem como atributos *nome*, *código*, *CNPJ* e *tipo de organização*, é identificada pelo *código* e está obrigatoriamente relacionada a exatamente uma filial.

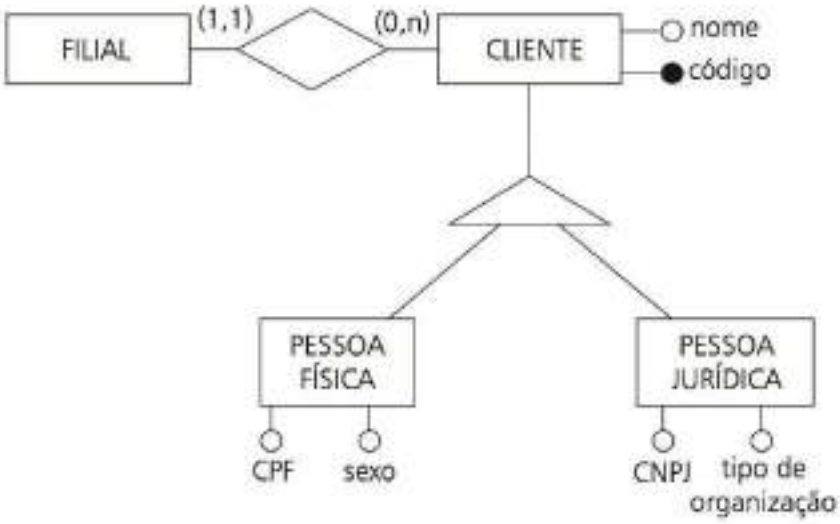


Figura 2.25 Generalização/especialização.

generalização/especialização total ou parcial

A generalização/especialização pode ser classificada em dois tipos, *total* ou *parcial*, de acordo com a obrigatoriedade ou não de a cada ocorrência da entidade genérica corresponder uma ocorrência da entidade especializada.

Em uma generalização/especialização *total* para cada ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas. Esse é o caso do exemplo da Figura 2.25, no qual a toda ocorrência da entidade *CLIENTE* corresponde uma ocorrência em uma das duas especializações. Esse tipo de generalização/especialização é simbolizado por um "t", conforme mostrado na Figura 2.26.

Em uma generalização/especialização *parcial*, nem toda ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada. Esse é o caso do exemplo da Figura 2.27, no qual nem toda entidade *FUNCIONÁRIO* possui uma entidade correspondente em uma das duas especializações (nem todo o funcionário é motorista ou secretária). Este tipo de generalização/especialização é simbolizado por um "p" conforme mostrado na figura. Usualmente, quando há uma especialização parcial, na entidade genérica (no exemplo, em *FUNCIONÁRIO*) aparece um atributo que identifica o tipo de ocorrência da entidade genérica (no exemplo, trata-se do atributo *tipo de funcionário*). Este atributo não é necessário no caso de especializações totais, já que a presença da ocorrência correspondente à entidade genérica em uma de suas especializações é suficiente para identificar o tipo da entidade.

Figura 2.26 Generalização/especialização total.

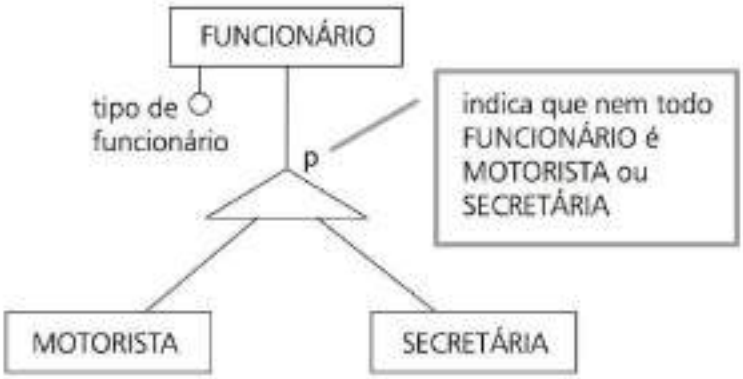


Figura 2.27 Generalização/especialização parcial.

■ generalização/especialização exclusiva ou compartilhada

Além da classificação em total e parcial, uma generalização/especialização também pode ser classificada em *compartilhada* e *exclusiva*.

Generalização/especialização *exclusiva* significa que, em uma hierarquia de generalização/especialização, uma ocorrência de entidade genérica é especializada no máximo uma vez, nas folhas da árvore de generalização/especialização. Este é o caso dos exemplos de generalização/especialização mostrados até aqui. Exemplificando, no modelo da Figura 2.27, uma instância de *FUNCIONÁRIO* aparece uma vez somente nas entidades especializadas (*MOTORISTA* ou *SECRETÁRIA*), já que um funcionário ou é motorista ou é secretária, mas não ambas as funções ao mesmo tempo. Este tipo de generalização/especialização é chamada de "exclusiva" pela exclusão mútua da aparição de uma ocorrência de *FUNCIONÁRIO* nas entidades *MOTORISTA* ou *SECRETÁRIA*.

Já a generalização/especialização *compartilhada* indica que, em uma hierarquia de generalização/especialização, uma ocorrência de entidade genérica pode aparecer em várias entidades nas folhas da árvore de generalização/especialização.

Um exemplo é mostrado na Figura 2.28. Neste diagrama, considera-se o conjunto de pessoas vinculadas a uma universidade. Neste caso, a especialização é compartilhada, já que a mesma pessoa pode aparecer em múltiplas especializações. Uma pessoa pode ser professor e aluno ou funcionário e aluno ao mesmo tempo. O fato de tratar-se de uma generalização/especialização

Figura 2.28 Generalização/especialização compartilhada.

compartilhada é indicado, no DER, pela letra "c" junto ao triângulo representativo da generalização/especialização, sendo a generalização/especialização exclusiva indicada pela letra "x" junto ao triângulo.

As combinações de tipos de entidades e as letras usadas para identificá-las no DER estão apresentadas na Tabela 2.1.

■ níveis de generalização/especialização

Uma entidade pode ser especializada em qualquer número de entidades, inclusive em uma única. Exemplificando, se no exemplo da Figura 2.27 apenas os motoristas possuíssem propriedades particulares, haveria apenas uma entidade especializada, a de motoristas.

Além disso, não há limite no número de níveis hierárquicos da generalização/especialização. Uma entidade especializada em uma generalização/especialização pode, por sua vez, ser entidade genérica em uma outra generalização/especialização. É admissível, inclusive, que uma mesma entidade seja uma especialização de diversas entidades genéricas (a chamada *herança múltipla*). A Figura 2.29 apresenta um DER em que aparecem múltiplos níveis de generalização/especialização, bem como o conceito de

Tabela 2.1 Tipos de generalizações/especializações

	Total (t)	Parcial (p)
Exclusiva (x)	xt	xp
Compartilhada (c)	ct	cp

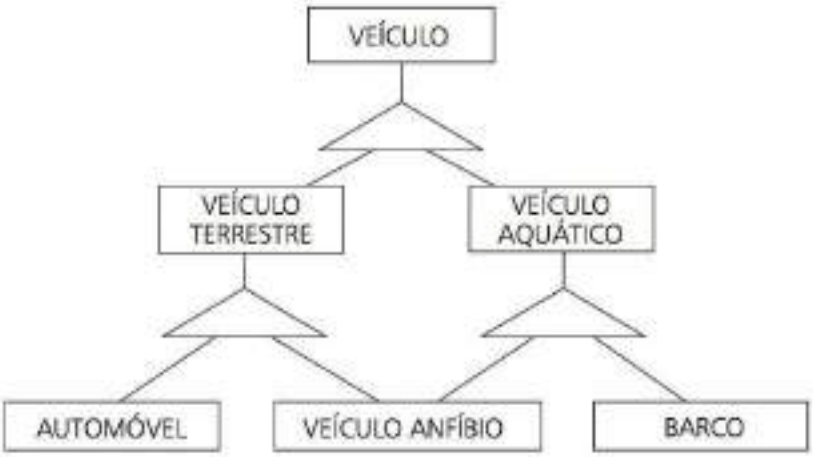


Figura 2.29 Generalização/especialização em múltiplos níveis e com herança múltipla.

herança múltipla. Exemplificando, uma entidade *BARCO* é uma especialização de um *VEÍCULO AQUÁTICO* que, por sua vez, é uma especialização de *VEÍCULO*. Assim, além de suas propriedades específicas, um barco tem também as propriedades de um veículo aquático, bem como as propriedades de um veículo em geral. O exemplo de herança múltipla aparece na entidade *VEÍCULO ANFÍBIO*. Um veículo anfíbio possui, além de suas propriedades específicas, tanto as propriedades de um veículo aquático, quanto as propriedades de um veículo terrestre, já que é especialização destas duas últimas entidades.

Observe que cada entidade especializada herda o identificador de sua entidade genérica. Portanto, não faz sentido definir identificador para entidades especializadas. Além disso, somente pode haver uma entidade genérica em cada hierarquia de generalização/especialização. A Figura 2.30 mostra uma hierarquia proibida pela restrição acima.

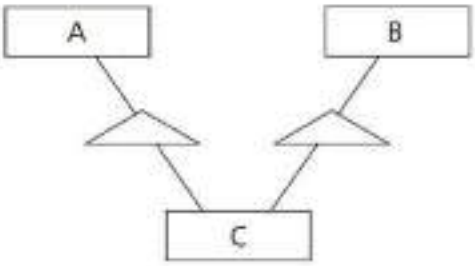


Figura 2.30 Hierarquia proibida: herança de vários identificadores.

A entidade C estaria herdando tanto o identificador da entidade B quanto o identificador da entidade B, o que não faz sentido.

2.6 → entidade associativa

Um relacionamento é uma associação entre entidades. Na modelagem ER não foi prevista a possibilidade de associar uma entidade com um relacionamento ou então de associar dois relacionamentos entre si. Na prática, quando estamos construindo um novo modelo ER ou modificando um modelo ER existente, surgem situações em que é desejável permitir a associação de uma entidade a um relacionamento. A título de exemplo, considere-se o modelo da Figura 2.31.

Suponha que seja necessário modificar este modelo da seguinte forma. É necessário saber que medicamentos existem e que medicamentos foram prescritos em cada consulta. Para saber que medicamentos existem, cria-se uma nova entidade, *MEDICAMENTO*. A questão agora é: com que entidade existente deve estar relacionada a nova entidade? Se *MEDICAMENTO* fosse relacionado a *MÉDICO*, ter-se-ia apenas a informação de que médico prescreveu que medicamentos, faltando a informação do paciente que os teve prescritos. Por outro lado, se *MEDICAMENTO* fosse relacionado à *PACIENTE*, faltaria a informação do médico que prescreveu o medicamento. Assim, deseja-se relacionar o medicamento à consulta, ou seja, deseja-se relacionar uma entidade (*MEDICAMENTO*) a um relacionamento (*CONSULTA*), o que não está previsto na abordagem ER. Para tal, foi criado um conceito especial, o de *entidade associativa*. Uma entidade associativa nada mais é que a redefinição de um relacionamento, que passa a ser tratado como se fosse também uma entidade. Graficamente, isso é feito como mostrado na Figura 2.32. O retângulo desenhado ao redor do relacionamento *CONSULTA* indica que este relacionamento passa a ser visto como uma entidade (associativa, já que é baseada em um relacionamento). Sendo *CONSULTA* também uma entidade, é possível associá-la através de relacionamentos a outras entidades, conforme mostra a figura.

Figura 2.31 DER a ser modificado.

Figura 2.32 Entidade associativa.

Caso não se desejasse usar o conceito de entidade associativa, seria necessário transformar o relacionamento *CONSULTA* em uma entidade, que então poderia ser relacionada a *MEDICAMENTO*, conforme mostrado na Figura 2.33. No modelo da figura, o relacionamento foi substituído por uma entidade homônima, junto com dois relacionamentos (parte representada em linhas densas). Para manter a equivalência com o modelo anterior (Figura 2.32), uma consulta está relacionada com exatamente um médico e exatamente um paciente (a cardinalidade mínima e máxima é um). Uma consulta é identificada pelo paciente e pelo médico a ela ligados. Tendo substituído o relacionamento *CONSULTA* pela entidade, basta relacionar a entidade *CONSULTA* com a entidade *MEDICAMENTO*.

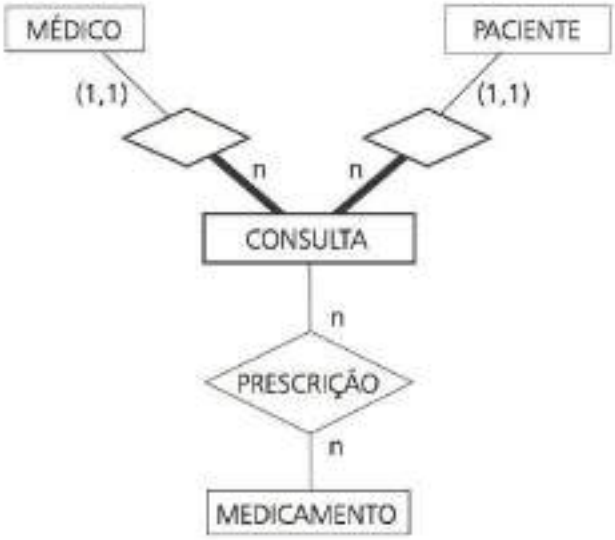


Figura 2.33 Substituindo relacionamento por entidade.

Observe-se que o diagrama da Figura 2.33 é equivalente ao diagrama da Figura 2.32. Equivalente aqui significa que ambos geram o mesmo banco de dados relacional. Uma discussão sobre a eqüivalência de modelos ER encontra-se na Seção Diferentes modelos podem ser equivalentes, na pág. 73.

2.7 → esquemas gráficos e textuais de modelos ER

Uma representação de um modelo é chamada de *esquema do banco de dados*. Até este ponto do livro, os esquemas de banco de dados sempre foram diagramas ER, isto é, sempre estão apresentados na forma gráfica. A Figura 2.34 resume os símbolos usados neste livro para a representação gráfica de esquemas ER.

Um esquema ER pode ter uma representação textual. Na Figura 2.35, aparece a sintaxe de uma linguagem textual para definição de esquemas ER. A sintaxe é dada na forma de uma gramática BNF. Nesta sintaxe, são usadas

Conceito	Símbolo
Entidade	
Relacionamento	
Atributo	
Atributo identificador	
Relacionamento identificador	(1,1)
Generalização/especialização	
Entidade associativa	

Figura 2.34 Símbolos usados na construção de esquemas ER.

```
ESQUEMA → Esquema: ESQUEMA_NOME
          SEÇÃO_ENTIDADE
          SEÇÃO_GENERALIZAÇÃO
          SEÇÃO_ENT_ASSOCIATIVA
          SEÇÃO_RELACIONAMENTO

SEÇÃO_ENTIDADE → {DECL_ENT}

DECL_ENT → Entidade: ENTIDADE_NOME
           [SEÇÃO_ATRIBUTO]
           [SEÇÃO_IDENTIFICADOR]

SEÇÃO_ATRIBUTO → Atributos: {DECL_ATRIB}
DECL_ATRIB → [(MIN_CARD,MAX_CARD)] ATRIBUTO_NOME [: DECL_TIPO]
MIN_CARD → 0 | 1
MAX_CARD → 1 | n
DECL_TIPO → inteiro | real | boolean | texto(INTEIRO) |
            enumeração(LISTA_VALORES)

SEÇÃO_IDENTIFICADOR → Identificadores: {DECL_IDENT}
DECL_IDENT → {IDENTIFICADOR}
IDENTIFICADOR → ATRIBUTO_NOME |
                ENTIDADE_NOME (via RELACIONAMENTO_NOME)

SEÇÃO_GENERALIZAÇÃO → {DECL_HIERARQUIA_GEN}
DECL_HIERARQUIA_GEN → Generalização [(COBERTURA)]: NOME_GEN
                      PAI: NOME_ENTIDADE
                      FILHO: LISTA_NOME_ENTIDADE

COBERTURA → t | p

SEÇÃO_ENT_ASSOCIATIVA → {DECL_ENT_ASSOC}
DECL_ENT_ASSOC → EntidadeAssociativa: NOME_RELACIONAMENTO

SEÇÃO_RELACIONAMENTO → {DECL_RELACION}
DECL_RELACION → Relacionamento: NOME_RELACIONAMENTO
                Entidades: {DECL_ENT-RELACIONADA}
                [Atributos: {DECL_ATRIB}]
                [Identificadores: {DECL_IDENT}]
DECL_ENT-RELACIONADA → [(CARD_MIN,CARD_MAX)] NOME_ENTIDADE
```

Figura 2.35 Gramática BNF de uma linguagem para definição de esquema.

as seguintes convenções: colchetes denotam opcionalidade, chaves denotam repetição, o sufixo *LISTA* denota uma seqüência de elementos separados por vírgulas e o sufixo *NOME* denota identificadores[3].

A Figura 2.36 apresenta um esquema ER textual correspondente ao esquema ER gráfico da Figura 2.20. Note-se que a representação gráfica e a textual aqui usadas não são exatamente equivalentes. A notação textual aqui usada é

[3] Para mais detalhes sobre o que é uma gramática BNF consulte algum livro sobre linguagens de programação ou sobre construção de compiladores.

```
Esquema: EMP_DEP

Entidade: EMPREGADO
          Atributos: CÓDIGO: inteiro
          Identificadores: CÓDIGO

Entidade: DEPENDENTE
          Atributos: NÚMERO_SEQUENCIA:
                     inteiro
                     NOME: texto(50)
        Identificadores: EMPREGADO via
                         EMP_DEP
                         NÚMERO_SEQUEN-
                         CIA

Relacionamento: EMP_DEP
          Entidades: (1,1) EMPREGADO
                     (0,n) DEPENDENTE
```

Figura 2.36 Esquema ER textual correspondente à Figura 2.20.

mais rica que a notação gráfica, pois inclui a possibilidade de definir um *tipo de atributo* (declaração *DECL_TIPO*).

Na prática, é comum combinar as duas formas de representar esquemas ER: a diagramática e a textual. Escolhe-se a forma de representar de acordo com o que se deseja representar. Entidades e relacionamentos, bem como hierarquias de generalização/especialização, são normalmente representadas de forma gráfica, pois a representação textual de grafos é difícil de ler. Já os atributos das entidades e dos relacionamentos, bem como a definição de identificadores podem ser feitos de forma textual, para não sobrecarregar o diagrama.

2.8 → exercícios

exercício 1 Dê ao menos cinco exemplos de cada um dos conceitos básicos da abordagem ER apresentados neste capítulo: entidade, relacionamento, atributo, generalização/especialização.

exercício 2 Explique a diferença entre uma entidade e uma ocorrência de entidade. Exemplifique.

exercício 3 O que é o papel de uma entidade em um relacionamento. Quando é necessário especificar o papel das entidades de um relacionamento?

exercício 4 Considere o relacionamento *CASAMENTO* que aparece no DER da Figura 2.7. Segundo este DER o banco de dados poderia conter um ca-

samento em que uma pessoa está casada consigo mesma? O DER permite que a mesma pessoa apareça em dois casamentos diferentes, uma vez como marido e outra vez como esposa? Caso uma destas situações possa ocorrer, como o DER deveria ser modificado para impedi-las?

exercício 5 Confeccione um possível diagrama de ocorrências para o relacionamento *SUPERVISÃO* (Figura 2.8) e suas respectivas entidades.

exercício 6 Confeccione um possível diagrama de ocorrências para o relacionamento *COMPOSIÇÃO* (Figura 2.9) e suas respectivas entidades.

exercício 7 Mostre como o modelo ER da Figura 2.11 pode ser representado sem o uso de relacionamentos ternários, apenas com relacionamentos binários.

exercício 8 Dê um exemplo de um relacionamento ternário. Mostre como a mesma realidade pode ser modelada somente com relacionamentos binários.

exercício 9 Para o exemplo de relacionamento ternário da questão anterior, justifique a escolha das cardinalidades mínima e máxima.

exercício 10 Considere o DER da Figura 2.12. Para que a restrição de cardinalidade mínima seja obedecida, que ocorrências de entidade devem existir no banco de dados, quando for incluída uma ocorrência de *EMPREGADO*? E quando for incluída uma ocorrência de *MESA*?

exercício 11 Construa um DER que modela a mesma realidade que é modelada pelo DER da Figura 2.16, usando apenas relacionamentos 1:n.

exercício 12 Considere o relacionamento *EMPREGADO-DEPENDENTE* que aparece na Figura 2.20. Considere que um dependente de um empregado possa ser também empregado. Como o modelo deveria ser modificado para evitar o armazenamento redundante das informações das pessoas que são tanto dependentes quanto empregados?

exercício 13 Invente exemplos de entidades com vários tipos de identificadores:

- uma entidade cujo identificador é composto por um único atributo;
- uma entidade cujo identificador é composto por mais de um atributo;

- uma entidade cujo identificador é composto por relacionamentos e
- uma entidade cujo identificador é composto por atributos e relacionamentos.

exercício 14 Construa um DER em que o conceito de entidade associativa é usado.

exercício 15 Dê ao menos três exemplos de entidades com relacionamentos identificadores (entidades fracas).

exercício 16 Considere o exemplo da Figura 2.13. Modifique as cardinalidades mínimas de forma a especificar o seguinte:

- Um curso não pode estar vazio, isto é, deve possuir ao menos uma disciplina em seu currículo.
- Um aluno, mesmo que não inscrito em algum curso, deve permanecer por algum tempo no banco de dados.

exercício 17 Sem usar atributos opcionais, nem atributos multivalorados, construa um DER que contenha as mesmas informações do DER da Figura 2.15.

exercício 18 O DER da Figura 2.28 modela uma generalização/especialização compartilhada. Construa um DER que modela a realidade descrita sem usar o conceito de generalização/especialização.

exercício 19 Para cada um dos quatro tipos de generalização/especialização (Tabela 2.1) conceba uma realidade que necessite o tipo em questão e construa um DER que a modela.

exercício 20 A Figura 2.37 apresenta um modelo de dados para uma farmácia. Descreva em português tudo o que está representado neste diagrama.

exercício 21 Invente nomes para os relacionamentos da Figura 2.37.

exercício 22 Dê uma justificativa para as cardinalidades mínimas do relacionamento entre *FORNECEDOR* e *FABRICANTE* no DER da Figura 2.37.

exercício 23 Explique o significado das cardinalidades mínima e máxima dos relacionamentos entre *MEDICAMENTO*, *VENDA* e *RECEITA MÉDICA*, no DER da Figura 2.37.

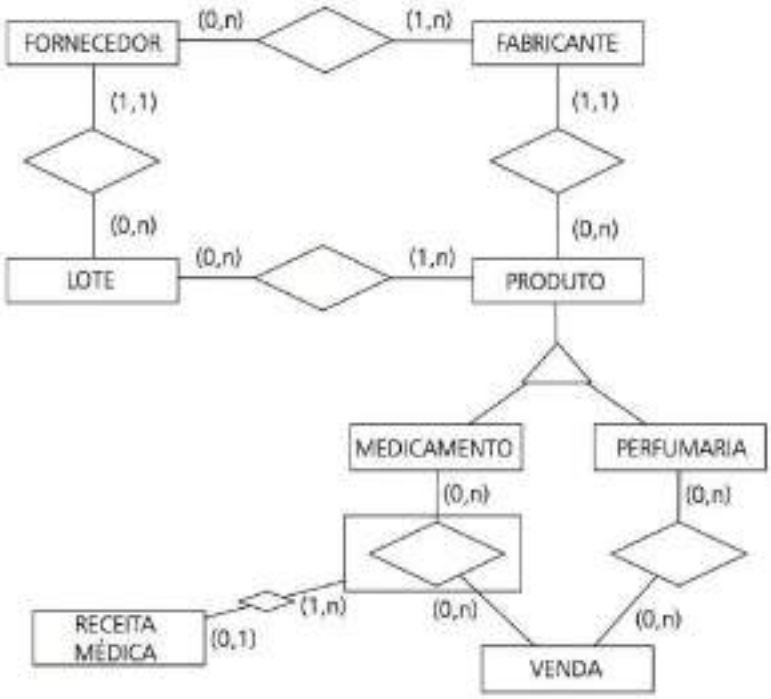


Figura 2.37 Diagrama ER de uma farmácia.

exercício 24 No modelo ER da Figura 2.37, em princípio, uma venda deve envolver ao menos um produto. Entretanto, isso não é exigido pelas cardinalidades mínimas dos relacionamentos entre *VENDA* e *MEDICAMENTO* e entre *VENDA* e *PERFUMARIA* no DER da Figura 2.37. Explique por quê.

exercício 25 Para cada entidade e cada relacionamento no DER da Figura 2.37 defina atributos quando possível. Para cada entidade, indique o(s) atributo(s) identificador(es).

exercício 26 Escreva um esquema ER textual para o esquema diagramático da Figura 2.37.

exercício 27 A Figura 2.38 apresenta um DER de parte de um sistema de recursos humanos em uma organização. Descreva em português tudo o que está representado neste diagrama.

exercício 28 Para cada entidade e cada relacionamento do DER da Figura 2.38 defina atributos quando possível. Para cada entidade, indique o(s) atributo(s) identificador(es).

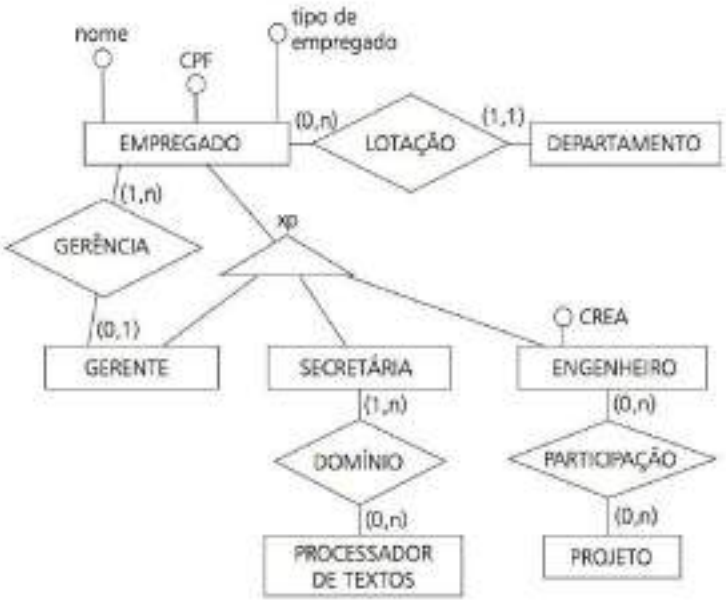


Figura 2.38 Diagrama ER para um sistema de recursos humanos.

exercício 29 Escreva um esquema ER textual para o esquema diagramático da Figura 2.38.

exercício 30 De acordo com o DER da Figura 2.38, que ações devem ser tomadas ao excluir-se do banco de dados uma secretária?

exercício 31 De acordo com o DER da Figura 2.38, uma secretária ou um engenheiro não podem ser gerentes. Por quê? Como o DER deveria ser modificado para permitir que tanto uma secretária, quanto um engenheiro pudessem ser também gerentes?

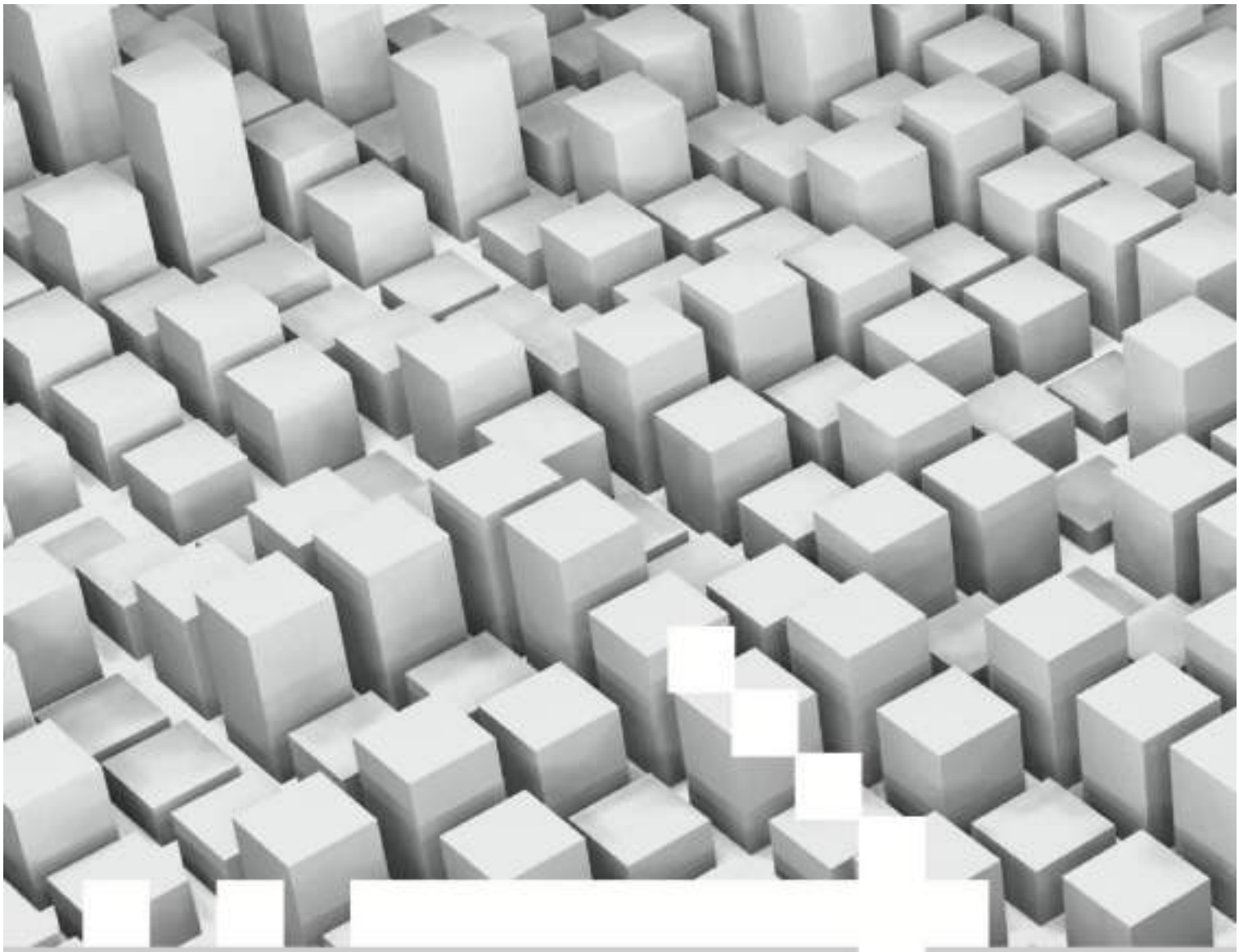
2.9 → leituras recomendadas

O artigo original sobre a abordagem ER é CHEN (1976). Na abordagem original de Chen apareciam apenas os conceitos de entidade, relacionamento e atributo. Ainda não apareciam os conceitos de generalização/especialização, nem de entidade associativa. Estes foram introduzidos mais tarde em diversos trabalhos como SANTOS; NEUHOLD; FURTADO (1980) e SCHEUERMANN; SCHIFFNER; WEBER (1980).

O trabalho de Chen baseou-se em diversas propostas de modelos de dados que foram feitas à época. Um dos artigos fundamentais sobre o assunto modelagem de dados é ABRIAL (1974). Este é historicamente o primeiro a tratar do problema de modelagem semântica de dados, isto é, da modelagem sem considerar aspectos de implementação. Os trabalhos de Kent (KENT (1978, 1979)) abordam em detalhes a diferença entre um modelo lógico que inclui detalhes de implementação e um modelo semântico de dados, abstrato e independente de implementação, como é a abordagem ER. Já o trabalho SMITH; SMITH (1977) apresenta os conceitos sobre os quais baseiam-se a maioria das abordagens de modelagem de dados, inclusive a abordagem ER.

Além da abordagem ER, outras abordagens de modelagem foram propostas na literatura. Apesar de a abordagem ER continuar sendo a abordagem de modelagem de dados mais aceita, é interessante estudar as propostas concorrentes, para compreender os pontos fracos da abordagem ER e as propostas para corrigi-los. A técnica NIAM/ORM (VERHEIJEN; BEKKUM (1982); NIJSSEN; HALPIN (1989)) provavelmente é o concorrente mais importante da abordagem ER pois teve, principalmente durante a década de 80 e na Europa, um número considerável de usuários. A principal característica de NIAM/ORM que a diferencia da abordagem ER é a ausência do conceito de atributo.

A abordagem ER é apresentada em vários livros-texto sobre projeto de banco de dados. Um texto completo e detalhado é BATINI; CERI; NAVATHE (1992). Outros livros conhecidos são TEOREY (1994) e MANINILA; RäIHä (1992), este último mais dedicado a aspectos teóricos.



capítulo 3

construindo modelos ER

No capítulo anterior, vimos como um diagrama ER é composto. Neste capítulo, vamos nos concentrar na construção de modelos ER dada uma determinada realidade. Começamos com a apresentação de uma coletânea de conselhos práticos e heurísticas a serem usados durante a modelagem conceitual. A seguir, são apresentadas notações alternativas à de Peter Chen para a confecção de diagramas ER. O capítulo encerra-se com um processo de modelagem e discutindo alternativas a este processo.

3.1 → propriedades de modelos ER

Nesta seção, discutimos algumas propriedades de modelos ER relevantes para a modelagem ER.

3.1.1 um modelo ER é um modelo formal

Um modelo ER é um modelo formal, preciso, não ambíguo. Isto significa que diferentes leitores de um mesmo modelo ER devem sempre entender exatamente o mesmo. Tanto é assim, que um modelo ER pode ser usado como entrada a uma ferramenta CASE (*Computer Aided Software Engineering*)[1] na geração de um banco de dados relacional. Por isso, é de fundamental importância que todos os envolvidos na confecção e no uso de diagramas ER sejam devidamente treinados.

Observa-se, em certas organizações, que modelos ER são subutilizados, servindo apenas como ferramenta para a apresentação informal de idéias. Isso pode ser evitado com treinamento formal de todos os envolvidos na modelagem e no projeto do banco de dados.

É importante que *todos* os que manipulam modelos ER sejam treinados para compreendê-lo. O fato de um DER ser gráfico e intuitivo pode transmitir a falsa impressão de ser compreensível até por alguém não treinado.

Para exemplificar alguns problemas que surgem ao usar modelos ER sem treinar as pessoas envolvidas, descrevo uma situação que já observei em algumas organizações. Os técnicos em computação da organização desenvolveram um modelo ER a partir de sua compreensão sobre o sistema a ser construído, obtida a partir de entrevistas com os futuros usuários do sistema. Para validar o modelo, este foi apresentado aos usuários. Entretanto, os usuários não foram treinados em modelagem e compreendiam o modelo apenas como uma descrição gráfica informal. Os usuários concordaram com o modelo ER que lhes foi apresentado. Mais tarde, já com o banco de dados em funcionamento, descobriu-se que os usuários não haviam entendido efetivamente o que foi modelado. O banco de dados implementado não era exatamente aquele

[1] Ferramenta CASE: *software* que dá suporte à construção de *software* (CASE vem do inglês *Computer Aided Software Engineering* que significa "engenharia de *software* suportada por computador").

desejado pelos usuários, apesar de corresponder exatamente ao modelo ER apresentado. Assim, para que modelos ER possam atingir seus objetivos, é necessário treinamento formal. Não estou sugerindo que os usuários tenham que ser treinados como modeladores. Basta que eles recebam algumas horas de treinamento na leitura e compreensão de diagramas ER.

3.1.2 modelos ER têm poder de expressão limitado

Em um modelo ER, são apresentadas apenas algumas propriedades de um banco de dados. Na realidade, a linguagem dos modelos ER é uma linguagem muito pouco poderosa e muitas propriedades desejáveis do banco de dados necessitam ser anotadas adicionalmente ao DER. A seguir, mostramos dois exemplos de DER, já vistos no capítulo anterior, para salientar o quão incompletos são estes modelos.

A Figura 3.1 mostra o DER que modela pessoas e casamentos já mostrado no capítulo anterior. Ao lado, são mostradas possíveis ocorrências de *PESSOA* e *CASAMENTO*. Entretanto, as ocorrências de *CASAMENTO* não correspondem ao nosso conhecimento da realidade (pelo menos se considerarmos a legislação brasileira à época da escrita deste livro). A pessoa *p3* aparece em dois casamentos, uma vez no papel de esposa e outra vez no papel de marido. O mesmo ocorre com a pessoa *p3*. Além disso, a pessoa *p5* aparece casada consigo mesma (o que poderia ser bom para um contribuinte do imposto de renda – ter-se-ia os descontos referentes a um dependente, sem incorrer nos custos referentes a um dependente).

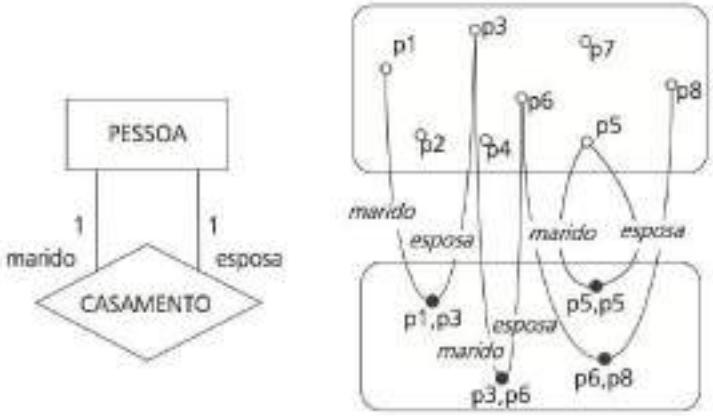


Figura 3.1 DER e diagrama de ocorrências para *CASAMENTO*.

Uma regra estabelecida pela realidade modelada e que deve ser obedecida pelo banco de dados é chamada de *restrição de integridade*. No caso dos casamentos, um exemplo de restrição de integridade é o fato de uma pessoa não poder ter mais que um marido e o fato de uma pessoa não poder casar consigo mesma. Algumas restrições de integridade podem ser expressas diretamente no modelo ER, por exemplo através de restrições de cardinalidade; já outras somente podem ser expressas em separado através de alguma outra linguagem. Como não há padrão aceito para a linguagem de especificação de restrições de integridade, estas normalmente são expressas em linguagem natural.

Caso se queira um modelo mais fiel da realidade apresentada na Figura 3.1, é necessário modificar o modelo ER ou então definir restrições adicionais. No caso em questão, é possível modificar o modelo ER para excluir os casamentos indesejáveis (ver exercício no capítulo anterior).

Aqui cabe a pergunta: até onde deve ser modificado um modelo ER para introduzir restrições de integridade? A resposta não é trivial. É necessário lembrar o objetivo que se tem ao construir um modelo ER: o de projetar um banco de dados. Neste contexto, o modelo ER nada mais é do que uma descrição abstrata das estruturas do banco de dados (das tabelas, no caso de um banco de dados relacional). O objetivo do modelo não é o de especificar todas as restrições de integridade. Assim, somente são incluídas construções em um modelo ER, quando estas possuem uma correspondência no banco de dados a ser implementado. Construções artificiais, isto é, construções incluídas no modelo apenas para satisfazer determinadas restrições de integridade são indesejáveis, pois distorcem os objetivos que se tem ao construir o DER.

Outro exemplo de restrições de integridade que não se deixam expressar através de um modelo ER aparece no modelo da Figura 3.2. Esta figura apresenta um exemplo de um DER modelando empregados e a hierarquia de supervisão em uma organização. O relacionamento *SUPERVISÃO* possui cardinalidade **1:n**, indicando que um empregado pode supervisionar muitos outros, mas possui no máximo um supervisor. Como está especificado, o modelo admite o diagrama de ocorrências que aparece na figura. Os relacionamentos mostrados informam que o empregado *e1* é supervisor do empregado *e3*, que por sua vez é supervisor de *e5*, o qual, por sua vez é supervisor de *e1*. Isto obviamente contraria nosso conhecimento sobre a realidade modelada, já que, em uma hierarquia de supervisão, não é permitido que um superior hie-

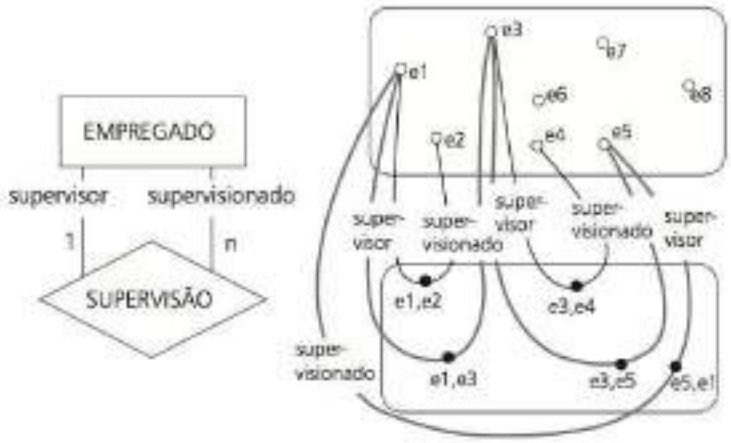


Figura 3.2 Relacionamento recursivo.

rárquico (no caso *e1*) apareça como supervisionado em um nível mais baixo da hierarquia (no caso, *e1* aparece como supervisionado de *e5*). Ao contrário do exemplo do casamento apresentado acima, não é possível introduzir esta restrição no modelo ER através de modificações. O problema é que esta é uma restrição de integridade *recursiva* e restrições recursivas não podem ser representadas através de modelos ER. Neste caso, resta apenas especificar a restrição a parte do DER.

3.1.3 diferentes modelos podem ser equivalentes

Na prática, muitas vezes observamos modeladores em acirradas discussões sobre como um determinado objeto da realidade modelada deve aparecer no modelo. Às vezes, tais discussões são absolutamente supérfluas, pois os diferentes modelos ER que estão considerando geram o mesmo banco de dados.

Há um conceito de *equivalência* entre modelos ER. De maneira informal, diz-se que dois modelos são equivalentes quando expressam o mesmo, ou seja, quando modelam a mesma realidade.

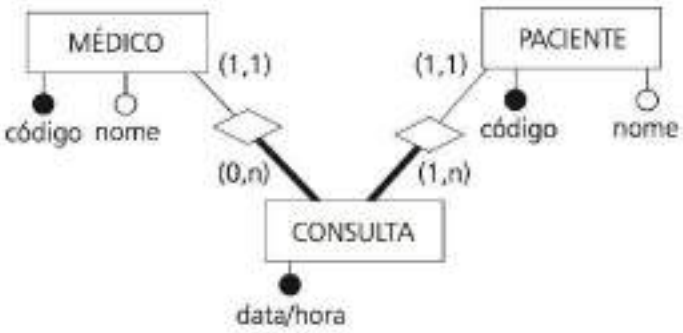
Para fins de projeto de banco de dados, dois modelos ER são equivalentes quando ambos geram o mesmo esquema de banco de dados. Assim, para analisar se dois modelos são equivalentes, é necessário considerar um conjunto de regras de tradução de modelos ER para modelos lógicos de banco de dados. Para este texto, vamos considerar as regras de tradução de modelo ER para modelo relacional apresentadas no capítulo 5. Dois modelos ER são

equivalentes caso gerem o mesmo modelo de banco de dados relacional, através do uso das regras de tradução apresentadas no capítulo 5. Quando falamos "o mesmo modelo de banco de dados relacional", estamos falando de bancos de dados que, abstraindo diferenças de nomes de construções (tabelas, atributos,...), tenham a mesma estrutura.

É claro que para entender perfeitamente este conceito de equivalência de modelos, o leitor deve conhecer as regras de tradução apresentadas no capítulo 5. Mesmo assim, considerando a definição informal de equivalência (dois modelos que expressam o mesmo), é possível compreender alguns exemplos da aplicação do conceito.

Um exemplo é o da equivalência entre um modelo que representa um conceito através de um relacionamento **n:n** e outro modelo que representa o mesmo conceito através de uma entidade. Considere os modelos ER apresentados na Figura 3.3, onde o relacionamento *CONSULTA* foi transformado em uma entidade. Os dois modelos são equivalentes, pois expressam o mesmo e geram o mesmo banco de dados.

a) CONSULTA como relacionamento n:n



b) CONSULTA como entidade

Figura 3.3 Transformando um relacionamento **n:n** em entidade.

A transformação de um relacionamento **n:n** em entidade segue o seguinte processo:

1. O relacionamento **n:n** é representado como uma entidade.
2. A entidade criada é relacionada às entidades que originalmente participavam do relacionamento.
3. A entidade criada tem como identificador:
   - os relacionamentos com as entidades que originalmente participavam do relacionamento e
   - os atributos que eram identificadores do relacionamento original (caso o relacionamento original tivesse atributos identificadores).
4. A cardinalidade da entidade criada em cada relacionamento de que participa é (1,1).
5. As cardinalidades das entidades que eram originalmente associadas pelo relacionamento transformado em entidade são transcritas ao novo modelo conforme mostrado na Figura 3.3.

Como todo relacionamento **n:n** pode ser transformado em entidade, é possível construir modelos sem relacionamentos **n:n**. Neste fato, baseiam-se algumas variantes da abordagem ER, que excluem o uso de relacionamentos **n:n**, e outras que excluem apenas o uso de relacionamentos **n:n** com atributos. Um exemplo são várias abordagens baseadas na Engenharia de Informações (página 99).

Outro exemplo de equivalência é entre um modelo ER com um relacionamento de cardinalidade **1:1** e com cardinalidade mínima "**1**" em ambos os lados, e outro modelo ER em que tal relacionamento foi substituído por uma única entidade.

3.2 → determinando construções

A determinação de qual construção da abordagem ER (entidade, relacionamento, atributo,...) será usada para modelar um objeto de uma realidade considerada não pode ser feita através da observação do objeto isoladamente. É necessário conhecer o contexto, isto é, o modelo dentro do qual o objeto aparece. Assim, a recomendação geral é considerar a decisão por uma construção para modelar um objeto como sujeita à alteração durante a modelagem. Não é aconselhável despender um tempo excessivo em lon-

gas discussões sobre como modelar um objeto. O próprio desenvolvimento do modelo e o aprendizado sobre a realidade refinarão e aperfeiçoarão o modelo.

Mesmo assim, existem alguns critérios que podem ser usados na escolha de construções de modelagem. Estes critérios são descritos nas próximas subseções.

3.2.1 atributo *versus* entidade relacionada

Uma questão que às vezes surge na modelagem de um sistema é quando modelar um objeto como sendo um *atributo* de uma entidade e quando modelar o mesmo objeto como sendo uma *entidade* autônoma relacionada a essa entidade.

Exemplificando, no caso de uma indústria de automóveis, como devemos registrar a cor de cada automóvel que sai da linha de produção → Caso considerarmos que cada automóvel possui uma única cor predominante, pode-se pensar em modelar a cor como um *atributo* da entidade *AUTOMÓVEL* (primeira opção da Figura 3.4). Outra opção é modelar a cor como uma *entidade* autônoma, que está relacionada à entidade *AUTOMÓVEL* (segunda opção da Figura 3.4).

Alguns critérios para esta decisão são:

- Caso o objeto cuja modelagem está em discussão esteja vinculado a outros objetos, ou seja, caso o objeto tenha propriedades (atributos, relacionamentos, entidades genéricas ou especializadas), o objeto deve ser

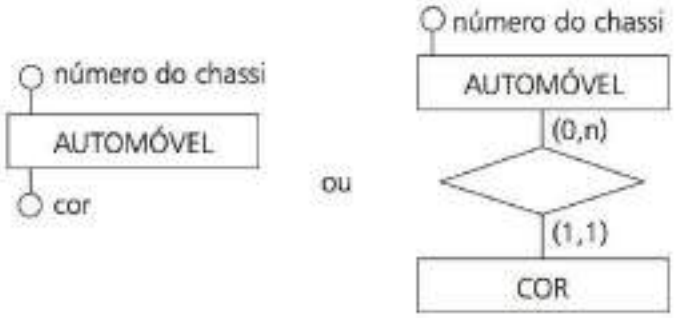


Figura 3.4 Escolhendo entre atributo e entidade relacionada.

modelado como *entidade*, já que um atributo não pode ter atributos, nem estar relacionado a outras entidades, nem ser generalizado ou especializado. Caso contrário, o objeto pode ser modelado como *atributo*. Assim, no exemplo das cores dos automóveis, poderíamos optar por modelar a cor como uma entidade, caso tivéssemos que registrar no banco de dados os possíveis fabricantes da tinta da referida cor (entidades relacionadas a cor), ou caso quiséssemos registrar as datas de início e fim (atributos) do uso de uma determinada cor. Caso não houvesse algum objeto relacionado à cor do automóvel, poderíamos modelá-la como atributo da entidade *AUTOMÓVEL*.

- Quando o conjunto de valores de um determinado objeto é fixo durante toda a vida do sistema, ele pode ser modelado como *atributo*, visto que o domínio de valores de um atributo é imutável. Quando existem transações no sistema, que alteram o conjunto de valores do objeto, o mesmo não deve ser modelado como atributo. Assim, retomando o exemplo das cores dos automóveis, caso existissem transações de criação/eliminação de cores, seria preferível a modelagem de cor como entidade relacionada à entidade *AUTOMÓVEL*.

3.2.2 atributo *versus* especialização

Outro conflito de modelagem para o qual há algumas regras de cunho prático é entre modelar um determinado objeto (por, exemplo, a categoria funcional de cada empregado de uma empresa) como *atributo* (categoria funcional como atributo da entidade *EMPREGADO* – Figura 3.5) ou como uma *especialização* (cada categoria funcional corresponde a uma especialização da entidade empregado – Figura 3.6).

Uma especialização deve ser usada quando sabe-se que as classes especializadas de entidades possuem propriedades (atributos, relacionamentos, generalizações, especializações) particulares. Assim, no exemplo acima, faz sentido

Figura 3.5 Modelando uma categoria funcional como atributo.

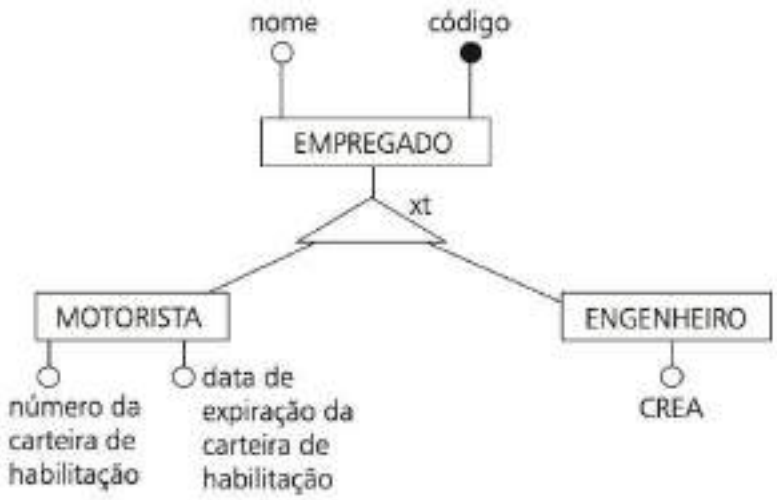


Figura 3.6 Modelando uma categoria funcional como especialização.

especializar a entidade empregado de acordo com a categoria funcional, no caso de empregados de uma ou mais categorias funcionais possuírem atributos ou relacionamentos próprios.

Na Figura 3.7, aparece outra variante para o exemplo dos empregados. Aqui deseja-se representar o fato de que os empregados são divididos em homens e mulheres. Caso, existam na realidade considerada propriedades específicas de homens ou mulheres, a solução adotada na Figura 3.7 (modelar sexo como especialização) é correta. Se para homens ou mulheres não existirem propriedades específicas de cada sexo, o modelo da Figura 3.8 (modelar sexo como atributo) seria o correto.

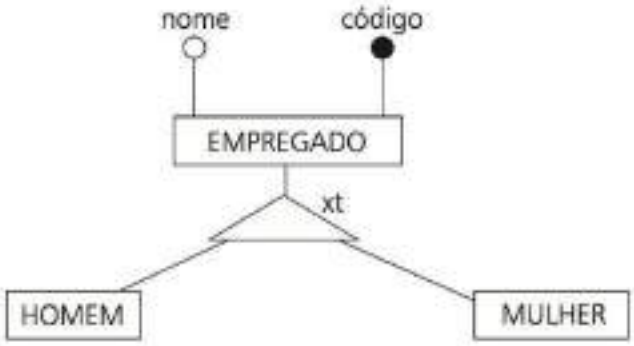


Figura 3.7 Modelando sexo como especialização.

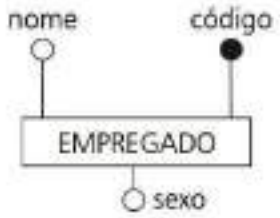


Figura 3.8 Modelando sexo como atributo.

3.2.3 entidade relacionada *versus* especialização

Na seção precedente, vimos que há critérios para decidir quando usar um atributo em contraposição a usar uma especialização. Nesta seção, veremos que há critérios para optar entre uma entidade relacionada e uma especialização.

Para que uma entidade possa ser considerada especialização de outra, é necessário que ela herde o identificador da entidade genérica.

Vamos considerar a seguinte realidade (uma extensão daquela descrita na Figura 2.28). Uma universidade deseja manter um cadastro de seus alunos e servidores. Cada um deles é identificado pelo número de um cartão de identificação emitido pela universidade. Além do número do cartão, são necessários o nome, a data de nascimento e o endereço de cada aluno e professor. Somente para servidores, a universidade deseja manter seu cargo (técnico, professor,...). Já para os alunos, a universidade deseja manter o nome da escola secundária na qual o aluno concluiu os estudos antes do ingresso na universidade, bem como o ano de ingresso na universidade. O modelo para esta realidade encontra-se representado na Figura 3.9.

Agora, vamos considerar alguns detalhes adicionais. Vamos supor que um aluno possa ingressar várias vezes na universidade, por exemplo para retomar um curso que havia trancado ou para ingressar em um novo curso. Vamos supor, também, que um servidor possa ocupar mais de um cargo ao mesmo tempo. Cada ocupação de um cargo por um servidor recebe um número que a identifica e é necessário saber as datas inicial e final da ocupação. Neste caso, o modelo deve ser modificado conforme mostra a Figura 3.10.

Vamos analisar o novo modelo. A entidade *SERVIDOR* deixa de existir. No novo modelo, não há propriedades específicas de servidor que devam ser

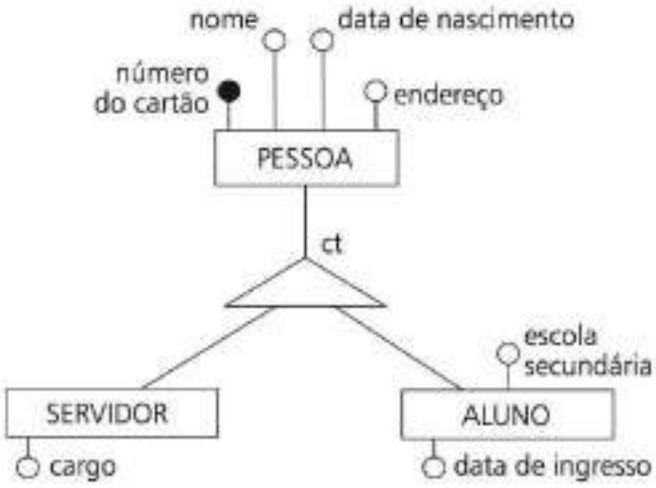


Figura 3.9 Servidor e aluno (versão 1).

armazenadas, apenas propriedades referentes a cada uma das ocupações de cargo por uma pessoa. Estas propriedades aparecem na entidade *SERVIDOR CARGO*. Esta entidade não pode ser considerada uma especialização da en-

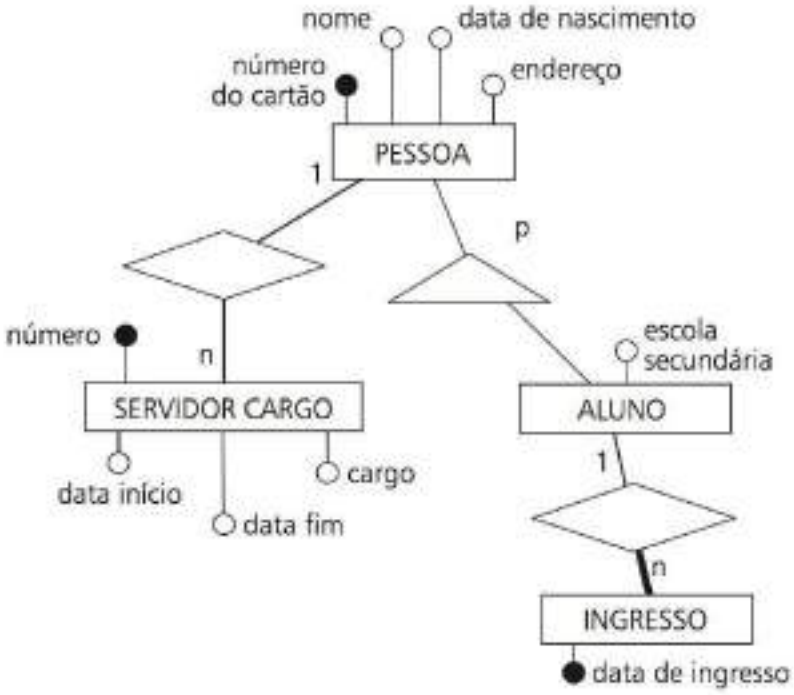


Figura 3.10 Servidor e aluno (versão 2).

tidade *PESSOA*, já que para uma instância de *PESSOA* podem existir várias instâncias da entidade *SERVIDOR CARGO*.

A entidade *ALUNO* continua existindo no novo modelo, já que há um atributo (escola secundária) que se refere à pessoa como aluno e não a cada ingresso do aluno na universidade. Para registrar os vários ingressos de um aluno, foi criada a entidade *INGRESSO*.

Resumindo, um objeto somente pode ser tratado como especialização de outro quando o objeto especializado herda a chave primária do objeto genérico. Isso significa, também, que para cada ocorrência do objeto genérico pode existir no máximo uma ocorrência na especialização.

3.2.4 atributos opcionais e multivalorados

Conforme vimos no capítulo anterior (Seção Atributo na página 53), atributos podem ser *opcionais* (nem toda ocorrência da entidade possui um valor do atributo) ou *multivalorados*. Entretanto, quando se inicia o processo de modelagem é aconselhável tentar restringir-se ao uso de atributos *obrigatórios* e *monovalorados* pelas razões abaixo listadas.

■ atributo opcional

Em muitos casos, atributos opcionais indicam subconjuntos de entidades que são modelados mais corretamente através de especializações. O modelo ER da Figura 3.11 apresenta uma entidade com diversos atributos opcionais. Segundo este modelo, nem todo empregado possui um registro no CREA (Conselho Regional de Engenharia e Arquitetura), nem todo empregado pos-

Figura 3.11 Atributos opcionais.

sui um registro no CRM (Conselho Regional de Medicina), etc. Entretanto, o modelo não representa que combinações de atributos são válidas. Será que um empregado pode possuir atributos CREA, CRM e data de expiração da carteira de habilitação, mas não possuir o número da carteira?

No exemplo, o uso de atributos opcionais esconde as diferentes categorias de empregados e suas entidades. A realidade é modelada com maior fidelidade, caso a entidade *EMPREGADO* seja especializada conforme mostra a Figura 3.12. Neste modelo, fica claro quais são os atributos de cada um dos subconjuntos particulares de *EMPREGADO*.

Assim, toda vez que aparecer um atributo opcional, é recomendável verificar se a modelagem através de entidades especializadas não é mais conveniente.

■ atributo multivalorado

Atributos multivalorados podem ser representados em diagramas ER, como mostra a Figura 3.13.

Entretanto, atributos multivalorados são considerados indesejáveis por muitos autores pelas seguintes razões:

- Nos SGBDs relacionais que seguem o padrão SQL/2, atributos multivalorados não possuem implementação direta. Não existe, em um SGBD relacional, uma construção como os "arrays" de Pascal, nem como os grupos

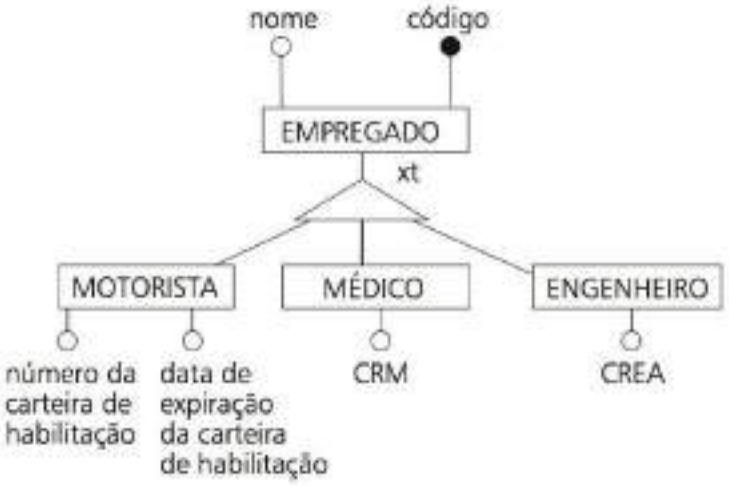


Figura 3.12 Aumentando a fidelidade do modelo através de especializações.

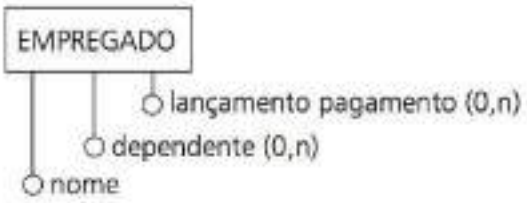


Figura 3.13 Usando atributos multivalorados.

repetidos de COBOL. Assim, caso esteja sendo projetado um banco de dados relacional, é aconselhável usar apenas atributos monovalorados. Esta restrição provavelmente tenderá a cair com a introdução dos chamados SGBD objeto/relacional, que são propostos no padrão SQL/3.

- Atributos multivalorados podem induzir a um erro de modelagem, que é o de ocultar entidades e relacionamentos em atributos multivalorados. Vamos dar ao exemplo da Figura 3.13 a seguinte interpretação. Como atributos de *EMPREGADO* aparecem o nome do empregado, seus dependentes e os lançamentos de pagamento que compõem seu contra-cheque. Entretanto, ao considerar a entidade *EMPREGADO* mais detalhadamente, observa-se que tanto dependentes, quanto os lançamentos possuem propriedades particulares. Cada dependente possui um nome e uma data de nascimento. Já um lançamento de pagamento possui um valor e um tipo de lançamento, sobre o qual também nos interessa manter informações, como o código do tipo e sua descrição. Assim, o modelo correto para a realidade em questão é o apresentado na Figura 3.14.

3.3 → verificação do modelo

Uma vez construído, um modelo ER deve ser validado e verificado. A verificação é o controle de qualidade que procura garantir que o modelo usado para a construção do banco de dados gerará um bom produto. Um modelo, para ser considerado bom, deve preencher uma série de requisitos, como ser completo, correto e não conter redundância.

3.3.1 um modelo deve ser correto

Um modelo está correto quando não contém erros de modelagem, isto é, quando os conceitos de modelagem ER são corretamente empregados para modelar a realidade em questão. Pode-se distinguir entre dois tipos de erros,

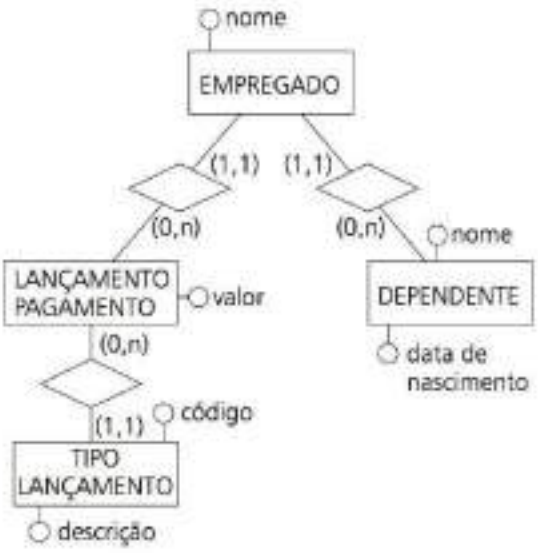


Figura 3.14 Substituindo atributos multivalorados por entidades relacionadas.

os erros *sintáticos* e os erros *semânticos*. Os erros sintáticos ocorrem quando o modelo não respeita as regras de construção de um modelo ER. Exemplos de erros sintáticos são o de associar atributos a atributos, o de associar relacionamentos a atributos, o de associar relacionamentos através de outros relacionamentos ou de especializar relacionamentos ou atributos. Já os erros semânticos ocorrem quando o modelo, apesar de obedecer as regras de construção de modelos ER (estar sintaticamente correto) reflete a realidade de forma inconsistente. Alguns exemplos de erros semânticos praticados freqüentemente são:

- `Estabelecer associações incorretas` – Um exemplo é associar a uma entidade um atributo que na realidade pertence a outra entidade. Por exemplo, em um modelo com entidades *CLIENTE* e *FILIAL*, associar à *CLIENTE* o nome da filial com o qual o cliente trabalha usualmente (nome de filial é um atributo de *FILIAL*).
- `Usar uma entidade do modelo como atributo de outra entidade` – Um exemplo seria ter, em um modelo, uma entidade *BANCO* e, no mesmo modelo, tratar banco como atributo de outra entidade *CLIENTE*. Cada objeto da realidade modelada deve aparecer uma única vez no modelo ER.
- `Usar o número incorreto de entidades em um relacionamento` – Um exemplo é o de fundir em um único relacionamento ternário dois relacionamentos binários independentes.

As regras de normalização de bases de dados relacionais apresentadas no capítulo 6 servem igualmente para verificar a correção de modelos ER.

3.3.2 um modelo deve ser completo

Um modelo completo deve fixar todas as propriedades desejáveis do banco de dados. Isso somente pode ser verificado por alguém que conhece profundamente o sistema a ser implementado. Uma boa forma de verificar se o modelo é completo é ver se todos os dados que devem ser obtidos do banco de dados estão presentes e se todas as transações de modificação do banco de dados podem ser executadas sobre o modelo.

Este requisito é aparentemente conflitante com a falta de poder de expressão de modelos ER que foi citada acima. Quando dizemos que um modelo deve ser completo, estamos exigindo que todas as propriedades *expressáveis* com modelos ER apareçam no modelo.

3.3.3 um modelo deve ser livre de redundância

Um modelo deve ser mínimo, isto é, não deve conter conceitos redundantes. Um tipo de redundância que pode aparecer é a de relacionamentos redundantes. *Relacionamentos redundantes* são relacionamentos resultantes da combinação de outros relacionamentos entre as mesmas entidades.

Os relacionamentos desenhados em linhas densas no DER da Figura 3.15 são redundantes. Um relacionamento é redundante quando é possível eliminá-lo do modelo ER sem que haja perda de informações no banco de dados. No caso do relacionamento *LOCALIZAÇÃO-FÁBR*, a associação entre entidades por ele expressa já está contida nos relacionamentos *LOCALIZAÇÃO-DEPTO* e *F-D*. Em outros termos, como o banco de dados informa em que departamento uma máquina está localizada e em que fábrica o departamento está localizado, ele informa, por conseqüência, em que fábrica uma máquina está localizada. Seguindo um raciocínio análogo é fácil verificar que o relacionamento *ATUAÇÃO*, entre *FÁBRICA* e *SINDICATO*, é redundante, pois a informação nele contida é derivada da informação fornecida pelos relacionamentos *ASSOCIAÇÃO*, *D-E* e *F-D*.

Outro tipo de redundância que pode aparecer em modelos ER é a de atributos redundantes. *Atributos redundantes* são atributos deriváveis a partir da

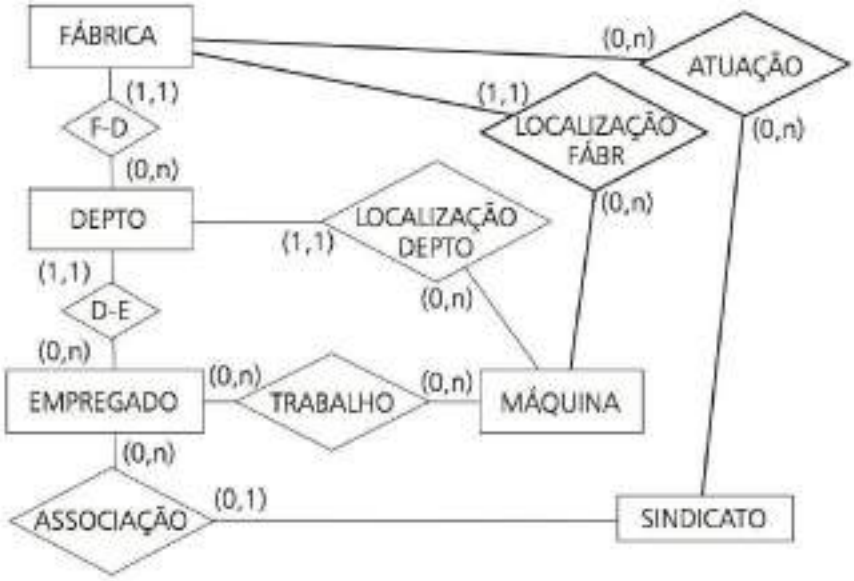


Figura 3.15 Relacionamentos redundantes.

execução de procedimentos de busca de dados e/ou cálculos sobre o banco de dados.

A Figura 3.16 apresenta um modelo ER que contém dois atributos redundantes (nº de empregados e código do departamento). O atributo código do departamento é redundante, pois pode ser obtido através do acesso à entidade *DEPARTAMENTO* associada à entidade *EMPREGADO* através do relacionamento de *LOTAÇÃO*[2]. Já o atributo nº de empregados é redundante pois pode ser obtido através de um processo de contagem sobre o relacionamento *LOTAÇÃO*.

Construções redundantes devem ser omitidas do modelo ER. Como o modelo ER não distingue construções derivadas das demais construções, o projetista do banco de dados poderia ser levado a implementá-las gerando uma redundância não controlada de dados. A redundância não controlada de dados em um banco de dados é indesejável por diversas razões, conforme discutido no capítulo 1. Observe que isso não significa que a redundância *controlada*

[2] Para o leitor afeito à terminologia de bancos de dados relacionais, código do departamento é uma *chave estrangeira*. Como veremos nos próximos capítulos, este atributo é usado em um banco de dados relacional para implementar o relacionamento *LOTAÇÃO*. Como um modelo ER não deve conter ainda detalhes de implementação com SGBD relacional, o atributo deve ser omitido.

Figura 3.16 Atributos redundantes.

de dados (redundância de dados da qual programas e usuários têm conhecimento) deva também ser necessariamente evitada. Às vezes, construções redundantes em um banco de dados podem servir para aumentar a performance de operações de busca de informações no banco de dados, mas nem por isso devem aparecer no modelo conceitual do banco de dados.

3.3.4 um modelo deve refletir o aspecto temporal

Certas aplicações exigem que o banco de dados guarde o histórico de alteração de informações. Por exemplo, em um banco de dados de uma seguradora, pode ser necessário conhecer não só o segurado atual de uma apólice, mas também os do passado. O modelo de um banco de dados que armazena somente os valores atuais de uma informação é diferente do modelo do banco de dados que armazena o histórico da informação. Portanto, é necessário considerar o *aspecto temporal* na modelagem de dados. Não há regras gerais de como proceder neste caso, mas é possível identificar alguns padrões que repetem-se freqüentemente na prática.

■ atributos cujos valores modificam ao longo do tempo

Alguns atributos de uma entidade, normalmente aqueles que não são identificadores da entidade, podem ter seus valores alterados ao longo do tempo (por exemplo, o endereço de um cliente pode ser modificado). Algumas vezes, por necessidades futuras de informações, ou até mesmo por questões legais, o banco de dados deve manter um registro histórico das informações. Um exemplo é o valor do salário de um empregado. Em um sistema de pagamento, interessa saber não apenas o estado atual, mas também o salário durante os últimos meses, por exemplo, para emitir uma declaração anual de rendimentos de cada empregado. Assim, salário não pode ser modelado

como um atributo, mas sim como uma entidade. A Figura 3.17 apresenta as duas alternativas de modelagem.

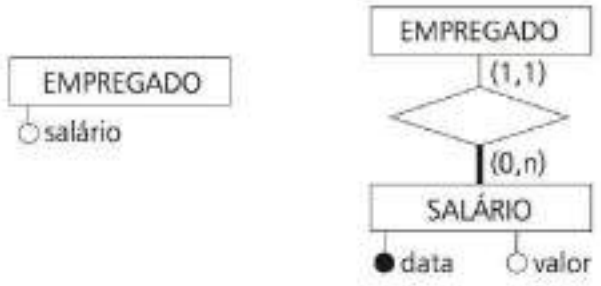
Na solução aqui apresentada, o atributo salário migrou para uma entidade (*SALÁRIO*), identificada pelo empredo relacionado e por um ponto no tempo (atributo data) em que o salário vigorou. Esta é apenas uma entre várias alternativas para representar atributos temporais. Outra alternativa é usar uma faixa de tempo (de-até) em vez de um ponto no tempo. Também é possível manter o valor atual do atributo na entidade original (no exemplo, na entidade *EMPREGADO*) e passar à nova entidade apenas os valores passados.

■ relacionamentos que modificam ao longo do tempo

Assim como atributos podem ter seus valores modificados ao longo do tempo, também os relacionamentos podem ser modificados (instâncias do relacionamento podem ser adicionadas/removidas) e também neste caso pode ser requerido que o banco de dados mantenha um registro histórico das alterações.

Quando é considerada a história de suas alterações, relacionamentos **1:1** ou **1:n** são transformados em **n:n**.

Para exemplificar, consideramos o relacionamento *ALOCAÇÃO* da Figura 3.18(a). Este relacionamento possui cardinalidade **1:1**, ou seja, cada empregado está alocado a no máximo uma mesa e cada mesa tem a ela alocado no máximo um empregado. Este modelo está correto, caso queiramos arma-



(a) Banco de dados contém apenas o salário atual

(b) Banco de dados contém a história dos salários

Figura 3.17 Modelando a dimensão temporal de atributos.

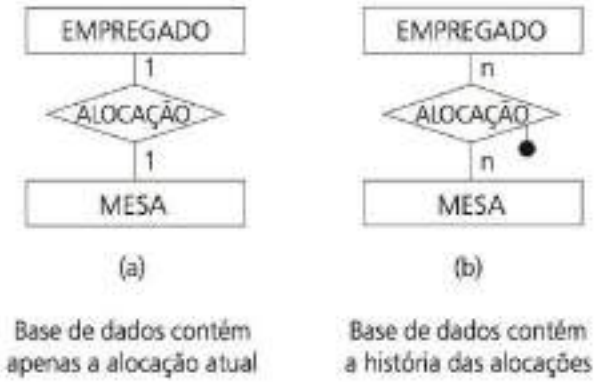


Figura 3.18 Modelando a dimensão temporal de relacionamentos 1:1.

zenar no banco de dados apenas a alocação atual de cada mesa. Entretanto, caso desejemos armazenar também a história das alocações, isto é, quais empregados estiveram alocados a quais mesas ao longo do tempo, é necessário modificar o modelo (Figura 3.18 (b)). O relacionamento passa a ter cardinalidade **n:n**, já que, ao longo do tempo, um empregado pode ter sido alocado a diversas mesas e uma mesa pode ter tido a ela alocados muitos empregados. Como um mesmo empregado pode ter sido alocado à mesma mesa múltiplas vezes, torna-se necessário um atributo identificador do relacionamento, a fim de distinguir uma alocação de um determinado empregado a uma mesa, das demais alocações deste empregado à mesma mesa. Com isso surge o atributo identificador data.

Conforme mencionado na página 81, em muitas versões da abordagem ER, principalmente em algumas usadas na prática (ver Seção Variantes da abordagem ER na página 99), não é permitido usar atributos identificadores de relacionamentos. Estes relacionamentos devem ser tratados como entidades. Neste caso, a entidade *ALOCAÇÃO* deveria ser representada como uma entidade e não como um relacionamento.

A Figura 3.19 apresenta uma situação semelhante, agora considerando um relacionamento de cardinalidade **1:n**. Se quisermos considerar a história das lotações de empregados ao longo do tempo, é necessário transformar o relacionamento para a cardinalidade **n:n**, já que, ao longo do tempo, um empregado pode ter atuado em diferentes departamentos. Neste caso, pode ocorrer também que atributos da entidade *EMPREGADO* migrem para o relacionamento. No exemplo, isto ocorre com o atributo *nº documento*

Figura 3.19 Modelando a dimensão temporal de relacionamentos **1:n**.

de lotação. Este atributo, que, na primeira versão do modelo, aparece na entidade *EMPREGADO*, migra, na nova versão, para o relacionamento, já que, na nova versão, um empregado pode estar relacionado com múltiplos departamentos.

A Figura 3.20 apresenta outro exemplo de relacionamento temporal, agora de cardinalidade **n:n**. Modela-se a inscrição de participantes nos cursos oferecidos por uma empresa de treinamento. Na primeira versão, considera-se apenas os cursos nos quais uma pessoa está inscrita em um determinado instante no tempo. Na segunda versão, considera-se todas as inscrições, inclusive as do passado. A modificação de uma versão para a outra consta

Figura 3.20 Modelando a dimensão temporal de relacionamentos **n:n**.

da transformação do atributo data em atributo identificador. Isto ocorre porque, na segunda versão, um participante pode aparecer relacionado múltiplas vezes a um determinado curso (caso ele tenha se inscrito múltiplas vezes no curso). O atributo passa a distinguir uma inscrição de uma pessoa em um curso, das demais inscrições desta pessoa no mesmo curso.

■ consultas a dados referentes ao passado

Muitas vezes, para evitar o crescimento desmedido do banco de dados, informações referentes ao passado são eliminadas. Entretanto, estas informações podem ser necessárias no futuro, por exemplo, por motivos legais, para a realização de auditorias ou para a tomada de decisões. Portanto, é necessário planejar, desde a modelagem, por quanto tempo as informações ficarão armazenadas no banco de dados. Caso informações antigas fiquem no banco de dados, podem ser necessários atributos para indicar o *status* da informação, se é atual ou antiga.

- `Planejar o arquivamento de informações antigas` – Para as informações que serão retiradas do banco de dados e armazenadas em arquivos convencionais, é necessário fazer um planejamento de como estas informações serão acessadas no futuro, caso venham a ser necessárias. Uma solução que poderia ser considerada é a de reincluir as informações no banco de dados quando elas forem necessárias no futuro. Isso permite que, para buscar as informações passadas, sejam utilizados os mesmos procedimentos que são usados para acessar as informações atuais. Entretanto, é necessário considerar que as informações em um banco de dados estão normalmente relacionadas a outras. Caso as ocorrências de entidade que se deseja devolver ao banco de dados estejam relacionadas a outras ocorrências, é necessário que estas estejam presentes. Se elas também tiverem sido excluídas, deverão ser igualmente devolvidas ao banco de dados. Essa inclusão pode ser propagada em cascata para outras entidades.
- `Planejar informações estatísticas` – Em alguns casos, informações antigas são necessárias apenas para a tomada de decisões. Neste caso, muitas vezes deseja-se apenas dados resultantes de cálculos ou estatísticas sobre as informações, como totais, contagens, médias,.... Assim pode ser conveniente manter no banco de dados estas informações compiladas e eliminar as informações usadas na compilação.

3.3.5 entidade isolada e entidade sem atributos

Uma entidade isolada é uma entidade que não apresenta relacionamento com outras entidades. Em princípio, entidades isoladas não estão incorretas. Uma entidade que muitas vezes aparece isolada é aquela que modela a organização na qual o sistema implementado pelo banco de dados está embutida. Tomemos como exemplo o banco de dados de uma universidade que aparece na Figura 2.13. A entidade *UNIVERSIDADE* pode ser necessária, caso se deseje manter no banco de dados alguns atributos da universidade. Mesmo assim, o modelo não deveria conter o relacionamento desta entidade com outras, como *ALUNO* ou *CURSO*. Como o banco de dados armazena informações referentes a uma única universidade, não é necessário informar no banco de dados em que universidade o aluno está inscrito ou a qual universidade o curso pertence.

Na prática a ocorrência de entidades isoladas em modelos é rara e por isso deve ser investigada em detalhes, para verificar se não foram esquecidos relacionamentos.

Outra situação que não está incorreta, mas deve ser investigada, é a de uma entidade sem atributos.

3.4 → estabelecimento de padrões

Modelos de dados são usados para a comunicação entre as pessoas da organização (usuários, analistas, programadores,...) e até mesmo para a comunicação com programas (ferramentas CASE, geradores de código,...). Assim, ao usar modelagem de dados, é necessário estabelecer padrões de confecção de modelos. Infelizmente, na prática e na literatura aparecem muitas versões, que distinguem-se umas das outras não só na representação gráfica, isto é, em sua sintaxe, mas também na semântica. Nesta seção, discutimos algumas variantes da abordagem ER e como garantir a utilização de uma variante escolhida.

3.4.1 variantes da abordagem ER

Quando de seu surgimento na literatura, a abordagem ER não era ainda suportada por ferramentas em computador de edição e manipulação, como as

ferramentas CASE hoje existentes. Com isto, cada autor de livro sobre o assunto, bem como cada usuário da abordagem tinha total liberdade para escolher uma representação gráfica e até mesmo uma semântica para a abordagem. A conseqüência é que hoje observa-se uma grande variedade de abordagens que levam o título de entidade-relacionamento. Até este ponto do livro, apresentamos a abordagem ER na forma que aparece mais freqüentemente na literatura. Esta notação é às vezes referida como notação tipo "Chen" pois, com algumas extensões, segue a notação proposta por Peter Chen em seu primeiro artigo sobre a abordagem.

Nas próximas seções, são apresentadas a notação *engenharia de informações*, uma notação para modelos ER muito usada na prática, a *abordagem UML*, uma abordagem usada para modelar um *software* orientado a objeto, mas que pode ser adaptada para a modelagem conceitual de dados e uma notação para cardinalidade, usada freqüentemente em textos europeus.

■ notação engenharia de informações

A engenharia de informações é um método de desenvolvimento de sistemas de informação proposto no início da década de 1980 por Martin e Finkelstein. Este método teve importância considerável na prática, comprovada pela existência de ferramentas CASE que o suportam. A característica que distingue o método de outros métodos de desenvolvimento (método estruturado, método orientado a objetos,...) é a ênfase que dá à modelagem de dados. Apesar de a engenharia de informações, como método de desenvolvimento de *software*, ter perdido importância em relação aos métodos baseados na orientação a objetos, a notação para modelagem de dados introduzida no método continua sendo utilizada em muitas ferramentas CASE.

A Figura 3.21 apresenta um exemplo de um DER na notação Chen até aqui empregada e o DER correspondente na notação Engenharia de Informações. Os pontos que caracterizam a notação engenharia de informações são os seguintes:

- Os atributos são anotados dentro dos retângulos representativos das entidades. Esta notação para atributos é hoje a mais aceita e usada em quase todas as variantes de notações.
- Chaves primárias são indicadas através de algum tipo de adorno no diagrama. No exemplo da Figura 3.21, as chaves primárias aparecem sublinhadas.

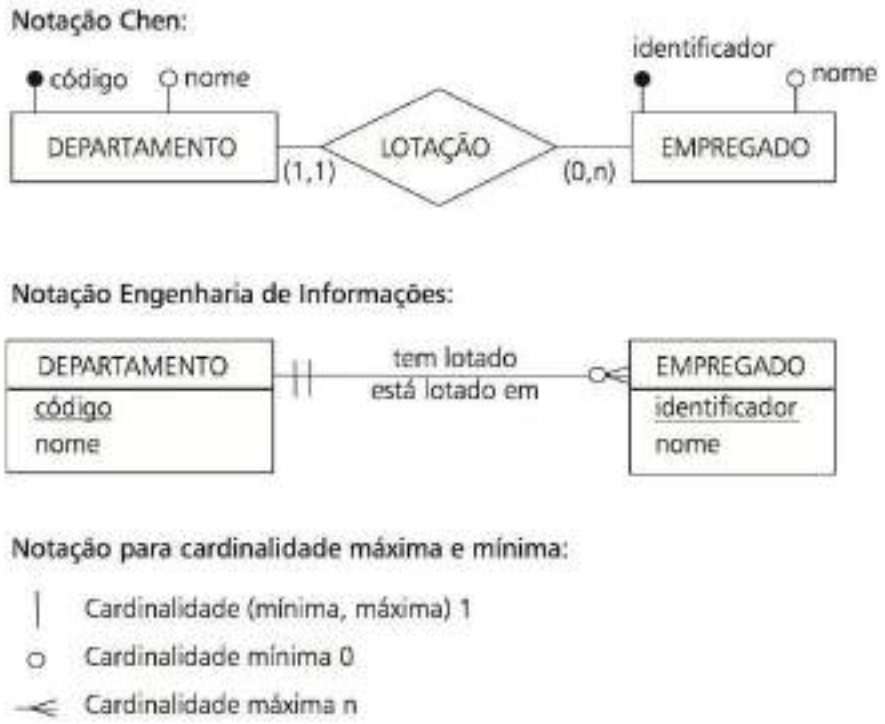


Figura 3.21 Comparação entre um diagrama ER na notação Chen e na notação engenharia de informações.

- Relacionamentos são representados apenas por uma linha que liga os símbolos representativos das entidades associadas. Isto tem as seguintes conseqüências:
  - A notação admite apenas relacionamentos *binários*, já que uma linha conecta apenas duas entidades. Relacionamentos ternários ou de grau maior são modelados através de uma entidade, que é associada através de relacionamentos binários, a cada uma das entidades que participam do relacionamento ternário.
  - Atributos aparecem exclusivamente em entidades. Com isto, objetos que seriam modelados como relacionamentos **n:n** na notação Chen tendem a ser modelados como entidades na notação engenharia de informações.
- A denominação de um relacionamento é escrita na forma de verbos em ambas as direções de leitura (*DEPARTAMENTO* tem lotado *EMPREGADO*, *EMPREGADO* está lotado em *DEPARTAMENTO*).
- A notação para cardinalidade máxima e mínima é gráfica. O símbolo mais próximo do retângulo representativo da entidade corresponde à

Figura 3.22 Modelo ER da Figura 2.38 na notação engenharia de informações.

cardinalidade máxima, o mais distante corresponde à cardinalidade mínima.

- Em algumas variantes, a generalização/especialização é chamada de subconjunto (subtipo) de entidades e é representada através do aninhamento dos símbolos de entidade conforme mostra a Figura 3.22 (trata-se do mesmo modelo ER apresentado na Figura 2.38).

Não há um padrão para diagramas ER. Assim, mesmo a notação engenharia de informações aqui apresentada é adotada de diferentes maneiras por autores de livros e de ferramentas CASE.

abordagem UML

A *UML* (*Unified Modeling Language* ou *Linguagem Unificada de Modelagem*) reúne um grande conjunto de notações destinadas à modelagem de *software* sob vários aspectos e em diferentes níveis de abstração, diferindo da abordagem ER.

Entretanto, com algumas ressalvas comentadas a seguir, para a etapa da modelagem conceitual pode ser utilizado um diagrama conhecido como *diagrama de classes*.

A terminologia usada na UML, originária da programação orientada a objeto, é diferente da terminologia da abordagem ER. A Tabela 3.1 apresenta a correspondência entre os conceitos da abordagem ER e os da abordagem UML.

Tabela 3.1 Correspondência entre os conceitos da abordagem ER e os da abordagem UML

ER	UML
entidade	classe
relacionamento	associação
cardinalidade	multiplicidade
generalização/ especialização	generalização

Do ponto de vista diagramático, o diagrama de classes lembra os modelos da engenharia de informações. A Figura 3.23 apresenta um diagrama de classes na notação UML. Este diagrama representa a mesma realidade modelada pelo diagrama ER da Figura 2.38. As principais características da abordagem UML são as seguintes.

- Atributos são anotados como na engenharia de informações.
- Relacionamentos, chamados *associações* em UML, são representados por linhas, como na engenharia de informações.

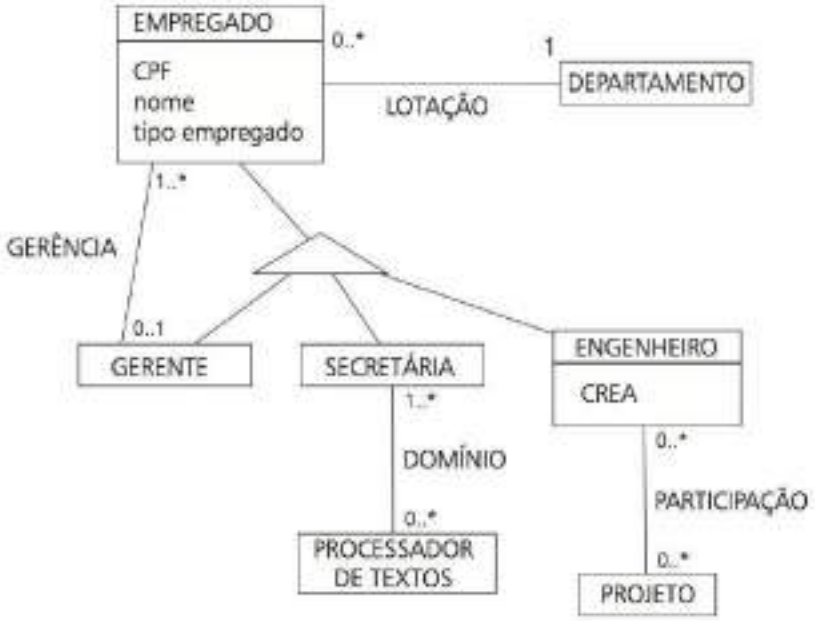


Figura 3.23 Modelo da Figura 2.38 na notação UML.

- A cardinalidade, chamada de *multiplicidade*, é anotada na forma de um par mínimo..máximo. Quando os valores da cardinalidade mínima e máxima são iguais, apenas um valor é indicado. Exemplificando, uma restrição de cardinalidade que, na notação usada no livro, é anotada como **(1,1)**, em UML é anotada simplesmente como **1**. A cardinalidade máxima **n** é representada por um asterisco (*).
- Não existe o conceito de *identificador*, como conhecemos na abordagem ER. O diagrama de classes foi concebido para a modelagem de objetos. Como objetos, na programação orientada a objeto, já possuem um identificador implícito e não manipulável pelo usuário, o conceito de identificador explícito, como tratado na abordagem ER, não aparece na abordagem UML.
- A UML inclui outros conceitos importantes em orientação a objetos, mas irrelevantes para o projeto de banco de dados. Por exemplo, no diagrama de classes, é possível especificar os *métodos* das classes.

Como dito acima, a UML não tem um conceito equivalente ao de identificador de entidade ou de relacionamento na abordagem ER. Entretanto, como veremos nos próximos capítulos, este conceito é central para o projeto de bases de dados relacionais. Como a UML é uma abordagem extensível (permite que novos conceitos sejam adicionados), é possível adicionar o conceito de identificador à abordagem. Neste caso, se uma ferramenta CASE for usada na modelagem, deve-se verificar se ela é capaz de tratar estas extensões.

notação européia para cardinalidades

Nos livros-texto e nas ferramentas CASE européias, aparece uma notação para modelos ER, cuja característica é a forma de anotar e interpretar as cardinalidades de relacionamentos. Esta notação provavelmente é originária de *MERISE*, uma metodologia de desenvolvimento de sistemas usada durante algum tempo na França.

A Figura 3.24 apresenta um DER na notação Chen e o DER correspondente na notação MERISE. Como se observa, além da modificação cosmética de substituir o losango representativo de um relacionamento por uma elipse, a posição em que são indicadas as cardinalidades mínima e máxima mudou. O motivo é que a interpretação dada à cardinalidade em MERISE é diferente da usual.

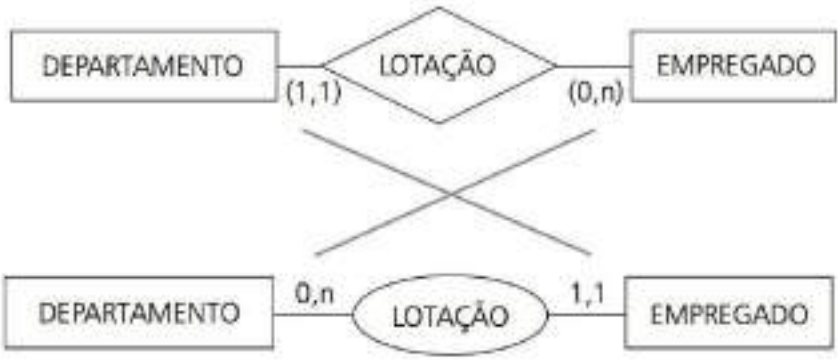


Figura 3.24 Notação Chen e correspondente notação MERISE.

Na interpretação usual de cardinalidade, ela indica quantas ocorrências de entidade (no mínimo e no máximo) podem estar associadas a uma ocorrência de determinada entidade. Esta interpretação é chamada de semântica *associativa*.

Em MERISE, usa-se a semântica *participativa*. Nesta, a cardinalidade indica quantas vezes uma ocorrência de entidade *participa* de um relacionamento. Exemplificando, na figura acima, na notação Chen, a cardinalidade (1,1) representa o fato de um empregado estar vinculado a no mínimo um e no máximo um departamento. Já no diagrama em MERISE, a anotação 1,1 indica que uma ocorrência da entidade *EMPREGADO* participa no mínimo uma e no máximo uma vez do relacionamento *LOTAÇÃO*.

Para relacionamentos binários, para passar da notação adotada no livro para a notação européia, basta trocar as cardinalidades de lado. Entretanto, para relacionamentos de grau maior que dois (ternários,...), com a mudança da semântica da cardinalidade, não é possível fazer uma tradução direta de uma notação para a outra. Para um exemplo, veja o Exercício 5 deste capítulo.

3.4.2 uso de ferramentas de modelagem

Na prática, não é aceitável que o diagrama ER seja construído manualmente. A criação de um DER à mão é muito trabalhosa pois, durante o processo de modelagem, as revisões são freqüentes. Além disso, quando ocorrem alterações do modelo durante a sua vida útil, diagramas feitos à mão dificilmente serão atualizados. Portanto, é recomendável que, desde o início da confecção do modelo ER, seja usada uma ferramenta em computador para apoio à mo-

delagem. Podem ser consideradas duas alternativas: o uso de uma ferramenta CASE ou o uso de programas de propósito geral.

■ uso de uma ferramenta CASE

O *software* ideal para acompanhar o projeto de um banco de dados é uma ferramenta CASE (*Computer Aided Software Engineering*). Uma ferramenta CASE pode apoiar o desenvolvimento de um banco de dados, tanto na fase de modelagem, quanto na fase de projeto do banco de dados.

Do ponto de vista da *modelagem*, o mínimo que se deve exigir de uma ferramenta CASE é:

- `Capacidade de edição diagramática` – A ferramenta CASE deve oferecer uma interface gráfica poderosa e fácil de usar, que permita construir o DER diretamente no computador. A modificação do DER (inclusão/eliminação/movimentação de símbolos) deve ser simplificada.
- `Dicionário de dados` – A ferramenta deve possuir um *dicionário de dados*, isto é, uma pequeno banco de dados, no qual toda descrição do DER está armazenada.
- `Integração entre o diagrama ER e o dicionário de dados` – O dicionário de dados deve ser integrado ao DER. A partir do DER deve ser possível visualizar e editar as entradas do dicionário de dados correspondentes a elementos selecionados do DER.

Do ponto de vista do *projeto*, a ferramenta deve suportar as etapas de projeto lógico e de engenharia reversa, descritas no capítulo 5.

■ uso de programas de propósito geral

Quando o orçamento é pequeno, pode ser difícil obter os recursos necessários à aquisição de uma ferramenta CASE. Neste caso, pode-se usar programas de propósito geral para editar o DER e montar o dicionário de dados:

- `Edição do DER` – Para editar o DER pode-se utilizar um programa de desenho de propósito geral. Há programas de desenho que permitem editar grafos e trabalham com os conceitos de nó e de arco. Nestes programas, quando o usuário mover um nó do diagrama, todos os arcos ligados a este nó são também movimentados. Este tipo de programa é particularmente adequado para a edição de diagramas ER.

- `Dicionário de dados` – Para construir o dicionário de dados pode-se utilizar um processador de textos, uma planilha eletrônica ou um banco de dados (esta é a melhor opção). A escolha provavelmente irá recair sobre o programa que for mais conhecido e dominado na organização em questão.

3.5 → estratégias de modelagem

O processo de construção de um modelo é um processo incremental, isto é, um modelo de um sistema não é construído em um único passo, mas em muitos pequenos passos, muitas pequenas transformações, desde o modelo inicial, até o modelo completo. Gradativamente, o modelo vai sendo enriquecido com novos conceitos e estes vão sendo ligados aos existentes ou os existentes vão sendo aperfeiçoados. Uma *estratégia* de modelagem ER é uma seqüência de passos (uma "receita de bolo") de transformação de modelos. Estudando o processo de construção de modelos ER, vários autores propuseram diferentes estratégias, que apresentamos a seguir.

Na prática, observa-se que nenhuma das estratégias propostas na literatura é universalmente aceita. Normalmente, é aplicada uma combinação das diversas estratégias de modelagem. Isso é compreensível, se considerarmos que o processo de modelagem é um processo de aprendizagem. Quando construímos o modelo de um sistema, estamos aprendendo fatos sobre aquele sistema. A seqüência de idéias que se tem durante um processo de aprendizagem é dificilmente controlável por uma estratégia.

Para definir qual a estratégia a ser usada na construção de um modelo ER, deve-se identificar qual a fonte de informações principal para o processo de modelagem. São duas as fontes de informação que iremos considerar: descrições de dados existentes e o conhecimento que as pessoas possuem sobre o sistema.

3.5.1 partindo de descrições de dados existentes

Uma possível fonte de informações para o processo de modelagem de dados são descrições de dados já existentes.

Exemplificando, esta situação ocorre quando deseja-se obter um modelo de dados para um sistema em computador existente. Neste caso, usa-se como

descrição dos dados as descrições dos arquivos utilizados pelo sistema em computador. Este caso é conhecido por *engenharia reversa*, pois objetiva obter uma especificação (o modelo) a partir de um produto existente (o sistema em computador).

Outro exemplo do uso de descrições de dados já existentes é quando são utilizadas descrições dos documentos (pastas, fichas, documentos fiscais,...) usados em um sistema não automatizado.

A construção de um modelo partindo de descrições de dados já existentes dá-se de acordo com a estratégia *ascendente* (*bottom-up* – de baixo para cima). Esta estratégia consta de partir de conceitos mais detalhados ("embaixo" em termos de níveis de abstração) e abstrair gradativamente. O processo de modelagem inicia com a identificação de atributos (conceitos mais detalhados). A partir daí, os atributos são agregados em entidades e as entidades são relacionadas e generalizadas. Essa forma de trabalhar é adequada quando já dispomos dos atributos. Normalmente, isso ocorre quando já temos uma definição dos dados que deverão estar armazenados no banco de dados. Iremos tratar dessa estratégia em detalhes nos capítulos 5 e 6.

3.5.2 partindo do conhecimento de pessoas

Outra fonte de informações para a construção de modelos ER são os conhecimentos que pessoas possuem sobre o sistema sendo modelado. Este é o caso, quando novos sistemas, para os quais não existem descrições de dados, estão sendo concebidos. Para este caso, podem ser aplicadas duas estratégias, a estratégia *descendente* ("*top-down*") e a estratégia "*inside-out*".

■ estratégia descendente

A estratégia descendente consta de partir de conceitos mais abstratos ("de cima") e ir gradativamente refinando estes conceitos em conceitos mais detalhados. Ou seja, segue-se o caminho inverso da estratégia ascendente. Aqui o processo de modelagem inicia com a identificação de entidades genéricas (conceitos mais abstratos). A partir daí, são definidos os atributos das entidades, seus relacionamentos, os atributos dos relacionamentos e as especializações das entidades.

Uma seqüência de passos para obter um modelo usando a estratégia descendente é a mostrada abaixo:

1 *Modelagem superficial* – Nesta primeira etapa, é construído um DER pouco detalhado (faltando domínios dos atributos e cardinalidades mínimas de relacionamentos) na seguinte sequência:
   - Enumeração das *entidades*.
   - Identificação dos *relacionamentos* e *hierarquias* de generalização/especialização entre as entidades. Para cada relacionamento identifica-se a *cardinalidade máxima*.
   - Determinação dos *atributos* de entidades e relacionamentos.
   - Determinação dos *identificadores* de entidades e relacionamentos.
   - Verificação do banco de dados quanto ao *aspecto temporal*.

2 *Modelagem detalhada* – Nesta etapa, completa-se o modelo com os domínios dos atributos e cardinalidades mínimas dos relacionamentos:
   - Adiciona-se os domínios dos atributos.
   - Define-se as cardinalidades mínimas dos relacionamentos. É preferível deixar estas cardinalidades por último, já que exigem conhecimento mais detalhado sobre o sistema, inclusive sobre as transações.
   - Define-se as demais restrições de integridade, que não podem ser representadas pelo modelo ER.

3 *Validação do modelo*
   - Procura-se construções redundantes ou deriváveis a partir de outras no modelo.
   - Valida-se o modelo com o usuário.

Em qualquer destes passos, é possível retornar aos passos anteriores. Exemplificando, durante a identificação de atributos é possível que sejam identificadas novas entidades, o que faria com que o processo retornasse ao primeiro passo.

■ estratégia *inside-out*

A estratégia *inside-out* (de dentro para fora) consta de partir de conceitos considerados mais importantes (centrais, parte-se "de dentro") e ir gradativamente adicionando conceitos periféricos a eles relacionados (ir "para fora"). Aqui, o processo inicia com a identificação de uma entidade particularmente impor-

tante no modelo e que, supõe-se, estará relacionada a muitas outras entidades. A partir daí, são procurados atributos, entidades relacionadas, generalizações e especializações da entidade em foco, e assim recursivamente até obter-se o modelo completo. A denominação da estratégia provém da idéia de que entidades importantes em um modelo e relacionadas a muitos outras são usualmente desenhadas no centro do diagrama, a fim de evitar o cruzamento de linhas.

Um exemplo da aplicação desta estratégia é mostrado na Figura 3.25, com a modelagem de uma parte de um sistema de recursos humanos. As linhas em tons de cinza indicam os passos da construção do modelo. No caso, considerou-se como conceito central e ponto de partida da modelagem a entidade *EMPREGADO*. Após, procurou-se entidades relacionadas a *EMPREGADO*, tendo sido identificada a entidade *DEPARTAMENTO*. No próximo passo, foram identificados os atributos e especializações de *EMPREGADO* e finalmente foram identificados os atributos e as entidades relacionados às especializações de *EMPREGADO*.

Os passos que ocorrem no processo *inside-out* são os mesmos que ocorrem no processo *top-down* ocorrendo, entretanto, uma iteração muito maior entre os passos 1.a, 1.b e 1.c.

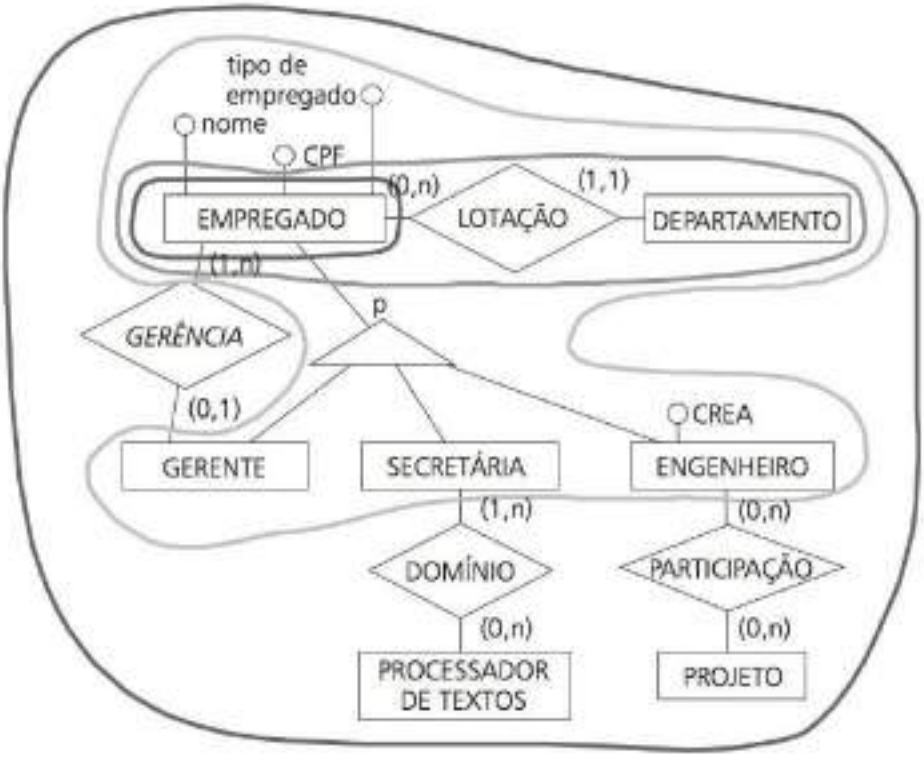


Figura 3.25 Aplicação da estratégia "de dentro para fora".

3.6 → exercícios

exercício 1 A Figura 3.26 apresenta um relacionamento que associa um produto de uma indústria com seus componentes (em inglês, *bill-of-materials*). Cada produto pode ter diversos componentes e cada componente pode aparecer dentro de diversos produtos. Assim trata-se de um relacionamento do tipo **n:n**. Uma restrição que deve ser imposta sobre o banco de dados em questão é a de que um produto não pode aparecer na lista de seus componentes. Pergunta-se:

- O modelo apresentado na figura contém esta restrição?
- Caso negativo, é possível alterar o modelo em questão para incluir esta restrição, se considerarmos que o nível de profundidade da hierarquia de composição de cada produto não excede três (tem-se apenas produtos prontos, produtos semi-acabados e matérias-primas)? Caso afirmativo, apresente a solução.
- É possível estender a solução do quesito anterior para uma hierarquia não limitada de níveis de composição?

exercício 2 Deseja-se modelar os clientes de uma organização. Cada cliente possui um identificador, um nome, um endereço e um país. Discuta as vantagens e desvantagens das duas alternativas de modelagem de país:

- Como atributo da entidade cliente.
- Como entidade relacionada a cliente.

exercício 3 A Figura 3.27 apresenta uma entidade e seus respectivos atributos, muitos deles opcionais e um multivalorado. Considere que há dois tipos de clientes, pessoas físicas e pessoas jurídicas. Pessoas físicas possuem código, CPF,

Figura 3.26 Composição de produtos.

nome, sexo (opcional), data de nascimento (opcional) e telefones (opcionais). Pessoas jurídicas possuem código, CNPJ, razão social e telefones (opcionais). Apresente um diagrama ER que modele mais precisamente esta realidade. Explique no que seu diagrama é mais preciso do que o mostrado na Figura 3.27.

exercício 4 Transforme o modelo ER resultante do exercício anterior da notação Chen para a notação da Engenharia de Informações.

exercício 5 Transforme o modelo ER apresentado na Figura 2.11 para a notação européia de cardinalidades. O novo modelo contém as mesmas restrições de integridade especificadas pelo modelo da Figura 2.11? Caso negativo, comente por que não.

exercício 6 No modelo da Figura 3.10, considere que, para cada servidor, seja necessário armazenar também seu CPF e seu número de inscrição no PASEP. Observe que tanto o CPF quanto o número do PASEP são informações que não dependem de cargos individuais do servidor. Como o modelo da Figura 3.10 deve ser modificado para incluir estas informações?

exercício 7 Estudo de caso – *Administradora de imóveis.*

Construa um diagrama ER (apenas entidades e relacionamentos com cardinalidades máximas) para a administradora de imóveis descrita abaixo.

A administradora trabalha tanto com administração de condomínios, quanto com a administração de aluguéis.

Uma entrevista com o gerente da administradora resultou nas seguintes informações:

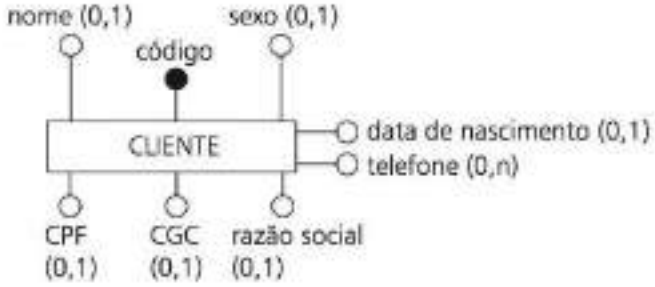


Figura 3.27 Cliente com atributos.

- A administradora administra condomínios formados por unidades condominiais.
- Cada unidade condominial é de propriedade de uma ou mais pessoas. Uma pessoa pode possuir diversas unidades.
- Cada unidade pode estar alugada para no máximo uma pessoa. Uma pessoa pode alugar diversas unidades.

exercício 8 Estudo de caso – *Locadora de mídias antigas* (adaptado do material de um curso de modelagem de dados da Oracle).

Uma pequena empresa especializada na locação de mídias antigas de áudio e vídeo, com fitas VHS e discos de vinil para aficionados da área. Esta locadora deseja controlar, por computador, o empréstimo de seu acervo de aproximadamente 2.000 fitas de vídeo.

Cada fita possui um número. Para cada filme, é necessário saber seu título e sua categoria (comédia, drama, aventura,...). Cada filme recebe um identificador próprio. Para cada fita é controlado que filme ela contém. Para cada filme há pelo menos uma fita, e cada fita contém somente um filme. Alguns poucos filmes necessitam duas fitas.

Os clientes podem desejar encontrar os filmes estrelados pelo seu ator predileto. Por isso, é necessário manter a informação dos atores que estrelam em cada filme. Nem todo filme possui estrelas. Para cada ator os clientes às vezes desejam saber o nome real, bem como a data de nascimento.

A locadora possui muitos clientes cadastrados. Somente clientes cadastrados podem alugar fitas. Para cada cliente é necessário saber seu pré-nome e seu sobrenome, seu telefone e seu endereço. Além disso, cada cliente recebe um número de associado.

Finalmente, desejamos saber que fitas cada cliente tem emprestadas. Um cliente pode ter várias fitas em um instante no tempo. Não são mantidos registros históricos de aluguéis.

exercício 9 Estudo de caso – *Sistema de reserva de passagens aéreas.*

O objetivo do trabalho é projetar um sistema de reservas para uma companhia de aviação. O sistema contará com um banco de dados central, que será

acessado por aplicações-clientes, rodando tanto dentro da própria companhia, quanto fora dela.

A transação central do sistema é a reserva. Uma reserva é identificada por um código gerado pelo sistema em computador. A reserva é feita para um único passageiro, do qual se conhece apenas o nome. A reserva compreende um conjunto de trechos de vôos, que acontecerão em determinada data/hora. Para cada trecho, a reserva é feita em uma classe (econômica, executiva, etc.).

Um vôo é identificado por um código e possui uma origem e um destino. Por exemplo, o vôo 595 sai de Porto Alegre com destino a São Paulo. Um vôo é composto de vários trechos, correspondendo às escalas intermediárias do vôo. Por exemplo, o vôo 595 é composto de dois trechos, um de Porto Alegre a Londrina, o outro de Londrina a São Paulo. Cabe salientar que há cidades que são servidas por vários aeroportos. Por isso, é importante informar ao passageiro que faz a reserva em qual aeroporto o vôo passa.

Às vezes, ao fazer a reserva, os clientes querem saber qual é o tipo de aeronave que será utilizada em determinado trecho do vôo. Alguns poucos vôos, principalmente internacionais, têm troca de aeronave em determinadas escalas.

Nem todos os vôos operam em todos os dias da semana. Inclusive, certos vôos têm pequenas mudanças de horário em certos dias da semana.

Cada reserva possui um prazo de validade. Caso os bilhetes não tenham sido emitidos até esgotar-se o prazo da reserva, a mesma é cancelada. Reservas podem ser prorrogadas.

Como o *check-in* de todos os vôos está informatizado, a companhia possibilita a reserva de assento para o passageiro. Reservas de assento podem ser feitas com até três meses de antecedência.

Além de efetivar reservas, o sistema deve servir para vários tipos de consultas que os clientes podem querer fazer:

- possibilidades de viagem de uma cidade ou de um aeroporto para outro;
- o mesmo, mas restrito a determinados dias da semana;
- horários de chegada ou de saída em determinados vôos;
- disponibilidade de vagas em um trecho de vôo;
- disponibilidade de determinados assentos em um trecho de vôo.

exercício 10 Estudo de caso – *Sistema para locadora de veículos.*

O objetivo deste estudo de caso é construir um modelo ER para o banco de dados de uma empresa de locação de veículos. A empresa em questão aluga automóveis, camionetas de passageiros e camionetas de carga.

Ela atende a dois mercados, o das pessoas físicas e o das pessoas jurídicas. Para acelerar o atendimento, é importante conhecer os dados de clientes que já tenham usado a locadora no passado. Para cada pessoa física, é necessário conhecer seu nome, sexo, data de nascimento, endereço e CPF. Já para as pessoas jurídicas, é necessário conhecer seu nome, CNPJ, inscrição estadual e endereço. Os clientes são identificados por um código interno à locadora.

A empresa tem uma grande rede de filiais, espalhada pelo sul do país. Em um momento no tempo, um veículo encontra-se sob responsabilidade de uma filial. Entre-tanto, como veículos podem ser alugados para viagens em um sentido somente, eles podem mudar de filial. Um veículo é identificado pela sua placa. Além disso, é necessário conhecer o número do chassi, o número do motor, o tipo de veículo e a cor de cada veículo.

O sistema em computador deverá registrar:

- os veículos disponíveis em determinada filial na data corrente;
- as reservas para veículos em uma filial, com previsão de que veículos estarão disponíveis em uma data futura;
- os veículos presentemente alugados pela filial, o ponto de entrega (caso seja diferente do de locação) e data de entrega prevista.

Os veículos são classificados por uma tabela de tipos. Por exemplo, P3 corresponde a automóveis pequenos, de quatro portas e com ar-condicionado e G4 a grandes automóveis de luxo. As reservas não são feitas para uma marca ou modelo de veículo, mas para um tipo de veículo.

Para tipos de automóveis, os clientes desejam saber o tamanho, classificado em pequeno, médio e grande, o número de passageiros, o número de portas, bem como se possui os seguintes acessórios: ar-condicionado, rádio, CD, MP3, direção hidráulica e câmbio automático. Para tipos de camionetas de passageiros, as informações são as mesmas que para automóveis. Já para tipos de camionetas de carga, as informações acima não são relevantes. Neste caso, os clientes desejam saber a capacidade de carga da camioneta.

Para cada tipo de veículo, há um determinado número de horas necessário para limpeza e revisão de entrega, entre uma reserva e outra.

Além disso, o sistema deve programar as revisões dos veículos, impedindo que sejam reservados quando há revisões pendentes. Esta programação é feita com base em um conjunto de parâmetros que são a quilometragem atual do veículo, a quilometragem média diária de um veículo do tipo, bem como em uma tabela de revisões do tipo de veículo.

A seguradora que segura os veículos exige que, para cada veículo alugado, seja mantida a identificação do motorista, o número de sua habilitação e data de vencimento da mesma. A habilitação não pode vencer dentro do prazo da locação.

exercício 11 Estudo de caso – *Sistema de preparação de congressos da IFIP.* (Esta é a adaptação de um conhecido estudo de caso proposto em um congresso de um grupo de trabalho da IFIP que tinha por objetivo comparar diferentes metodologias de desenvolvimento de sistemas de informação. O estudo de caso foi concebido como estudo de caso padrão em que cada metodologia a ser comparada deveria ser aplicada.)

Deseja-se construir um sistema em computador para apoiar a organização de congressos de trabalho da IFIP (Federação Internacional de Processamento de Informações). A IFIP é uma sociedade que congrega as sociedades nacionais de Informática de diversos países.

Dentro da IFIP, atuam grupos de trabalho (GT). Cada GT trata de um determinado assunto. Cada GT recebe um código e é denominado segundo o assunto de que trata. Um dos objetivos de um GT é o de promover de forma regular congressos de trabalho.

Um congresso de trabalho é um encontro com número limitado de participantes (50-100) dos quais espera-se participação ativa. No congresso, são apresentados e discutidos trabalhos originais dos participantes. Para possibilitar que os participantes obtenham financiamento para sua ida ao congresso, é importante assegurar que a seleção dos trabalhos apresentados seja feita de maneira imparcial, por especialistas reconhecidos na área do congresso. O congresso é identificado por um código e possui um nome, um local e uma data de realização.

A IFIP recomenda que a preparação do congresso seja feita por dois comitês, o comitê organizador e o comitê de programa, montados especialmente para cada congresso.

O *comitê de programa* (CP) executa as seguintes funções:

- O CP está encarregado da recepção de artigos submetidos, controlando se eles estão dentro do prazo. Um artigo é um trabalho que um ou mais autores pretendem apresentar no congresso e que será publicado em seus anais. Os anais são normalmente publicados por editoras comerciais. Por isso, os artigos devem ser originais, isto é, cada artigo somente pode ser submetido a um congresso. Os artigos são numerados seqüencialmente dentro de um congresso, à medida que vão sendo recebidos. É necessário saber o título do artigo, bem como seus autores.
- Antes do início da avaliação dos artigos, o CP deve montar um corpo de avaliadores externos, que irá ler cada artigo e emitir uma opinião sobre o artigo. Normalmente, mas não necessariamente, avaliadores externos são pessoas que já participaram de congressos anteriores sobre o mesmo tema. Somente será cadastrado como avaliador de um congresso aquele que, após convidado para tal, o aceitar explicitamente.
- À medida que os artigos forem sendo recebidos, eles devem ser distribuídos aos avaliadores previamente cadastrados. Cada artigo deve ser distribuído a pelo menos três avaliadores. Um avaliador nunca deve estar com mais que um artigo em um instante no tempo.
- O CP está encarregado de fazer o julgamento dos artigos. No julgamento, o CP decide quais artigos serão aceitos e quais serão rejeitados, com base nos pareceres dos avaliadores. O julgamento acontece um certo tempo depois do fim do prazo de recepção de artigos.
- Após o julgamento, o CP deve montar o programa do congresso, isto é, agrupar os artigos aceitos em sessões e escolher um moderador para cada sessão. Uma sessão é um grupo de artigos que tratam de um assunto comum e que são apresentados em um determinado horário. Por tratar-se de um congresso de trabalho, não há sessões em paralelo, isto é, em um congresso, em um momento no tempo, não pode ocorrer mais que uma sessão. Assim, a sessão é identificada pelo congresso e pelo horário em que ocorre.
- Além das tarefas acima, o CP está encarregado de toda a comunicação com os autores e com os avaliadores.

O *comitê organizador* (CO) está encarregado de toda parte financeira e organização local do congresso. Para fins do estudo de caso, vamos considerar apenas as seguintes atividades:

- Com boa antecedência, o CO deve emitir as chamadas de trabalho do congresso. A chamada de trabalhos é um texto que divulga o congresso e que convida as pessoas a submeter artigos ao congresso. Esta chamada é amplamente divulgada em listas eletrônicas, periódicos, etc. Além disso, ela deve ser enviada pessoalmente, por correio eletrônico, às seguintes pessoas:
  - membros dos GTs que promovem o congresso;
  - pessoas que, de alguma forma, já participaram de congressos promovidos pelos GTs que promovem o congresso e
  - pessoas sugeridas para tal pelos membros dos GTs que promovem o congresso.

Neste envio, devem ser evitadas redundâncias.

- Após montado o programa pelo CP, o CO deve emitir os convites para participação no congresso. Por tratar-se de congressos de trabalho, apenas pessoas convidadas podem inscrever-se. São convidados:
  - os autores de trabalhos submetidos ao congresso,
  - os avaliadores externos do congresso,
  - os membros do CO e do CP do congresso, e
  - os membros dos GTs que promovem o congresso.
- Finalmente, cabe ao CO executar as inscrições dos convidados que aceitarem o convite, controlando o prazo de inscrição e verificando se a pessoa é realmente convidada.

Observar que uma mesma pessoa pode atuar em muitas funções, inclusive em um mesmo congresso. A única restrição importante é que uma pessoa não pode avaliar o seu próprio artigo.

Como na preparação do congresso há considerável comunicação entre as diversas pessoas envolvidas, é necessário manter sempre atualizados os dados de acesso à pessoa, como seu endereço, seu telefone, seu número de fax e principalmente seu endereço de correio eletrônico.

O objetivo do exercício é construir um modelo ER de um banco de dados que será compartilhado via Internet entre os diversos comitês dos congressos em

organização pela IFIP. Neste banco de dados, estarão não só as informações sobre os congressos em organização, mas também sobre os do passado.

exercício 12 Estudo de caso – *Sistema de almoxarifado*. (O enunciado deste trabalho foi adaptado de um livro de Peter Coad sobre projeto orientado a objetos.)

O almoxarifado pertence a um grupo de empresas do ramo industrial e serve para estocar peças destinadas às várias empresas do grupo. Cada empresa do grupo é considerada um cliente do almoxarifado.

O almoxarifado está organizado em corredores. Cada corredor possui vários receptáculos para peças (um receptáculo é uma bacia retangular de material plástico). Os receptáculos são todos do mesmo tamanho. Os corredores são numerados e os receptáculos são numerados por corredor. Por exemplo, o receptáculo 2-10 é o décimo receptáculo do segundo corredor.

Em uma das extremidades do almoxarifado encontra-se o setor de recepção de peças. Lá chegam as peças entregues pelos fornecedores. Quando ocorre a chegada de peças, a primeira atividade é registrar na ordem de compra a chegada das peças. Uma cópia de toda ordem de compra é sempre enviada ao setor de recepção. Assim, neste setor sempre sabe-se quais as peças que estão por ser entregues. As ordens de compra são geradas no setor de compras e apenas repassadas ao almoxarifado.

Uma entrega corresponde sempre a uma ordem de compra. Entretanto, são admitidas entregas parciais, isto é, entregas que não completam a ordem de compra. Em uma entrega podem ser entregues diferentes quantidades de diferentes peças.

As peças recebidas são colocadas sobre um estrado. Este estrado é então levado para o almoxarifado por uma empilhadeira e as peças são distribuídas nos receptáculos. Um estrado pode conter diferentes peças. Para cada peça, procura-se um receptáculo que já contenha unidades da peça em questão e que ainda tenha espaço para a carga chegada. Caso não haja um receptáculo nestas condições, procura-se um receptáculo vazio.

A saída do almoxarifado se dá contra pedidos de clientes. Um pedido pode solicitar vários tipos de peças. Todas as peças que atendem um pedido são

juntadas, embaladas e colocadas em uma rampa de carga (numerada) onde encosta o caminhão do cliente. Não há pedidos pendentes, isto é, os clientes sempre pedem quantidades de peças que há em estoque.

O objetivo do sistema é aumentar o lucro do almoxarifado, ajudando sua equipe a guardar e recuperar itens mais rapidamente e a conhecer as quantidades estocadas.

O almoxarifado é de grande porte e constantemente há várias empilhadeiras circulando por ele tanto para estocar entregas quanto para buscar peças referentes a um pedido.

Outros detalhes do sistema são fornecidos a seguir.

O almoxarifado somente atende empresas. É necessário manter um cadastro de clientes com CGC, nome, endereço e telefone de contato. Para cada peça é necessário conhecer seu UPC (*Universal Product Code*), descrição e número interno à organização.

Para cada entrega, o setor de recepção monta uma lista de distribuição, que instrui o operador sobre que peças, em que quantidade ele deve estocar em que receptáculos.

Para cada pedido, o setor de saída monta uma lista de busca, que instrui o operador sobre que peças, em que quantidade ele deve buscar em que receptáculos.

Em termos de processos, é necessário que o sistema:

- Dê as ordens de distribuição de peças chegadas para cada chegada.
- Dê as ordens para busca para cada pedido.
- Mantenha a quantidade estocada de cada item e de cada receptáculo.
- Informe que peças em que quantidade devem ser estocadas ou buscadas em que receptáculos.

Em termos específicos de transações devem ser consideradas:

- Transações de chegada.
- Registro da chegada de produtos.
- Instruções para estocagem (em que estrado, em que receptáculos).

- Confirmação da estocagem em um receptáculo.
- Transações de saída de produtos.
- Registro de um pedido.
- Geração da lista de busca.
- Confirmação da busca.
- Consolidação de receptáculos (juntar as peças de mesmo tipo de dois receptáculos diferentes).

Em sua primeira versão, o sistema ainda não fará o controle de empilhadeiras ou estrados disponíveis e em uso.

3.7 → leituras recomendadas

Abaixo estão listados alguns livros que apresentam metodologias de desenvolvimento de sistemas de informação e que incluem a etapa de modelagem conceitual. Eles abordam o problema da modelagem ER com grau variável de profundidade.

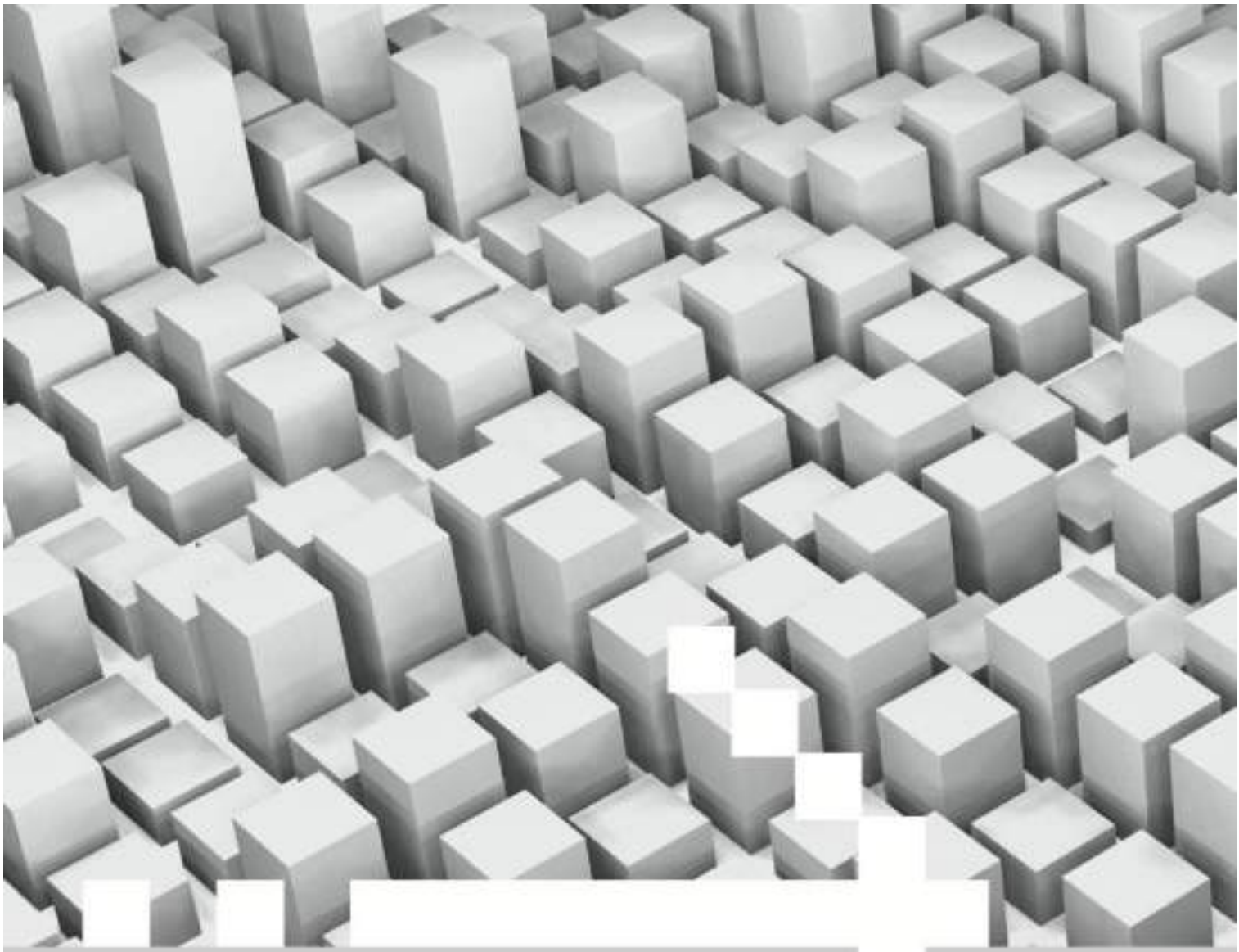
O livro de BATINI; CERI; NAVATHE (1992) é o mais completo sobre o processo de modelagem conceitual. Dedica um capítulo exclusivamente a estratégias para obter o modelo de dados. Este livro é, também, um exemplo do uso da notação européia de cardinalidades.

A notação de Engenharia de Informações pode ser encontrada em algumas obras mais antigas. O livro de MARTIN (1990) descreve a chamada Engenharia de Informações. O tomo II contém alguns conselhos práticos de como modelar. Uma versão gaúcha da Engenharia de Informações é apresentada em KIPPER (1993). É o manual de uma metodologia de desenvolvimento usada durante algum tempo na PROCERGS, Cia. de Processamento de Dados do Rio Grande do Sul. A notação empregada difere da de Martin, sendo semelhante a de Chen. Uma versão mais atual da notação Engenharia de Informações é apresentada por BARKER (1990). Este livro introduz a modelagem de dados sob a perspectiva de um fornecedor de *software* de banco de dados, a Oracle

Corporation. O livro apresenta a notação suportada pela ferramenta CASE da Oracle, semelhante à notação da Engenharia de Informações.

Para a abordagem UML há inúmeras publicações. Uma introdução sucinta é o livro de FOWLER; SCOTT (2000).

O livro TARDIEU; ROCHFELD; COLLETI (1983) apresenta a metodologia Merise, na qual aparece a notação que leva seu nome.



capítulo 4

abordagem relacional

Este capítulo apresenta uma introdução sucinta ao modelo de dados usado nos sistemas de gerência de banco de dados do tipo relacional. É apresentado apenas um conjunto mínimo de conceitos, com o objetivo de permitir que o leitor compreenda o projeto de bancos de dados relacionais, que é discutido nos próximos capítulos.

Especificamente, o capítulo detalha como um banco de dados relacional é organizado (que estruturas de dados são usadas, como elas estão relacionadas), mas não discute como um banco de dados relacional pode ser modificado ou acessado, ou seja, não apresenta as linguagens de manipulação de dados, como SQL. Para maiores detalhes sobre sistemas de BD relacionais, o leitor deve procurar livros específicos (ver leituras recomendadas deste capítulo).

Além dos SGBDs relacionais, existem outros tipos de sistemas no mercado. Entretanto, hoje, há um claro predomínio dos SGBDs relacionais, principalmente fora das plataformas de grande porte. Mesmo nestes ambientes, os SGBD relacionais estão gradativamente substituindo os SGBDs de outras abordagens (hierárquica, rede, sistemas proprietários). Além disso, muitos conceitos usados no projeto de BD, como o conceito de normalização, foram criados em combinação com a abordagem relacional. Por esses motivos, vamos considerar unicamente a abordagem relacional neste livro.

4.1 → composição de um banco de dados relacional

Um banco de dados relacional é composto de *tabelas* ou *relações*. A terminologia *tabela* é mais comum nos produtos comerciais e na prática. Já a terminologia *relação* foi utilizada na literatura original sobre a abordagem relacional (daí a denominação "relacional") e é mais comum na área acadêmica. Neste livro, preferimos adotar a terminologia usada na prática. Entretanto, sempre que apresentarmos um novo conceito, citaremos, entre parênteses, também a terminologia acadêmica.

4.1.1 tabela

Uma tabela é um conjunto não ordenado de *linhas* (tuplas, na terminologia acadêmica). Um exemplo de tabela (tabela `Empregado`) é apresentado na Figura 4.1. No exemplo, a tabela armazena dados sobre empregados de uma organização.

Cada linha é composta por uma série de *campos* (valor de atributo, na terminologia acadêmica). No exemplo, cada linha da tabela corresponde a um

empregado e cada campo é uma informação referente a este empregado (seu código, seu nome,...).

Cada campo é identificado por um *nome de campo* (nome de atributo, na terminologia acadêmica). Na representação gráfica (Figura 4.1) os nomes de campo são representados no cabeçalho da tabela. No exemplo, os nomes de campo são `CodigoEmp`, `Nome`, `CodigoDepto` e `CategFuncional`.

O conjunto de campos homônimos de todas as linhas de uma tabela formam uma *coluna*.

Comparando uma tabela de um banco de dados relacional com um arquivo convencional do sistema de arquivos de um computador, identificam-se as seguintes diferenças:

- As linhas de uma tabela *não têm ordenação*. A ordem de recuperação pelo SGBD é arbitrária, a menos que a instrução de consulta tenha especificado explicitamente uma ordenação. Não é possível referenciar linhas de uma tabela por posição. Já em arquivos convencionais, o programador tem controle sobre a ordem de armazenamento e pode referenciar registros por sua posição relativa dentro do arquivo.
- Os valores de campo de uma tabela são *atômicos* e *monovalorados*. Ser atômico significa que o campo não pode ser composto de outros. Ser monovalorado significa que o campo possui um único valor e não um conjunto de valores. Exemplos de estruturas de dados que não têm estas restrições aparecem em muitas linguagens de programação, como C, Java ou Pascal. Exemplificando, em Pascal, um campo não necessita ser atômico, pois pode ser um registro, ou seja, pode ser composto por outros campos. Ainda em Pascal, um campo pode ser multivalorado, caso seja um arranjo (*array*) composto por vários valores.

Empregado

CodigoEmp	Nome	CodigoDepto	CategFuncional
E5	Souza	D1	C5
E3	Santos	D2	C5
E2	Silva	D1	C2
E1	Soares	D1	—

Figura 4.1 Tabela.

- As linguagens de consulta a bases de dados relacionais permitem o *acesso por quaisquer critérios* envolvendo os campos de uma ou mais linhas. Já em arquivos convencionais, para buscar registros com base em valores de seus campos de forma rápida, usualmente é necessário que exista algum tipo de *caminho de acesso*. Um caminho de acesso é uma estrutura auxiliar, como um índice ou uma cadeia de ponteiros, que acelera a recuperação de registros por determinados critérios, evitando a leitura exaustiva de todos os registros de um arquivo. Caminhos de acesso também existem em bancos de dados relacionais, mas não são visíveis pelos programadores, isto é, os programadores escrevem consultas sobre o banco de dados sem considerar a existência ou não de caminhos de acesso.

4.1.2 chave

O conceito básico para identificar linhas e estabelecer relações entre linhas de tabelas de um banco de dados relacional é o de *chave*. Em um banco de dados relacional, há ao menos três tipos de chaves a considerar: a chave *primária*, a chave *alternativa* e a chave *estrangeira*.

chave primária

Uma chave primária é uma coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela. Por exemplo, na tabela `Empregado` da Figura 4.1, a chave primária é a coluna `CódigoEmp`.

A Figura 4.2 apresenta um exemplo de uma tabela (`Dependente`) que possui uma chave primária *composta* (colunas `CodEmp` e `NoDepen`). Neste caso, nenhum dos campos que compõem a chave é suficiente para distinguir uma linha das demais já que, tanto um código de empregado (`CodEmp`) pode

Dependente

CodEmp	NoDepen	Nome	Tipo	DataNasc
E1	01	João	Filho	12/01/2001
E1	02	Maria	Filha	20/10/2003
E2	01	Ana	Esposa	12/12/1970
E5	01	Paula	Esposa	14/08/1981
E5	02	José	Filho	03/05/1985

Figura 4.2 Tabela com chave primária composta.

aparecer em diferentes linhas, quanto um número de dependente (NoDepen) pode aparecer em diferentes linhas. É necessário considerar ambos os valores (CodEmp e NoDepen) para identificar uma linha na tabela, ou seja, para identificar um dependente.

Pela definição acima, na tabela da Figura 4.2, qualquer combinação de colunas que contenha as colunas CodEmp e NoDepen é uma chave primária. Por isso, nas definições formais de chave primária, exige-se que essa seja *mínima*. Uma chave é mínima quando todas as suas colunas forem efetivamente necessárias para garantir o requisito de unicidade de valores da chave. Exemplificando, alguém poderia considerar a combinação de colunas CodEmp, NoDepen e Tipo como sendo uma chave primária. Entretanto, se eliminarmos desta combinação a coluna Tipo, continuamos frente a uma chave primária. Portanto, a combinação de colunas CodEmp, NoDepen e Tipo não obedece ao princípio da minimalidade e não deve ser considerada uma chave primária.

Cabe salientar que, na abordagem relacional, o termo "chave" é empregado com uma conotação diferente daquela usada na área de organização de arquivos e em alguns sistemas operacionais. Em arquivos convencionais, entende-se por chave qualquer coluna sobre a qual será definido um índice ou algum outro tipo de estrutura de acesso. Na abordagem relacional, ao definir uma chave primária, não está-se definindo nenhum caminho de acesso. Está-se definindo apenas uma *restrição de integridade*, isto é, uma regra que deve ser obedecida em todos os estados válidos do BD. No caso da chave primária, a regra definida pela chave é a de unicidade de valores nas colunas que compõem a chave.

■ chave estrangeira

Uma *chave estrangeira* é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela. A chave estrangeira é o mecanismo que permite a implementação de relacionamentos em um banco de dados relacional.

No banco de dados da Figura 4.3, a coluna CodigoDepto da tabela Emp é uma chave estrangeira em relação a chave primária da tabela Dept. Isto significa que, na tabela Emp, os valores do campo CodigoDepto de todas as linhas devem aparecer na coluna de mesmo nome da tabela Emp. A inter-

Dept

CodigoDepto	NomeDepto
D1	Compras
D2	Engenharia
D3	Vendas

Emp

CodEmp	Nome	CodigoDepto	CategFuncional	CPF
E1	Souza	D1	—	132.121.331-20
E2	Santos	D2	C5	891.221.111-11
E3	Silva	D2	C5	341.511.775-45
E5	Soares	D1	C2	631.692.754-88

Figura 4.3 Chave estrangeira.

pretação desta restrição é que todo empregado deve estar associado a um departamento.

A existência de uma chave estrangeira impõe restrições que devem ser garantidas ao executar diversas operações de alteração do banco de dados:

- `Quando da inclusão de uma linha na tabela que contém a chave estrangeira` – Neste caso, deve ser garantido que o valor da chave estrangeira apareça na coluna da chave primária referenciada. No exemplo da Figura 4.3, isto significa que um novo empregado deve atuar em um departamento já existente no banco de dados.
- `Quando da alteração do valor da chave estrangeira` – Deve ser garantido que o novo valor de uma chave estrangeira apareça na coluna da chave primária referenciada.
- `Quando da exclusão de uma linha da tabela que contém a chave primária referenciada pela chave estrangeira` – Deve ser garantido que, na coluna chave estrangeira, não apareça o valor da chave primária que está sendo excluída. No exemplo da Figura 4.3, isto significa que um departamento não pode ser excluído, caso nele ainda existirem empregados.
- `Quando da alteração do valor da chave primária referenciada pela chave estrangeira` – Deve ser garantido que, na coluna chave estrangeira, não apareça o valor antigo da chave primária que está sendo alterada. No exemplo da Figura 4.3, isto significa que, caso um departamento possua empregados, seu código não pode ser modificado.

A palavra "estrangeira" usada para denominar este tipo de chave pode ser enganosa. Ela pode levar a crer que a chave estrangeira sempre referencia uma chave primária de *outra* tabela. Entretanto, esta restrição não existe. Uma chave primária pode referenciar a chave primária da própria tabela, como mostra a Figura 4.4. Nesta tabela, a coluna `CodEmpGerente` é o código de outro empregado, o gerente do empregado correspondente à linha em questão. Como todo gerente é ele mesmo também um empregado, existe a restrição de que todo valor da coluna `CodEmpGerente` deve aparecer na coluna `CodEmp`. Assim, a coluna `CodEmpGerente` é chave estrangeira em relação à chave primária da própria tabela `Emp`.

■ chave alternativa

Em alguns casos, mais de uma coluna ou combinações de colunas podem servir para distinguir uma linha das demais. Uma das colunas (ou combinação de colunas) é escolhida como chave primária. As demais colunas ou combinações são denominadas chaves *alternativas*. A Figura 4.5 mostra um exemplo de uma tabela com dados de empregados (`Emp`), na qual tanto a coluna `CodEmp` quanto a coluna `CPF` podem ser usadas para distinguir uma linha das demais. Nesta tabela, como a coluna `CodEmp` foi escolhida como chave primária, diz se que a coluna `CPF` é uma chave alternativa.

Quando, em uma tabela, mais de uma coluna ou combinações de colunas servem para distinguir uma linha das demais, surge a questão de que critério deve ser usado para determinar qual das possíveis colunas (ou combinação de colunas) será usada como chave primária. No exemplo da Figura 4.5, por que a coluna `CodEmp` foi usada como chave primária e não a coluna `CPF`? Por que `CPF` não foi usado como chave primária e `CodEmp` como chave alternativa? Se considerarmos apenas a tabela em que a coluna aparece, não há diferen-

Emp

CodEmp	Nome	CodigoDepto	CodEmpGerente
E1	Souza	D1	—
E2	Santos	D2	E5
E3	Silva	D2	E5
E5	Soares	D1	E2

Figura 4.4 Chave estrangeira dentro da própria tabela.

Emp

CodEmp	Nome	CodigoDepto	CategFuncional	CPF
E1	Souza	D1	—	132.121.331-20
E2	Santos	D2	C5	891.221.111-11
E3	Silva	D2	C5	341.511.775-45
E5	Soares	D1	C2	631.692.754-88

Figura 4.5 Chave alternativa (coluna `CPF`).

ça entre uma coluna ser chave primária ou alternativa. Em ambos os casos, apenas está sendo especificada a unicidade de valores de chave. Entretanto, ao considerarmos chaves estrangeiras, a diferenciação entre chave primária e chave alternativa passa a ser relevante. Quando especificamos que uma chave é primária, estamos especificando, além da unicidade de valores, também o fato de esta coluna ser usada nas chaves estrangeiras que referenciam a tabela em questão. Assim, no caso da tabela da Figura 4.5, estamos especificando que tanto os valores de `CodEmp`, quanto os valores de `CPF` são únicos e, adicionalmente, que a coluna `CodEmp` será usada nas chaves estrangeiras que referenciam a tabela `Emp`.

domínios e valores vazios

Quando uma tabela do banco de dados é definida, para cada coluna da tabela deve ser especificado um conjunto de valores (alfanumérico, numérico,...) que os campos da respectiva coluna podem assumir. Este conjunto de valores é chamado de *domínio da coluna* ou *domínio do campo*.

Além disso, deve ser especificado se os campos da coluna podem estar *vazios* (*null*[1] em inglês) ou não. Estar vazio indica que o campo não recebeu valor de seu domínio. Na Figura 4.4, o campo `CategFuncional` da linha correspondente ao empregado de código *E1* está vazio. Isto indica que o empregado *E1* não possui categoria funcional ou que esta ainda não foi informada. Na Figura 4.4, aparece outro exemplo de um campo vazio. No caso, o campo `CodEmpGerente` vazio indica que o empregado não possui superior hierár-

[1] Alguns tradutores e até mesmo autores de textos sobre banco de dados em português têm usado a palavra "nulo" em vez de "vazio", como tradução literal da palavra *null* em inglês. Esta tradução não me parece adequada, pois pode levar a confusões entre o vazio, diferente de todos os valores dos domínios dos campos, e o valor zero (nulo), que pode aparecer no domínio dos campos numéricos.

quico. As colunas nas quais não são admitidos valores vazios são chamadas de colunas *obrigatórias*. As colunas nas quais podem aparecer campos vazios são chamadas de colunas *opcionais*.

Normalmente, os SGBDs relacionais exigem que todas as colunas que compõem a chave primária sejam obrigatórias. A mesma exigência não é feita para as demais chaves (ver exemplo de chave estrangeira vazia na Figura 4.4).

restrições de integridade

Um dos objetivos primordiais de um SGBD é a manutenção da *integridade de dados* sob seu controle. Dizer que os dados de um banco de dados estão íntegros significa dizer que eles refletem corretamente a realidade representada pelo banco de dados e que são consistentes entre si. Para tentar garantir a integridade de um banco de dados, os SGBDs oferecem o mecanismo de *restrição de integridade*. Uma restrição de integridade é uma regra de consistência de dados que é garantida pelo próprio SGBD. No caso da abordagem relacional, costuma-se classificar as restrições de integridade nas seguintes categorias:

- `Integridade de domínio` – Restrições deste tipo especificam que o valor de um campo deve obedecer a definição de valores admitidos para a coluna (o *domínio* da coluna). Nos primeiros SGBDs relacionais, era possível usar apenas domínios pré-definidos (número inteiro, número real, alfanumérico de tamanho definido, data, ...). Em SGBDs mais recentes, o usuário pode definir domínios próprios de sua aplicação (por exemplo, o domínio dos dias da semana ou das unidades da federação).
- `Integridade de vazio` – Através deste tipo de restrição de integridade é especificado se os campos de uma coluna podem ou não ser vazios (se a coluna é obrigatória ou opcional). Como já foi mencionado, campos que compõem a chave primária sempre devem ser diferentes de vazio.
- `Integridade de chave` – Trata-se da restrição que define que os valores da chave primária e alternativa devem ser únicos.
- `Integridade referencial` – É a restrição que define que os valores dos campos que aparecem em uma chave estrangeira devem aparecer na chave primária da tabela referenciada.

As restrições dos tipos acima especificados devem ser garantidas automaticamente por um SGBD relacional, isto é, não deve ser exigido que o programador escreva procedimentos para garanti-las explicitamente. Há outras restrições de integridade que não se encaixam em nas categorias acima e

que normalmente não são garantidas pelo SGBD. Essas restrições são chamadas de restrições *semânticas*. Alguns exemplos de restrições desse tipo poderiam ser:

- Um empregado do departamento denominado "Finanças" não pode ter a categoria funcional "Engenheiro".
- Um empregado não pode ter um salário maior que seu superior imediato.

4.1.3 modelo de banco de dados relacional

A especificação de um banco de dados relacional, ou seja, um *modelo* de banco de dados relacional, deve conter no mínimo a definição dos seguintes itens:

- tabelas que formam o banco de dados,
- colunas que as tabelas possuem e
- restrições de integridade.

Conforme mencionamos na introdução, um modelo de banco de dados pode ter diferentes representações, cada uma um *esquema* do banco de dados. Na prática, para representar esquemas relacionais, são usadas várias notações. A seguir, discutimos duas alternativas de notação para esquemas de banco de dados relacional, uma textual e outra diagramática.

esquema textual de BD relacional

Na prática, na linguagem padrão de manipulação de banco de dados relacional SQL, existe um grande conjunto de comandos para a manutenção do esquema do banco de dados. Esta linguagem inclui comandos para, entre outras coisas, criar uma nova tabela, excluir uma tabela existente e alterar a estrutura de uma tabela.

Nesta seção, vamos apresentar apenas uma notação resumida para esquemas textuais de BD relacional. Esta notação é incompleta, mas compacta, e é útil para exemplos como os mostrados no livro, bem como para discussões sobre a estrutura geral do banco de dados, quando não se deseja entrar no maior nível de detalhe.

A Figura 4.6 apresenta o esquema correspondente às tabelas da Figura 4.3 usando a notação resumida.

Nesta notação, são listadas as tabelas e, para cada tabela, enumerados, entre parênteses, os nomes das colunas que compõem a tabela. As colunas que compõem a chave primária aparecem sublinhadas. Após a definição de cada tabela, aparecem as definições das chaves estrangeiras que aparecem na tabela na forma:

<nome de coluna ch. estrangeira> `referencia`*<nome de tabela>*

quando tratar-se de uma chave estrangeira composta de uma única coluna, ou na forma:

(<nome de coluna>$_1$*, <nome de coluna>*$_2$*, ...)* `referencia`*<nome de tabela>*

quando tratar-se de uma chave estrangeira composta por múltiplas colunas.

■ esquema diagramático de BD relacional

Outra alternativa de representação de esquema de banco de dados relacional é através de diagramas. Muitas ferramentas CASE trabalham com notações deste tipo. Assim como não há padrão de notação diagramática para esquemas ER, também não há padrão para notação de esquemas relacionais.

Para exemplificar uma notação, a Figura 4.7 apresenta um exemplo de esquema diagramático para o banco de dados da Figura 4.3. Este diagrama foi confeccionado com o uso de uma ferramenta CASE comercial.

De maneira geral, esquemas diagramáticos de BD relacional estão organizados como descrito a seguir.

- Cada tabela é representada por um retângulo.
- As colunas que compõem a tabela são listadas dentro do retângulo representativo da tabela. Muitas vezes, notações adicionais indicam o domínio de cada coluna. No exemplo, os domínios são definidos pelas anotações `INTEGER` e `VARCHAR(40)`. Também a indicação das colunas que compõem

```
Emp (CodEmp,Nome,CodigoDepto,CategFuncional,CPF)
  CodigoDepto referencia Dept
Dept (CodigoDepto,Nome)
```

Figura 4.6 Esquema textual do banco de dados da Figura 4.3.

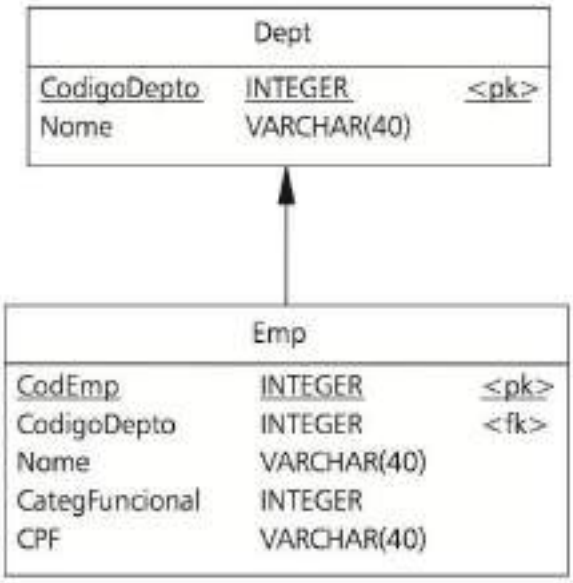


Figura 4.7 Esquema diagramático do banco de dados da Figura 4.3.

a chave primária pode aparecer no diagrama. No exemplo, as colunas que compõem a chave primária são indicadas pela sigla `pk` (de *primary key* – chave primária).

- As setas representam as chaves estrangeiras. No exemplo, a seta leva da tabela que contém a chave estrangeira para a seta que contém a chave primária (há notações na quais a seta tem o sentido inverso). As colunas que compõem as chaves estrangeiras são indicadas pela sigla `fk` (de *foreign key* – chave estrangeira).

4.2 → consultas sobre o banco de dados

Conforme mencionamos na introdução deste capítulo, não é nossa intenção fazer uma introdução completa à abordagem relacional. Mesmo assim, apresentamos um exemplo de uma consulta a um banco de dados relacional, a fim de mostrar algumas características importantes das linguagens relacionais.

A linguagem usada neste exemplo é SQL, a linguagem padrão de definição e manipulação do banco de dados. A instrução a seguir refere-se a uma consulta sobre o banco de dados da Figura 4.6:

```
SELECT Emp.Nome
FROM   Emp, Dept
```

```
WHERE  Dept.Nome = 'Computação' AND
       Emp.CodigoDepto = Dept.CodigoDepto AND
       Emp. CategFuncional='Programador'
```

A consulta busca os nomes dos empregados que estejam vinculados a um departamento denominado Computação e que pertencem à categoria funcional Programador. Quanto a esta consulta, cabem as seguintes observações:

- Na instrução SQL, o programador não faz referência a algum tipo de caminho de acesso. Quem decide quais caminhos de acesso serão eventualmente usados quando da execução da instrução é o SGBD.
- Quando em uma instrução estão envolvidas duas tabelas, a associação entre as linhas das duas tabelas é feita normalmente por uma operação chamada de *junção* (*join*). No exemplo as linhas de `Emp` são associadas às linhas de `Dept` pela igualdade do código do departamento.

4.3 → exercícios

exercício 1 Considere o banco de dados relacional definido parcialmente abaixo (faltam as chaves da tabela `Empregado`):

```
Empregado (CodEmpregado,Nome,NoPIS-PASEP)
Dependente(CodEmpregado,NoDependente,Nome)
  CodEmpregado referencia Empregado
```

Na tabela `Empregado`, tanto `CodEmpregado` quanto `NoPIS-PASEP` podem ser chave primária. Qual você escolheria como chave primária? Por quê?

exercício 2 Pegue um livro sobre organização de arquivos e veja o que significa "chave secundária". Explique por que o conceito de chave secundária não aparece na abordagem relacional.

exercício 3 A seguir aparece um esquema parcial para um banco de dados relacional. Identifique neste esquema as chaves primárias e as chaves estrangeiras:

```
Aluno        (CodigoAluno,Nome,CodigoCurso)
Curso        (CodigoCurso,Nome)
```

```
Disciplina   (CodigoDisciplina,Nome,Creditos,
              CodigoDepartamento)
Curriculo    (CodigoCurso,CodigoDisciplina,
              Obrigatória-Opcional)
Conceito     (CodigoAluno,CodigoDisciplina,Ano-Semestre,
              Conceito)
Departamento (CodigoDepartamento,Nome)
```

exercício 4 Considere um banco de dados com o seguinte esquema:

```
Paciente (CodigoConvenio,NumeroPaciente,Nome)
  CodigoConvenio referencia Convenio
Convenio (CodigoConvenio,Nome)
Medico   (CRM,Nome,Especialização)
Consulta (CodigoConvenio,NumeroPaciente,CRM,
          Data-Hora)
  (CodigoConvenio,NumeroPaciente) referencia Paciente
  CRM referencia Medico
```

Explique quais verificações devem ser feitas pelo SGBD para garantir a integridade referencial nas seguintes situações:

- Uma linha é incluída na tabela `Consulta`.
- Uma linha é excluída da tabela `Paciente`.
- O código do CRM em uma linha de `Consulta` é alterado.
- O código do CRM em uma linha de `Médico` é alterado.

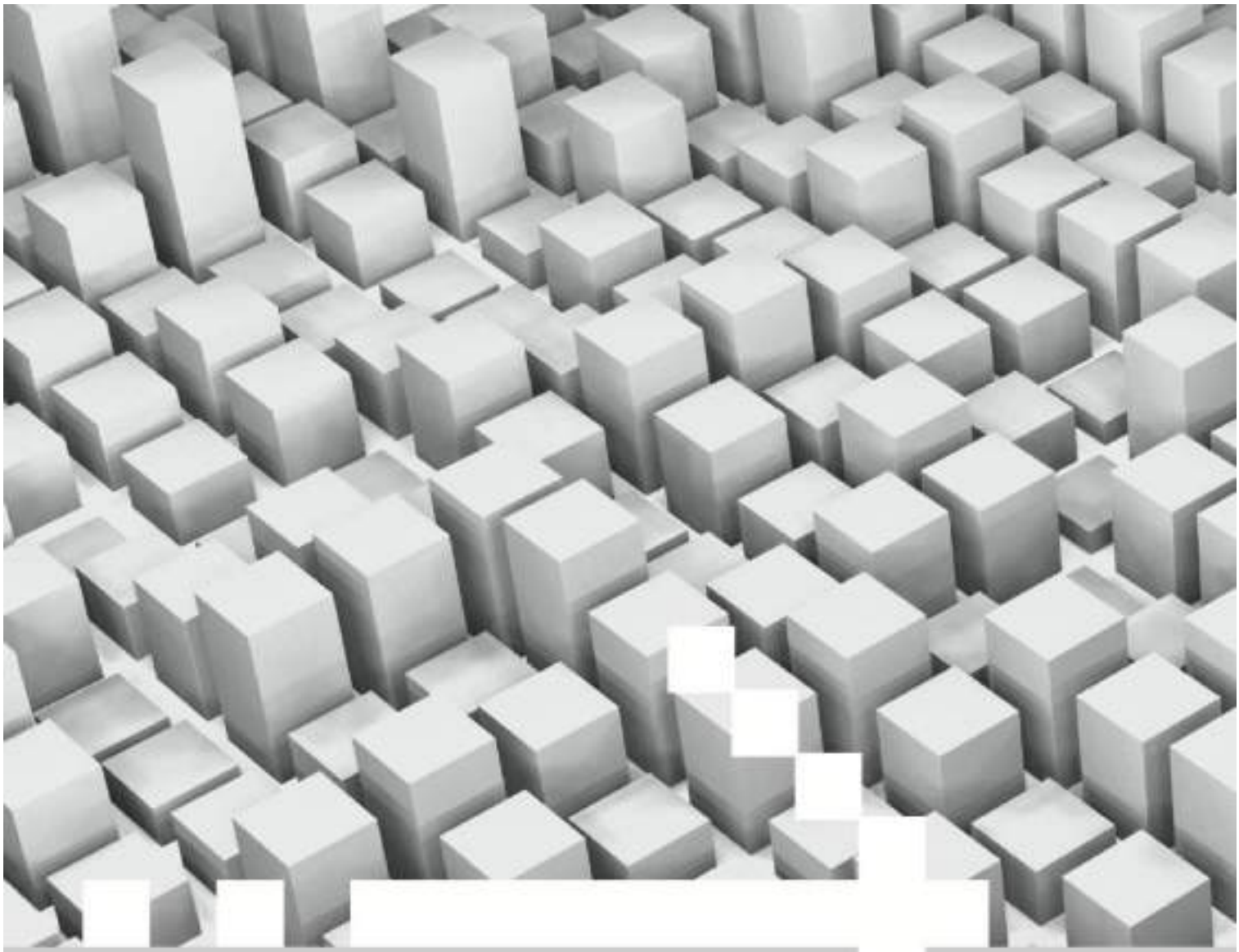
exercício 5 Usando a notação apresentada neste capítulo, construa um esquema diagramático para o banco de dados cujo esquema textual aparece no Exercício 4.

4.4 → leituras recomendadas

A abordagem relacional foi criada no final da década de 60 por um pesquisador da IBM, Ted Codd, que, visionariamente, entendeu a importância de separar detalhes internos referentes ao desempenho do SGBD dos aspectos funcionais relevantes aos programadores. O artigo considerado como o marco inicial da abordagem relacional é CODD (1970).

Os livros de ELMASRI; NAVATHE (2002) e SILBERSCHATZ; KORTH; SUDARSHAN (1999) são introduções completas ao assunto banco de dados, abordando exaustivamente não só SGBD da abordagem relacional, mas também SGBD de outros tipos. Além disso, ambos contêm capítulos que tratam do problema de organização de arquivos.

Um bom livro sobre os SGBDs relacionais comerciais e a linguagem SQL segundo o padrão estabelecido em 1992 é o livro de GROFF; WEINBERG (1999). O livro mais parece um manual de linguagem do que propriamente um livro acadêmico, mas faz uma excelente cobertura da linguagem SQL padrão e de suas implementações em vários SGBDs.



capítulo 5

transformações entre modelos

Nos capítulos anteriores, mostramos duas formas de modelagem de dados, a abordagem ER e a abordagem relacional. Estas abordagens propõem modelar os dados em diferentes níveis de abstração. A abordagem ER é voltada à modelagem de dados de forma independente do SGBD considerado, sendo adequada para a construção do *modelo conceitual*. Já a abordagem relacional modela os dados no nível de SGBD relacional. Um modelo neste nível de abstração é chamado de *modelo lógico*. Neste capítulo, vamos considerar a relação entre estes dois níveis de modelagem.

Inicialmente, vamos apresentar o *projeto lógico* de BD relacional. O projeto lógico consta da transformação de um modelo ER em um modelo lógico, que implementa, em nível de SGBD relacional, os dados representados abstratamente no modelo ER. O termo "implementação" significa que ocorre uma transformação de um modelo mais abstrato para um modelo que contém mais detalhes de implementação. Neste livro somente o projeto de BD relacional é tratado. Para outros tipos de SGBD (p.ex.: orientado a objetos ou objeto/relacional) outras regras de projeto são necessárias.

Após discutir o projeto lógico, vamos apresentar o processo inverso ao projeto, chamado de *engenharia reversa* de modelos relacionais. Neste processo, parte-se de um modelo relacional e obtém-se um diagrama ER, que representa de forma abstrata os dados armazenados no BD.

A Figura 5.1 dá uma visão geral dos dois processos de transformação entre modelos.

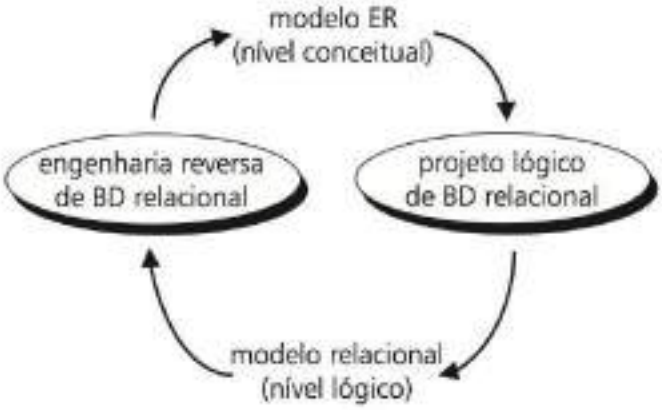


Figura 5.1 Transformações entre modelo ER e modelo relacional.

5.1 → visão geral do projeto lógico

Um determinado modelo ER pode ser implementado através de diversos modelos relacionais, que contêm as informações especificadas pelo diagrama ER. Cada um destes modelos relacionais alternativos pode ser visto

como uma implementação correta do modelo ER considerado. Entretanto, estes diferentes modelos relacionais podem resultar em diferentes performances do sistema construído sobre o banco de dados. Além disso, os diferentes modelos relacionais podem implicar em maior facilidade ou dificuldade de desenvolvimento e manutenção do sistema construído sobre o banco de dados.

As regras de projeto lógico aqui apresentadas são baseadas na experiência acumulada por muitos autores, no projeto de muitas bases de dados diferentes. Estas regras refletem um consenso de como deve ser projetado um banco de dados eficiente.

Entretanto, o modelo por elas fornecido pode ser considerado como um modelo relacional inicial. Nos casos em que este modelo relacional inicial não atenda aos requisitos de performance da BD projetada, há um processo de refinamento e melhoria do modelo, até ser atingido o modelo relacional satisfatório.

A Figura 5.2 resume os passos do projeto lógico.

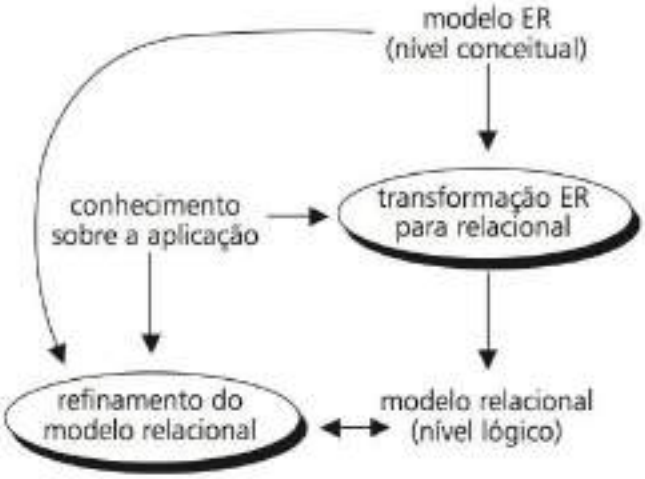


Figura 5.2 Visão geral do projeto lógico.

5.2 → transformação ER para relacional

Nesta seção, são apresentadas regras para a transformação de um modelo ER em um modelo relacional.

As regras foram definidas tendo em vista os dois *objetivos* básicos do projeto de BD:

- Obter um banco de dados que permita *boa performance* de instruções de consulta e alteração do banco de dados. Obter boa performance significa basicamente diminuir o número de acessos a disco, já que estes consomem o maior tempo na execução de uma instrução de banco de dados.
- Obter um banco de dados que *simplifique o desenvolvimento* e a manutenção de aplicações.

Estes são os dois objetivos centrais. Além destes, as regras de transformação procuram obter um banco de dados que ocupe pouco espaço em disco. O objetivo de reduzir espaço de armazenamento era, até algum tempo atrás, tão importante quanto o objetivo de melhorar a performance e simplificar o desenvolvimento. Entretanto, nos últimos anos, o preço dos meios de armazenamento tem diminuído, o que fez com que a redução de espaço ocupado, principalmente se em detrimento dos demais objetivos, diminuísse de importância.

A fim de alcançar os dois objetivos citados, as regras de tradução foram definidas tendo por base, entre outros, os seguintes *princípios*.

- `Evitar junções` – *Ter os dados necessários a uma consulta em uma única linha* – Um SGBD relacional normalmente armazena os dados de uma linha de uma tabela contiguamente em disco. Com isto, todos os dados de uma linha tabela são trazidos para a memória em uma única operação de acesso a disco. Isto significa que, uma vez encontrada uma linha de uma tabela, seus campos estão todos disponíveis, sem necessidade de acessos adicionais a disco.

  Quando for necessário buscar em disco dados de diversas linhas associadas pela igualdade de campos (por exemplo, buscar os dados de um empregado e os dados de seu departamento no banco de dados da Figura 4.3) é necessário usar a operação de *junção*. Os SGBDs procuram implementar a junção de forma eficiente, já que ela é uma operação executada

muito freqüentemente. Mesmo assim, a junção envolve diversos acessos a disco. Assim, quando for possível, é preferível ter os dados necessários a uma consulta em uma única linha, ao invés de tê-los distribuídos em diversas linhas, exigindo a sua junção.

- `Diminuir o número de chaves` – Para a implementação eficiente do controle da unicidade da chave primária ou chave alternativa, o SGBD usa normalmente uma estrutura de acesso auxiliar, um índice. O índice permite que o SGBD teste rapidamente a existência do valor de uma chave primária sendo incluída, sem que seja necessário fazer uma leitura exaustiva de toda tabela.

  Também para o controle de chaves estrangeiras são utilizados índices. O índice é usado pelo SGBD quando uma chave primária estiver sendo excluída ou alterada. Neste caso, o índice permite o acesso rápido à tabela que contém a chave estrangeira, para verificação da existência do antigo valor da chave primária. Exemplificando, no caso do banco de dados da Figura 4.3, o índice para a chave estrangeira seria um índice na tabela `Emp` pela coluna `CodigoDepto`. Este índice seria usado, por exemplo, quando uma linha de `Dept` fosse excluída, para garantir a ausência na tabela `Emp` de linhas referenciando a linha de `Dept` excluída.

  A regra geral de projeto físico de banco de dados é que, para cada chave primária, alternativa ou estrangeira, é necessário definir um índice.

  Pela forma em que são implementados, índices tendem a ocupar espaço considerável em disco. Além disso, a inserção ou remoção de entradas em um índice podem exigir múltiplos acessos a disco. Assim sendo, quando for necessário escolher entre duas alternativas de implementação, deve ser preferida aquela que exige o menor número de chaves e, conseqüentemente, o menor número de índices.

  Para exemplificar, vamos considerar que se deseja armazenar dados sobre clientes em um banco de dados relacional. Deseja-se armazenar, para cada cliente, seu código, seu nome, o nome da pessoa de contato, o endereço e o telefone. Estes dados poderiam ser implementados através de uma das seguintes alternativas:

```
Cliente(CodCliente,Nome,
        NomeContato,Endereço,Telefone)
```

ou

```
Cliente (CodCliente,Nome,NomeContato)
ClienteEnder(CodCliente,Endereço,Telefone)
  CodCliente referencia Cliente
```

Na primeira alternativa, o SGBD cria apenas um índice por código de cliente, a chave primária da tabela. Na segunda alternativa, o SGBD cria, para cada tabela, um índice por código de cliente. Como cada cliente aparece nas duas tabelas, os dois índices possuem exatamente as mesmas entradas, resultando em armazenamento e processamento dobrados. A primeira alternativa é a preferida se considerarmos este princípio de projeto.

- `Evitar campos opcionais` – Campos opcionais são campos que podem assumir o valor vazio (`NULL` em SQL). Os SGBDs relacionais usualmente não desperdiçam espaço ao armazenar um campo vazio, pois usam técnicas de compressão de dados e registros de tamanho variável no armazenamento interno de linhas. Assim, em princípio, não há problemas em usar este tipo de campo.

  Uma situação que pode gerar problemas é aquela na qual a obrigatoriedade ou não do preenchimento de um campo depende do valor de outros campos. Neste caso, em alguns SGBDs, o controle da obrigatoriedade deve ser feito pelos programas que acessam o banco de dados, o que deve ser evitado.

Nas seções que seguem, é explicado o processo de projeto lógico, ou seja, os passos envolvidos na transformação de um modelo conceitual ER em um modelo lógico relacional. Neste processo, são aplicados os princípios acima descritos.

O processo de projeto lógico consta dos seguintes passos:

1. Implementação inicial de entidades e respectivos atributos
2. Implementação de relacionamentos e respectivos atributos
3. Implementação de generalizações/especializações

5.2.1 implementação inicial de entidades

Este passo é razoavelmente óbvio: cada *entidade* é traduzida para uma *tabela*. Neste processo, cada *atributo* da entidade define uma *coluna* desta tabela. Os atributos identificadores da entidade definem as colunas que compõem a chave primária da tabela.

Trata-se aqui de uma tradução *inicial*. Pelas regras que seguem nas próximas seções, as tabelas definidas nesta etapa ainda poderão ser fundidas, no caso

de algumas alternativas de implementação de relacionamentos **1:1** e hierarquias de generalização/especialização.

A Figura 5.3 apresenta um exemplo da transformação de uma entidade em uma tabela. A figura mostra o DER e o esquema relacional correspondente. A entidade *PESSOA* com seus atributos código, nome, endereço, data de admissão e data de nascimento é transformada na tabela denominada `Pessoa` com colunas denominadas `CodigoPess`, `Nome`, `Endereço`, `DataNasc` e `DataAdm`. Como o atributo código é identificador da entidade, a coluna correspondente a este atributo é a chave primária da tabela.

■ nomes de atributos e nomes de colunas

Não é aconselhável simplesmente transcrever os nomes de atributos para nomes de colunas. Nomes de colunas serão referenciados freqüentemente em programas e outras formas de texto em computador. Assim, para diminuir o trabalho dos programadores é conveniente manter os nomes de colunas curtos. Além disso, em um SGBD relacional, o nome de uma coluna não pode conter brancos, nem hífens. Assim, nomes de atributos compostos de diversas palavras devem ser abreviados. Com base nestas considerações, os nomes de atributos data de nascimento e data de admissão foram traduzidos para os nomes de colunas `DataNasc` e `DataAdm` respectivamente.

Nas linguagens de banco de dados, o nome da tabela é muitas vezes usado como qualificador do nome da coluna. Exemplificando, para referenciar a coluna `Nome` da tabela `Pessoa` muitas vezes é usado um termo na forma `Pessoa.Nome`. Por isso, não é recomendado incluir, no nome de uma coluna, o nome da tabela em que ela aparece. Assim, é preferível usar o nome de coluna `Nome`, a usar os nomes de coluna `NomePess` ou `NomePessoa`. A exceção a esta regra é a coluna chave primária da entidade. Como esta coluna pode

Esquema relacional correspondente:

`Pessoa (CodigoPess,Nome,Endereço,DataNasc,DataAdm)`

Figura 5.3 Transformação de entidade em tabela.

aparecer em outras tabelas, na forma de chave estrangeira, é recomendável que os nomes das colunas que compõem a chave primária sejam sufixados ou prefixados com o nome ou sigla da tabela na qual aparecem como chave primária. Por este motivo, a coluna chave primária da tabela do exemplo recebeu a denominação de `CodigoPess`.

Outra recomendação quanto à nomeação de colunas é relativa ao uso de abreviaturas. Muitas vezes usa-se determinadas abreviaturas para tipos de campos que se repetem, como `Cod` para um código e `No` ou `Num` para um número. A recomendação é que se use sempre a mesma abreviatura em todo o banco de dados.

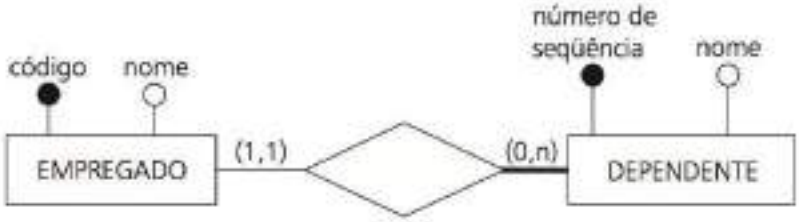
■ relacionamento identificador

A tradução de uma entidade que possui um *relacionamento identificador* tem algumas particularidades que são discutidas nesta seção.

Para iniciar, vamos considerar como exemplo a entidade *DEPENDENTE* mostrada no diagrama da Figura 5.4. Segundo este modelo, um dependente é identificado pelo código do empregado ao qual ele está vinculado e por um número de seqüência que distingue os diversos dependentes de um mesmo empregado.

A regra de implementação de relacionamentos identificadores é a seguinte:

- Para cada relacionamento identificador, é criada uma chave estrangeira na tabela que implementa a entidade identificada pelo relacionamento



Esquema relacional correspondente à entidade *DEPENDENTE*:

`Dependente(CodigoEmp,NoSeq,Nome)`

Figura 5.4 Implementação de entidade com relacionamento identificador.

identificador. Esta chave estrangeira é formada pelas colunas da chave primária da tabela referenciada pela chave estrangeira. Na Figura 5.4, isto significa que uma coluna `CodigoEmp` (chave primária da tabela correspondente à entidade *EMPREGADO*) deve ser adicionada à tabela `Dependente`.

- A chave primária da tabela que implementa a entidade identificada pelo relacionamento identificador é formada por:
  - colunas correspondentes aos atributos identificadores da entidade, se existirem, e
  - chaves estrangeiras que implementam os relacionamentos identificadores.

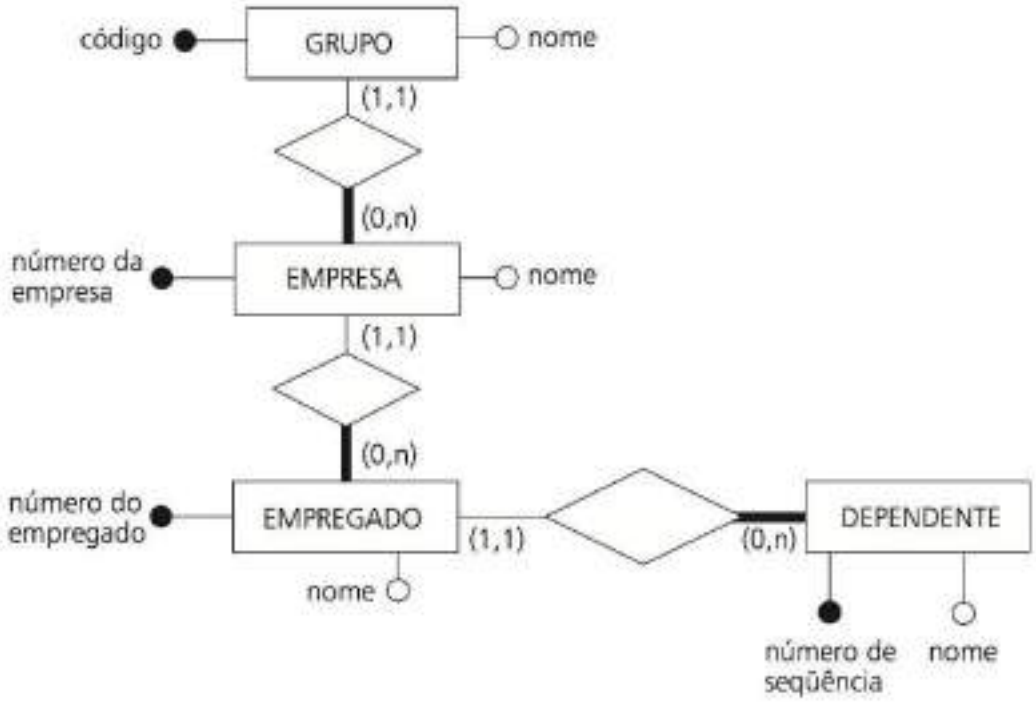
  No exemplo da Figura 5.4, isto significa que a chave primária da tabela `Dependente` é composta pelas colunas `CodigoEmp` (chave estrangeira que implementa o relacionamento identificador) e `NoSeq` (coluna que implementa o atributo identificador número de seqüência da entidade *DEPENDENTE*).

Cabe observar que, quando a entidade que possui o relacionamento identificador referencia uma entidade, que, por sua vez, também tem um relacionamento identificador, é necessário propagar por vários níveis a importação de chaves estrangeiras para a chave primária.

Como exemplo, vamos considerar o modelo ER da Figura 5.5. Neste caso, para compor a chave primária da tabela `Dependente`, é necessário usar, além do número de seqüência deste dependente, também o identificador do empregado. Entretanto, um empregado é identificado por seu número e pelo identificador da empresa à qual ele está vinculado. Por sua vez, a empresa é identificada por um número e pelo identificador do grupo ao qual ela pertence. Em outros termos, um dependente é identificado pela combinação das seguintes informações:

- código do grupo da empresa à qual seu empregado está vinculado;
- número da empresa à qual seu empregado está vinculado;
- número de seu empregado e
- seu número de seqüência.

Esta linha de raciocínio leva à chave primária da tabela `Dependente` que é mostrada na Figura 5.5.



Esquema relacional correspondente:

```
Grupo       (CodGrup,Nome)
Empresa     (CodGrup,NoEmpresa,Nome)
Empregado   (CodGrup,NoEmpresa,NoEmpreg,Nome)
Dependente  (CodGrup,NoEmpresa,NoEmpreg,NoSeq,Nome)
```

Figura 5.5 Implementação de entidade com relacionamento identificador: entidade referencia entidade identificada por relacionamento identificador.

5.2.2 implementação de relacionamentos

O fator determinante para a tradução a ser adotada no caso de relacionamentos é a *cardinalidade* mínima e máxima das entidades que participam do relacionamento. Antes de detalhar a alternativa proposta para cada tipo de relacionamento, apresentamos as três alternativas básicas de tradução de relacionamentos para o modelo relacional.

■ tabela própria

Nesta tradução, o relacionamento é implementado através de uma *tabela própria* que contém as seguintes colunas:

- colunas correspondentes aos identificadores das entidades relacionadas,
- colunas correspondentes aos atributos do relacionamento.

Caso o relacionamento sendo traduzido tiver a cardinalidade **n:n**, a chave primária desta tabela é formada pelas colunas correspondentes aos identificadores das entidades relacionadas e pelas colunas correspondentes aos atributos identificadores do relacionamento, caso estes existam. Cada conjunto de colunas que corresponde ao identificador de uma entidade é chave estrangeira em relação à tabela que implementa a entidade referenciada. Caso for um relacionamento com cardinalidades **1:n** ou **1:1**, a chave primária tem uma composição diferente, conforme discutido nas seções específicas para estes tipos de relacionamentos (páginas 147 e 152).

Um exemplo deste tipo de tradução é apresentado na Figura 5.6. Nesta figura, a parte do esquema do banco de dados referente à regra em questão está apresentada em negrito. Esta convenção será usada no restante da apresentação das regras de tradução de relacionamentos.

A tabela `Atuação` implementa o relacionamento *ATUAÇÃO*. A chave primária da tabela é formada pelas colunas `CodEng` e `CodProj`, que correspondem aos identificadores das entidades relacionadas (*ENGENHEIRO* e *PROJETO*). Cada uma destas colunas é uma chave estrangeira da tabela que implementa a entidade relacionada. A coluna `Função` corresponde ao atributo do relacionamento.

Esquema relacional correspondente:

```
Engenheiro (CodEng,Nome)
Projeto(CodProj,Título)
Atuação (CodEng,CodProj,Função)
  CodEng referencia Engenheiro
  CodProj referencia Projeto
```

Figura 5.6 Tradução de relacionamento por *tabela própria*.

adição de colunas

A outra alternativa de implementação de um relacionamento é a adição de colunas em uma das tabelas correspondentes às entidades que participam do relacionamento. Um exemplo deste tipo de tradução é apresentado na Figura 5.7. Este tipo de tradução somente é possível quando uma das entidades que participa do relacionamento tem cardinalidade máxima **1** (no exemplo, trata-se da entidade *EMPREGADO*). A tradução consta em inserir na tabela correspondente à entidade com cardinalidade máxima **1** as seguintes colunas:

- colunas correspondentes ao identificador da entidade relacionada (no exemplo, o identificador da entidade *DEPARTAMENTO*); estas colunas formam uma chave estrangeira em relação à tabela que implementa a entidade relacionada – no caso, a coluna `CodDept`,
- colunas correspondentes aos atributos do relacionamento – no caso, a coluna `DataLota`.

fusão de tabelas de entidades

A terceira forma de implementar um relacionamento é através da fusão das tabelas referentes às entidades envolvidas no relacionamento. Um exemplo deste tipo de tradução é apresentado na Figura 5.8. Esta tradução somente pode ser aplicada quando o relacionamento é de tipo **1:1**. A tradução consta em implementar, em uma única entidade, todos os atributos de ambas as entidades, bem como os atributos eventualmente existentes no relacionamento.

Esquema relacional correspondente:

```
Departamento(CodDept,Nome)
Empregado(CodEmp,Nome,CodDept,DataLota)
CodDept referencia Departamento
```

Figura 5.7 Tradução de relacionamento por *adição de coluna*.

Esquema relacional correspondente:

`Conferência(CodConf,Nome,DataInstComOrg,EnderComOrg)`

Figura 5.8 Tradução de relacionamento através de fusão de tabelas.

5.2.3 detalhes da implementação de relacionamentos

A alternativa específica que deve ser usada na tradução de um relacionamento é determinada pelas *cardinalidades* mínima e máxima das entidades envolvidas nos relacionamentos.

A Tabela 5.1 dá uma visão geral das alternativas que podem ser usadas. Para cada combinação de cardinalidades de relacionamento, a tabela mostra qual a alternativa preferida e outras alternativas, se houver. A alternativa preferida é indicada pelo símbolo V. Para alguns tipos de relacionamentos, existem outras alternativas que geram implementação correta, mas que, pelos princípios por trás do projeto lógico (página 138), não constituem a melhor implementação. Elas são indicadas pelos símbolos ± e ∓, em ordem decrescente de preferência de uso. Finalmente, as alternativas que não fazem sentido, porque levam a construções inválidas na abordagem relacional, são indicadas pelo símbolo x.

Nas seções que seguem, a regra de tradução correspondente a cada um dos tipos de relacionamento é justificada e exemplificada.

5.2.4 relacionamentos 1:1

■ ambas as entidades têm participação opcional

A Figura 5.9 apresenta um exemplo de relacionamento **1:1** no qual a participação de ambas as entidades é *opcional* (a cardinalidade mínima de ambas as entidades no relacionamento é zero). De acordo com a Tabela 5.1, a alternativa preferida de tradução de relacionamentos com esta cardinalidade é a

Tabela 5.1 Alternativas para implementação de relacionamentos

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
Relacionamentos 1:1			
(0,1) ◇ (0,1)	±	V	x
(0,1) ◇ (1,1)	∓	±	V
(1,1) ◇ (1,1)	∓	∓	V
Relacionamentos 1:n			
(0,1) ◇ (0,n)	±	V	x
(0,1) ◇ (1,n)	±	V	x
(1,1) ◇ (0,n)	∓	V	x
(1,1) ◇ (1,n)	∓	V	x
Relacionamentos n:n			
(0,n) ◇ (0,n)	V	x	x
(0,n) ◇ (1,n)	V	x	x
(1,n) ◇ (1,n)	V	x	x

V: Alternativa preferida
±: Pode ser usada, primeira opção
∓: Pode ser usada, segunda opção
x: Não cabe como solução

Figura 5.9 Implementação de relacionamento **1:1** com participação opcional de ambas as entidades.

adição de colunas na tabela referente a uma das entidades que participam do relacionamento. Como é um relacionamento **1:1**, qualquer das entidades que participam do relacionamento pode ser a escolhida.

Uma solução poderia ser:

```
Mulher(IdentM,Nome,IdentH,Data,Regime)
   IdentH referencia Homem
Homem (IdentH,Nome)
```

Neste esquema, as colunas referentes ao relacionamento estão marcadas em negrito. Trata-se de colunas referentes aos atributos de casamento, bem como a coluna `IdentH`, chave estrangeira que implementa o relacionamento. Neste caso, optou-se, arbitrariamente, por adicionar colunas à tabela `Mulher`. Da mesma forma, poderiam ter sido adicionadas colunas (identificador da mulher e atributos de casamento) à tabela `Homem`.

A outra alternativa seria a de gerar uma *tabela própria* para o relacionamento, conforme o esquema a seguir:

```
Mulher(IdentM,Nome)  [Homem (IdentH,Nome)
Casamento(IdentM,IdentH,Data,Regime)
  IdentM referencia Mulher
  IdentH referencia Homem
```

A tabela que implementa o relacionamento é a tabela `Casamento`. Nesta tabela, as colunas `IdentH` e `IdentM` são ambas chaves estrangeiras, implementando desta forma a vinculação da linha de casamento às linhas de homem e mulher correspondentes. Como se trata de um relacionamento **1:1**, tanto a coluna `IdentH`, quanto a coluna `IdentM` podem ser consideradas para a chave primária. No exemplo, a coluna `IdentM` foi arbitrariamente escolhida como chave primária, sendo `IdentH` uma chave alternativa.

A primeira alternativa (adição de colunas) é a preferida, pois minimiza a necessidade de junções, já que os dados de uma pessoa (na opção escolhida, a mulher) estão na mesma linha que os dados do casamento.

A desvantagem da primeira alternativa e que pode levar à utilização da segunda alternativa (tabela própria) é a de basear-se no uso de *colunas opcionais*, isto é, no uso de colunas que admitem valores vazios (no exemplo, as colunas `IdentH`, `Data` e `Regime` da tabela `Mulher`). Esta alternativa transfere a responsabilidade pela verificação da opcionalidade de campos do SGBD para os programas que atualizam o banco de dados. No caso da tabela `Mulher` do exemplo, nas linhas que correspondem a mulheres que não são casadas, os campos correspondentes ao casamento devem estar todos vazios. Já nas linhas correspondentes a mulheres casadas, os três campos devem estar preenchidos. Não há linhas em que, dentre os três campos, alguns estejam vazios e outros preenchidos. O controle que garante que os três campos estejam preenchidos ou vazios não é feito pelo SGBD, mas sim pela própria aplicação. Cabe observar que em alguns SGBDs que implementam restrições de integridade (cláusula *Check* ou similar) é possível transferir ao SGBD a responsabilidade pelo controle das colunas opcionais. Neste caso, a segunda alternativa seria a preferida.

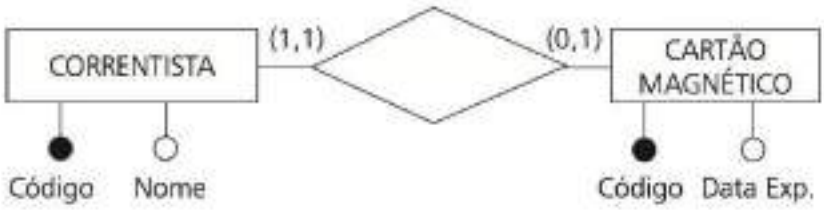
A alternativa de implementação por *fusão de tabelas* não é aplicável neste caso. A tabela resultante da fusão das tabelas referentes às duas entidades teria as seguintes colunas:

```
Tabela(IdentM,NomeM,IdentH,NomeH,Data,Regime)
```

O problema aqui é que, como ambas as entidades têm participação opcional, tanto `IndentH`, quanto `IdentM` são opcionais. Assim, nenhuma destas colunas poderia ser usada como chave primária, já que chaves primárias sempre são colunas obrigatórias.

■ uma entidade tem participação opcional e a outra tem participação obrigatória

Outro tipo de relacionamento **1:1** é aquele no qual uma das entidades tem participação *obrigatória*, enquanto que a outra entidade tem participação *opcional* (a cardinalidade mínima de uma das entidades é um, a cardinalidade mínima da outra entidade é zero). Um exemplo desta situação é apresentado na Figura 5.10.



Esquema relacional correspondente:

Correntista(CodCorrent,Nome,CodCartão,DataExp)

Figura 5.10 Implementação de relacionamento **1:1** com participação obrigatória de uma entidade e participação opcional da outra.

Neste caso, a tradução preferida é através da *fusão das tabelas* correspondentes às duas entidades.

Alternativamente, poderia ser considerada a tradução através da adição de colunas à tabela correspondente à entidade que obrigatoriamente está associada através do relacionamento em questão (no exemplo, esta entidade é cartão magnético).

```
Correntista(CodCorrent,Nome)
Cartão(CodCartão,DataExp,CodCorrent)
  CodCorrent referencia Correntista
```

■ ambas as entidades tem participação obrigatória

O último tipo de relacionamentos **1:1** é aquele no qual ambas as entidades tem participação *obrigatória* no relacionamento (a cardinalidade mínima de ambas as entidades é um). Um exemplo desta situação é apresentado na Figura 5.11.

Neste caso, a tradução preferida é através da *fusão das tabelas* correspondentes às duas entidades.

Nenhuma das demais alternativas é adequada. Em ambas as alternativas, as entidades que participam do relacionamento seriam representadas através de duas tabelas distintas. Estas tabelas teriam a mesma chave primária e relação um-para-um entre suas linhas. Essa implementação viola os princípios de evitar junções e diminuir o número de chaves primárias estabelecidos no início do capítulo.

Esquema relacional correspondente:

```
Conferência(CodConf,Nome,DataInstComOrg,EnderComOrg)
```

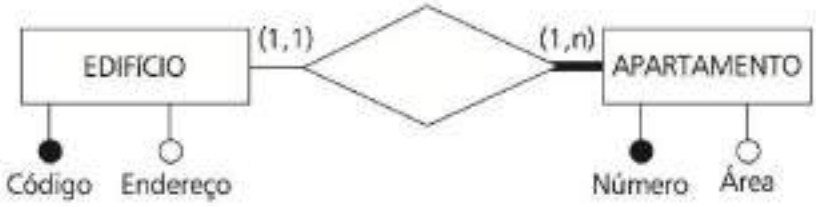
Figura 5.11 Implementação de relacionamento **1:1** com participação obrigatória de ambas as entidades.

5.2.5 relacionamentos 1:n

No caso de relacionamentos **1:n**, a alternativa preferida de implementação é a de *adição de colunas* (ver na página 146).

Um exemplo desta tradução é apresentado na Figura 5.12.

Cabe observar que, neste exemplo, a coluna `CódigoEd` da tabela `Apartamento` (que implementa o relacionamento do apartamento com seu edifício), além de ser chave estrangeira, é também parte da chave primária. Esta situação é típica de uma entidade com *relacionamento identificador*, cuja tradução foi discutida na página 142.



Esquema relacional correspondente:

```
Edifício(CódigoEd, Endereço)
Apartamento(CódigoEd,NúmeroAp,ÁreaAp)
  CódigoEd referencia Edifício
```

Figura 5.12 Tradução de relacionamentos **1:n** através de adição de colunas.

No caso de a entidade com cardinalidade máxima **1** ser opcional, isto é, possuir cardinalidade mínima **0**, poderia ser considerada uma implementação alternativa. Um exemplo deste tipo de relacionamento é mostrado na Figura 5.13. A entidade *VENDA* está *opcionalmente* ligada à entidade *FINANCEIRA*.

Para este tipo de relacionamento, pode ser usada alternativamente a implementação através de tabela própria.

A implementação através de *adição de colunas* à tabela de entidade `Venda` (implementação preferida) é a seguinte:

```
Financeira(CodFin,Nome)
Venda(IdVend,Data,CodFin,NoParc,TxJuros)
  CodFin referencia Financeira
```

As colunas em negrito na tabela Venda implementam o relacionamento.

A implementação através de *tabela própria* (implementação alternativa) é a seguinte:

```
Financeira(CodFin,Nome)
Venda      (IdVend,Data)
Financiam (IdVend,CodFin,NoParc,TxJuros)
  IdVend referencia Venda
  CodFin referencia Financeira
```

A implementação por tabela própria tem duas desvantagens em relação à implementação por adição de colunas:

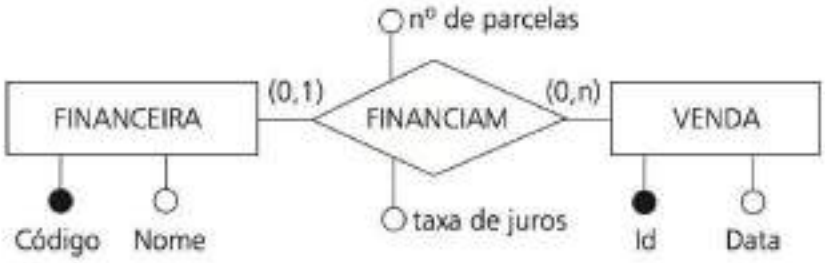


Figura 5.13 Tradução de relacionamentos **1:n** no qual a entidade com cardinalidade máxima um é opcional.

- Operações que envolvem acesso a dados de uma venda e do respectivo financiamento exigem junções. Na primeira alternativa, isto não ocorre, já que os dados da venda e de seu financiamento estão na mesma linha.
- As tabelas `Venda` e `Financiam` possuem a mesma chave primária, sendo o conjunto de valores de `Financiam` um subconjunto de `Venda`. Tem-se o problema acima mencionado de armazenamento e processamento duplicados de chave primária.

A única vantagem que a implementação por tabela própria apresenta é o fato de nela haver campos que são opcionais em certas linhas e obrigatórios em outras. Este é o caso dos campos `CodFin`, `NoParc` e `TxJuros` da tabela `Venda` na alternativa de adição de colunas. Estes campos estão obrigatoriamente preenchidos em caso de venda a prazo e vazios em caso contrário.

5.2.6 relacionamentos n:n

Independentemente da cardinalidade mínima, relacionamentos **n:n** são sempre implementados através de tabela própria. Um exemplo de implementação de relacionamentos **n:n** é apresentado na Figura 5.6.

A alternativa de adicionar colunas a uma das tabelas correspondentes às entidades que participam do relacionamento não é aplicável. Cada entidade está associada a um número variável de entidades. Para implementar o relacionamento através da adição de colunas, seria necessária uma coluna *multivalorada*, que comportasse um conjunto de valores de chaves primárias, referente à entidade associada. Entretanto, como vimos no capítulo anterior, as colunas na abordagem relacional são sempre *monovaloradas*. Assim, esta alternativa não é viável, pelas próprias características da abordagem relacional.

5.2.7 relacionamentos de grau maior que 2

As alternativas de implementação de relacionamentos apresentadas até este ponto aplicam-se somente à implementação de relacionamentos binários, isto é, que envolvem apenas duas entidades. Como visto na Seção Relacionamento ternário, na página 44, algumas variantes da abordagem ER admitem relacionamentos de grau maior que 2. Para fins de exemplo, vamos considerar o modelo ER apresentado na Figura 5.14.

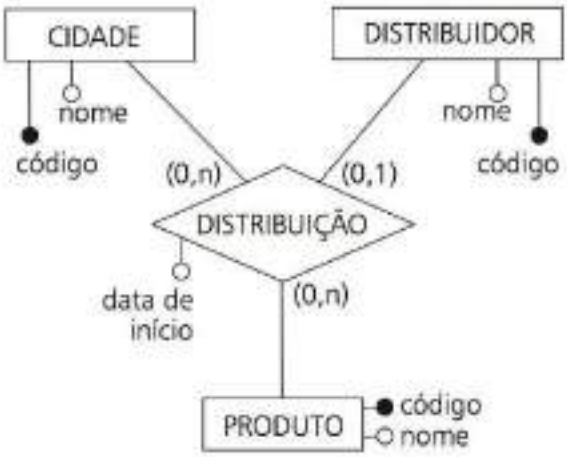


Figura 5.14 Relacionamento ternário a ser implementado.

Para relacionamentos de grau maior que 2, não são definidas regras específicas. A implementação de um relacionamento de grau maior que 2 dá-se na seguinte seqüência de passos:

1. O relacionamento é transformado em uma entidade. Esta nova entidade é ligada através de um relacionamento binário a cada uma das entidades que participavam do relacionamento original.
2. As regras de implementação de entidades e relacionamentos binários apresentadas acima são aplicadas às entidades e aos relacionamentos binários assim criados.

A Figura 5.15 mostra o resultado da transformação do relacionamento ternário que aparece no modelo da Figura 5.14 em uma entidade (resultado da aplicação do passo 1 acima).

A implementação deste modelo seguindo as regras apresentadas acima resulta no seguinte esquema relacional:

```
Produto(CodProd,Nome) [Cidade (CodCid,Nome)
Distribuidor(CodDistr,Nome)
Distribuição (CodProd,CodDistr,CodCid,DataDeInicio)
    CodProd referencia Produto
    CodDistr referencia Distribuidor
    CodCid referencia Cidade
```

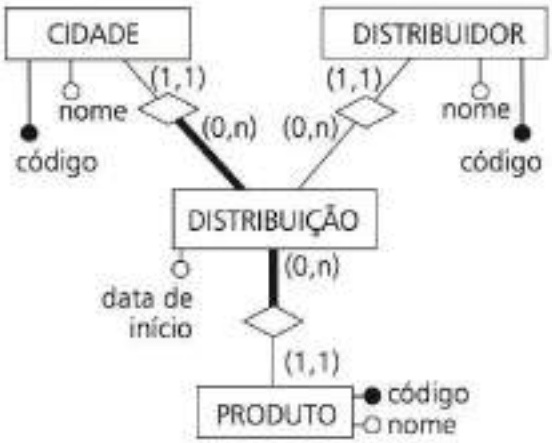


Figura 5.15 Transformação de relacionamento ternário em entidade.

5.2.8 implementação de generalização/especialização

Para a implementação de hierarquias de generalização/especialização na abordagem relacional, há duas alternativas principais a considerar: (1) uso de uma tabela para cada entidade e (2) uso de uma única tabela para toda hierarquia de generalização/especialização. A seguir apresentamos as duas alternativas para, depois, discutir quando usar cada uma.

■ uma tabela por hierarquia

Nesta alternativa, todas as tabelas referentes às especializações de uma entidade genérica são fundidas em uma única tabela, composta por:

- chave primária correspondente ao identificador da entidade mais genérico;
- caso não exista, uma coluna `Tipo`, que identifica que tipo de entidade especializada está sendo representada por cada linha da tabela;
- uma coluna para cada atributo da entidade genérica;
- colunas referentes aos relacionamentos dos quais participa a entidade genérica e que sejam implementados através da alternativa de adicionar colunas à tabela da entidade genérica;
- uma coluna para cada atributo de cada entidade especializada (estas colunas devem ser definidas como opcionais, já que somente terão valores quando a linha for referente à entidade especializada em questão);

- colunas referentes aos relacionamentos dos quais participa cada entidade especializada e que sejam implementados através da alternativa de adicionar colunas à tabela da entidade (estas colunas devem ser definidas como opcionais, já que somente terão valores quando a linha for referente à entidade especializada em questão).

Observe-se que, pela definição acima, uma entidade especializada pode não gerar uma coluna na implementação. Isto ocorrerá caso a entidade especializada não tenha atributos e caso todos os relacionamentos dos quais ela participe sejam implementados através de tabelas próprias.

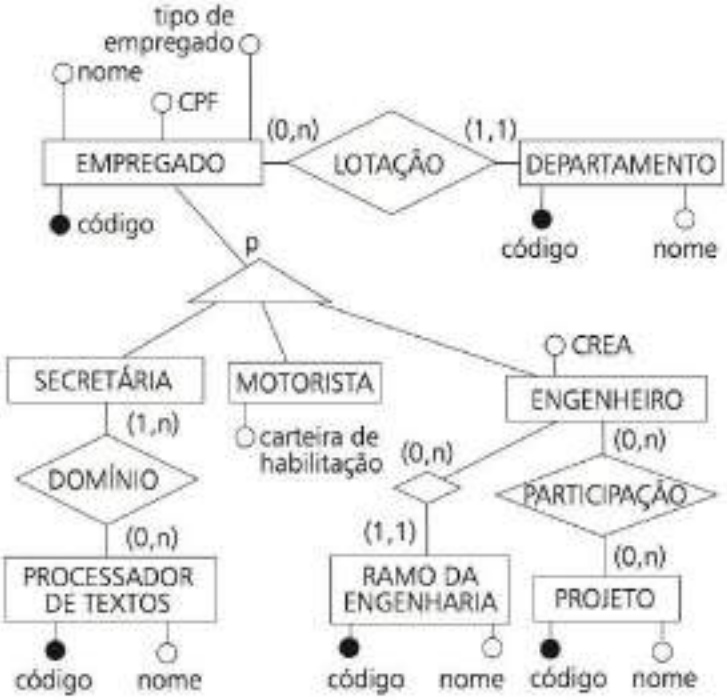
Um exemplo de implementação usando uma única tabela para toda hierarquia de especialização da entidade *EMPREGADO* aparece na Figura 5.16.

A tabela `Emp`, que implementa a entidade *EMPREGADO* e suas especializações, contém as seguintes colunas:

- `CódigoEmp`, chave primária da tabela, correspondente ao identificador da entidade;
- as colunas `Tipo`, `Nome` e `CPF` referentes aos atributos da entidade genérica;
- a coluna `CódigoDept`, que implementa o relacionamento *LOTAÇÃO*;
- a coluna `CartHabil` que implementa os atributos da entidade especializada *MOTORISTA*;
- a coluna `CREA` que implementa os atributos da entidade especializada *ENGENHEIRO*;
- a coluna `CódigoRamo`, que implementa o relacionamento entre *ENGENHEIRO* e *RAMO DA ENGENHARIA*.

Pelas regras apresentadas na seção anterior, os relacionamentos *PARTICIPAÇÃO* e *DOMÍNIO* são implementados por tabela própria, não gerando colunas na tabela correspondente à hierarquia de generalização/especialização. Além disso, a entidade *SECRETÁRIA* não gera uma coluna, já que não possui atributos, nem participa de relacionamentos que gerem colunas.

As colunas que correspondem às entidades especializadas (`CartHabil`, `CREA` e `CódigoRamo`) devem ser definidas como colunas *opcionais*. Essa definição é necessária, pois uma linha referente a um empregado, que não pertença a alguma das classes especializadas, terá todos os campos anteriormente listados



Esquema relacional correspondente:

```
Emp(CódigoEmp,Tipo,Nome,CPF,CódigoDept,
    CartHabil,CREA,CódigoRamo)
  CódigoDept referencia Depto
  CódigoRamo referencia Ramo
Depto(CódigoDept, Nome)
Ramo(CódigoRamo,Nome)
ProcessTexto(CódigoProc,Nome)
Domínio(CódigoEmp,CódigoProc)
  CódigoEmp referencia Emp
  CódigoProc referencia ProcessaTexto
Projeto(CódigoProj,Nome)
Participação(CódigoEmp,CódigoProj)
  CódigoEmp referencia Emp
  CódigoProj referencia Projeto
```

Figura 5.16 Hierarquia de generalização/especialização e sua implementação através de tabela única.

vazios. Já uma linha correspondente a uma entidade especializada terá alguns campos vazios e outros preenchidos. Exemplificando, uma linha referente a um engenheiro teria os campos `CREA` e `CódigoRamo` preenchidos e o campo `CartHabil` vazio.

As demais tabelas que implementam o modelo da Figura 5.16, definidas usando as regras apresentadas nas seções anteriores, são:

- `Depto`, que implementa a entidade *DEPARTAMENTO*
- `Ramo`, que implementa a entidade *RAMO DA ENGENHARIA*
- `ProcessTexto`, que implementa a entidade *PROCESSADOR DE TEXTO*
- `Domínio`, que implementa o relacionamento *DOMÍNIO*
- `Projeto`, que implementa a entidade *PROJETO*
- `Participação`, que implementa o relacionamento *PARTICIPAÇÃO*

■ uma tabela por entidade especializada

A outra alternativa de implementação de uma hierarquia de generalização/especialização é criar uma tabela para cada entidade que compõe a hierarquia, aplicando as regras correspondentes à implementação de entidades e relacionamentos já apresentadas nas seções anteriores.

O único acréscimo que deve ser feito àquelas regras é a inclusão da chave primária da tabela correspondente à entidade genérica, em cada tabela correspondente a uma entidade especializada. Exemplificando, a implementação do modelo ER da Figura 5.16 resultaria no esquema relacional apresentado na Figura 5.17 (parte referente à hierarquia de generalização/especialização em negrito).

```
Emp (CódigoEmp,Tipo,Nome,CPF,CódigoDept)
    CódigoDept referencia Depto
Motorista(CódigoEmp,CartHabil)
    CódigoEmp referencia Emp
Engenheiro(CódigoEmp,CREA,CódigoRamo)
    CódigoEmp referencia Emp
    CódigoRamo referencia Ramo
Depto(CódigoDept,Nome)
Ramo(CódigoRamo,Nome)
ProcessTexto(CódigoProc,Nome)
Domínio(CódigoEmp,CódigoProc)
    CódigoEmp referencia Emp
    Código Proc referencia ProcessTexto
Projeto(CódigoProj,Nome)
Participação(CódigoEmp,CódigoProj)
    CódigoEmp referencia Engenheiro
    CódigoProj referencia Projeto
```

Figura 5.17 Generalização/especialização: implementação através de uma tabela por entidade especializada.

Para a entidade *EMPREGADO* e cada uma de suas especializações, foi criada uma tabela. Estas tabelas têm, todas, a mesma chave primária. A tabela `Emp` contém uma linha para cada empregado, independentemente de seu tipo. Nela aparecem as informações comuns a todos os empregados. As informações referentes a cada tipo particular de empregado estão nas tabelas `Motorista` e `Engenheiro`. Em cada uma delas, aparecem linhas somente para empregados pertencentes ao tipo representado pela tabela.

Nas tabelas referentes às entidades especializadas, a chave primária é também chave estrangeira em relação à tabela de empregados. Isso ocorre porque a toda ocorrência de uma entidade especializada corresponde uma ocorrência de entidade genérica, ou seja, a toda linha de uma tabela de entidade especializada corresponde uma linha da tabela da entidade genérica.

■ comparação entre as duas alternativas de implementação

A seguir são discutidas as vantagens de cada tipo de implementação em relação à outra.

- *Vantagens da implementação com tabela única*
  - Todos os dados referentes a uma ocorrência de entidade genérica, bem como os dados referentes a ocorrências de sua especialização, estão em uma única linha. Não há necessidade de realizar junções, quando a aplicação deseja obter dados referentes a uma ocorrência de entidade genérica juntamente com uma ocorrência de entidade especializada.
  - A chave primária é armazenada uma única vez, ao contrário da alternativa com múltiplas tabelas, na qual a chave primária aparece tanto na tabela referente à entidade genérica, quanto na tabela referente à entidade especializada.
- *Vantagens da implementação com uma tabela por entidade especializada*
  - As colunas opcionais que aparecem são apenas aquelas referentes a atributos que podem ser vazios do ponto de vista da aplicação. Na solução alternativa, todas as colunas referentes a atributos e relacionamentos das entidades especializadas devem ser definidas como opcionais.
  - O controle de colunas opcionais passa a ser feito pela aplicação, com base no valor da coluna TIPO e não pelo SGBD, como ocorre na solução alternativa.

O projetista deverá ponderar as vantagens e desvantagens de ambas as soluções e optar por aquela que, considerando os fatores acima, seja a mais adequada ao seu problema.

■ subdivisão da entidade genérica

Além das duas alternativas acima, alguns textos de projeto de banco de dados apresentam uma terceira implementação para a generalização/especialização. Nesta alternativa, cria-se uma tabela para cada entidade especializada que não possua outra especialização (entidade folha da árvore). Esta tabela contém tanto os dados da entidade especializada, quanto os de suas entidades genéricas. No caso do modelo ER da Figura 5.16, a implementação é a apresentada na Figura 5.18 (parte referente à hierarquia de generalização/especialização em negrito).

A diferença desta alternativa em relação a anterior é que, nesta alternativa, as tabelas correspondentes às especializações contém, não só os atributos da entidade especializada, mas também os atributos de suas generalizações. A tabela contém não só os atributos específicos da entidade *MOTORISTA*, mas também os atributos referentes à sua generalização (atributos *CódigoEmp*, *Tipo*,*Nome*, *CPF* e *CódigoDept*). De forma análoga, a tabela `Engenheiro` contém não só os atributos específicos de engenheiro, mas também os de empregado. Finalmente, a tabela `EmpOutros` contém dados de todas as demais categorias de empregados.

```
EmpOutros(CódigoEmp,Tipo,Nome,CPF,CódigoDept)
    CódigoDept referencia Depto
Motorista(CódigoEmp,Nome,CPF,CódigoDept,CartHabil)
Engenheiro(CódigoEmp,Nome,CPF,CódigoDept,CREA,CódigoRamo)
  CódigoRamo referencia Ramo
Depto(CódigoDept,Nome)
Ramo(CódigoRamo,Nome)
ProcessTexto(CódigoProc,Nome)
Domínio(CódigoEmp,CódigoProc)
  CódigoEmp referencia EmpOutros
  CódigoProc referencia ProcessTexto
Projeto(CódigoProj,Nome)
Participação(CódigoEmp,CódigoProj)
  CódigoEmp referencia Engenheiro
  CódigoProj referencia Projeto
```

Figura 5.18 Generalização/especialização: implementação através de subdivisão da entidade genérica.

É importante observar que, nesta alternativa, as colunas `CódigoEmp` que aparecem como chave nas tabelas referentes às diversas especializações de empregado não são chave estrangeira, já que não existe uma tabela onde todos os empregados estão reunidos, como ocorria na alternativa anterior. Além disso, quando a especialização for total (e não parcial como no exemplo), deixa de existir a tabela que coleciona as linhas referentes à entidades para as quais não há especialização (no exemplo, a tabela `EmpOutros`).

Essa alternativa tem como vantagens sobre as anteriores o fato de eliminar os problemas de colunas opcionais e chaves primárias redundantes, típicos das soluções anteriormente apresentadas.

Entretanto, esta alternativa apresenta uma desvantagem do ponto de vista funcional que supera as vantagens oferecidas quanto ao uso mais eficiente de recursos. Nesta alternativa, para garantir a unicidade da chave primária, a aplicação que faz inclusões de linhas de empregados deve verificar todas as tabelas referentes às especializações. Exemplificando, quando for incluído um novo empregado, a aplicação deverá testar a sua existência nas três tabelas (`EmpOutros`, `Motorista` e `Engenheiro`). Essa verificação fica a cargo da aplicação, já que os SGBDs relacionais não são capazes de realizá-la automaticamente.

Adicionalmente, não há como especificar ao SGBD restrições de integridade referenciais que façam referência ao conjunto de empregados como um todo (já que este não aparece no banco de dados).

5.2.9 refinamento do modelo relacional

O processo de tradução acima descrito leva a um banco de dados correto do ponto de vista da abordagem relacional e que implementa os dados que são especificados pelo modelo conceitual.

Entretanto, o banco de dados projetado não deve ser visto como final, pois pode conter redundância de dados, conforme explicado mais adiante.

Além disso, deve ser considerado que, em todo processo de engenharia, está envolvido um compromisso entre o ideal e o realizável dentro das restrições de recursos impostas pela prática. No processo de engenharia de banco de dados, particularmente na implementação de modelos conceituais, às vezes

também é necessário traçar um compromisso entre o ideal, representado pelas regras de implementação apresentadas, e o alcançável frente a limitações de performance. Algumas vezes, o esquema de BD criado através do uso dessas regras não atende plenamente os requisitos de performance impostos ao sistema. Neste caso, é necessário buscar uma alternativa de implementação que resulte em um melhor desempenho do sistema. Cabe salientar que estas alternativas somente devem ser aplicadas como último recurso, pois, do ponto de vista do desenvolvimento de programas sobre o banco de dados, elas são sempre piores que as alternativas que haviam sido apresentadas anteriormente.

Nas seções seguintes, apresentamos, inicialmente, um exemplo de situação em que as regras de tradução apresentadas introduzem redundância de dados no banco de dados e, posteriormente alguns exemplos de alternativas de implementação que podem ser adotadas para a melhoria do desempenho.

■ redundância no banco de dados projetado

Conforme mencionado acima, há situações em que a aplicação das regras de tradução apresentadas leva a um banco de dados com redundância de dados. O problema está ligado ao uso de *relacionamentos identificadores*.

Para mostrar um exemplo dessa situação, vamos considerar o modelo ER mostrado na Figura 5.19 que corresponde ao banco de dados de uma empresa que ministra cursos e que mantém informações sobre as avaliações dos cursos. A entidade *CURSO* representa os cursos ministrados, havendo

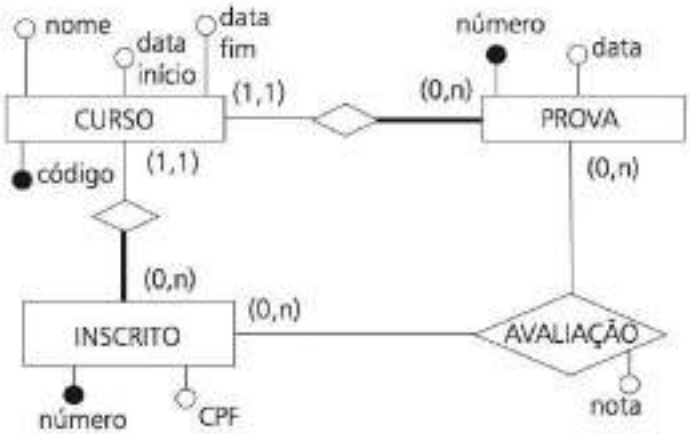


Figura 5.19 Modelo ER com relacionamentos identificadores.

uma ocorrência para cada curso. Um curso é identificado por um código, tem um nome, uma data de início e uma data de fim. A entidade *INSCRITO* representa cada um dos inscritos em um curso. Esta entidade é identificada pelo curso no qual ocorreu a inscrição e por um número seqüencial que distingue as inscrições de um curso. Já a entidade *PROVA* corresponde às provas que são realizadas em um curso, e é identificada pelo curso e por um número seqüencial da prova dentro do curso. Finalmente, o relacionamento *AVALIAÇÃO* armazena a nota que é atribuída a um inscrito em uma prova.

A aplicação das regras de tradução descritas acima leva ao esquema relacional mostrado na Figura 5.20.

Entretanto, o banco de dados da Figura 5.20 apresenta redundância de dados. As colunas `CodCursoProva` e `CodCursoInscrito` contêm exatamente os mesmos valores, já que os inscritos em cursos somente realizam provas do curso em que estão inscritos, e não em outros cursos. O problema é que esta restrição de integridade não está expressa no modelo ER e, portanto, não é considerada pelas regras de tradução para o modelo lógico.

Por esta razão, após aplicar as regras de tradução de ER para relacional apresentadas na seção anterior, é necessário fazer uma revisão do modelo relacional, para eliminar eventuais redundâncias de dados.

No exemplo da Figura 5.20, as colunas `CodCursoProva` e `CodCursoInscrito` devem ser fundidas em uma única coluna, o que resulta no modelo da Figura 5.21.

```
Curso(CodCurso,Nome,DataIni,DataFim)
Prova(CodCurso,NumeroProva,DataProva)
    CodCurso referencia Curso
Inscrito(CodCurso,NumeroInscrito,CPF)
    CodCurso referencia Curso
Avaliacao(CodCursoProva,NumeroProva,
          CodCursoInscrito,NumeroInscrito,Nota)
    (CodCursoProva,NumeroProva) referencia Prova
    (CodCursoInscrito,NumeroInscrito) referencia Inscrito
```

Figura 5.20 Modelo relacional resultante da aplicação das regras de projeto lógico sobre o modelo ER da Figura 5.19.

Curso(CodCurso,Nome,DataIni,DataFim)
Prova(CodCurso,NumeroProva,DataProva)
CodCurso referencia Curso
Inscrito(CodCurso,NumeroInscrito,CPF)
CodCurso referencia Curso
Avaliacao(CodCurso,NumeroProva,NumeroInscrito,Nota)
(CodCurso,NumeroProva) referencia Prova
(CodCurso,NumeroInscrito) referencia Inscrito

Figura 5.21 Modelo relacional referente ao modelo ER da Figura 5.19, após a eliminação de redundância.

■ manutenção de informações redundantes no banco de dados

Como vimos na Seção Um modelo deve ser livre de redundância página 87, as informações redundantes (informações que podem ser obtidas a partir de outras existentes no banco de dados) devem ser eliminadas do modelo conceitual. Na seção precedente, vimos também a eliminação de informações redundantes do modelo relacional.

Entretanto, por questões de desempenho das aplicações que acessam o banco de dados, às vezes, informações redundantes são desejáveis no banco de dados. Isto pode ocorrer com atributos redundantes, eventualmente resultantes de computações que envolvam vários acessos ao banco de dados, e que são acessados muito freqüentemente, mas sofrem poucas alterações. Neste caso, considerando o desempenho do sistema como um todo, pode ser mais eficiente manter o atributo redundante armazenado no banco de dados.

A Figura 5.22 apresenta um exemplo, no qual aparece um atributo redundante em um fragmento do modelo de dados de um sistema de reserva de passagens aéreas. Trata-se do atributo *número de reservas*, que pode ser obtido pela contagem de todas as reservas relacionadas ao vôo. Do ponto de vista conceitual, o atributo *número de reservas* deveria ser eliminado, por ser redundante. Entretanto, do ponto de vista de desempenho global do sistema, poderia ser importante manter uma coluna com este valor, caso a consulta ao número de reservas fosse freqüente e sua computação demandasse tempo considerável.

■ simulação de atributos multivalorados

Conforme discutimos na Seção Tabela, na página 120, na abordagem relacional, não existem colunas multivaloradas.

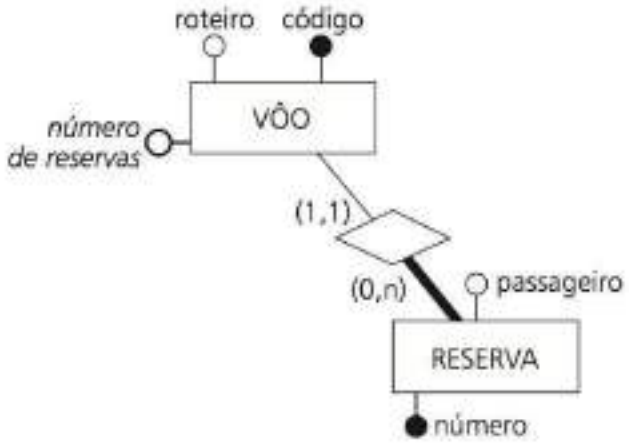


Figura 5.22 Atributo redundante mantido no projeto lógico.

Por esta razão, modelos ER destinados ao projeto de bancos de dados relacionais são, normalmente, construídos sem o uso de atributos multivalorados. A Figura 5.23 apresenta um diagrama ER com atributos multivalorados e o diagrama obtido pela transformação do atributo multivalorado em uma entidade separada.

A implementação do diagrama da Figura 5.23, caso sejam seguidas as regras apresentadas, é a seguinte:

```
Cliente (CodCli,Nome)
Telefone(CodCli,Número)
    CodCli referencia Cliente
```

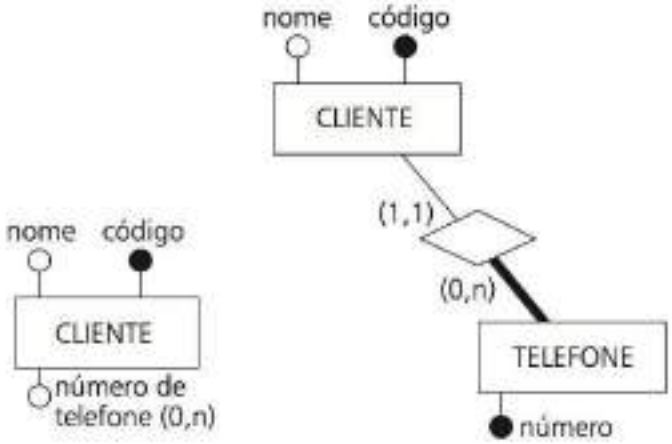


Figura 5.23 Eliminando atributos multivalorados.

Entretanto, esta implementação pode trazer problemas de desempenho, caso seja necessário obter todos os telefones de um cliente com freqüência. Esta operação implica em busca de várias linhas de `Telefone`, tantas linhas quantos telefones o cliente possuir.

Dadas certas condições de contorno, é possível conceber uma implementação simplificada, que atenda com mais eficiência às buscas aos telefones de um cliente. Consideremos as seguintes condições de contorno:

- São raros os clientes que possuem mais que dois telefones. Quando isso ocorrer, é suficiente armazenar apenas dois números.
- Não há consultas ao banco de dados usando o número de telefone como critério de seleção. Os números de telefone são apenas exibidos ou impressos junto às demais informações de cliente.

Neste caso, uma implementação "desnormalizada"[1] como a mostrada a seguir pode permitir um maior desempenho das aplicações:

```
Cliente(CodCli,Nome,NumTel1,NumTel2)
```

Nesta implementação, optou-se por simular uma coluna multivalorada através da criação de diversas colunas `NumTel` sufixadas por um número para distingui-las. Essa alternativa permite que os telefones de um cliente sejam obtidos mais rapidamente, já que encontram-se todos dentro da mesma linha da tabela. Além disso, implica em menos espaço ocupado, já que o espaço necessário à implementação da chave primária da tabela *Telefone*, na primeira alternativa, é considerável.

Do ponto de vista funcional, o inconveniente desta alternativa é que operações que tratam com os telefones de um cliente como uma coleção ficam mais complexos em SQL. Por exemplo, uma eventual consulta usando o número de telefone como critério de busca torna-se mais complicada, já que devem ser referenciados todos os nomes das colunas referentes ao atributo multivalorado.

■ relacionamentos mutuamente exclusivos

Outro caso que permite uma implementação alternativa à especificada pelas regras de projeto é aquele no qual uma entidade participa de forma

[1] No próximo capítulo veremos o conceito de *normalização*.

mutuamente exclusiva de dois ou mais relacionamentos. Participar de forma mutuamente exclusiva significa que uma ocorrência da entidade que participa de um dos relacionamentos em questão, não participa dos demais relacionamentos. Um exemplo é apresentado na Figura 5.24. Pode-se supor que uma ocorrência da entidade *VENDA* participe de exatamente um dos dois relacionamentos e de somente um deles. Observe que esta informação não está contida no diagrama, já que não há uma convenção para registrá-la no DER[2].

A implementação para este modelo, caso forem seguidas as regras apresentadas, é a seguinte:

```
PessFis(CPF,Nome)
PessJur(CNPJ,RazSoc)
Venda (No,data,CPF,CPF)
    CPF referencia PessFis
    CNPJ referencia PessJur
```

Nesta implementação, as colunas `CPF` e `CNPJ` são especificadas como opcionais, já que, em cada linha, um ou outro campos serão vazios. Aparecem

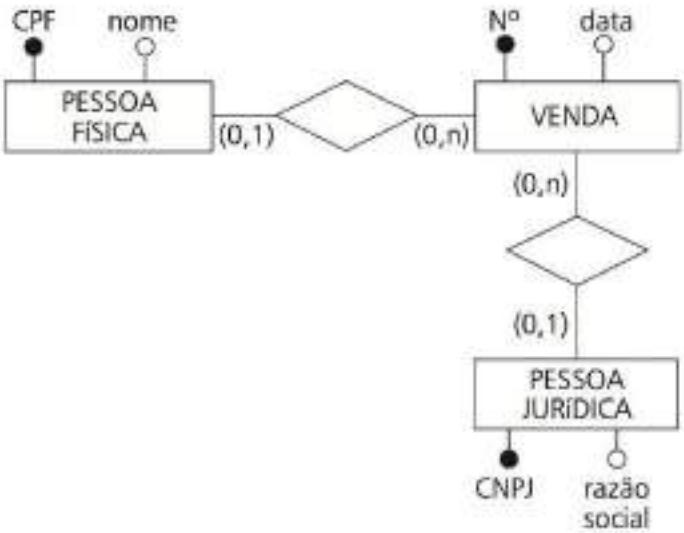


Figura 5.24 Relacionamentos mutuamente exclusivos.

[2] Algumas convenções de diagramas ER prevêem notações específicas para a mútua exclusão entre relacionamentos.

assim os problemas típicos de colunas opcionais. Uma implementação alternativa é criar uma única coluna na qual aparece o CPF ou o CNPJ do comprador, conforme mostrado abaixo.

```
PessFis(CPF,Nome)
PessJur(CNPJ,RazSoc)
Venda (No,data,CPF/CNPJ,TipoCompr)
```

Além da fusão das colunas CPF e CNPJ, deve ser criada uma coluna TipoCompr, que informa se o campo na coluna CPF/CNPJ é referente a um comprador pessoa física ou a uma pessoa jurídica. Através desta alternativa evita-se o uso de colunas opcionais.

Entretanto, a alternativa apresenta igualmente uma desvantagem pois não é possível especificar ao SGBD que o campo CPF/CNPJ é chave estrangeira, já que ele não referencia uma única tabela, mas duas (PessFis e PessJur), de forma alternativa de acordo com o valor do campo TipoCompr.

Apresentadas as alternativas de projeto acima, cabe reforçar a observação que já foi feita anteriormente: alternativas de implementação que fogem às regras de projeto lógico apresentadas somente devem ser tentadas em último caso, quando outras alternativas para a melhoria do desempenho não solucionarem o problema. Normalmente, bases de dados que não foram projetadas de acordo com as regras implicam em custos mais altos de desenvolvimento e de manutenção das aplicações desenvolvidas sobre o banco de dados.

5.3 → engenharia reversa de modelos relacionais

De forma geral, um processo de *engenharia reversa* pode ser definido como um processo de abstração, que parte de um modelo de implementação e resulta em um modelo conceitual, que descreve abstratamente a implementação em questão. O termo *engenharia reversa* vem do fato de usar-se, como ponto de partida do processo, um produto implementado (o modelo de implementação) para obter sua especificação (o modelo conceitual).

No caso de banco de dados, fala-se de engenharia reversa quando modelos de dados mais ricos em detalhes de implementação são transformados em modelos de dados mais abstratos.

Um caso específico de engenharia reversa de banco de dados é o da *engenharia reversa de modelos relacionais*. Neste tipo de engenharia reversa, tem-se, como ponto de partida, um modelo lógico de um banco de dados relacional e, como resultado, um modelo conceitual, no caso deste livro, na abordagem ER. Este é o processo inverso ao de projeto lógico (Figura 5.1).

A engenharia reversa de modelos relacionais pode ser útil quando não se tem um modelo conceitual para um banco de dados existente. Isso pode acontecer quando o banco de dados foi desenvolvido de forma empírica, sem o uso de uma metodologia de desenvolvimento, ou quando o esquema do banco de dados sofreu modificações ao longo do tempo, sem que as mesmas tenham sido registradas no modelo conceitual.

Como veremos no capítulo 6, a engenharia reversa de modelos relacionais é um passo dentro de um processo mais amplo de engenharia reversa de arquivos e documentos convencionais.

O processo de engenharia reversa de um modelo relacional consta dos seguintes passos:

1 Identificação da construção ER correspondente a cada tabela
2 Definição de relacionamentos **1:n** e **1:1**
3 Definição de atributos
4 Definição de identificadores de entidades e relacionamentos

Estes passos são detalhados nas seções que seguem.

O processo será exemplificado sobre um banco de dados para um sistema acadêmico, cujo esquema é apresentado na Figura 5.25.

5.3.1 identificação da construção ER correspondente a cada tabela

Na primeira etapa da engenharia reversa de um banco de dados relacional, define-se, para cada tabela do modelo relacional, qual a construção que lhe corresponde no modelo ER.

Uma tabela pode corresponder a:

- uma entidade,

```
Disciplina(CodDisc,NomeDisc)
Curso(CodCr,NomeCr)
Curric(CodCr,CodDisc,Obr/Opc)
    CodCr referencia Curso
    CodDisc referencia Disciplina
Sala(CodPr,CodSl,Capacidade)
    CodPr referencia Prédio
Prédio(CodPr,Endereço)
Turma(Anosem,CodDisc,SiglaTur,Capacidade,CodPr,CodSl)
    CodDisc referencia Disciplina
    (CodPr,CodSl) referencia Sala
Laboratório (CodPr,CodSl,Equipam)
    (CodPr,CodSl) referencia Sala
```

Figura 5.25 Esquema de banco de dados acadêmico para engenharia reversa.

- um relacionamento **n:n**,
- uma entidade especializada.

O fator que determina qual a construção ER que corresponde a uma tabela é a composição de sua chave primária. Tabelas podem ser classificadas em três tipos de acordo com sua chave primária:

- `Regra 1: Chave primária composta por mais de uma chave estrangeira` – A tabela que possui uma chave primária composta de múltiplas chaves estrangeiras implementa um *relacionamento* ***n:n*** entre as entidades correspondentes às tabelas referenciadas pelas chaves estrangeiras. Um exemplo de tabela deste tipo é `Curric`, que tem como chave primária `CodCr` e `CodDisc`. Ambas as colunas são chave estrangeira em relação às tabelas `Curso` e `Disciplina` respectivamente. Portanto, a tabela `Curric` representa um relacionamento entre as entidades correspondentes às tabelas `CodCr` e `CodDisc`. No exemplo, a única tabela deste tipo é a tabela `Curric`.
- `Regra 2: A chave primária completa forma uma chave estrangeira` – Quando toda chave primária (todas as suas colunas) compõe uma única chave estrangeira, a tabela representa uma entidade que forma uma *especialização* da entidade correspondente à tabela referenciada pela chave estrangeira. Um exemplo de tabela deste tipo é a tabela `Laboratório`, que possui como chave primária as colunas `CodPr` e `CodSl`, as quais são chave estrangeira da tabela de salas. A restrição de integridade referencial em questão especifica que uma linha na tabela de laboratórios somente existe quando uma linha com a mesma chave existir na tabela de salas. No

modelo ER, isso significa que uma ocorrência da entidade laboratório somente pode existir quando a ocorrência correspondente da entidade sala existir, ou seja, significa que a entidade laboratório é uma especialização de sala. No exemplo, a única tabela deste tipo é a tabela `Laboratório`.

- `Regra 3: Demais casos` – Quando a chave primária da tabela não for composta de múltiplas chaves primárias (Regra 1), nem for toda ela uma chave estrangeira (Regra 2), a tabela representa uma *entidade*. Exemplificando, a tabela `Curso`, cuja chave primária, a coluna `CodCr` não contém chaves estrangeiras, representa uma entidade. Da mesma forma, a tabela `Sala` também representa uma entidade. Sua chave primária (colunas `CodPr` e `CodSl`) contém apenas uma chave estrangeira (coluna `CodSl`). Assim, não obedece ao requisito da multiplicidade de chaves estrangeiras (Regra 1), nem ao requisito de toda chave primária ser estrangeira (Regra 2) e enquadra-se na presente regra. O mesmo é válido para as tabelas `Disciplina`, `Prédio` e `Turma`.

Tendo feito a classificação de tabelas segundo a composição da chave primária e com isso identificado as construções de ER correspondentes a cada tabela, é possível montar um diagrama ER inicial, conforme mostra a Figura 5.26.

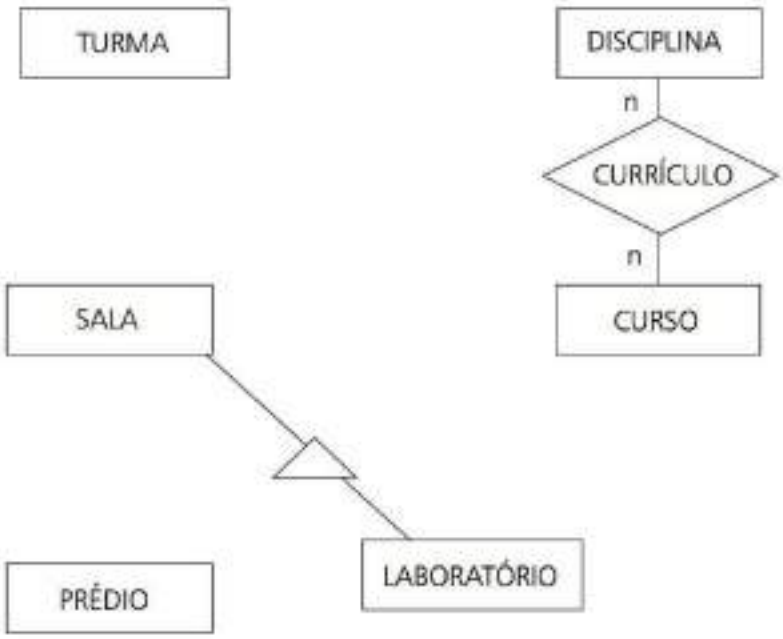


Figura 5.26 Diagrama ER inicial para o BD acadêmico.

5.3.2 definição de relacionamentos 1:n ou 1:1

Toda chave estrangeira que não se enquadra nas regras 1 e 2 apresentadas na seção anterior representa um relacionamento **1:n** ou **1:1**. Em outros termos, toda chave estrangeira que não corresponde a um relacionamento **n:n** (Regra 1), nem a uma entidade especializada (Regra 2) representa um relacionamento **1:n** ou **1:1**.

Assim, nesta regra, estão sendo tratadas chaves estrangeiras que:

- contêm colunas que não fazem parte de chaves primárias, ou
- fazem parte de uma chave primária, e esta chave primária não contém outras chaves estrangeiras e contém colunas que não são chaves estrangeiras.

A regra nem sempre permite definir se a cardinalidade do relacionamento é **1:n** ou **1:1**. Quando a chave estrangeira é parte de uma chave primária, trata-se de um relacionamento **1:n**. Entretanto, nos demais casos, para definir a cardinalidade, é necessário verificar os possíveis conteúdos do banco de dados.

No exemplo, as chaves estrangeiras que representam relacionamentos **1:n** ou **1:1** são as seguintes:

Tabela Sala:

```
CodPr referencia Prédio
```

Tabela Turma:

```
CodDisc referencia Disciplina
(CodPr,CodSl) referencia Sala
```

Com este passo, podemos completar a definição dos relacionamentos no diagrama ER, conforme mostra a Figura 5.27. No exemplo, todos os relacionamentos referentes às chaves estrangeiras acima são do tipo **1:n**.

5.3.3 definição de atributos

Neste passo, para cada coluna que não seja chave estrangeira, é definido um atributo na entidade/relacionamento correspondente à tabela que contém a coluna. Observe-se que colunas de chaves estrangeiras não correspondem a atributos no diagrama ER, mas sim a relacionamentos, e por isso já foram

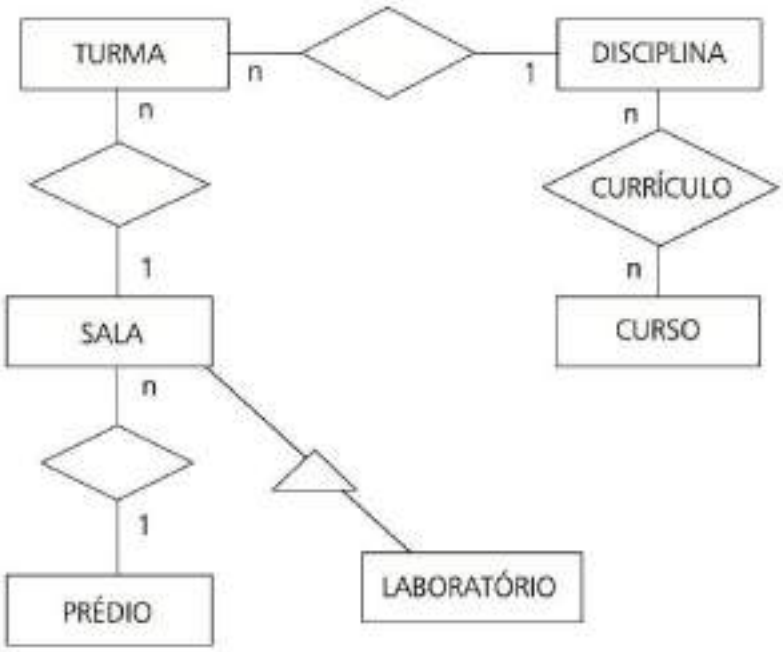


Figura 5.27 Definição dos relacionamentos **1:n** e **1:1**.

tratadas nas etapas anteriores. Para o exemplo, a execução deste passo da engenharia reversa resulta no diagrama ER da Figura 5.28.

5.3.4 definição de identificadores de entidades

No último passo da engenharia reversa, são definidos os identificadores das entidades e dos relacionamentos. A regra para a definição dos identificadores é a seguinte:

- `Coluna da chave primária que não é chave estrangeira` – Toda coluna que faz parte da chave primária e que *não* é chave estrangeira corresponde a um *atributo identificador* da entidade ou relacionamento.
- `Coluna da chave primária que é chave estrangeira` – Toda coluna que faz parte da chave primária e que *é* chave estrangeira corresponde a um identificador externo da entidade. Exemplificando, a coluna `CodDisc`, que é parte da chave primária da tabela `Turma` é também chave estrangeira em relação à tabela `Disciplina`. Portanto, a entidade *TURMA* é identificada também pelo relacionamento com *DISCIPLINA*.

Executando este passo da engenharia reversa sobre o modelo do exemplo chegamos à Figura 5.29.

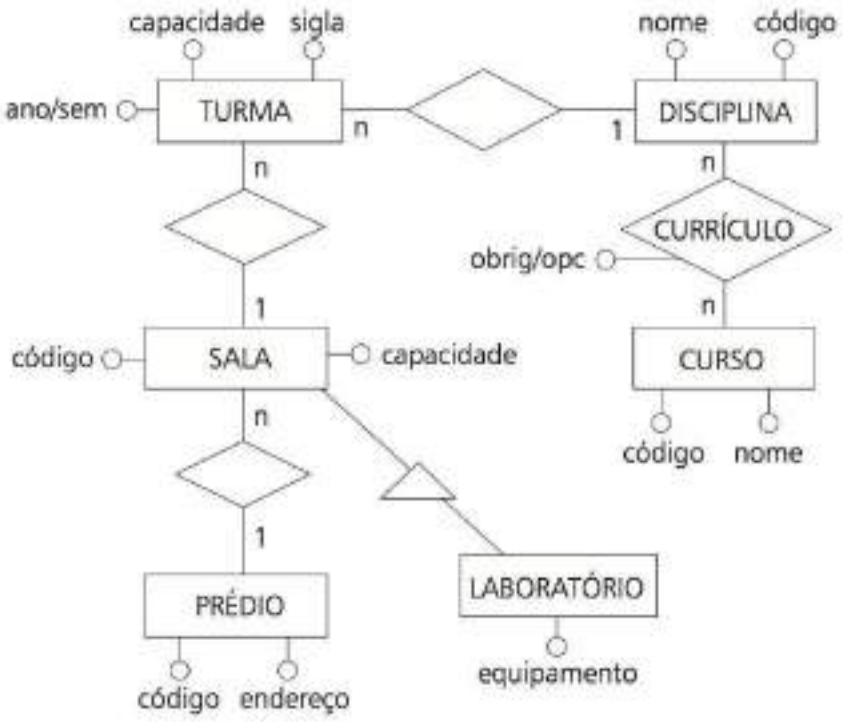


Figura 5.28 Definição dos atributos.

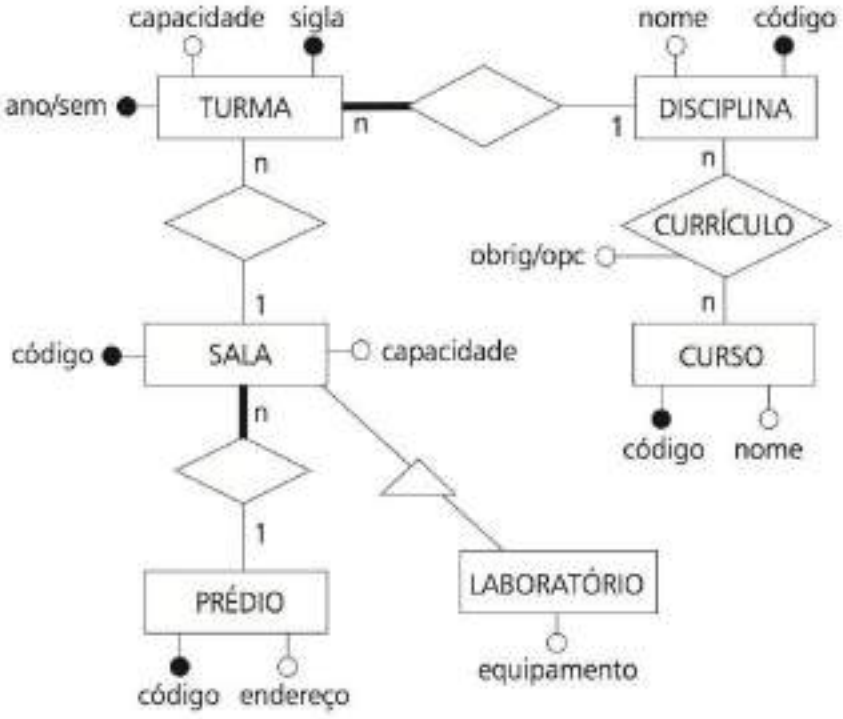


Figura 5.29 Definição dos identificadores de entidades.

5.4 → exercícios

exercício 1 Considere as seguintes alternativas de implementação de um banco de dados relacional:

Alternativa 1:

```
Aluno(CodAl,Nome,CodCurso,Endereco)
```

Alternativa 2:

```
Aluno(CodAl,Nome,CodCurso)
EnderecoAluno(CodAl,Endereco)
    CodAl referencia Aluno
```

Em ambos os casos está sendo representado um conjunto de alunos e informações (código, nome, código de curso, endereço) a ele referentes. À luz dos princípios que baseiam as regras de tradução de diagramas ER para modelo relacional, discuta qual das duas alternativas é preferível.

exercício 2 Usando as regras de transformação de modelos ER para modelo lógico relacional apresentadas neste capítulo, projete um BD relacional para o modelo ER da Figura 5.30. Para não sobrecarregar o diagrama os atributos das entidades são listados a seguir. Os atributos identificadores estão sublinhados.

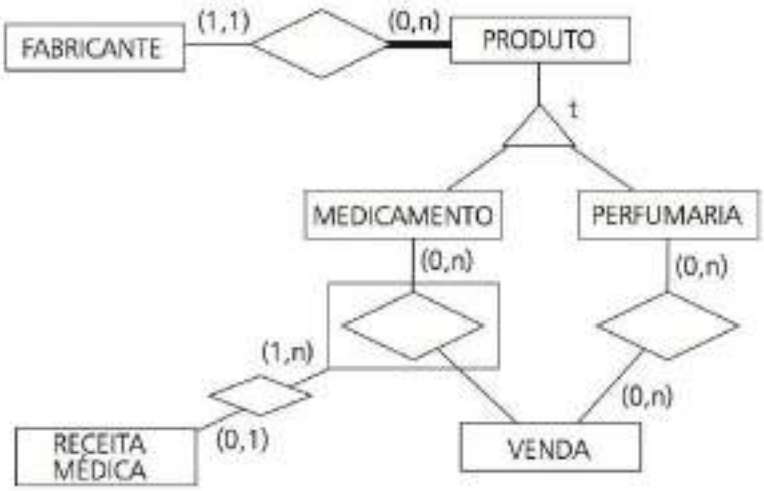


Figura 5.30 Modelo ER para uma farmácia.

```
Produto(Número, NomeComercial, TipoEmbalagem,
        Quantidade, PreçoUnitário)
Fabricante(CNPJ,Nome,Endereço)
Medicamento(Tarja,Fórmula)
Perfumaria(Tipo)
Venda(Data,NúmeroNota,NomeCliente,CidadeCliente)
PerfumariaVenda(Quantidade,Imposto)
MedicamentoReceitaVenda(Quantidade,Imposto)
ReceitaMédica(CRM,Número,Data)
```

exercício 3 Usando as regras de transformação de modelos ER para modelo lógico relacional apresentadas neste capítulo, projete um BD relacional para o modelo ER da Figura 5.31. Para não sobrecarregar o diagrama os atributos das entidades são listados a seguir. Os atributos identificadores estão sublinhados.

```
Escritório(Número, Local)
Cliente(NúmeroCartMotorista, EstadoCartMotorista,
Nome, Endereço, Telefone)
Contrato aluguel(Número, Data, Duração)
Veículo(Número, DataPróximaManutenção, Placa)
Tipo de Veículo(Código, Nome, ArCondicionado)
Automóvel(NúmeroPortas, DireçãoHidráulica,
          CâmbioAutomático,Rádio)
Ônibus(NúmeroPassageiros, Leito, Sanitário)
```

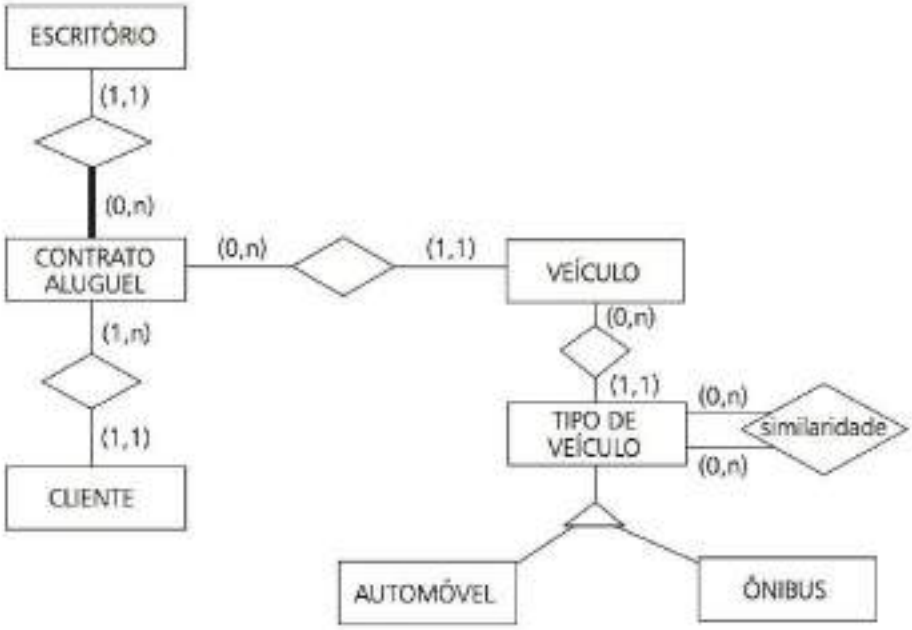


Figura 5.31 Modelo ER para locadora de veículos.

exercício 4 Abaixo é apresentado um esquema lógico de um BD relacional que armazena dados sobre produtos e vendas em uma loja. Usando as regras de engenharia reversa apresentadas anteriormente, construa um diagrama ER para este BD.

```
Produto(CodigoTipoProd,NumeroProd,DescricaoProd,PreçoProd)
   CodigoTipoProd referencia TipoProd
```

/* tabela de produtos de uma loja – `CodigoTipoProd` é o código do tipo do produto, `NumeroProd` é seu código, `DescriçãoProd` é uma descrição do produto e `PreçoProd` é seu preço */

```
Similaridade(CodigoTipoProd,NumeroProd,
             CodigoTipoProdSim,NumeroProdSim
   (CodigoTipoProd,NumeroProd) referencia Produto
   (CodigoTipoProdsim,NumeroProdSim) referencia Produto
```

/* tabela de similaridade de produtos – para cada produto, informa quais são seus produtos similares*/

```
TipoProd(CodigoTipoProd,DescricaoTipoProd)
```

/* tabela de tipos de produtos, com código e descrição */

```
Venda(NúmeroNF,DataVenda,CodReg,CodEmp)
  (CodigoReg) referencia Registradora
  (CodEmp) referencia Empregado
```

/* tabela que informa as vendas que ocorreram na loja – informa o número da nota fiscal, a data da venda, a registradora na qual ocorreu, bem como o empregado que a realizou */

```
ItemVenda(NúmeroNF,CodigoTipoProd,NumeroProd,
          QtdeItem,PreçoItem)
  (NúmeroNF) referencia Venda
  (CodigoTipoProd,NumeroProd) referencia Produto
```

/* tabela com informações dos itens de uma venda, isto é, que produtos e em que quantidade e com que preço foram vendidos em uma venda */

```
Registradora(CodReg, SaldoReg)
```

/* tabela com código e saldo de cada registradora na loja */

```
Empregado(CodEmp, NomeEmp, SenhaEmp)
```

/* tabela com código, nome e senha de cada empregado da loja */

exercício 5 Abaixo é apresentado um esquema lógico de um BD relacional que armazena dados genealógicos. Usando as regras de engenharia reversa apresentadas anteriormente, construa um diagrama ER para este BD.

```
Pessoa(PessID, PessNome, NascLocID, DataNasc,
       FalecLocID, DataFalec, ProfID, FilhoCasamID, Sexo)
   NascLocID referencia Local
   FalecLocID referencia Local
   ProfID referencia Profiss
   FilhoCasamID referencia Casam
```

/* Tabela de pessoas: contém o identificador da pessoa, seu nome, local (identificador) e data de nascimento, local (identificador) e data de falecimento, profissão, identificador do casamento que gerou a pessoa e sexo */

```
Local(LocID,Cidade,País)
```

/* Tabela de locais */

```
Profiss(ProfID,ProfNome)
```

/* Tabela de profissões */

```
Casam(CasamID, MaridoPessID, EsposaPessID,
      DataCasam, CasamLocID)
   MaridoPessID referencia Pessoa
   EsposaPessID referencia Pessoa
   CasamLocID referencia Local
```

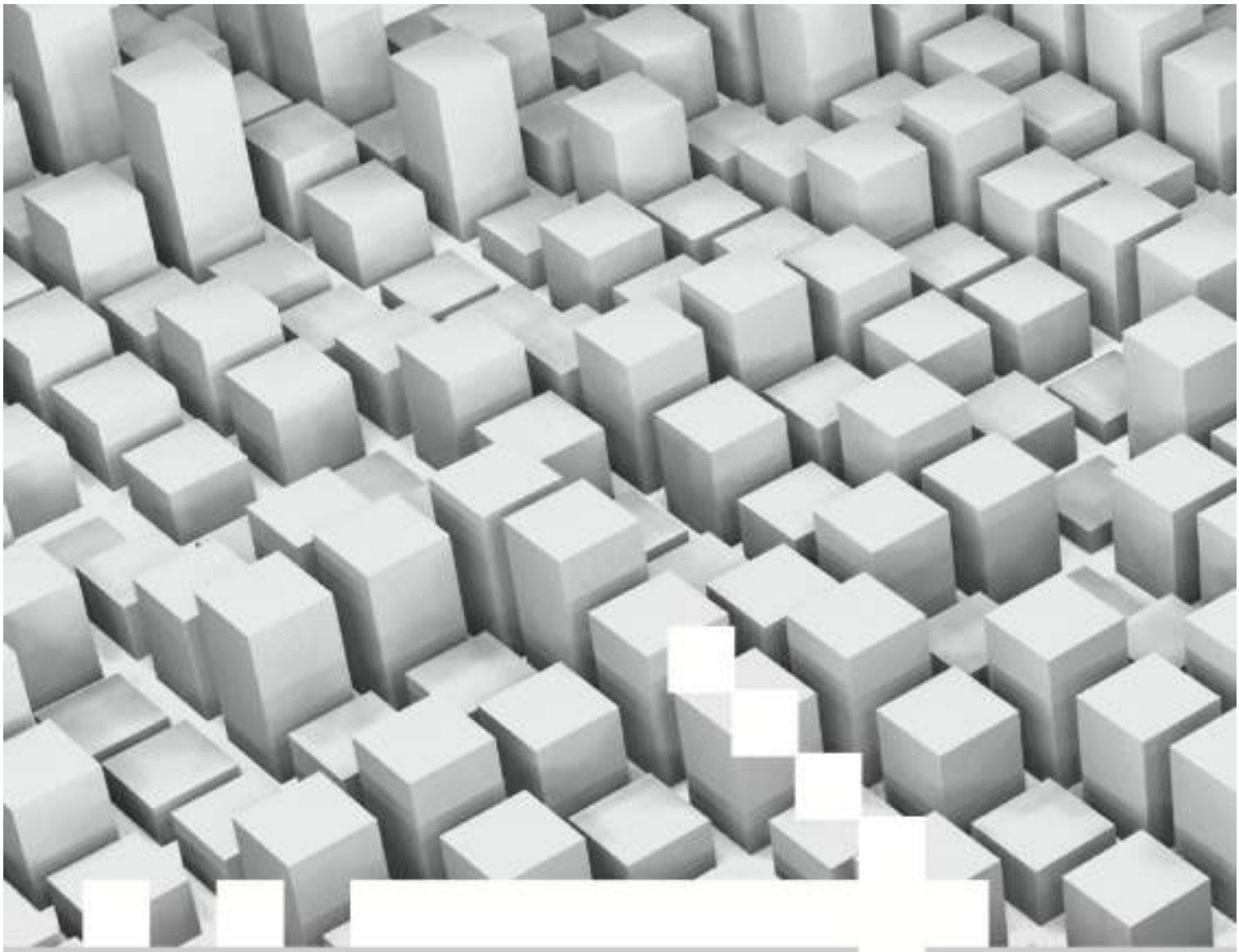
/* Tabela de casamentos: contém identificador do casamento, identificador do marido, identificador da esposa, data do casamento e local (identificador) */

5.5 → leituras recomendadas

Muitos dos livros que tratam da abordagem relacional dedicam um capítulo ao projeto de banco de dados relacional a partir de modelos ER. Este é o caso dos livros ELMASRI; NAVATHE (2002) e SILBERSCHATZ; KORTH; SUDARSHAN (1999) que já havíamos citado no capítulo referente a abordagem relacional.

Da mesma forma, livros sobre modelagem e projeto de banco de dados (BATINI; CERI; NAVATHE (1992); SETZER (1987); TEOREY (1994)) abordam o problema do projeto lógico. O livro de BATINI; CERI; NAVATHE (1992) contém uma cobertura completa, tanto do projeto do banco de dados, quanto da engenharia reversa a partir de modelos relacionais.

Os artigos DUMPALA; ARORA (1983), TEOREY; YANG; FRY: (1986) e WONG; KATZ (1979) são os pioneiros no projeto lógico de banco de dados relacionais a partir de modelos ER. Já os artigos JOHANNESSON; KALMAN (1989) e NAVATHE; AWONG (1987) apresentam as idéias básicas sobre como deve ser feita a engenharia reversa de modelos relacionais para modelos ER.



capítulo 6

engenharia reversa de arquivos e normalização

No capítulo 5, foi apresentado um processo de engenharia reversa de modelos relacionais, que permite obter um modelo conceitual para um modelo lógico relacional.

Neste capítulo, veremos a engenharia reversa de arquivos, que possibilita a obtenção de um modelo lógico relacional a partir do modelo lógico de um banco de dados não-relacional. Ele permite a engenharia reversa de qualquer conjunto de dados para os quais se disponha de uma descrição, como documentos, arquivos manuais, arquivos convencionais em computador ou bancos de dados gerenciados por SGBD não-relacional.

Como base teórica para este processo será usado o conceito de normalização, uma técnica que elimina a redundância de dados de arquivos.

6.1 → introdução

Muitos dos sistemas de informação hoje usados foram desenvolvidos ao longo dos últimos 20 anos e não utilizam bancos de dados relacionais, sendo chamados de *sistemas legados* (*legacy systems*). Os dados desses sistemas estão armazenados em arquivos de linguagens de terceira geração, como COBOL ou Basic, ou então em bancos de dados da era pré-relacional, como IMS ou ADABAS. Raramente, os arquivos destes sistemas estão documentados através de modelos conceituais.

Adicionalmente, há bancos de dados relacionais que não possuem documentação na forma de um modelo conceitual.

No entanto, há situações no ciclo de vida de um sistema nas quais um modelo conceitual é de grande valia.

Um exemplo é a manutenção rotineira do *software* de um sistema de informações. Neste caso, o modelo conceitual pode ser usado como documentação abstrata dos dados, durante discussões entre usuários, analistas e programadores. A existência de um modelo conceitual permite que pessoas que não conheçam o sistema possam aprender mais rapidamente o seu funcionamento.

Outro exemplo no qual é importante possuir o modelo conceitual de um banco de dados já implementado é o da migração do banco de dados para uma nova plataforma de implementação, por exemplo, usando um SGBD relacional. A disponibilidade de uma documentação abstrata, na forma de um modelo conceitual dos dados do sistema existente, pode acelerar o processo de migração, pois permite encurtar a etapa de modelagem de dados do novo banco de dados.

O presente capítulo mostra como executar o processo de engenharia reversa de arquivos convencionais ou de banco de dados pré-relacionais. De forma geral, o processo apresentado serve para a construção de um modelo entidade-relacionamento dentro da estratégia *ascendente*, estratégia de modelagem recomendada para o caso em que se parte de descrições de dados existentes (Seção Partindo de descrições de dados existentes, na página 102).

6.2 → visão geral do processo de engenharia reversa

A Figura 6.1 apresenta uma visão geral do processo de engenharia reversa de arquivos convencionais. O processo parte das descrições dos arquivos que compõem o sistema existente. Portanto, é um processo que segue a estratégia ascendente de construção de modelos ER referida na Seção Partindo de descrições de dados existentes (página 102).

O primeiro passo é a representação da descrição de cada arquivo existente na forma de um esquema de uma tabela relacional não-normalizada. Este

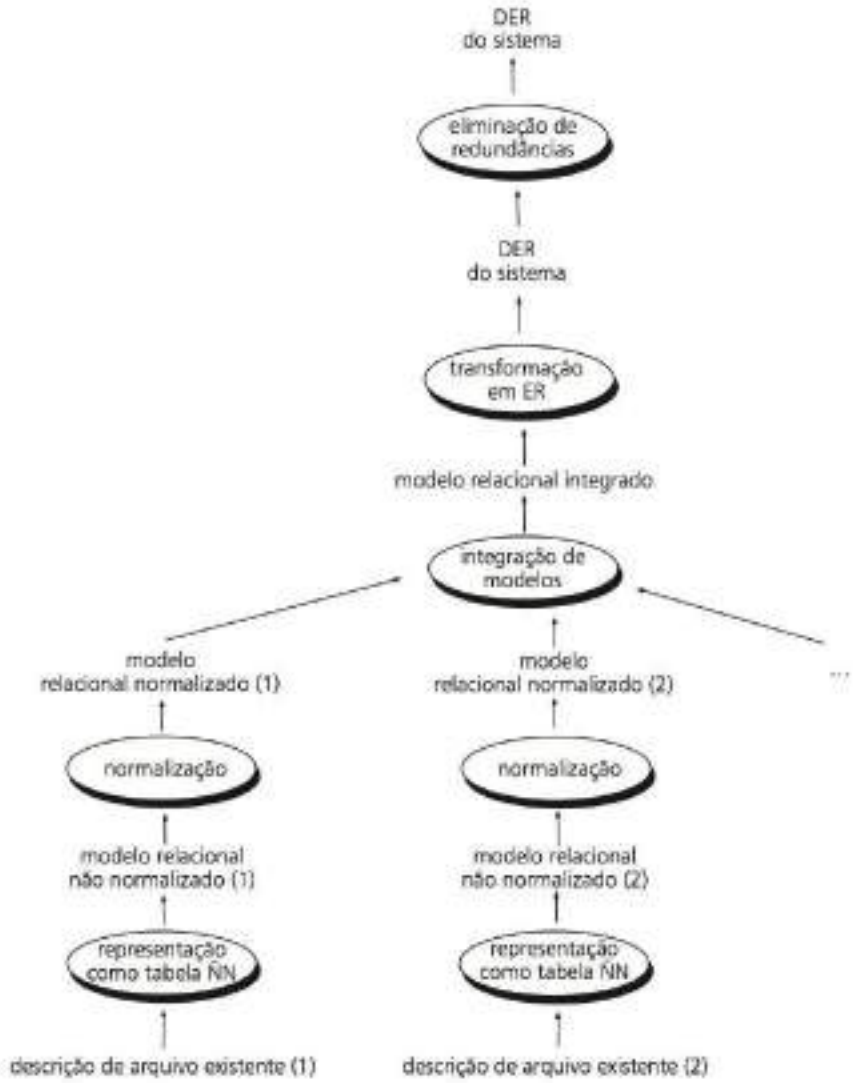


Figura 6.1 Visão geral do processo de engenharia reversa.

primeiro passo obtém uma descrição independente do tipo de arquivo que está sendo utilizado. A partir daí, o processo trabalha apenas com tabelas relacionais, o que o torna independente do tipo de arquivo que está sendo usado como entrada ao processo de normalização.

A seguir, este esquema de tabela não-normalizada passa por um processo conhecido por *normalização*, através do qual é obtido um modelo relacional, contendo as descrições das tabelas correspondentes ao arquivo em questão. O objetivo do processo de normalização é:

- reagrupar informações de forma a eliminar redundâncias de dados que possam existir nos arquivos, e
- reagrupar informações de uma forma que permita a obtenção de um modelo ER.

Uma vez normalizados todos os arquivos do sistema, os diferentes esquemas relacionais obtidos pela normalização são integrados, gerando o esquema relacional do banco de dados do sistema. Nesta etapa, as informações comuns a diferentes arquivos são identificadas e representadas uma única vez.

Finalmente, a partir do esquema relacional obtido, usando as regras para a engenharia reversa de um modelo relacional mostradas no capítulo anterior, temos o modelo ER do sistema existente.

6.3 → documento exemplo

O processo de normalização pode ser executado sobre qualquer tipo de representação de dados. Pode partir da descrição de um arquivo em computador, do *lay-out* de um relatório ou de uma tela, etc.

Para exemplificar o processo de normalização descrito a seguir, usamos um documento (Figura 6.2), que poderia fazer parte de um sistema de gerência de projetos. Para cada projeto, são informados o código, a descrição e o tipo do projeto, bem como os empregados que atuam no projeto. Já para cada empregado do projeto, são informados o seu número, nome e categoria funcional, bem como a data em que o empregado foi alocado ao projeto e o tempo pelo qual o empregado foi alocado ao projeto. A Figura 6.2 apresenta um exemplo de um possível conteúdo deste documento.

RELATÓRIO DE ALOCAÇÃO A PROJETO

CÓDIGO DO PROJETO: LSC001 *TIPO:* Novo Desenv.

DESCRIÇÃO: Sistema de Estoque

CÓDIGO DO EMPREGADO	*NOME*	*CATEGORIA FUNCIONAL*	*SALÁRIO*	*DATA DE INÍCIO NO PROJETO*	*TEMPO ALOCADO AO PROJETO*
2146	João	A1	4	1/11/91	24
3145	Sílvio	A2	4	2/10/91	24
6126	José	B1	9	3/10/92	18
1214	Carlos	A2	4	4/10/92	18
8191	Mário	A1	4	1/11/92	12

CÓDIGO DO PROJETO: PAG02 *TIPO:* Manutenção

DESCRIÇÃO: Sistema de RH

CÓDIGO DO EMPREGADO	*NOME*	*CATEGORIA FUNCIONAL*	*SALÁRIO*	*DATA DE INÍCIO NO PROJETO*	*TEMPO ALOCADO AO PROJETO*
8191	Mário	A1	4	1/05/93	12
4112	João	A2	4	4/01/91	24
6126	José	B1	9	1/11/92	12

Figura 6.2 Exemplo de documento a ser normalizado.

6.4 → representação na forma de tabela não-normalizada

O primeiro passo do processo de engenharia reversa consta em transformar a descrição do documento ou arquivo a ser normalizado em um esquema de uma tabela relacional. A Figura 6.3 apresenta a tabela correspondente ao documento da Figura 6.2. Essa tabela é dita *não-normalizada* (ou mais precisamente *não-primeira-forma-normal*) pois possui uma *tabela aninhada*. Usaremos a abreviatura ÑN para identificar esta forma de representar a tabela. Uma *tabela aninhada* (também chamada por outros autores de *grupo repetido* ou *coluna multivalorada* ou ainda *coluna não atômica*) é uma tabela que aparece como valor de campo dentro de outra tabela.

No caso da Figura 6.3, na coluna `Emp`, aparece, para cada linha de departamento, uma tabela aninhada, que contém os dados dos empregados referentes ao departamento em questão. O conceito de tabela aninhada é o correspondente, na abordagem relacional, aos conceitos de vetor ou *array* de

CódProj	Tipo	Descr	Emp					
			CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	Novo Desenv.	Sistema de Estoque	2146	João	A1	4	1/11/91	24
			3145	Silvio	A2	4	2/10/91	24
			6126	José	B1	9	3/10/92	18
			1214	Carlos	A2	4	4/10/92	18
			8191	Mário	A1	4	1/11/92	12
PAG02	Manutenção	Sistema de RH	8191	Mário	A1	4	1/05/93	12
			4112	João	A2	4	4/01/91	24
			6126	José	B1	9	1/11/92	12

Figura 6.3 Documento exemplo na forma de tabela não-normalizada.

linguagens, como Java e Pascal, ou de item de grupo repetido (especificado com cláusula OCCURS) da linguagem COBOL.

A tabela da Figura 6.3 pode ser descrita pelo seguinte esquema de tabela relacional:

```
Proj (CodProj,Tipo, Descr,
            (CodEmp,Nome, Cat, Sal, DataIni, TempAl))
```

No esquema acima, foi usada uma extensão da notação de esquema relacional apresentada no capítulo 5. A extensão consta do uso de parênteses aninhados para descrever tabelas aninhadas. No caso de tabelas aninhadas, as colunas sublinhadas indicam a coluna ou grupo de colunas que servem para distinguir diferentes linhas da tabela aninhada referente a uma linha da tabela de seu nível externo. Assim, a coluna `CodProj` distingue as linhas (cada uma referente a um projeto) da tabela `Proj`. Já a coluna `CodEmp` serve para distinguir diferentes linhas de empregado dentro do grupo referente a um projeto.

Para apresentar outro exemplo de representação não-normalizada de documentos, mostramos, nas Figuras 6.4 e 6.5, a definição de um arquivo convencional que contém dados de alunos usando as notações de Pascal e de COBOL, respectivamente.

O arquivo contém dados referentes a alunos de uma universidade. Cada registro contém informações referentes a um aluno. O registro do aluno contém seu código, seu nome, uma lista de ingressos em cursos e uma lista de disciplinas cursadas. A lista de ingressos em curso contém o código do curso em que o

```
type reg_aluno = record
    cod_al: integer;
    nome_al: char_60;
    ingressos_cursos_al: array [1..10] of record
        cod_curso: integer;
        semestre_ingresso: integer
        end;
    disciplinas_cursadas_al: array [0..200] of record
        cod_disc: integer;
        semestres_cursados: array [1..20] of record
            semestre_disc: integer;
            nota_disc: integer
            end
    end
  end;
arq_aluno = file of reg_aluno;
```

Figura 6.4 Registro de aluno (Pascal).

```
FD   Arq-Alunos 01   Reg-Al.
    03   Cod-Al
    03   Nome-Al
    03   Ingr-Cursos-al OCCURS 1 TO 10
        05   Cod-Curso
        05   Sem-ingresso
    03   Disc-Curs-Al OCCURS 0 to 200
        05   Cod-Disc
        05   Sem-Cursado OCCURS 1 TO 20
            07   Sem-Disc-Cursada
            07   Nota-Disc
```

Figura 6.5 Registro de aluno (COBOL).

aluno ingressou, junto com o semestre de ingresso no curso. A lista de disciplinas cursadas contém uma entrada para cada disciplina que o aluno cursou. Para cada disciplina cursada, é informado seu código, junto com os diversos semestres em que o aluno cursou a disciplina. Para cada semestre, é informado o semestre e a nota que o aluno obteve na disciplina no semestre em questão.

Para o arquivo em questão, a representação não-normalizada seria a seguinte:

```
Arq-Alunos(Cod-Al,Nome-Al,
                (Cod-Curso,Sem-ingresso)
                (Cod-Disc,
                        (Sem-Disc-Cursada, Nota-Disc)))
```

Observe que as descrições de arquivos não fornecem as colunas que são chave primária, já que este conceito não está presente nas linguagens referidas. Cabe observar ainda que, na representação não-normalizada, não são associados nomes às tabelas aninhadas, já que estes não são necessários para o processo de engenharia reversa.

6.5 → normalização

Obtido o esquema relacional correspondente ao documento, passa-se ao processo de *normalização*. Este processo baseia-se no conceito de *forma normal*. Uma forma normal é uma regra que deve ser obedecida por uma tabela para que esta seja considerada "bem projetada". Há diversas formas normais, isto é, diversas regras, cada vez mais rígidas, para verificar tabelas relacionais. Neste trabalho, vamos considerar quatro formas normais, denominadas simplesmente primeira, segunda, terceira e quarta forma normal, abreviadamente 1FN, 2FN, 3FN e 4FN.

6.5.1 passagem à primeira forma normal (1FN)

primeira forma normal (1FN)
=
diz-se que uma tabela está na primeira forma normal quando ela não contém tabelas aninhadas

O primeiro passo da normalização é a transformação do esquema de tabela não-normalizada em um esquema relacional na primeira forma normal. Uma tabela encontra-se na 1FN quando ela não contém tabelas aninhadas. Portanto, a passagem à 1FN consta da eliminação das tabelas aninhadas eventualmente existentes.

Para transformar um esquema de tabela não-normalizada em um esquema na 1FN há duas alternativas:

- `Construir uma única tabela com redundância de dados` – Cria-se uma tabela na qual os dados das linhas externas à tabela aninhada são repetidos para cada linha da tabela aninhada. No caso da tabela da Figura 6.3, o esquema resultante seria o seguinte:

```
ProjEmp(CodProj,Tipo,Descr,CodEmp,Nome,Cat,Sal, DataIni,TempAl)
```

Nesta tabela, os dados do projeto aparecem repetidos para cada linha da tabela de empregados.

- `Construir uma tabela para cada tabela aninhada` – Cria-se uma tabela referente a própria tabela que está sendo normalizada e uma tabela para cada tabela aninhada. No caso da tabela da Figura 6.3, o esquema resultante seria o seguinte:

```
Proj    (CodProj,Tipo,Descr)
ProjEmp(CodProj,CodEmp,Nome,Cat,Sal,DataIni,TempAl)
```

Considerando apenas a correção do processo de normalização, a primeira alternativa (tabela única) é a preferida. Ao decompor uma tabela em várias tabelas, como ocorre na segunda alternativa, podem ser perdidas relações entre informações.

Entretanto, para fins práticos, preferimos a segunda alternativa (*decomposição de tabelas*), mesmo sabendo que ela pode levar a modelos imperfeitos. No caso de uma tabela não-normalizada, como a do exemplo, que possui apenas uma tabela aninhada, fica fácil visualizar a tabela na 1FN, na primeira alternativa. Entretanto, quando houver diversas tabelas aninhadas, eventualmente com diversos níveis de aninhamento, fica difícil visualizar a tabela na 1FN. Para verificar o tipo de problema que pode ser causado pela decomposição em várias tabelas, o leitor pode estudar o Exercício 17 deste capítulo.

A passagem à primeira forma normal por decomposição de tabelas é feita nos seguintes passos:

1. É criada uma tabela na 1FN referente à tabela não-normalizada e que contém apenas as colunas com valores atômicos, isto é, sem as tabelas aninhadas. A chave primária da tabela na 1FN é idêntica à chave da tabela não-normalizada.
2. Para cada tabela aninhada, é criada uma tabela na 1FN composta pelas seguintes colunas:
   - a chave primária de cada uma das tabelas nas quais a tabela em questão está aninhada,
   - as colunas da própria tabela aninhada.

3 São definidas as chaves primárias das tabelas na 1FN que correspondem a tabelas aninhadas.

Conforme mencionado, na tabela do exemplo, a decomposição gera duas tabelas com o seguinte esquema:

```
1FN
Proj (CodProj,Tipo,Descr)
ProjEmp (CodProj,CodEmp,Nome,Cat,Sal,DataIni,TempAl)
```

O conteúdo destas tabelas, caso considerarmos o conteúdo do documento original que está sendo normalizado, é apresentado na Figura 6.6.

Observe que a tabela `ProjEmp` que refere-se à tabela aninhada contida na tabela não-normalizada de partida. Ela contém as seguintes colunas:

- Coluna `CodProj`, que é a chave primária da tabela não-normalizada externa à tabela aninhada em questão.
- Colunas da tabela aninhada em questão.

Já a chave primária da tabela `ProjEmp` é composta pelas colunas `CodProj` e `CodEmp`. Isto ocorre pelo fato de o mesmo empregado poder trabalhar em múltiplos projetos. Assim, na tabela `ProjEmp` aparecem múltiplas linhas para

Proj:

CódProj	Tipo	Descr
LSC001	Novo Desenv.	Sistema
PAG02	Manutenção	Sistema de RH

ProjEmp:

CódProj	CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	2146	João	A1	4	1/11/91	24
LSC001	3145	Sílvio	A2	4	2/10/91	24
LSC001	6126	José	B1	9	3/10/92	18
LSC001	1214	Carlos	A2	4	4/10/92	18
LSC001	8191	Mário	A1	4	1/11/92	12
PAG02	8191	Mário	A1	4	1/05/93	12
PAG02	4112	João	A2	4	4/01/91	24
PAG02	6126	José	B1	9	1/11/92	12

Figura 6.6 Tabelas referentes ao exemplo na 1FN.

um mesmo empregado e é necessário usar o código do projeto para distingui-las.

Cabe observar que, caso cada empregado trabalhasse em um único projeto, a chave primária seria composta apenas pela coluna `CodEmp`. Neste caso, a coluna `CodProj` não faria parte da chave primária e seria apenas uma das colunas não chave da tabela em questão.

Neste ponto do processo de normalização, pode ser difícil encontrar nomes adequados para as tabelas. No exemplo, os nomes das tabelas foram inspirados nas chaves primárias das tabelas. Assim, a segunda tabela, que tem como chave primária as colunas `CodProj` e `NumEmp` foi denominada `ProjEmp`. Como os nomes das tabelas não são relevantes ao processo de normalização, a recomendação é estabelecer os nomes definitivos das tabelas apenas ao final da normalização.

Outro exemplo de passagem à primeira forma normal é o do arquivo de alunos visto acima (Figuras 6.4 e 6.5). Conforme mostrado anteriormente, sua representação não-normalizada é a seguinte:

ÑN

```
Arq-Alunos(Cod-Al,Nome-Al,
                  (Cod-Curso,Sem-ingresso)
                  (Cod-Disc,
                           (Sem-Disc-Cursada,Nota-Disc)))
```

Como esta tabela ÑN contém três tabelas aninhadas, ela gerará quatro tabelas na 1FN (tantas tabelas quantos abre parênteses aparecem na forma ÑN). Na 1FN, as tabelas, ainda sem a chave primária, são as seguintes:

1FN

```
Alunos        (Cod-Al,Nome-Al)
AlunoCurso    (Cod-Al,Cod-Curso,Sem-ingresso)
AlunoDisc     (Cod-Al,Cod-Disc)
AlunoDiscSem(Cod-Al,Cod-Disc,Sem-Disc-Cursada,Nota-Disc)
```

A primeira tabela corresponde à tabela ÑN sem suas tabelas aninhadas. Já a tabela `AlunoCurso` na 1FN corresponde à primeira tabela aninhada na forma ÑN. Observe que esta tabela contém, além das colunas `CodCurso` e `Sem-ingresso`, a coluna `Cod-Al`, chave primária da tabela na qual a encontra-se tabela aninhada, na forma ÑN.

Pelo mesmo motivo, a tabela `AlunoDiscSem`, que corresponde à terceira tabela aninhada, contém as colunas `Cod-Al` e `Cod-Disc`. Estas são as chaves primárias das tabelas nas quais a tabela aninhada em questão encontra-se na forma ÑN.

As chaves primárias das tabelas na 1FN são as seguintes:

1FN

```
Alunos        (Cod-Al,Nome-Al)
AlunoCurso    (Cod-Al,Cod-Curso,Sem-ingresso)
AlunoDisc     (Cod-Al,Cod-Disc)
AlunoDiscSem(Cod-Al,Cod-Disc,Sem-Disc-Cursada,Nota-Disc)
```

Conforme mencionado, a chave primária de uma tabela na 1FN não é necessariamente a concatenação das chaves primárias das colunas chaves primárias na forma ÑN. A seguir, apresentamos um exemplo em que, ao contrário dos que foram mostrados até agora, a chave primária da tabela na 1FN não é a concatenação das chaves das tabelas na forma ÑN. Considere a tabela não-normalizada apresentada abaixo:

ÑN

```
Arq-Candidatos(Cod-Curso,Nome-Curso,Numero-Vagas-Curso,
                          {Cod-Cand,Nome-Cand,Escore-Cand})
```

Esta tabela representa um arquivo que armazena informações sobre um concurso vestibular. O arquivo contém um registro para cada curso, com código, nome e número de vagas do curso. Além disso, para cada curso, há uma lista dos candidatos aprovados. Supõe-se que cada candidato tenha sido aprovado em um curso somente.

A passagem à 1FN gera as tabelas abaixo:

1FN

```
Cursos        (Cod-Curso,Nome-Curso,Numero-Vagas-Curso)
Candidatos    (Cod-Curso,Cod-Cand,Nome-Cand,Escore-Cand)
```

Observe que, na segunda tabela, correspondente à tabela aninhada da forma ÑN, a chave primária é apenas a coluna `Cod-Cand`(código do candidato), já que um candidato pode aparecer somente uma vez no arquivo.

Para determinar a chave primária de uma tabela na 1FN, que corresponda a uma tabela aninhada na forma ÑN, deve-se proceder como segue:

1 Tomar como parte da chave primária da tabela na 1FN a chave primária da tabela ÑN.
2 Verificar se esta chave primária é suficiente para identificar as linhas da tabela na 1FN.
   - Caso seja suficiente, a chave primária da tabela na 1FN é a mesma que a da tabela aninhada na forma ÑN.
   - Caso contrário, deve-se determinar quais as demais colunas necessárias para identificar as linhas da tabela na 1FN, compondo assim a chave primária na 1FN.

6.5.2 dependência funcional

Para entender as duas formas normais que serão apresentadas a seguir é necessário compreender o conceito de *dependência funcional*.

> dependência funcional
> =
> em uma tabela relacional, diz-se que uma coluna C_2 *depende funcionalmente* de uma coluna C_1 (ou que a coluna C_1 *determina* a coluna C_2) quando, em todas as linhas da tabela, para cada valor de C_1 que aparece na tabela, aparecer o mesmo valor de C_2

O conceito fica mais compreensível se considerarmos um exemplo. A tabela da Figura 6.7 contém, entre outras colunas irrelevantes ao exemplo, as colunas `Código` e `Salário`. Diz-se que a coluna `Salário` *depende funcionalmente* da coluna `Código` (ou que a coluna `Código` *determina* a coluna `Salário`) pelo fato de cada valor de `Código` estar associado sempre ao mesmo valor de `Salário`. Exemplificando, o valor "E1" da coluna `Código` identifica sempre o mesmo valor de `Salário` ("10").

Para denotar esta dependência funcional, usa-se uma expressão na forma `Código` → `Salário`. A expressão denota que a coluna `Salário` depende funcionalmente da coluna `Código`. Diz-se, também, que a coluna `Código` é o *determinante* da dependência funcional.

De forma geral, o determinante de uma dependência funcional pode ser um *conjunto de colunas* e não somente uma coluna como na definição acima.

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

Figura 6.7 Parte de tabela com dependências funcionais.

Exemplificando, na tabela da Figura 6.8, a coluna `C` depende das colunas `A` e `B`, ou seja, há uma dependência funcional `(A,B)` → `C`.

Deve-se observar que uma dependência funcional é uma restrição de integridade, ou seja, ela deve ser obedecida por todos os estados do banco de dados. Assim, para determinar uma dependência funcional, não basta observar o estado atual do banco de dados. É necessário conhecer suas regras de construção.

6.5.3 passagem à segunda forma normal (2FN)

A passagem à segunda forma normal (2FN) elimina um certo tipo de redundância de dados. Para exemplificar, tomamos a tabela `ProjEmp` da Figura 6.3. Nesta tabela, os dados referentes a empregados (`Nome`, `Cat` e `Sal`) estão

A	B	C	D	E
B	5	2	20	x
C	4	2	15	c
B	6	7	20	a
B	5	2	20	y
C	2	2	15	b
C	4	2	15	k
A	10	5	18	k
A	12	3	18	f
A	10	5	18	f
B	5	2	20	c
C	4	2	15	x
A	10	5	18	r
C	4	2	15	s

Figura 6.8 Tabela com exemplos de dependências funcionais.

redundantes para os empregados que trabalham em mais de um projeto. Um exemplo é o do empregado de código "8191". A passagem à segunda forma normal elimina este tipo de redundância de dados.

Uma tabela encontra-se na segunda forma normal (2FN) quando, além de encontrar-se na primeira forma normal, cada coluna não chave depende da chave primária *completa*. Uma tabela que não se encontra na segunda forma normal contém *dependências funcionais parciais*, ou seja, contém colunas não chave que dependem apenas de uma parte da chave primária.

segunda forma normal (2FN)
=
uma tabela encontra-se na segunda forma normal quando, além de estar na 1FN, não contém dependências parciais

dependência parcial
=
uma dependência (funcional) parcial ocorre quando uma coluna depende apenas de parte de uma chave primária composta

Obviamente, uma tabela que está na 1FN e que possui apenas uma coluna como chave primária não contém dependências parciais, já que, nesta tabela, é impossível uma coluna depender de uma parte da chave primária, visto que a chave primária não é composta por partes (por diversas colunas). Assim, toda tabela que está na 1FN e que possui apenas uma coluna como chave primária já está na 2FN. O mesmo aplica-se para uma tabela que contenha apenas colunas chave primária.

No exemplo, a tabela `Proj` encontra-se na 2FN por possuir uma chave primária simples (composta por apenas uma coluna).

Já a tabela `ProjEmp` deve ser examinada para procurar dependências parciais, pois possui uma chave primária composta de mais de uma coluna. As colunas `Nome`, `Cat` e `Sal` dependem, cada uma, apenas da coluna `CodEmp`, já que nome, categoria funcional e salário são determinados tão -somente pelo código do empregado. Por sua vez, as colunas `DataIni` e `TempAl` dependem da chave primária completa (para determinar a data em que um empregado começou a trabalhar em um projeto, bem como para determinar o tempo

pelo qual ele foi alocado ao projeto é necessário conhecer tanto o código do projeto quanto o número do empregado).

Para passar à 2FN, isto é, para eliminar as dependências de parte da chave primária é necessário dividir a tabela `ProjEmp` em duas tabelas com o seguinte esquema:

1FN

```
ProjEmp (CodProj,CodEmp,DataIni,TempAl)
Emp (CodEmp,Nome,Cat,Sal)
```

O processo de passagem à 2FN é ilustrado na Figura 6.9. Assim o modelo relacional correspondente ao arquivo em questão é o seguinte, na 2FN.

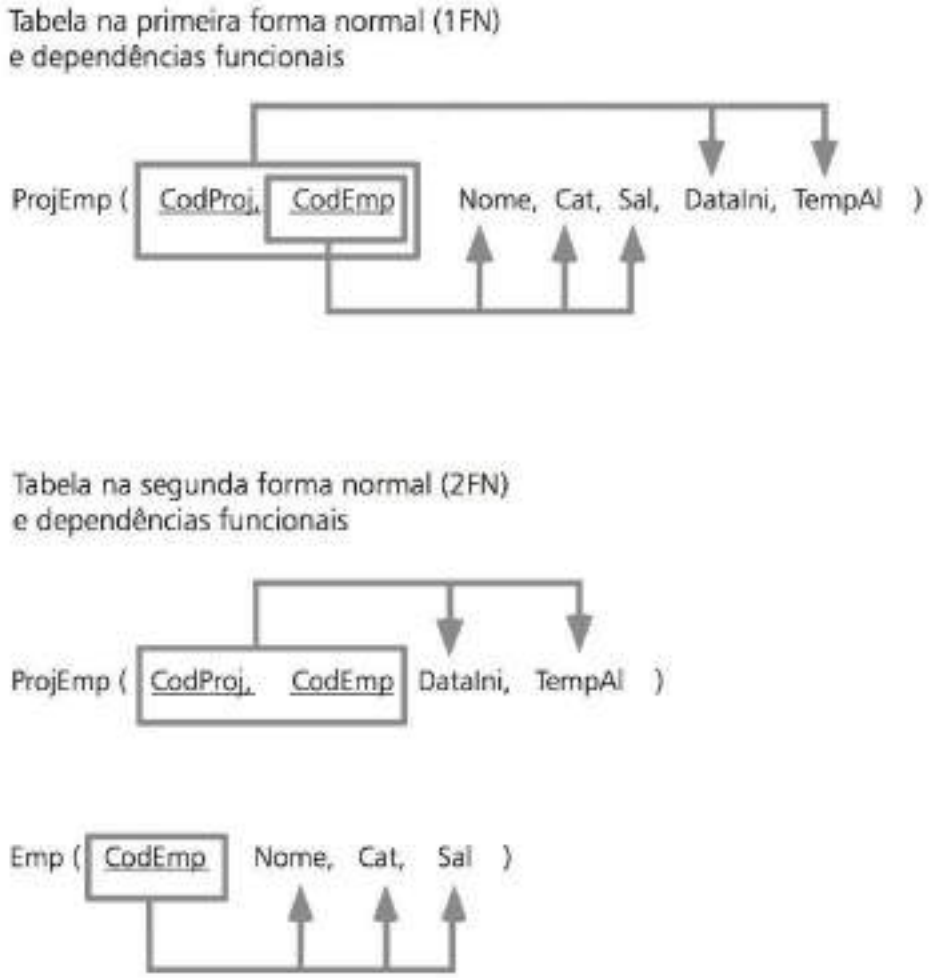


Figura 6.9 Passagem à 2FN.

```
2FN
Proj (CodProj,Tipo,Descr)
ProjEmp (CodProj,CodEmp,DataIni,TempAl)
Emp (CodEmp,Nome,Cat,Sal)
```

O conteúdo das tabelas na 2FN, caso considerarmos o conteúdo do documento original que está sendo normalizado, é apresentado na Figura 6.10.

De forma mais precisa, o processo de passagem da 1FN à 2FN é o descrito a seguir.

1 Copiar para a 2FN cada tabela que tenha chave primária simples ou que não tenha colunas além da chave. No exemplo, é o que acontece com a tabela `Proj`.

Proj:

CódProj	Tipo	Descr
LSC001	Novo Desenv.	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

ProjEmp:

CódProj	CodEmp	DataIni	TempAl
LSC001	2146	1/11/91	24
LSC001	3145	2/10/91	24
LSC001	6126	3/10/92	18
LSC001	1214	4/10/92	18
LSC001	8191	1/11/92	12
PAG02	8191	1/05/93	12
PAG02	4112	4/01/91	24
PAG02	6126	1/11/92	12

Emp:

CodEmp	Nome	Cat	Sal
2146	João	A1	4
3145	Silvio	A2	4
6126	José	B1	9
1214	Carlos	A2	4
8191	Mário	A1	4
4112	João	A2	4

Figura 6.10 Tabelas referentes ao exemplo na 2FN.

2 Para cada tabela com chave primária composta e com pelo menos uma coluna não chave (no exemplo, a tabela `ProjEmp`):

a Criar, na 2FN, uma tabela com as chaves primárias da tabela na 1FN (no exemplo, trata-se da tabela `ProjEmp` na 2FN).

b Para cada coluna não chave, fazer a seguinte pergunta:
"a coluna depende de toda a chave ou de apenas parte dela?"

- Caso a coluna dependa de toda a chave (as colunas `DataIni` e `TempAl` da tabela `ProjEmp`):
  - Criar a coluna correspondente na tabela com a chave completa na 2FN (a coluna `DataIni` e `TempAl` da tabela `ProjEmp` na 2FN).
- Caso a coluna dependa apenas de parte da chave (as colunas `Nome`, `Sal` e `Cat` da tabela `ProjEmp` na 2FN):
  - **i.** Criar, caso ainda não existir, uma tabela na 2FN que tenha como chave primária a parte da chave que é determinante da coluna em questão (a tabela `Emp` na 2FN).
  - **ii.** Criar a coluna dependente dentro da tabela na 2FN (as colunas `Nome`, `Sal` e `Cat` da tabela `Emp` na 2FN).

6.5.4 passagem à terceira forma normal (3FN)

Na passagem à terceira forma normal, elimina-se outro tipo de redundância. Para exemplificar, vamos considerar a tabela `Emp` da Figura 6.10. Vamos supor que o salário (coluna `Sal`) de um empregado seja determinado pela sua categoria funcional (coluna `Cat`). Neste caso, a informação de qual salário é pago a uma categoria funcional está representado redundantemente na tabela, tantas vezes quantos empregados a categoria funcional possui. A passagem à 3FN elimina este tipo de redundância de dados.

Uma tabela encontra-se na 3FN quando, além de estar na 2FN, toda coluna não chave depende *diretamente* da chave primária, isto é, quando não há dependências funcionais *transitivas* ou *indiretas*. Uma dependência funcional transitiva ou indireta acontece quando uma coluna não chave primária depende funcionalmente de outra coluna ou combinação de colunas não chave primária.

terceira forma normal (3FN)
=
uma tabela encontra-se na terceira forma normal quando, além de estar na 2FN, não contém dependências transitivas

dependência transitiva
=
uma dependência funcional transitiva ocorre quando uma coluna, além de depender da chave primária da tabela, depende de outra coluna ou conjunto de colunas da tabela

A passagem à 3FN consiste em dividir tabelas, de forma a eliminar as dependência transitivas. O processo de passagem à 3FN para o exemplo é mostrado na Figura 6.11.

Com isso, na 3FN, o modelo referente ao documento sendo normalizado é o seguinte:

```
3FN
Proj (CodProj,Tipo,Descr)
ProjEmp (CodProj,CodEmp,DataIni,TempAl)
Emp (CodEmp,Nome,Cat)
Cat (Cat,Sal)
```

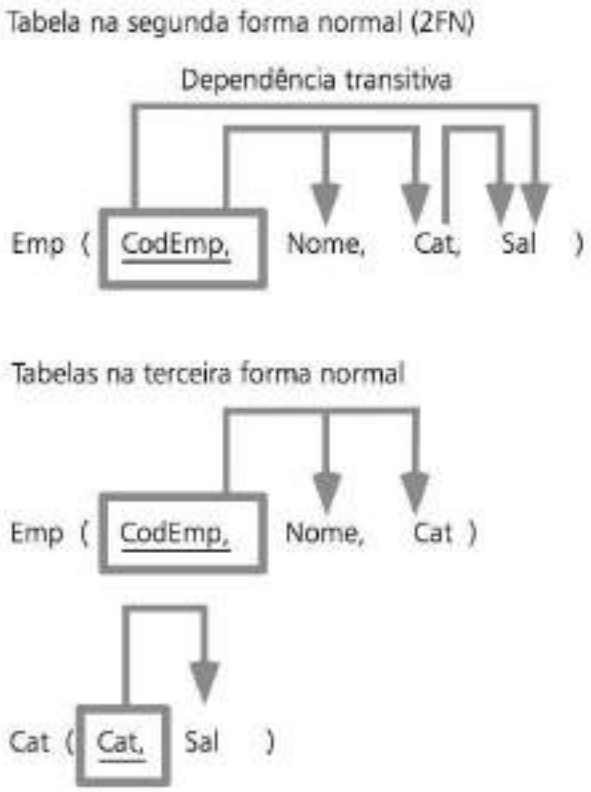


Figura 6.11 Passagem à 3FN para o exemplo.

As tabelas referentes à terceira forma normal do exemplo têm seu conteúdo mostrado na Figura 6.12.

De forma mais precisa, o processo de passagem da 2FN à 3FN é o seguinte:

1. Copiar para o esquema na 3FN cada tabela que tenha menos que duas colunas não-chave, pois neste caso não há como haver dependências transitivas.
2. Para tabelas com duas ou mais colunas não-chave:
   - **a** Criar uma tabela no esquema da 3FN com a chave primária da tabela em questão.
   - **b** Para cada coluna não-chave fazer a seguinte pergunta:
     "a coluna depende de alguma outra coluna não-chave (dependência transitiva ou indireta)?"
     - Caso a coluna depender apenas da chave:
       - **i.** Copiar a coluna para a tabela na 3FN

Proj:

CódProj	Tipo	Descr
LSC001	Novo Desenv.	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

ProjEmp:

CódProj	CodEmp	DataIni	TempAl
LSC001	2146	1/11/91	24
LSC001	3145	2/10/91	24
LSC001	6126	3/10/92	18
LSC001	1214	4/10/92	18
LSC001	8191	1/11/92	12
PAG02	8191	1/05/93	12
PAG02	4112	4/01/91	24
PAG02	6126	1/11/92	12

Emp:

CodEmp	Nome	Cat
2146	João	A1
3145	Sílvio	A2
6126	José	B1
1214	Carlos	A2
8191	Mário	A1
4112	João	A2

Cat:

Cat	Sal
A1	4
A2	4
B1	9

Figura 6.12 Tabelas referentes ao exemplo na 3FN.

- Caso a coluna depender de outra coluna:
  - **i.** Criar, caso ainda não exista, uma tabela no esquema na 3FN que tenha como chave primária a coluna da qual há a dependência indireta.
  - **ii.** Copiar a coluna dependente para a tabela criada.
  - **iii.** A coluna determinante deve permanecer também na tabela original.

6.5.5 passagem à quarta forma normal (4FN)

Para a maioria dos documentos e arquivos, a decomposição até a 3FN é suficiente para obter o esquema de um banco de dados correspondente ao documento. Na literatura, aparecem outras formas normais, como a forma normal de Boyce/Codd, a 4FN e a 5FN. Destas, a única que tem importância na prática da engenharia reversa é a quarta forma normal (4FN).

Para explicar a quarta forma normal, vamos utilizar um exemplo. Suponha que alguém tenha projetado um banco de dados relacional para o DER da Figura 6.13.

Neste DER, o relacionamento UTILIZAÇÃO indica que deseja-se manter a informação de qual empregado usa qual equipamento em qual projeto.

Caso seja projetado um banco de dados relacional para este exemplo usando as regras de projeto mostradas no capítulo anterior, o esquema do banco de

Figura 6.13 Exemplo de DER para utilização de equipamentos.

dados conterá quatro tabelas. Três servirão para implementar as entidades. Uma quarta tabela:

```
Util (CodProj,CodEmp,CodEquip)
```

servirá para implementar o relacionamento.

Agora suponha que os requerimentos da aplicação tenham mudado. O DER que representa os novos requerimentos é o apresentado na Figura 6.14. Como se observa, a granulosidade das informações que o usuário deseja manter não é mais tão grande como no caso da Figura 6.13. Agora deseja-se saber qual equipamento é usado em qual projeto (relacionamento *Proj-Eq*) e deseja-se saber que empregado está alocado a que projeto (relacionamento *Proj_Emp*). A informação de qual equipamento é usado por qual empregado passa a ser uma informação derivada: um empregado pode usar todos os equipamentos alocados aos projetos dos quais ele participa.

Continuando o exemplo, vamos supor que o banco de dados originalmente projetado para o relacionamento ternário não tenha sido modificado, apesar da existência de novos requerimentos. Um possível conteúdo para a tabela UTILIZAÇÃO é mostrado na Figura 6.15.

A tabela da Figura 6.15 obviamente contém redundância de dados. Exemplificando, a informação de que os empregados {"1","2","3"} trabalham no projeto "1" está representada duas vezes na tabela. Isso ocorre, por-

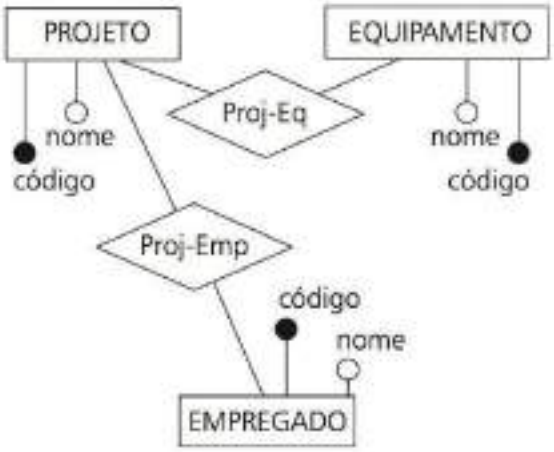


Figura 6.14 Exemplo da utilização de equipamentos – nova versão.

CodProj	CodEmp	CodEquip
1	1	1
1	2	1
1	3	1
1	1	2
1	2	2
1	3	2
2	2	2
2	2	4
3	3	1
3	4	1
3	3	3
3	4	3
3	3	5
3	4	5
4	2	5

Figura 6.15 Tabela que implementa o relacionamento UTILIZAÇÃO.

que para cada equipamento usado no projeto é necessário informar todos os seus empregados. A conseqüência é que também a informação de quais equipamentos são usados em um projeto está armazenada redundantemente. Exemplificando, o fato de o projeto "1" usar os equipamentos {"1","2"} está representada três vezes, já que o projeto conta com três empregados.

Entretanto, se validarmos essa tabela contra as formas normais vistas até aqui, observamos que a tabela encontra-se na 3FN. Está na 1FN por não conter tabelas aninhadas, está na 2FN por não possuir colunas não-chave (as três colunas compõem a chave) e está na 3FN pela mesma razão.

Para evitar este tipo de redundância de dados, é necessário considerar uma outra forma normal, a *quarta forma normal* (4FN), que se baseia no conceito de *dependência funcional multivalorada*.

Uma coluna ou conjunto de colunas depende multivaloradamente de uma coluna (determinante) da mesma tabela quando um valor do atributo determinante identifica repetidas vezes um *conjunto* de valores na coluna dependente. No exemplo (ver Figura 6.16), a coluna `CodEmp` depende multivaloradamente da coluna `CodProj`, já que um valor de `CodProj` (por exemplo "1") determina múltiplas vezes um conjunto de valores de `CodEmp` (no caso o conjunto de empregados do projeto, {"1","2","3"}, que aparece duas vezes).

CodProj	CodEmp	CodEquip
1	1	1
1	2	1
1	3	1
1	1	2
1	2	2
1	3	2
2	2	

Figura 6.16 Dependência funcional multivalorada.

Para representar uma dependência funcional multivalorada usa-se uma expressão semelhante à usada para representar dependências funcionais simples, substituindo a seta simples por uma seta dupla. A tabela `UTILIZAÇÃO` contém as seguintes dependências funcionais multivaloradas:

```
CodProj ↠ CodEmp
CodProj ↠ CodEquip
```

Uma tabela está na 4FN caso, além de estar na 3FN, não possua mais que uma dependência funcional multivalorada.

> quarta forma normal (4FN)
> =
> uma tabela encontra-se na quarta forma normal, quando, além de estar na 3FN, não contém dependências multivaloradas

Portanto, a tabela `UTILIZAÇÃO` não está na 4FN e deve ser decomposta em duas tabelas:

```
ProjEmp (CodProj,CodEmp)
ProjEquip (CodProj,CodEquip)
```

Estas tabelas correspondem à implementação dos dois relacionamentos da Figura 6.14.

Se considerarmos o conteúdo da tabela apresentada na Figura 6.15, o conteúdo das tabelas acima seria aquele apresentado na Figura 6.17.

ProjEquip:

CodProj	CodEquip
1	1
1	2
2	2
2	4
3	1
3	3
3	5
4	5

ProjEmp:

CodProj	CodEmp
1	1
1	2
1	3
2	2
3	3
3	4
4	2

Figura 6.17 Tabelas do exemplo na 4FN.

6.5.6 problemas da normalização

Nesta seção, vamos discutir alguns problemas que podem aparecer na prática da normalização de arquivos.

■ chaves primárias omitidas ou incorretas

Em arquivos convencionais, o conceito de chave primária não é obrigatório, como ocorre na abordagem relacional. Assim, é possível encontrar arquivos que não possuem chave primária. Quando um arquivo convencional não possui chave primária ou quando a chave primária nele usada difere da usual na organização, deve-se proceder como se a chave primária aparecesse no arquivo, isto é, deve-se inseri-la na forma ÑN.

Exemplificando, em um arquivo com dados sobre empregados de uma organização, enviado para fins de fiscalização a um órgão governamental, pode estar omitido o identificador de empregado usado na organização, já que este é irrelevante para o órgão fiscalizador.

Uma variante da situação descrita é a de documentos nos quais uma chave primária é propositadamente omitida, por não ser relevante para os leitores do arquivo. Um exemplo é apresentado no documento do Exercício 6 deste capítulo.

Outra situação análoga é o uso de uma chave alternativa, em vez da chave primária usual do arquivo. Exemplificando, no caso mencionado acima, se o órgão governamental fosse a receita federal, o arquivo poderia ter como

chave primária o CPF do empregado, em vez da chave primária normalmente usada na organização.

Em ambos os casos, se a normalização fosse executada diretamente sobre o arquivo convencional, apareceriam tabelas espúrias, com chaves primárias diferentes das usadas para as entidades representadas. Assim, tão logo uma destas situações seja detectada, deve-se introduzir, já na forma ÑN, a chave primária usual da tabela, como se ela aparecesse no arquivo normalizado.

■ atributos relevantes implicitamente representados

Atributos podem aparecer em arquivos convencionais implicitamente, na forma de ordenação de registros ou de listas, como ponteiros físicos, etc. Quando esta situação ocorrer, deve-se proceder como se o atributo aparecesse explicitamente no documento.

Um exemplo é o da ordenação de registros ou elementos de listas, onde a própria ordem é uma informação relevante. Para exemplificar, vamos retomar o exemplo do concurso vestibular apresentado anteriormente, agora com uma modificação. Na nova versão, não existe a coluna `Escore-Cand`, que continha a nota do aluno. Consideramos agora que os alunos aparecem ordenados no registro do curso, de acordo com sua classificação no vestibular. Seguindo os passos usuais da normalização, a tabela na forma ÑN seria a seguinte:

ÑN

```
Arq-Candidatos (Cod-Curso,Nome-Curso,Numero-Vagas-Curso,
                          (Cod-Cand,Nome-Cand))
```

O processo de normalização desta tabela resultaria nas seguintes tabelas:

4FN

```
Cursos (Cod-Curso,Nome-Curso,Numero-Vagas-Curso)
Candidatos (Cod-Curso,Cod-Cand,Nome-Cand)
```

Como na abordagem relacional as linhas não estão ordenadas, a informação da classificação dos candidatos em um curso foi perdida durante o processo de normalização. Assim, o procedimento correto teria sido o de incluir explicitamente, na tabela, já na forma ÑN, a informação que aparece implicitamente no arquivo na forma da ordenação dos registros (coluna `Ordem-Cand`). A tabela na forma ÑN que resultaria neste caso é apresentada a seguir:

ÑN

```
Arq-Candidatos (Cod-Curso,Nome-Curso,Numero-Vagas-Curso,
                        (Cod-Cand,Nome-Cand,Ordem-Cand))
```

Outro exemplo de atributo implicitamente representado são apontadores ou referências físicas a registros. Estes apontadores devem ser substituídos, quando da passagem a forma ÑN, pela chave primária do arquivo que está sendo referenciado.

■ atributos irrelevantes, redundantes ou derivados

Arquivos convencionais podem conter atributos que não são relevantes do ponto de vista conceitual e que existem no arquivo por questões técnicas ou de desempenho da implementação em questão. Exemplos destes atributos são campos contendo o número de ocorrências de listas, o tamanho de outros campos, estampas de tempo, etc. Estes campos devem ser eliminados já na passagem à forma não-normalizada.

6.6 → integração de modelos

A normalização de cada um dos arquivos ou documentos de um sistema conduz à definição de um conjunto de tabelas. O passo seguinte da engenharia reversa é o de integrar os modelos obtidos para cada arquivo no modelo *global* do banco de dados (ver Figura 6.1). Esse processo é conhecido na literatura de banco de dados por *integração de visões* ou *integração de esquemas*.

Esse processo tem dois objetivos:

- Os atributos de uma mesma entidade (ou de um mesmo relacionamento) podem estar armazenados em diferentes arquivos. Após o processo de normalização, estes atributos aparecem como colunas em diferentes tabelas. O processo de integração de visões objetiva juntar estas tabelas em uma única tabela que representa a entidade ou relacionamento em questão.
- O processo de normalização garante que cada tabela, se considerada isoladamente, esteja livre de redundância de dados. Entretanto, certos casos de redundância de dados entre tabelas podem permanecer e devem ser eliminados durante o processo de integração de visões.

O processo de integração de modelos dá-se em três passos:

1. integração de tabelas com a mesma chave,
2. integração de tabelas com chave contida e
3. verificação de 3FN.

A seguir, cada um desses passos é descrito em detalhes.

6.6.1 integração de tabelas com mesma chave

O processo de integração de modelos inicia pela junção de tabelas que possuem a *mesma* chave primária. Aqui, quando falamos de "mesma" chave primária, estamos exigindo que os *domínios* e os *conteúdos* das colunas que compõem a chave primária sejam iguais. Quando isso ocorrer, as diferentes tabelas devem ser juntadas em uma única tabela no modelo global.

Para exemplificar, retomamos o sistema de gerência de projetos mostrado anteriormente. O documento cuja normalização foi apresentada resultou no modelo relacional abaixo:

```
Documento 1:
Proj (CodProj,Tipo,Descr)
ProjEmp (CodProj,CodEmp,DataIni,TempAl)
Emp (CodEmp,Nome,Cat)
Cat (Cat,Sal)
```

Adicionalmente, vamos considerar que outro documento tenha sido normalizado, resultando no modelo relacional abaixo:

```
Documento 2:
Proj (CodProj,DataInicio,Descr,CodDepto)
Depto (CodDepto,NomeDepto)
ProjEquipamento (CodProj,CodEquipam,DataIni)
ProjEmp (CodProj,CodEmp,FunçãoEmpProj)
Equipamento (CodEquipam,Descrição)
```

Caso a coluna `CodProj` tenha o mesmo valor nas tabelas `Proj` dos dois modelos, estas duas tabelas devem ser juntadas no modelo global. Observe que a coluna `Tipo` é proveniente do documento 1, as colunas `DataInicio` e `CodDepto` são provenientes do documento 2 e a coluna `Descr` é proveniente de ambos os documentos.

Da mesma forma, caso as colunas `CodProje` `CodEmp` tenham os mesmos valores nas tabelas `ProjEmp` de ambos os modelos, estas duas tabelas devem ser fundidas em uma única tabela.

O modelo global resultante da integração dos dois modelos acima é o seguinte.

Modelo integrado:

```
Proj (CodProj,Tipo,Descr,DataInicio,CodDepto)
ProjEmp (CodProj,CodEmp,DataIni,TempAl,FunçãoEmpProj)
Emp (CodEmp,Nome,Cat)
Cat (Cat,Sal)
Depto (CodDepto,NomeDepto)
ProjEquipamento (CodProj,CodEquipam,DataIni)
Equipamento (CodEquipam,Descrição)
```

Como se vê no exemplo, todo o processo de integração de modelos baseia-se na comparação dos *nomes* de colunas e de tabelas dentro dos diferentes modelos. Nesta comparação, podem aparecer dois tipos de *conflitos de nomes*. *Homônimos* ocorrem quando dois objetos diferentes aparecem sob o mesmo nome. Por exemplo, diferentes campos, como data de nascimento e data de admissão de empregado aparecem em dois modelos diferentes sob o mesmo nome `DataEmp`. *Sinônimos* ocorrem quando um mesmo objeto aparece sob diferentes nomes. Por exemplo, a coluna descrição de projeto aparece em um modelo sob o nome `DescriçãoProj` e em outro modelo sob o nome `TítuloProj`.

Conflitos de nomes são resolvidos através de renomeação. Infelizmente, ainda não há técnicas na literatura que resolvam de forma sistemática o problema do conflito de nomes. Assim, a resolução deste conflito continua dependendo muito da experiência e do conhecimento do modelador.

6.6.2 integração de tabelas com chaves contidas

Outra situação na qual ocorre a integração de tabelas é quando uma tabela contém somente a chave primária e esta chave primária é subconjunto da chave primária de outra tabela. Note que, novamente, quando falamos de uma chave primária estar contida dentro da outra, estamos exigindo que a chave primária tenha o mesmo domínio e os mesmos valores.

Como exemplo, vamos considerar as duas tabelas a seguir:

```
AlunoDisc(Cod-Al,Cod-Disc)
AlunoDiscSem(Cod-Al, Cod-Disc,Sem-Disc-Cursada,Nota-Disc)
```

Vamos considerar que a primeira tabela informa que um aluno cursou uma disciplina, enquanto que a segunda tabela informa a nota obtida pelo aluno em uma disciplina em um semestre. Neste caso, as colunas `Cod-Al` e `Cod-Disc` da tabela `AlunoDisc` contêm exatamente os mesmos valores que as colunas `CodAle` `CodDisc` da tabela `AlunoDiscSem`. As informações contidas na tabela `AlunoDisc` já estão na tabela `AlunoDiscSem`. Portanto, a tabela `AlunoDisc` é redundante e pode ser eliminada sem perda de informações.

Observe que, para integrar uma tabela em outra, estamos exigindo que a tabela contenha somente a chave primária. Para exemplificar um caso em que esta exigência não é cumprida, vamos considerar as duas tabelas abaixo. A primeira tabela informa se um aluno teve ou não bolsa de estudos em um semestre, enquanto que a segunda informa a nota obtida pelo aluno em uma disciplina cursada em um semestre. Consideramos que as colunas `Cod-Al` e `Sem-Disc` da tabela `AlunoSem` contenham exatamente os mesmos valores que as colunas `CodAl` e `Sem-Disc` da tabela `AlunoDiscSem`.

```
AlunoSem(Cod-Al,Sem-Disc,BolsaSimNao)
AlunoDiscSem(Cod-Al, Cod-Disc,Sem-Disc,Nota-Disc)
```

Neste exemplo, a tabela `AlunoSem` não deve ser integrada à tabela `AlunoDiscSem`, pois, apesar de a chave primária da primeira tabela estar contida na chave primária da segunda tabela, a primeira possui atributos não-chave. Se o atributo *BolsaSimNão* fosse incluído na segunda tabela, a informação da obtenção ou não de bolsa por um estudante apareceria múltiplas vezes no banco de dados, tantos quantas disciplinas o aluno cursou no semestre.

Outro caso no qual a integração das tabelas não deve ser feita é quando os valores das colunas chave primária não são iguais, apesar de possuírem os mesmos nomes e domínios. Como exemplo, vamos considerar as tabelas abaixo:

```
AlunoMatric(Cod-Al,Sem-Disc)
AlunoDiscSem(Cod-Al,Cod-Disc,Sem-Disc,Nota-Disc)
```

Neste exemplo, consideramos que a primeira tabela (`AlunoMatric`) representa o fato de o aluno estar matriculado em um semestre. A segunda tabela (`AlunoDiscSem`) representa a nota que o aluno obteve em uma disciplina em

um semestre. Podem haver estados do banco de dados em que um aluno aparece associado a um semestre na tabela `AlunoMatric`, sem que exista informação sobre este aluno/semestre na tabela `AlunoDiscSem`. Este seria o estado do banco de dados durante o semestre letivo, quando um aluno já está matriculado, mas ainda não obteve notas. Neste caso, a primeira tabela não pode ser eliminada, pois contém informações que não aparecem na segunda tabela.

6.6.3 volta à 2FN

A integração de dois modelos, que caso considerados isoladamente estão na 4FN, pode conduzir a um modelo que está na 2FN mas não na 3FN.

Consideremos os modelos abaixo, ambos obtidos a partir de diferentes arquivos:

```
Arquivo 1:
Departamento (CodDepto,NomeDepto,CodGerenteDepto)
Arquivo 2:
Departamento (CodDepto,LocalDepto,NomeGerenteDepto)
```

A integração destes dois modelos resultaria no modelo integrado mostrado a seguir.

```
Modelo integrado:
Departamento (CodDepto,NomeDepto,CodGerenteDepto,
              LocalDepto,NomeGerenteDepto)
```

No modelo integrado, a tabela `Departamento` encontra-se na 2FN, mas não na 3FN, se considerarmos que a coluna não-chave `NomeGerenteDepto` depende funcionalmente da coluna não-chave `CodGerenteDepto`.

O problema ocorrido é que o Arquivo 2 referencia o gerente de um departamento, não através da chave primária de gerentes (que é seu código), mas através do nome do gerente. Esse é mais um exemplo das conseqüências da omissão de chaves primárias em arquivos convencionais, que havia sido citado acima.

Assim, deve-se, após a integração de modelos, verificar as tabelas a fim de garantir que todas estejam efetivamente na 3FN.

6.7 → finalização de modelo

6.7.1 construção do modelo ER

Após a integração dos modelos obtidos a partir dos diversos arquivos e documentos normalizados, segue a construção do modelo ER (ver Figura 6.1). Nesta construção, usam-se as regras apresentadas no capítulo anterior para a transformação de modelos relacionais em modelos ER.

6.7.2 verificação do modelo ER – limitações da normalização

A normalização não conduz necessariamente a um modelo ER perfeito, pois ela apenas elimina campos multivalorados e redundância de dados detectadas pelas formas normais descritas.

Conforme mencionado na Seção Passagem à primeira forma normal (1FN) (página 190), na passagem à 1FN optamos pela alternativa de decompor tabelas. Esta alternativa, apesar de ser mais simples na prática, pode levar a imperfeições no modelo (ver exemplo no Exercício 17).

Além das formas normais aqui descritas, outras foram propostas na literatura, como a forma normal de Boyce/Codd e a quinta forma normal (ver referências bibliográficas no final do capítulo). Como estas formas normais tratam de casos que aparecem pouco freqüentemente na prática, preferimos não apresentá-las neste texto.

Assim, o último passo da engenharia reversa é a verificação do modelo ER obtido, procurando corrigir imperfeições ainda existentes. Para ver exemplos de alguns problemas que podem permanecer no modelo normalizado, o leitor pode estudar os exemplos de engenharia reversa que aparecem nos exercícios a seguir, particularmente os referentes ao sistema de preparação de congressos e ao sistema de controle do almoxarifado.

6.8 → exercícios

A lista de exercícios a seguir contém, além de outras questões, dois estudos de caso de engenharia reversa do banco de dados de um sistema a partir de descrições de documentos e de arquivos convencionais.

Um sistema é o de preparação de congressos da IFIP que foi apresentado no Exercício 11 do capítulo 3. A ele, referem-se os exercícios 4 até 12 deste capítulo. O outro sistema é o de controle de um almoxarifado que foi apresentado no Exercício 12 do capítulo 3. A ele, referem-se os exercícios 13 até 22 deste capítulo. Para resolver estes exercícios, é recomendável que o leitor tenha lido a descrição do respectivo sistema no capítulo 3.

exercício 1 Considerando apenas o conteúdo atual da tabela apresentada na Figura 6.8, identifique quais as dpendências funcionais nela contidas.

exercício 2 Considere a tabela abaixo:

```
ItemVenda (NúmeroNF,CodigoTipoProd,NumeroProd,
           DescricaoProd,DataVenda,CodReg,CodEmp,QtdeItem,
           PreçoItem,NomeEmp,DescricaoTipoProd)
```

O significado das colunas acima é aquele apresentado no Exercício 4 do capítulo. A tabela apresentada está na primeira forma normal. Apresente a segunda e terceira formas normais.

exercício 3 No contexto de um sistema de controle acadêmico, considere a seguinte tabela:

```
Matricula(CodAluno,CodTurma,CodDisciplina,NomeDisciplina,
          NomeAluno,CodLocalNascAluno,NomeLocalNascAluno)
```

As colunas possuem o seguinte significado:

- `CodAluno`-código do aluno matriculado
- `CodTurma`-código da turma na qual o aluno está matriculado (código é o identificador de turma)
- `CodDisciplina`-código que identifica a disciplina da turma
- `NomeDisciplina`-nome de uma disciplina da turma
- `NomeAluno`-nome do aluno matriculado
- `CodLocalNascAluno`-código da localidade em que nasceu o aluno
- `NomeLocalNascAluno`-nome da localidade em que nasceu o aluno

Verifique se a tabela obedece a segunda e a terceira forma normais. Caso não obedeça, faça as transformações necessárias.

exercício 4 (Refere-se ao sistema de preparação de congressos descrito no Exercício 11 do capítulo 3.) A Figura 6.18 apresenta uma lista de todos os artigos submetidos a um congresso.

No cabeçalho, aparece o código e o nome do congresso. A seguir, são listados os códigos e nomes dos GTs que promovem o congresso. Posteriormente, em várias colunas são listados o código do artigo, seu título,

Relação de artigos submetidos ao congresso

Congresso: *DB25 – Advances in Database Systems*

GTs promotores: *GT3.1 – Database Systems*
GT3.3 – Database Conceptual Modeling

Código do artigo	Título do artigo	Assunto principal	Código do autor	Nome do autor
1	*Semantic Integration in Heterogeneous Databases*	*Heterogeneous Databases*	*2*	*Wen-Suan Li*
			4	*Chris Clifton*
2	*Providing Dynamic Security in a Federated Database*	*Heterogeneous Databases*	*21*	*N.B. Idris*
			7	*W.A. Gray*
			32	*R.F. Churchhouse*
3	*Efficient and effective clustering methods*	*Spatial databases*	*12*	*Raymond R. Ng*
			14	*Jiawei Han*
4	*Automated Performance Tuning*	*Performance and Optimization*	*36*	*Kurt. P. Brown*
5	*Bulk Loading into an OODB*	*Object oriented databases*	*1*	*Janet L. Wiener*

Congresso: *OO03 – Object Oriented Modeling*

GTs promotores: *GT4.6 – Software Engineering*

Código do artigo	Título do artigo	Assunto principal	Código do autor	Nome do autor
1	*Temporal aspects in OO models*	*Temporal modeling*	*2*	*Wen-Suan Li*

Figura 6.18 Lista de artigos submetidos aos congressos.

seu assunto principal, seu código de autor e os códigos e nomes dos vários autores do artigo. Observar que o mesmo código de artigo pode aparecer em diferentes congressos, já que a numeração de artigos inicia em um em cada congresso diferente.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 5 (Refere-se ao sistema de preparação de congressos descrito no Exercício 11 do capítulo 3) A Figura 6.19 apresenta a definição de um arquivo convencional que armazena dados dos artigos submetidos aos vários congressos. A definição do artigo está na linguagem COBOL, na qual muitos sistemas legados estão escritos. O leitor poderá fazer este exercício mesmo que não conheça COBOL, apenas com base nas explicações a seguir.

O arquivo contém um registro para cada artigo submetido ao congresso. A exemplo do exercício precedente, um artigo é identificado pelo código do congresso ao qual foi submetido e pelo seu código. A descrição em COBOL nos informa que cada registro de artigo contém: o código do congresso, o nome do congresso, o código do artigo, o título do artigo, uma lista com

```
FD    ARQ-ARTIGOS.
01    REG-ARTIGO.
      03    COD-CONGR
      03    NOME-CONGR
      03 NUMERO-ART
      03 TIT-ART
      03 NO-DE-AUTORES
      03 NO-DE-REVISORES
      03 NO-DE-TEMAS
      03 AUTOR OCCURS 1 TO 20 TIMES
            DEPENDING ON NO-DE-AUTORES.
         05    COD-AUTOR
         05    NOME-AUTOR
      03    TEMA OCCURS 1 TO 5 TIMES
            DEPENDING ON NO-DE-TEMAS.
         05 COD-TEMA
         05 NOME-TEMA
      03    REVISOR OCCURS 1 TO 5 TIMES
            DEPENDING ON NO-DE-REVISORES.
         05 COD-REVISOR
         05 NOME-REVISOR
         05 STATUS-REVISAO
```

Figura 6.19 Arquivo COBOL que armazena os artigos submetidos.

os códigos e nomes dos diversos autores, uma lista com código e nome dos temas de que trata o artigo e uma lista com código, nome e *status* da revisão dos vários especialistas que estão julgando ou julgaram o artigo. As listas são especificadas em COBOL na forma de cláusulas `OCCURS`. Os campos `NO-DE-AUTORES`, `NO-DE-TEMAS` e `NO-DE-REVISORES` apenas servem para controlar as respectivas listas de campos. Portanto, não devem ser considerados na normalização, já que são redundantes (ver Seção Problemas da normalização). O campo de *status* da revisão pode conter dois valores diferentes e informa se o artigo já recebeu parecer do revisor ou não.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 6 (Refere-se ao sistema de preparação de congressos descrito no Exercício 11 do capítulo 3.) A Figura 6.20 apresenta o programa de um congresso.

No cabeçalho constam o código e o nome do congresso. A seguir é listada a data a que se referem as informações seguintes (um congresso pode ocorrer em vários dias subseqüentes). Depois, são listadas as seções que são apresentadas no dia. Lembre que uma seção é um horário do congresso. Algumas seções, como a cerimônia de abertura (*Opening Ceremony*) ou o intervalo (*break*), não têm apresentação de artigos. Outras possuem apresentação de artigos, que são listados em ordem de apresentação após o título da seção. Esta seções possuem um moderador (*chair*) que aparece nomeado junto ao título da seção.

Este documento exemplifica vários casos especiais que devem ser tratados na normalização.

O documento omite vários códigos de entidades (ver Seção Problemas da normalização na página 207), que não são importantes para o leitor alvo do documento, mas são importantes na normalização. Estão omitidos o código do artigo e os códigos das pessoas que aparecem no modelo (autor e moderador).

Além disso, o documento apresenta de forma implícita (ver Seção Problemas da normalização na página 207) a informação da seqüência de apresentação dos artigos em uma seção.

Execute a normalização do documento, mostrando cada uma das formas normais.

DB25 – *Advances in Database Systems*
Conference Program

Monday September 12
09:00-10:30 Registration
10:00-10:45 Opening Ceremony
10:45-11:00 Break
11:00-12:30 Heterogeneous and Federated Databases (chair: Hervé Gallaire)
Semantic Integration in Heterogeneous Databases Using Neural Networks, Wen-Syan Li and Chris Clifton – USA
Providing Dynamic Security Control in a Federated Database, N.B. Idris, W.A. Gray and R.F. Churchhouse – UK
An approach for Building Secure Database Federations, Dirk Jonscher and Klaus R. Dittrich- Switzerland
12:30-14:00 Lunch
14:00-15:30 Issues on Architectures (Chair: Claudia Medeiros)
Optimization of Algorithms for Exploiting the Parallelism-Communication Tradeoff in Pipelined Parallelism, Waqar Hasan and Rajeev Motwani-USA
Dali: A High Performance Main Memory Storage Manager, H.V. Jagadish, Daniel Lieuwen, Rajeev Rastogi, Avi Silberschatz and S. Sudarshan- USA
Some Issues in Design of Distributed Deductive Databases, Mukesh K. Mohania and N.L. Sarda- India
15:30-16:30 Break
16:00-17:30 Performance and Optimization (Chair: Avi Silberschatz)
Towards Automated Performance Tuning for Complex Workloads, Kurt P. Brown, Manish Mehta, Michael Carey and Miron Livny- USA
...

Tuesday September 13
09:00-10:30 Object Oriented Databases (chair: Malkolm Atkinson)
Supporting Exceptions to Schema Consistency to Ease Schema Evolution in OODBMS, Eric Amiel, Maria-Jo Bellosta, Eric Dujardin and Eric Simon-France
...

Figura 6.20 Programa de um congresso.

exercício 7 (Refere-se ao sistema de preparação de congressos descrito no Exercício 11 do capítulo 3.) A Figura 6.21 apresenta a carta de aceitação de um artigo. Esta carta é enviada para o autor principal, portanto, há uma carta por artigo. Como cada artigo tem como identificador o código de seu congresso e seu código, estes dois campos são também os identificadores de uma carta de aceitação. Como no exercício anterior, há várias chaves primárias omitidas no documento. Deixamos ao leitor que as identifique e as inclua na normalização.

June 30, 1994

Carlos A. Heuser
Instituto de Informática – UFRGS
Av. Bento Gonçalves 9500,
Bloco IV Agronomia- Campus do Vale CEP 91501-970
Porto Alegre – RS. C.P. 15064 BRAZIL
Re: Paper *#97 – "Partitioning and aggregation in object oriented modeling"*

Dear Colleague:

I am pleased to inform you that the above referenced paper has been accepted for the DBOO94 - Database Modeling 94 . As you know, the conference will be held September 19-23, 1994 at ITESM, Estado de Mexico.

The instructions for the camera-ready version are attached to this letter. I'd like to draw your attention to the referees' report enclosed and to the 12-page limit required for publication in the Proceedings. **The final camera-ready copy of the manuscript should be received by us no later than July 25 .**

Your paper will only be included in the Conference Proceedings and in the final program if we receive along with the final manuscript the registration form and payment of the appropriate registration fee of the paper's presenter.

A preliminary program is enclosed. Any additional up-to-date information will be sent to you by e-mail or fax. If you have any questions about the conference or your participation, please contact the organizers at the address below. Alternatively, you can contact the IFIP representative in your country.

I am looking forward to seeing you at the conference.

Yours sincerely

Dirk Rastogi
Program Committee Chairman

Figura 6.21 Carta de aceitação de artigo.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 8 (Refere-se ao sistema de preparação de congressos descrito no Exercício 11 do capítulo 3.) A Figura 6.22 apresenta a definição de um arquivo convencional que armazena dados dos congressos. A definição está em COBOL. Valem aqui as mesmas observações feitas com referência ao arquivo de artigos apresentado anteriormente.

```
FD    ARQ-CONGRESSO.
01    REG-CONGRESSO.
      03    COD-CONG
      03    NOME-CONG
      03    DATA-INSC-CONG
      03 NO-DE-GTS
     03    NO-DE-INSCR
      03    GT OCCURS 1 TO 3 TIMES
DEPENDING ON NO-DE-GTS.
            05 COD-GT
            05 NOME-GT
     03    INSCRITO OCCURS 0 TO 200 TIMES
                   DEPENDING ON NO-DE-INSCR.
         05    COD-INSCR
         05    NOME-INSCR
         05    PAÍS-INSCR
```

Figura 6.22 Arquivo COBOL de congressos.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 9 Realize a integração dos modelos referentes aos cinco documentos e arquivos que você normalizou nos exercícios precedentes.

exercício 10 Identifique as chaves estrangeiras no modelo resultante da integração feita no exercício anterior.

exercício 11 Construa um diagrama ER a partir do modelo relacional obtido no exercício anterior. Utilize o processo de engenharia reversa descrito na Seção Engenharia reversa de modelos relacionais (página 169 em diante).

exercício 12 Identifique as diferenças do modelo ER construído no exercício anterior com o modelo ER construído no Exercício 11 do capítulo 3. Comente a causa das diferenças.

exercício 13 (Refere-se ao sistema de controle de almoxarifado descrito no Exercício 12 do capítulo 3.) A Figura 6.23 apresenta uma lista de estoque. Esta lista está organizada por corredor e apresenta, para cada receptáculo, qual a peça armazenada (código e descrição), juntamente com a quantidade de unidades da peça estocadas no receptáculo.

Execute a normalização do documento, mostrando cada uma das formas normais.

Lista de estoque em dd/mm/aa

Corredor: (---NoCorredor----)

Receptáculo	**Código da Peca**	**Descricao**	**Quantidade**
(---NoRecptac)	(---CodPeca---)	(---DescricaoPeca---)	(---Quant---)
(---NoRecptac)	(---CodPeca---)	(---DescricaoPeca---)	(---Quant---)

...

Lista de estoque em dd/mm/aa
Corredor: (---NoCorredor----)

Receptáculo	**Código da Peca**	**Descricao**	**Quantidade**
(---NoRecptac)	(---CodPeca---)	(---DescricaoPeca---)	(---Quant---)
(---NoRecptac)	(---CodPeca---)	(---DescricaoPeca---)	(---Quant---)

...

Figura 6.23 Lista de estoque por corredor do almoxarifado.

exercício 14 (Refere-se ao sistema de controle de almoxarifado descrito no Exercício 12 do capítulo 3) A Figura 6.24 apresenta a estrutura de um arquivo COBOL que armazena as listas de distribuição. Este arquivo registra, para cada entrega, identificada pelo código da ordem de compra (COD-OC) e pela data da entrega, os seguintes dados: o número do estrado que será usado na distribuição, o código da empilhadeira a ser usada, o operador da empilhadeira e uma lista de itens que identificam os corredores a serem visitados durante a distribuição. A lista informa, para cada corredor, quais os receptáculos

```
FD  LISTA-DISTRIB.
01  REG-LISTA.
      03  COD-OC
03  DATA-ENTREGA
03  NO-ESTRADO
03  NO-DE-CORREDORES
      03 COD-EMPLHADEIRA
      03 OPER-EMPILHADEIRA
      03 ITEM-CORR OCCURS 1 TO 20 TIMES
                     DEPENDING ON NO-DE-CORREDORES.
         05  NO-CORR
         05  NO-DE-RECEPTACULOS
         05  ITEM-RECEPT OCCURS 1 TO 5 TIMES
                        DEPENDING ON NO-DE-RECEPTACULOS.
               07  NO-RECEPT
               07  COD-PECA
               07  DESCR-PECA
               07  QTDE
```

Figura 6.24 Lista de distribuição (arquivo COBOL).

que serão visitados e, para cada receptáculo qual o código e a descrição da peça a ser armazenada, juntamente com a quantidade de unidades a serem colocadas no receptáculo.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 15 (Refere-se ao sistema de controle de almoxarifado descrito no Exercício 12 do capítulo 3.) A Figura 6.25 mostra um boleto que é emitido para cada pedido feito por um cliente. Este boleto informa ao cliente em que rampa seu pedido será entregue.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 16 (Refere-se ao sistema de controle de almoxarifado descrito no Exercício 12 do capítulo 3.) A Figura 6.26 apresenta a estrutura de um documento que lista a situação de atendimento de ordens de compra (OC). Uma OC pode estar parcialmente atendida, isto é, para cada OC podem ocorrer entregas parciais (apenas algumas peças, parte da quantidade).

O documento lista em seu cabeçalho o código da OC, sua data e seu fornecedor (código e nome). Para cada peça encomendada na OC, o documento lista o código, a descrição e a quantidade pedida. A seguir, para cada entrega é listada a data da entrega e a quantidade entregue. A soma da quantidade já entregue é listada a seguir. Observe que certas peças podem não ter tido entregas.

exercício 17 (Refere-se ao sistema de controle de almoxarifado descrito no Exercício 12 do capítulo 3.) A Figura 6.27 apresenta a estrutura de um pedido. No cabeçalho aparece o número de pedido, a data e o cliente que o realizou (código, nome e uma lista com número variado de telefones). Depois são listados, para cada peça pedida, seu código, sua descrição e a quantidade pedida.

Ao Cliente
(---codCliente---) (---nomeCliente---)

Informamos que seu pedido (---noPedido---) **será entregue na rampa** (---noRampa---)

Figura 6.25 Boleto indicativo de rampa de saída de peças.

Situação de atendimento de Ocs pendentes em dd/mm/aa.

OC número: (---noOC---) **Data da OC:** (---data---)
Código do Fornecedor: (---codFornec---)
Nome do Fornecedor: (---NomeFornec)

Código Peça	Descrição Peça	Quantidade Pedida	Data Entrega	Quantidade Entregue
(---)	(---)	(---)	(---data---)	(---)
			(---data---)	(---)
		Total entregue:	(---)	
(---)	(---)	(---)	(---data---)	(---)
		Total entregue:	(---)	
(---)	(---)	(---)	(---data---)	(---)
			(---data---)	(---)
			(---data---)	(---)
		Total entregue:	(---)	
(---)	(---)	(---)		
		Total entregue:	0	

OC número: (---noOC---) **Data da OC:** (---data---)
Código do Fornecedor: (---codFornec---)
Nome do Fornecedor: (---NomeFornec)

Código Peça	Descrição Peça	Quantidade Pedida	Data Entrega	Quantidade Entregue
(---)	(---)	(---)	(---data---)	(---)
			(---data---)	(---)
		Total entregue:	(---)	

Figura 6.26 Situação de atendimento de ordens de compra.

Pedido

Pedido: (---noPed---) **Data do pedido:** (---data---)
Código do Cliente: (---codCliente---)
Nome do Cliente: (---nomeCliente)
Telefones p/ contato: (---noTel---) (---noTel---) (---noTel---)

Código Peça	Descrição Peça	Quantidade Pedido
(---)	(---)	(---)
(---)	(---)	(---)

Figura 6.27 Pedido.

Lista de Busca

Pedido: (---noPed---) **Data do pedido:** (---data---)
Código do Cliente: (---codClie---)
Nome do Cliente: (---NomeCliente)

Corredor: (---noCorr---)

Número Receptáculo	Código Peça	Descrição Peça	Quantidade a Buscar
(---)	(---)	(---)	(---)
(---)	(---)	(---)	(---)
(---)	(---)	(---)	(---)

Corredor: (---noCorr---)

Número Receptáculo	Código Peça	Descrição Peça	Quantidade a Buscar
(---)	(---)	(---)	(---)
(---)	(---)	(---)	(---)

Figura 6.28 Lista de busca.

Execute a normalização do documento, mostrando cada uma das formas normais.

exercício 18 (Refere-se ao sistema de controle de almoxarifado descrito no Exercício 12 do capítulo 3.) A Figura 6.28 apresenta a estrutura da lista de busca. Para cada pedido é emitida uma lista de busca. O cabeçalho da lista de busca contém o número de pedido, sua data e o seu cliente (código e nome). Após, para cada corredor, são listados o receptáculo visitado, o código da peça a apanhar, sua descrição e a quantidade a buscar.

exercício 19 Realize a integração dos modelos referentes aos documentos e arquivos do sistema de almoxarifado que você normalizou nos exercícios precedentes.

exercício 20 Identifique as chaves estrangeiras no modelo resultante da integração feita no exercício anterior.

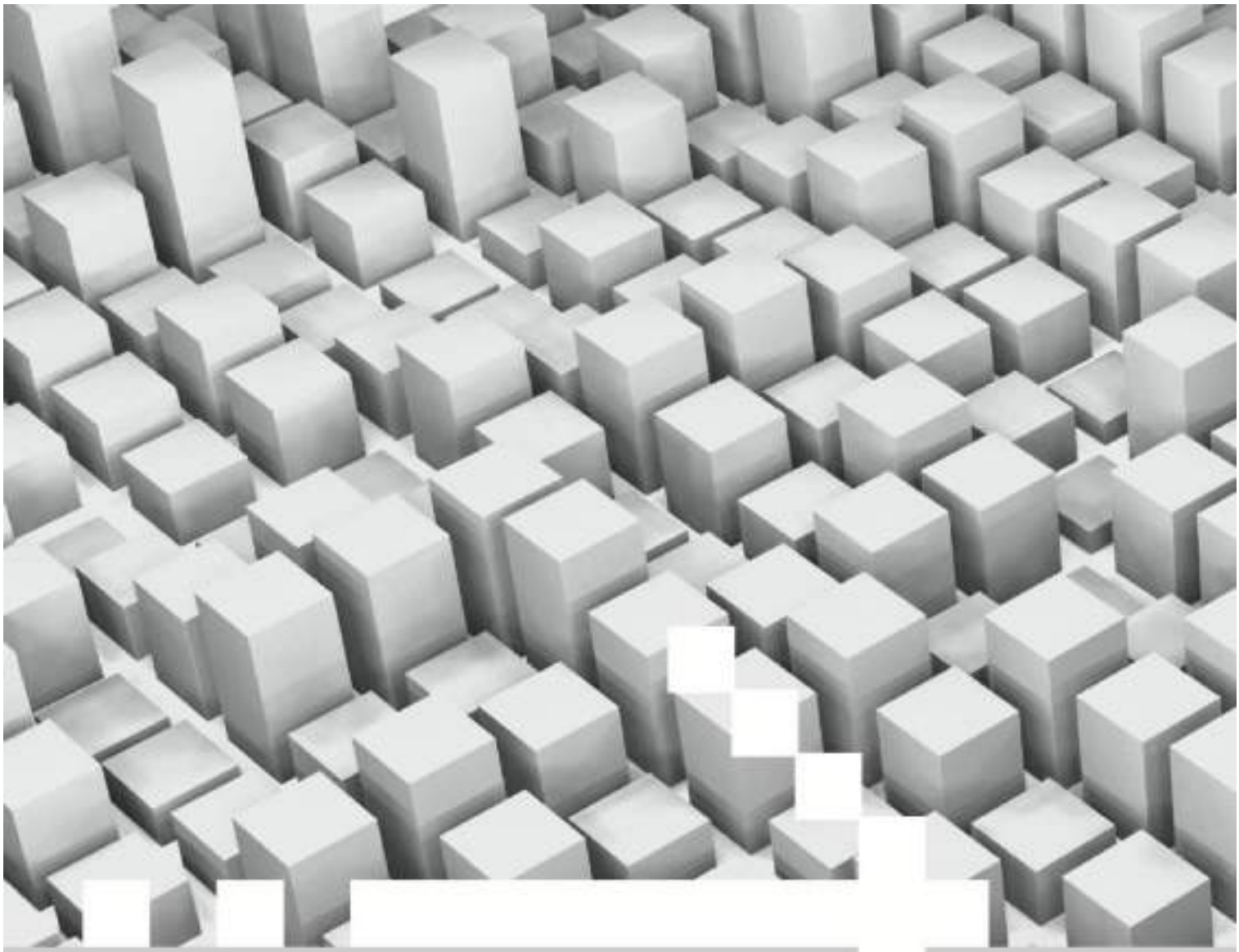
exercício 21 Construa um diagrama ER a partir do modelo relacional obtido no exercício anterior. Utilize o processo de engenharia reversa descrito na Seção Engenharia reversa de modelos relacionais (página 169 em diante).

exercício 22 Identifique as diferenças do modelo ER construído no exercício anterior com o modelo ER construído no Exercício 12 do capítulo 3. Comente a causa das diferenças.

6.9 → leituras recomendadas

A idéia de normalização que serve de base a este capítulo surgiu juntamente com a abordagem relacional e foi proposta por Ted Codd, o criador da abordagem relacional. As três primeiras formas normais aparecem nos artigos originais de Codd (CODD (1971) e CODD (1972). A normalização é apresentada na maioria dos textos introdutórios de banco de dados como os já citados ELMASRI; NAVATHE (2002) e SILBERSCHATZ; KORTH; SUDARSHAN (1999).

A engenharia reversa é apresentada no livro de BATINI; CERI; NAVATHE (1992). Uma boa cobertura do assunto aparece também no livro de SETZER (1987).



capítulo 7

soluções de exercícios selecionados

Este capítulo apresenta as soluções de exercícios selecionados. Procuramos privilegiar os problemas mais complicados.

Em muitos exercícios, particularmente nos estudos de caso, não há uma única solução correta. Como as descrições dos sistemas são informais, elas se prestam a diferentes interpretações. Por este motivo, em alguns estudos de caso mais complexos, não é apresentada somente uma solução, mas são discutidas alternativas, tanto corretas quanto incorretas.

exercício 2.4

Grande parte do exercício está resolvida na Seção modelos ER têm poder de expressão limitado (página 73). Lá é apresentado um diagrama de ocorrências no qual:

- uma pessoa pode estar casada com ela mesma e
- uma pessoa pode participar de dois casamentos, desde que no papel de marido em um casamento e no papel de esposa em outro.

A Figura 7.1 mostra como é possível excluir essas duas possibilidades do modelo ER. A mudança que foi feita em relação ao modelo original (Figura 2.7) foi a de transformar *CASAMENTO* em entidade e, com isso, abrir a possibilidade de especificar a cardinalidade do relacionamento de casamento com pessoa.

Outra alternativa que poderia ser considerada é a de especializar a entidade *PESSOA* em homens e mulheres. Essa solução somente deve ser considerada caso existam atributos diferentes para homens e mulheres.

exercício 2.5

A Figura 7.2 apresenta um possível diagrama de ocorrências para o relacionamento de *SUPERVISÃO*. Este diagrama modela as seguintes instâncias do relacionamento:

- *e1* é supervisor de *e2*
- *e1* é supervisor de *e3*
- *e3* é supervisor de *e4*
- *e3* é supervisor de *e5*

Na Figura 3.2 (página 75), é mostrado outro exemplo de diagrama de ocorrências, contendo uma situação que, apesar de permitida pelo diagrama, não corresponde à nossa intenção ao modelar o relacionamento de *SUPERVISÃO*.

Figura 7.1 Implementando restrições no casamento.

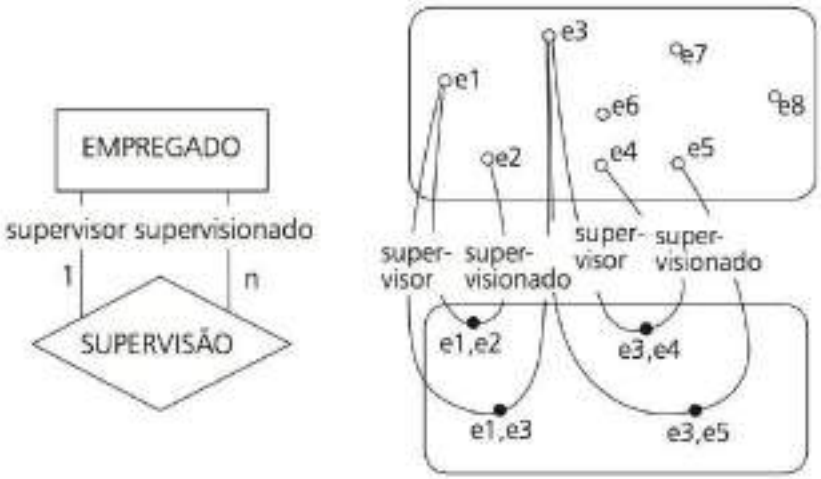


Figura 7.2 Diagrama de ocorrências para o relacionamento *SUPERVISÃO*.

■ exercício 2.7

A Figura 7.3 mostra o resultado da transformação do modelo ER da Figura 2.11 em um modelo que não contém relacionamentos ternários.

Observe que uma ocorrência da entidade *DISTRIBUIÇÃO* é identificada pelas ocorrências de *CIDADE* e de *PRODUTO* aos quais está relacionada. Esta é a tradução para a cardinalidade do relacionamento *DISTRIBUIÇÃO*, que aparece no modelo entidade-relacionamento original. Esta cardinalidade especifica que, para cada par formado por um produto e uma cidade, pode haver somente um distribuidor.

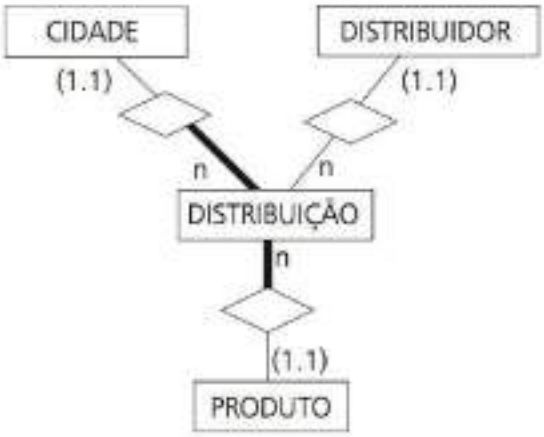


Figura 7.3 Transformação de relacionamento ternário em entidade.

Uma solução que, à primeira vista, pode parecer equivalente ao relacionamento ternário é a apresentada na Figura 7.4. Esta solução não é equivalente à original. O modelo representando na nova solução contém menos informações que o modelo original. O leitor poderá certificar-se deste fato se construir e comparar exemplos com ocorrências para o modelo original e para o modelo da Figura 7.4.

■ exercício 2.11

A transformação do relacionamento *ATUAÇÃO* da Figura 2.16 em entidade resulta no modelo ER da Figura 7.5.

Observe que uma ocorrência de *ATUAÇÃO* é identificada pelos relacionamentos com as entidades *PROJETO* e *ENGENHEIRO*.

■ exercício 2.12

A Figura 7.6 apresenta um modelo ER que resulta da modificação do modelo da Figura 2.20. A modificação consta em possibilitar que um dependente seja empregado. Caso se mantivesse o modelo original, o nome do dependente seria armazenado redundantemente. A solução adotada foi a de especializar

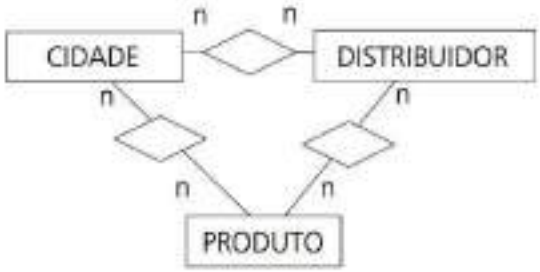


Figura 7.4 Tentativa de transformação de relacionamento ternário.

Figura 7.5 Resultado da transformação de um relacionamento em entidade.

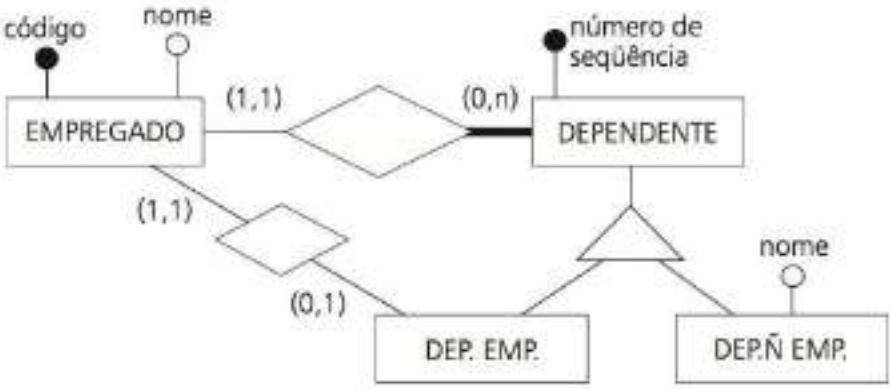


Figura 7.6 Dependente pode ser empregado.

a entidade *DEPENDENTE* em duas, DEP.Ñ EMP., que contém os atributos dos dependentes que não são empregados e DEP.EMP., que não contém atributos mas está relacionada à entidade empregado correspondente.

■ exercício 2.17

A solução encontra-se no modelo ER da Figura 7.7. Caso não sejam usados atributos multivalorados, é necessário criar uma entidade para cada atributo multivalorado. Observe o identificador desta entidade. Um telefone é identificado pelo seu número e pelo cliente correspondente. Isso permite que diferentes clientes tenham o mesmo número de telefone (o que era permitido pelo modelo original).

■ exercício 2.18

A solução para modelar uma especialização compartilhada é usar relacionamentos para ligar as entidades especializadas à entidade genérica (Figura 7.8).

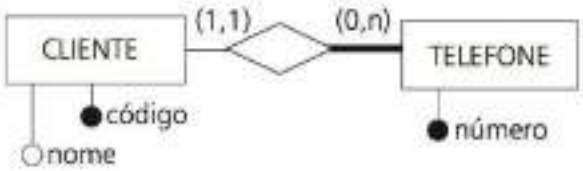


Figura 7.7 Transformação de atributo multivalorado em entidade.

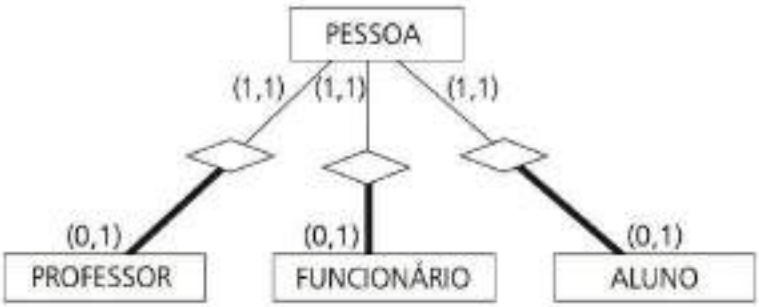


Figura 7.8 Tratando especialização compartilhada através de relacionamentos.

Cada entidade especializada é identificada pela entidade genérica a qual está relacionada. Lembre que na generalização/especialização as entidades especializadas herdam o identificador da entidade genérica.

Observe as cardinalidades dos relacionamentos. Cada ocorrência de uma entidade especializada (PROFESSOR, FUNCIONÁRIO e *ALUNO*) deve estar relacionada a exatamente uma ocorrência da entidade genérica (*PESSOA*). Já uma ocorrência da entidade genérica (*PESSOA*) pode estar relacionada a, no máximo, uma ocorrência de cada uma das entidades especializadas.

Observe que, como trata-se de um relacionamento compartilhado, uma ocorrência da entidade genérica pode, inclusive, estar relacionada a mais de uma ocorrência de entidade especializada, desde que cada ocorrência pertença a uma entidade especializada diferente. Exemplificando, uma ocorrência de *PESSOA* pode estar relacionada tanto a uma ocorrência de FUNCIONÁRIO quanto a uma ocorrência de *ALUNO*.

Recorde que, de forma geral, generalizações/especializações podem ser classificadas nos seguintes tipos:

- exclusiva ou compartilhada e
- total ou parcial.

O fato de ser uma generalização/especialização *exclusiva* limita uma ocorrência da entidade genérica a aparecer em somente uma das entidades especializadas. Observe que esta restrição não pode ser transposta ao modelo ER, quando a generalização/especialização é transformada em relacionamentos.

O fato de ser uma generalização/especialização *total* exige que uma ocorrência da entidade genérica apareça em pelo menos uma das entidades especializadas. Também esta restrição não pode ser transposta ao modelo ER, quando a generalização/especialização é transformada em relacionamentos.

■ exercício 2.24

Não é possível expressar esta restrição, pelo fato de o modelo ER não possuir uma notação que mostre que a união de dois relacionamentos (no caso, o de *VENDA* com *MEDICAMENTO* e o de *VENDA* com PERFUMARIA) tem cardinalidade mínima **1**. Esta restrição teria que ser especificada fora do modelo ER.

■ exercício 2.30

O modelo ER expressa que um processador de textos não pode existir no banco de dados sem que exista uma secretária que o domine (cardinalidade mínima da entidade *PROCESSADOR DE TEXTOS* no relacionamento DOMÍNIO). Assim, cada vez que uma secretária for excluída, é necessário verificar cada processador de textos por ela dominado. Caso ela seja a última a dominar determinado processador de textos, a secretária não poderá ser excluída, ou, alternativamente, a exclusão da secretária deverá ser propagada à exclusão do processador de textos em questão.

■ exercício 2.31

A especialização de *EMPREGADO* é *exclusiva*, isto é, uma ocorrência da entidade genérica não pode aparecer em mais de uma de suas especializações. Como as entidades SECRETÁRIA, *ENGENHEIRO* e GERENTE são especializações de *EMPREGADO* em uma mesma hierarquia de generalização/especialização, um empregado não pode aparecer em mais de uma delas.

Para permitir que uma secretária ou um engenheiro sejam gerentes, é necessário transformar a generalização/especialização em *compartilhada*.

Alternativamente, também é possível retirar a entidade GERENTE da hierarquia de generalização/especialização na qual aparece com as entidades SECRETÁRIA e *ENGENHEIRO*. Neste caso, GERENTE passa a ser um auto-relacionamento de *EMPREGADO*.

exercício 3.1

1 O relacionamento apresentado não inclui a restrição de que um produto não pode ser composto por ele mesmo.

2 A Figura 7.9 apresenta um modelo ER que exclui a possibilidade de um produto ser composto por ele mesmo. Cabe observar que esta solução somente deve ser adotada caso existam atributos específicos das várias especializações de *PRODUTO*. Caso não existam atributos específicos para estas especializações, a melhor alternativa seria manter o modelo original, expressando a restrição de integridade em questão fora do modelo.

3 Caso deseje-se armazenar no banco de dados uma hierarquia com número não limitado de níveis de composição, fica impossível registrar no modelo o fato de um produto não poder aparecer em sua composição. Essa restrição exige o uso de recursividade em sua expressão, o que não é possível em modelos ER.

exercício 3.3

No modelo, há atributos específicos de pessoa física, outros específicos de pessoa jurídica e alguns comuns a ambos os tipos de clientes. Este é o caso típico para a aplicação do conceito de generalização/especialização. A Figura 7.10 apresenta um modelo para a mesma realidade usando este conceito.

O novo modelo é mais preciso que o original. No original, os atributos das entidades especializadas apareciam como opcionais. O modelo não informava quais atributos pertencem a qual tipo de cliente e se o atributo é ou não obrigatório para um tipo de cliente.

Figura 7.9 Estrutura de produto com número fixo de níveis.

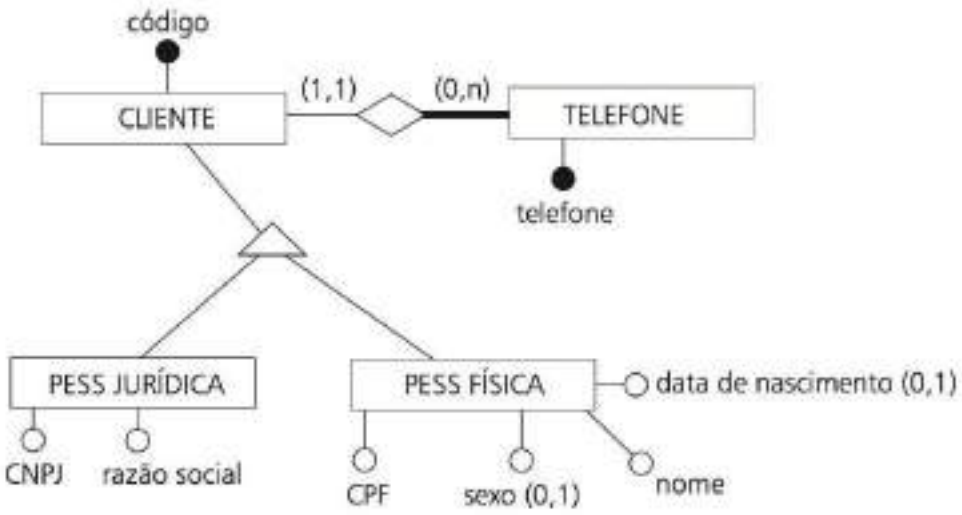


Figura 7.10 Substituindo atributos opcionais por especializações.

No novo modelo, fica claro que atributos pertencem a cada uma das especializações e se eles são ou não obrigatórios.

■ exercício 3.5

O modelo ER apresentado na Figura 2.11 através da notação deste livro é apresentado na Figura 7.11 através da notação européia.

A característica básica desta notação é a interpretação da cardinalidade de relacionamento. Nela, a cardinalidade informa quantas vezes uma ocorrência de uma entidade participa do relacionamento. Assim, a cardinalidade **n** junto

Figura 7.11 Modelo da Figura 2.11 na notação européia.

à entidade DISTRIBUIDOR informa que uma ocorrência desta entidade pode participar várias vezes do relacionamento (um distribuidor pode ter várias distribuições).

É importante observar que, neste diagrama, não está expressa a restrição de que um distribuidor tem exclusividade de distribuição para um determinado produto, em uma determinada cidade, conforme especificado no modelo ER original, na Figura 2.11.

Assim, observa-se que, para relacionamentos com grau maior que dois, não há equivalência entre as notações de cardinalidades, a notação européia e a notação usada neste livro.

■ exercício 3.6

A alteração do modelo mostrado na Figura 3.10 para permitir que cada servidor (não cada cargo de servidor) tenha associado seu CPF e seu número de inscrição no PASEP é mostrada na Figura 7.12.

Neste novo modelo, temos informações que são específicas de uma pessoa que é servidor. Estas informações não devem aparecer na entidade *SERVIDOR CARGO*. Como um servidor pode ter vários cargos, as informações ficariam redundantes para cada cargo do servidor. Por outro lado, colocar as informações na própria entidade *PESSOA* resulta em atributos opcionais. Assim, a solução mais adequada é criar uma nova especialização da entidade *PESSOA*, a entidade *SERVIDOR*, que contém as informações específicas daquela pessoa que é servidor. As informações relativas a cada cargo de um servidor permanecem na entidade *SERVIDOR CARGO*.

■ exercício 3.7 – administradora de imóveis

Para o estudo de caso da administradora de imóveis, vamos seguir os passos da estratégia *descendente* de modelagem descritos na Seção Partindo do conhecimento das pessoas (página 103).

Enumeração de entidades. Uma primeira leitura do enunciado pode resultar nas seguintes entidades: *ADMINISTRADORA*, *CONDOMÍNIO*, *UNIDADE*, *PESSOA*

Identificação de relacionamentos. Os relacionamentos que podem ser identificados são os seguintes:

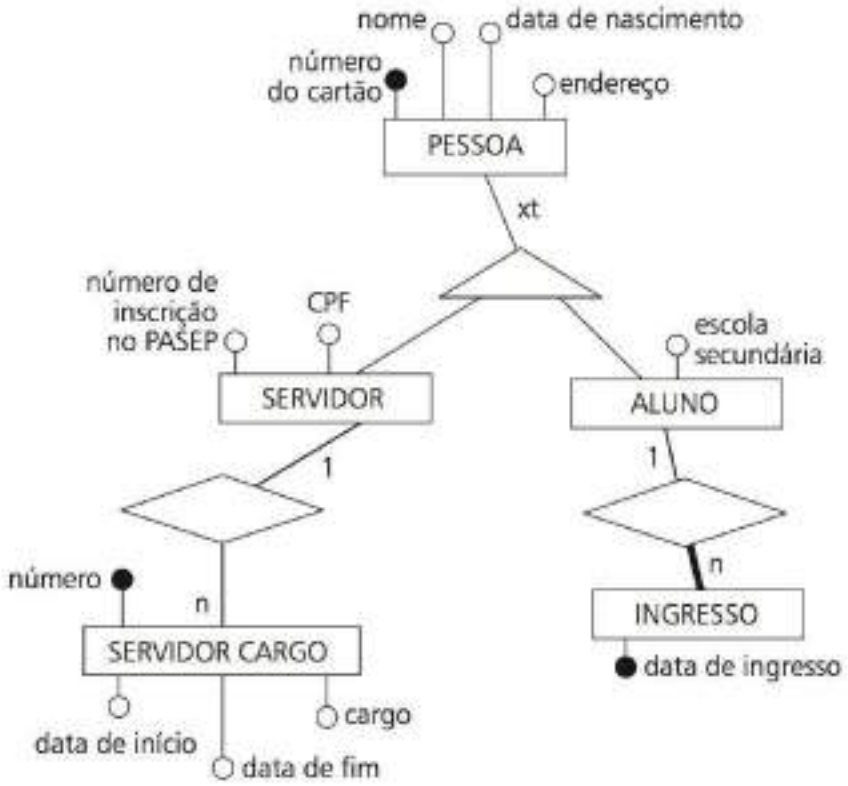


Figura 7.12 Alteração do modelo da Figura 3.10: servidor tem informações associadas.

- *COMPOSIÇÃO* entre *CONDOMÍNIO* e *UNIDADE*
- *PROPRIEDADE* entre *UNIDADE* e *PESSOA*
- *ALUGUEL* entre *UNIDADE* e *PESSOA*

Um relacionamento que poderia parecer necessário é o que liga *ADMINISTRADORA* com *CONDOMÍNIO*. Caso o BD que esteja sendo modelado seja destinado a apenas uma administradora (como no exemplo), este relacionamento não deve aparecer no modelo. Como a entidade *ADMINISTRADORA* possui uma única instância, não é necessário manter no BD a informação de qual administradora administra qual condomínio. Este relacionamento somente deveria aparecer caso várias administradoras pudessem coexistir no mesmo BD (ver discussão na Seção Entidade isolada e entidade sem atributos na página 94.)

Identificação de cardinalidades máximas As cardinalidades identificadas aparecem no DER da Figura 7.13.

Figura 7.13 Diagrama ER para a administradora de imóveis.

■ exercício 3.8 – locadora de mídias antigas

Seguindo os passos do processo de modelagem, chegamos aos seguintes resultados.

Enumeração de entidades As entidades identificadas são: *LOCADORA*, *FILME*, *FITA*, *CLIENTE*, *CATEGORIA* e *ATOR*.

Uma alternativa de modelagem seria considerar a categoria de um filme como atributo de *FILME*. Como consideramos que o conjunto de categorias de filmes não é imutável e pode variar ao longo da vida do BD, preferimos modelar a categoria como uma entidade (ver discussão na Seção Atributo *versus* entidade relacionda, na página 78).

Outra decisão é a de como modelar o empréstimo, se através de uma entidade ou um relacionamento. Optamos por modelá-lo como relacionamento.

Identificação de relacionamentos

- entre *FILME* e *FITA*
- *EMPRÉSTIMO* entre *FITA* e *CLIENTE*
- entre *FILME* e *CATEGORIA*
- *ESTRELA* entre *ATOR* e *FILME*

Como no estudo de caso precedente, a entidade *LOCADORA* está isolada por possuir uma única instância.

As cardinalidades máximas são simples de obter a partir do enunciado e aparecem no diagrama ER da Figura 7.14.

Determinação de atributos Um primeiro levantamento de atributos, com base na leitura do enunciado, resulta nos atributos que aparecem no DER da Figura 7.14. O único atributo que não aparece explicitamente no enunciado é o número do rolo (atributo de fita). Este atributo é necessário para filmes multirolo, para identificar que rolo do filme está armazenado na fita.

Determinação de identificadores Cada entidade do modelo deve ter seus identificadores (atributos e/ou relacionamentos) definidos. Alguns identificadores aparecem explicitamente no enunciado do problema:

- Cada fita é identificada por seu número.
- Cada filme possui um identificador.
- Cada cliente é identificado por seu número.

Para as demais entidades é necessário criar um identificador. Nomes ou outros atributos que ocupem muito espaço de armazenamento não são recomendados, caso se tenha em vista uma implementação em SGBD relacional, já que eles resultam em estruturas internas de acesso pouco eficientes. Por este motivo, é necessário criar atributos identificadores para as entidades *CATEGORIA* e *ATOR*.

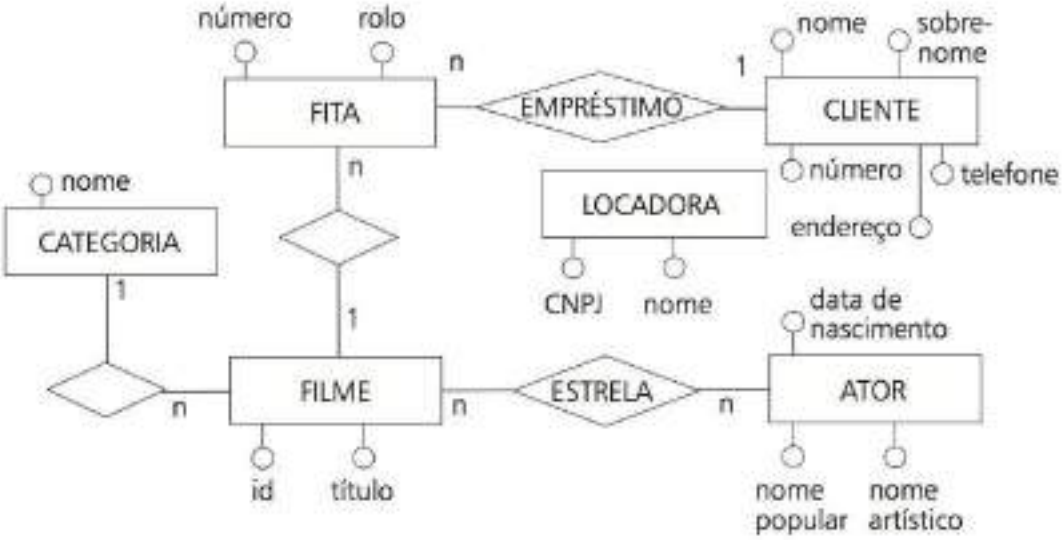


Figura 7.14 Diagrama ER para locadora de mídias antigas (primeira versão).

Verificação de aspectos temporais Não há atributos nem relacionamentos dos quais deva ser mantida história. Não há alterações a fazer no modelo, no que tange a aspectos temporais.

Domínios dos atributos Nesta etapa devem ser definidos os domínios dos atributos. Isso normalmente é feito já com o uso de uma ferramenta CASE. Assim, nos estudos de caso, deixaremos de lado esta etapa.

Definição de cardinalidades mínimas Na Figura 7.15 estão apresentadas as cardinalidades mínimas que podem ser deduzidas do enunciado do problema

■ exercício 3.9 – sistema de reserva de passagens aéreas

Enumeração de entidades A leitura do enunciado nos leva às seguintes entidades: *COMPANHIA*, *RESERVA*, *PASSAGEIRO*, *TRECHO*, *VÔO*, *CIDADE*, *AEROPORTO*, *TIPO AERONAVE*, *HORÁRIO* e *ASSENTO*.

Não foi criada uma entidade correspondente às pessoas que efetivaram a reserva. Não serão armazenadas informações sobre estas pessoas. Decidiu-se não cadastrar pessoas de forma central, pois esta operação demandaria tempo e seriam necessárias informações adicionais sobre a pessoa para resolver o problema de homônimos. Por isso, esta informação será modelada como um atributo da reserva.

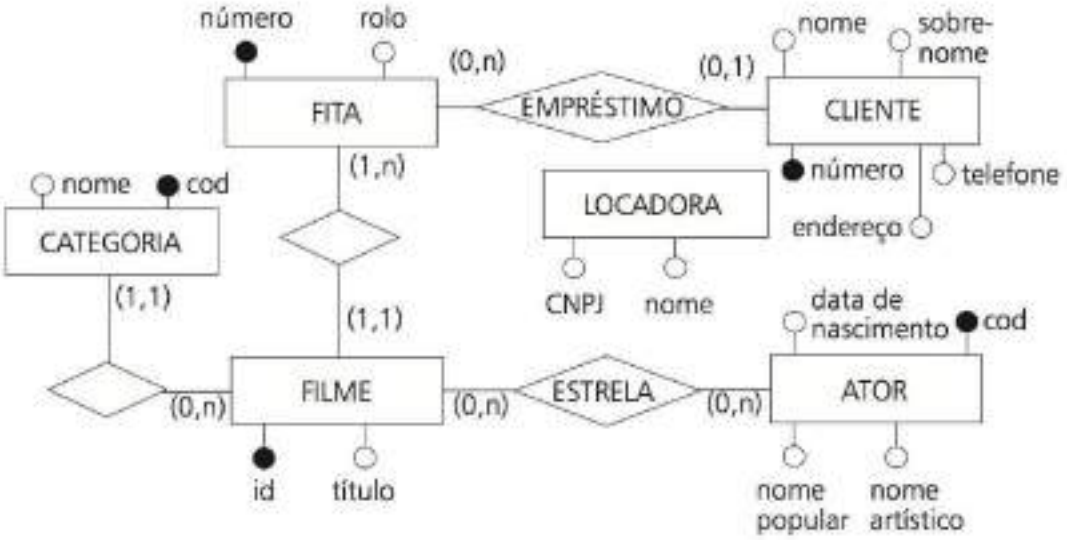


Figura 7.15 Diagrama ER para locadora de mídias antigas (final).

Identificação de relacionamentos Os relacionamentos identificados aparecem no diagrama ER da Figura 7.16. O relacionamento RSRV-TRCH associa a reserva aos vários trechos de vôo que a compõem. Ele representa cada trecho de vôo reservado.

Este relacionamento está sendo tratado como uma entidade associativa, pois, para marcar o assento, é necessário relacionar cada trecho da reserva com o assento reservado.

Determinação de atributos e identificadores Para não sobrecarregar o diagrama, listamos os atributos das entidades. Os atributos identificadores estão sublinhados e os relacionamentos identificadores aparecem no diagrama ER da Figura 7.16.

```
RESERVA (codigo reserva,passageiro,prazo)
VOO (número)
TRECHO ()
AEROPORTO (código,nome)
```

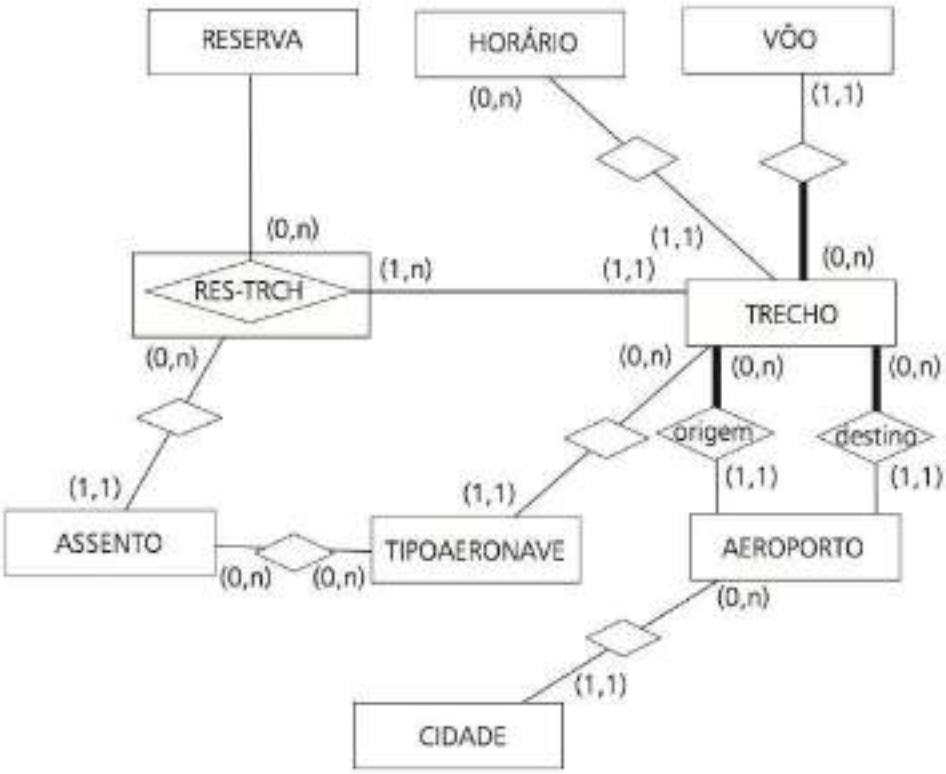


Figura 7.16 Diagrama ER para sistema de reserva de passagens aéreas.

```
CIDADE (código,nome,país)
TIPO AERONAVE (código,descrição)
HORARIO (dia semana,horário partida,horário chegada)
ASSENTO (número,classe)
RSRV-TRCH (data)
```

Restrições que não podem ser representadas no modelo Neste sistema aparecem diversas restrições que não podem ser representadas em modelos ER:

- Uma reserva de trecho somente pode ser realizada caso existam vagas no trecho em questão, na data em questão.
- Uma reserva para um assento somente pode ser feita se o assento em questão existir no tipo de aeronave utilizada no trecho de vôo em questão.

Uma observação geral sobre a solução adotada é que ela é conceitual e procura apenas fixar as necessidades de informação do sistema. O modelo não inclui redundâncias de dados que objetivem melhorar a performance de determinadas operações executadas freqüentemente. Estas considerações devem ficar para uma fase posterior do projeto do BD. Neste caso, atributos redundantes, como o número de vagas em um trecho de vôo, em uma data, inclusive discriminado por classe, poderia ser necessário. Isso levaria à necessidade de criação de uma entidade *TRECHO*-DIA correspondendo a cada viagem específica de um trecho de vôo em uma data. Para uma discussão sobre a inclusão de redundância no banco de dados projetado, ver a Seção Refinamento do modelo relacional na página 162.

■ exercício 3.10 – sistema para locadora de veículos

Enumeração de entidades Uma primeira leitura do enunciado nos leva às seguintes entidades: *LOCADORA*, *TIPO AUTOM OU CAMIONETA PASS*, *TIPO CAMIONETA CARGA*, *VEÍCULO*, *PESSOA FÍSICA*, *PESSOA JURÍDICA* e *FILIAL*. Além destas, são necessárias as entidades *RESERVA* e *LOCAÇÃO* para manter informações sobre as duas transações centrais da locadora.

Há vários atributos e relacionamentos comuns às entidades *TIPO AUTOM OU CAMIONETA PASS* e *TIPO CAMIONETA CARGA*. Por este motivo, é usada uma generalização das três entidades (*TIPO VEÍCULO*).

Um raciocínio análogo pode ser aplicado às entidades *PESSOA FÍSICA* e *PESSOA JURÍDICA*, levando à generalização *CLIENTE*.

A entidade *REVISÃO* é usada para manter as informações sobre as revisões que devem ser feitas em veículos de um tipo. Essa informação é multivalorada (há várias revisões para um tipo de veículo) e, por isso, não pode ser armazenada em um atributo de *TIPO VEÍCULO*. A entidade *MOTORISTA* destina-se a armazenar informações sobre a habilitação do motorista que está dirigindo o veículo. Estas informações não foram colocadas em *CLIENTE*, já que um cliente pessoa jurídica pode ter diferentes motoristas cadastrados.

Identificação de relacionamentos Os relacionamentos identificados aparecem no diagrama ER da Figura 7.17.

Entre as entidades *RESERVA* e *FILIAL*, são usados dois relacionamentos, um para representar a filial onde o veículo será retirado (origem) e outro para representar a filial onde o veículo será devolvido (destino). A *LOCAÇÃO* está opcionalmente ligada a uma filial de destino. Este relacionamento serve para informar em que filial o veículo será devolvido, caso seja devolvido em uma filial diferente daquela em que foi retirado.

O relacionamento entre *MOTORISTA* e *CLIENTE* serve para informar quais são os motoristas cadastrados por um cliente.

O relacionamento entre *MOTORISTA* e *LOCAÇÃO* serve para informar o motorista responsável pelo automóvel. Observe que não há um relacionamento

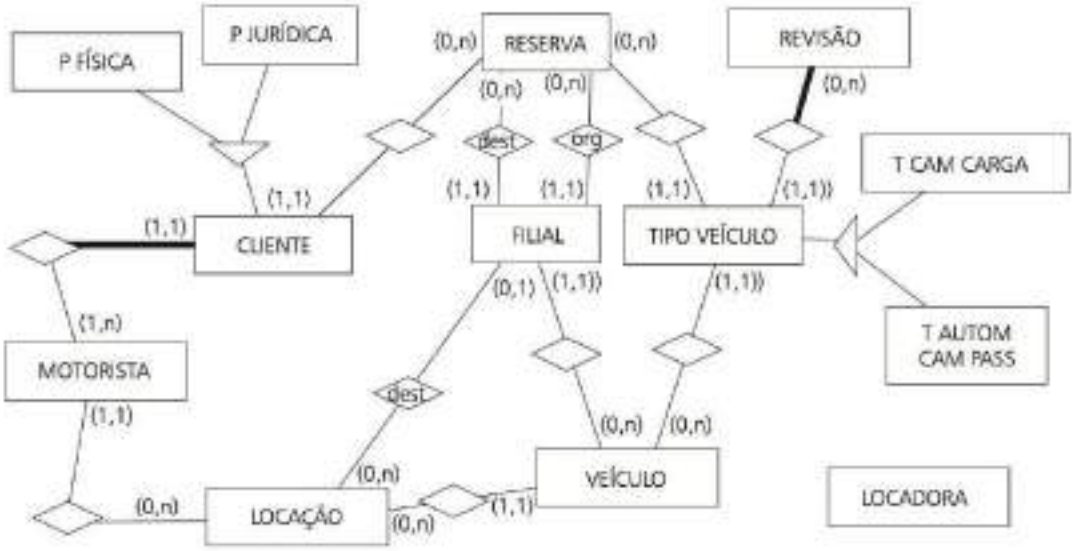


Figura 7.17 Diagrama ER para sistema de locadora de veículos.

direto entre *LOCAÇÃO* e *CLIENTE*, já que este seria redundante (para cada locação, tem-se seu motorista e, para cada motorista, tem-se o cliente correspondente).

Determinação de atributos e identificadores Os atributos das entidades são os seguintes:

```
CLIENTE (código,nome,endereço)
P FÍSICA (sexo,data nascimento,CPF)
P JURÍDICA (CNPJ,inscrição estadual)
FILIAL (código,localização)
VEÍCULO (placas,número chassi,número motor,cor,quilometragem,
         data medida quilometragem,quilometragem,última revisão)
TIPO VEÍCULO (código,tipo,horas limpeza, quilometragem,média diária)
T AUTOM CAM PASS (tamanho,número passageiros, ar-condicionado,
                  rádio,MP3,CD, direção hidráulica,
                  câmbio automático)
T CAM CARGA (capcacidade carga)
REVISÃO (quilometragem)
MOTORISTA (número habilitação,data vencimento, identidade,nome)
RESERVA (número,data retirada,data devolução)
LOCAÇÃO (número,data retirada,data devolução)
LOCADORA (CNPJ,nome,endereço,telefone)
```

Os atributos identificadores estão sublinhados. Os relacionamentos identificadores estão representados no diagrama ER da Figura 7.17.

Restrições que não podem ser representadas no modelo As restrições que não estão apresentadas no modelo acima são:

- A habilitação de um motorista não pode vencer durante o período previsto para a locação.
- Um veículo cuja quilometragem exceda a quilometragem de sua próxima revisão não pode ser locado.
- Para um cliente pessoa física, somente deve haver um motorista cadastrado. Neste caso, não deve ser informado o nome do motorista, já que ele é a própria pessoa física.
- Somente pode ser feita uma reserva quando houver previsão de disponibilidade de veículos do tipo, na filial de origem, na data da reserva.
- Uma locação para a qual não tenha sido feita reserva somente pode ocorrer na mesma condição acima.

Assim como na solução do sistema de reservas de passagens aéreas (Exercício 3.9), a presente solução é puramente conceitual, não incluindo dados redundantes que porventura melhorem o desempenho de determinadas operações sobre o banco de dados.

exercício 3.11 – sistema de preparação de congressos da IFIP

Uma primeira leitura do exemplo pode nos levar a uma série de entidades modelando papéis de pessoas: autor, moderador, inscrito, membro de GT, etc. Além disso, um dos objetivos do sistema, que aparece implícito no enunciado, é o de criar um cadastro central de pessoas. Isso fica claro quando o enunciado exige que a pessoa não deva aparecer várias vezes em certos relatórios ou que é necessário identificar os vários papéis que uma pessoa cumpriu no passado. Assim, fica evidente que é necessária uma entidade *PESSOA*. Uma solução que poderia ser tentada é a de usar o conceito de generalização/especialização, na forma apresentada na Figura 7.18.

Esta solução não é adequada para o problema. Não há propriedades específicas de uma pessoa em cada um dos papéis (autor, inscrito, membro de GT, etc.). Mesmo que houvessem propriedades (por exemplo, a data de inscrição de uma pessoa em um congresso), estas propriedades seriam do relacionamento entre a pessoa e o congresso.

Por esse motivo, optamos por modelar os papéis que as pessoas cumprem através de relacionamentos, conforme apresentado no diagrama da Figura 7.19.

Os atributos identificados são os seguintes:

Figura 7.18 Usando generalização/especialização para modelar papéis de pessoas (incorreto).

```
CONGRESSO(código,nome,data realização,local,
          prazo submissão artigos,prazo inscrição)
GT(código,nome)
PESSOA(código,nome,instituição,endereço,telefone,fax,e-mail)
ARTIGO(número,título,aceito/rejeitado)
SESSÃO(código,data,hora,título)
MEMBRO-GT(cargo)
AVALIAÇÃO(parecer,estado avaliação)
```

O atributo estado *avaliação* serve para informar se uma artigo já foi distribuído a um avaliador ou se já retornou do avaliador e recebeu parecer.

Observe que não há um relacionamento *CONVIDADO*. Do ponto de vista conceitual, este relacionamento é desnecessário. As pessoas que são convidadas a um congresso já se encontram no modelo (são os autores de trabalhos submetidos ao congresso, os membros do CO e do CP do congresso, e assim por diante). Assim, este relacionamento é redundante diante das informações já existentes e não deve aparecer no modelo conceitual. Observe que não estamos afirmando que, mais tarde, durante o projeto lógico, este relacionamento não deva ser criado, para atender requisitos de desempenho da aplicação.

Pelas regras de equivalência de modelos vistas na Seção Diferentes modelos podem ser equivalentes, é possível transformar em entidades os diversos relacionamentos **n:n** que aparecem no modelo da Figura 7.19. Nessa variante de solução, os papéis de pessoa, como *MODERADOR*, *AVALIADOR*, *AUTOR* e outros, são transformados em entidades. A Figura 7.20 apresenta, de forma parcial, um esboço de como seria o modelo nesta alternativa.

Observe que as entidades originárias dos relacionamentos **n:n** não são especializações de *PESSOA*, como na Figura 7.18, mas sim entidades relacionadas à entidade *PESSOA*. Além disso, os identificadores das entidades que representam os papéis de pessoa são diferentes do identificador de *PESSOA*. Por exemplo, uma ocorrência de *MEMBRO GT* é identificada pela pessoa que é membro, bem como pelo GT da qual ela participa, enquanto uma ocorrência de *PESSOA* é identificada pelo seu código.

- Finalmente, cabe salientar que há uma série de restrições de integridade que não estão expressas no modelo ER. A seguir, listamos estas restrições:

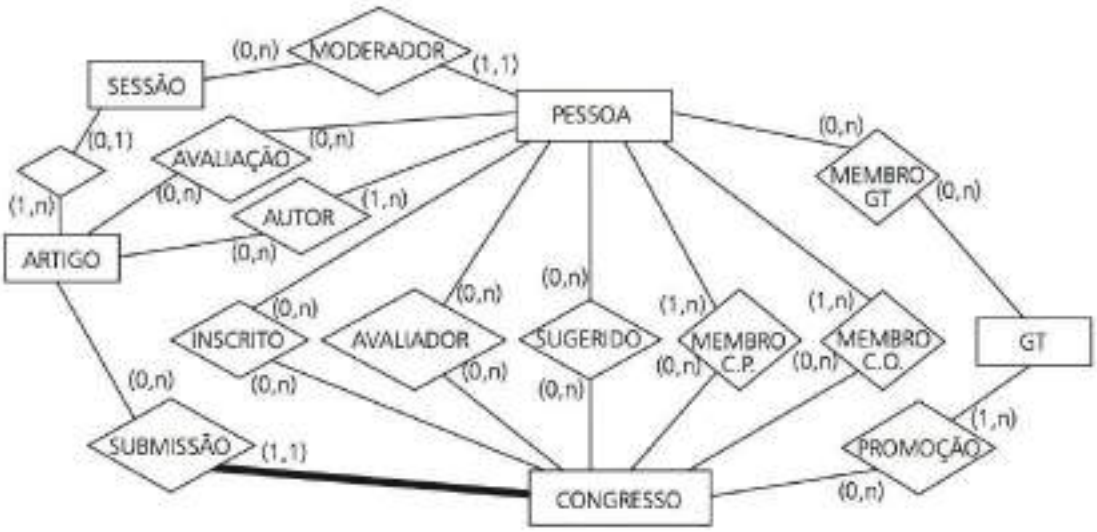


Figura 7.19 Diagrama ER para o sistema de preparação de congressos.

- Um avaliador de um artigo deve ser avaliador cadastrado no congresso.
- Um avaliador de um artigo não deve ser autor deste artigo.
- Um artigo que tenha sido julgado deve ter pelo menos três avaliações.
- Um inscrito deve ter sido convidado para o congresso (ver enunciado para verificar quem é convidado).
- Somente pode aparecer em uma sessão um artigo aceito.

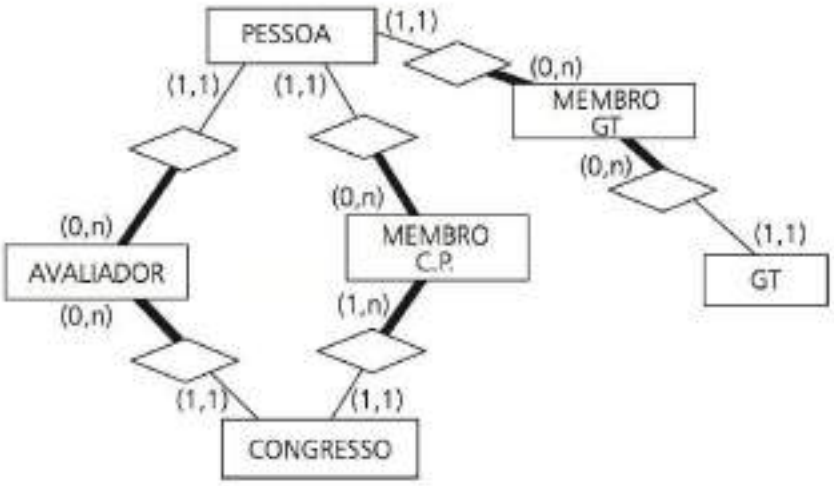


Figura 7.20 Diagrama ER para o sistema de preparação de congressos (parcial – usando entidades em vez de relacionamentos **n:n**).

exercício 3.12 – sistema de almoxarifado

A Figura 7.21 apresenta o diagrama ER para o sistema de almoxarifado. A solução apresentada modela os itens de uma ordem de compra, de um pedido, de uma ordem de compra, de uma entrega, etc. como entidade e não como relacionamento **n:n** (entre ordem de compra, pedido, ...e peça). As duas soluções são equivalentes. Aqui, preferimos usar a primeira, apenas para demonstrar a sua utilização, diferentemente dos modelos das questões anteriores, onde usamos relacionamentos **n:n**.

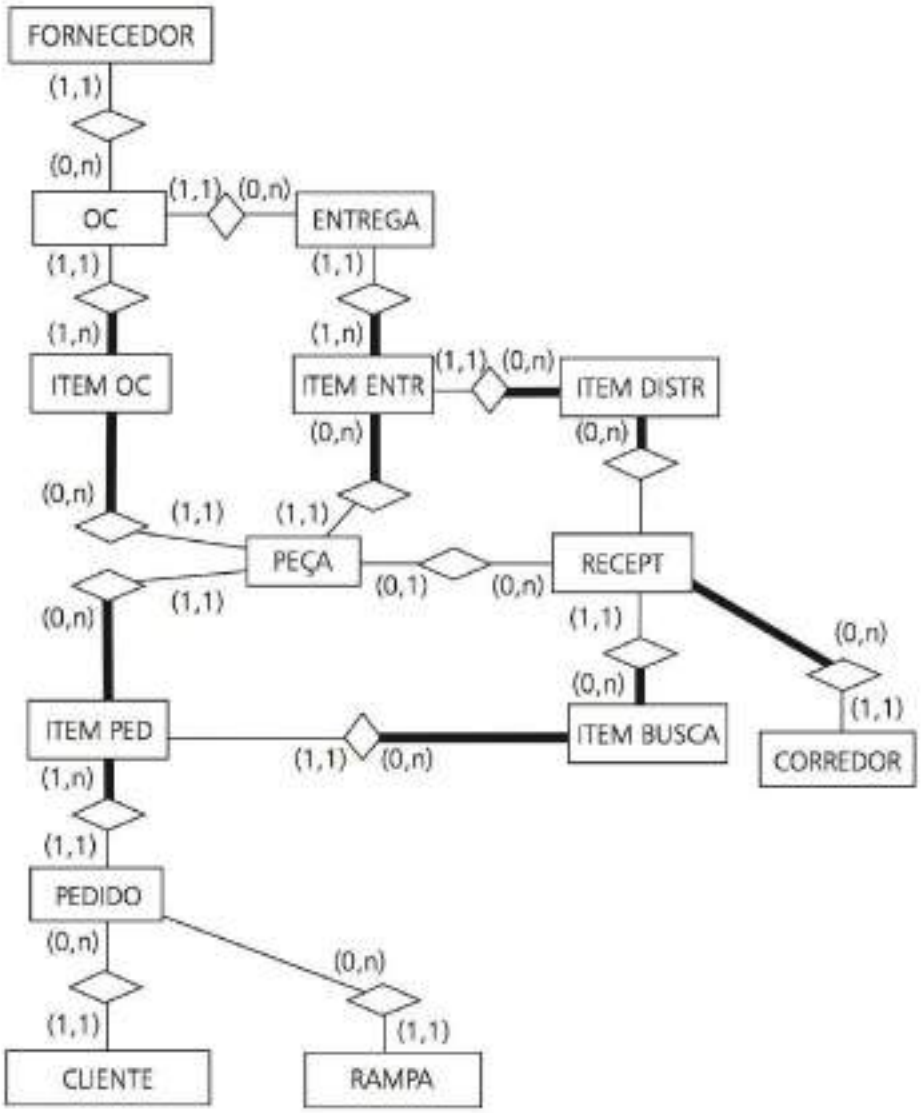


Figura 7.21 Diagrama ER para o sistema de almoxarifado.

exercício 4.1

Em um banco de dados relacional, a chave primária de uma tabela é aquela chave única utilizada como chave estrangeira em tabelas que referenciam a chave em questão. No caso do exercício, `CodigoEmpregado` é a chave primária da tabela Empregado, já que ela é usada na tabela `Dependente` como chave estrangeira que referencia `Empregado`.

exercício 4.2

Em organização de arquivos, o termo *chave secundária* é usado para denotar um campo ou conjunto de campos para os quais existe um índice que admite duplicatas. Conforme mencionado na Seção chave, na abordagem relacional o conceito de chave possui a conotação de *restrição de integridade* e não de caminho de acesso. Por este motivo, o termo chave secundária não é usado na abordagem relacional.

exercício 4.3

A Figura 7.22 mostra as chaves primárias e estrangeiras para o modelo do exercício.

exercício 4.4

1 Uma linha é incluída na tabela `Consulta`.

A tabela `Consulta` contém duas chaves estrangeiras, `(CodigoConvenio,NumeroPaciente)` e `CRM`. Quando ocorrer uma in-

```
Aluno (CodigoAluno,Nome,CodigoCurso)
    CodigoCurso referencia Curso
Curso(CodigoCurso,Nome)
Disciplina(CodigoDisciplina,Nome,Creditos,CodigoDepartamento)
    CodigoDepartamento referencia Departamento
Curriculo(CodigoCurso,CodigoDisciplina,Obrigatória-Opcional)
    CodigoCurso referencia Curso
    CodigoDisciplina referencia Disciplina
Conceito(CodigoAluno,CodigoDisciplina,Ano-Semestre,Conceito)
    CodigoAluno referencia Aluno
    CodigoDisciplina referencia Disciplina
Departamento(CodigoDepartamento,Nome)
```

Figura 7.22 Modelo relacional para o Exercício 4.3.

clusão em `Consulta`, é necessário verificar se estas chaves aparecem nas respectivas tabelas (`Paciente` e `Medico`).

2 Uma linha é excluída da tabela `Paciente`.

A tabela `Paciente` é referenciada em outra tabela (`Consulta`) por uma chave estrangeira. Assim, ao realizar a exclusão, é necessário verificar se não mais existem linhas em `Consulta` que referenciem a linha de `Paciente` que está sendo excluída.

■ exercício 5.1

A alternativa 1 (`Aluno(CodAl,Nome,CodCurso,Endereco)`) é preferível, caso considerarmos os princípios nos quais estão baseadas as regras de tradução de modelos ER para modelos relacionais. Os princípios são (página 138):

- Evitar junções – ter os dados necessários a uma consulta em uma única linha
- Diminuir o número de chaves primárias
- Evitar campos opcionais

A alternativa 1 atende aos dois primeiros princípios. O terceiro princípio não é considerado neste caso, já que nenhuma das alternativas implica em criar campos opcionais.

Quanto ao primeiro princípio, a alternativa 1 minimiza a necessidade de junções. Na alternativa 2, cada vez que forem necessários dados do aluno junto com dados de seu endereço será necessário fazer uma junção entre as duas tabelas.

Quanto ao segundo princípio, a alternativa 1 implica em um menor número de chaves primárias, já que implica em uma única tabela.

Conforme mencionado na Seção Transformação ER para relacional, as regras de tradução apresentadas representam um conjunto de heurísticas que, em geral, levam a um banco de dados bem projetado. Podem existir casos em que as regras não devem ser adotadas, principalmente por limitações do SGBD ou considerações de performance. Assim, podem haver situações em que a alternativa 2 é usada, mesmo violando os princípios acima. A seguir consideramos uma situação em que a alternativa 2 poderia ser preferível.

Considere que os dados de aluno (nome e código do curso) e endereço sofrem constantemente alterações de valores. Além disso, considere que o nome e

o código do aluno são alterados através de transações diferentes das que alteram o endereço do aluno e que estas alterações ocorrem de forma concorrente. A maioria dos SGBDs relacionais não permite que duas transações diferentes alterem a mesma linha concorrentemente. Neste caso, a alternativa 1 implica em *seqüencializar* as transações que alteram informações referentes a um mesmo aluno, fazendo com que uma transação tenha que esperar pela outra. Já na alternativa 2, como os dados estão em tabelas diferentes, as transações podem ser executadas de forma concorrente.

■ exercício 5.2

A Figura 7.23 mostra o esquema relacional referente ao modelo ER da Figura 5.30.

Em relação a este modelo, cabem os seguintes comentários.

O relacionamento entre *FABRICANTE* e *PRODUTO* foi implementado através da chave estrangeira `CNPJ` dentro da tabela `Produto`. Como o relacionamento é identificador, a chave estrangeira faz parte da chave primária da tabela.

- A especialização de *PRODUTO* foi implementada através de tabela única (ver Seção Relacionamentos 1:1 para uma discussão de alternativas). Para identificar o tipo de produto (medicamento ou perfumaria) foi criada uma coluna (`TipoProd`) na tabela `Produto`.
- A entidade associativa (relacionamento entre *VENDA* e *MEDICAMENTO*) foi implementada como um relacionamento **n:n** através de tabela própria.

```
Produto (CNPJ,NúmeroProd,NomeComercial,TipoEmbalagem,Quantidade,
         PreçoUnitário, TipoProd,Tarja,Fórmula,Tipo)
    CNPJ referencia Fabricante
Fabricante (CNPJ,Nome,Endereço)
Venda(Data,NúmeroNota,NomeCliente, CidadeCliente)
PerfumariaVenda(CNPJ,NúmeroProd,NúmeroNota, Quantidade,Imposto)
    (CNPJ,NúmeroProd) referencia Produto
  NúmeroNota referencia Venda
MedicamentoVenda(CNPJ,NúmeroProd,NúmeroNota,
                 Quantidade,Imposto,CRM,Número)
    (CNPJ,NúmeroProd) referencia Produto
NúmeroNota referencia Venda
    (CRM,Número) referencia ReceitaMédica
ReceitaMédica(CRM,Número,Data)
```

Figura 7.23 Modelo relacional para o Exercício 5.2.

- O relacionamento com *RECEITA MÉDICA* foi implementado através de chave estrangeira, dentro da tabela referente à entidade associativa.

exercício 5.3

A Figura 7.24 mostra o esquema relacional referente ao modelo ER da Figura 5.31.

A implementação segue as regras apresentadas. A única decisão tomada foi a de implementar a especialização de *TIPO DE VEÍCULO* através de uma tabela para cada entidade especializada (ver discussão na Seção Implementação de generalização/especialização).

exercício 5.4

A Figura 7.25 apresenta o modelo ER obtido através da aplicação das regras de engenharia reversa de modelos relacionais.

Os atributos de cada entidade/relacionamento estão listados na Figura 7.26 (os atributos identificadores estão sublinhados).

```
Escritório (NúmeroEscr,Local)
Contrato aluguel (NúmeroEscr,NúmeroContr,Data,Duração, NúmeroVeic,
                  NúmeroCartMotorista, EstadoCartMotorista)
    NúmeroEscr referencia Escritório
    NúmeroVeic referencia Veículo
    (NúmeroCartMotorista,EstadoCartMotorista)
                                  referencia Cliente
Cliente (NúmeroCartMotorista,EstadoCartMotorista,
                                  Nome,Endereço,Telefone)
Veículo (Número,DataPróximaManutenção,Placa,CódigoTipo)
    CódigoTipo referencia TipoVeiculo
TipoVeículo (CódigoTipo,Nome,ArCondicionado)
Similaridade (CódigoTipo,CódigoTipoSimilar)
    CódigoTipo referencia TipoVeiculo
    CódigoTipoSimilar referencia TipoVeiculo
Automóvel (CódigoTipo,NúmeroPortas,DireçãoHidráulica,
           CâmbioAutomático,Rádio)
    CódigoTipo referencia TipoVeículo
Ônibus (CódigoTipo,NúmeroPassageiros,Leito,Sanitário)
    CódigoTipo referencia TipoVeículo
```

Figura 7.24 Modelo relacional para o Exercício 5.3.

Figura 7.25 Modelo ER obtido através de engenharia reversa para o Exercício 5.4.

A aplicação das regras é direta. No modelo, apenas foram apresentadas as cardinalidades que puderam ser obtidas a partir do modelo ER:

- A cardinalidade do relacionamento *ITEM VENDA* é **n:n**, pois ele representa uma tabela cuja chave primária contém múltiplas chaves estrangeiras. A regra não determina a cardinalidade mínima.
- O mesmo se aplica ao relacionamento *SIMILAR*.
- Cada ocorrência da entidade *PRODUTO* está associada a exatamente uma ocorrência de *TIPO DE PRODUTO*. Esta cardinalidade é obtida pela seqüência de raciocínio a seguir:
  - O relacionamento representa a chave estrangeira `CodigoTipoProd` da tabela `Produto`. Como no modelo relacional todos os campos são mo-

```
Produto (NumeroProd,DescricaoProd,PreçoProd)
TipoProd (CodigoTipoProd,DescricaoTipoProd)
Venda (NúmeroNF,DataVenda)
ItemVenda (QtdeItem,PreçoItem)
Registradora (CodReg,SaldoReg)
Empregado (CodEmp,NomeEmp,SenhaEmp)
```

Figura 7.26 Atributos do modelo ER do Exercício 5.4.

novalorados, conclui-se que cada produto pode estar associado a, no máximo, um tipo de produto (cardinalidade máxima **1**).
  - Além disso, a chave estrangeira `CodigoTipoProd` faz parte de uma chave primária. Colunas que fazem parte da chave primária são obrigatórias. Daí conclui-se que cada produto deve estar obrigatoriamente associado a um tipo de produto (cardinalidade mínima **1**).
- O relacionamento entre *VENDA* e *EMPREGADO* representa uma chave estrangeira dentro da tabela `Venda` que não faz parte da chave primária. A única restrição de cardinalidade que pode ser derivada do modelo relacional é que cada venda tem a ela associado no máximo um empregado (cardinalidade máxima **1**). Como o modelo relacional do exercício é incompleto e não informa quais colunas são obrigatórias e quais colunas são opcionais, não é possível determinar se uma venda está obrigatoria ou opcionalmente associada a um empregado. Já na outra direção do relacionamento (quantas vendas estão associadas a um empregado), não é possível derivar alguma informação referente à cardinalidade a partir do modelo relacional fornecido.
- O mesmo aplica-se ao relacionamento entre *VENDA* e *REGISTRADORA*.

exercício 5.5

A Figura 7.27 apresenta o diagrama ER obtido por engenharia reversa do modelo relacional do exercício.

Os atributos das entidades e relacionamentos estão listados abaixo (atributos identificadores sublinhados):

```
Pessoa(PessID,PessNome,DataNasc,DataFalec,Sexo)
Local(LocID,Cidade,País)
Profissão(ProfID,ProfNome)
Casamento(CasamID,DataCasam)
```

exercício 6.1

Considerando apenas o conteúdo atual da tabela apresentada na Figura 6.8, as dependências funcionais nela contidas são:

```
(A,B) → C
A → D
D → A
```

Figura 7.27 Modelo ER obtido por engenharia reversa.

■ exercício 6.2

A passagem à segunda forma normal (2FN) consta da eliminação de *dependências funcionais parciais*, isto é, de colunas não-chave que dependem apenas de uma parte da chave primária e não da chave primária completa. No exercício, as dependências funcionais parciais são:

```
(CodigoTipoProd,NumeroProd) → DescricaoProd
CodigoTipoProd → DescricaoTipoProd
NúmeroNF → DataVenda
NúmeroNF → CodReg
NúmeroNF → CodEmp
NúmeroNF → NomeEmp
```

Observe que o identificador de um produto é a chave composta pelo código do tipo do produto e pelo número do produto. A passagem à 2FN resulta nas seguintes tabelas:

2FN

```
ItemVenda (NúmeroNF,CodigoTipoProd,NumeroProd,QtdeItem,PreçoItem)
Produto (CodigoTipoProd,NumeroProd,DescricaoProd)
TipoProd (CodigoTipoProd,DescricaoTipoProd)
Venda (NúmeroNF,DataVenda,CodReg,CodEmp,NomeEmp)
```

A passagem à 3FN consta da eliminação das *dependências funcionais transitivas* ou *indiretas*, isto é, de colunas não-chave que dependem de outras colunas não-chave. No exercício, há uma dependência funcional transitiva na tabela Venda que é CodEmp → NomeEmp. Devido a esta dependência, na passagem à 3FN é criada a tabela Empregado. O modelo relacional resultante da passagem à 3FN é o seguinte:

3FN

```
ItemVenda (NúmeroNF,CodigoTipoProd,NumeroProd,QtdeItem,PreçoItem)
Produto (CodigoTipoProd,NumeroProd,DescricaoProd)
TipoProd (CodigoTipoProd,DescricaoTipoProd)
Venda (NúmeroNF,DataVenda,CodReg,CodEmp)
Empregado (CodEmp,NomeEmp)
```

■ exercício 6.3

A tabela não se encontra na 2FN pois contém dependências funcionais parciais. A passagem a 2FN resulta nas seguintes tabelas:

```
(CodAluno,CodTurma)
(CodAluno,NomeAluno,CodLocalNascAluno,NomeLocalNascAluno)
(CodTurma,CodDisciplina,NomeDisciplina)
```

As duas últimas tabelas não estão na 3FN, já que contém dependências transitivas. Ao eliminá-las, temos o seguinte modelo relacional:

```
(CodAluno,CodTurma)
(CodAluno,NomeAluno,CodLocalNascAluno)
(CodLocalNascAluno,NomeLocalNascAluno)
(CodTurma,CodDisciplina)
(CodDisciplina,NomeDisciplina)
```

■ exercício 6.4

A seguir apresentamos cada uma das formas normais referentes ao documento. A primeira apresentada é chamada de não-normalizada (ÑN) pois não foi verificada quanto a alguma das formas normais.

ÑN

```
(CodCongr,NomeCongr,
        (CodGT,NomeGT),
        (NumeroArt,TitArt,AssPrincArt,
                (CodAutor,NomeAutor)))
```

1FN

(CodCongr,NomeCongr)
(CodCongr,CodGT,NomeGT)
(CodCongr,NumeroArt,TitArt,AssPrincArt)
(CodCongr,NumeroArt,CodAutor,NomeAutor)

2FN

(CodCongr,,NomeCongr)
(CodCongr,CodGT)
(CodGT,NomeGT)
(CodCongr,NumeroArt,TitArt,AssPrincArt)
(CodCongr,NumeroArt,CodAutor)
(CodAutor,NomeAutor)

3FN=2FN

■ exercício 6.5

A seguir, são apresentadas as formas normais:

ÑN

Comentário: Os campos NO-DE-AUTORES, NO-DE-REVISORES e NO-DE-TEMAS contêm informações redundantes, servindo para controlar o número de ocorrências dos grupos repetidos AUTOR, TEMA e REVISOR. Por este motivo, de acordo com a regra descrita na Seção Problemas da normalização na página 207, estes campos são eliminados já na passagem à forma ÑN.

```
(CodCongr,NumeroArt,NomeCongr,TitArt,
                    (CodAutor,NomeAutor)
                    (CodTema,NomeTema)
                    (CodRevisor,NomeRevisor,StatusRevisao))
```

1FN

(CodCongr,NumeroArt,NomeCongr,TitArt)
(CodCongr,NumeroArt,CodAutor,NomeAutor)
(CodCongr,NumeroArt,CodTema,NomeTema)
(CodCongr,NumeroArt,CodRevisor,NomeRevisor,StatusRevisao)

2FN

(CodCongr,NumeroArt,TitArt)
(CodCongr,,NomeCongr)
(CodCongr,NumeroArt,CodAutor)

```
(CodAutor,NomeAutor)
(CodCongr,NumeroArt,CodTema)
(CodTema,NomeTema)
(CodCongr,NumeroArt,CodRevisor,StatusRevisao)
(CodRevisor,NomeRevisor)
```

3FN=2FN

■ exercício 6.6

O documento é um bom exemplo das eventuais dificuldades da normalização de documentos preparados para leitura por pessoas e não para processamento eletrônico.

ÑN

```
(CodCongr,NomeCongr,
          (Data,
               (Hora,TituloSessao,CodModerador, NomeModerador,
                    (CodArt,TituloArt,
                           (CodAutor,NomeAutor,
                                    PaisAutor)))))
```

Comentários:

- Algumas sessões de um congresso, como a de registro (*registration*) ou a de intervalo (*break*) não possuem moderador. As sessões técnicas, nas quais são apresentados artigos, possuem um moderador. Para simplificar, considerou-se que cada sessão possui um campo "moderador" e que este campo é opcional.
- O documento contém chaves primárias propositadamente omitidas (ver Seção Problemas da normalização na pág. 205). No caso, foram omitidos o número do artigo, o código do autor e o código do moderador. Conforme explicado naquela seção, estes campos devem ser tratados como se aparecessem no documento.
- O documento informa o país dos autores. Esta informação aparece fatorada para todos os autores de um artigo que são do mesmo país. Para simplificar tratou-se o documento como se a informação aparecesse para cada autor.

1FN

```
(CodCongr,NomeCongr)
(CodCongr,Data)
```

```
(CodCongr,Data,Hora,TituloSessao,CodModerador,NomeModerador)
(CodCongr,Data,Hora,CodArt,TituloArt)
(CodCongr,Data,Hora,CodArt,CodAutor,NomeAutor,PaisAutor)
```

Comentário: Cada artigo é apresentado uma única vez. Essa informação é necessária para determinar a chave primária da quarta e da quinta tabelas. Nestas tabelas, a chave primária da tabela não é a concatenação das colunas chave primária das tabelas na forma ÑN, como foi nos exemplos até aqui apresentados. Para detalhes, ver Seção Passagem à primeira forma normal (1FN) na página 190.

2FN

```
(CodCongr,NomeCongr)
(CodCongr,Data
(CodCongr,Data,Hora,TituloSessao,CodModerador,NomeModerador)
(CodCongr,CodArt,TituloArt,Data,Hora)
(CodCongr,CodArt,CodAutor
(CodAutor,NomeAutor,PaisAutor)
```

3FN

```
(CodCongr,NomeCongr)
(CodCongr,Data
(CodCongr,Data,Hora,TituloSessao,CodModerador)
(CodModerador,NomeModerador)
(CodCongr,CodArt,TituloArt,Data,Hora)
(CodCongr,CodArt,CodAutor
(CodAutor,NomeAutor,PaisAutor)
```

■ exercício 6.7

Comentários:

- O documento é uma carta que é enviada para o autor principal do artigo. Assim, o documento é identificado pelo código do congresso e o número do artigo (identificação de artigo).
- Como no exercício anterior, há uma chave primária omitida, que é o identificador do autor.
- O documento não contém grupos multivalorados de dados. Por isso, não há tabelas aninhadas.

ÑN

```
(CodAutorPrinc,NomeAutor,InstituicaoAutor,RuaNumAutor,
    CidadeAutor,EstadoAutor,PaisAutor,NumArtigo,
```

```
    TituloArtigo,CodCongr,NomeCongr,DataCongr,
    LocalCongr,DataLimCongr)
```

1FN=ÑN

2FN

```
(CodCongr,NumArtigo,CodAutorPrinc,NomeAutor,
    InstituicaoAutor,RuaNumAutor,CidadeAutor,EstadoAutor,
    PaisAutor,TituloArtigo)
(CodCongr,NomeCongr,DataCongr,LocalCongr, DataLimCongr)
```

3FN

```
(CodCongr,NumArtigo,CodAutorPrinc,TituloArtigo)
(CodAutorPrinc,NomeAutor,InstituicaoAutor,RuaNumAutor, CidadeAutor,
    EstadoAutor,PaisAutor)
(CodCongr,NomeCongr,DataCongr,LocalCongr,DataLimCongr)
```

■ exercício 6.8

ÑN

```
(CodCongr,NomeCongr,DataInscrCongr,
    (CodGT,NomeGT),
    (CodInscr,NomeInscr,PaisInscr))
```

1FN

```
(CodCongr,NomeCongr,DataInscrCongr)
(CodCongr,CodGT,NomeGT)
(CodCongr,CodInscr,NomeInscr,PaisInscr)
```

2FN

```
(CodCongr,NomeCongr,DataInscrCongr)
(CodCongr,CodGT)
(CodGT,NomeGT)
(CodCongr,CodInscr)
(CodInscr,NomeInscr,PaisInscr)
```

3FN=2FN

■ exercício 6.9

Os modelos normalizados de cada documento estão listados a seguir. Cada tabela recebeu um nome, facilitando a identificação das tabelas que devem ser juntadas.

modelo do exercício 6.4

```
Congresso(CodCongr,NomeCongr)
CongrGT(CodCongr,CodGT)
GT(CodGT,NomeGT)
Artigo(CodCongr,NumeroArt,TitArt,AssPrincArt)
ArtigoAutor(CodCongr,NumeroArt,CodAutor)
Autor(CodAutor,NomeAutor)
```

modelo do exercício 6.5

```
Artigo(CodCongr,NumeroArt,TitArt)
Congresso(CodCongr,NomeCongr)
ArtigoAutor(CodCongr,NumeroArt,CodAutor)
Autor(CodAutor,NomeAutor)
CongrTema(CodCongr,NumeroArt,CodTema)
Tema(CodTema,NomeTema)
CongrRevisao(CodCongr,NumeroArt,CodRevisor,StatusRevisao)
Revisor(CodRevisor,NomeRevisor)
```

modelo do exercício 6.6

```
Congresso(CodCongr,NomeCongr)
CongrData(CodCongr,Data)
Sessão(CodCongr,Data,Hora,TituloSessao,CodModerador)
Moderador(CodModerador,NomeModerador)
Artigo(CodCongr,CodArt,TituloArt,Data,Hora,)
ArtigoAutor(CodCongr,CodArt,CodAutor)
Autor(CodAutor,NomeAutor,PaisAutor)
```

modelo do exercício 6.7

```
Artigo(CodCongr,NumArtigo,CodAutorPrinc,TituloArtigo)
Autor(CodAutorPrinc,NomeAutor,InstituicaoAutor,
      RuaNumAutor,CidadeAutor,EstadoAutor,PaisAutor)
Congresso(CodCongr,NomeCongr,DataCongr,LocalCongr,DataLimCongr)
```

modelo do exercício 6.8

```
Congresso(CodCongr,NomeCongr,DataInscrCongr)
CongrGT(CodCongr,CodGT)
GT(CodGT,NomeGT)
CongrInscr(CodCongr,CodInscr)
Inscrito(CodInscr,NomeInscr,PaisInscr)
```

modelo integrado

```
Artigo(CodCongr,NumeroArtigo,TituloArt,AssPrincArt,
       CodAutorPrinc,Data,Hora)
ArtigoAutor(CodCongr,NumeroArt,CodAutor)
Pessoa(CodPess,NomePess,Instituicao,RuaNum,Cidade,Estado,Pais)
Congresso(CodCongr,NomeCongr,DataCongr,LocalCongr,
          DataLimCongr,DataInscrCongr)
CongrGT(CodCongr,CodGT)
CongrInscr(CodCongr,CodInscr)
ArtRevisao(CodCongr,NumeroArt,CodRevisor,StatusRevisao)
ArtTema(CodCongr,NumeroArt,CodTema)
GT(CodGT,NomeGT)
Sessão(CodCongr,Data,Hora,TituloSessao,CodModerador)
Tema(CodTema,NomeTema)
```

Comentários:

- Nos diferentes modelos, a mesma tabela aparece sob diferentes nomes (o problema dos sinônimos descrito na Seção Integração de tabelas com mesma chave na página 210). Trata-se de uma única tabela, que deve constar uma única vez no modelo integrado. Por exemplo, as várias tabelas que contêm dados de pessoas (autor, moderador, revisor,...) foram unidas na tabela `Pessoa`. O mesmo problema de sinônimos aplica-se a campos. O campo número do artigo aparece uma vez sob a denominação `CodArtigo` e outra sob a denominação `NumArtigo`.
- A tabela `CongrData` foi eliminada pelo fato de seus dados estarem contidos na tabela `Sessão`. Para detalhes, ver Seção Integração de tabelas com chaves contidas na página 211.

exercício 6.10

```
Artigo(CodCongr,NumeroArtigoTituloArt,AssPrincArt,
       CodAutorPrinc,Data,Hora)
   CodCongr referencia Congresso
   (CodCongr,Data,Hora) referencia Sessão
   CodAutorPrinc referencia Pessoa
ArtigoAutor(CodCongr,NumeroArt,CodAutor)
   (CodCongr,NumeroArt) referencia Artigo
   CodAutor referencia Pessoa
Pessoa(CodPess,NomePess,Instituicao,RuaNum,Cidade, Estado,Pais)
Congresso(CodCongr,NomeCongr,DataCongr,LocalCongr,
          DataLimCongr,DataInscrCongr)
```

```
    CongrGT(CodCongr,CodGT) [CodCongr referencia
    Congresso [CodGT referencia GT
CongrInscr(CodCongr,CodInscr)
    CodCongr referencia Congresso
    CodInscr referencia Pessoa
ArtRevisao(CodCongr,NumeroArt,CodRevisor,StatusRevisao)
    (CodCongr,NumeroArt) referencia Artigo
    CodRevisor referencia Pessoa
ArtTema(CodCongr,NumeroArt,CodTema)
    (CodCongr,NumeroArt) referencia Artigo
    CodTema referencia Tema
GT(CodGT,NomeGT)
Sessão(CodCongr,Data,Hora,TituloSessao,CodModerador)
    CodCongr referencia Congresso
    CodModerador referencia Pessoa
Tema(CodTema,NomeTema)
```

Observações:

- As várias colunas que compõem uma chave estrangeira não necessariamente foram mostradas juntas na tabela. Este é o caso da chave estrangeira composta pelas colunas `CodCongr`, `Data` e `Hora`, que aparece na tabela `Artigo` e que referencia a tabela `Sessão`.
- A chave estrangeira não necessariamente tem o mesmo nome da chave primária que referencia. Este é o caso das várias chaves estrangeiras que referenciam a tabela `Pessoa` no exemplo.

■ exercício 6.12

A Figura 7.28 apresenta o diagrama ER obtido através da engenharia reversa do modelo relacional obtido no exercício precedente. Os atributos das entidades e dos relacionamentos do diagrama ER são os seguintes:

```
Artigo(NumeroArtigo,TituloArt,AssPrincArt)
Pessoa(CodPess,NomePess,Instituicao,RuaNum,Cidade, Estado,Pais)
Congresso(CodCongr,NomeCongr,DataCongr,LocalCongr,
          DataLimCongr,DataInscrCongr)
GT(CodGT,NomeGT)
Sessão(Data,Hora,TituloSessao)
Tema(CodTema,NomeTema)
```

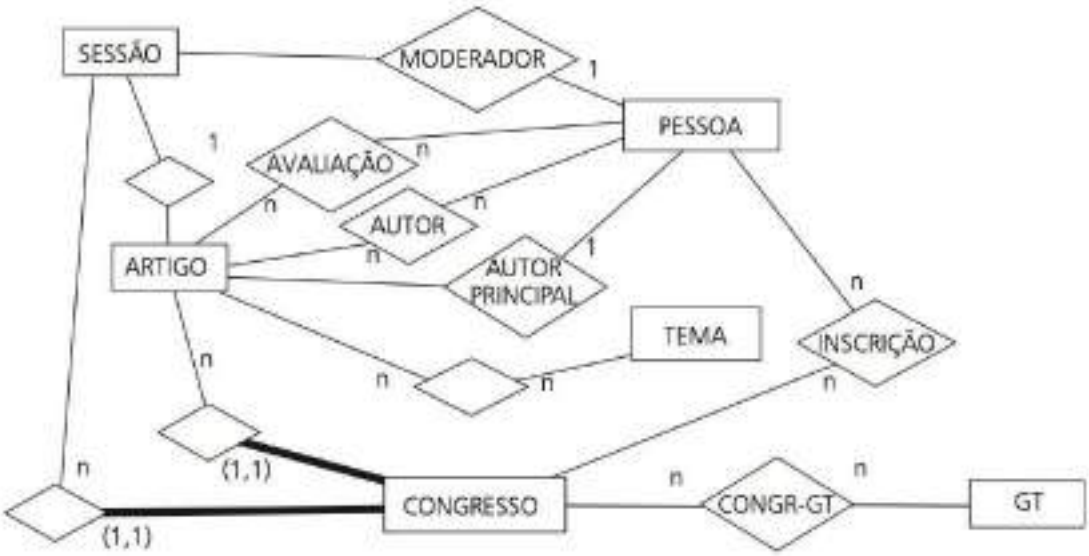


Figura 7.28 Diagrama ER obtido através de engenharia reversa.

Em linhas gerais, os símbolos do diagrama ER têm a mesma disposição que aquela apresentada no modelo do sistema de preparação de congressos que aparece na Figura 7.19. Com isso, é possível comparar o modelo obtido através da engenharia reversa com o modelo obtido através da estratégia descendente, usando como entrada do processo a descrição do problema.

O modelo obtido por engenharia reversa é diferente daquele obtido pela estratégia descendente partindo da descrição do problema (Figura 7.19). Vários elementos do modelo que haviam sido identificados na Figura 7.19 não aparecem no modelo, como os relacionamentos referentes aos membros do GT, aos membros dos comitês de programa e organizador. A engenharia reversa somente identifica os elementos do modelo ER que aparecem nos documentos ou arquivos tratados.

Por outro lado, da engenharia reversa resultaram alguns elementos (como a entidade *TEMA*) que não apareciam no modelo da Figura 7.19, já que eles não haviam sido referenciados na descrição do problema, mas aparecem nos arquivos e documentos processados na engenharia reversa.

O modelo obtido pela engenharia reversa não é necessariamente um modelo perfeito. Ele pode ainda conter anomalias e até mesmo redundâncias de dados, conforme descrito a seguir.

A engenharia reversa não detecta relacionamentos redundantes (ver Seção Um modelo deve ser livre de redundância na página 87). Este é o caso do relacionamento entre a entidade *SESSÃO* e a entidade *CONGRESSO*. O relacionamento é redundante, uma vez que as informações nele contidas podem ser obtidas através dos relacionamentos entre *SESSÃO* e *ARTIGO* e entre *ARTIGO* e *CONGRESSO*. Este relacionamento aparece no modelo, pelo fato de o código do congresso fazer parte da chave primária da entidade *SESSÃO*.

Outra anomalia do modelo obtido pela engenharia reversa são os relacionamentos entre um artigo e a pessoa que é autora do artigo. Foram identificados dois relacionamentos, um que informa o autor principal de cada artigo e outro que informa o conjunto de todos os autores do artigo. Do ponto de vista da modelagem conceitual, estes relacionamentos não estão incorretos. Entretanto, do ponto de vista do desenvolvimento de aplicações sobre o banco de dados em questão, a separação dos autores de um artigo em dois conjuntos complica o tratamento (alterações e consultas) de autores, já que exige um tratamento especial para o autor principal do artigo. O tratamento de autores é simplificado se for utilizado um único relacionamento de autoria, tendo este relacionamento um atributo, que informa se o autor é ou não o autor principal do artigo.

Além dos problemas acima mencionados, o modelo demonstra o resultado de um documento conter uma chave primária implícita e de ela não ter sido tratada durante a normalização (ver Seção Problemas da normalização na página 207). Na entidade *ARTIGO*, aparece uma coluna denominada `AssPrincArt`. Esta coluna contém a descrição do assunto ou *tema* principal do artigo. Esta informação não deveria aparecer aqui na forma de um atributo de artigo, e sim, de um relacionamento entre as entidades *ARTIGO* e *TEMA*. O problema ocorreu pelo fato de não termos detectado, quando da normalização do documento referente ao Exercício 6.4, o fato de todo assunto ou tema possuir um código. Deveríamos ter incluído, já na normalização daquele documento, o código do tema. Fica como exercício para o leitor a revisão do resultado dos exercícios incorporando esta correção.

exercício 6.13

ÑN

```
(NoCorr,
        (NoRecept,CodPeca,DescrPeca,QtdeEstoq))
```

Observação: O campo data de emissão do documento foi desconsiderado (ver discussão sobre campos irrelevantes na Seção Problemas da normalização na página 207).

1FN
(NoCorr)
(NoCorr,NoRecept,CodPeca,DescrPeca,QtdeEstoq)

2FN = 1FN

3FN
(NoCorr)
(NoCorr,NoRecept,CodPeca,QtdeEstoq)
(CodPeca,DescrPeca)

■ exercício 6.14

ÑN
```
(CodOC,DataEntrega,NoEstrado,CodEmpilh,CodOperEmpilh,
                    (NoCorr,
                            (NoRecept,CodPeca,DescrPeca,
                                 QtdePeca)))
```

Observação: Os campos NO-DE-CORREDORES e NO-DE-RECEPTACULOS foram desconsiderados por serem redundantes, servindo apenas para controlar o tamanho dos itens de grupo do COBOL (ver Seção Problemas da normalização na página 207).

1FN
(CodOC,DataEntrega,NoEstrado,CodEmpilh,CodOperEmpilh)
(CodOC,DataEntrega,NoCorr)
(CodOC,DataEntrega,NoCorr,NoRecept,CodPeca,DescrPeca,QtdePeca)

2FN = 1FN

3FN
(CodOC,DataEntrega,NoEstrado,CodEmpilh,CodOperEmpilh)
(CodOC,DataEntrega,NoCorr)
(CodOC,DataEntrega,NoCorr,NoRecept,CodPeca,QtdePeca)
(CodPeca,DescrPeca)

exercício 6.15

ÑN

(NoPed,CodCli,NomeCli,NoRampa)

1FN=ÑN

2FN=1FN

3FN

(NoPed,CodCli,NoRampa)
(CodCli,NomeCli)

exercício 6.16

ÑN

(NoOC,DataOC,CodFornec,NomeFornec,
    (CodPeca,DescrPeca,QuantPed,
        (DataEntr,QuantEntr)))

1FN

(NoOC,DataOC,CodFornec,NomeFornec)
(NoOC,CodPeca,DescrPeca,QuantPed)
(NoOC,CodPeca,DataEntr,QuantEntr)

2FN

(NoOC,DataOC,CodFornec,NomeFornec)
(NoOC,CodPeca,QuantPed)
(CodPeca,DescrPeca)
(NoOC,CodPeca,DataEntr,QuantEntr)

3FN

(NoOC,DataOC,CodFornec)
(CodFornec,NomeFornec)
(NoOC,CodPeca,QuantPed)
(CodPeca,DescrPeca)
(NoOC,CodPeca,DataEntr,QuantEntr)

exercício 6.17

ÑN

(NoPed,DataPed,CodCli,NomeCli,
    (NoTel),
    (CodPeca,DescrPeca,QuantPed))

1FN

(NoPed,DataPed,CodCli,NomeCli)
(NoPed,NoTel)
(NoPed,CodPeca,DescrPeca,QuantPed)

2FN

(NoPed,DataPed,CodCli,NomeCli)
(NoPed,NoTel)
(NoPed,CodPeca,QuantPed)
(CodPeca,DescrPeca)

3FN

(NoPed,DataPed,CodCli)
(CodCli,NomeCli)
(NoPed,NoTel)
(NoPed,CodPeca,QuantPed)
(CodPeca,DescrPeca)

Este exercício exemplifica o problema referido na página 190 (Seção Passagem à primeira forma normal (1FN)), quanto à opção de usar a decomposição de tabelas na passagem à 1FN. No exemplo, os números de telefone que aparecem são os números dos telefones do cliente. Entretanto, ao decompor a tabela ÑN, o telefone fica desvinculado do cliente correspondente, na tabela:

(NoPed, NoTel)

Quando o esquema relacional for transformado em um modelo ER, esta tabela corresponderá a uma entidade `Telefone` vinculada à entidade `Pedido` e não à entidade `Cliente`, como seria o correto. Deixamos a correção deste problema para depois da construção do modelo ER.

■ exercício 6.18

ÑN

```
(NoPed,DataPed,CodCli,NomeCli,
      (NoCorr,
            (NoRecept,CodPeca,DescrPeca,QuantBusca)))
```

1FN

(NoPed,DataPed,CodCli,NomeCli)
(NoPed,NoCorr)

(NoPed,NoCorr,NoRecept,CodPeca,DescrPeca,QuantBusca)

2FN = 1FN

3FN

(NoPed,DataPed,CodCli)
(CodCli,NomeCli)
(NoPed,NoCorr)
(NoPed,NoCorr,NoRecept,CodPeca,QuantBusca)
(CodPeca,DescrPeca)

exercício 6.19

Os modelos normalizados de cada documento estão listados a seguir. Cada tabela recebeu um nome, facilitando a identificação das tabelas que devem ser juntadas.

modelo do exercício 6.13

Corredor(NoCorr)
Receptáculo(NoCorr,NoRecept,CodPeca,QtdeEstoq)
Peca(CodPeca,DescrPeca)

modelo do exercício 6.14

Entrega(CodOC,DataEntrega,NoEstrado,CodOperEmpilh)
DistrCorr(CodOC,DataEntrega,NoCorr)
ItemDistr(CodOC,DataEntrega,NoCorr,NoRecept,CodPeca,QtdePeca)
Peca(CodPeca,DescrPeca)

modelo do exercício 6.15

Pedido(NoPed,CodCli,NoRampa)
Cliente(CodCli,NomeCli)

modelo do exercício 6.16

OC(NoOC,DataOC,CodFornec)
Fornec(CodFornec,NomeFornec)
ItemOC(NoOC,CodPeca,QuantPed)
Peca(CodPeca,DescrPeca)
ItemEntr(NoOC,CodPeca,DataEntr,QuantEntr)

modelo do exercício 6.17

Pedido(NoPed,DataPed,CodCli)
Cliente(CodCli,NomeCli)
Telefone(NoPed,NoTel)
ItemPedido(NoPed,CodPeca,QuantPed)
Peca(CodPeca,DescrPeca)

modelo do exercício 6.18

Pedido(NoPed,DataPed,CodCli)
Cliente(CodCli,NomeCli)
BuscaCorredor(NoPed,NoCorr)
ItemBusca(NoPed,NoCorr,NoRecept,CodPeca,QuantBusca)
Peca(CodPeca,DescrPeca)

modelo integrado

Corredor(NoCorr)
Receptáculo(NoCorr,NoRecept,CodPeca,QtdeEstoq)
Peca(CodPeca,DescrPeca)
Entrega(NoOC,DataEntrega,NoEstrado,CodOperEmpilh)
DistrCorr(NoOC,DataEntrega,NoCorr)
ItemDistr(NoOC,DataEntrega,NoCorr,NoRecept,CodPeca,QtdePeca)
Pedido(NoPed,CodCli,NoRampa,DataPed)
Cliente(CodCli,NomeCli)
OC(NoOC,DataOC,CodFornec)
Fornec(CodFornec,NomeFornec)
ItemOC(NoOC,CodPeca,QuantPed)
ItemEntr(NoOC,CodPeca,DataEntr,QuantEntr)
Telefone(NoPed,NoTel)
ItemPedido(NoPed,CodPeca,QuantPed)
BuscaCorredor(NoPed,NoCorr)
ItemBusca(NoPed,NoCorr,NoRecept,CodPeca,QuantBusca)

Observações:

- Na transcrição dos esquemas das tabelas ao modelo integrado, foi feito o tratamento de sinônimos (ver Seção Integração de tabelas com mesma chave na página 210). A coluna referente ao código da ordem de compra aparecia sob os nomes NoOC e CodOC.
- Foi feito o tratamento de tabelas com chaves contidas (ver Seção Integração de tabelas com chaves contidas na página 211). A tabela DistrCorr

(modelo do Exercício 6.14) foi eliminada por estar contida na tabela ItemDistr do mesmo exercício. Pela mesma razão, foi eliminada a tabela BuscaCorredor (modelo do Exercício 6.18). Deve ser observado que a tabela Corredor (modelo do Exercício 6.13) não foi integrada à tabela Receptáculo do mesmo modelo. Ocorre que podem haver estados do banco de dados nos quais um corredor já foi incluído na tabela de Corredor, mas não aparece ainda na tabela Receptáculo (estamos considerando que corredores e receptáculos são incluídos no banco de dados em transações diferentes). Este é o mesmo caso do terceiro exemplo da Seção Integração de tabelas com chaves contidas (página 211).

■ exercício 6.20

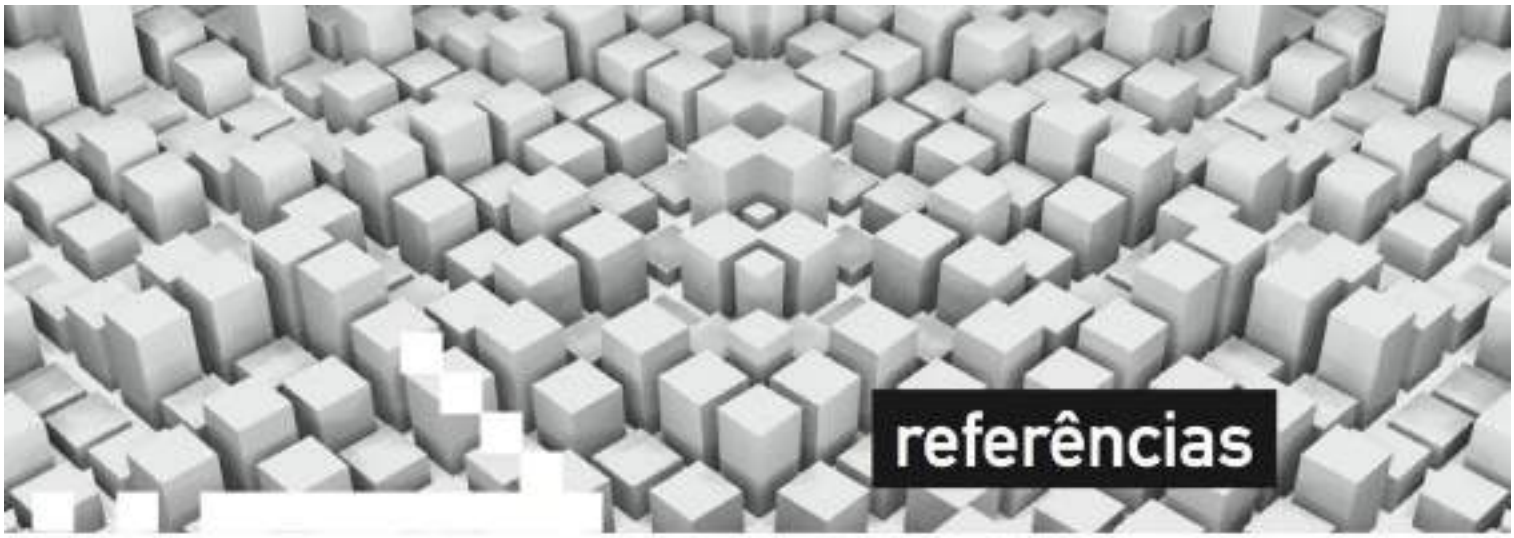
```
Corredor(NoCorr)
Receptáculo(NoCorr,NoRecept,CodPeca,QtdeEstoq)
    NoCorr referencia Corredor
    CodPeca referencia Peca
Peca(CodPeca,DescrPeca)
Entrega(NoOC,DataEntrega,NoEstrado,CodOperEmpilh)
    NoOC referencia OC
ItemDistr(NoOC,DataEntrega,NoCorr,NoRecept,CodPeca,QtdePeca)
    (NoOC,DataEntrega,CodPeca) referencia ItemEntr
    (NoCorr,NoRecept) referencia Receptáculo
```

Observação: NoOC não foi considerado como chave estrangeira da tabela OC, nem (NoOC, DataEntrega) foi considerado chave estrangeira da tabela Entr.

Estas chaves estrangeiras seriam redundantes com as demais chaves estrangeiras que aparecem no modelo. Fica como exercício para o leitor identificar as chaves estrangeiras com as quais estas seriam redundantes.

```
Pedido(NoPed,CodCli,NoRampa,DataPed)
    CodCli referencia Cliente
Cliente(CodCli,NomeCli)
OC(NoOC,DataOC,CodFornec)
    CodFornec referencia Fornec
Fornec(CodFornec,NomeFornec)
ItemOC(NoOC,CodPeca,QuantPed)
    NoOC referencia OC
    CodPeca referencia Peca
ItemEntr(NoOC,CodPeca,DataEntr,QuantEntr)
    (NoOC,DataEntr) referencia Entrega
```

```
    (NoOC,CodPeca) referencia ItemOC
Telefone(NoPed,NoTel)
    NoPed referencia Pedido
ItemPedido(NoPed,CodPeca,QuantPed)
    NoPed referencia Pedido
    CodPeca referencia Peca
ItemBusca(NoPed,NoCorr,NoRecept,CodPeca,QuantBusca)
    (NoPed,CodPeca) referencia Peca
    (NoCorr,NoRecept) referencia Receptáculo
```



referências

ABRIAL, J. Data semantics. In: DATA BASE MANAGEMENT, 1974, Amsterdam. *Anais...* Amsterdam: North-Holland, 1974. p.1–59.

BARKER, R. *Case*method*: entity relationship modelling. Wokingham: Addison-Wesley, 1990.

BATINI, C.; CERI, S.; NAVATHE, S. *Conceptual database design:* an entity relationship approach. Redwood City: Benjamin-Cummings, 1992.

CHEN, P. P. The entity-relationship model-toward a unified view of data. *ACM Trans. on Database Syst.*, Salt Lake City, v. 1, n. 1, p. 9–36, Mar. 1976.

CODD, E. F. A relational model of data for large shared data banks. *Commun. of the ACM*, New York, v. 13, n.6, p. 377–387, June 1970.

CODD, E. F. Further normalization of the data base relational model. In: RUSTIN, R. (Ed.). *Database systems*. Englewood Cliffs: Prentice Hall, 1972. p.33–64. (Courant Computer Symposia Series, v.6).

CODD, E. F. Normalized data structure: a brief tutorial. In: ACM-SIGFIDET WORKSHOP ON DATA DESCRIPTION, ACCESS AND CONTROL, 1971, San Diego. *Proceedings...* New York: ACM, 1971, p. 321-350.

DUMPALA, S.; ARORA, S. Schema translation using the entity-relationship approach. In: INTERNATIONAL CONFERENCE ON ENTITY RELATIONSHIP APPROACH, 3. th, 1983, Amsterdam. *Proceedings...* Amsterdam: North-Holland, 1983, p. 337-356

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados - Fundamentos e Aplicações*. 3.ed. Rio de Janeiro: LTC, 2002.

FOWLER, M.; SCOTT, K. *UML essencial*. 2.ed. Porto Alegre: Bookman, 2000.

GROFF, J. R.; WEINBERG, P. N. *SQL:* the complete reference. Berkeley: McGrawHill, 1999.

JOHANNESSON, P.; KALMAN, K. A method for translating relational schemas into conceptual schemas. In: ENTITY-RELATIONSHIP APPROACH TO DATABASE DESIGN AND QUERYING, PROCEEDINGS OF THE EIGHT INTERNATIONAL CONFERENCE ON ENITY-RELATIONSHIP APPROACH, 1989, TORONTO. *Anais*... Amsterdam: North-Holland, 1989. p.271–285.

KENT, W. *Data and reality*. Amsterdam: North-Holland, 1978.

KENT, W. Limitations of record-based information models. *ACM Trans. on Database Syst.*, Salt Lake City, v. 4, n. 1, p. 107–131, 1979.

KIPPER, E. F. et al. *Engenharia de informações*: conceitos técnicas e métodos. Porto Alegre: Sagra, 1993.

MANINILA, H.; RÄIHÄ, K. *The design of relational databases.* Wokingham: Addison-Wesley, 1992.

MARTIN, J. *Information engineering*. Upper Saddle River: Prentice-Hall, 1990.

NAVATHE, S. B.; AWONG, A. M. Abstracting relational and hierarchical data with a semantic data model. In: ENTITY-RELATIONSHIP APPROACH, PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE ON ENTITY-RELATIONSHIP APPROACH, 1987, NEW YORK. *Anais*... Amsterdam: North-Holland, 1987. p. 305–333.

NIJSSEN, G.; HALPIN, T. *Conceptual schema and relational database design*: a fact oriented approach. Upper Saddle River: Prentice-Hall, 1989.

PRESSMAN, R. S. *Engenharia de software*. 5. ed. Rio de Janeiro: Mc-Graw Hill, 2002.

SANTOS, C. S. dos; NEUHOLD, E. J.; FURTADO, A. L. A data type approach to the entity-relationship approach. In: ENTITY-RELATIONSHIP APPROACH TO SYSTEMS ANALYSIS AND DESIGN. PROCEEDINGS OF THE 1ST INTERNATIONAL

CONFERENCEONTHE ENTITY-RELATIONSHIP APPROACH, 1980, Los Angeles. *Anais*... Amsterdam: North-Holland, 1980. p. 103–119.

SCHEUERMANN, P.; SCHIFFNER, G.; WEBER, H. Abstraction capabilities and invariant properties modelling within the entity-relationship approach. In: ENTITY-RELATIONSHIP APPROACH TO SYSTEMS ANALYSIS AND DESIGN. PROCEEDINGS OF THE 1ST INTERNATIONAL CONFERENCE ON THE ENTITY-RELATIONSHIP APPROACH, 1980, Los Angeles. Anais... Amsterdam: North-Holland, 1980. p. 121–140.

SETZER, V. W. *Banco de dados*. 2.ed. São Paulo: Edgard Blücher, 1987.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistemas de Banco de Dados*. 3.ed. São Paulo: Makron Books, 1999.

SMITH, J. M.; SMITH, D. C. P. Database abstractions: aggregation and generalization. *ACM Trans. on Database Syst.*, Salt Lake City, v. 2, n. 2, p. 105–133, June 1977.

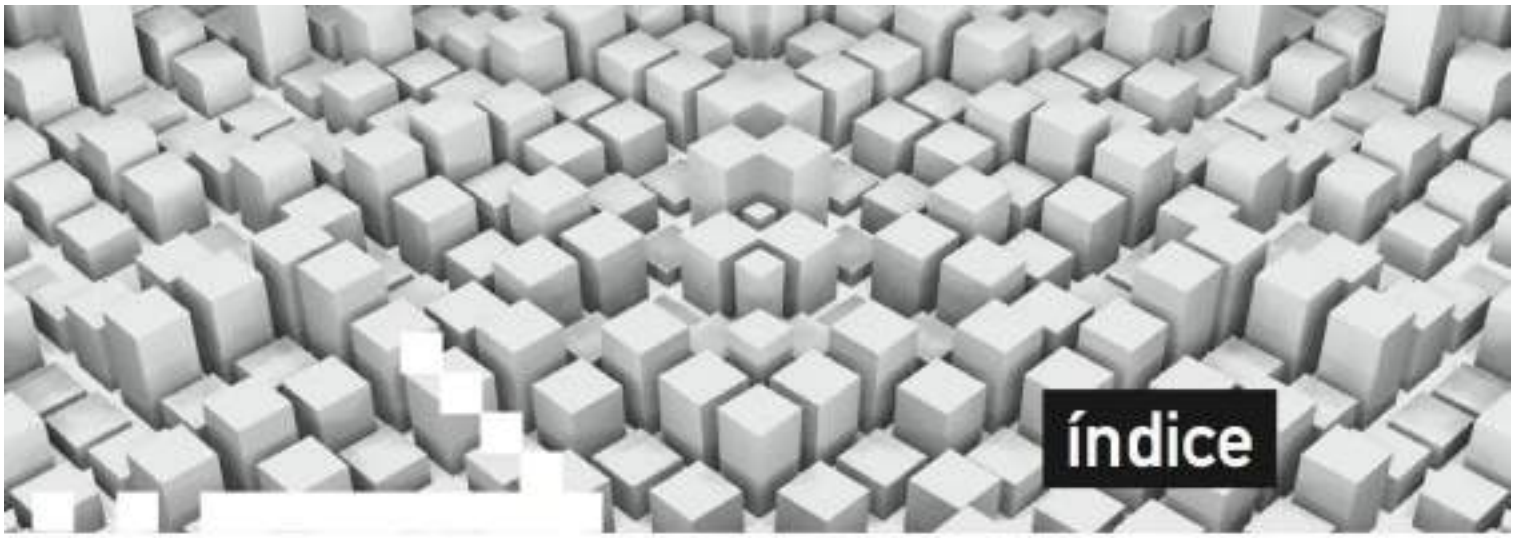
TARDIEU, H.; ROCHFELD, A.; COLLETI, R. *Le methode Merise*: principles et outils. Paris: Les Editions d'Organization, 1983.

TEOREY, T. *Database modeling & design: the fundamental principles*. 2nd. ed. San Francisco: Morgan Kaufmann, 1994.

TEOREY, T. J.; YANG, D.; FRY:, J. P. A logical design methodology for relational databases sing the extended entity-relationship model. *ACM Computing Surveys*, New York, v. 18, n. 2, p. 197-222, June 1986.

VERHEIJEN, G.; BEKKUM, J. van. NIAM: an information analysis method. In: IFIP TC-8 CONFERENCE ON COMPARATIVE REVIEW OF INFORMATION SYSTEMS METHODOLOGIES (CRIS-1), 1982. *Proceedings*... Amsterdam: North- Holland, 1982, p. 537–590.

WONG, E.; KATZ, R. H. Logical design and schema conversion for relational and DBTG databases. In: ENTITY-RELATIONSHIP APPROACH TO SYSTEMS ANALYSIS AND DESIGN. PROC. INTERNATIONAL CONFERENCE ON THE ENTITY-RELATIONSHIP APPROACH, 1., 1979, Los Angeles. *Anais*... Amsterdam: North-Holland, 1979. p. 311–322.



índice

Os números em itálico referem-se a páginas de exercícios corrigidos.

1:1 *ver* Relacionamento, **1:1**
1:n *ver* Relacionamento, **1:n**
n:n *ver* Relacionamento, **n:n**
1FN *ver* Forma normal, primeira
2FN *ver* Forma normal, segunda
3FN *ver* Forma normal, terceira
4FN *ver* Forma normal, quarta

Abordagem de modelagem de dados, *25*
Abordagem entidade-relacionamento *ver* Modelo entidade-relacionamento
Abordagem relacional, 120-131
Abordagem UML, *97-99*
Associação *ver* Relacionamento
Atributo, *48*, 48-54, 140, *241*
  cardinalidade de, 49
  critérios de escolha, 77-85
  de relacionamento, *49*
  domínio de, *49*, 79, 104, *242*
  e entidade relacionada, 78-79
  e especialização, 79-81
  identificador, *50-53*, *243*, *246*
  implícito na normalização, 208-209
  irrelevante ou derivado, 209
  modificado com o tempo, 89-90
  monovalorado, *49*
  multivalorado, *49*, 84-85, 165-167, *233*
  nome de, 121, 141-142
  obrigatório, *49*
  opcional, *49*, 83-84, *236-237*
  redundante, 87-89, 165, 209
  temporal, *89-90*
Auto-relacionamento, 38-39

Banco de dados, *22*
  compartilhamento de dados, 20-*22*
  consulta ao, 130-131
  engenharia reversa de, 29
  sintonia de, *29*

Campo, *121-122*
  domínio de, *126-127*
  monovalorado, 121
  multivalorado, 122, 154
  nome de, 121
Cardinalidade, *39*, 39-48, 104
  de atributo, 49
  máxima, 40-43, *239-240*
    máxima **1**, 40, 146
    máxima **n**, 40
  mínima, 45, 45-46, *242*
    mínima **0**, 45
    mínima **1**, 45, *235*
CASE *ver* Ferramentas de modelagem
Chave, *122-128*
  alternativa, *125-126*
  engenharia reversa, 171-172

estrangeira, *123-125*, 130, 145-146, *251*, *265*, *273*
primária, *122-123*, 143, 156, 197-198, *251*
nome, 141-142
omitida ou incorreta, 207-208
restrição de integridade, 123
secundária, *251*
Coluna, *121*, 140, 144-145
domínio de, *126*-127, 129-130, 210-211
engenharia reversa, 174-175
monovalorada, 121, 154
multivalorada, 121, 154, 187
nome de, 141-142
obrigatória, 127
opcional, 127, 140, 150
redundante, 164

Dependência funcional, *195-196*
indireta , *258*
multivalorada, *205*
parcial, *197*, *257*
transitiva, *258*
DER *ver* Diagrama entidade-relacionamento
Diagrama de ocorrências, *37*, 37-38, *230-231*
Diagrama entidade-relacionamento, 25-26, 34
símbolos utilizados no, 62
variantes de notação, 94-100
Dicionário de dados, 49, 101
Domínio de atributo, 49, 79, 104, *242*
Domínio de campo, 126-127
Domínio de coluna, 126-127, 129-130, 210-211

Engenharia de informações, notação, 95-97
Engenharia reversa, 102-103, *169-175*
de arquivo, 184-214, *258-262*, *267-271*
de modelos relacionais, 136, 169-175, *254-256*, *265-267*
processo, 170
Entidade, *34*, *34-36*, 140, *238*, *240*, *242*, *244*
associativa, 60-62
critérios de escolha, 77-85
e especialização, 81-83
fraca, 52
identificador de, 50-53, *241*, *242*, *246*
identificador mínimo, 52
implementação na abordagem relacional, 140-144
isolada, 94
ocorrência de, 35
papel de, *38*
sem atributos, 94
Entidade-relacionamento
abordagem, 25, 34-64, 136-137
diagrama, *25-26*
ER *ver* Entidade-relacionamento
Erro semântico, 86
Erro sintático, 86
Especialização *ver Generalização e especialização*
Esquema de banco de dados, 62-64
esquema gráfico *ver* Diagrama entidade-relacionamento
esquema textual, 62-64
relacional, 128-130
notação diagramática, 129-130
notação textual, 128-129

Ferramentas de modelagem, 100-102
ferramenta CASE, *72*, 94, 101, *242*
programas de propósito geral, 101-102
Forma normal, *190 ver também Normalização*
não-normalizada, *187-190*
primeira, *190*
quarta, *206*
segunda, *197*, *257*, *258*
terceira, *200*, *258*

Generalização e especialização, *54*, 54-60, *236-237*, *238*, *246*
compartilhada, *57-58*, *233-235*, *235*
exclusiva, *57*, *235*
formas de, 79-83
herança múltipla, *58*
implementação na abordagem relacional, 156-162
níveis de, 58-60
parcial, 56
total, 56, *233-235*
Gerência de banco de dados *ver* Sistema de gerência de banco de dados

Herança de propriedades, 55

Identificador de entidade, 50-53, *241-242*, *246*
Identificador de relacionamento, 53-54, 143, 163
Instância, *37*
Integração de modelos, 209-213, *262-264*, *271-273*
tabelas com a mesma chave, 210-211
tabelas com chaves contidas, 211-213
Integração de visões *ver* Integração de modelos
Integridade referencial, 127

Junção, 131, 138-139

Linguagem de modelagem de dados, *24*
Linguagem de terceira geração, 184
Linha, *120*

Modelo de banco de dados relacional, 128-130
modelo diagramático, 129-130
modelo textual, 128-129
Modelo de dados, 24, 24-29
conceitual, *25*, 25-26, 27-29, 169, 184
físico, *27*, 29
lógico, *26*, 26-27, 29, 136-137, 184
Modelo entidade-relacionamento, 25, *34*, *34-64*, 136-137
construção, 77-85, 102-105, 214, *240-242*
engenharia reversa, *102-103*, *169-175*
estratégia ascendente, 103
estratégia descendente, 103-104, *238-240*
estratégia *inside-out*, 104-105
especialização, 81-83
propriedades, 72-77
aspecto temporal, 89-93, *242*
capacidade de expressão limitada, 73-75, 87, *235-236*
correção, 85-87
equivalência, 75-77
precisão, 72-73, 87-89
redundância, 87-89, 165, 209
representação gráfica, 25-26, 34, 62-100
representação textual, 62-64
variantes, 94-102
abordagem UML, 97-99
notação engenharia de informações, 95-97
notação européia para cardinalidades, 99-100
verificação, 85-94, 214
Multiplicidade, 99

Normalização, 190-209
limitações, 214
passagem à primeira forma normal (1FN), 190-195
passagem à quarta forma normal (4FN), 203-207
passagem à segunda forma normal (2FN), 196-200
passagem à terceira forma normal (3FN), 200-203
problemas de, 207-209

retorno à segunda forma normal (2FN), 213
Notação européia para cardinalidade, 99-100, *237-238*
NULL ver Vazio (Valor de campo)

Primeira forma normal, *ver* Forma normal, primeira (1FN)
Projeto de banco de dados, 29
fase de projeto físico, 29
fase de projeto lógico, 29, 170, *252-253*
objetivos, 138
passos, 140
princípios, 138-140
visão geral, 136-137

Quarta forma normal, 206

Redundância de dados, *20, 20-22*
controlada, *21*
não-controlada, *21, 21-22*
Redundância em modelos ER *ver* Modelo entidade-relacionamento
Relacionamento, *36*, 36-46, 144, *238-239, 240-241, 243, 245-246*
1:1 (um-para-um), 40-43, 147-152, 151
1:n (um-para-muitos), 40-43, 152-154
n:n (muitos-para-muitos), 40-43, 154
transformação em entidade, 77, 155, *231-232*
atributo, *49*
auto-relacionamento, 38-39
binário, *40-41, 40-43*
de grau maior que dois, *43-45*, 154-156
identificador de, 53-54, 143, 163
implementação na abordagem relacional, 142-143
implementação, 144-156
adição de colunas, 146
fusão de tabelas, 146-147, 151
tabela própria, 144-145
modificado com o tempo, 90-93
mutuamente exclusivo, 167-169
ocorrência, 37
recursivo, 75, *236*
redundante, *87*
temporal, *90-93*
ternário, *43-45*, 155, *231*
Restrição de integridade, *74*, 123, *127-128, 244, 246, 248*
semântica, 128

Segunda forma normal, *197, 257-258*
SGBD *ver* Sistema de gerência de banco de dados
Sistema de gerência de banco de dados, *23*
pré-relacional, 184
relacional, 26, 120, 135
linguagem, 130, 141
Sistema legado, 184
SQL, 128, 130-131

Tabela, *120-122*
aninhada, *187*, 190
e engenharia reversa, 170-172
implementação inicial, 140-147
não-normalizada, *187-190*
não-primeira-forma-normal, 187-190

Tempo de armazenagem de dados, 93
Terceira forma normal, 200, *258*
Tupla *ver* Linha

Valor de atributo *ver* Campo
Vazio (Valor de campo), 126-127, 140