

# Projeto de Banco de Dados

Série  
Livros Didáticos

Número  
4

4ª edição



Instituto de Informática  
da UFRGS



Carlos Alberto Heuser



<http://groups-beta.google.com/group/digitalsource>

# Projeto de Banco de Dados

Carlos A. Heuser<sup>\*</sup>

---

<sup>\*</sup> © Carlos A. Heuser, 1998 - A publicação comercial deste texto está planejada. Ele deve ser considerado como comunicação pessoal do autor



# Sumário

<b>PREFÁCIO</b>	<b>V</b>
<b>1 INTRODUÇÃO</b>	<b>1</b>
1.1 Banco de Dados	2
1.1.1 Compartilhamento de dados	2
1.1.2 Sistema de Gerência de Banco de Dados	4
1.2 Modelos de Banco de Dados	5
1.2.1 Modelo conceitual	5
1.2.2 Modelo lógico	6
1.2.3 Modelo conceitual como modelo de organização	7
1.3 Projeto de BD	8
Exercícios	9
Referências Bibliográficas	9
<b>2 ABORDAGEM ENTIDADE-RELACIONAMENTO</b>	<b>11</b>
2.1 Entidade	12
2.2 Relacionamento	13
2.2.1 Conceituação	13
2.2.2 Cardinalidade de relacionamentos	15
2.2.3 Cardinalidade máxima	16
2.2.4 Classificação de relacionamentos binários	17
2.2.5 Relacionamento ternário	19
2.2.6 Cardinalidade mínima	20
2.2.7 Exemplo de uso de entidades e relacionamentos	21
2.3 Atributo	22
2.3.1 Identificando entidades	24
2.3.2 Identificando relacionamentos	27
2.4 Generalização/especialização	28
2.5 Entidade associativa	32
2.6 Esquemas gráficos e textuais de modelos ER	34
Exercícios	37
Referências Bibliográficas	40
<b>3 CONSTRUINDO MODELOS ER</b>	<b>43</b>
3.1 Propriedades de modelos ER	44
3.1.1 Um modelo ER é um modelo formal	44
3.1.2 Abordagem ER têm poder de expressão limitado	44
3.1.3 Diferentes modelos podem ser equivalentes	46
3.2 Identificando construções	48
3.2.1 Atributo versus entidade relacionada	48
3.2.2 Atributo versus generalização/especialização	49
3.2.3 Atributos opcionais e multi-valorados	50

<b>3.3</b>	<b>Verificação do modelo</b>	<b>53</b>
3.3.1	Modelo deve ser correto	53
3.3.2	Modelo deve ser completo	53
3.3.3	Modelo deve ser livre de redundâncias	54
3.3.4	Modelo deve refletir o aspecto temporal	55
3.3.5	Entidade isolada e entidade sem atributos	59
<b>3.4</b>	<b>Estabelecendo padrões</b>	<b>59</b>
3.4.1	Variantes de modelos ER	59
3.4.2	Uso de ferramentas de modelagem	62
<b>3.5</b>	<b>Estratégias de modelagem</b>	<b>63</b>
3.5.1	Partindo de descrições de dados existentes	64
3.5.2	Partindo do conhecimento de pessoas	64
	<b>Exercícios</b>	<b>66</b>
	<b>Referências Bibliográficas</b>	<b>74</b>
<b>4</b>	<b>ABORDAGEM RELACIONAL</b>	<b>75</b>
<b>4.1</b>	<b>Composição de um Banco de Dados Relacional</b>	<b>76</b>
4.1.1	Tabelas	76
4.1.2	Chaves	77
4.1.3	Domínios e valores vazios	80
4.1.4	Restrições de integridade	80
<b>4.2</b>	<b>Especificação de banco de dados relacional</b>	<b>81</b>
<b>4.3</b>	<b>Consultas à base de dados</b>	<b>82</b>
	<b>Exercícios</b>	<b>83</b>
	<b>Referências Bibliográficas</b>	<b>83</b>
<b>5</b>	<b>TRANSFORMAÇÕES ENTRE MODELOS</b>	<b>85</b>
<b>5.1</b>	<b>Visão geral do projeto lógico</b>	<b>86</b>
<b>5.2</b>	<b>Transformação ER para relacional</b>	<b>87</b>
5.2.1	Implementação inicial de entidades	89
5.2.2	Implementação de relacionamentos	91
5.2.3	Detalhes da implementação de relacionamentos	93
5.2.4	Implementação de generalização/especialização	100
5.2.5	Refinamento do modelo relacional	105
<b>5.3</b>	<b>Engenharia reversa de modelos relacionais</b>	<b>108</b>
5.3.1	Identificação da construção ER correspondente a cada tabela	110
5.3.2	Identificação de relacionamentos 1:n ou 1:1	111
5.3.3	Definição de atributos	113
5.3.4	Definição de identificadores de entidades	113

<b>Exercícios</b>	<b>114</b>
<b>Referências Bibliográficas</b>	<b>117</b>
<b>6      ENGENHARIA REVERSA DE ARQUIVOS E NORMALIZAÇÃO</b>	<b>119</b>
6.1 <b>Introdução</b>	<b>120</b>
6.2 <b>Visão geral do processo de engenharia reversa</b>	<b>120</b>
6.3 <b>Documento Exemplo</b>	<b>122</b>
6.4 <b>Representação na forma de tabela não normalizada</b>	<b>122</b>
6.5 <b>Normalização</b>	<b>125</b>
6.5.1      Passagem à primeira forma normal (1FN)	125
6.5.2      Dependência funcional	129
6.5.3      Passagem à segunda forma normal (2FN)	130
6.5.4      Passagem à terceira forma normal (3FN)	133
6.5.5      Passagem à quarta forma normal	136
6.5.6      Problemas da normalização	139
6.6 <b>Integração de modelos</b>	<b>141</b>
6.6.1      Integração de tabelas com mesma chave	141
6.6.2      Integração de tabelas com chaves contidas	143
6.6.3      Volta à 2FN	144
6.7 <b>Construção do modelo ER e Eliminação de Redundâncias</b>	<b>144</b>
6.8 <b>Verificação do modelo ER - Limitações da Normalização</b>	<b>144</b>
<b>Exercícios</b>	<b>145</b>
<b>Referências Bibliográficas</b>	<b>156</b>
<b>7      SOLUÇÕES DE EXERCÍCIOS SELECIONADOS</b>	<b>157</b>
<b>ÍNDICE REMISSIVO</b>	<b>192</b>



# Prefácio

## Objetivos do livro

Sistemas de gerência de banco de dados (SGBD) surgiram no início da década de 70 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 de pesquisa na área de banco de dados. Esse investimento resultou em um tipo de SGBD, o *SGBD relacional*. A partir da década de 80 e devido ao barateamento das plataformas de hardware/software para executar SGBD relacional, este tipo de SGBD passou a dominar o mercado, tendo se convertido em padrão internacional. O desenvolvimento de sistemas de informação ocorre hoje 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, é executado hoje com auxílio de técnicas padronizadas e suportadas por ferramentas CASE. Formou-se ao longo do tempo um conjunto de conhecimentos sobre projeto de BD que é largamente 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 usualmente ocorre em três etapas. A primeira etapa, a *modelagem conceitual*, procura capturar formalmente os requisitos de informação de um banco de dados. A segunda etapa, o *projeto lógico*, objetiva definir, a nível de SGBD, as estruturas de dados que implementarão os requisitos identificados na modelagem conceitual. A terceira etapa, o *projeto físico*, define parâmetros físicos de acesso ao BD, procurando otimizar a performance do sistema como um todo.

Este livro objetiva ensinar 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 notações da abordagem ER, o livro apresenta regras e heurísticas para 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



## **Público alvo**

O livro objetiva atender a três públicos distintos.

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 Projeto de Banco de Dados (correspondente a matéria obrigatória T3 do Currículo de Referência 96 da SBC). O livro se origina 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 livro necessita-se 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 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 desejem sistematizar o projeto de seus bancos de dados. Para estes leitores, a parte referente a 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 está dedicado a apresentar a abordagem entidade-relacionamento. O objetivo do capítulo é ensinar os conceitos básicos do modelo ER e a notação gráfica para 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 ER em direção a modelos semânticos, como o conceito de generalização/especialização e entidade associativa.

Enquanto o Capítulo 2 objetiva a *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 a usar na construção de modelos. Além disso são discutidas notações alternativas a de Peter Chen.

O Capítulo 4 é uma introdução ao modelo relacional. Não se trata aqui de mostrar de forma completa o funcionamento de 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*, ou seja a transformação de um modelo lógico de banco de dados relacional em modelo conceitual.

Finalmente, o Capítulo 6 cobre a *engenharia reversa de arquivos*, ou seja apresenta um conjunto de regras para transformar a descrição de um conjunto de arquivos convencionais em um modelo de dados relacional. O conjunto de regras está baseado na normalização de banco de dados relacional, que é apresentada no capítulo. Por tratar-se de um texto introdutório, são apresentadas apenas as formas normais da primeira a quarta.



## Introdução

Este capítulo apresenta os conceitos básicos da área de banco de dados que são necessário à compreensão do projeto de banco de dados. São apresentados conceitos como banco de dados, sistema de gerência de banco de dados e modelo de dados. Além disso é fornecida uma visão geral do projeto de banco de dados

O leitores que já conhece os fundamentos de banco de dados provavelmente poderá passar diretamente 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. Consideramos que nesta indústria 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 propriamente dita, 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 mencionado acima, os dados de um produto são usados em várias funções. Estes dados são necessários no planejamento de 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 no setor de compras, pois este necessita saber que componentes devem ser adquiridos. Já o setor de vendas também necessita conhecer dados de produtos, como por exemplo 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).

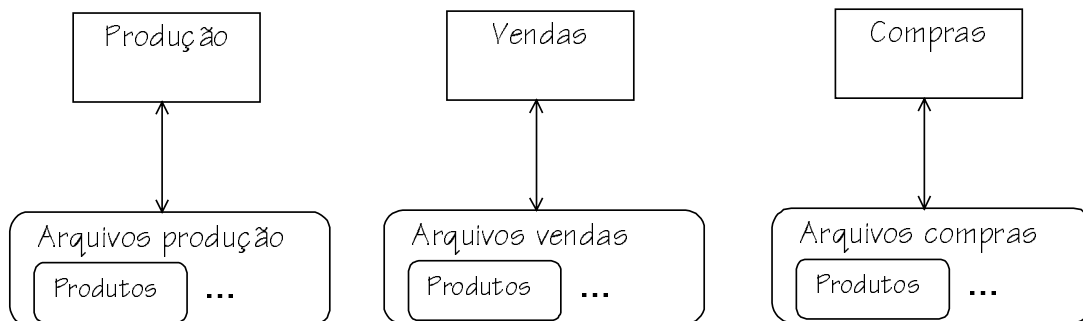


Figura 1.1: Sistemas isolados

Neste caso, surge o problema da *redundância de dados*. Redundância de dados ocorre quando uma determinada informação está representada no sistema em computador várias vezes. No caso do exemplo, estão redundantes as informações referentes a um produto, que aparecem nos arquivos de produtos de cada um dos três sistemas.

Há duas formas 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 performance 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:

❑ **Redigitação**

A mesma informação é digitada várias vezes. No caso do exemplo da indústria, os dados de um produto são digitados no setor de vendas, no setor de produção e no setor de compras. Além de exigir trabalho desnecessário, a redigitação 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 demais 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)*.

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

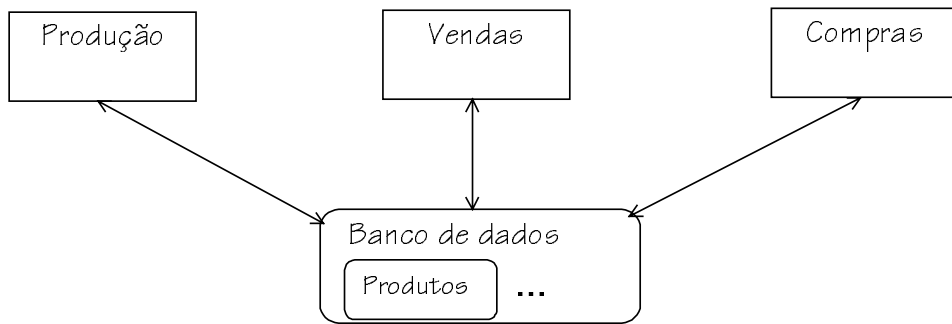


Figura 1.2: Sistemas integrados com dados compartilhados

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*, conforme descrito na próxima seção.

### 1.1.2 Sistema de Gerência de Banco de Dados

A programação de aplicações em computadores sofreu profundas modificações desde seus primórdios. No início, quando usavam-se 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 em dia, a grande maioria dos programas comunica-se com os usuários através de interfaces gráficas de janelas. Entretanto, normalmente, os programas não contêm todo código referente a 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, 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* (SGBD).

sistema de gerência de banco de dados
=
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 torna programas mais facilmente compreensíveis. A *produtividade* de progra-

madores 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 da atualidade. Entretanto, muitas das idéias apresentadas nas seções referentes à modelagem de dados aplicam-se também a outros tipos, como os SGBD 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 citado acima, 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*. Como veremos adiante, um mesmo modelo de dados pode ser apresentado de várias formas. Cada apresentação do modelo recebe a denominação *esquema de banco de dados*.

De acordo com a intenção do modelador, um banco de dados pode ser modelado (descrito) há vários níveis de abstração. Um modelo de dados que servirá para explicar a um usuário 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*.

### 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.



<p>modelo conceitual</p> <p>=</p> <p>modelo de dados abstrato, que descreve a estrutura de um banco de dados de forma independente de um SGBD particular</p>
--

A técnica mais difundida de modelagem conceitual 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.

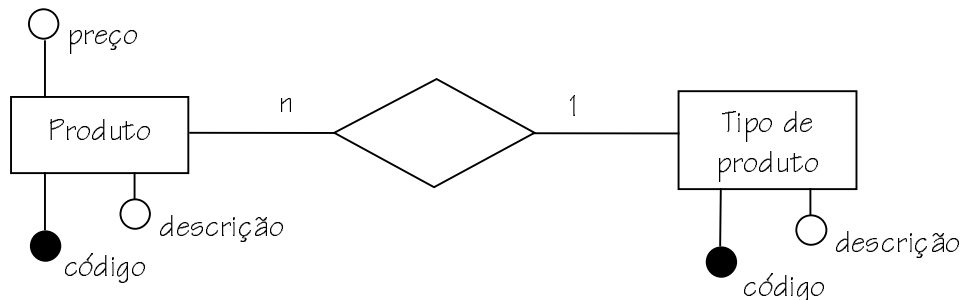


Figura 1.3: Exemplo de modelo conceitual

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.

### 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 particular de SGBD que está sendo usado.

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 para o BD acima deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas.

Abaixo é apresentado o modelo lógico (de forma textual) da Figura 1.4:

TipoDeProduto(CodTipoProd, DescrTipoProd)

Produto(CodProd, DescrProd, PreçoProd, CodTipoProd)

CodTipoProd referencia TipoDeProduto

TipoDeProduto	
CodTipoProd	DescrTipoProd
1	Computador
2	Impressora

Produto			
CodProd	DescrProd	PrecoProd	CodTipoProd
1	PC desktop modelo X	2.500	1
2	PC notebook ABC	3.500	1
3	Impressora jato de tinta	600	2
4	Impressora laser	800	2

Figura 1.4: Exemplo de tabelas de BD relacional

O modelo lógico descreve a estrutura do banco de dados, conforme vista pelo usuário do SGBD. Detalhes de armazenamento interno de informações, que não tem influência sobre a programação de aplicações no SGBD, mas podem influenciar a performance da aplicações (por exemplo, as estruturas de arquivos usadas no acesso as informações) não fazem parte do modelo lógico. Estas são representadas 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 a performance. As linguagens e notações para o modelo físico não são padronizadas e variam de produto a produto. A tendência em produtos mais modernos é esconder o modelo físico do usuário e transferir a tarefa de otimização ao próprio SGBD.

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

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 ar-

quívios em computador e entidades da organização modelada, observou-se que um mesmo modelo conceitual pode ser usado em duas funções:

- ❑ 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 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. Na outra interpretação, como modelo abstrato de um banco de dados, o diagrama nos informa que o banco de dados deverá conter informações sobre produtos e tipos de produtos, que para cada tipo de produto são armazenados seus 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. Sabe-se da prática da engenharia de software 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 portanto são mais simples de compreender por usuários leigos em Informática, que modelos que envolvem detalhes de implementação.

### 1.3 PROJETO DE BD

O projeto de um novo BD dá-se em duas fases:

#### 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 de forma independente 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.

O processo acima 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* de banco de dados. A engenharia reversa é explicada nos capítulos 5 e 6.

## EXERCÍCIOS

**Exercício 1.1:** Enumere as principais diferenças entre o processamento de dados com arquivos convencionais e o processamento de dados com SGBD.

**Exercício 1.2:** 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 1.3:** Defina, sem retornar ao capítulo acima, os seguintes conceitos: banco de dados, sistema de gerência de banco de dados, modelo de dados, modelo conceitual, modelo lógico, modelagem conceitual e projeto lógico. Verifique a definição que você fez contra a apresentada no capítulo.

**Exercício 1.4:** A definição do fator de bloco de um arquivo faz parte de que modelo: do modelo conceitual, do modelo lógico ou do modelo físico?

**Exercício 1.5:** A definição do tipo de um dado (numérico, alfanumérico,...) faz parte de que modelo: do modelo conceitual, do modelo lógico ou do modelo físico?

**Exercício 1.6:** Qual a diferença entre a redundância de dados controlada e a redundância de dados não controlada? Dê exemplos de cada uma delas.

## REFERÊNCIAS BIBLIOGRÁFICAS

Os conceitos apresentados neste capítulo aparecem em qualquer bom livro de fundamentos de banco de dados. Dois exemplos são os livros de Korth e Silberschatz [2] e de Elmasri e Navathe [1], este último sem tradução para Português.

- [1] Elmasri, R. & Navathe, S.B. *Fundamentals of Database Systems*. Second Edition. Benjamin/Cummings, Redwood City, California, 1994
- [2] Korth, H. & Silberschatz, A. *Sistemas de Bancos de Dados*. 2ª edição, Makron Books, 1994



## Abordagem Entidade Relacionamento

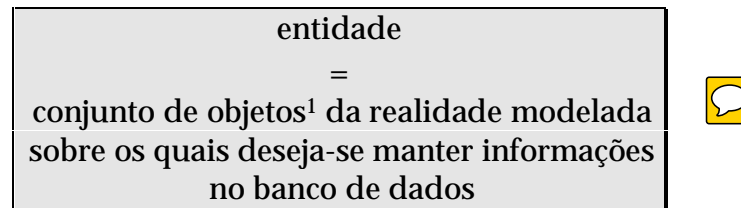
Como vimos no Capítulo 1, a primeira etapa do projeto de um banco de dados é a construção de um modelo conceitual, a chamada *modelagem conceitual*. O objetivo da modelagem conceitual é obter uma descrição abstrata, independente de implementação em computador, dos dados que serão armazenados no banco de dados.

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

Este capítulo tem por objetivo apresentar os conceitos centrais da abordagem ER: *entidade*, *relacionamento*, *atributo*, *generalização/especialização* e *entidade associativa*. Junto com os 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 outras notações para representar diagramas ER.

## 2.1 ENTIDADE

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



Uma entidade representa, no modelo conceitual, 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 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<sup>2</sup>).

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.



Figura 2.1: Representação gráfica de entidades

Como dito acima, cada retângulo representa um *conjunto* de objetos sobre os quais deseja-se guardar informações. Assim, no exemplo, 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 (alguns autores usam 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. Essa terminologia é a primeira vista 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

---

<sup>1</sup> O termo “objeto” possui aqui a conotação que é lhe 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.

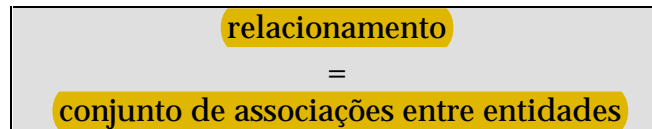
<sup>2</sup> Neste ponto, aquele que já possui conhecimento sobre a modelagem ER poderá estar pensando: “Obviamente endereço é um atributo e não uma entidade!”. Se você é um desses leitores, peço um pouco de paciência, pois esta questão será esclarecida mais adiante.

projeto de BD, no qual falamos com grande frequência sobre conjuntos de objetos e raramente sobre indivíduos. Por esse motivo preferimos usar o par de termos *entidade* e *ocorrência de entidade*.

## 2.2 RELACIONAMENTO

### 2.2.1 Conceituação

Além de especificar os objetos sobre os quais deseja-se manter informações, o DER deve permitir a especificação das propriedades dos objetos que serão armazenadas no BD. 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.



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.



Figura 2.2: Representação gráfica de relacionamento

Este modelo expressa que o BD mantém informações sobre:

- ❑ um conjunto de objetos classificados como pessoas (relacionamento PESSOA)
- ❑ um conjunto de objetos classificados como departamentos (relacionamento DEPARTAMENTO)
- ❑ um conjuntos de associações, que ligam um departamento a uma pessoa. (relacionamento LOTAÇÃO).

Da mesma forma que fizemos com entidades, quando quisermos nos referir a associações particulares dentro de um conjunto, vamos nos referir a *ocorrências 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 entidades participantes de uma ocorrência de relacionamento são indicadas pelas linhas que ligam o círculo negro repre-



sentativo da ocorrência de relacionamento aos círculos brancos representativos das ocorrências de entidades relacionadas. Assim, a Figura 2.3 representa que 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 abaixo). O modelo apresentado permite que uma ocorrência de entidade (por exemplo, a pessoa **p3**) não esteja associada a nenhuma ocorrência de entidade através do relacionamento, ou 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.

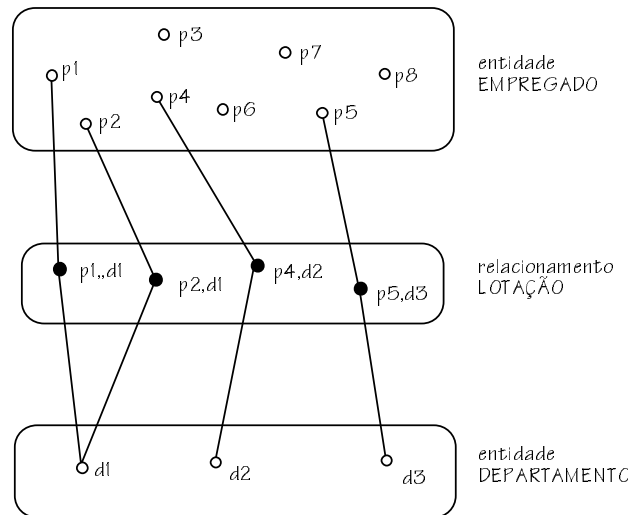


Figura 2.3: Diagrama de ocorrências

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. 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.

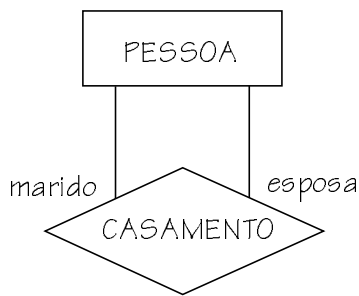


Figura 2.4: Auto-relacionamento

A Figura 2.5 apresenta um possível diagrama de ocorrências para o DER da Figura 2.4. Os papéis (marido e esposa) das ocorrências de entidades em cada ocorrência de relacionamento foram anotadas nas linhas que ligam os círculos representativos das ocorrências de entidades e relacionamentos.

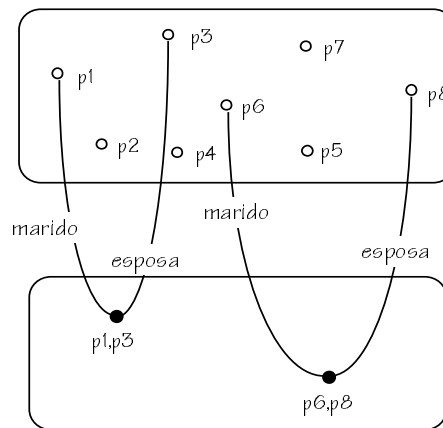


Figura 2.5: Diagrama de ocorrências para o relacionamento casamento

## 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

### 2.2.3 Cardinalidade máxima

Para exemplificar o conceito de cardinalidade vamos retomar o exemplo da Figura 2.2. Vamos considerar as seguintes cardinalidade máximas:

- ❑ 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 fins práticos, não é necessário distinguir entre diferentes cardinalidades máximas maiores que 1. Por este motivo, apenas duas cardinalidades máximas são relevantes: a cardinalidade máxima 1 e a cardinalidade máxima “muitos”, referida pela letra *n*. Assim, no exemplo acima, diz-se que a cardinalidade máxima da entidade DEPARTAMENTO no relacionamento LOTAÇÃO é *n*.

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

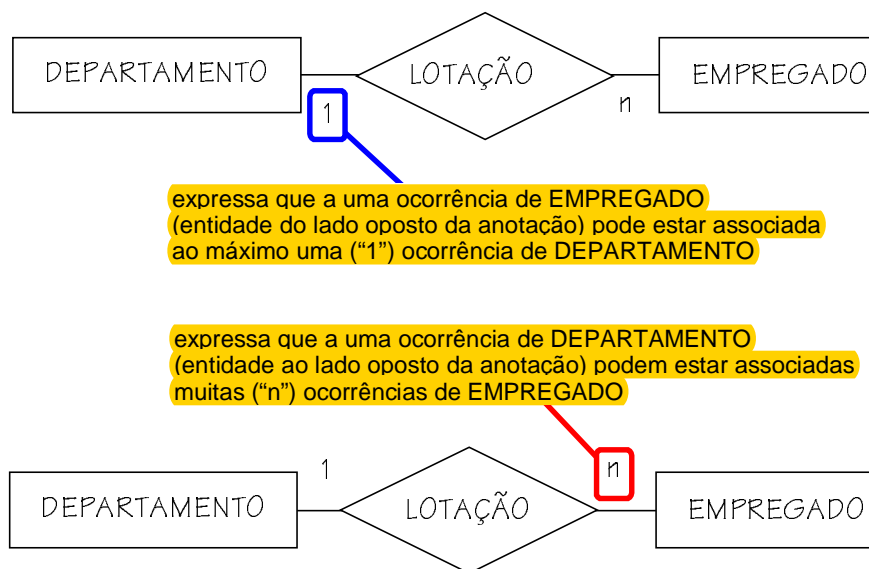


Figura 2.6: Cardinalidade máxima de relacionamento

## 2.2.4 Classificação de relacionamentos binários

A cardinalidade máxima pode ser usada para classificar relacionamentos binários. Um *relacionamento binário* é aquele cujas ocorrências envolvem duas entidades, como todos vistos até aqui. Podemos classificar os relacionamentos em **n:n** (muitos-para-muitos), **1:n** (um-para-muitos) e **1:1** (um-para-um). A Figura 2.7, a Figura 2.8 e a Figura 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 (uma instância de pessoa pode estar associada via relacionamento a no máximo outra pessoa no papel de esposa) e no máximo uma esposa.

Observe que este relacionamento, apesar de envolver apenas uma entidade, é também considerado como um relacionamento binário. 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). Por este motivo, o relacionamento de **CASAMENTO** é classificado como sendo binário.

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 que um aluno pode estar inscrito em no máximo um curso.

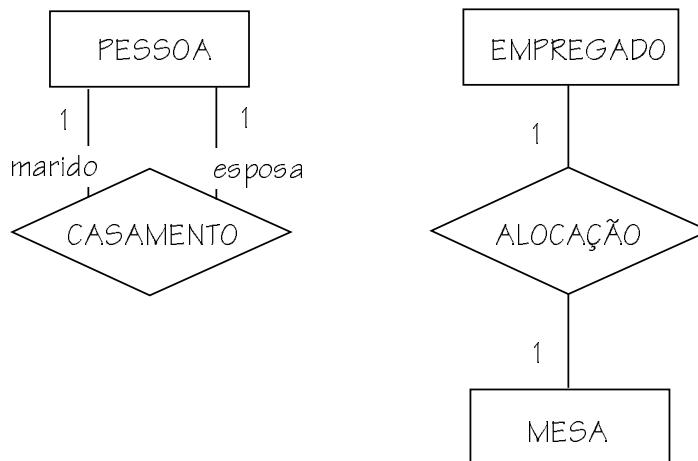


Figura 2.7: Relacionamentos 1:1

O relacionamento **INSCRIÇÃO** da Figura 2.8 representa a associação entre cursos de uma Universidade pública e seus alunos. Por tratar-se de uma universidade pública, cada aluno pode estar vinculado a um curso no máximo.

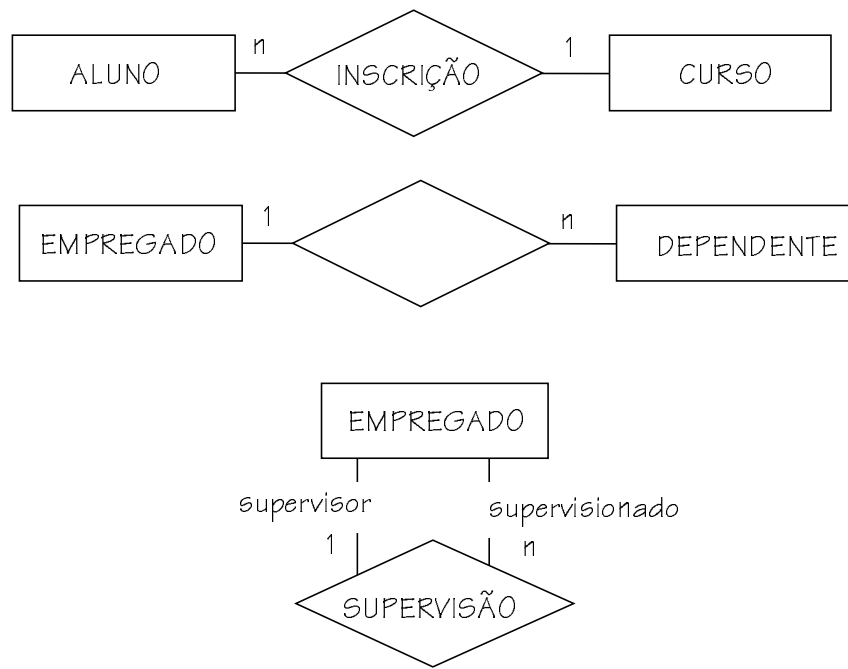


Figura 2.8: Relacionamentos 1:n

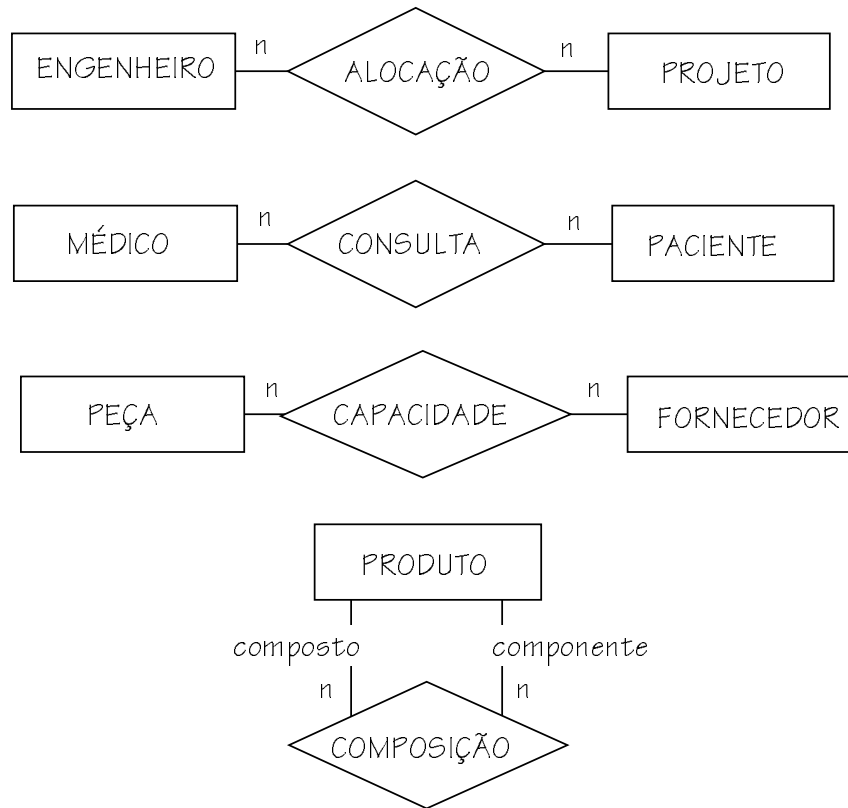


Figura 2.9: Relacionamentos n:n

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 relaci-**

onamento, 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 o de um auto-relacionamento.

### 2.2.5 Relacionamento ternário

Todos exemplos até aqui mostrados são de relacionamentos binários, ou seja, de relacionamentos que associam exatamente duas entidades. 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 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.

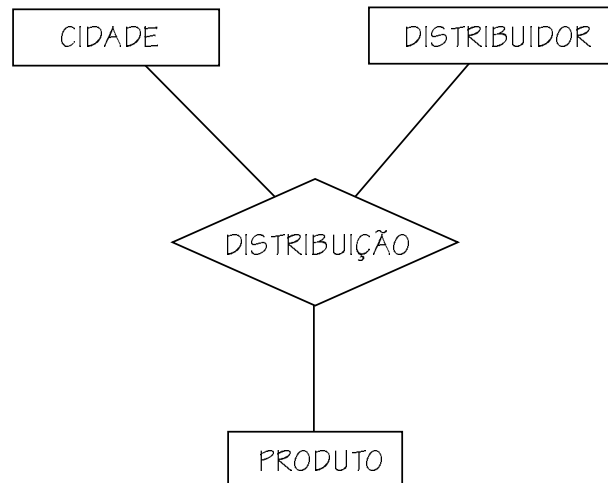


Figura 2.10: Relacionamento ternário

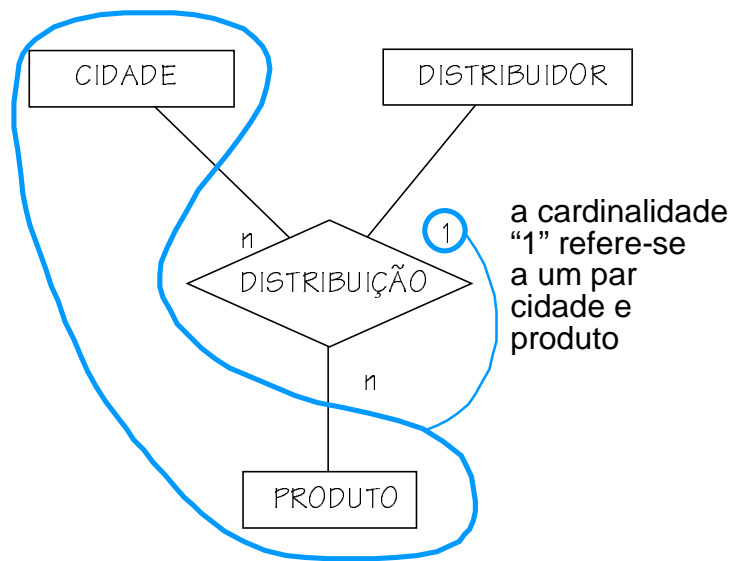


Figura 2.11: Cardinalidade em relacionamentos ternários

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. Em outros termos, não há concorrência pela distribuição de um produto em uma cidade.

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.

### 2.2.6 Cardinalidade mínima

Além da cardinalidade máxima, uma outra informação que pode ser representada por um modelo ER é o número mínimo de ocorrências de entidade que são 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 também recebe a denominação de “associação *opcional*”.

A cardinalidade mínima é anotada no diagrama junto a cardinalidade máxima, conforme mostrado na Figura 2.12. Nesta figura, aparece novamente o exemplo da alocação de empregados a mesas que já foi apresentado anteriormente. 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).

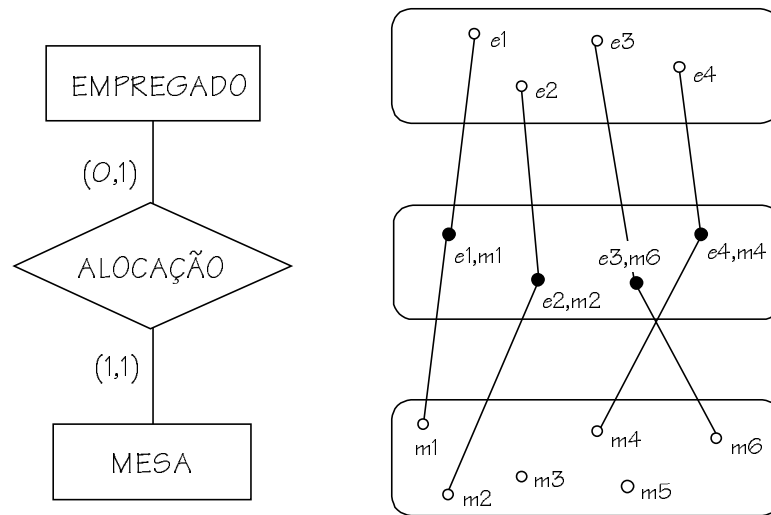


Figura 2.12: Cardinalidade mínima de relacionamento

### 2.2.7 Exemplo de uso de entidades e relacionamentos

A Figura 2.13 apresenta um exemplo de um diagrama entidade-relacionamento 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 usual evitar-se 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.

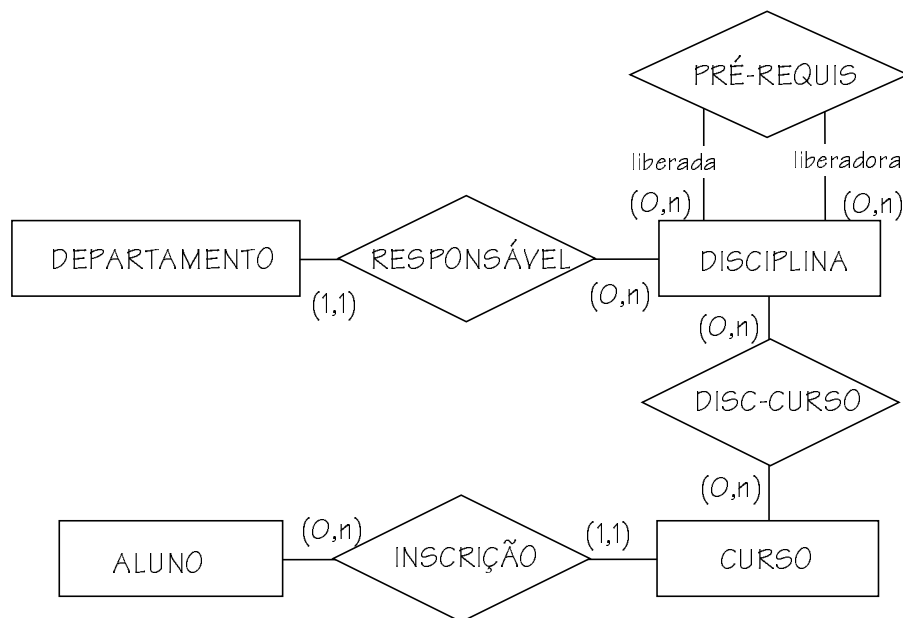


Figura 2.13: DER para o controle acadêmico de uma universidade



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-requisitos.
- ❑ 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 isso 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 antes da eliminação de um departamento. Da forma como a restrição foi especificada, é possível incluir o departamento em uma transação, para, depois, em transações subsequentes, 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 as 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).

### **2.3 ATRIBUTO**

Para associar informações a ocorrências de entidades ou de relacionamentos usa-se o conceito de *atributo*.

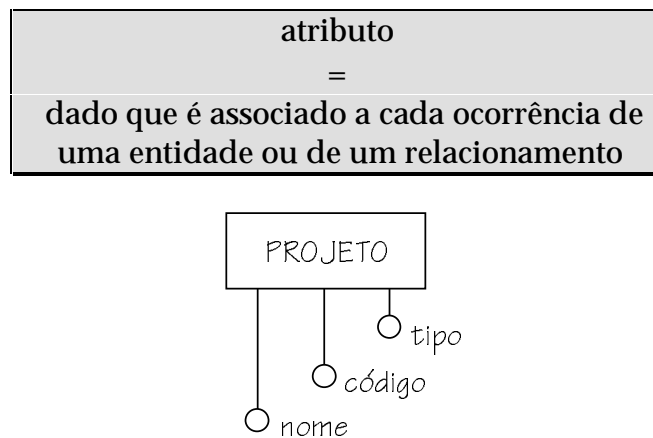


Figura 2.14: Atributos de uma entidade

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

Na prática, atributos não são representados graficamente, para não sobrecarregar os diagramas, já que muitas vezes entidades possuem 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 de ser usado um software para construção de modelos ER, o próprio software encarrega-se do armazenamento da lista de atributos de cada entidade em um *dicionário de dados*

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 mono-valorados (cardinalidade máxima “1” – cada entidade possui no máximo um valor associado). Já o atributo telefone, é um atributo opcional (cardinalidade mínima 0) e multi-valorado (cardinalidade máxima n).

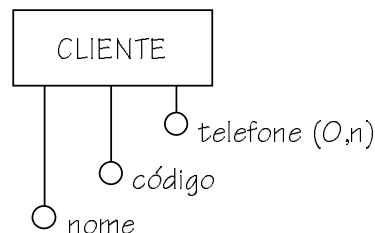


Figura 2.15: Cardinalidade de Atributos

Assim como entidades possuem atributos, 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. Esta 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.

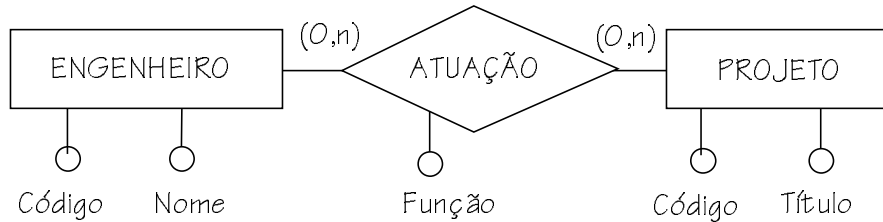


Figura 2.16: Atributo de relacionamento n:n

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 à prazo. Vendas à prazo são relacionadas a uma financeira, através do relacionamento **FINANCIAMENTO**. Os atributos **nº de parcelas** e **taxa de juros** são atributos do relacionamento.

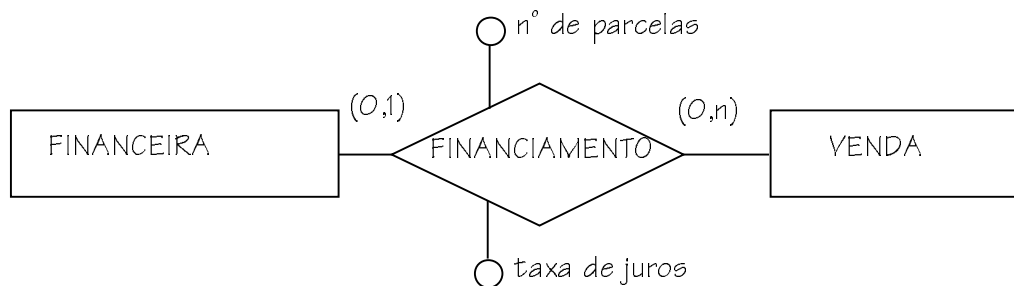


Figura 2.17: Atributo de relacionamento 1:n

Estes dois atributos poderiam ter sido incluídos na entidade **VENDA**. Neste caso, seriam atributos opcionais, já que nem toda venda é à 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 à prazo.

### 2.3.1 Identificando entidades

Cada entidade deve possuir um *identificador*. Um identificador é um conjunto de um ou mais atributos (e possivelmente relacionamentos, como visto abaixo) cujos valores servem para distinguir uma ocorrência da entidade das demais ocorrências da mesma entidade.

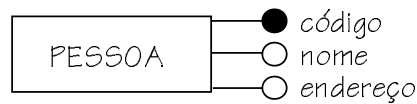


Figura 2.18: Identificador simples

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

A Figura 2.19 mostra um exemplo no qual o identificador da entidade é composto por diversos 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 sequencialmente a partir de um e as prateleiras são numeradas sequencialmente 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.

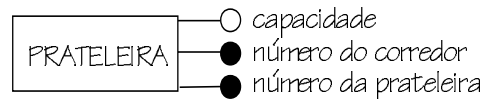


Figura 2.19: Identificador composto

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 diagrama apresenta empregados de 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 sequência 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.

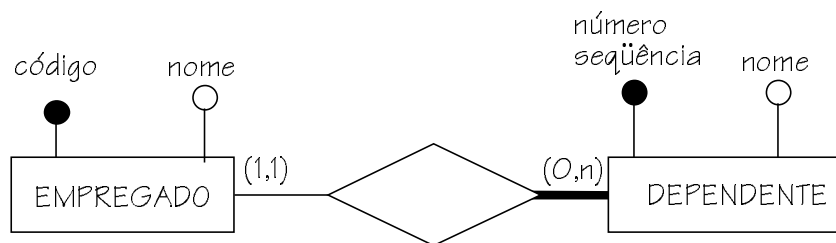


Figura 2.20: Relacionamento identificador

Nesse caso, alguns autores dizem que a entidade **DEPENDENTE** é uma entidade *fraca*. O termo “fraca” deriva-se 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 as entidades chamadas “fracas” por este critério podem, dependendo da realidade modelada, ser centrais a um modelo. Considere o fragmento de um DER sobre empresas, mostrado na Figura 2.21. 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. Pela definição acima, tanto **EMPRESA** quanto **FILIAL** são entidades fracas. Entretanto, ao considerarmos que a maior parte das entidades que eventualmente comporiam o restante do modelo estariam ligadas a **EMPRESA** ou **FILIAL** vemos que o conceito de “fraqueza” introduzido acima não se aplica ao caso em questão.

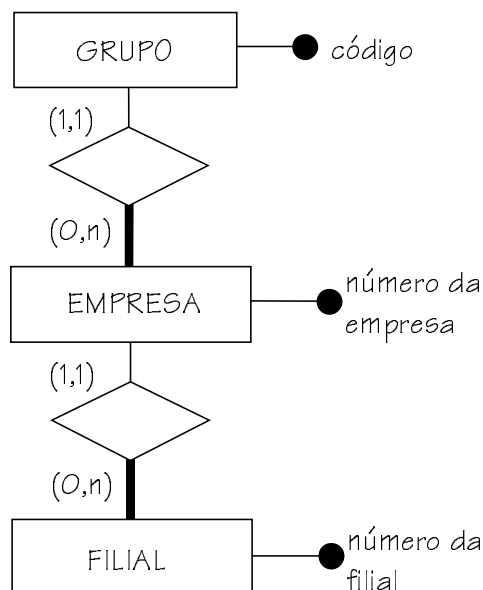


Figura 2.21: Entidades com relacionamentos identificadores

identificador de entidade
=
conjunto de atributos e relacionamentos cujos valores distinguem uma ocorrência da entidade das demais

O identificador de uma entidade, seja ele simples, composto por diversos atributos, ou composto por identificadores externos, deve obedecer 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. 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.

- 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 CIC (identificador único do contribuinte junto a 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.

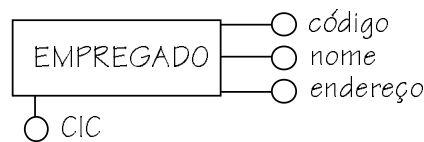


Figura 2.22: Identificadores alternativos

### 2.3.2 Identificando relacionamentos

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



Figura 2.23: Relacionamento identificado por suas entidades

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 de MÉDICO e de PACIENTE (Figura 2.24). Entre um determinado médico e um determinado paciente podem 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 seu paciente. A diferenciação dá-se através de *atributos identificadores de relacionamentos*. No caso da Figura 2.24, o atributo identificador do relacionamento é data/hora.

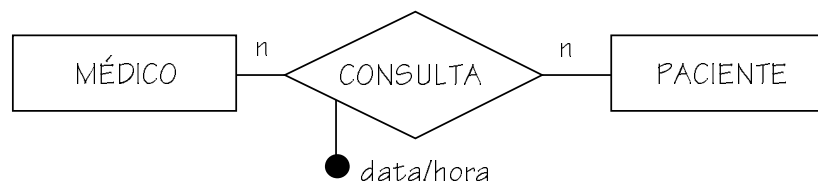


Figura 2.24: Identificador de relacionamento

Assim, de forma geral, um relacionamento é identificado pelas entidades dele participantes, bem como pelos atributos identificadores eventualmente existentes.

## 2.4 GENERALIZAÇÃO/ESPECIALIZAÇÃO

Além de relacionamentos e atributos, propriedades podem ser atribuídas a entidades através do conceito de *generalização/especialização*. Através deste conceito é possível atribuir propriedades particulares a um subconjunto das ocorrências (*especializadas*) de uma entidade *genérica*. O símbolo 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 um 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, **CIC** 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**, **CIC** 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**, **CGC** e **tipo de organização**, é identificada pelo **código** e está obrigatoriamente relacionada a exatamente uma filial.

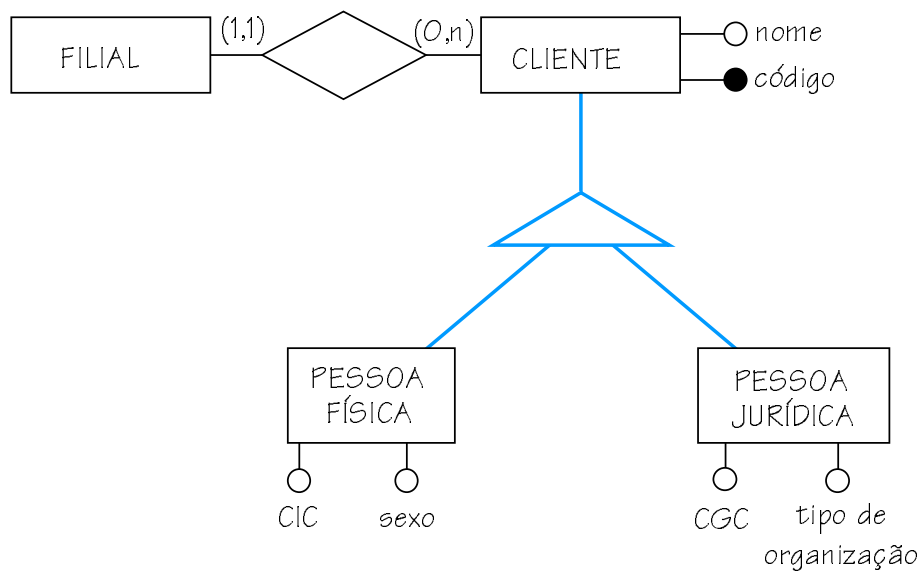


Figura 2.25: Generalização/especialização

A generalização/especialização pode ser classificada em dois tipos, *total* ou *parcial*, de acordo com a obrigatoriedade ou não de a uma 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.

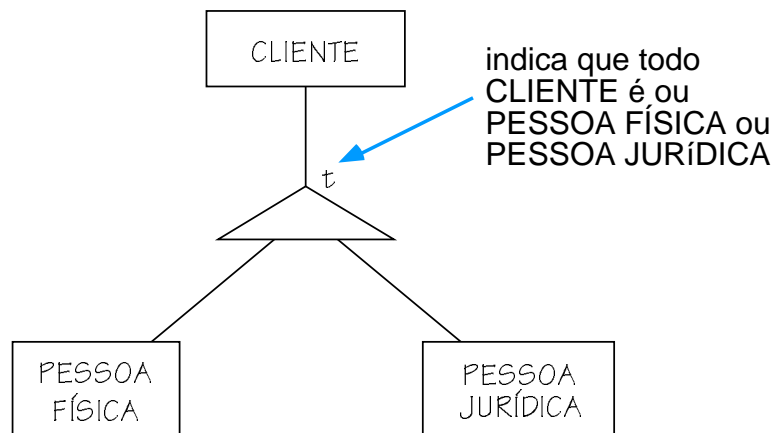


Figura 2.26: Generalização/especialização total

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). Esse tipo de generalização/especialização é simbolizado por um “p” conforme mostrado na figura. Usualmente, quando há uma especialização parcial, na



entidade genérica (no caso do exemplo, em FUNCIONÁRIO) aparece um atributo que identifica o tipo de ocorrência da entidade genérica (no caso do 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 a entidade genérica em uma de suas especializações é suficiente para identificar o tipo da entidade.

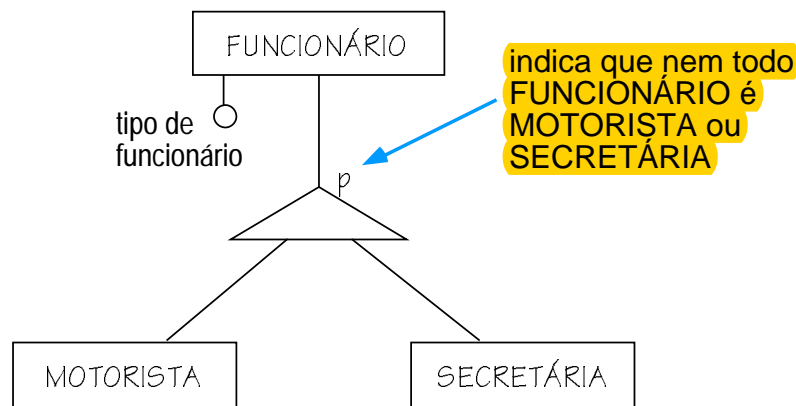


Figura 2.27: Generalização/especialização parcial

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 possuísssem 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 especialização de diversas entidades genéricas (a chamada *herança múltipla*). A Figura 2.28 apresenta um DER em que aparecem múltiplos níveis de generalização/especialização, bem como o conceito de herança múltipla. Exemplificando, uma entidade BARCO é uma especialização de um VEÍCULO AQUÁTICO, que por sua vez é uma especialização de VEÍCULO. Assim um barco tem, além de suas propriedades específicas, 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.

A generalização/especialização como tratada neste livro é *exclusiva*. Generalização/especialização exclusiva significa que uma ocorrência de entidade genérica aparece, para cada hierarquia generalização/especialização, no máximo uma vez, nas folhas da árvore de generalização/especialização. Exemplificando, na Figura 2.28, um veículo é automóvel ou veículo anfíbio ou barco e somente um destes.

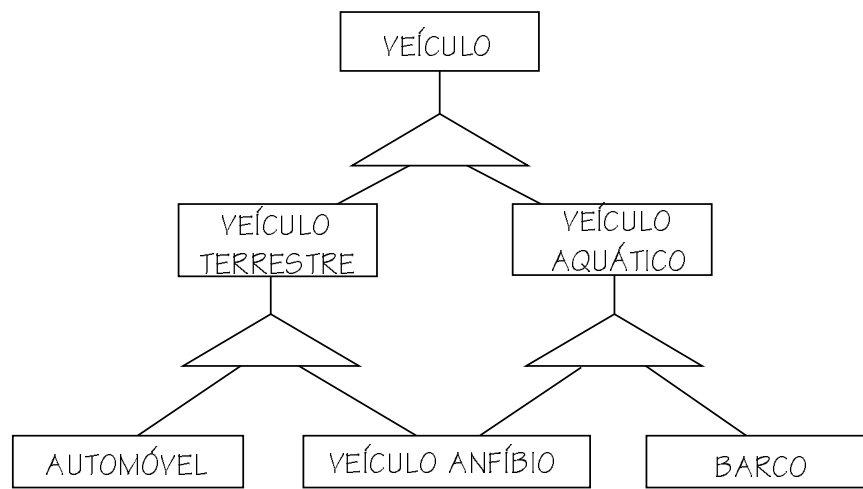


Figura 2.28: Generalização/especialização em múltiplos níveis e com herança múltipla

Há autores que permitem também especializações não exclusivas. Um exemplo é mostrado na Figura 2.29. Neste diagrama, considera-se o conjunto de pessoas vinculadas a uma universidade. Neste caso, a especialização não é exclusiva, já que a mesma pessoa pode aparecer em múltiplas especializações. Uma pessoa pode ser professor de um curso e ser aluno em outro curso, ou ser aluno de pós-graduação. Por outro lado, uma pessoa pode acumular o cargo de professor em tempo parcial com o cargo de funcionário, ou, até mesmo, ser professor de tempo parcial em dois departamentos diferentes, sendo portanto duas vezes professor. O principal problema que este tipo de generalização/especialização apresenta é que neste caso as entidades especializadas não podem herdar o identificador da entidade genérica. No caso, o identificador de pessoa não seria suficiente para identificar professores, já que uma pessoa pode ser múltiplas vezes professor.

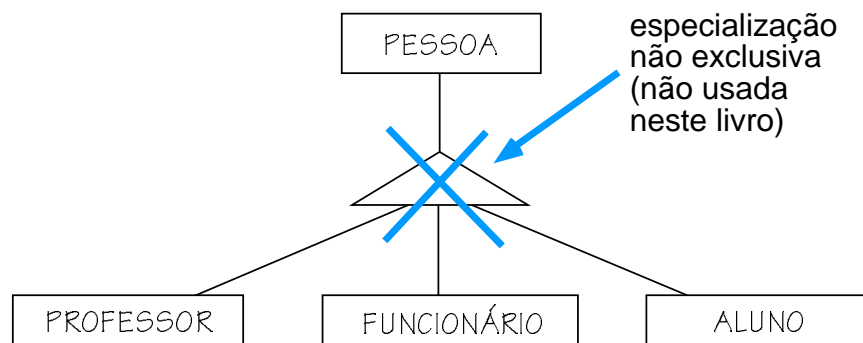


Figura 2.29: Generalização/especialização não exclusiva

Em casos como o mostrado, usa-se o conceito de relacionamento, ao invés do conceito de generalização/especialização. O modelo deveria conter três relacionamentos, associando a entidade **PESSOA** com as entidades correspondentes a cada um dos papéis de **PESSOA** (**PROFESSOR**, **FUNCIONÁRIO** e **ALUNO**).

## 2.5 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 está-se construindo um novo DER ou modificando um DER existente, surgem situações em que é desejável permitir a associação de uma entidade a um relacionamento. A título de exemplo, considere-se o diagrama da Figura 2.30.



Figura 2.30: DER a ser modificado

Suponha que seja necessário modificar este diagrama com a adição da informação de que, em cada consulta, um ou mais medicamentos podem ser prescritos ao paciente. Para tal, criar-se-ia 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 a **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 2.31. O retângulo desenhado ao redor do relacionamento **CONSULTA** indica que este relacionamento passa a ser visto como uma entidade (associativa, já que baseada em um relacionamento). Sendo **CONSULTA** também uma entidade, é possível associá-la através de relacionamentos a outras entidades, conforme mostra a figura.

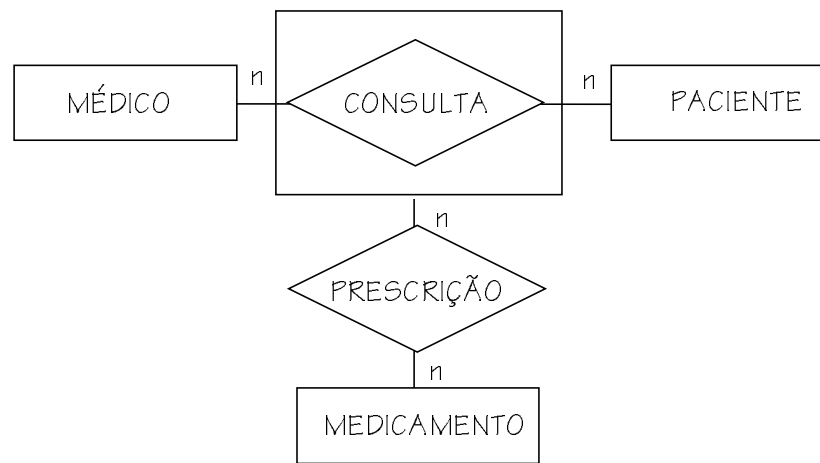


Figura 2.31: Entidade associativa

Observe-se que, 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 2.32.

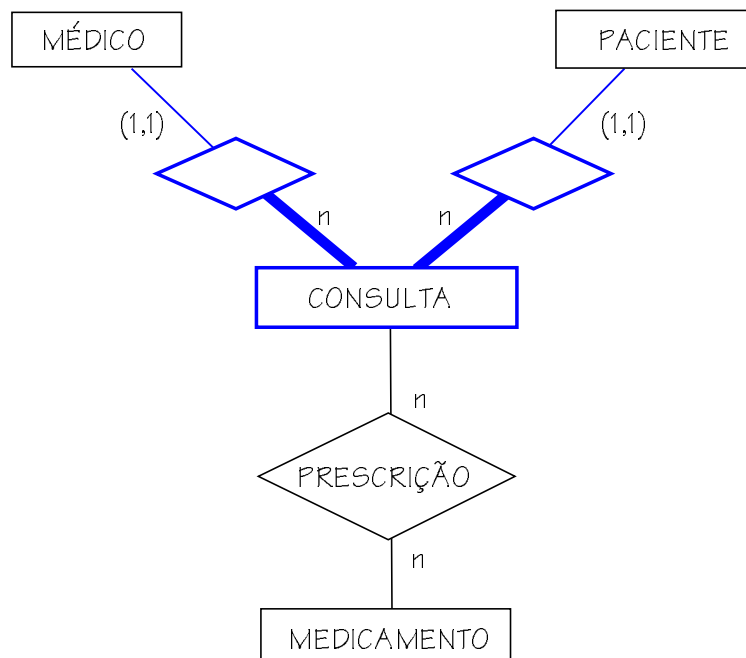


Figura 2.32: Substituindo relacionamento por entidade

Na figura, o relacionamento foi substituído por uma entidade homônima, junto com dois relacionamentos (parte representada em linhas densas). Observe-se que, para manter equivalência com o diagrama anterior, uma consulta está relacionada com exatamente um médico e exatamente um paciente (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**. Observe-se que o diagrama da Figura 2.32 é equivalente ao diagrama da Figura 2.31. Equivalente aqui significa que ambos geram

o mesmo banco de dados relacional. A transformação de modelos ER em descrições de BD relacionais, necessária à compreensão do conceito de equivalência, é apresentada em um dos próximos capítulos.

## **2.6 ESQUEMAS GRÁFICOS E TEXTUAIS DE MODELOS ER**

A descrição de um modelo é chamada, na terminologia de banco de dados, de *o esquema do banco de dados*.

Até este ponto os esquemas ER sempre foram diagramas ER, isto é sempre estão apresentados na forma gráfica. A Figura 2.33 resume os símbolos usados neste livro para a representação gráfica de esquemas ER.

Um esquema ER pode ser um texto. Na Figura 2.34 aparece a sintaxe de uma linguagem textual para definição de esquemas ER. A sintaxe é dada na forma de uma gramática BNF<sup>3</sup>. Nesta sintaxe, são usadas 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.

---

<sup>3</sup> Se você não sabe o que é uma gramática BNF consulte um livro sobre linguagens de programação ou sobre construção de compiladores.


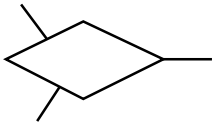


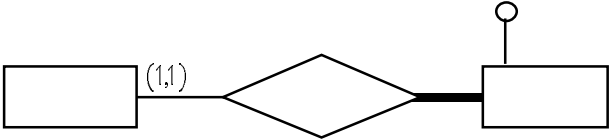
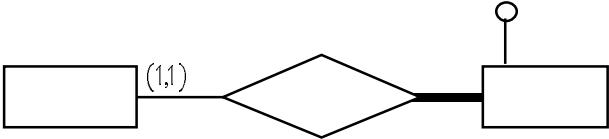
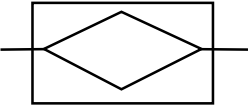
Conceito	Símbolo
Entidade	
Relacionamento	
Atributo	
Atributo identificador	
Relacionamento identificador	
Generalização/especialização	
Entidade associativa	

Figura 2.33: Símbolos usados na construção de esquemas ER

A Figura 2.35 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, em nosso caso, é mais rica, pois inclui a possibilidade de definir um *tipo de atributo* (declaração [DECL\\_TIPO](#)).

Na prática é usual combinar as duas formas de representar esquemas ER: a diagramática e a textual. Escolhe-se a forma de representar de acordo com a sua adequação. 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 atributos identificadores é feita de forma textual, pois sobrecarregaria o diagrama.

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 2.34: Gramática BNF de uma linguagem para definição de esquema ER textual

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\_SEQUENCIA

Relacionamento: EMP\_DEP

Entidades: (1,1) EMPREGADO  
(0,n) DEPENDENTE

Figura 2.35: Esquema ER textual correspondente à Figura 2.20

## EXERCÍCIOS

**Exercício 2.1:** Dê ao menos cinco exemplos dos conceitos básicos da abordagem ER apresentados neste capítulo: entidade, relacionamento, atributo, generalização/especialização.

**Exercício 2.2:** Explique a diferença entre uma entidade e uma ocorrência de entidade. Exemplifique.

**Exercício 2.3:** O que é o papel de uma entidade em um relacionamento. Quando é necessário especificar o papel das entidades de um relacionamento?

**Exercício 2.4:** Considere o relacionamento **CASAMENTO** que aparece no DER da Figura 2.7. Segundo este DER o banco de dados poderia conter um casamento 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 com esposa? Caso uma destas situações possa ocorrer, como deveria ser modificado o DER para impedi-las?

**Exercício 2.5:** Confeccione um possível diagrama de ocorrências para o relacionamento **SUPERVISÃO** (Figura 2.8) e suas respectivas entidades.

**Exercício 2.6:** Confeccione um possível diagrama de ocorrências para o relacionamento **COMPOSIÇÃO** (Figura 2.9) e suas respectivas entidades.

**Exercício 2.7:** Mostre como o modelo ER da Figura 2.11 pode ser representado sem uso de relacionamentos ternários, apenas usando relacionamentos binários.

**Exercício 2.8:** Dê um exemplo de um relacionamento ternário. Mostre como a mesma realidade pode ser modelada somente com relacionamentos binários.

**Exercício 2.9:** Para o exemplo de relacionamento ternário da questão anterior, justifique a escolha das cardinalidades mínima e máxima.

**Exercício 2.10:** 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 2.11:** Construa um DER que modela a mesma realidade que a mostrada no DER da Figura 2.16, usando apenas relacionamentos 1:n.



**Exercício 2.12:** 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 2.13:** Construa um DER em que o conceito de entidade associativa é usado.

**Exercício 2.14:** Dê ao menos três exemplos de entidades com relacionamentos identificadores (entidades fracas).

**Exercício 2.15:** 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 nenhum curso, deve permanecer por algum tempo no banco de dados.

**Exercício 2.16:** Sem usar atributos opcionais, nem atributos multi-valorados, construa um DER que contenha as mesmas informações do DER da Figura 2.15

**Exercício 2.17:** O DER da Figura 2.29 modela uma generalização/especialização não exclusiva. Como dito no texto do capítulo que descreve este DER, generalizações/especializações deste tipo não são usadas neste livro. Construa um DER que modela a realidade descrita sem usar o conceitos de generalização/especialização não exclusiva.

**Exercício 2.18:** A Figura 2.36 apresenta um modelo de dados para uma farmácia. Descreva em português tudo o que está representado neste diagrama.

**Exercício 2.19:** Invente nomes para os relacionamentos da Figura 2.36.

**Exercício 2.20:** Dê uma justificativa para as cardinalidades mínimas do relacionamento entre **FORNECEDOR** e **FABRICANTE** no DER da Figura 2.36.

**Exercício 2.21:** Explique o significado das cardinalidades mínima e máxima do relacionamento ternário (entre **MEDICAMENTO**, **VENDA** e **RECEITA MÉDICA**) no DER da Figura 2.36.

**Exercício 2.22:** 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 2.36. Explique porque.

**Exercício 2.23:** Para cada entidade e cada relacionamento no DER da Figura 2.36 defina, quando possível, atributos. Para cada entidade, indique o(s) atributo(s) identificador(es).

**Exercício 2.24:** Escreva um esquema ER textual para o esquema diagramático da Figura 2.36.

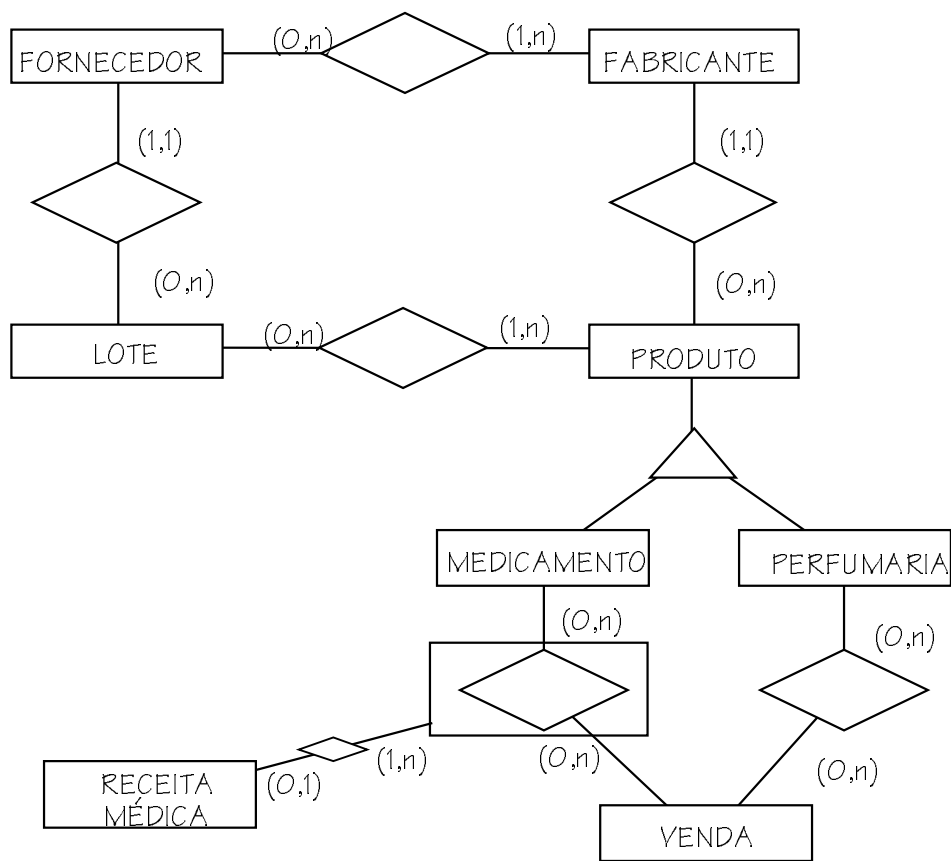


Figura 2.36: Diagrama ER de uma farmácia

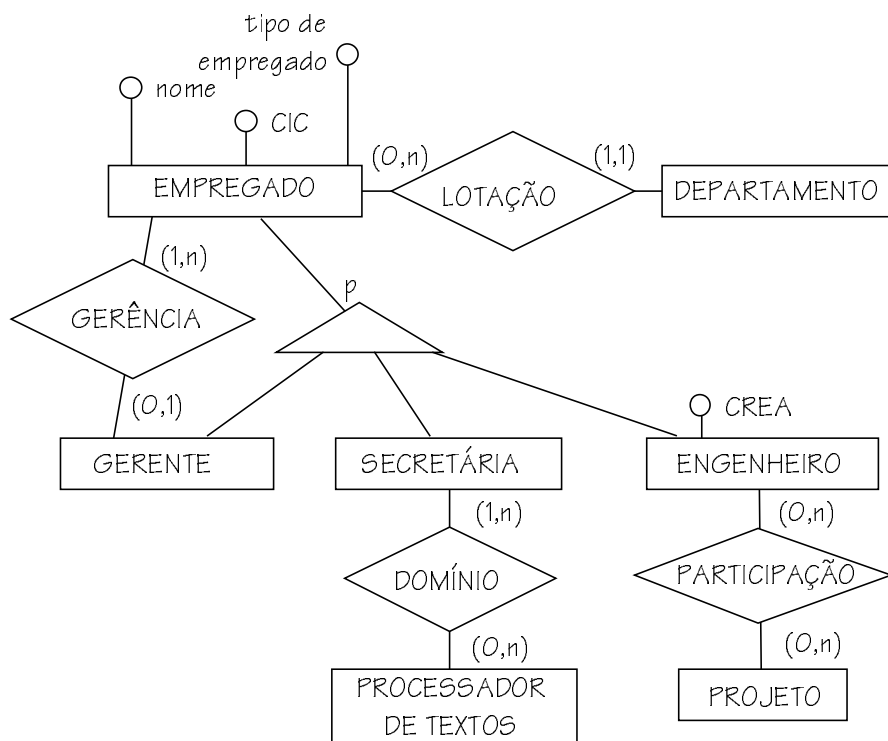


Figura 2.37: Diagrama ER para sistema de recursos humanos

**Exercício 2.25:** A Figura 2.37 apresenta um DER de parte de um sistema de recursos humanos em uma organização. Descreva em português tudo que está representado neste diagrama.

**Exercício 2.26:** Para cada entidade e cada relacionamento do DER da Figura 2.37 defina, quando possível, atributos. Para cada entidade, indique o(s) atributo(s) identificador(es).

**Exercício 2.27:** Escreva um esquema ER textual para o esquema diagramático da Figura 2.37

**Exercício 2.28:** De acordo com o DER da Figura 2.37, que ações devem ser tomadas ao excluir-se do banco de dados uma secretária?

**Exercício 2.29:** De acordo com o DER da Figura 2.37, uma secretária ou um engenheiro não podem ser gerentes. Porque? Como o DER deveria ser modificado para permitir que tanto uma secretária, quanto um engenheiro pudessem ser também gerentes?

## REFERÊNCIAS BIBLIOGRÁFICAS

O artigo original sobre a abordagem ER é o trabalho de Chen [3]. 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 [7,8].

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 o de Abrial [1]. 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 [4,5] abordam em detalhe 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 de Smith e Smith [9] apresenta os conceitos sobre os quais a maioria das abordagens de modelagem de dados, inclusive a abordagem ER, baseiam-se.

Além da abordagem ER, diversas 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 [11] é provavelmente 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 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 bastante completo e detalhado é o de Batini, Ceri e Navathe [2]. Outros livros conhecidos são o de Teorey [10] e o de Maninila e Rähä [6], este último mais dedicado a aspectos teóricos.

- [1] Abrial, J.R. Data Semantics, in Klimbie, J.W.. and Koffeman, K.L. (editors) *Data Base Management*, North-Holland, 1974, 1–59.

- [2] Batini, C.; Ceri, S.; Navathe, S.B. *Conceptual Database Design - an Entity Relationship Approach*. , Benjamin/Cummings, Redwood City, CA, 1992
- [3] Chen, P. The Entity Relationship Model—Toward a Unified View of Data, *ACM Transactions on Database Systems*, 1:1, March 1976, pp 9–37.
- [4] Kent, W. *Data and Reality*, North-Holland, 1978.
- [5] Kent, W. Limitations of Record-Based Information Models”, *ACM Transactions on Database Systems*, 4:1, March 1979, pp 107–131.
- [6] Maninila, H.; Rähkä, K.-J. *The Design of Relational Databases*. Addison-Wesley, Wokingham, England, 1992.
- [7] dos Santos, C.S., Neuhold, E., & Furtado, A. A Data Type Approach to the Entity Relationship Model, in Chen, P. (editor) *Entity-Relationship Approach to Systems Analysis and Design*, (Proceedings of the First International Conference on Entity-Relationship Approach, Los Angeles, California, December 1979), North-Holland, 1980, pp 103–120.
- [8] Scheuermann, P., Schiffner, G., & Weber, H. Abstraction Capabilities and Invariant Properties Modeling within the Entity-Relationship Approach, in Chen, P. (editor) *Entity-Relationship Approach to Systems Analysis and Design*, (Proceedings of the First International Conference on Entity-Relationship Approach, Los Angeles, California, December 1979), North-Holland, 1980, pp 121-140.
- [9] Smith, J. & Smith, D. Database Abstractions: Aggregation and Generalization, *ACM Transactions on Database Systems*, 2:2, June 1977, pp 105–133.
- [10] Teorey, T.J. *Database Modeling & Design - The Fundamental Principles*. Second Edition. Morgan Kaufmann, San Francisco, CA, 1994
- [11] Verheijen, G.M.A. and VanBekkum, J. NIAM: An Information Analysis Method, in Olle, T., Sol, H., and Verrijn-Stuart, A. (editors) *Information System Design Methodologies: a Comparative Review*, North-Holland, 1982, pp. 537–590.



## Construindo modelos ER

No capítulo anterior, vimos como um diagrama ER é composto. Neste capítulo, vamos nos concentrar na construção de modelos ER, isto é, no problema de como, dada uma determinada realidade, obter o modelo ER.

O capítulo inicia apresentando uma coletânea de conselhos práticos e heurísticas a usar durante a modelagem conceitual. A seguir são apresentadas notações alternativas a de Peter Chen para confecção de diagramas ER. O capítulo finaliza apresentando um processo de modelagem e discutindo alternativas a este processo.

### 3.1 PROPRIEDADES DE MODELOS ER

Nesta seção, discutimos algumas propriedades de modelos ER que são importantes ter em mente quando da modelagem ER.

#### 3.1.1 Um modelo ER é um modelo formal

Um DER é um modelo formal, preciso, não ambíguo. Isto significa que diferentes leitores de um mesmo DER devem sempre entender exatamente o mesmo. Tanto é assim, que um DER pode ser usado como entrada a uma ferramenta CASE<sup>4</sup> na geração de um banco de dados relacional. Por isso, é de fundamental importância que todos os envolvidos na confecção e uso de diagramas ER estejam treinados na sua perfeita compreensão.

Observa-se em certas organizações, que modelos ER são sub-utilizados, servindo apenas como ferramenta para apresentação informal de idéias. Isso pode ser evitado com treinamento formal de todos envolvidos na modelagem e projeto do banco de dados.

Note-se que é importante que efetivamente *todos* os que manipulam modelos ER estejam treinados em sua compreensão. O fato de um DER ser gráfico e intuitivo pode transmitir a falsa impressão de ser compreensível até por alguém não treinado.

Para exemplificar o tipo de problemas que surgem ao usar diagramas ER sem treinar as pessoas envolvidas, descrevo uma situação que já observei em algumas organizações. Os técnicos em computação da organização desenvolveram um DER 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 DER 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 BD implementado não era exatamente aquele desejado pelos usuários, apesar de corresponder exatamente ao DER apresentado. Assim, para que modelos ER possam atingir seus objetivos, é necessário treinamento formal. Não estou sugerindo aqui 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 Abordagem ER têm poder de expressão limitado

Em um modelo ER, são apresentadas apenas algumas propriedades de um banco de dados. Em realidade, a linguagem dos DER é uma linguagem muito pouco poderosa e muitas propriedades desejáveis do banco de dados necessitam ser anotadas adicionalmente ao DER. Abaixo mostramos dois exemplos

---

<sup>4</sup>Ferramenta CASE: software que dá suporte a construção de software (CASE vem do inglês “computer aided software engineering” que significa “engenharia de software suportada por computador”)

de DER, já vistos no capítulo anterior, para salientar o quão incompletos são estes modelos.

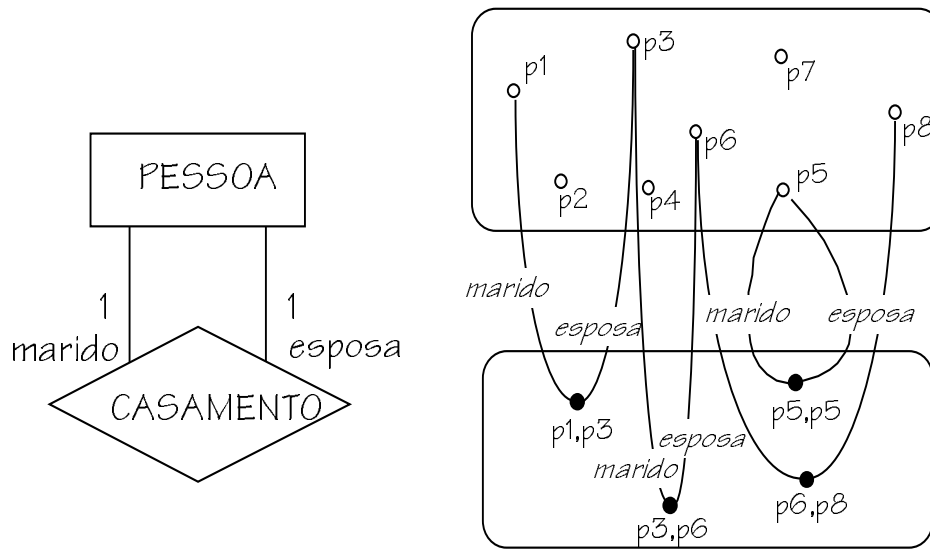


Figura 3.1: DER e diagrama de ocorrências para CASAMENTO

A Figura 3.1 mostra o DER que modela pessoas e casamentos já mostrado no capítulo anterior. Ao lado, são mostrados 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). A pessoa **p3** aparece em dois casamentos, uma vez no papel de esposa e outra vez no papel de marido. 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). Assim, para fazer com que o DER corresponda a realidade que deseja-se modelar, é necessário modificá-lo ou então definir restrições adicionais. No caso em questão, é possível modificar o DER para excluir os casamentos indesejáveis (ver exercício no capítulo anterior).

Aqui cabe a pergunta: até onde deve ser modificado um DER para introduzir restrições de integridade? A resposta não é trivial. Aqui entra o bom senso do modelador. É necessário lembrar o objetivo que se tem ao construir um diagrama ER: o de projetar um banco de dados. Neste contexto, o DER 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 diagrama não é o de especificar todas restrições de integridade. Assim, somente são incluídas construções em um DER, quando estas possuem uma correspondência no banco de dados a ser implementada. 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.

Entretanto, há situações em que, mesmo através de modificações no DER, é impossível incluir no diagrama as restrições desejadas.



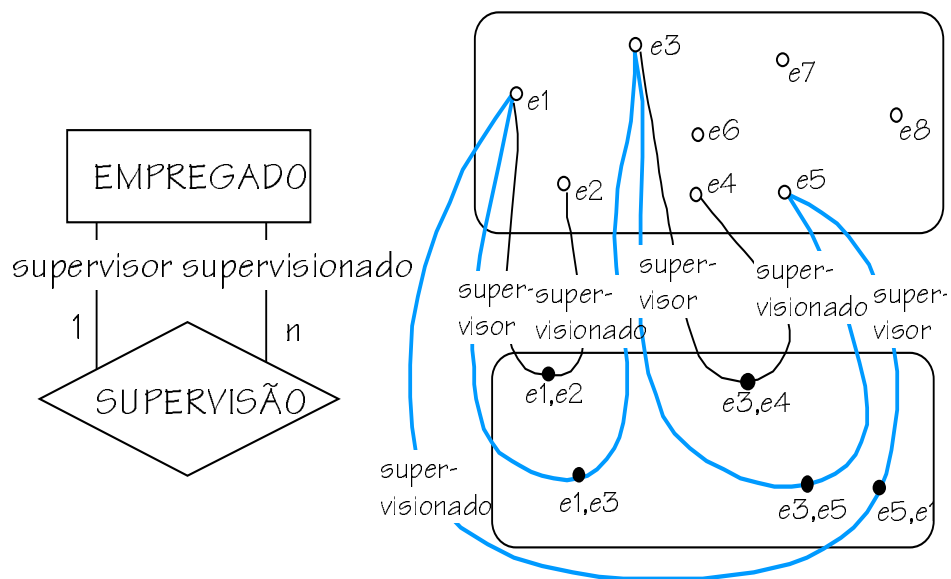


Figura 3.2: Relacionamento recursivo

A Figura 3.2 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**. Isso obviamente contraria nosso conhecimento sobre a realidade modelada, já que, em uma hierarquia de supervisão, não é permitido que um superior hierárquico (no caso **e1**) apareça como supervisionado em um nível mais baixo da hierarquia (no caso, **e1** aparece como supervisionado de **e5**). Essa restrição, ao contrário do exemplo do casamento apresentado acima, não é possível de se introduzir no DER através de modificações. O problema é que a restrição é uma restrição de integridade *recursiva* e estas não podem ser representadas através de diagramas ER. Neste caso, resta apenas especificar a restrição a parte do DER.

### 3.1.3 Diferentes modelos podem ser equivalentes

Na prática, muitas vezes observa-se analistas em acirradas discussões a fim de decidir como um determinado objeto da realidade modelada deve aparecer no modelo. Às vezes, tais discussões são absolutamente supérfluas, pois os diferentes modelos ER, em qualquer das opções defendidas pelos diferentes analistas, 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.

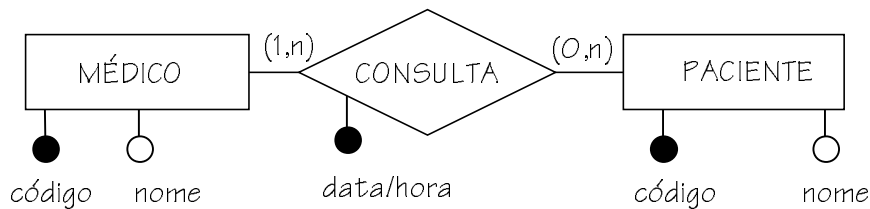
Para definir o conceito de equivalência de forma mais precisa, é necessário considerar o BD que é projetado a partir do modelo ER. Para fins de projeto de BD, dois modelos ER são equivalentes, quando ambos geram o mesmo esquema de BD. Assim, para analisar se dois modelos são equivalentes, é ne-

cessário considerar um conjunto de regras de tradução de modelos ER para modelos lógicos de BD. Para os fins deste 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 BD relacional através do uso das regras de tradução apresentadas no Capítulo 5. Quando falamos “o mesmo modelo de BD relacional” estamos falando de bancos de dados que, abstraindo de diferenças de nomes de estruturas (tabelas, atributos, ...) tenham a mesma estrutura.

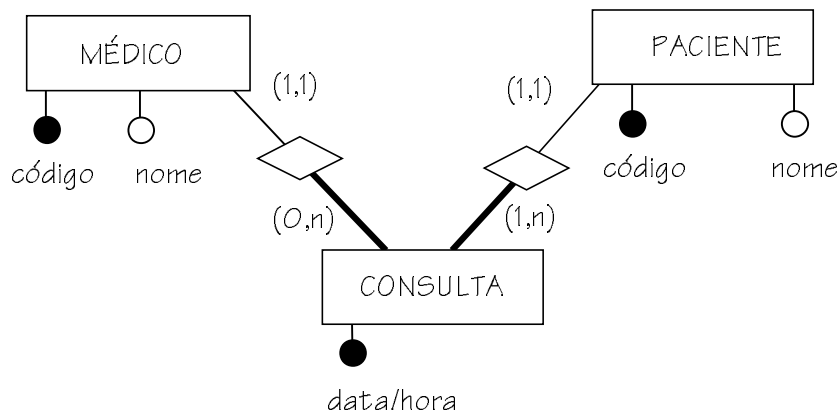
É claro que para entender perfeitamente este conceito de equivalência de modelos, o leitor deve conhecer as regras de tradução apresentadas no Capítulo 5. Mesmo assim, considerando a definição informal de equivalência (dois modelos que expressam o mesmo), é possível compreender alguns casos da aplicação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

Um caso 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. Um exemplo são 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 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:
  - as entidades que originalmente participavam do relacionamento
  - os atributos que eram identificadores do relacionamento original (caso o relacionamento original tivesse atributos identificadores)
4. As cardinalidades da entidade criada nos relacionamentos de que participa é sempre (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 um relacionamento  $n:n$  pode ser transformado em entidade, é possível construir modelos sem relacionamentos  $n:n$ . Neste fato, baseiam-se diversas variantes da abordagem ER, que excluem relacionamentos  $n:n$ , ou excluem apenas relacionamentos  $n:n$  com atributos. Um exemplo são várias abordagens baseadas na Engenharia de Informações (ver Seção 3.4.1.1).

Um outro caso de equivalência, é entre um relacionamento de cardinalidade  $1:1$  e com cardinalidade mínima “1” em ambos os lados pode ser substituído por uma única entidade.

## 3.2 IDENTIFICANDO CONSTRUÇÕES

Apenas observando um determinado objeto da realidade modelada, é difícil determinar se tal objeto deve ser modelado como uma entidade, um atributo ou um relacionamento. Para definir que construção de ER será usada na modelagem do objeto, é necessário conhecer o seu contexto, isto é, o modelo dentro do qual ele aparece. Assim, a recomendação geral que se dá para a identificação de objetos é a de considerar a decisão por um ou outro tipo de representação no modelo (entidade, relacionamento, atributo) como sujeita a alteração durante a modelagem. Não é aconselhável despender um tempo excessivo em longas discussões sobre como modelar um objeto. O próprio desenvolvimento do modelo e o aprendizado sobre a realidade irão refinando e aperfeiçoando o modelo. Mesmo assim, existem alguns indicativos para a escolha de construções de modelagem. Estes indicativos passam a ser descritos a seguir.

### 3.2.1 Atributo versus entidade relacionada

Uma questão que às vezes surge na modelagem de um sistema é entre modelar um objeto como sendo um atributo de uma entidade ou 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 seria modelar a cor

como uma entidade autônoma, que está relacionada à entidade **AUTOMÓVEL** (segunda opção da Figura 3.4).

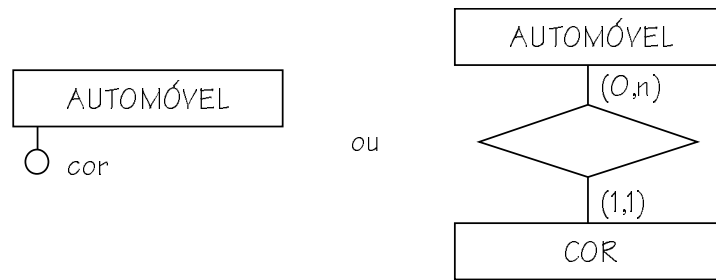


Figura 3.4: Escolhendo entre atributo e entidade relacionada

Alguns critérios para esta decisão são:

- ❑ Caso o objeto cuja modelagem está em discussão esteja vinculado a outros objetos (atributos, relacionamentos, entidades genéricas ou especializadas), o objeto deve ser 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 caso do exemplo das cores dos automóveis, poder-se-ia optar por modelar a cor como uma entidade, caso se tivesse que registrar no banco de dados os possíveis fabricantes da tinta da referida cor (entidades relacionadas a cor), ou caso se quisesse registrar as datas de início e fim (atributos) do uso de uma determinada cor. Caso não houvesse nenhum objeto relacionado à cor do automóvel, poder-se-ia 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 a entidade automóvel.

### 3.2.2 Atributo versus generalização/especialização

Um 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 através de 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 caso do exemplo acima, faz sentido especializar a entidade empregado de acordo com a categoria funcional, no caso de as classes particulares possuírem atributos ou relacionamentos próprios (um exemplo é o caso da Figura 3.5).

Ainda no exemplo dos empregados, o sexo do empregado é melhor modelado como atributo de empregado, caso não existam propriedades particulares de homens e mulheres a modelar na realidade considerada.

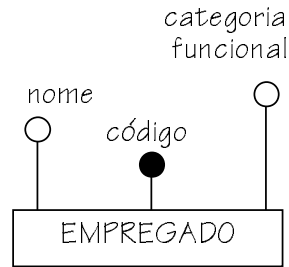


Figura 3.5: Modelando categoria funcional como atributo

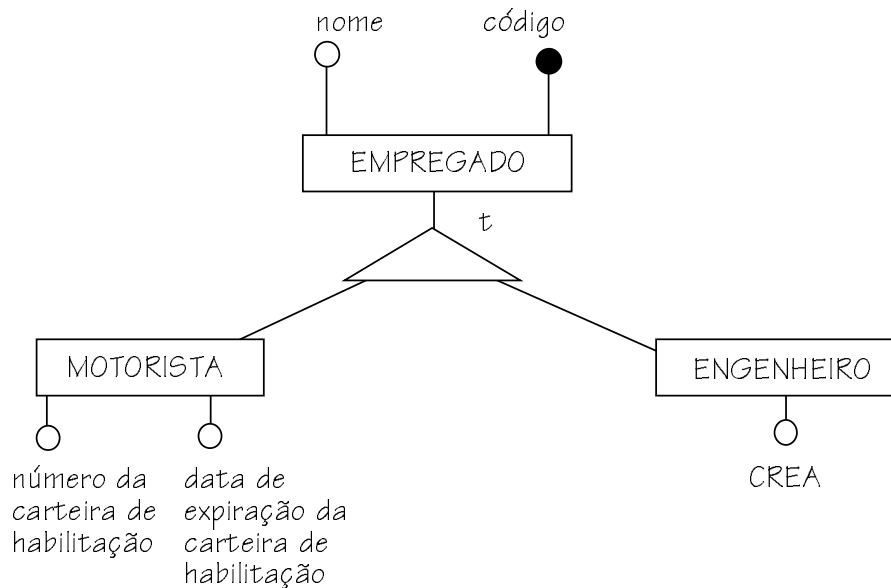


Figura 3.6: Modelando categoria funcional como especialização

### 3.2.3 Atributos opcionais e multi-valorados

Conforme vimos no capítulo anterior (Seção 2.3), atributos podem ser *opcionais* (nem toda ocorrência da entidade possui um valor do atributo) ou *multi-valorados*. Entretanto, quando se inicia o processo de modelagem é aconselhável tentar restringir-se ao uso de atributos *obrigatórios* e *mono-valorados* pelas razões abaixo listadas.

#### 3.2.3.1 Atributo opcional

Em muitos casos, atributos opcionais indicam subconjuntos de entidades que são modelados mais corretamente através de especializações. O DER da Figura 3.7 apresenta uma entidade com diversos atributos opcionais. Segundo este DER, nem todo empregado possui um registro no CREA (Conselho Regional de Engenharia e Arquitetura), nem todo empregado possui 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, data de expiração da carteira de habilitação mas não possuir o número da carteira?

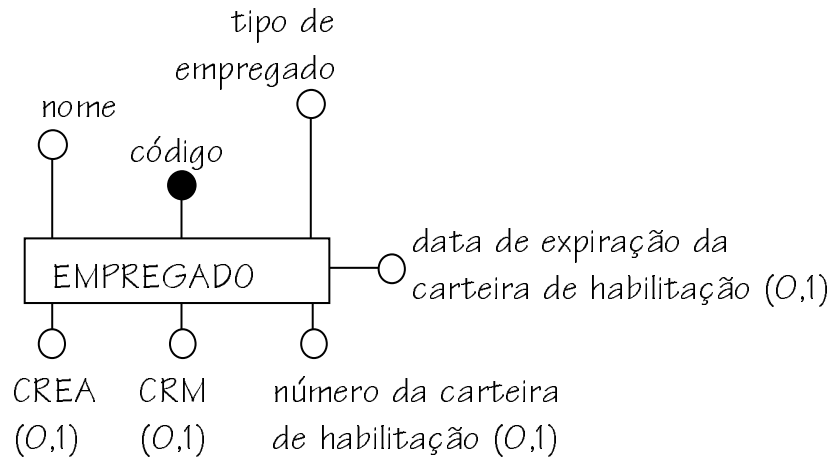


Figura 3.7: Atributos opcionais

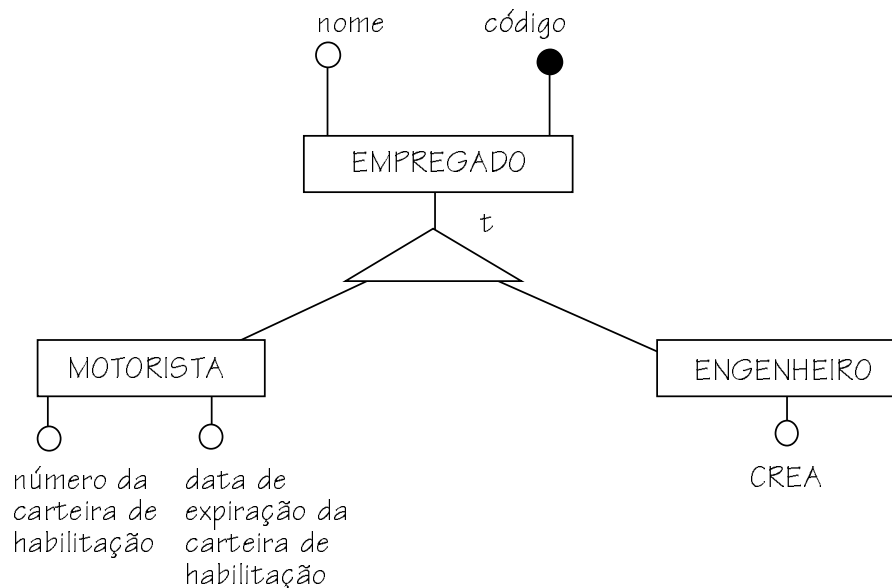


Figura 3.8: Aumentando a fidelidade do modelo através de especializações

O problema que ocorre no exemplo, é que 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 4.7. Neste modelo fica claro quais os atributos de cada um dos subconjuntos particulares de **EMPREGADO**. Assim, toda vez que aparecer um atributo opcional é aconselhável verificar se a modelagem através de entidades especializadas não é mais conveniente.

### 3.2.3.2 Atributo multi-valorado

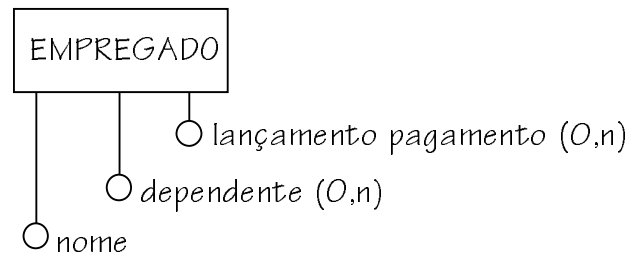


Figura 3.9: Usando atributos multi-valorados

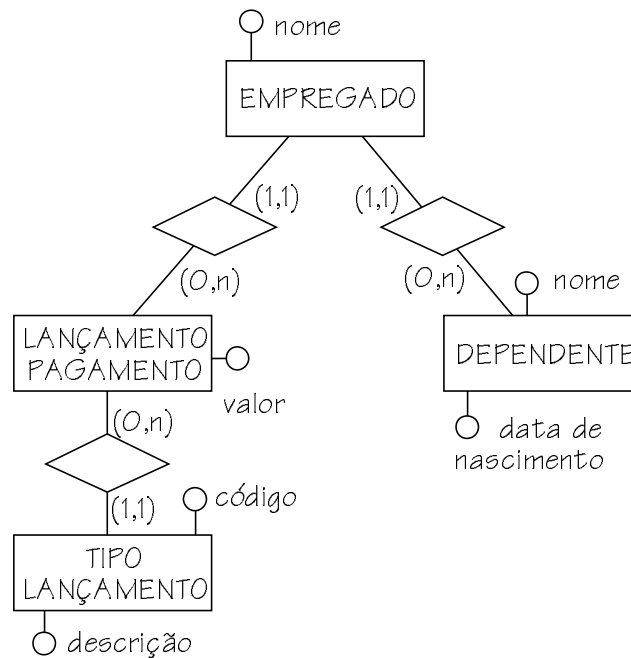


Figura 3.10: Substituindo atributos multi-valorados por entidades relacionadas

Atributos multi-valorados são indesejáveis por duas razões:

- ❑ Nos SGBD relacionais que seguem o padrão SQL/2, atributos multi-valorados não possuem implementação direta. Não existe em um SGBD relacional uma construção como os “arrays” de Pascal, nem como os grupos repetidos de COBOL. Assim, caso esteja sendo projetada um banco de dados relacional, é aconselhável usar apenas atributos mono-valorados.
- ❑ Atributos multi-valorados podem induzir a um erro de modelagem, que é o de ocultar entidades e relacionamentos em atributos multi-valorados. A Figura 3.9 mostra um DER em que são modelados empregados. Como atributos de EMPREGADO aparecem o nome do empregado, seus dependentes e os lançamentos de pagamento que compõem seu contracheque. 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 3.10.

### 3.3 VERIFICAÇÃO DO MODELO

Uma vez confeccionado, 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, ser correto e não conter redundâncias.

#### 3.3.1 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, os erros *sintáticos* e os erros *semânticos*. 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 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 a **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 usar banco como atributo de uma 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 próximo capítulo servem também para verificar a correção de modelos ER.

#### 3.3.2 Modelo deve ser completo

Um modelo completo deve fixar todas propriedades desejáveis do banco de dados. Isso obviamente 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 verificar 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 mo-



delo deve ser completo, estamos exigindo que todas propriedades *expressáveis* com modelos ER apareçam no modelo.

### 3.3.3 Modelo deve ser livre de redundâncias

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 que são resultado da combinação de outros relacionamentos entre as mesmas entidades. A Figura 3.11 apresenta um DER com relacionamentos redundantes.

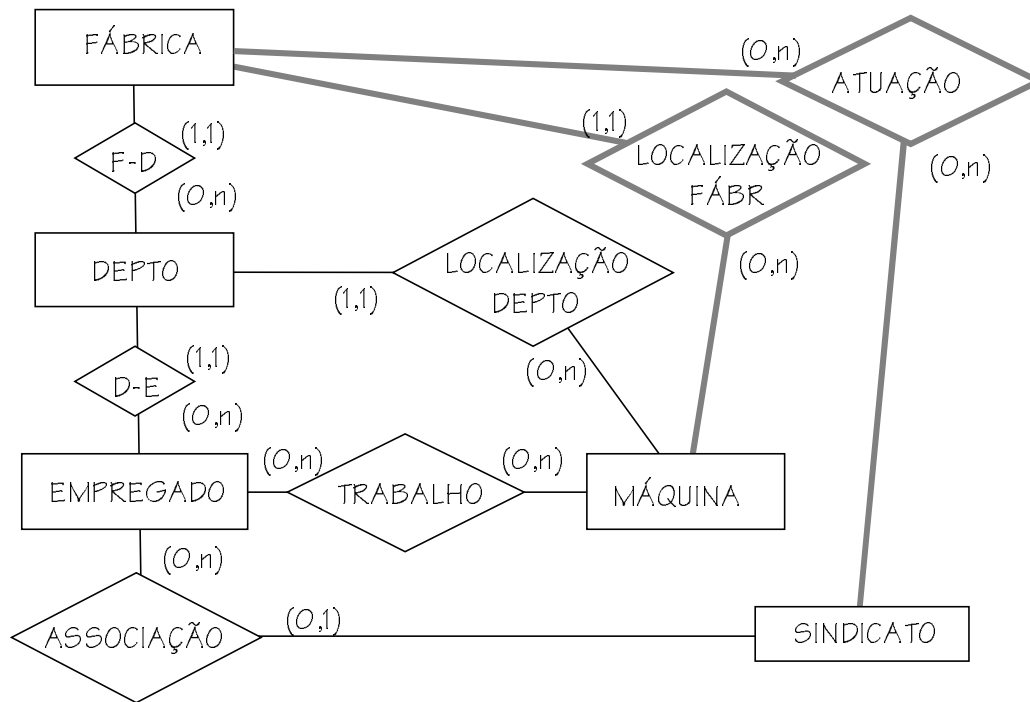


Figura 3.11: Relacionamentos redundantes

Os relacionamentos desenhados em linhas densas no DER da Figura 3.11 são redundantes. O relacionamento **LOCALIZAÇÃO-FÁBR** entre **MÁQUINA** e **FÁBRICA** é redundante. Um relacionamento é redundante, quando é possível eliminá-lo do diagrama 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-DEPT** 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, ela informa por consequência em que fábrica uma máquina está localizada. Seguindo 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**.

Um outro tipo de redundância que pode aparecer em modelos ER é a de atributos redundantes. *Atributos redundantes* são atributos deriváveis a partir da execução de procedimentos de busca de dados e/ou cálculos sobre o banco de dados.

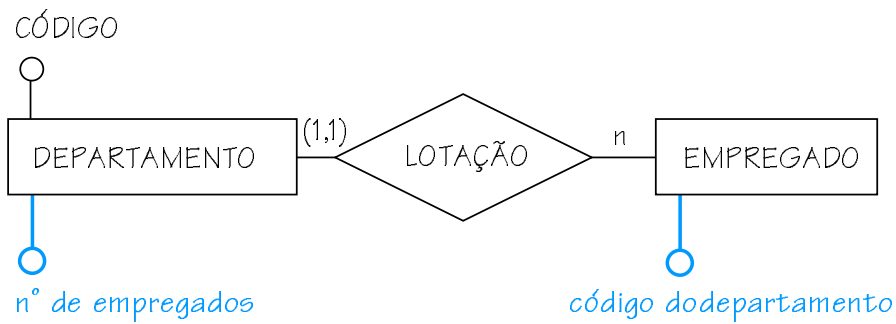


Figura 3.12: Atributos redundantes

A Figura 3.12 apresenta um DER 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**<sup>5</sup>. 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 diagrama ER não distingue construções derivadas das demais construções, o projetista do banco de dados poderia ser levado a implementá-las gerando redundância não controlada de dados. Redundância não controlada de dados em um banco de dados é indesejável por diversas razões. Construções redundantes normalmente implicam desperdício de espaço de armazenamento. Construções redundantes podem resultar em informações incorretas, caso não sejam alteradas em sincronia com as informações das quais são derivadas. Observe que isso não significa que redundância *controlada* 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 Modelo deve refletir o aspecto temporal

Usualmente, ao iniciar a modelagem ER, a preocupação é obter um modelo que descreva os estados válidos e corretos do banco de dados. O primeiro modelo tende a refletir um estado momentâneo do banco de dados. Entretanto, é necessário lembrar que assim como informações são incluídas no banco de dados, elas também podem ter que ser eliminadas do banco de dados. Um banco de dados não pode crescer indefinidamente. Informações ultrapassadas ou desnecessárias podem ser eliminadas. Portanto, é necessário considerar o *aspecto temporal* na modelagem de dados. Não há regras gerais de

---

<sup>5</sup>Para o leitor afeito a terminologia de bancos de dados relacionais, *código do departamento* é uma assim chamada *chave estrangeira*. Este atributo é usado em um banco de dados relacional, como veremos nos próximos capítulos, para implementar o relacionamento **LOTAÇÃO**. Como no modelo ER ainda não entramos em detalhes de implementação com SGBD relacional, o atributo deve ser omitido.

como proceder neste caso, mas é possível identificar alguns padrões que repetem-se freqüentemente na prática.

#### 3.3.4.1 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 questões de 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. Num sistema de pagamento, não interessa saber 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 como uma entidade. A figura 4.12 apresenta as duas alternativas de modelagem.

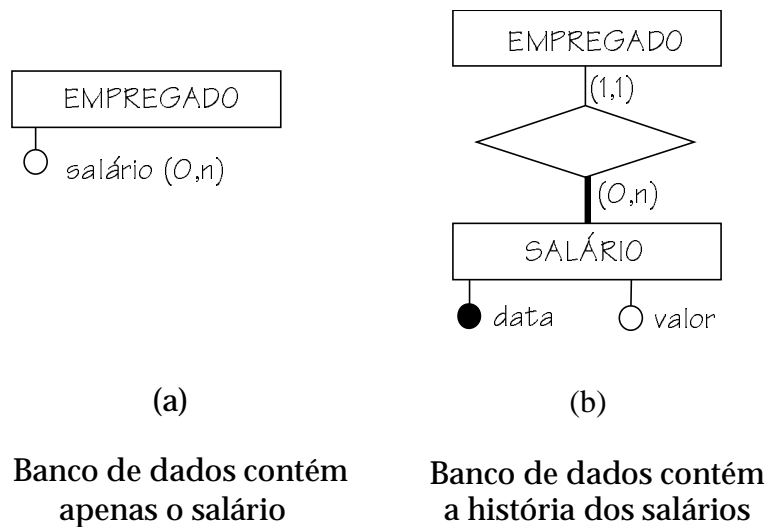


Figura 3.13: Modelando a dimensão temporal de atributos

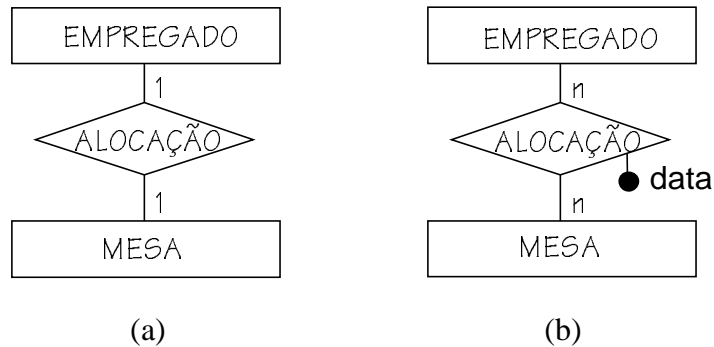
#### 3.3.4.2 Relacionamentos que modificam ao longo do tempo

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

Em geral, relacionamentos que, ao considerar apenas o estado atual do banco de dados, possuem cardinalidade 1:1 ou 1:n são transformados em cardinalidade n:n, quando é considerada a história das alterações de relacionamento.

Para exemplificar, consideramos o relacionamento **ALOCAÇÃO** da Figura 3.14(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 deseje-se armazenar no banco de dados apenas a alocação atual de cada mesa. Entretanto, caso deseje-se armazenar também a história das alocações, isto é, que empregados

estiveram alocados a que mesas ao longo do tempo, é necessário modificar o modelo (Figura 3.14 (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 a mesma mesa múltiplas vezes, torna-se necessário um atributo identificador do relacionamento, afim 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**.



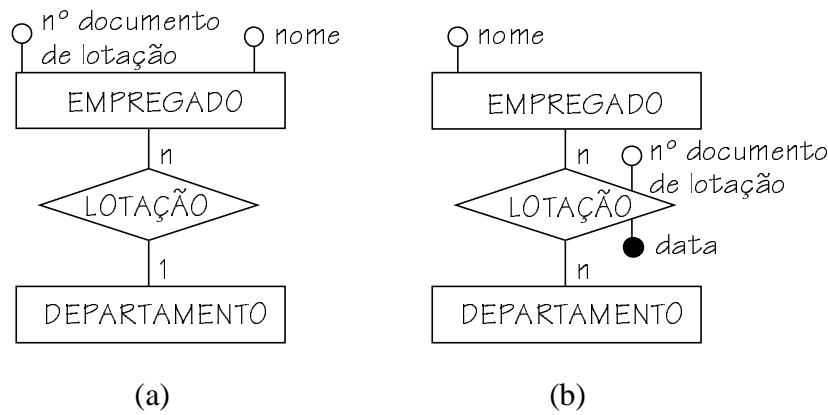
Base de dados contém apenas a alocação atual

Base de dados contém a história das alocações

Figura 3.14: Modelando a dimensão temporal de relacionamentos 1:1

A Figura 3.15 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 caso do exemplo, isto ocorre com o atributo **nº documento 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.

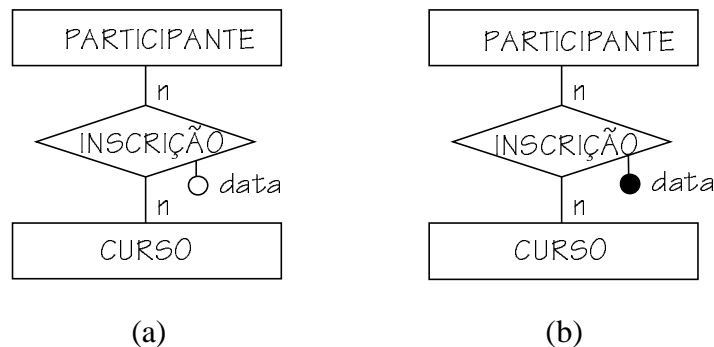
A Figura 3.16 apresenta mais um caso de consideração de história de relacionamentos. Neste caso, trata-se de um relacionamento de cardinalidade  $n:n$ . Modela-se a inscrição de participantes nos cursos oferecidos por uma empresa de treinamento. Na primeira versão, considera-se apenas os cursos nos quais 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 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.



Base de dados contém apenas a lotação atual

Base de dados contém a história das lotações

Figura 3.15: Modelando a dimensão temporal de relacionamentos 1:n



Base de dados contém apenas a inscrição atual

Base de dados contém a história das inscrições

Figura 3.16: Modelando a dimensão temporal de relacionamentos n:n

### 3.3.4.3 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 realização de auditorias ou para 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 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 usados 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 à base de dados estejam relacionadas a outras ocorrências, é necessário que estas estejam presentes. Se elas também tiverem sido excluídas, deverão ser igualmente devolvidas à base de dados. Essa inclusão pode ser propagada em cascata para outras entidades.

#### ❑ *Planejar informações estatísticas*

Em alguns casos, informações antigas são necessárias apenas para 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 nenhum 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 BD está embutida. Tomemos como exemplo o BD de uma universidade que aparece na Figura 2.13. A entidade **UNIVERSIDADE** pode ser necessária, caso se deseje manter no BD alguns atributos da universidade. Mesmo assim, o modelo não deveria conter o relacionamento desta entidade com outras, como **ALUNO** ou **CURSO**. Como o BD modela uma única universidade, não é necessário informar no BD em que universidade o aluno está inscrito ou a qual universidade o curso pertence.

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

Uma outra situação que não está incorreta, mas deve ser investigada, é a de uma entidades sem atributos.

## **3.4 ESTABELECENDO PADRÕES**

Modelos de dados são usados para 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 não aparece uma versão apenas de modelo ER, mas muitas, 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 de modelos ER**

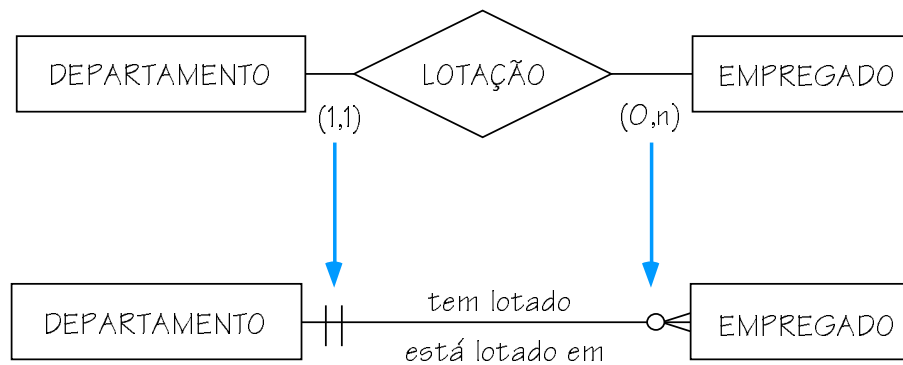
Quando de seu surgimento na literatura, a abordagem ER não era ainda suportada por ferramentas em computador de edição e manipulação, como as

ferramentas CASE hoje existentes. Com isso, 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 consequência 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 e que é muitas vezes usada na prática. Essa 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.

Além desta notação duas famílias de notações têm importância, a notação *Engenharia de Informações* e a notação *MERISE*.

#### 3.4.1.1 Notação Engenharia de Informações

A Engenharia de Informações é uma metodologia de desenvolvimento de sistemas de informação proposta no início da década de 80 por Martin e Finkelstein. Essa metodologia tem uma importância considerável na prática, comprovada pela existência de ferramentas CASE que a suportam. A característica mais importante desta metodologia é seu embasamento na modelagem de dados. A organização é vista fundamentalmente através do modelo de dados.



Notação para cardinalidade máxima e mínima:

- | Cardinalidade (mínima, máxima) 1
- Cardinalidade mínima 0
- ≧ Cardinalidade máxima n

Figura 3.17: Notação Engenharia de Informações

Para a Engenharia de Informações, foi definida uma notação especial de diagramas ER, conhecida como notação Engenharia de Informações, ou também notação James Martin.

A Figura 3.17 apresenta um exemplo de um DER na notação Chen até aqui empregada e o correspondente DER na notação Engenharia de Informações. As diferenças principais são as seguintes:

- ❑ Na notação Engenharia de Informações, relacionamentos são representados apenas por uma linha que liga os símbolos representativos das entidades associadas. Isso têm as seguintes consequências:
  - A notação admite apenas relacionamentos binários, já que uma linha apenas conecta duas entidades. Relacionamentos ternários ou de grau maior são modelados através de uma entidade associada através de relacionamentos binários a cada uma das entidades que participam do relacionamento ternário.
  - Atributos aparecem exclusivamente em entidades. Com isso, objetos que seriam modelados como relacionamentos n:n na notação de Chen tendem a ser modelados como entidades na notação de Engenharia de Informações.
- ❑ A denominação de um relacionamento é escrita na forma de verbos em ambas 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 a cardinalidade máxima, o mais distante a cardinalidade mínima.
- ❑ A generalização/especialização é chamada de subconjunto (subtipo) de entidades e é representada através do aninhamento dos símbolos de entidade conforme mostra a Figura 3.18.

A Figura 3.18 apresenta um exemplo mais abrangente representado com a notação da Engenharia de Informações (trata-se do mesmo modelo ER apresentado na Figura 2.37)

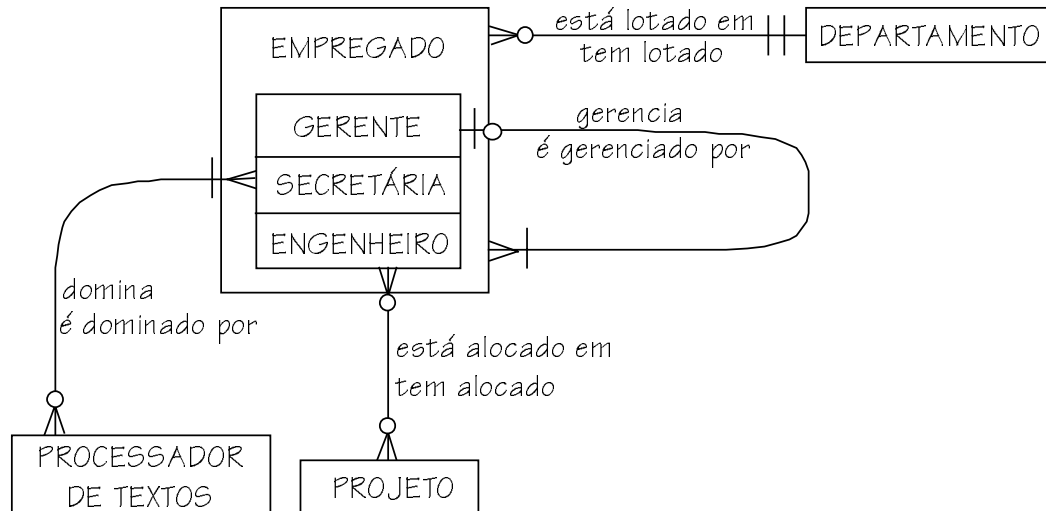


Figura 3.18: Modelo ER da Figura 2.37 na notação de Engenharia de Informações

### 3.4.1.2 Notação MERISE

MERISE é a metodologia de desenvolvimento de sistemas muito popular na França. Uma etapa desta metodologia, é a de modelagem de dados. Na modelagem de dados, MERISE adota diagramas ER, com uma notação um pouco



diferente da usual. Em princípio, esta notação não seria de maior importância para o leitor deste texto. Entretanto, há um grupo padronização na ISO (“International Standards Organization”) que está estudando a padronização de notações de DER. Relatórios preliminares deste grupo indicam que uma das técnicas que pode vir a ser adotada é a empregada em MERISE.

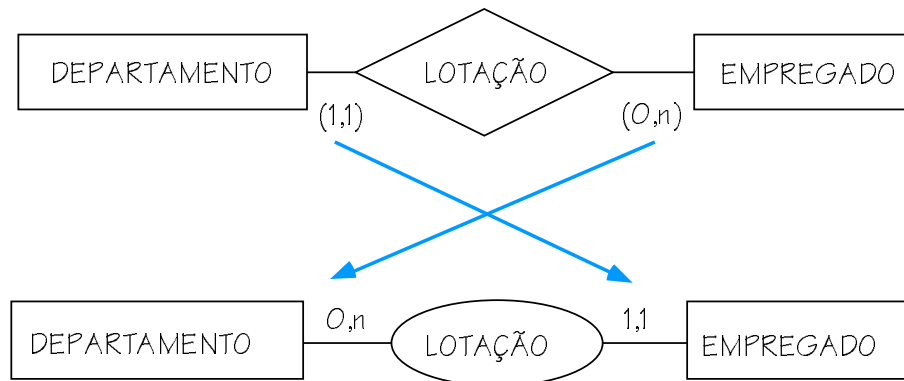


Figura 3.19: Notação Chen e correspondente notação MERISE

A Figura 3.19 apresenta um DER na notação Chen e o DER correspondente na notação MERISE. Como se observa, além da modificação cosmética de representar o relacionamento por uma elipse, ao invés de um losango, a posição em que são indicadas as cardinalidades mínima e máxima mudou. O motivo é que a interpretação dada a cardinalidade em MERISE é diferente da usual. 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 a 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.

### 3.4.2 Uso de ferramentas de modelagem

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

#### 3.4.2.1 Uso de uma ferramenta CASE

O software ideal para acompanhar o projeto de um banco de dados é uma ferramenta CASE (CASE - “computer aided software engineering”). Uma ferramenta CASE pode apoiar o desenvolvimento de um banco de dados tanto a

nível de modelagem quanto a nível 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, um pequeno banco de dados, onde 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.

### 3.4.2.2 Uso de programas de propósito geral

Em organizações com pequeno orçamento para a Informática, 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 trabalham com os conceitos de nodo e de arco. Nestes programas, quando o usuário mover um nodo do diagrama, todos arcos ligados a este nodo são também movimentados. Este tipo de programas são particularmente adequados para 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 passos pequenos, muitas pequenas transformações do 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, diferentes autores propuseram diferentes estratégias, que apresentamos abaixo.

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 sequência de idéias que se tem durante um processo de aprendizagem é dificilmente controlável por uma estratégia.

Para definir qual a estratégia a usar 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 pessoas possuem sobre o sistema.

### **3.5.1 Partindo de descrições de dados existentes**

Uma opção de 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.

Para estes casos, a estratégia mais adequada é a estratégia “*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 dispõe-se dos atributos. Normalmente, isso ocorre, quando já possui-se uma definição dos dados que deverão estar armazenados no banco de dados. Iremos tratar essa estratégia em detalhe em um capítulo a parte.

### **3.5.2 Partindo do conhecimento de pessoas**

A outra opção é partir do conhecimento 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 “*top-down*” e a “*inside-out*”.

#### **3.5.2.1 Estratégia “top-down”**

Esta estratégia 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 “*bottom-up*”. 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 sequência de passos para obter um modelo usando a estratégia “top-down” 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:
  - a) Enumeração das *entidades*.
  - b) Identificação dos *relacionamentos* e *hierarquias* de generalização/especialização entre as entidades.  
Para cada relacionamento identifica-se a *cardinalidade máxima*.
  - c) Determinação dos *atributos* de entidades e relacionamentos.
  - d) Determinação dos *identificadores* de entidades e relacionamentos.
  - e) O banco de dados é verificado quanto ao *aspecto temporal*.
2. *Modelagem detalhada* - Nesta etapa, completa-se o modelo com os domínios dos atributos e cardinalidades mínimas dos relacionamentos:
  - a) Adiciona-se os domínios dos atributos.
  - b) Define-se as cardinalidades mínimas dos relacionamentos. É preferível deixar estas cardinalidades por último, já que exigem conhecimento bem mais detalhado sobre o sistema, inclusive sobre as transações.
  - c) Define-se as demais restrições de integridade que não podem ser representadas pelo DER.
3. *Validação do modelo*
  - a) Procura-se construções redundantes ou deriváveis a partir de outras no modelo.
  - b) Valida-se o modelo com o usuário.

Em qualquer destes passos é possível retornar a 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.

### 3.5.2.2 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 importante 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 muitas outras são usualmente desenhadas no centro do diagrama, afim de evitar cruzamento de linhas. Um exemplo da aplicação desta estratégia é mostrado na Figura 3.20.

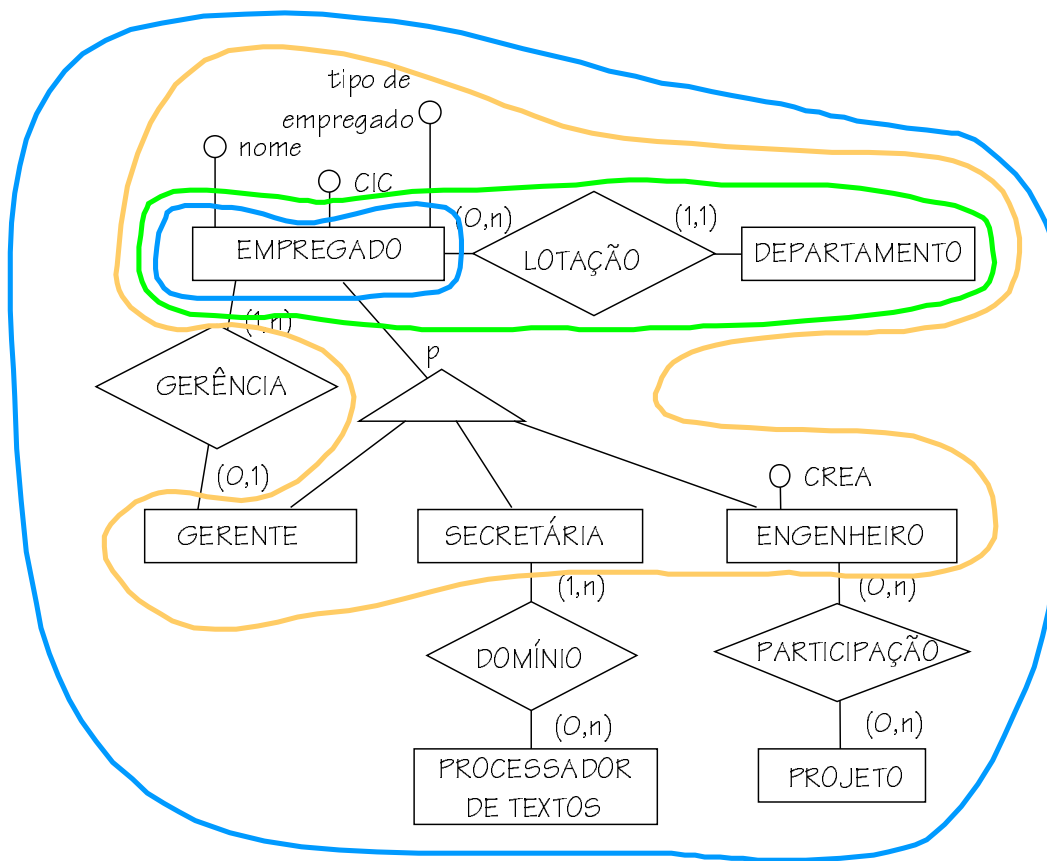


Figura 3.20: Aplicação da estratégia “de dentro para fora”

Na Figura 3.20, está mostrada 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.

## EXERCÍCIOS

**Exercício 3.1:** A Figura 3.21 apresenta um relacionamento que associa um produto de uma indústria com seus componentes (em inglês, “bill-of-materials”). Cada produto pode possuir 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:

a) O modelo apresentado na figura contém esta restrição?

- b) 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.
- c) É possível estender a solução do quesito anterior para uma hierarquia não limitada de níveis de composição?

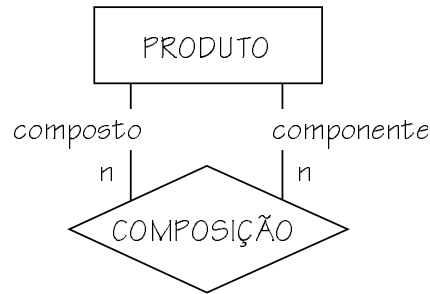


Figura 3.21: Composição de produtos

**Exercício 3.2:** Deseja-se modelar os clientes de uma organização. Cada cliente possui um identificador, um nome, um endereço e um país. Discuta os prós e contras das duas alternativas de modelagem de país:

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

**Exercício 3.3:** A Figura 3.22 apresenta uma entidade e respectivos atributos, muitos deles opcionais e um multi-valorado. Considere que há dois tipos de clientes, pessoas físicas e pessoas jurídicas. Pessoas físicas possuem código, CIC, nome, sexo (opcional), data de nascimento (opcional) e telefones (opcionais). Pessoas jurídicas possuem código, CGC, razão social e telefones (opcionais). Apresente um diagrama ER que modele mais precisamente esta realidade. Explique no que seu diagrama é mais preciso que o mostrado na figura 4.21.

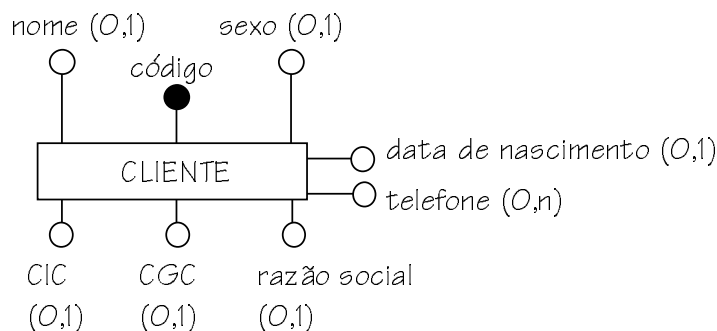


Figura 3.22: Cliente com atributos

**Exercício 3.4:** Transforme o modelo ER resultante do exercício anterior da notação Chen para a notação da Engenharia de Informações.

**Exercício 3.5:** 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:

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

### **Exercício 3.6: Estudo de caso - Locadora de vídeos**

(adaptado do material de um curso de modelagem de dados da Oracle)

Uma pequena locadora de vídeos possui ao redor de 2.000 fitas de vídeo, cujo empréstimo deve ser controlado.

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 prenome 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 3.7: 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, qual é o aeroporto no qual o vôo passa.

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

Nem todos vôos operam em todos dias de 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:

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

### **Exercício 3.8: Estudo de caso - Sistema para locadora de veículos**

O objetivo deste estudo de caso é construir um modelo ER para o BD 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 CIC. Já para as pessoas jurídicas é necessário conhecer seu nome, CGC, inscrição estadual e endereço. Os clientes são identificados por um código interno a 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. Entretanto, 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:

- a) os veículos disponíveis em determinada filial na data corrente,
- b) as reservas para veículos em uma filial, com previsão de que veículos estarão disponíveis em uma data futura,
- c) 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 (Uno, Palio, etc.) e G4 a grandes automóveis de luxo (Omega ou similar). 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, toca-fitas, CD, 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 3.9: 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 comparar 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 funções listadas abaixo:

- ❑ 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 sequencialmente 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 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 GT que promovem o congresso

- pessoas que de alguma forma já participaram de congressos promovidos pelos GT que promovem o congresso
- pessoas sugeridas para tal pelos membros dos GT 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 GT 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 efetivamente convidada.

Observar que uma mesma pessoa pode atuar em muitas funções, mesmo 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 a 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 BD, estarão não só as informações sobre os congressos em organização, mas também sobre os do passado.

### **Exercício 3.10: 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 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 o de 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 quando 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 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 quantidade ele deve buscar em que receptáculos.

Em termos de processos, é necessário que o sistema processe o seguinte:

- ☐ 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 controle de empilhadeiras ou estrados disponíveis e em uso.

## REFERÊNCIAS BIBLIOGRÁFICAS

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 e Navathe [2] é o mais completo sobre o processo de modelagem conceitual. Dedicar um capítulo exclusivamente a estratégias para obter o modelo de dados.

O livro de Martin [4] descreve a chamada Engenharia de Informações. Apresenta a notação que leva este nome. O tomo II contém alguns conselhos práticos de como modelar.

Uma versão gaúcha da Engenharia de Informações é apresentada em [3]. É o manual da metodologia de desenvolvimento da PROCERGS. A notação empregada difere da de Martin, sendo semelhante a de Chen.

Barker [1] apresenta 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 a notação da Engenharia de Informações.

O livro [5] apresenta a metodologia Merise, na qual aparece a notação que leva seu nome.

- [1] Barker, R. CASE Method™: Entity Relationship Modeling. Addison-Wesley, 1990
- [2] Batini, C., Ceri, S., and Navathe, S. *Database Design: An Entity-Relationship Approach*, Benjamin/Cummings 1992.
- [3] Kipper, E.F. et al. *Engenharia de Informações: Conceitos Técnicas e métodos*, Sagra DC-Luzzato, Porto Alegre, 1993.
- [4] Martin, James. *Information Engineering: Books I, II & III*. Englewood Cliffs: Prentice-Hall, 1990.
- [5] Tardieu, H., Rochfeld, A. and Colleti, R. *Le Methode Merise: Principes et Outils*. Les Editions d'Organization, Paris, 1983

## Abordagem Relacional

Este capítulo apresenta uma introdução sucinta ao modelo de dados que é usado nos sistemas de gerência de banco de dados do tipo relacional. Não é uma introdução completa à abordagem 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 por exemplo, SQL. Para maiores detalhes sobre sistemas de BD relacionais, o leitor deve procurar livros específicos (ver a bibliografia deste capítulo).

Além dos SGBD relacionais, existem outros tipos de sistemas no mercado. Entretanto, hoje, há um claro predomínio dos SGBD relacionais, principalmente fora das plataformas de grande porte. Mesmo nestes ambientes, os SGBD relacionais estão gradativamente substituindo os SGBD 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 e nos livros-texto. 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 Tabelas

Uma tabela é um conjunto não ordenado de *linhas* (tuplas, na terminologia acadêmica). Cada linha é composta por uma série de *campos* (valor de atributo, na terminologia acadêmica).

Cada campo é identificado por *nome de campo* (nome de atributo, na terminologia acadêmica). O conjunto de campos das linhas de uma tabela que possuem o mesmo nome formam uma *coluna*.

Emp	CódigoEmp	Nome	CodigoDepto	CateqFuncional
E5		Souza	D1	C5
E3		Santos	D2	C5
E2		Silva	D1	C2
E1		Soares	D1	—

Figura 4.1: Tabela

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 estão ordenadas*. 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 *mono-valorados*. Em arquivos convencionais, campos podem ser compostos por outros campos (“itens de grupo”) e campos podem ser multi-valorados (“arrays” de Pascal, ou grupos repetidos de COBOL).
- ❑ As linguagens de consulta a bases de dados relacionais permitem o *acesso por quaisquer critérios* envolvendo os campos de uma ou mais linhas. Em arquivos convencionais, para buscar registros com base em valores de seus campos de forma rápida é usualmente 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 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 à base de dados sem considerar a existência ou não de caminhos de acesso.

### 4.1.2 Chaves

O conceito básico para estabelecer relações entre linhas de tabelas de um banco de dados relacional é o da *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*.

#### 4.1.2.1 Chave primária

Uma chave primária é uma coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela. Na tabela *Emp* 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 *CódigoEmp* e *NoDepen*). Neste caso, apenas um dos valores dos campos que compõem a chave não é suficiente para distinguir uma linha das demais, já que tanto um código de empregado (*CódigoEmp*) pode aparecer em diferentes linhas, quanto um número de dependente (*NoDepen*) pode aparecer em diferentes linhas. É necessário considerar ambos valores (*CódigoEmp* e *NoDepen*) para identificar uma linha na tabela, ou seja para identificar um dependente.

Dependente				
CódigoEmp	NoDepen	Nome	Tipo	DataNasc
E1	01	João	Filho	12/12/91
E1	02	Maria	Esposa	01/01/50
E2	01	Ana	Esposa	05/11/55
E5	01	Paula	Esposa	04/07/60
E5	02	José	Filho	03/02/85

Figura 4.2: Tabela com chave primária composta

Pela definição acima, na tabela da Figura 4.2, qualquer combinação de colunas que contenha as colunas *CódigoEmp* 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 suas colunas forem efetivamente necessárias para garantir o requisito de unicidade de valores da chave. Exemplificando, alguém poderia considerar a combinação de colunas *CódigoEmp*, *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 *CódigoEmp*, *NoDepen* e *Tipo* não obedece o princípio da minimalidade e não deve ser considerada uma chave.

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 estados válidos da BD. No caso da chave primária, a regra é a de unicidade de valores nas colunas que compõem a chave.

#### 4.1.2.2 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**. Isso significa que, na tabela **Emp**, não podem aparecer linhas que contenham um valor do campo **CodigoDepto** que não exista na coluna de mesmo nome da tabela **Emp**. A interpretação desta restrição é que todo empregado deve estar associado a um departamento.

Dept				
CodigoDepto		NomeDepto		
D1		Compras		
D2		Engenharia		
D3		Vendas		

Emp				
CodigoEmp	Nome	CodigoDepto	CategFuncional	CIC
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

A existência de uma chave estrangeira impõe restrições que devem ser garantidas em diversas situações 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 caso do exemplo da Figura 4.3, isso 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 caso do exemplo da Figura 4.3, isso significa que um departamento não pode ser excluído, caso nele ainda existirem empregados.

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 **CódigoEmpGerente** é o código de um outro empregado, o gerente do empregado correspondente a linha em questão. Como todo gerente é ele mesmo também um empregado, existe a restrição de que todo valor da coluna **CódigoEmpGerente** deve aparecer na coluna **CódigoEmp**. Assim, a coluna **CódigoEmpGerente** é chave estrangeira em relação a chave primária da própria tabela **Emp**.

Emp			
CódigoEmp	Nome	CodigoDepto	CódigoEmpGerente
E5	Souza	D1	—
E3	Santos	D2	E5
E2	Silva	D1	E5
E1	Soares	D1	E1

chave estrangeira:  
referencia a chave primária  
da própria tabela

Figura 4.4: Chave estrangeira dentro da própria tabela

#### 4.1.2.3 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 **CódigoEmp** quanto a coluna **CIC** podem ser usadas para distinguir uma linha das demais. Nesta tabela, como a coluna **CódigoEmp** foi escolhida como chave primária, diz-se que a coluna **CIC** é uma chave alternativa.

Emp				
CódigoEmp	Nome	CodigoDepto	CategFuncional	CIC
E5	Souza	D1	C5	132.121.331-20
E3	Santos	D2	C5	891.221.111-11
E2	Silva	D1	C2	341.511.773-45
E1	Soares	D1	—	631.692.754-88

Figura 4.5: Chave alternativa (coluna CIC)

Quando, em uma tabela, mais de uma coluna ou combinações de colunas podem servir para distinguir uma linha das demais, surge a questão de que critério deve-se usar para determinar qual das possíveis colunas (ou com-

binação de colunas) será usada como chave primária. No exemplo da Figura 4.5, a questão é que critério foi usado para preferir a coluna **CódigoEmp** como chave primária e considerar a coluna **CIC** como chave alternativa. Porque **CIC** não foi usado como chave primária e **CódigoEmp** como chave alternativa? No fundo, se considerarmos apenas a tabela em que a coluna aparece, não há diferença entre uma coluna ser chave primária ou alternativa. Em ambos 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 **CódigoEmp** quanto os valores de **CIC** são únicos e adicionalmente que a coluna **CódigoEmp** será usada nas chaves estrangeiras que referenciam a tabela **Emp**.

### 4.1.3 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” em inglês) ou não. Estar vazio indica que o campo não recebeu nenhum valor de seu domínio<sup>6</sup>. Na Figura 4.5, o campo **CategFuncional** da linha correspondente ao empregado de código **E1** está vazio. Isso 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 **CodigoEmpGerente** vazio indica que o empregado não possui superior hierárquico. 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 SGBD relacional exigem que todas 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).

### 4.1.4 Restrições de integridade

Um dos objetivos primordiais de um SGBD é a *integridade de dados*. 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 SGBD oferecem o mecanismo de *restrições de integridade*. Uma restrição de integridade é uma regra de consistência de dados que é garantida pelo pró-

---

<sup>6</sup>Alguns tradutores e até mesmo autores de textos sobre banco de dados em português tem usado a palavra “nulo” ao invés de “vazio”, lembrando a palavra “null” em inglês. Esta tradução não me parece adequada, pois pode levar a confusões entre o vazio (diferente de todos valores dos domínios dos campos) e o valor zero (nulo), que pode aparecer no domínio dos campos numéricos

prio 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 SGBD relacionais comerciais, é possível usar apenas domínios pré-definidos (número inteiro, número real, alfanumérico de tamanho definido, data, ...). O usuário do SGBD não pode definir domínios próprios de sua aplicação (por exemplo, o domínio dos dias da semana ou das unidades da federação).

❑ *Integridade de vazios*

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 muitas outras restrições de integridade que não se encaixam em nenhuma das categorias acima e que normalmente não são garantidas pelo SGBD. Essas restrições são chamadas de restrições *semânticas*. Alguns exemplos de restrições deste 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.

## 4.2 ESPECIFICAÇÃO DE BANCO DE DADOS RELACIONAL

A especificação de um banco de dados relacional (chamada de *esquema* do banco de dados) deve conter no mínimo a definição do seguinte:

- ❑ Tabelas que formam o banco de dados
- ❑ Colunas que as tabelas possuem
- ❑ Restrições de integridade

Na prática, na definição de esquemas relacionais são usadas diversas notações, que variam de um SGBD para o outro. Nesta seção, vamos apresentar apenas uma notação resumida para modelos lógicos relacionais. Essa notação é incompleta mas compacta, que 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 detalhamento.

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

Emp (CodigoEmp,Nome,CodigoDepto,CategFuncional,CIC)  
CodigoDepto referencia Dept  
Dept (CodigoDepto,Nome)

Figura 4.6: Esquema do banco de dados da Figura 4.3

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 da 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.

### 4.3 CONSULTAS À BASE 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 afim de mostrar algumas características importantes das linguagens relacionais.

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

```
SELECT    Emp.Nome
FROM      Emp, Dept
WHERE     Dept.Nome LIKE "Computação" AND
          Emp.CodigoDepto = Dept.CodigoDepto AND
          Emp.CategFuncional="Programador"
```

A consulta acima busca os nomes dos empregados que estejam vinculados a um departamento em cujo nome aparece a palavra **Computação** e que pertencem a categoria funcional **Programador**. Quanto a esta consulta, cabem as seguintes observações:

- ❑ Na instrução SQL, o programador não faz referência a nenhum tipo de caminho de acesso. Quem decide que 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 caso do exemplo as linhas de **Emp** são associadas às linhas de **Dept** pela igualdade do código do departamento.

## EXERCÍCIOS

**Exercício 4.1:** Considere o banco de dados relacional definido parcialmente abaixo (faltam as chaves da tabela [Empregado](#)):

[Empregado\(CodigoEmpregado, Nome, NoPIS-PASEP\)](#)  
[Dependente\(CodigoEmpregado, NoDependente, Nome\)](#)  
[CodigoEmpregado](#) referencia [Empregado](#)

Na tabela [Empregado](#), tanto [CodigoEmpregado](#) quanto [NoPIS-PASEP](#) podem ser chave primária. Qual você escolheria como chave primária? Porque?

**Exercício 4.2:** Tome um livro sobre organização de arquivos e veja o que significa “chave secundária”. Explique porque o conceito de chave secundária não aparece na abordagem relacional.

**Exercício 4.3:** Abaixo aparece um esquema parcial para um banco de dados relacional. Identifique neste esquema as chaves primárias e 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 4.4:** Para o banco de dados cujo esquema está definido abaixo, explique que verificações devem ser feitas pelo SGBD para garantir integridade referencial nas seguintes situações:

- a) Uma linha é incluída na tabela [Consulta](#).
- b) Uma linha é excluída da tabela [Paciente](#).  
[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](#)

## REFERÊNCIAS BIBLIOGRÁFICAS

Os livros de Elmasri e Navathe [1] e Korth e Silberschatz [3] 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 SGBD relacionais comerciais e a linguagem SQL segundo o padrão estabelecido em 92 é o livro de Groff e Weinberg [2]. O livro tem mais um formato de um manual de linguagem que propriamente um livro acadêmico, mas faz uma excelente cobertura da linguagem SQL padrão e de suas implementações em vários SGBD.

- [1] Elmasri, R. & Navathe, S.B. Fundamentals of Database Systems. Second Edition. Benjamin/Cummings, Redwod City, California, 1994
- [2] Groff, J.R.; Weinberg, P.N. *LAN Times Guide to SQL*. Mc-Graw-Hill, Berkeley, CA, 1994
- [3] Korth, H. & Silberschatz, A. *Sistemas de Bancos de Dados*. 2ª edição, Makron Books, 1994

## 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. É adequada para construção do *modelo conceitual*. Já a abordagem relacional modela os dados a nível de SGBD relacional. Um modelo neste nível de abstração é chamado de *modelo lógico*.

No presente 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, a 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 é tratado somente o projeto de BD relacional.

A seguir, vamos apresentar o processo inverso ao projeto, que é chamado de *engenharia reversa* de BD relacional. 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.



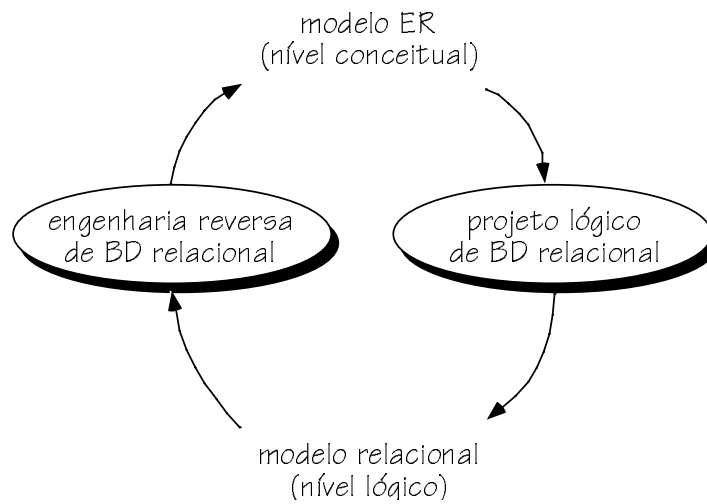


Figura 5.1: Transformações entre modelo ER e modelo relacional

## 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. Todos podem ser considerados 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 maior facilidade, ou dificuldade de desenvolvimento e manutenção do sistema construído sobre o banco de dados.

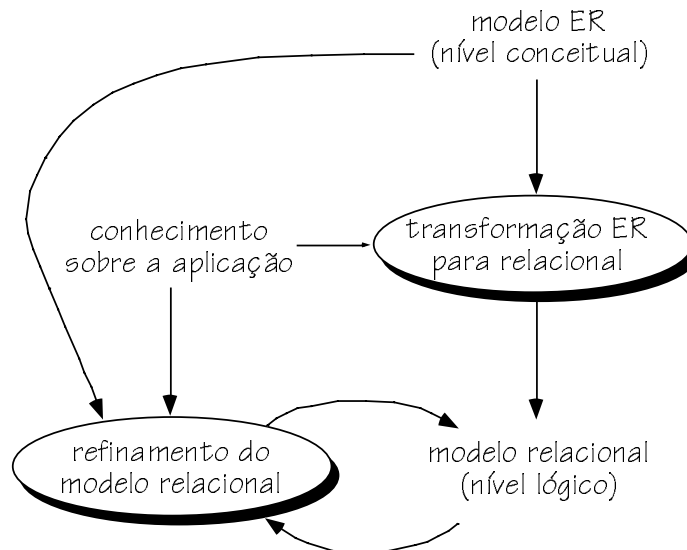


Figura 5.2: Visão geral do projeto lógico

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 atende 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.

## 5.2 TRANSFORMAÇÃO ER PARA RELACIONAL

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

As regras foram definidas tendo em vista dois objetivos básicos:

- ❑ 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 dois objetivos são os 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 performance e simplificar o desenvolvimento. Entretanto, nos últimos anos, o preço dos meios de armazenamento vem diminuindo constantemente, o que fez com que a redução de espaço ocupado, principalmente se em detrimento dos demais objetivos, diminuísse de importância.

Afim de alcançar estes objetivos, 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 isso, todos dados de uma linha são trazidos para a memória em uma operação de acesso a disco. Isso significa que, uma vez encontrada uma linha de uma tabela, seus campos estão todos disponíveis sem necessidade de acesso 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) é necessário usar a operação de *junção*. Os SGBD 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 somente, ao invés de tê-los distribuídos em diversas linhas, exigindo a sua junção.
- ❑ *Diminuir o número de chaves primárias*  
Para a implementação eficiente do controle da unicidade da chave primária, o SGBD usa normalmente uma estrutura de acesso auxiliar, um

índice, para cada chave primária. Índices, pela forma que são implementados, tendem a ocupar espaço considerável em disco. Além disso, a inserção ou remoção de entradas em um índice podem exigir diversos acessos ao disco. Assim sendo, quando for necessário escolher entre duas alternativas de implementação, uma na qual dados aparecem em uma única tabela e outra na qual os mesmos dados aparecem em duas ou mais tabelas com a mesma chave primária e mesmo número de linhas, a implementação por uma única tabela deve ser preferida.

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 gerada pelas regras apresentadas neste capítulo.

#### ❑ *Evitar campos opcionais*

Campos opcionais são campos que podem assumir o valor **VAZIO** (NULL em SQL). Os SGBD relacionais usualmente não desperdiçam espaço pelo fato de campos de uma linha estarem vazios, pois usam técnicas de compressão de dados e registros de tamanho variável no armazenamento interno de linhas. Além disso, há uma cláusula de SQL que especifica ao SGBD se o campo deve estar preenchido ou pode estar vazio. Assim, em princípio, não há problemas em usar este tipo de campos.

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 SGBD, o controle da obrigatoriedade deve ser feito pelos programas que acessam o banco de dados, o que deve ser evitado.

Estas regras usam como entrada, além do próprio modelo ER, também alguns conhecimentos sobre volumes de dados e volumes de transações. Esses conhecimentos permitem escolher uma alternativa de implementação, quando diversas alternativas podem ser usadas para implementar um conceito da abordagem ER.

A transformação de um modelo ER em um modelo relacional dá-se nos seguintes passos:

1. Tradução inicial de entidades e respectivos atributos

2. Tradução de relacionamentos e respectivos atributos
3. Tradução de generalizações/especializações

### 5.2.1 Implementação inicial de entidades

Esse 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 correspondem às 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 correspondente esquema relacional. A entidade **PESSOA** com seus atributos **código**, **nome** e **endereço** é transformada na tabela denominada **Pessoa** com colunas denominadas **CodigoPess**, **Nome** e **Endereço**. Como o atributo **código** é identificador da entidade, a coluna correspondente a este atributo é a chave primária da tabela.



Esquema relacional correspondente:

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

Figura 5.3: Transformação de entidade em tabela

#### 5.2.1.1 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 de programadores é conveniente manter os nomes de colunas curtos. Além disso, em um SGBD relacional, o nome de uma coluna não pode conter brancos. 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.

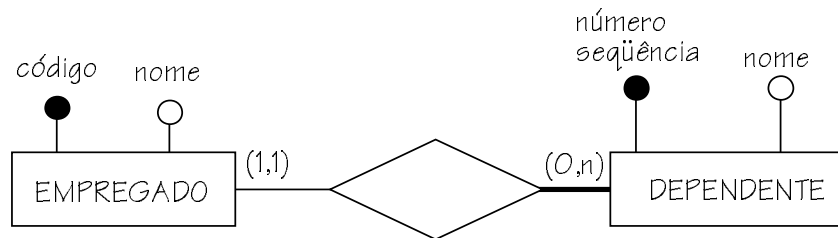
Nas linguagens de banco de dados, o nome da tabela é muitas vezes usado como qualificador do nome da coluna. Exemplificando, para referenciar a coluna **Nome** da tabela **Pessoa** muitas vezes é usado um termo na forma **Pessoa.Nome**. Por isso, não é recomendado incluir no nome de uma coluna o nome 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 aparecer em outras tabelas na forma de chave estrangeira, é recomendável que os nomes das colunas que compõem a chave primária sejam sufixadas ou prefixadas 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 a 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 toda o banco de dados.

#### 5.2.1.2 Relacionamento identificador

Há uma situação na qual a tradução de uma entidade para uma tabela não é trivial. Trata-se da situação na qual uma entidade possui um *relacionamento identificador*. Um exemplo de entidade deste tipo é a entidade [DEPENDENTE](#) mostrada no diagrama da Figura 5.4. 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 tradução de identificadores externos é que, para cada identificador externo seja criada uma coluna (ou várias no caso de o identificador externo ser composto de vários atributos) na tabela em questão, coluna esta que fará parte da chave primária da tabela. A Figura 5.4 mostra o esquema relacional para esta tradução da entidade [DEPENDENTE](#).

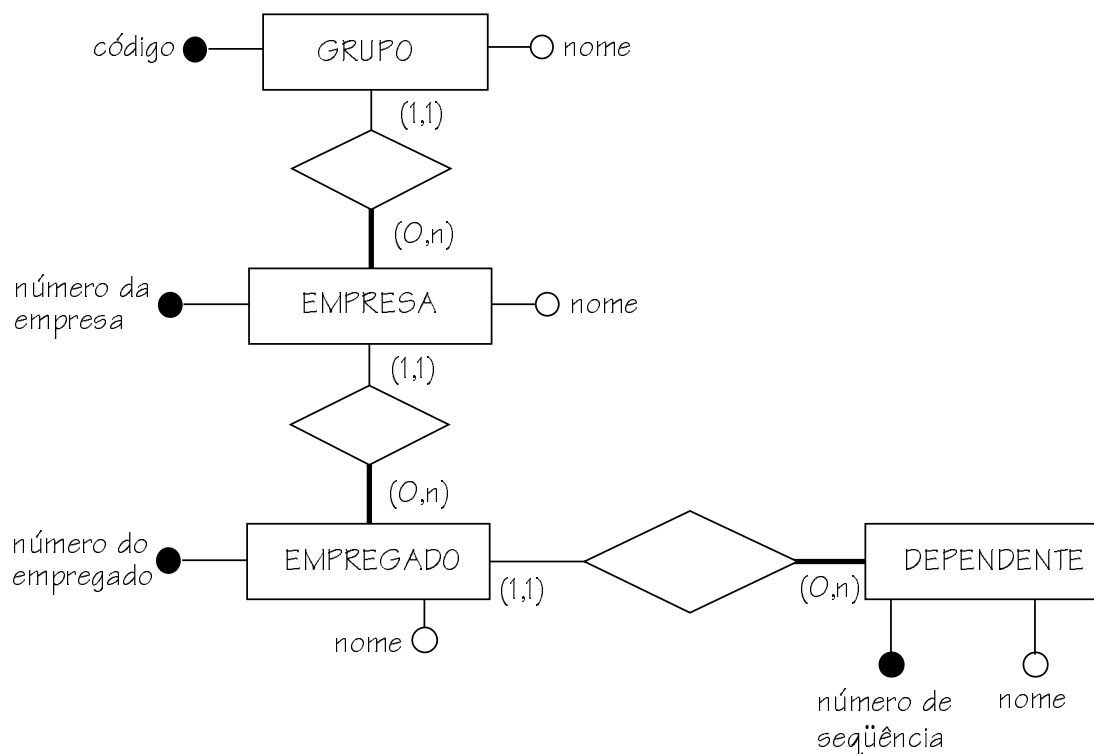


Esquema relacional correspondente:

[Dependente \(CodigoEmp,NoSeq,Nome\)](#)

Figura 5.4: Tradução de entidade com identificador externo

Cabe observar que, na composição da chave primária de uma tabela que possui identificador externo, pode ser necessário colecionar atributos de diversas entidades, conforme mostrado 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: Chave primária composta por diversos identificadores externos

No caso do exemplo, 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 a 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
- ☐ seu número de seqüência.

Essa linha de raciocínio nos leva à chave primária da tabela **Dependente**, que é mostrada na Figura 5.5.

### 5.2.2 Implementação de relacionamentos

O fator determinante para a tradução a adotar 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 relaciona-

mento, apresentamos três formas básicas de tradução de relacionamentos para o modelo relacional.

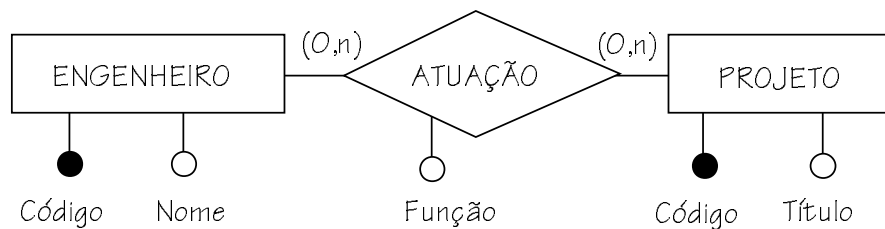
### 5.2.2.1 Tabela própria

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

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

A chave primária desta tabela é o conjunto das colunas correspondentes aos identificadores das entidades relacionadas. Cada conjunto de colunas que corresponde ao identificador de uma entidade é chave estrangeira em relação a tabela que implementa a entidade referenciada.

Um exemplo deste tipo de tradução é apresentado na Figura 5.6. A parte do esquema do banco de dados que se refere à regra em questão está apresentada em negrito. Essa convenção será usada no restante da apresentação das regras de tradução de relacionamentos.



Esquema relacional correspondente:

Engenheiro (CodEng,Nome)

Projeto (CodProj,Título)

Atuação (CodEng,CopProj,Função)

**CodEng referencia Engenheiro**

**CodProj referencia Projeto**

Figura 5.6: Tradução de um relacionamento para uma tabela

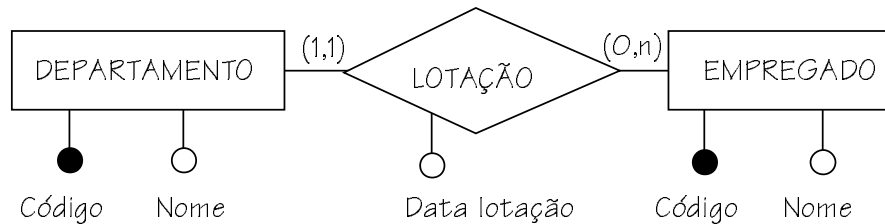
A tabela **Atuação** implementa o relacionamento **ATUAÇÃO**. A chave primária da tabela é formada pelas colunas **CodEng** e **CodProj**, que correspondem aos identificadores das entidades relacionadas (**ENGENHEIRO** e **PROJETO**). Cada uma destas colunas é chaves estrangeira das tabela que implementa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

### 5.2.2.2 Colunas adicionais dentro de tabela de entidade

A outra alternativa de implementação de um relacionamento é a inserção de colunas em uma tabela correspondente a uma das entidades que participam do relacionamento. Um exemplo deste tipo de tradução é apresentado na Figura 5.7. Esse tipo de tradução somente é possível quando uma das entidades que participa do relacionamento tem cardinalidade máxima um (no caso do exemplo, trata-se da entidade **EMPREGADO**). A tradução consta em inse-

rir na tabela correspondente à entidade com cardinalidade máxima 1 as seguintes colunas:

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



Esquema relacional correspondente:

**Departamento** (CodDept,Nome)

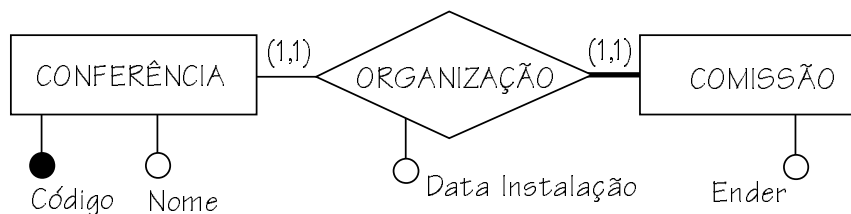
**Empregado** (CodEmp,Nome,CodDept,DataLota)

**CodDept referencia Departamento**

Figura 5.7: Tradução de relacionamento para colunas de uma tabela correspondente a uma entidade relacionada

### 5.2.2.3 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 de implementar todos atributos de ambas entidades, bem como os atributos do relacionamento em uma única entidade.



Esquema relacional correspondente:

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

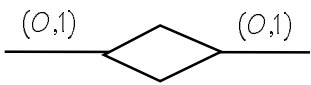
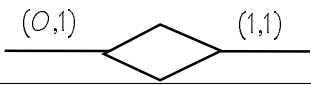
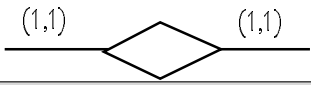
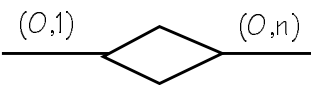
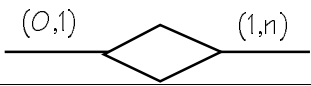
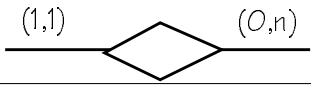
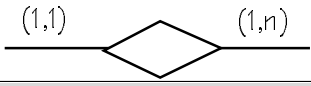
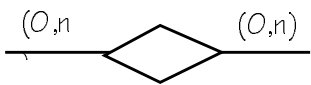
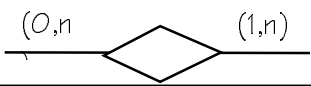
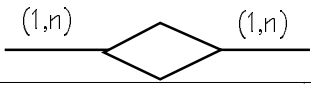
Figura 5.8: Tradução de relacionamento através de fusão de tabelas

### 5.2.3 Detalhes da implementação de relacionamentos

A regra específica que deve ser usada na tradução de um relacionamento é determinada pelas cardinalidades mínima e máxima das entidades envolvi-



das nos relacionamentos. A Tabela 5.1 dá uma visão geral das regras usadas. Para cada tipo de relacionamento a tabela mostra qual a alternativa que deve ser usada (alternativa preferida, indicada pelo símbolo ✓). Para alguns tipos de relacionamentos há ainda segunda alternativa a ser usada (indicada pelo símbolo ±), bem como uma alternativa que não deve ser usada (indicada pelo símbolo X). Nas seções que seguem, a regra de tradução correspondente a cada um dos tipos de relacionamentos é justificada e exemplificada.

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
<b>Relacionamentos 1:1</b>			
	±	✓	X
	X	±	✓
	X	±	✓
<b>Relacionamentos 1:n</b>			
	±	✓	X
	±	✓	X
	X	✓	X
	X	✓	X
<b>Relacionamentos n:n</b>			
	✓	X	X
	✓	X	X
	✓	X	X

✓ Alternativa preferida    ± Pode ser usada    X Não usar

Tabela 5.1: Regras para implementação de relacionamentos

### 5.2.3.1 Relacionamentos 1:1

#### 5.2.3.1.1 Ambas entidades têm participação opcional

A Figura 5.9 apresenta um exemplo de relacionamento 1:1 no qual a participação de ambas entidades é opcional (a cardinalidade mínima de ambas entidades no relacionamento é zero).

De acordo com a Tabela 5.1, a alternativa preferida de tradução de relacionamentos é a inserção 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.

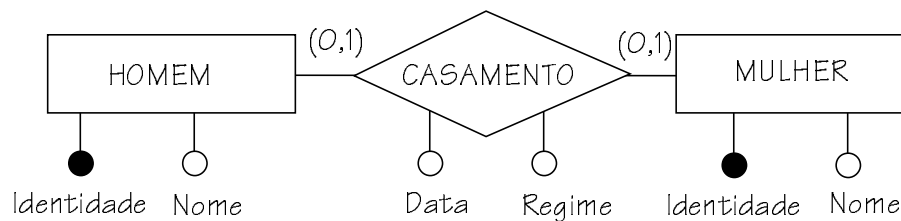


Figura 5.9: Implementação de relacionamento 1:1 com participação obrigatória de ambas entidades

Uma solução poderia ser a apresentada abaixo:

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 abaixo:

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 escolhida arbitrariamente como chave primária, sendo **IdentH** uma chave alternativa.

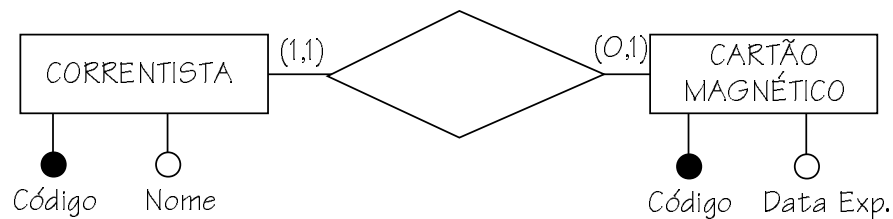
A primeira alternativa é a preferida, pois minimiza a necessidade de junções, já que os dados de uma pessoa (na opção escolhida a mulher) estão na mesma linha que os dados do casamento.

A desvantagem da primeira alternativa é que pode levar à utilização da segunda alternativa 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 estão vazios e outros preenchidos. O controle que garante que os três campos estão preenchidos ou que os três campos estão vazios não é feito pelo SGBD, e com isso tem que ser feito pela própria aplicação.

Cabe observar que em alguns SGBD, que implementam restrições de integridade (cláusula “*Assertion*” ou similar) é possível transferir ao SGBD a responsabilidade pelo controle das colunas opcionais. Neste caso, a segunda alternativa seria a preferida.

#### 5.2.3.1.2 Uma entidade tem participação opcional e a outra tem participação obrigatória

Outra 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 entidade é zero). Um exemplo desta situação é apresentada na Figura 5.10.



Esquema relacional correspondente:

**Correntista** (CodCorrent, Nome, CodCartão, DataExp)

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

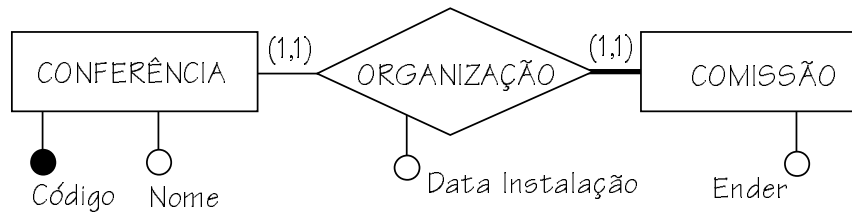
Neste caso, a tradução preferida é através da fusão das tabelas correspondentes às duas entidades.

Alternativamente, poderia ser considerada a tradução através da adição de colunas à tabela correspondente à entidade com cardinalidade mínima 0. No caso do exemplo, a tradução resultaria no esquema abaixo:

Correntista ([CodCorrent](#),Nome)  
Cartão([CodCartão](#),DataExp,[CodCorrent](#))  
**CodCorrent referencia Correntista**

#### 5.2.3.1.3 Ambas entidades tem participação obrigatória

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



Esquema relacional correspondente:

[Conferência](#) ([CodConf](#),Nome,DataInstComOrg,EnderComOrg)

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

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

Nenhuma das demais alternativas atende plenamente. Em ambas 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.

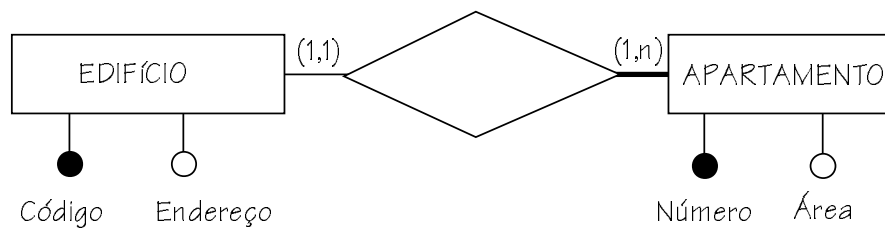
#### 5.2.3.2 Relacionamentos 1:n

No caso de relacionamentos 1:n, a alternativa preferida de implementação é a de *adição de colunas* (Seção 5.2.2.2).

Um exemplo desta tradução é apresentado na Figura 5.12.

Cabe observar que no caso particular deste 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 Seção 5.2.1.2.

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 a entidade [FINANCEIRA](#).



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

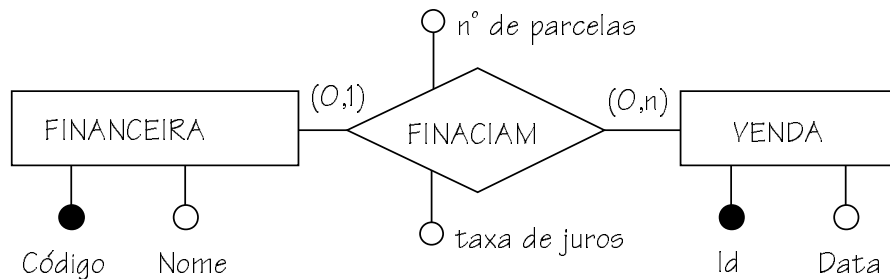


Figura 5.13: Tradução de relacionamentos 1:n no qual a entidade com cardinalidade máxima um é opcional

Para este tipo de relacionamento, pode ser usada alternativamente 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)

Fianciam (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:

- ❑ 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 **Fianciam** possuem a mesma chave primária, sendo o conjunto de valores de **Fianciam** 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 à prazo e vazios em caso contrário.

#### 5.2.3.3 Relacionamentos n:n

Independentemente da cardinalidade mínima, relacionamentos n:n, são sempre implementados através de uma 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 *multi-valorada*, 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 *mono-valoradas*. Assim, esta alternativa não é viável pelas próprias características da abordagem relacional.

#### 5.2.3.4 Relacionamentos de grau maior que dois

As regras apresentadas até este ponto, aplicam-se somente à implementação de relacionamentos binários, isto é, que envolvem apenas duas entidades. Para relacionamentos de grau maior que dois, não são definidas regras específicas. A implementação de um relacionamento de grau maior que dois dá-se na seguinte seqüência de passos:

- 1 O relacionamento é transformado em uma entidade. Esta nova entidade é ligado através de um relacionamento binário a cada uma das entidades que participavam do relacionamentos 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 5.14 mostra um relacionamento ternário junto com sua transformação em uma entidade e três relacionamentos binários.

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,Nome)

CodProd referencia Produto

CodDistr referencia Distribuidor

CodCid referencia Cidade

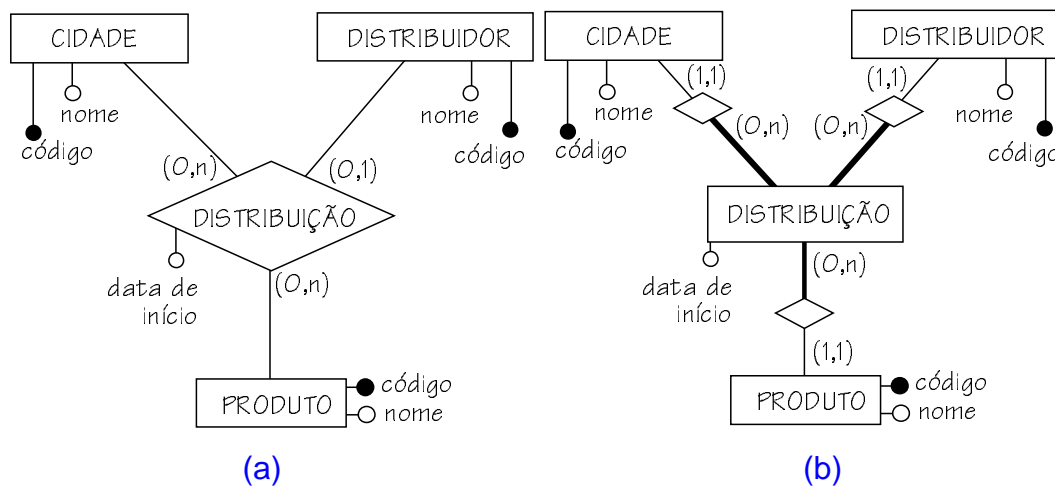


Figura 5.14: Relacionamento ternário (a) e sua transformação em entidade e relacionamentos binários (b)

## 5.2.4 Implementação de generalização/especialização

Para a implementação de hierarquias de generalização/especialização na abordagem relacional, há duas alternativas 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 uma ou outra.

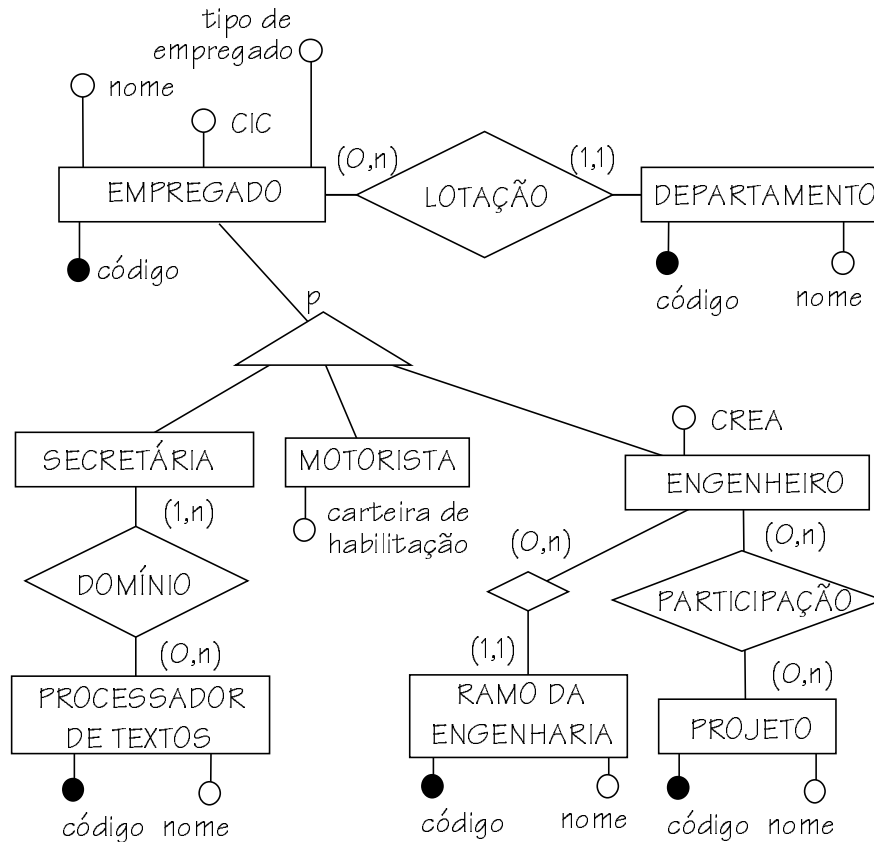
### 5.2.4.1 Uma tabela por hierarquia

Nesta alternativa, todas tabelas referentes às especializações de uma entidade genérica são fundidas em uma única tabela. Esta tabela terá:

- ☐ 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 nenhuma coluna na implementação. Isto ocorrerá caso a entidade especializada não tenha atributos e caso todos relacionamentos dos quais ele 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 5.15.



Esquema relacional correspondente:

**Emp** (CódigoEmp, Tipo, Nome, CIC, 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 ProcessTexto

**Projeto** (CódigoProj, Nome)

**Participação** (CódigoEmp, CódigoProj)

CódigoEmp referencia Emp

CódigoProj referencia Projeto

Figura 5.15: Hierarquia de generalização/especialização e sua implementação através de tabela única

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 **CIC** referentes aos atributos da entidade genérica
- ❑ A coluna **CódigoDept**, que implementa o relacionamento **LOTAÇÃO**
- ❑ A coluna **CarHabil** que implementa os atributos da entidades 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 nenhuma coluna já que não possui atributos nem participa de relacionamentos que gerem colunas.

As colunas que correspondem às entidades especializadas (**CarHabil**, **CREA** e **CódigoRamo**) devem ser definidas todas como colunas *opcionais*. Essa definição é necessária, pois uma linha referente a um empregado, que não pertença a nenhuma das classes especializadas, terá todos campos acima listados 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 **CarHabil** vazio.

As demais tabelas que implementam o modelo da figura 6.14, definidas usando as regras apresentadas nas seções anteriores são:

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

#### 5.2.4.2 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 5.15 resultaria no seguinte esquema relacional (parte referente à hierarquia de generalização/especialização em **negrito**):

**Emp** (CódigoEmp,Tipo,Nome,CIC,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 Emp

CódigoProj referencia Projeto

Para a entidade **EMPREGADO** e cada uma de suas especializações, foi criada uma tabela. Estas tabelas tem 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 destas tabelas 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.

#### **5.2.4.3 Comparação entre as duas alternativas de implementação**

A seguir são discutidas as vantagens de cada tipo de implementação em relação a 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 os prós e contras de ambas soluções e optar por aquela que, considerando os fatores acima, seja a mais adequada ao seu problema.

#### 5.2.4.3.1 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 5.15 a implementação é a apresentada abaixo (parte referente à hierarquia de generalização/especialização em negrito):

**EmpOutros** (CódigoEmp,Tipo,Nome,CIC,CódigoDept)

**CódigoDept referencia Depto**

**Motorista**(CódigoEmp, Nome,CIC,CódigoDept,CarHabil)

**Engenheiro**(CódigoEmp, Nome,CIC,CódigoDept,  
**CREA,CódigoRamo**)

**CódigoRamo referencia Ramo**

Depto (CódigoDept, Nome)

Ramo (CódigoRamo,Nome)

ProcessTexto (CódigoProc,Nome)

Domínio (CódigoEmp,CódigoProc)

Código Proc referencia ProcessTexto

Projeto (CódigoProj,Nome)

Participação (CódigoEmp,CódigoProj)

CódigoProj referencia Projeto

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**, **CIC** e **CódigoDept**). De forma análoga, a tabela **Engenheiro** contém não só os atributos específicos de engenheiro, mas também os de pessoa. Finalmente, a tabela **EmpOutros** contém dados de todas as demais categorias de empregados.

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 empregados estão reunidos, como ocorria na alternativa anterior. Além disso, quando a especialização for total (e não parcial como no caso de exemplo), deixa de existir a tabela que coleciona as linhas referentes à entidades para as quais não há especialização (no caso do 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 sobrepuja as vantagens que oferece do ponto de vista do 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 SGBD relacionais não são capazes de realizá-la automaticamente. Pior ainda, 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). Exemplificando, as restrições de integridade referenciais que apareciam nas soluções anteriores, e que especificam que as colunas [CódigoEmp](#) que aparecem nas tabelas [Domínio](#) e [Participação](#) são chaves estrangeiras em relação ao conjunto de todos empregados, não podem ser especificadas.

### 5.2.5 Refinamento do modelo relacional

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 das regras acima não atende plenamente os requisitos de performance impostos ao sistema. Neste caso, é necessário buscar uma alternativa de implementação que resulte em melhor performance do sistema. Cabe salientar, que estas alternativas somente deveriam ser tentadas em último caso, pois, do ponto de vista do desenvolvimento de programas sobre o banco de dados, são sempre piores que as alternativas que haviam sido apresentadas anteriormente. Nas seções abaixo, apresentamos alguns exemplos de alternativas de implementação que podem ser adotadas.

#### 5.2.5.1 Relacionamentos mutuamente exclusivos

Um caso que permite uma implementação alternativa à especificada pelas regras acima é aquele no qual uma entidade participa de forma *mutuamente exclusiva* de dois ou mais relacionamentos. Participar de forma mutuamente ex-

clusiva significa que uma ocorrência da entidade que participa de um dos relacionamentos em questão, não participa de nenhum dos demais relacionamentos. Um exemplo é apresentado na Figura 5.16. Pode-se supor que uma ocorrência da entidade **VENDA** participe de exatamente um dos dois relacionamentos e de somente de um deles. Observe que esta informação não está contida no diagrama, já que não há uma convenção para registrá-la no DER<sup>7</sup>. Ela deveria estar registrada em alguma documentação paralela.

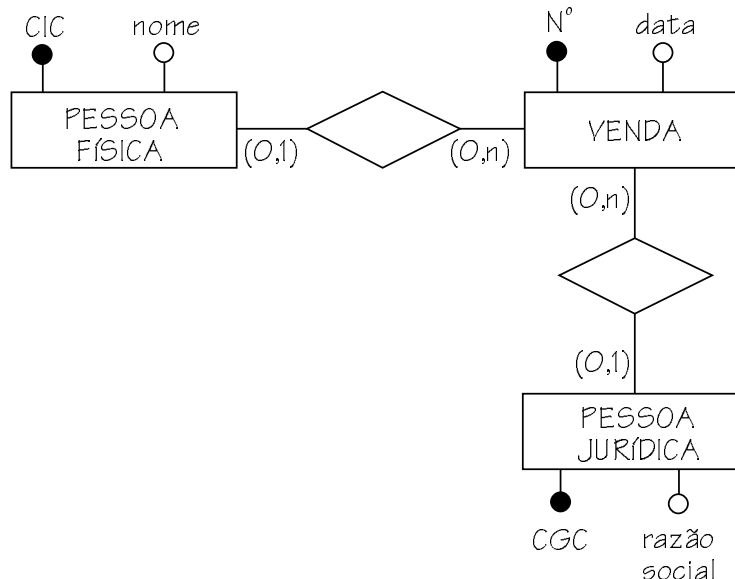


Figura 5.16: Relacionamentos mutuamente exclusivos

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

**PessFis**(CIC, Nome)

**PessJur**(CGC, RazSoc)

**Venda**(No, data, CIC, CGC)

CIC referencia PessFis

CGC referencia PessJur

Nesta implementação, as colunas **CIC** e **CGC** são especificadas como opcionais, já que em cada linha um ou outro campos serão vazios. Aparecem assim os problemas típicos de colunas opcionais. Uma implementação alternativa é criar uma única coluna na qual aparece o **CIC** ou o **CGC** do comprador, conforme mostrado abaixo.

**PessFis**(CIC, Nome)

**PessJur**(CGC, RazSoc)

**Venda**(No, data, CIC/CGC, TipoCompr)

Além da fusão das colunas **CIC** e **CGC**, deve ser criada uma coluna **TipoCompr**, que informa se o campo na coluna **CIC/CGC** é referente a um comprador pessoa física ou a uma pessoa jurídica. Através desta alternativa

<sup>7</sup>Algumas notações de diagramas ER prevêm notações específicas para a mútua exclusão entre relacionamentos.

evita-se o uso de colunas opcionais. Entretanto, a alternativa apresenta igualmente uma desvantagem. Nesta alternativa, não é possível especificar ao SGBD que o campo **CIC/CGC** é chave estrangeira, já ele não referencia uma única tabela, mas duas (**PessFis** e **PessJur**), de forma alternativa de acordo com o valor do campo **TipoCompr**.

#### 5.2.5.2 Simulação de atributos multi-valorados

Conforme discutimos no capítulo 4, atributos multi-valorados não são desejáveis em diagramas ER, já que não possuem implementação direta na abordagem relacional. A Figura 5.17 apresenta um diagrama ER com atributos multi-valorados e o diagrama obtido pela transformação do atributo multi-valorado em uma entidade separada.

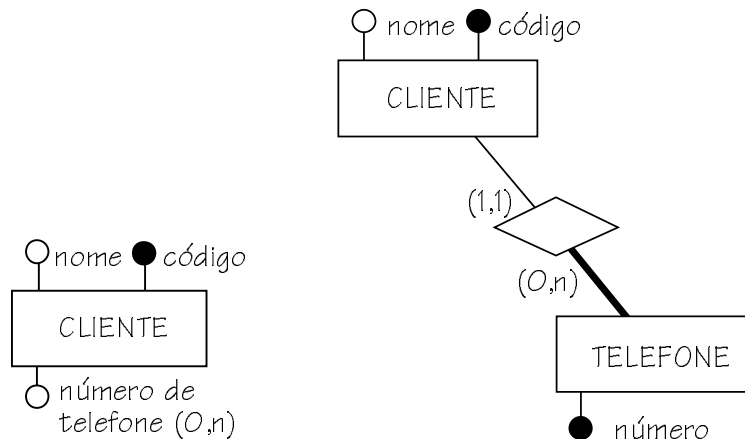


Figura 5.17: Eliminando atributos multi-valorados

A implementação do diagrama da Figura 5.17, caso sejam seguidas as regras apresentadas, é a seguinte:

**Cliente** (CodCli,Nome)

**Telefone** (CodCli,Número)

CodCli referencia Cliente

Entretanto, esta implementação pode trazer problemas de performance, conforme discutido abaixo. Consideremos as seguintes condições de contorno:

- ❑ São raros os clientes que possuem mais que dois telefones. Quando isso ocorrer, é suficiente armazenarmos 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 juntos às demais informações de cliente.

Neste caso, uma implementação “desnormalizada”<sup>8</sup> como a mostrada abaixo pode permitir maior performance:

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

Nesta implementação, optou-se por simular uma coluna multi-valorada através da criação de diversas colunas **NumTel** sufixadas por um número para

---

<sup>8</sup>No próximo capítulo veremos o conceito de *normalização*.

distinguí-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. O inconveniente que esta alternativa apresenta do ponto de vista funcional é que 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 multi-valorado

### 5.2.5.3 Informações redundantes

Como vimos na Seção 3.3.3, informações redundantes, que podem ser obtidas a partir de outras existentes no banco de dados, devem ser eliminadas do modelo conceitual. Entretanto, por questões de performance, muitas vezes, quando do projeto do banco de dados, informações redundantes são reinseridas no esquema. Isso ocorre muito freqüentemente com atributos que resultam de uma operação que envolve diversas entidades do banco de dados. Caso o valor destes atributos tenha que ser obtido com freqüência ou sirva freqüentemente como critério de busca de informações no banco de dados, pode ser mais eficiente, do ponto de vista da performance global do sistema, armazenar o atributo derivado redundantemente.

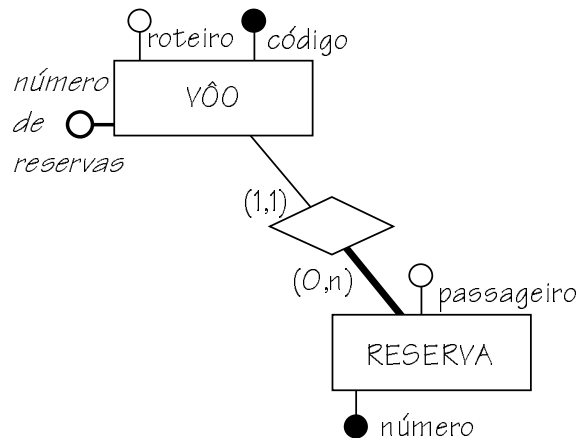


Figura 5.18: Atributo redundante

A Figura 5.18 apresenta um exemplo no qual aparece um atributo redundante. Trata-se do atributo **número de reservas**, que pode ser obtido pela contagem de todas as reservas relacionadas ao voo. Assim, do ponto de vista conceitual, o atributo **número de reservas** deveria ser eliminado. Entretanto, do ponto de vista de performance, provavelmente seria importante manter uma coluna com este valor, visto que ele seria necessário em um grande número de buscas no banco de dados e sua computação freqüente demandaria tempo excessivo.

## 5.3 ENGENHARIA REVERSA DE MODELOS RELACIONAIS

De forma geral, um processo de *engenharia reversa* pode ser definido como é um processo de abstração, que parte de um modelo de implementação e re-



sulta 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 transforma-se modelos de banco de dados mais ricos em detalhes de implementação 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.

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 desenvolvida 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.

Conforme 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.

"engenharia reversa:de modelos relacionais:processo"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 abaixo.

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



### 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 a nível de modelo ER.

Uma tabela pode corresponder a:

- ☐ uma entidade
- ☐ um relacionamento n:n
- ☐ uma entidade especializada

O fator determinante da 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 a tabela **Curric** que tem como chave primária **CodCr** e **CodDisc**. Ambas 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: Toda chave primária é uma chave estrangeira*

A tabela cuja chave primária é toda ela uma chave estrangeira 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,CodSi**), 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. A nível de modelo ER, isso significa que uma ocorrência da entidade laboratório somente pode existir quando a correspondente ocorrência da entidade sala existe, 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 acima), nem for toda uma chave estrangeira (regra 2 acima), 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 **CodSi**) contém apenas uma chave estrangeira (coluna **CodSi**). Assim, não obedece ao requisito da multiplicidade de chaves estrangeiras (regra 1), nem o 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 feita 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 5.19.

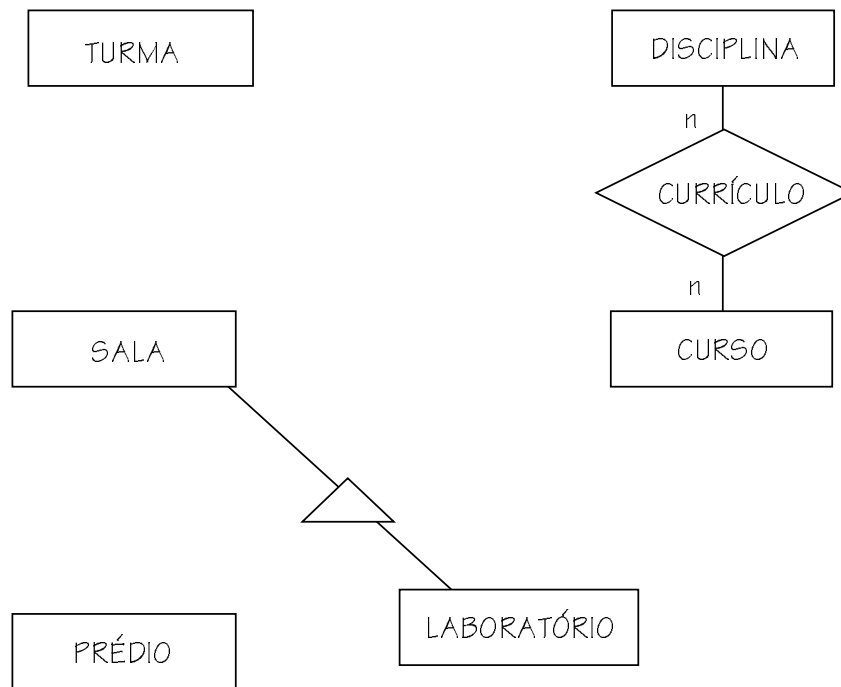


Figura 5.19: Diagrama ER inicial para o BD acadêmico

### 5.3.2 Identificação de relacionamentos 1:n ou 1:1

Toda chave estrangeira que não se enquadra nas regras 1 e 2 acima, ou seja, toda chave estrangeira que não faz parte de uma chave primária composta por múltiplas chaves estrangeiras, nem é toda ela uma chave primária, representa um relacionamento 1:n ou 1:1. Em outros termos, toda chave estrangeira que não corresponde a um relacionamento n:n, nem a uma entidade especializada representa um relacionamento 1:n ou 1:1. A regra não permite definir se a cardinalidade do relacionamento é 1:n ou 1:1. Para definir qual dos dois tipos de relacionamentos está sendo representado pela chave estrangeira, é necessário verificar os possíveis conteúdos do banco de dados. No caso do 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,CodSI) referencia Sala

Com isso podemos completar a definição dos relacionamentos no diagrama ER, conforme mostra a Figura 5.20. No caso do exemplo, todos relacionamentos referentes as chaves estrangeiras acima são do tipo 1:n.

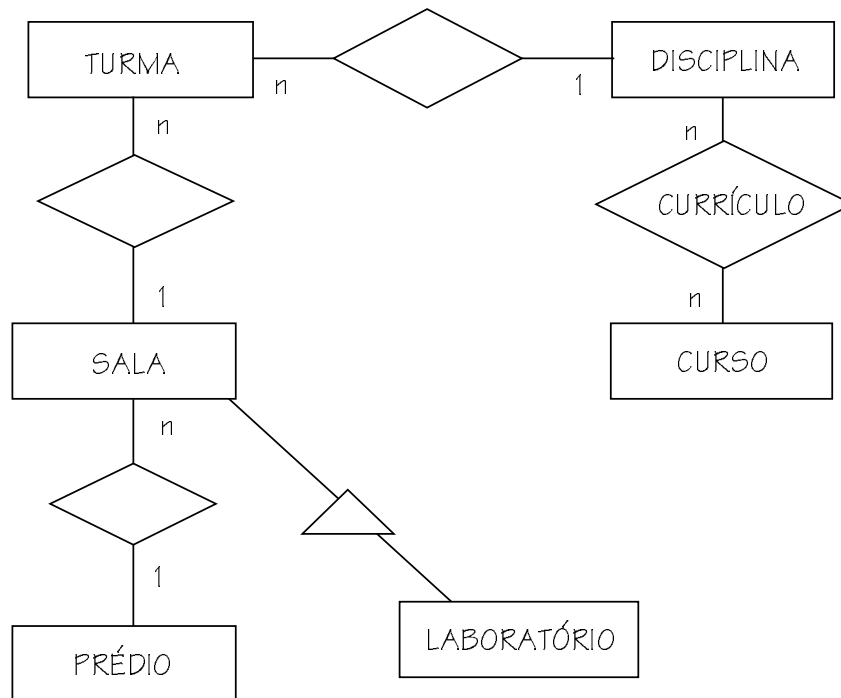


Figura 5.20: Definição dos relacionamentos 1:n e 1:1

### 5.3.3 Definição de atributos

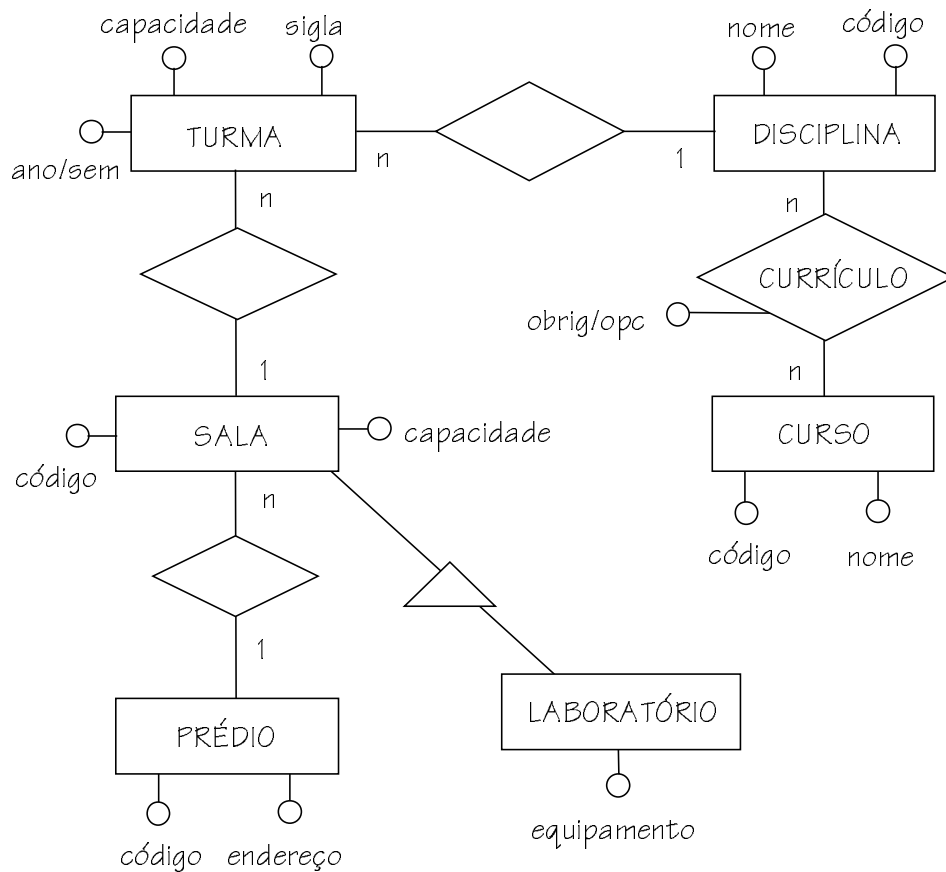


Figura 5.21: Definição dos atributos

Nesta etapa, para cada coluna de uma tabela, que não seja chave estrangeira, é definido um atributo na entidade/relacionamento correspondente à tabela. Observe-se que colunas chave estrangeira não correspondem a atributos no diagrama ER, mas sim a relacionamentos, e por isso já foram tratadas nas etapas anteriores. Para o caso do exemplo, a execução deste passo da engenharia reversa resulta no diagrama ER da Figura 5.21.

### 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 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 tam-

bém chave estrangeira em relação a 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 5.22.

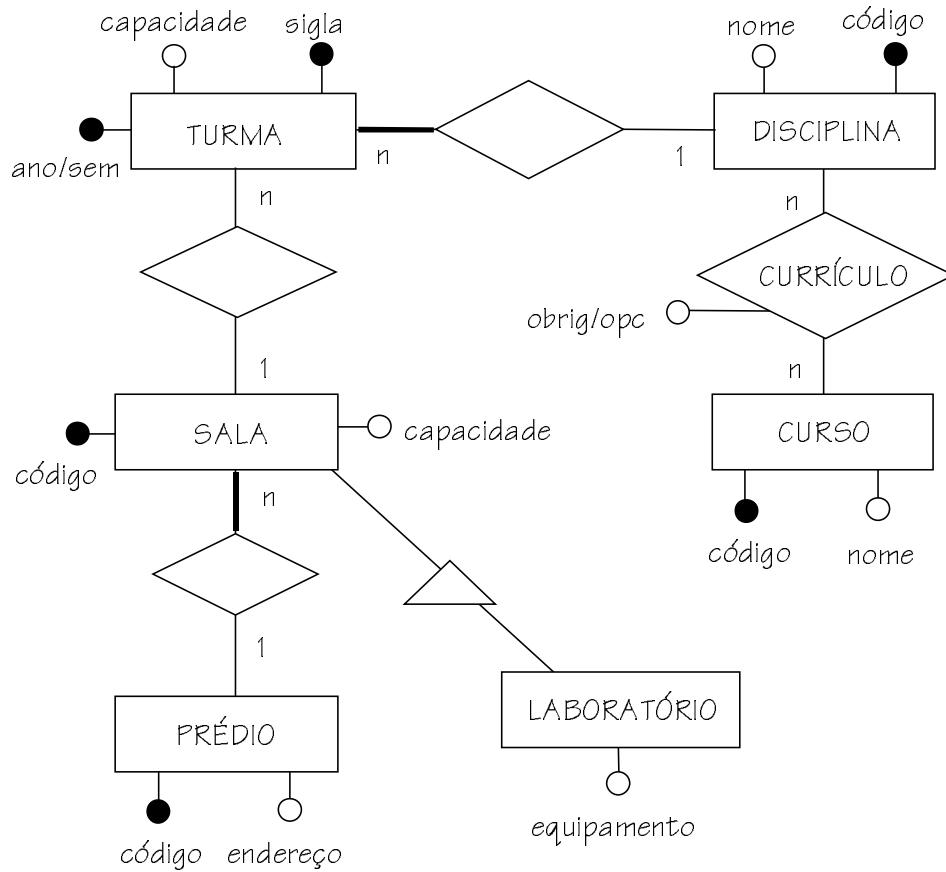


Figura 5.22: Definição dos identificadores de entidades

## EXERCÍCIOS

**Exercício 5.1:** 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 casos está sendo representado um conjunto de alunos e informações (código, nome, código de curso, endereço) a ele referentes. Discuta à luz dos princípios que baseiam as regras de tradução de diagramas ER para modelo relacional, qual das duas alternativas é preferível.

**Exercício 5.2:** 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 5.23. Para não sobrecarregar o diagrama os atributos das entidades são listados abaixo. Os atributos identificadores estão sublinhados.

Produto (Número, NomeComercial, TipoEmbalagem, Quantidade, PreçoUnitário)

Fabricante (CGC, Nome, Endereço)

Medicamento(Tarja,Fórmula)

Perfumaria(Tipo)

Venda(Data,NúmeroNota, NomeCliente, CidadeCliente)

PerfumariaVenda(Quantidade, Imposto)

MedicamentoReceitaVenda(Quantidade, Imposto)

ReceitaMédica(CRM, Número, Data)

**Exercício 5.3:** 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 5.24. Para não sobrecarregar o diagrama os atributos das entidades são listados abaixo. 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)

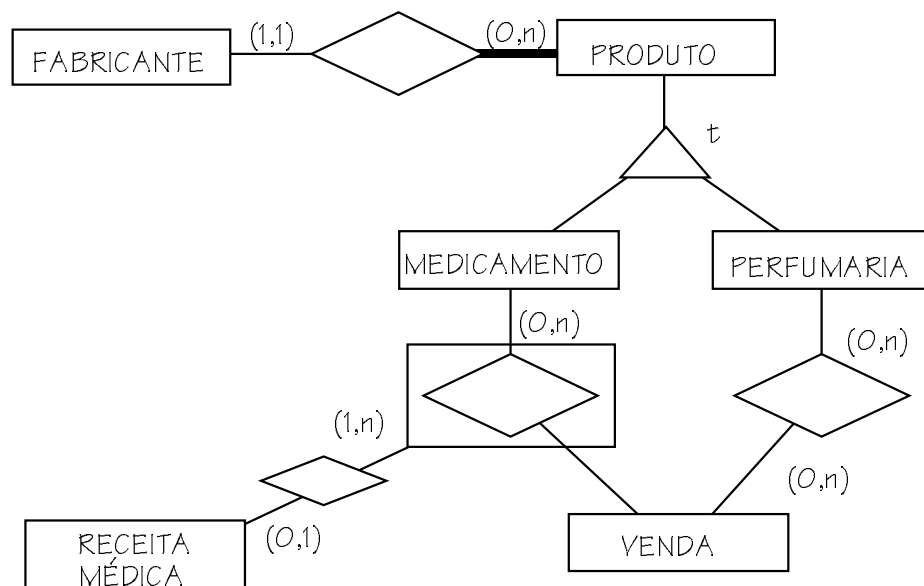


Figura 5.23: Modelo ER para uma farmácia

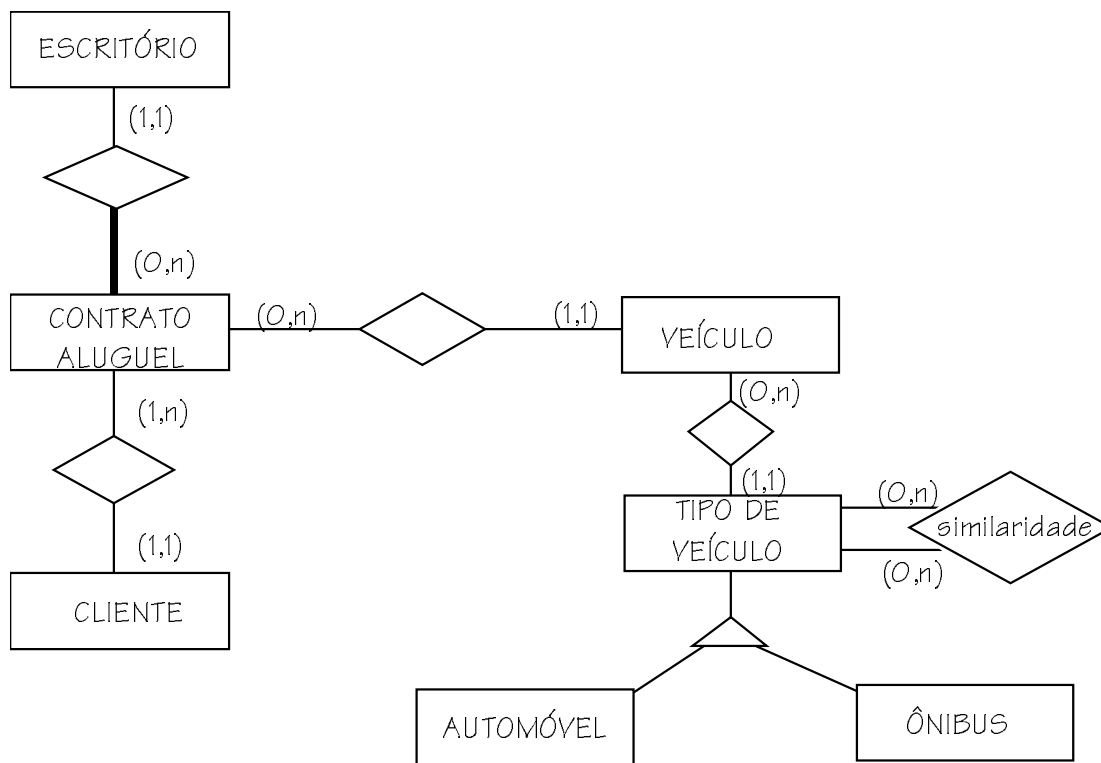


Figura 5.24: Modelo ER para locadora de veículos

**Exercício 5.4:** Abaixo é apresentado um esquema lógico de um BD relacional que armazena dados sobre produtos em uma indústria. Usando as regras de engenharia reversa apresentadas acima, 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

(CodEmo) 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,  
QtdItem,PreçoItem)

(NúmeroNF) referencia Venda

(CodigoTipoProd,NumeroProd) referencia Produto

/\* tabela com informa os 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 5.5:** Abaixo é apresentado um esquema lógico de um BD relacional que armazena dados genealógicos. Usando as regras de engenharia reversa apresentadas acima, 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) \*/

## REFERÊNCIAS BIBLIOGRÁFICAS

Muitos dos livros que tratam da abordagem relacional dedicam um capítulo ao projeto de banco de dados relacional a partir do diagrama ER. Este é o caso



dos livros [3,5] que já havíamos citado no capítulo referente a abordagem relacional.

Da mesma forma, livros sobre modelagem e projeto de banco de dados [1,7,10] abordam o problema do projeto lógico. O livro de Batini, Ceri e Navathe [1] contém uma cobertura completa tanto do projeto do banco de dados, quanto da engenharia reversa a partir de modelos relacionais.

Os artigos [2, 8,11] são os pioneiros no projeto lógico de banco de dados relacionais a partir de modelos ER. Já os artigos [4,6] apresentam as idéias básicas sobre como deve ser feita a engenharia reversa de modelos relacionais para modelos ER.

- [1] Batini, C. & Ceri, S. & Navathe, S.B. *Conceptual Database Design: an Entity-Relationship Approach*. Benjamin/Cummings, Redwood City, California, 1992
- [2] Dumpala, S.R. & Arora, K. Schema Translation Using the Entity-Relationship Approach. In: Davis, C.et al. (editors): *Entity-Relationship Approach to Software Engineering*, (Proceedings of the Third International Conference on Entity Relationship Approach, Anaheim, California, October 1983), North-Holland, 1983
- [3] Elmasri, R. & Navathe, S.B. *Fundamentals of Database Systems*. Second Edition. Benjamin/Cummings, Redwood City, California, 1994
- [4] Johannesson, P. & Kalman, K. A Method for Translating Relational Schemas into Conceptual Schemas. in: Lochovsky, F.H. (ed.) *Entity Relationship Approach to Database Design an Querying* Elsevier, 1990, 271-285.
- [5] Korth, H. & Silberschatz, A. *Sistemas de Bancos de Dados*. 2ª edição, Makron Books, 1994
- [6] Navathe, S.B. & Awong, A.M. Abstracting Relational and Hierarchical Data with Semantic Data Model. In: March, S. (ed.) *Entity Relationship Approach: a Bridge to the User*. North-Holland, 1988
- [7] Setzer, V.W. *Banco de Dados*. Segunda edição. Ed. Edgard Blücher, 1987
- [8] Teorey, T. Yang, D. & Fry, J. A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, *ACM Computing Surveys*, 18:2, July, 1986.
- [10] Teorey, T.J. *Database Modeling & Design - The Fundamental Principles*. Second Edition. Morgan Kaufmann, San Francisco, CA, 1994
- [11] Wong, E. & Katz, R. Logical Design and Schema Conversion for Relational and DBTG Databases. In: Chen, P. (ed.) *Entity-Relationship Approach to Systems Analysis and Design*. North-Holland, pp. 311-322,1980

## 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, será vista a *engenharia reversa de arquivos*. Este processo permite obter um modelo lógico relacional a partir do modelo lógico de um banco de dados não relacional. Ele permite que se execute 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 objetiva eliminar redundâncias de dados de arquivos.

Ao final do capítulo, são apresentados vários exercícios que compõem dois estudos de caso de engenharia reversa de dois sistemas, um de preparação de congressos e outro de controle de um almoxarifado.

## 6.1 INTRODUÇÃO

Um percentual significativo dos sistemas de informação hoje usados foram desenvolvidos ao longo dos últimos vinte anos e não utilizam bancos de dados relacionais. São os chamados *sistemas legados* (“*legacy systems*”). Os dados destes sistemas estão armazenados em arquivos de linguagens de terceira geração, como Basic, COBOL, MUMPS e outras, ou então em bancos de dados da era pré-relacional, como IMS, ADABAS, DMS-II e os SGBD do tipo CODASYL (IDMS, IDS/2,...). Raramente, os arquivos destes sistemas estão documentados através de modelos conceituais. Além disso, muitos dos bancos de dados relacionais existentes, principalmente em micro-computadores, não possuem documentação na forma de modelo conceitual.

No entanto, há situações no ciclo de vida de um sistema nas quais um modelo conceitual pode ser de grande valia.

Um exemplo é a manutenção rotineira de 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 implementada 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 em muito o processo de migração, pois permite encurtar a etapa de modelagem de dados da 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.

## 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.

O primeiro passo é a representação da descrição de cada arquivo existente na forma de um esquema de uma tabela relacional não-normalizada. Este primeiro passo objetiva obter 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.

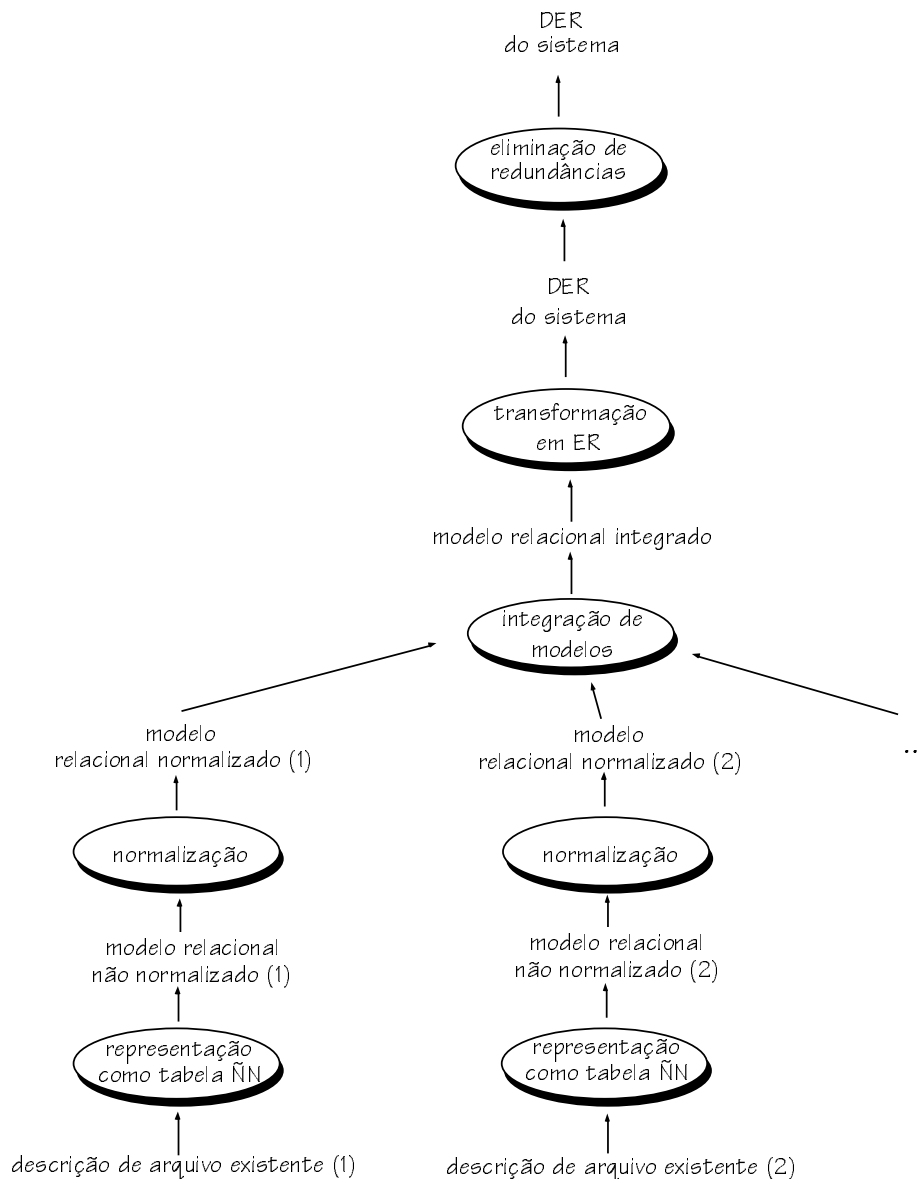


Figura 6.1: Visão geral do processo de engenharia reversa

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.
- ❑ Reagrupar informações de uma forma que permita a obtenção de um modelo ER.

Uma vez normalizados todos 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 assim obtido, usando as regras para engenharia reversa de um modelo relacional mostradas no capítulo anterior, é possível obter 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 que é descrito abaixo 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 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

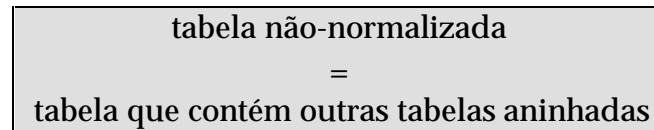
CÓDIGO DO EMPREGADO	NOME	CATEGORIA FUNCIONAL	SALÁRIO	DATA DE INÍCIO NO PROJETO	TEMPO ALOCADO AO PROJETO
8191	Mário	A1	4	1/05/93	12
4112	João	A2	4	4/01/91	24
6126	José	B1	9	1/11/92	12

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 precisa-

mente *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 multi-valorada* ou ainda *coluna não atômica*) é uma coluna que ao invés de conter valores atômicos, contém tabelas aninhadas.



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 linguagens como Pascal ou de item de grupo repetido (especificado com cláusula “OCCURS”) da linguagem COBOL.

CódProj	Tipo	Descr	Emp					
			CodEmp	Nome	Cat	Sal	Datalni	TempAl
LSC001	Novo Desenv.	Sistema de Estoque	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
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

A tabela da Figura 6.3 pode ser descrita pelo seguinte esquema de tabela relacional:

Proj (CodProj, Tipo, Descr,  
(CodEmp, Nome, Cat, Sal, Datalni, 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 um outro exemplo de representação não normalizada de documentos, mostramos na Figura 6.4 e na Figura 6.5 a definição de um arquivo que contém dados de alunos usando a notação de Pascal e de COBOL respectivamente.

```

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)

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 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 as 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. No caso deste trabalho, vamos considerar quatro formas normais. As formas normais são denominadas simplesmente primeira, segunda, terceira e quarta forma normal, abreviadamente 1FN, 2FN, 3FN e 4FN

### 6.5.1 Passagem à primeira forma normal (1FN)

O próximo passo da normalização consta da transformação do esquema de tabela não normalizada em um esquema relacional na primeira forma normal (1FN). Uma tabela encontra-se na 1FN quando não contém tabelas aninhadas. Portanto, a passagem à 1FN consta da eliminação das tabelas aninhadas eventualmente existentes.

primeira forma normal (1FN)
=
diz-se que uma tabela está na primeira forma normal, quando ela não contém tabelas aninhadas

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 7.4, 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. A motivação é de ordem prática. 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 6.16**.

Na decomposição de tabelas, a passagem à primeira forma normal por decomposição de tabelas é feita nos seguintes passos:

- 1 É criada uma tabela na 1FN referente a 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 a 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 na qual 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, no caso da tabela 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 a tabela ProjEmp. Esta tabela refere-se a 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 a 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 um mesmo empregado e é necessário usar o código do projeto para distinguir entre elas.

Proj:

CódProj	Tipo	Descr
LSC001	Novo Desenv.	Sistema de Estoque
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

Cabe observar que, caso cada empregado trabalhasse em um único projeto apenas, 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 caso do 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.

Um outro exemplo de passagem a primeira forma normal é o do arquivo de alunos visto acima (Figura 6.4 e Figura 6.5). Conforme mostrado acima, 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-AI, Nome-AI)
AlunoCurso	(Cod-AI, Cod-Curso, Sem-ingresso)
AlunoDisc	(Cod-AI, Cod-Disc)
AlunoDiscSem	(Cod-AI, Cod-Disc, Sem-Disc-Cursada, Nota-Disc)

A primeira tabela corresponde a tabela  $\tilde{N}N$  sem suas tabelas aninhadas. Já a tabela **AlunoCurso** na 1FN corresponde a primeira tabela aninhada na forma  $\tilde{N}N$ . Observe que esta tabela contém, além das colunas **CodCurso** e **Sem-ingresso**, também a coluna **Cod-AI**, chave primária da tabela na qual a tabela aninhada encontra-se na forma  $\tilde{N}N$ .

Pelo mesmo motivo, a tabela **AlunoDiscSem**, que corresponde a terceira tabela aninhada, contém as colunas **Cod-AI** e **Cod-Disc**. Estas são as chaves primárias das tabelas nas quais a tabela aninhada em questão encontra-se na forma  $\tilde{N}N$ .

As chaves primárias das tabelas na 1FN são as seguintes:

## 1FN

Alunos	( <u>Cod-AI</u> , Nome-AI)
AlunoCurso	( <u>Cod-AI</u> , <u>Cod-Curso</u> , Sem-ingresso)
AlunoDisc	( <u>Cod-AI</u> , <u>Cod-Disc</u> )
AlunoDiscSem	( <u>Cod-AI</u> , <u>Cod-Disc</u> , <u>Sem-Disc-Cursada</u> , Nota-Disc)

Conforme já foi mencionado acima, a chave primária de uma tabela na 1FN não necessariamente é a concatenação das chaves primárias das colunas chaves primárias na forma  $\tilde{N}N$ . Abaixo apresentamos um exemplo em que, ao contrário dos exemplos até agora mostrados, a chave primária da tabela na 1FN não é a concatenação das chaves das tabelas na forma  $\tilde{N}N$ . Considere a tabela não normalizada apresentada abaixo.

## $\tilde{N}N$

Arq-Candidatos	( <u>Cod-Curso</u> , Nome-Curso, Numero-Vagas-Curso, ( <u>Cod-Cand</u> , 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 no curso. Supõe-se que cada candidato tenha sido aprovado em um curso somente.

A passagem à 1FN gera as tabelas abaixo:

## 1FN

Cursos	( <u>Cod-Curso</u> , Nome-Curso, Numero-Vagas-Curso)
Candidatos	(Cod-Curso, <u>Cod-Cand</u> , Nome-Cand, Escore-Cand)

Observe que na segunda tabela, correspondente à tabela aninhada da forma  $\tilde{N}N$ , a chave primária é apenas a coluna **Cod-Cand** (código do candidato), já que um candidato pode aparecer uma vez somente no arquivo.

De forma geral, para determinar a chave primária de uma tabela na 1FN que corresponda uma tabela aninhada na forma  $\tilde{N}N$  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 que são necessárias para identificar as linha 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*.

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 linhas da tabela, para cada valor de  $C_1$  que aparece na tabela, aparece o mesmo valor de  $C_2$ .

O conceito fica mais fácil de se entender, se considerarmos um exemplo.

...	Código	...	Salário	...
	E1		10	
	E3		10	
	E1		10	
	E2		5	
	E3		10	
	E2		5	
	E1		10	

Figura 6.7: Extrato de tabela com dependência funcionais

A tabela da Figura 6.1 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 está dependência funcional, usa-se uma expressão na forma **Código**  $\rightarrow$  **Salário**. A expressão denota que a coluna **Salário** depende funcionalmente da coluna **Código**. Diz-se 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. Exemplificando, na tabela da Figura 6.8, a coluna **C** depende das colunas **A** e **B**.

A	B	C	D
B	5	2	20
C	4	2	15
B	6	7	20
B	5	2	20
C	2	2	15
C	4	2	15
A	10	5	18
A	12	3	18
A	10	5	18
B	5	2	20
C	4	2	15
A	10	5	18
C	4	2	15

Dependências funcionais:

$(A,B) \rightarrow C$

$A \rightarrow D$

Figura 6.8: Tabela com exemplos de dependências funcionais

### 6.5.3 Passagem à segunda forma normal (2FN)

A passagem à segunda forma normal (2FN) objetiva eliminar um certo tipo de redundância de dados. Para exemplificar, tomamos a tabela [ProjEmp](#) da Figura 6.6. Nesta tabela, os dados referentes a empregados ([Nome](#), [Cat](#) e [Sal](#)) estão 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 objetiva eliminar 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 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

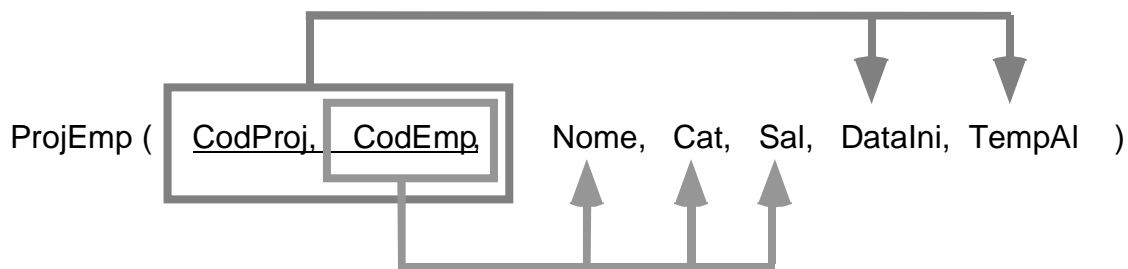
Obviamente, uma tabela que está na 1FN e que possui apenas uma coluna como chave primária não contém dependências parciais, já que nesta tabela é impossível uma coluna depender de uma parte da chave primária, visto que a chave primária não é composta por partes (por diversas colunas). Assim, toda tabela 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 caso do documento exemplo, a tabela **Proj** encontra-se na 2FN por possuir uma chave primária simples (composta de 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 **NumEmp**, já que nome, categoria funcional e salário são determinados tão somente pelo número 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).

Tabela na 1FN e dependências funcionais



Tabelas na 2FN e dependências funcionais

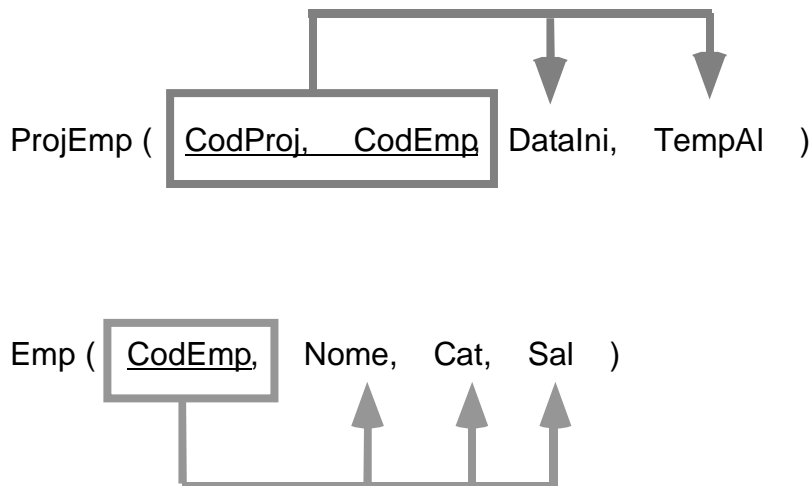


Figura 6.9: Passagem à 2FN

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:

**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, na 2FN é o seguinte.

## 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 a 2FN é o seguinte.

- ❑ Copiar para a 2FN cada tabela que tenha chave primária simples ou que não tenha colunas além chave. No caso do exemplo, é o que acontece com a tabela **Proj**.
- ❑ Para cada tabela com chave primária composta e com pelo menos uma coluna não chave (no exemplo, a tabela **ProjEmp**):
  - Criar na 2FN uma tabela com as chaves primárias da tabela na 1FN (no exemplo, trata-se da tabela **ProjEmp** na 2FN)
  - 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) ⇨
    - ◇ 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).
    - ◇ Criar a coluna dependente dentro da tabela na 2FN (as colunas **Nome**, **Sal** e **Cat** da tabela **Emp** na 2FN).

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	Sílvio	A2	4
6126	José	B1	9
1214	Carlos	A2	4
8191	Mário	A1	4
8191	Mário	A1	4
4112	João	A2	4
6126	José	B1	9

Figura 6.10: Tabelas referentes ao exemplo na 2FN

#### 6.5.4 Passagem à terceira forma normal (3FN)

Na passagem à terceira forma normal, elimina-se um 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 que salário é pago a uma categoria funcional está representado redundantemente na tabela, tantas vezes quantos empregados possui a categoria funcional. A passagem à 3FN objetiva eliminar este tipo de redundância de dados.

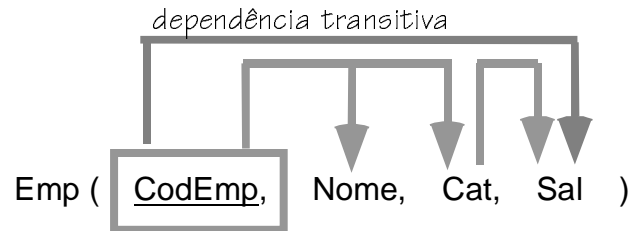
Uma tabela encontra-se na 3FN quando, além de estar na 2FN, toda coluna não chave depende *diretamente* de chave primária, isto é, quando não há dependências funcionais *transitivas* ou *indiretas*.



terceira forma normal (3FN)
=
uma tabela encontra-se na terceira forma normal, quando, além de estar na 2FN, não contém dependências transitivas

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. A passagem à 3FN consta em dividir tabelas de forma a eliminar as dependências transitivas. O processo de passagem à 3FN para o exemplo é mostrado na Figura 6.11.

Tabela na segunda forma normal (2FN)



Tabelas na terceira forma normal (3FN)

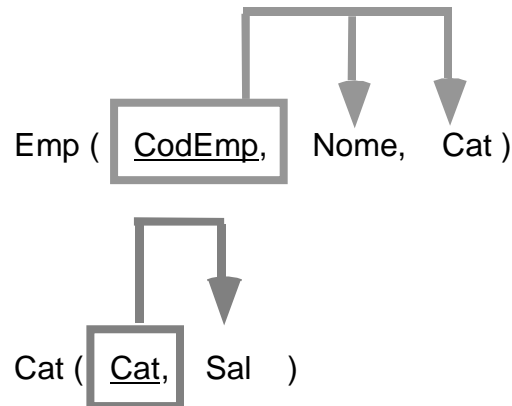


Figura 6.11: Passagem à 3FN para o exemplo

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 )

As tabelas referentes a terceira forma normal do documento exemplo têm seu conteúdo mostrado na Figura 6.12.

Proj:

CódProj	Tipo	Descr
LSC001	Novo Desenv.	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

ProjEmp:

CódProj	NúmEmp	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:

NúmEmp	Nome	Cat
2146	João	A1
3145	Sílvio	A2
6126	José	B1
1214	Carlos	A2
8191	Mário	A1
8191	Mário	A1
4112	João	A2
6126	José	B1

Cat:

Cat	Sal
A1	4
A2	4
B1	9

Figura 6.12: Tabelas referentes ao exemplo na 3FN

De forma mais precisa, o processo de passagem da 2FN a 3FN é o seguinte:

- ❑ 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.
- ❑ 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 dependa apenas da chave ⇨

- \* Copiar a coluna para a tabela na 3FN

Caso a coluna depender de outra coluna ⇨

- \* Criar, caso ainda não exista, uma tabela no esquema na 3FN que tenha como chave primária a coluna da qual há a dependência indireta.
- \* Copiar a coluna dependente para a tabela criada.
- \* A coluna determinante deve permanecer também na tabela original.

### 6.5.5 Passagem à quarta forma normal

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 4FN 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 que empregado usa que equipamento em que projeto.

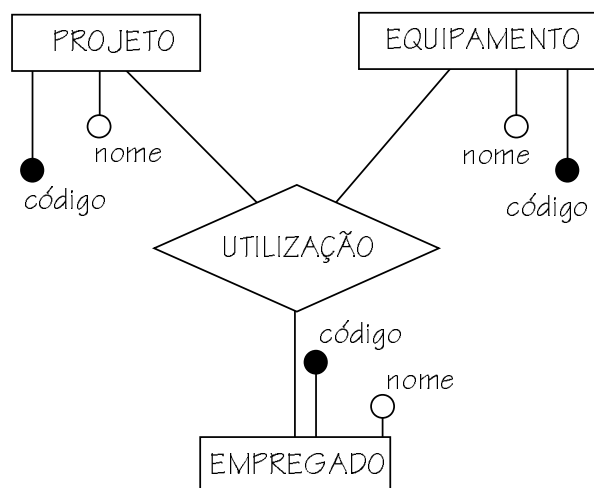


Figura 6.13: Exemplo de DER para utilização de equipamentos

Caso seja projetada um banco de dados relacional para este exemplo usando as regras de projeto mostradas no capítulo anterior, o esquema do banco de dados conterá quatro tabelas. Três tabelas 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 que equipamento é usado em que projeto (relacionamento [Proj-Eq](#)) e deseja-se saber que empregado está alocado a que projeto (relacionamento [Proj\\_Emp](#)). A informação de que equipamento é usado por que empregado passa a ser uma informação derivada: um empregado pode usar todos equipamentos alocados aos projetos dos quais ele participa.

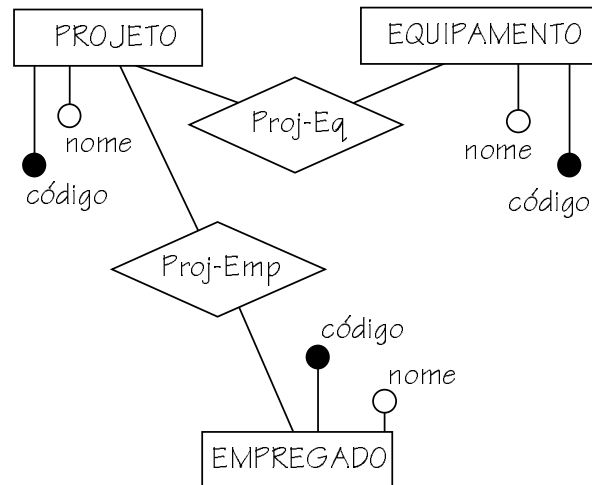


Figura 6.14: Exemplo da utilização de equipamentos - nova versão

Continuando o exemplo, vamos supor que o banco de dados projetada para o relacionamento ternário não tenha sido modificado, apesar da existência de novos requerimentos. Vamos verificar como ficaria o conteúdo da tabela [UTILIZAÇÃO](#) neste caso (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, porque para cada equipamento usado no projeto é necessário informar todos seus empregados. A consequência é que também a informação de que 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 verificarmos a tabela contra as formas normais vistas até este ponto, observa-se 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). Esta forma normal baseia-se no conceito de *dependência funcional multi-valorada*.

Uma coluna ou conjunto de colunas depende multi-valoradamente de uma coluna (determinante) da mesma tabela quando um valor do atributo determinante identifica repetidas vezes um *conjunto* de valores na coluna dependente. No caso do exemplo (ver Figura 6.16), a coluna [CodEmp](#) depende multi-valoradamente 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	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

CodProj	CodEmp	CodEquip
1	1	1
1	2	1
1	3	1
1	1	2
1	2	2
1	3	2
2	2	2

Figura 6.16: Dependência funcional multi-valorada

Para representar uma dependência funcional multi-valorada usa-se uma expressão semelhante a usada para representar dependências funcionais simples substituindo a seta simples por uma seta dupla. Na tabela UTILIZAÇÃO, há as seguintes dependências:

CodProj  $\twoheadrightarrow$  CodEmp

CodProj  $\twoheadrightarrow$  CodEquip

Uma tabela está na 4FN caso, além de estar na 3FN, não possua mais que uma dependência funcional multi-valorada. Portanto, a tabela UTILIZAÇÃO não está na 4FN e deve ser decomposta em duas tabelas:

ProjEmp (CodProj,CodEmp)

ProjEquip (CodProj,CodEquip)

quarta forma normal (4FN)
=
uma tabela encontra-se na quarta forma normal, quando, além de estar na 3FN, não contém dependências multi-valoradas

Estas tabelas correspondem à implementação dos dois relacionamentos da Figura 6.14. Estas tabelas, no caso do exemplo, teriam o conteúdo mostrado na Figura 6.17.

ProjEmp:

CodProj	CodEmp
1	1
1	2
1	3
2	2
3	3
3	4
4	2

ProjEquip:

CodProj	CodEquip
1	1
1	2
2	2
2	4
3	1
3	3
3	5
4	5

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.

### 6.5.6.1 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  $\tilde{N}$ .

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 6.5.

Outra situação análoga é o uso de uma chave alternativa, ao invés 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 CIC do empregado, ao invés 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 ÑÑ, a chave primária usual da tabela, como se ela aparecesse no arquivo normalizado.

#### **6.5.6.2 Atributos relevantes implicitamente representados**

Atributos podem aparecer em arquivos convencionais de forma implícita, na forma de ordenação de registros ou de listas, na forma de 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 acima, agora com uma modificação. Na nova versão, não existe mais 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 ÑÑ seria a seguinte.

**ÑÑ**

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 linhas não estão ordenadas, a informação da classificação dos candidatos em um curso foi perdida no processo de normalização. Assim, o procedimento correto teria sido o de incluir explicitamente na tabela já na forma ÑÑ (coluna **Ordem-Cand**) a informação que aparece implicitamente no arquivo na forma da ordenação dos registros. A tabela na forma ÑÑ que resultaria neste caso é a 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 é o caso do uso de 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.

### 6.5.6.3 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 performance da implementação em questão. Exemplos deste tipo de atributos são campos com o número de ocorrências de listas, com o tamanho de outros campos, com estampas de tempo, etc. Este campos devem ser eliminados já quando da passagem a 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:

- d) 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.
- e) O processo de normalização garante que cada tabela, se considerada isoladamente, esteja livre de redundâncias de dados. Entretanto, certos casos de redundâncias de dados entre tabelas podem ainda 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 destes três passos é descrito em detalhe.

### 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 acima. 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)

Vamos considerar ainda que um outro documento tenha sido normalizado, resultando no modelo relacional abaixo.

**Documento2:**

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 documentos.

Da mesma forma, caso as colunas **CodProj** e **CodEmp** tenham os mesmos valores nas tabelas **ProjEmp** de ambos 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 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 diferentes objetos 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 outra vez sob o nome **TítuloProj**.

Conflitos de nomes são resolvidos através de renomeação. Infelizmente, não há ainda técnicas na literatura que resolvam de forma sistemática o problema do conflito de nomes. Assim, resolução deste conflito continua dependendo muito da experiência e do conhecimento do modelador.

### 6.6.2 Integração de tabelas com chaves contidas

Uma outra situação na qual tabelas são fundidas é aquela na qual uma tabela contém somente a chave primária e a 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 abaixo mostradas.

AlunoDisc            (Cod-AI, Cod-Disc)

AlunoDiscSem        (Cod-AI, 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-AI](#) e [Cod-Disc](#) da tabela [AlunoDisc](#) contém exatamente os mesmos valores que as colunas [Cod-AI](#) e [Cod-Disc](#) 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-AI](#) e [Sem-Disc](#) da tabela [AlunoSem](#) contenham exatamente os mesmos valores que as colunas [Cod-AI](#) e [Sem-Disc](#) da tabela [AlunoDiscSem](#).

AlunoSem            (Cod-AI, Sem-Disc, BolsaSimNao)

AlunoDiscSem        (Cod-AI, Cod-Disc, Sem-Disc, Nota-Disc)

Neste exemplo, a tabela [AlunoSem](#) não deve ser integrada à tabela [AlunoDiscSem](#), pois, apesar da chave primária da primeira tabela estar contida na chave primária da segunda tabela, a primeira possui atributos não chave. Se o atributo [BolsaSimNao](#) 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.

Um 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-AI, Sem-Disc)

AlunoDiscSem        (Cod-AI, 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 matri-

culado, 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 abaixo mostrado.

**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 uma coluna não chave ([NomeGerenteDepto](#)) depende funcionalmente de outra 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 consequências da omissão de chaves primárias em arquivos convencionais, que havia sido citado acima.

Assim, deve-se, após a integração de modelos, verificar as tabelas, afim de garantir que todas estejam efetivamente na 3FN.

## 6.7 CONSTRUÇÃO DO MODELO ER E ELIMINAÇÃO DE REDUNDÂNCIAS

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 transformação de modelos relacionais em modelos ER.

## 6.8 VERIFICAÇÃO DO MODELO ER - LIMITAÇÕES DA NORMALIZAÇÃO

O processo de normalização não conduz necessariamente a um modelo ER perfeito. A normalização apenas elimina campos multi-valorados e elimina redundâncias de dados detectadas pelas formas normais descritas.

Conforme mencionado, na Seção 6.5.1, aqui optamos pela alternativa de decompor tabelas na passagem à 1FN. Esta alternativa, apesar de mais simples de tratar na prática pode levar a imperfeições no modelo (ver exemplo no **Exercício 6.16**).

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 bibliográfica ao final do Capítulo). Como estas formas normais tra-

tam de casos que aparecem pouco frequentemente 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 tipos de 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.

## EXERCÍCIOS

A lista de exercícios abaixo contém 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 sistema de preparação de congressos da IFIP que foi apresentado no **Exercício 3.9**. A ele referem-se os exercícios **Exercício 6.3** até **Exercício 6.11**. O outro sistema é o sistema de controle de um almoxarifado que foi apresentado no **Exercício 3.10**. A ele referem-se os exercícios **Exercício 6.12** até **Exercício 6.21**. Para resolver estes exercícios é recomendável que o leitor tenha lido a descrição do respectivo sistema no Capítulo 3.

**Exercício 6.1:** 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 5.4**. A tabela apresentada está na primeira forma normal. Apresente a segunda e terceira formas normais.

**Exercício 6.2:** No contexto de um sistema de controle acadêmico, considere a tabela abaixo:

**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 6.3:** (refere-se ao sistema de preparação de congressos descrito no **Exercício 3.9**) A Figura 6.18 apresenta uma lista todos 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. Após, em várias colunas são listados o código do artigo, seu título, 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.

## 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	<i>Semantic Integration in Heterogeneous Databases</i>	<i>Heterogeneous Databases</i>	2	<i>Wen-Suan Li</i>
2	<i>Providing Dynamic Security in a Federated Database</i>	<i>Heterogeneous Databases</i>	4 21	<i>Chris Clifton</i> <i>N.B. Idris</i>
3	<i>Efficient and effective clustering methods</i>	<i>Spatial databases</i>	7 32 12	<i>W.A. Gray</i> <i>R.F. Churchhouse</i> <i>Raymond R. Ng</i>
4	<i>Automated Performance Tuning</i>	<i>Performance and Optimization</i>	14 36	<i>Jiawei Han</i> <i>Kurt. P. Brown</i>
5	<i>Bulk Loading into an OODB</i>	<i>Object oriented databases</i>	1	<i>Janet L. Wiener</i>

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	<i>Temporal aspects in OO models</i>	<i>Temporal modeling</i>	2	<i>Wen-Suan Li</i>

Figura 6.18: Lista de artigos submetidos aos congressos

**Exercício 6.4:** (refere-se ao sistema de preparação de congressos descrito no Exercício 3.9) 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 na explicações abaixo.

### Arquivo COBOL (artigos)

```
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

O arquivo contém um registro para cada artigo submetido a congresso. Lembre do exercício precedente, que 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 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 6.5.6.3). 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 6.5:** (refere-se ao sistema de preparação de congressos descrito no Exercício 3.9) A Figura 6.20 apresenta o programa de um congresso.

## **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 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

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 subsequentes). Após 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 tem 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 6.5.6.1), que não são importantes para seu leitor. 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 6.5.6.2) a informação da sequência de apresentação dos artigos em uma seção.

Execute a normalização do documento, mostrando cada uma das formas normais.

**Exercício 6.6:** (refere-se ao sistema de preparação de congressos descrito no **Exercício 3.9**) A Figura 6.21 apresenta a carta de aceitação de um artigo. Esta carta enviada para o autor principal do artigo. 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 caso do exercício anterior, há várias chaves primárias omitidas no documento. Deixamos ao leitor que as identifique e inclua na normalização.

Execute a normalização do documento, mostrando cada uma das formas normais.

**Exercício 6.7:** (refere-se ao sistema de preparação de congressos descrito no **Exercício 3.9**) 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 que as feitas com referência ao arquivo de artigos apresentado acima.

Execute a normalização do documento, mostrando cada uma das formas normais.

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

```

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

**Exercício 6.8:** Realize a integração dos modelos referentes aos cinco documentos e arquivos que você normalizou nos exercícios precedentes.

**Exercício 6.9:** Identifique as chaves estrangeiras no modelo resultante da integração feita no exercício anterior.

**Exercício 6.10:** Construa um diagrama ER a partir do modelo relacional obtido no Exercício anterior. Utilize o processo de engenharia reversa descrito na Seção 5.3.

**Exercício 6.11:** Identifique as diferenças do modelo ER construído no exercício anterior com o modelo ER construído no **Exercício 3.9**. Comente a causa das diferenças.

#### Lista de estoque em dd/mm/aa

**Corredor:** (---NoCorredor---)

Receptáculo	Codigo da Peca	Descricao	Quantidade
(---NoRecptac)	(---CodPeca---	(---DescricaoPeca---	(---Quant---
(---NoRecptac)	(---CodPeca---	(---DescricaoPeca---	(---Quant---
...			

**Corredor:** (---NoCorredor---)

Receptáculo	Codigo da Peca	Descricao	Quantidade
(---NoRecptac)	(---CodPeca---	(---DescricaoPeca---	(---Quant---
(---NoRecptac)	(---CodPeca---	(---DescricaoPeca---	(---Quant---
...			

Figura 6.23: Lista de estoque por corredor do almoxarifado

**Exercício 6.12:** (refere-se ao sistema de controle de almoxarifado descrito no **Exercício 3.10**) 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.

```
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 0 TO 20 TIMES
                DEPENDING ON NO-DE-CORREDORES.
        05      NO-CORR
        05      NO-DE-RECEPTACULOS
        05      ITEM-RECEPT OCCURS 0 TO 5 TIMES
                DEPENDING ON NO-DE-RECETACULOS.
        07      NO-RECEPT
        07      COD-PECA
        07      DESCR-PECA
        07      QTDE
```

Figura 6.24: Lista de distribuição

**Exercício 6.13:** (refere-se ao sistema de controle de almoxarifado descrito no **Exercício 3.10**) 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 usada na distribuição, o código da empilhadeira a usar, o operador da empilhadeira e uma lista de itens que identificam os corredores a visitar durante a distribuição. A lista informa, para cada corredor, quais os receptáculos que serão visitados e, para cada receptáculo qual o código e a descrição da peça a armazenar, juntamente com a quantidade de unidades a colocar no receptáculo.

Execute a normalização do documento, mostrando cada uma das formas normais.

#### Ao Cliente

(---codCliente---) (---nomeCliente---)

**Informamos que seu pedido** (---noPedido---) **estará sendo entregue na rampa** (---noRampa---)

Figura 6.25: Boleto indicativo de rampa de saída de peças

**Exercício 6.14:** (refere-se ao sistema de controle de almoxarifado descrito no **Exercício 3.10**) 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.

**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---)	(---)
		<b>Total entregue:</b>		(---)
(---)	(---)	(---)	(---data---)	(---)
		<b>Total entregue:</b>		(---)
(---)	(---)	(---)	(---data---)	(---)
			(---data---)	(---)
			(---data---)	(---)
		<b>Total entregue:</b>		(---)
(---)	(---)	(---)		
		<b>Total entregue:</b>		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---)	(---)
		<b>Total entregue:</b>		(---)

Figura 6.26: Situação de atendimento de ordens de compra

**Exercício 6.15:** (refere-se ao sistema de controle de almoxarifado descrito no **Exercício 3.10**) 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 6.16:** (refere-se ao sistema de controle de almoxarifado descrito no **Exercício 3.10**) 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). Após e listado, para cada peça pedida, seu código, sua descrição e a quantidade pedida.

Execute a normalização do documento, mostrando cada uma das formas normais.

**Exercício 6.17:** (refere-se ao sistema de controle de almoxarifado descrito no **Exercício 3.10**) 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 6.18:** Realize a integração dos modelos referentes aos documentos e arquivos do sistema de almoxarifado que você normalizou nos exercícios precedentes.

**Exercício 6.19:** Identifique as chaves estrangeiras no modelo resultante da integração feita no exercício anterior.

### Pedido

**Pedido:** (---noPed---) **Data do pedido:** (---data---)

**Código do Cliente:** (---codClie---)

**Nome do Cliente:** (---NomeCliente)

**Telefones p/ contato:** (---noTel---) (---noTel---) (---noTel---)

<b>Código Peça</b>	<b>Descrição Peça</b>	<b>Quantidade Pedido</b>
(---)	(---)	(---)
(---)	(---)	(---)

Figura 6.27: Pedido

**Exercício 6.20:** Construa um diagrama ER a partir do modelo relacional obtido no exercício anterior. Utilize o processo de engenharia reversa descrito na Seção 5.3.

**Exercício 6.21:** Identifique as diferenças do modelo ER construído no exercício anterior com o modelo ER construído no **Exercício 3.10**. Comente a causa das diferenças.

## 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

## REFERÊNCIAS BIBLIOGRÁFICAS

A idéia de normalização que serve de base a este capítulo surgiu junto com a abordagem relacional no artigo original de Codd [2]. A normalização é apresentada na maioria dos textos introdutórios de banco de dados [3,4].

A engenharia reversa é apresentada no livro de Batini, Ceri e Navathe [1]. Uma boa cobertura do assunto aparece também no livro de Setzer [5].

- [1] Batini, C., Ceri, S., and Navathe, S. *Database Design: An Entity-Relationship Approach*, Benjamin/Cummings 1992.
- [2] Codd, E.F. A relational model for large shared data banks. *Communications of the ACM*. **13**:6, pp. 377-387, 1970
- [3] Elmasri, R. & Navathe, S.B. *Fundamentals of Database Systems*. Second Edition. Benjamin/Cummings, Redwood City, California, 1994
- [4] Korth, H. & Silberschatz, A. *Sistemas de Bancos de Dados*. 2ª edição, Makron Books, 1994
- [5] Setzer, V.W. *Banco de Dados*. Segunda edição. Ed. Edgard Blücher, 1987

# Soluções de exercícios selecionados

O presente capítulo tem por objetivo apresentar as soluções de exercícios selecionados. Procurei apresentar soluções para todos os problemas mais complicados.

Em muitos casos, 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 3.1.2. Lá é apresentado um diagrama de ocorrências que mostra que:

- ☐ Uma pessoa pode estar casada com ela mesma e
- ☐ Uma pessoa pode participar de dois casamentos, desde que na posição de marido em um casamento e de esposa em outro.

A Figura 7.1 mostra como é possível excluir as duas possibilidades acima do modelo ER. A mudança que foi feita em relação ao modelo original (Figura 2.4) foi a de transformar **CASAMENTO** em entidade e, com isso, abrir a possibilidade de especificar a cardinalidade do relacionamento de casamento com pessoa.

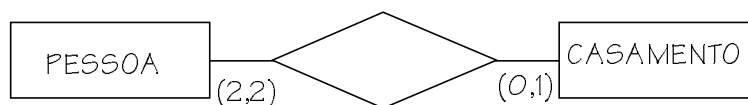


Figura 7.1: Implementando restrições no casamento



Uma 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  $e5$

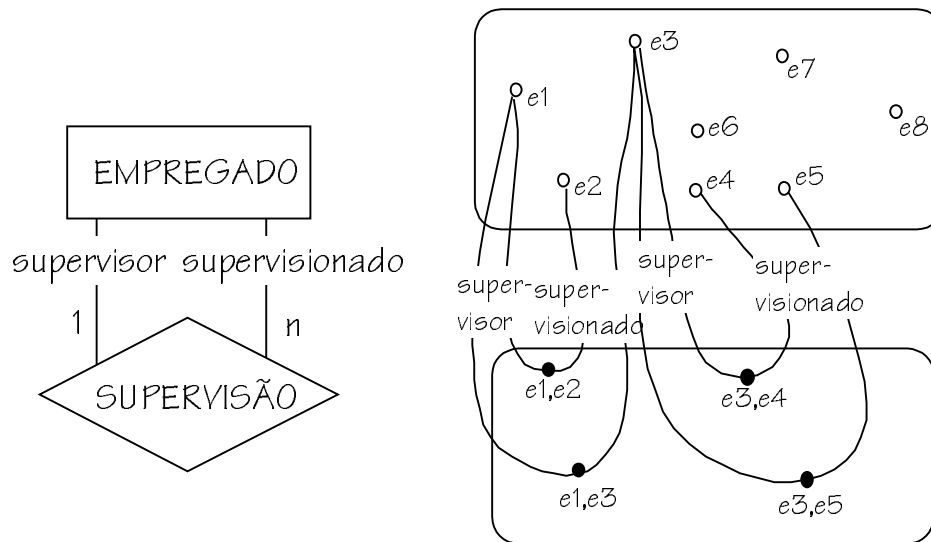


Figura 7.2: Diagrama de ocorrências para o relacionamento **SUPERVISÃO**

Na Seção 3.1.2, é mostrado um outro exemplo de diagrama de ocorrências, contendo uma situação, que, apesar de permitida pelo diagrama, não corresponde a nossa intenção ao modelar o relacionamento de **SUPERVISÃO**.

**Exercício 2.7.** A Figura 7.3 mostra o resultado da transformação do modelo ER da Figura 2.10 em um diagrama que não contém relacionamentos ternários.

Observe que uma instância de **DISTRIBUIÇÃO** é identificada pelos três relacionamentos de que participa.

Estritamente falando, este modelo não é equivalente ao modelo original com relacionamento ternário. A diferença entre os modelos está no fato de o modelo sem relacionamento ternário não conter a restrição que especificava que um produto, em uma cidade, somente pode possuir um distribuidor. Esta restrição somente é representável no modelo com relacionamentos ternários, já que somente neste é possível falar de cardinalidade em relação a pares de entidades. Entretanto, se desconsiderarmos este detalhe, ambos modelos são equivalentes.

Uma solução que a primeira vista pode parecer equivalente ao relacionamento ternário é a apresentada na Figura 7.4. Esta solução não é equivalente a original. Nela ocorre perda de informações. 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.

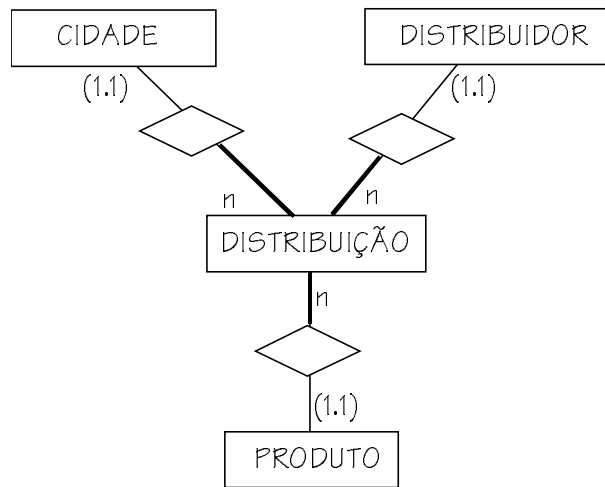


Figura 7.3: Transformação de relacionamento ternário em entidade

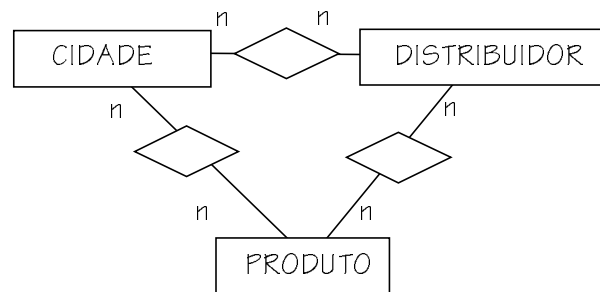


Figura 7.4: Tentativa de transformação de relacionamento ternário

**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**.

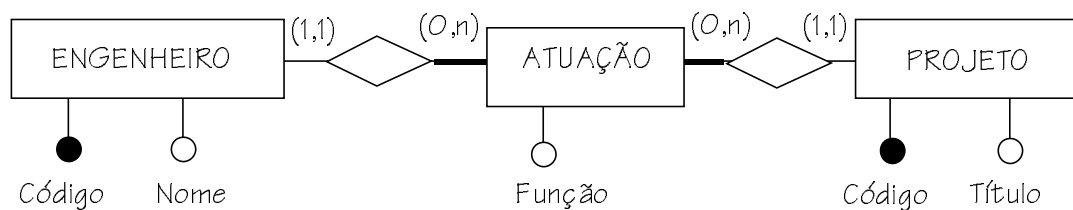


Figura 7.5: Resultado de transformação de relacionamento em entidade

**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 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 a entidade empregado correspondente.

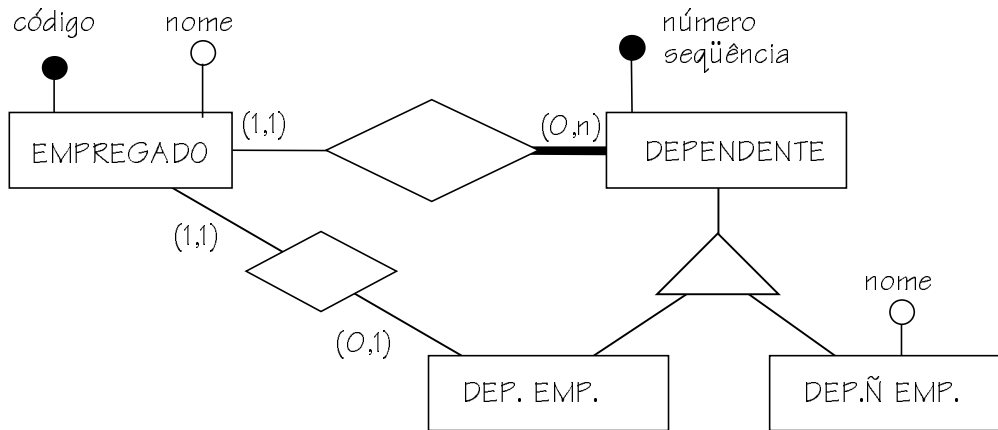


Figura 7.6: Dependente pode ser empregado

**Exercício 2.16** Caso não seja usados atributos multi-valorados é necessário criar uma entidade para cada atributo multi-valorado. Observar o identificador desta entidade. Um telefone é identificado pelo seu nome e pelo cliente correspondente. Isso permite que diferentes clientes tenham o mesmo número de telefone (o que era permitido pelo modelo original).

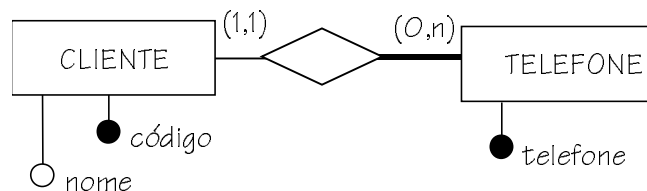


Figura 7.7: Transformação de atributo multi-valorado em entidade

**Exercício 2.17** A solução para modelar uma especialização não exclusiva é usar relacionamentos para ligar as entidades especializadas à entidade genérica (Figura 7.8).

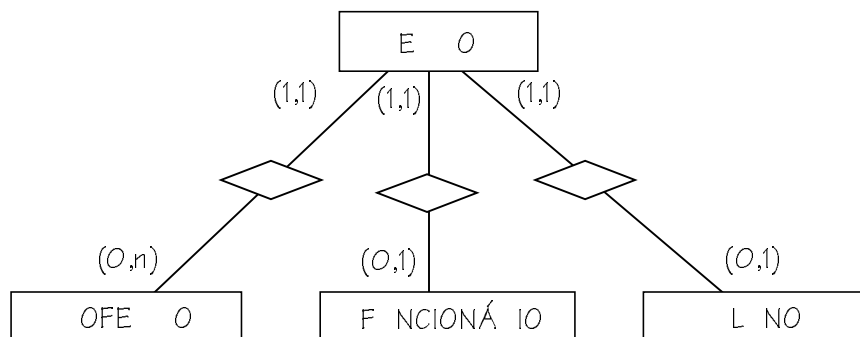


Figura 7.8: Tratando especialização não exclusiva através de relacionamentos

Observe as cardinalidades dos relacionamentos. Pelo enunciado do problema, uma pessoa pode possuir vários contratos de professor.

Observe também que nesta modelagem cada entidade necessita de um identificador. Caso deseje-se usar como identificador da entidade especializada o mesmo identificador da entidade genérica (o que é possível caso o relacionamento seja 1:1), a entidade especializada passa a ser tratada como entidade fraca. Ela é identificada pelo relacionamento com a entidade genérica.

**Exercício 2.22** Não é possível expressar esta restrição pelo fato de o modelo ER não possuir uma notação que expresse que a união de dois relacionamentos (no caso, o de **VENDA** com **MEDICAMENTO** e o de **VENDA** com **PERFUMARIA**) tem cardinalidade mínima um. Esta restrição teria que ser especificada fora do modelo ER.

**Exercício 2.28** 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, para cada processador de textos por ela dominada. 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 a exclusão do processador de textos em questão.

**Exercício 2.29** Pela definição de especialização que consideramos neste livro, a mesma 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 ambas especializações de **EMPREGADO** na 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 retirar a entidade **GERENTE** da mesma hierarquia de generalização/especialização na qual aparecem **SECRETÁRIA** e **ENGENHEIRO**. Neste caso, **GERENTE** passa a ser um auto-relacionamento de **EMPREGADO**.

### **Exercício 3.1**

- a) O relacionamento apresentado não inclui a restrição de que um produto não pode ser composto por ele mesmo.
- b) 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.
- c) 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 está dentro do poder de expressão de modelos ER.

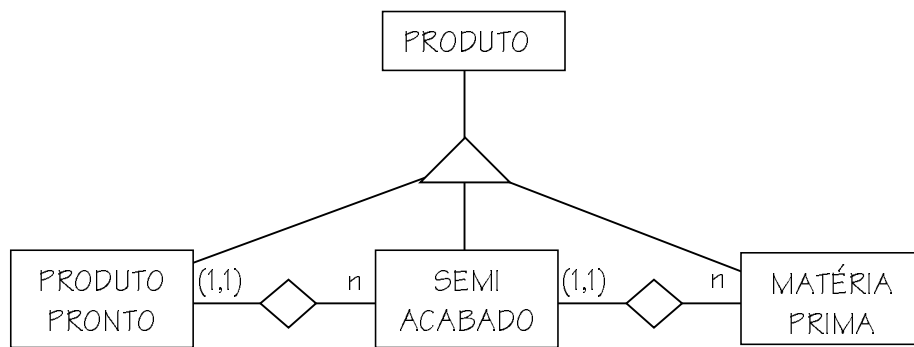


Figura 7.9: Estrutura de produto com número fixo de níveis

**Exercício 3.3** No modelo, há atributos específicos de pessoa física, outros específicos de pessoa jurídica e alguns comuns a ambos tipos de clientes. Este é o caso típico para a aplicação do conceito de generalização/especialização. A apresenta um modelo para a mesma realidade usando este conceito.

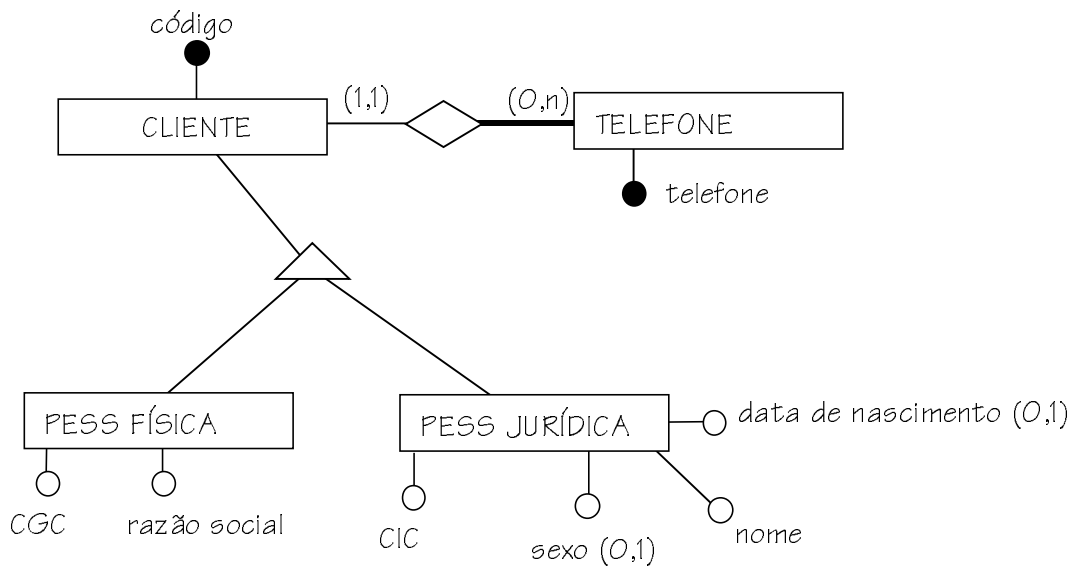


Figura 7.10: Substituindo atributos opcionais por especializações

O novo modelo é mais preciso que o original. No original, os atributos das entidades especializadas apareciam como opcionais. O modelo não informava que atributos pertenciam a qual tipo de cliente e se o atributo era ou não obrigatório para um tipo de cliente.

No novo modelo, fica claro que atributos pertencem a cada uma das especializações e se eles são ou não obrigatórios.

**Exercício 3.5** Para o estudo de caso da administradora de imóveis, vamos seguir os passos do processo de modelagem descrito na Seção 3.5.2.1.

#### 1. Enumeração de entidades

Uma primeira leitura do enunciado resulta nas seguintes entidades: **ADMINISTRADORA**, **CONDOMÍNIO**, **UNIDADE**, **PESSOA**

#### 2. Identificação de relacionamentos

Os relacionamentos que podem ser identificados são os seguintes:

- **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 uma administradora somente (como é o caso de 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 que administradora administra que condomínio. Este relacionamento somente deveria aparecer, caso várias administradoras pudessem coexistir no mesmo BD (ver discussão na Seção 3.3.5)

### 3. *Identificação de cardinalidades máximas*

As cardinalidades identificadas aparecem no DER da

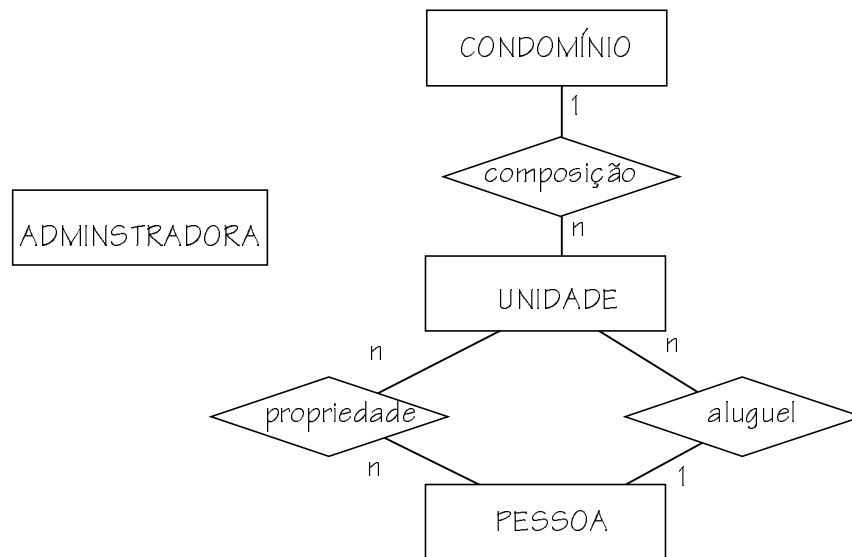


Figura 7.11: Diagrama ER para a administradora de imóveis

**Exercício 3.6** Seguindo os passos do processo de modelagem, chegamos aos seguintes resultados.

#### 1. *Enumeração de entidades*

**LOCADORA, FILME, FITA, CLIENTE, CATEGORIA, 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 3.2.1).

Outra decisão é a de como modelar o empréstimo, se através de uma entidade ou um relacionamento. Optamos por modelá-lo como relacionamento.

#### 2. *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 7.12.

### 3. Determinação de atributos

Um primeiro levantamento de atributos, com base na leitura do enunciado resulta nos atributos que aparecem no DER da Figura 7.12. O único atributo que não aparece explicitamente no enunciado é o número do rolo (atributo de fita). Este atributo é necessário para filmes multi-rolo, para identificar que rolo do filme está armazenado na fita.

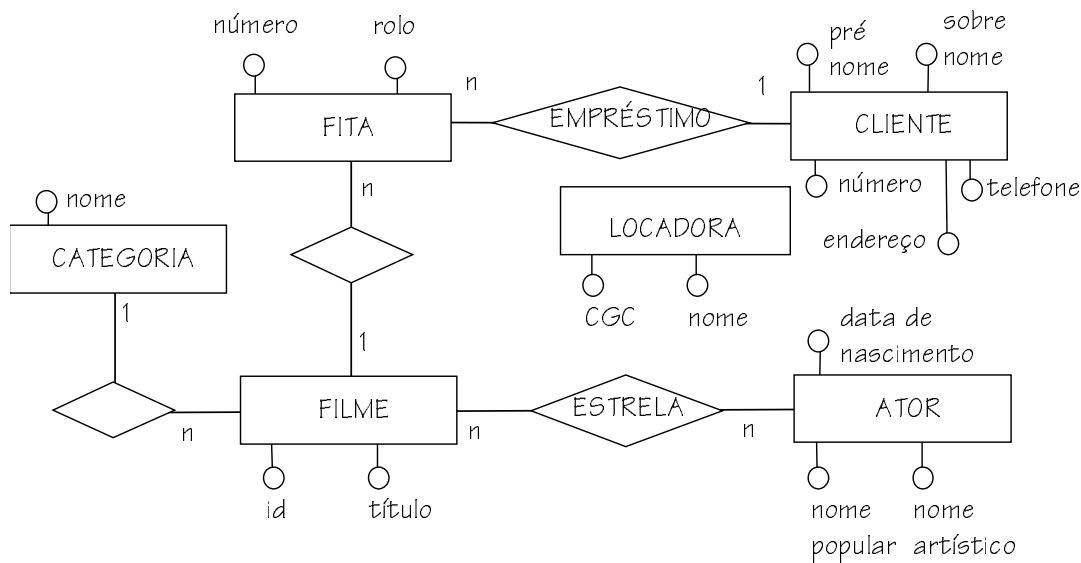


Figura 7.12: Diagrama ER para locadora de vídeos (primeira versão)

### 4. 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**.

### 5. Verificação de aspectos temporais

Não há atributos nem relacionamentos dos quais deva ser mantida história. Não há alterações a fazer no que tange a aspectos temporais.

## 6. Domínios dos atributos

Nesta etapa devem ser definidos os domínios dos atributos. Isso normalmente é feito já com uso de uma ferramenta CASE. Assim, nos estudos de caso, deixaremos de lado esta etapa.

## 7. Definição de cardinalidades mínimas

Na Figura 7.13 estão apresentadas as cardinalidades mínimas que podem ser deduzidas do enunciado do problema

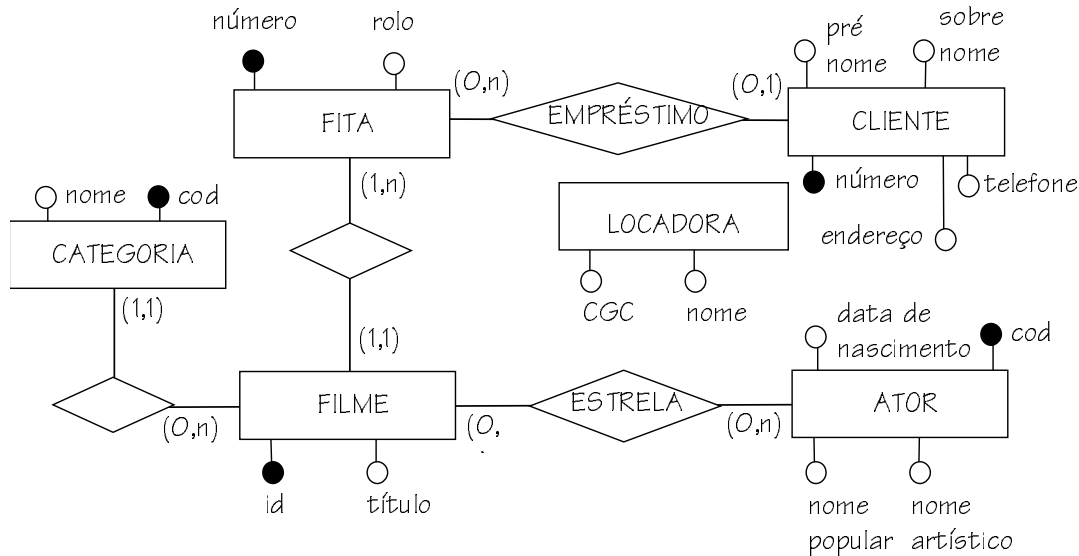


Figura 7.13:Diagrama ER para locadora de vídeo (final)

## Exercício 3.7

### 1. Enumeração de entidades

A leitura do enunciado nos leva as seguintes entidades: **COMPANHIA**, **RESERVA**, **PASSAGEIRO**, **TRECHO**, **VOO**, **CIDADE**, **AEROPORTO**, **TIPO-AERONAVE**, **HORARIO**, **ASSENTO**.

Não foi criada uma entidade correspondente as 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.

### 2. Identificação de relacionamentos

Os relacionamentos identificados aparecem no diagrama ER da Figura 7.14.

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.



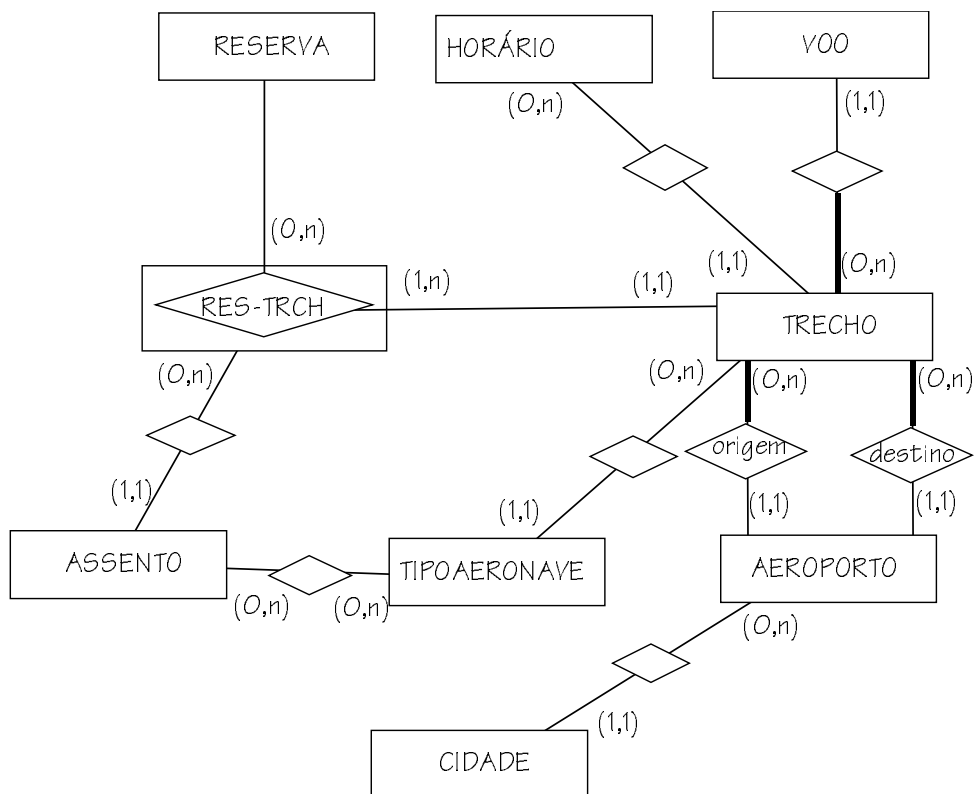


Figura 7.14: Diagrama ER para sistema de reserva de passagens aéreas

### 3. Determinação de atributos e identificadores

Para não sobrecarregar o diagrama, listamos os atributos das entidades abaixo. Os atributos identificadores estão sublinhados e os relacionamentos identificadores aparecem no diagrama ER da Figura 7.14.

RESERVA (codigo reserva, passageiro, prazo)

VOO (número)

TRECHO ()

AEROPORTO (código, nome)

CIDADE (código, nome, país)

TIPO AERONAVE (código, descrição)

HORARIO (dia semana, horário partida, horário chegada)

ASSENTO (número, classe)

RSRV-TRCH (data)

### 4. Restrições que não podem ser representadas no modelo

Neste sistema aparecem diversas restrições que não se deixam representar 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 voo 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 muito 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 a necessidade de criação de uma entidade **TRECHO-DIA** correspondendo a cada viagem específica de um trecho de vôo em uma data.

### **Exercício 3.8 Sistema para locadora de veículos**

#### **1. 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**).

Raciocínio análogo pode se 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 do tipo. Essa informação é multi-valorada (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.

#### **2. Identificação de relacionamentos**

Os relacionamentos identificados aparecem no diagrama ER da Figura 7.15.

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 em que 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 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 informar o motorista que está responsável pelo automóvel. Observe que não há um relacionamento 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).

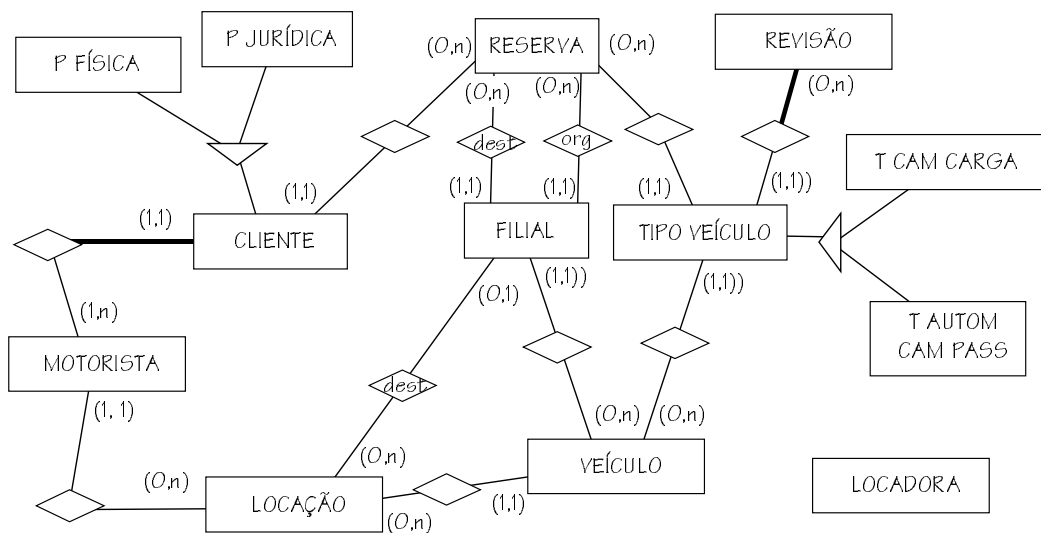


Figura 7.15: Diagrama ER para sistema de locadora de veículos

### 3. Determinação de atributos e identificadores

Os atributos das entidades são os seguintes:

CLIENTE (código, nome, endereço)

P FÍSICA (sexo, data nascimento, CIC)

P JURÍDICA (CGC, inscrição estadual)

FILIAL (código, localização)

VEÍCULO (placas, número chassis, 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, toda-fitas, CD, direção hidráulica, câmbio automático)

T CAM CARGA (capacidade carga)

REVISÃO (quilometragem)

MOTORISTA (número habilitação, data vencimento, identidade, nome)

RESERVA (número, data retirada, data devolução)

LOCAÇÃO (número, data retirada, data devolução)

LOCADORA (CGC, nome, endereço, telefone)

Os atributos identificadores estão sublinhados na lista acima. Os relacionamentos identificadores estão representados no diagrama ER da Figura 7.15.

### 4. 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 caso existam veículos do tipo previstos para estarem disponíveis 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 3.7**), a presente solução é puramente conceitual, não incluindo redundâncias de dados que procurem melhorar a performance de determinadas operações.

### Exercício 3.9 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 resultado, 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 7.16.

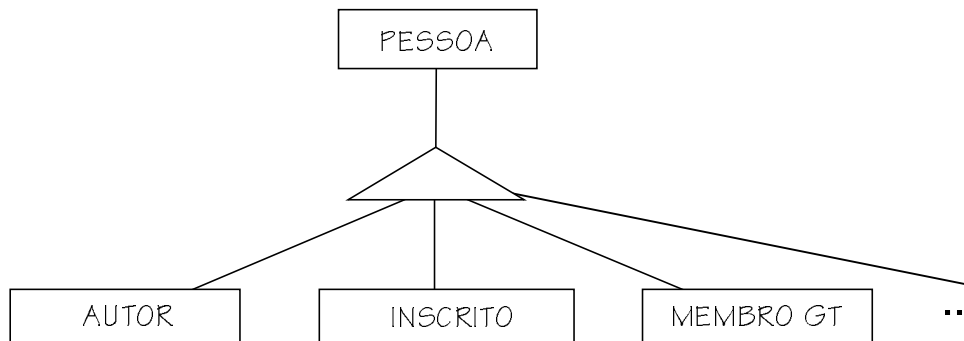


Figura 7.16: Usando generalização/especialização para modelar papéis de pessoas (incorreto)

Essa solução não está correta, se considerarmos a definição de generalização/especialização que apresentamos na Seção 2.4. A mesma pessoa pode cumprir vários papéis, inclusive em um único congresso. O conceito de generalização/especialização não deve ser usado quando uma entidade genérica pode aparecer mais de uma vez em suas especializações (ver discussão).

Por esse motivo, optamos por modelar os papéis que as pessoas cumprem através de relacionamentos, conforme apresentado no diagrama da Figura 7.17.

Os atributos identificados são os seguintes:

CONGRESSO	( <u>código</u> , nome, data realização, local, prazo submissão artigos, prazo inscrição)
GT	( <u>código</u> , nome)
PESSOA	( <u>código</u> , nome, instituição, endereço, telefone, fax, e-mail)
ARTIGO	( <u>número</u> , título, aceito/rejeitado)
SESSÃO	( <u>código</u> , data, hora, título)

PESSOA-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 com as 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 performance da aplicação.

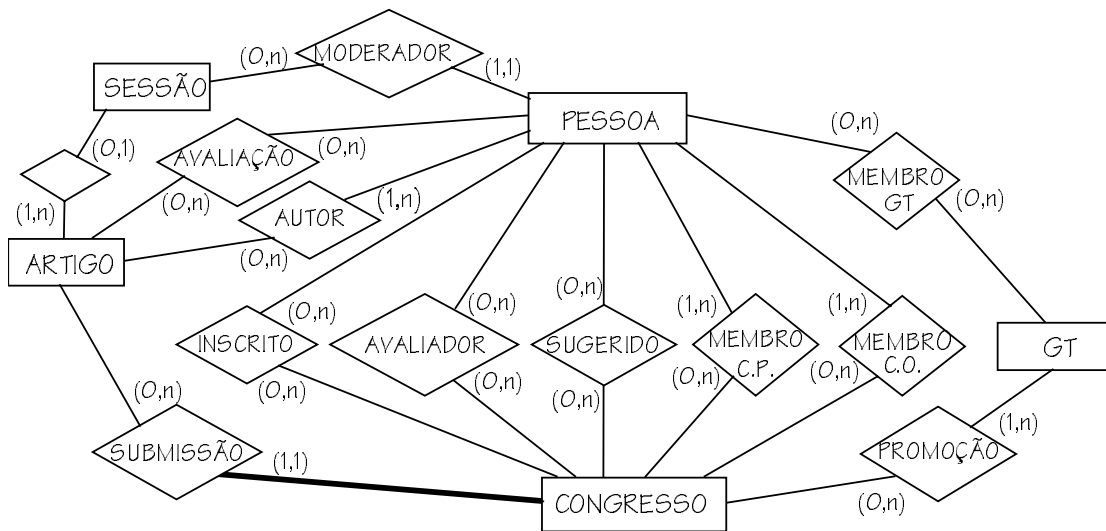


Figura 7.17: Diagrama ER para o sistema de preparação de congressos

Pelas regras de equivalência de modelos vistas na Seção 3.1.3, é possível transformar em entidades os diversos relacionamentos **n:n** que aparecem no modelo da Figura 7.17. Nessa variante de solução, os papéis de pessoa, como **MODERADOR**, **AVALIADOR**, **AUTOR** e outros, são transformados em entidades. A Figura 7.18 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 7.16, 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.

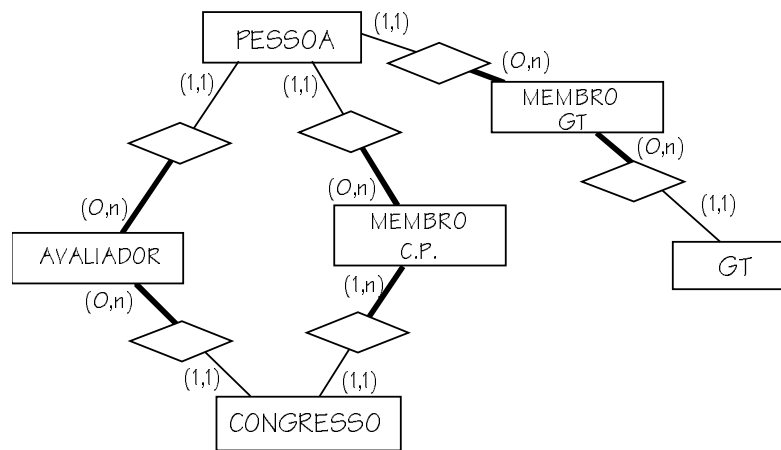


Figura 7.18: Diagrama ER para o sistema de preparação de congressos (parcial - usando entidades ao invés de relacionamentos n:n)

Uma outra particularidade do sistema em questão, é que ele contém uma série de restrições de integridade que não estão expressas no modelo ER. A seguir, listamos estas restrições:

- ☐ 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.

### Exercício 3.10 - Sistema de almoxarifado

A Figura 7.19 apresenta o diagrama ER para o sistema de almoxarifado. Na solução apresentada modela os itens de uma ordem de compra, de um pedido etc. como uma entidade e não como um 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.

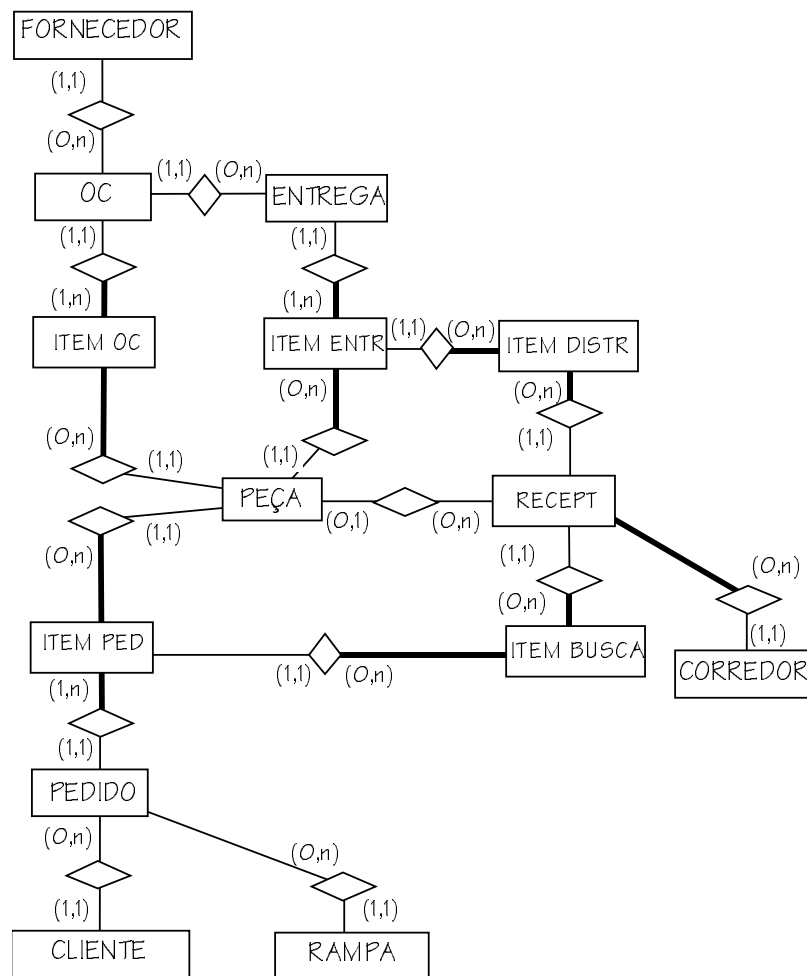


Figura 7.19: Diagrama ER para o sistema de almoxarifado

**Exercício 4.1** - Em um banco de dados relacional, a chave primária de uma tabela é aquela chave única que é utilizada como chave estrangeira em tabelas que referenciam a chave em questão. Assim, 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 um campo ou conjunto de campos para os quais existe um índice que admite duplicatas. Conforme mencionado na Seção 4.1.2.1, 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** - Abaixo estão indicadas as chaves primárias e estrangeiras:

[Aluno](#) ([CodigoAluno](#), [Nome](#), [CodigoCurso](#))

[CodigoCurso](#) referencia [Curso](#)

[Curso](#) ([CodigoCurso](#), [Nome](#))

[Disciplina](#) ([CodigoDisciplina](#), [Nome](#), [Creditos](#), [CodigoDepartamento](#))

[CodigoDepartamento](#) referencia [Disciplina](#)

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](#))

#### Exercício 4.4

- a) Uma linha é incluída na tabela [Consulta](#).

A tabela [Consulta](#) contém duas chaves estrangeiras, ([CodigoConvenio,NumeroPaciente](#)) e [CRM](#). Quando ocorrer uma inclusão em [Consulta](#), é necessário verificar se estas chaves aparecem nas respectivas tabelas ([Paciente](#) e [Medico](#)).

- b) 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 (Seção 5.2):

- ☐ 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 5.1, as regras de tradução apresentadas representam um conjunto de heurísticas que, em geral, levam a uma base de dados bem projetada. 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. Abaixo consideramos uma situação em que a alternativa 2 poderia ser preferível.

Considere-se que os dados de aluno (nome e código do curso) e endereço sofrem constantemente alterações de valores. Além disso, considere-se 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 SGBD 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 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** O esquema relacional referente ao modelo ER da Figura 5.23 é o seguinte:

Produto (CGC, NúmeroProd, NomeComercial, TipoEmbalagem,  
Quantidade, PreçoUnitário, TipoProd, Tarja,Fórmula, Tipo)  
CGC referencia Fabricante  
Fabricante (CGC,Nome,Endereço)  
Venda(Data,NúmeroNota,NomeCliente,CidadeCliente)  
PerfumariaVenda(CGC, NúmeroProd, NúmeroNota,Quantidade,Imposto)  
(CGC,NúmeroProd) referencia Produto  
NúmeroNota referencia Venda  
MedicamentoVenda(CGC, NúmeroProd, NúmeroNota,  
Quantidade,Imposto, CRM, Número)  
(CGC,NúmeroProd) referencia Produto  
NúmeroNota referencia Venda  
(CRM, Número) referencia ReceitaMédica  
ReceitaMédica(CRM,Número,Data)

Comentários:

- ❑ O relacionamento entre Fabricante e Produto foi implementado através da chave estrangeira **CGC** 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 5.2.4.3 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.
- ❑ O relacionamento com Receita Médica foi implementada através de chave estrangeira dentro da tabela referente a entidade associativa.

**Exercício 5.3** O esquema relacional referente ao modelo ER da Figura 5.24 é o seguinte:

Escritório (NúmeroEscr, Local)  
Contrato aluguel (NúmeroEscr, NúmeroContr, Data, Duração,  
NúmeroVeic, NúmeroCarMotorista, EstadoCarMotorista)  
NúmeroEscr referencia Escritório  
NúmeroVeic referencia Veículo  
Cliente (NúmeroCartMotorista, EstadoCartMotorista, Nome, Endereço,  
Telefone)

Veículo (Número, DataPróximaManutenção, Placa, CódigoTipo)

CódigoTipo referencia TipoVeículo

TipoVeículo (CódigoTipo, Nome, ArCondicionado)

Similaridade (CódigoTipo, CódigoTipoSimilar)

Automóvel (CódigoTipo, NúmeroPortas, DireçãoHidráulica,  
CâmbioAutomático, Rádio)

CódigoTipo referencia TipoVeículo

Ônibus (CódigoTipo, NúmeroPassageiros, Leito, Sanitário)

CódigoTipo referencia TipoVeículo

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 5.2.4.3)

**Exercício 5.4** A Figura 7.20 apresenta o modelo ER obtido através da aplicação das regras de engenharia reversa de modelos relacionais.

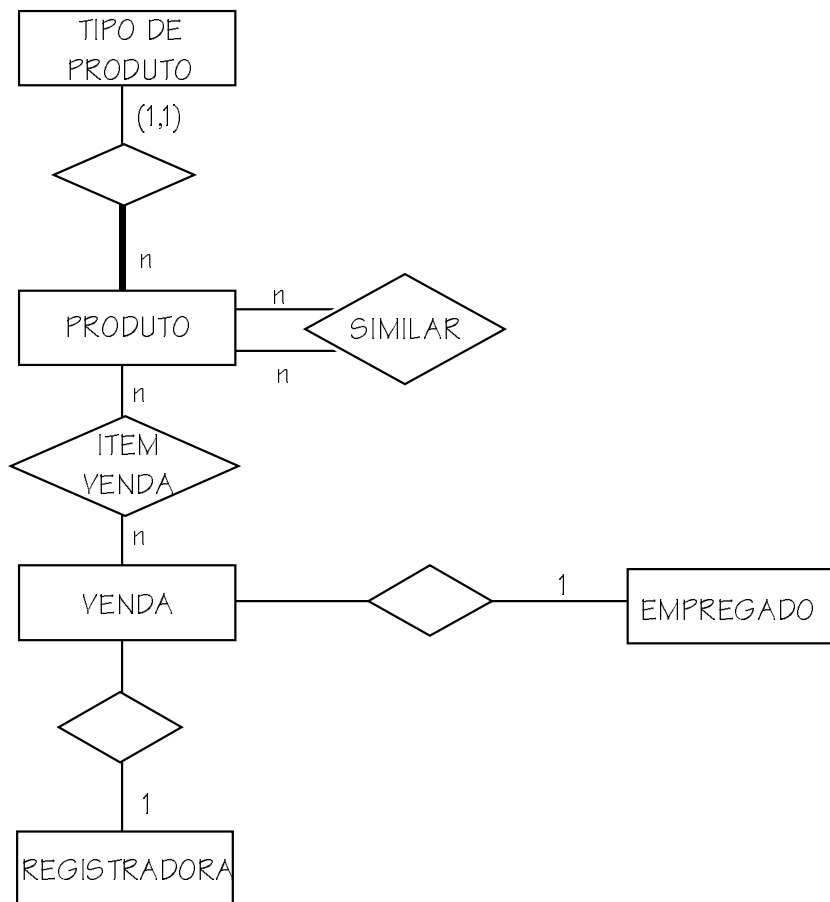


Figura 7.20: Modelo ER obtido através de engenharia reversa para o Exercício 5.4

Os atributos de cada entidade/relacionamento estão listados abaixo (atributos identificadores estão sublinhados):

Produto (NumeroProd, DescricaoProd, PreçoProd)

TipoProd (CodigoTipoProd, DescricaoTipoProd)

Venda (NúmeroNF, DataVenda)

ItemVenda (QtdeItem, PreçoItem)

Registradora (CodReg, SaldoReg)

Empregado (CodEmp, NomeEmp, SenhaEmp)

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 pelo seqüência de raciocínio abaixo:
  - O relacionamento representa a chave estrangeira **CodigoTipoProd** da tabela **Produto**. Como no modelo relacional todos campos são monovalorados, 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-se derivar 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 que colunas são obrigatórias e que colunas são opcionais, não é possível determinar se uma venda obrigatoriamente está associada a um empregado ou se está opcionalmente associada. Já na outra direção do relacionamento (quantas vendas estão associadas a um empregado) não é possível derivar nenhuma informação referente a cardinalidade a partir do modelo relacional fornecido.
- ❑ O mesmo aplica-se ao relacionamento entre **Venda** e **Registradora**.

**Exercício 5.5** A Figura 7.21 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)

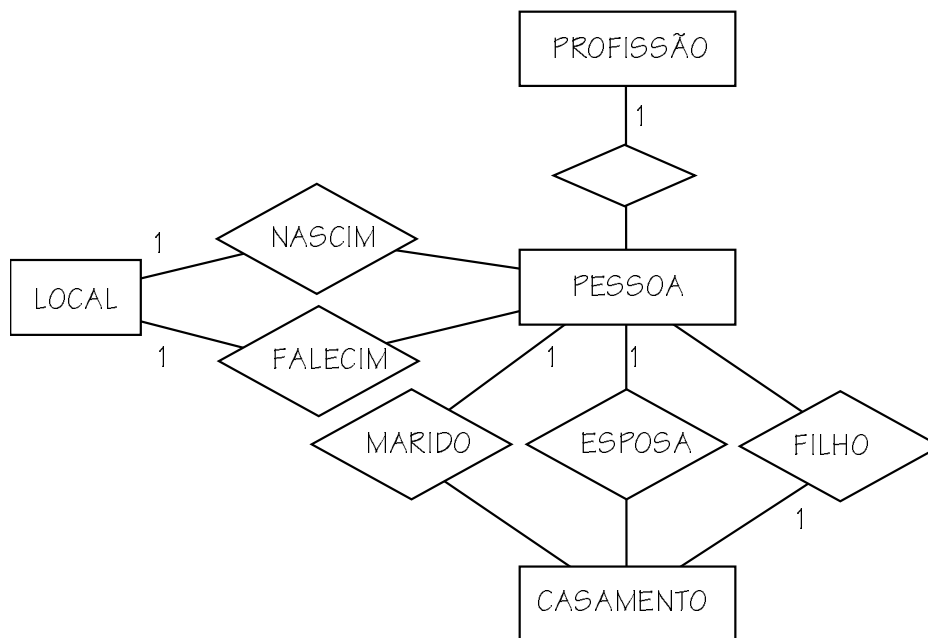


Figura 7.21: Modelo ER obtido por engenharia reversa

**Exercício 6.1** 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 caso do exercício, as dependências funcionais parciais são:

$(\text{CodigoTipoProd}, \text{NumeroProd}) \rightarrow \text{DescricaoProd}$

$\text{CodigoTipoProd} \rightarrow \text{DescricaoTipoProd}$

$\text{NumeroNF} \rightarrow \text{DataVenda}$

$\text{NumeroNF} \rightarrow \text{CodReg}$

$\text{NumeroNF} \rightarrow \text{CodEmp}$

$\text{NumeroNF} \rightarrow \text{NomeEmp}$

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 (NumeroNF, CodigoTipoProd, NumeroProd,  
QtdeItem, PreçoItem)

Produto (CodigoTipoProd, NumeroProd, DescricaoProd)

TipoProd (CodigoTipoProd, DescricaoTipoProd)

Venda (NumeroNF, 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 caso do exercício, há uma dependência funcional transitivas na tabela [Venda](#) que é  $\text{CodEmp} \rightarrow \text{NomeEmp}$ . 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 6.2** 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. Sua eliminação resulta no seguinte modelo relacional:

(CodAluno, CodTurma)

(CodAluno, NomeAluno, CodLocalNascAluno)

(CodLocalNascAluno, NomeLocalNascAluno)

(CodTurma, CodDisciplina)

(CodDisciplina, NomeDisciplina)

**Exercício 6.3** 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 nenhuma 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 6.4** A seguir é apresentada cada uma das forma normais:

### ÑN

Comentário: Os campos **NO-DE-AUTORES**, **NO-DE-REVISORES** e **NO-DE-TEMAS** contém 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 6.5.6.3, estes campos são eliminados já na passagem à forma ÑÑ.

(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

O arquivo contém um registro para cada artigo submetido a congresso. Lembre do exercício precedente, que 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 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 6.5.6.3). 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 6.5** O documento é um bom exemplo das dificuldades que pode apresentar a normalização de documentos preparados para leitura por pessoas e não para processamento eletrônico.

ÑÑ

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 apresentado 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 6.5.6.1). No caso foram omitidos o número do artigo e 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.

(CodCongr, NomeCongr,  
                   (Data,  
                     (Hora, TituloSessao, CodModerador, NomeModerador  
                       (CodArt, TituloArt,  
                         (CodAutor, NomeAutor, PaisAutor))))))

#### 1FN

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 ÑÑ, como foi nos exemplos até aqui apresentados. Para detalhes ver seção 6.5.1.

(CodCongr, NomeCongr)  
 (CodCongr, Data)  
 (CodCongr, Data, Hora, TituloSessao, CodModerador, NomeModerador)  
 (CodCongr, Data, Hora, CodArt, TituloArt)  
 (CodCongr, Data, Hora, CodArt, CodAutor, NomeAutor, PaisAutor)

#### 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 6.6

ÑN

Comentários:

- ❑ O documento é uma carta que é enviado 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 multi-valorados de dados. Por isso, não há tabelas aninhadas

(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 6.7

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 6.8

Os modelos normalizados de cada documento estão listados abaixo. Adicionalmente, cada tabela recebeu um nome. Isto facilita identificar tabelas que devam ser juntadas.

Modelo do Exercício 6.3



Congresso ([CodCongr](#), [NomeCongr](#))  
CongrGT ([CodCongr](#), [CodGT](#))  
GT ([CodGT](#), [NomeGT](#))  
Artigo ([CodCongr](#), [NumeroArt](#), [TitArt](#), [AssPrincArt](#))  
ArtigoAutor ([CodCongr](#), [NumeroArt](#), [CodAutor](#))  
Autor ([CodAutor](#), [NomeAutor](#))

#### Modelo do Exercício 6.4

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 O arquivo contém um registro para cada artigo submetido a congresso Lembre do exercício precedente, que 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 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 6.5.6.3). 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 6.5

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 6.6

Artigo ([CodCongr](#), [NumArtigo](#), [CodAutorPrinc](#), [TituloArtigo](#))  
Autor ([CodAutorPrinc](#), [NomeAutor](#), [InstituicaoAutor](#), [RuaNumAutor](#),  
[CidadeAutor](#), [EstadoAutor](#), [PaisAutor](#))  
Congresso ([CodCongr](#), [NomeCongr](#), [DataCongr](#), [LocalCongr](#),  
[DataLimCongr](#))

### Modelo do Exercício 6.7

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 6.6). 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 dentro da tabela [Sessão](#). Para detalhes ver Seção 6.6.2.

### Exercício 6.9

Artigo ([CodCongr](#), [NumeroArtigo](#) [TituloArt](#), [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 aparecem necessariamente juntas na tabela. Este é o caso da chave estrangeira CodCongr, Data, 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 estrangeira que referenciam a tabela Pessoa no exemplo.

#### **Exercício 6.10 e Exercício 6.11**

A Figura 7.22 apresenta do diagrama ER obtido através da engenharia reversa do modelo relacional obtido no exercício precedente.

Os atributos das entidades e 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)

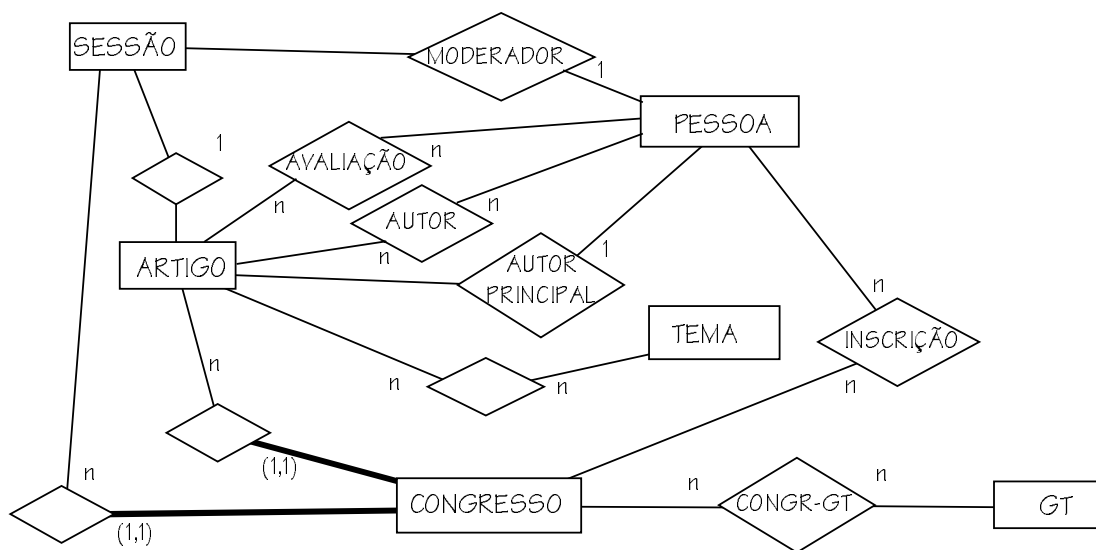


Figura 7.22: Diagrama ER obtido através de engenharia reversa

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 7.17. Com isso é possível comparar o diagrama obtido através da engenharia reversa com aquele obtido através da modelagem a partir da descrição do problema.

O modelo obtido por engenharia reversa é diferente daquele obtido pela modelagem a partir da descrição do problema (Figura 7.17). Vários elementos do modelo que haviam sido identificados na Figura 7.17 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 7.17, 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 abaixo.

A engenharia reversa não detecta relacionamentos redundantes (ver Seção 3.3.3). Este é o caso do relacionamento entre a entidade **SESSÃO** e a entidade **CONGRESSO**. O relacionamento é redundante, já 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 de 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 autores do artigo. Do ponto de vista

de 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 mais simples 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 6.5.6.1). 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 6.3** o fato de que 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 6.12**

ÑÑ

(NoCorr,  
(NoRecept, CodPeca, DescrPeca, QtdeEstoque))

Observação: O campo data de emissão do documento foi desconsiderado (ver Seção 6.5.6.3).

1FN

(NoCorr)  
(NoCorr , NoRecept, CodPeca, DescrPeca, QtdeEstoque)

2FN = 1FN

3FN

(NoCorr)  
(NoCorr , NoRecept, CodPeca, QtdeEstoque)  
(CodPeca, DescrPeca)

#### **Exercício 6.13**

ÑÑ

(CodOC, DataEntrega, NoEstrado, 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 6.5.6.3).

1FN

(CodOC, DataEntrega, NoEstrado, CodOperEmpilh)  
(CodOC, DataEntrega, NoCorr)  
(CodOC, DataEntrega, NoCorr, NoRecept, CodPeca, DescrPeca,  
QtdePeca)

2FN = 1FN

3FN

(CodOC, DataEntrega, NoEstrado, CodOperEmpilh)  
(CodOC, DataEntrega, NoCorr)  
(CodOC, DataEntrega, NoCorr, NoRecept, CodPeca, QtdePeca)  
(CodPeca, DescrPeca)

#### **Exercício 6.14**

ÑN

(NoPed, CodCli, NomeCli, NoRampa)

1FN=ÑN

2FN=1FN

3FN

(NoPed, CodCli, NoRampa)  
(CodCli, NomeCli)

#### **Exercício 6.15**

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 6.16**

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 Seção 6.5.1, quanto a opção de usar a decomposição de tabelas na passagem a 1FN. No caso do exemplo, os números de telefone que aparecem no exemplo, 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 após a construção do modelo ER.

### Exercício 6.17

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 6.18**

Os modelos normalizados de cada documento estão listados abaixo. Adicionalmente, cada tabela recebeu um nome. Isto facilita identificar tabelas que devam ser juntadas.

Modelo do Exercício 6.12

Corredor (NoCorr)

Receptáculo (NoCorr, NoRecept, CodPeca, QtdeEstoque)

Peca (CodPeca, DescrPeca)

Modelo do Exercício 6.13

Entrega (CodOC, DataEntrega, NoEstrado, CodOperEmpilh)

DistrCorr (CodOC, DataEntrega, NoCorr)

ItemDistr (CodOC, DataEntrega, NoCorr, NoRecept,  
CodPeca, QtdePeca)

Peca (CodPeca, DescrPeca)

Modelo do Exercício 6.14

Pedido (NoPed, CodCli, NoRampa)

Cliente (CodCli, NomeCli)

Modelo do Exercício 6.15

OC (NoOC, DataOC, CodFornec)

Fornec (CodFornec, NomeFornec)

ItemOC(NoOC, CodPeca, QuantPed)

Peca(CodPeca, DescrPeca)

ItemEntr (NoOC, CodPeca, DataEntr, QuantEntr)

Modelo do Exercício 6.16

Pedido (NoPed, DataPed, CodCli)

Cliente (CodCli, NomeCli)

Telefone (NoPed, NoTel)

ItemPedido (NoPed, CodPeca, QuantPed)

Peca (CodPeca, DescrPeca)

Modelo do Exercício 6.17

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, QtdeEstoque)  
 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:

- ❑ Quando da transcrição dos esquemas das tabelas ao modelo integrado, foi feito o tratamento de sinônimos (ver Seção 6.6). 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 6.6.2). A tabela **DistrCorr** - (modelo do Exercício 6.13) foi eliminada por estar contida dentro da tabela **ItemDistr** do mesmo exercício. Pela mesma razão, foi eliminada a tabela **BuscaCorredor** (modelo do Exercício 6.17). Deve ser observado que a tabela **Corredor** (modelo do Exercício 6.12) 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 6.6.2.

#### Exercício 6.19

Corredor (NoCorr)  
 Receptáculo (NoCorr , NoRecept, CodPeca, QtdeEstoque)  
     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 seria 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

# Índice Remissivo

## 0

0 (cardinalidade mínima), 20

## 1

1 (cardinalidade máxima), 16  
1 (cardinalidade mínima), 20  
1:1 (relacionamento), 17  
1:n (relacionamento), 17  
1FN. *Consulte* forma normal, primeira

## 2

2FN. *Consulte* forma normal, segunda

## 3

3FN. *Consulte* forma normal, terceira

## 4

4FN. *Consulte* forma normal, quarta

## A

abordagem entidade-relacionamento, 6, 11  
abordagem relacional, 75, 76  
adição de colunas, 92, 95, 96, 97, 98  
arquivo convencional, 76, 123  
atributo, 23, 89, 164, 166, 168  
  cardinalidade, 23  
  de entidade, 23  
  de relacionamento, 24  
  domínio, 165  
  identificador, 25, 89  
  implícito  
    na normalização, 140  
  mono-valorado, 23  
  multi-valorado, 23, 107, 160  
    quando usar, 52  
  nome de, 89  
  obrigatório, 23

opcional, 23  
  quando usar, 50  
quando usar, 48  
redundante, 54, 108  
temporal, 56  
auto-relacionamento, 14

## B

banco de dados, 3  
BD. *Consulte* banco de dados

## C

campo, 76  
  domínio, 80  
  mono-valorado, 76, 99  
  multi-valorado, 76, 99  
  nome, 76  
cardinalidade. *Consulte* relacionamento, cardinalidade  
CASE ("computer aided software engineering", 44, 59, 60, 62, 63, 74, 165  
chave, 77  
  alternativa, 79, 140  
  estrangeira, 78, 81, 92, 93, 172, 173, 184  
    engenharia reversa, 110, 111  
  estrangeria, 191  
  primária, 77, 80, 81, 87, 89, 91, 92, 100, 102, 130, 139, 172  
    engenharia reversa, 110, 113  
    nome, 89  
  secundária, 172  
coluna, 76, 89, 92, 129  
  domínio, 80  
  engenharia reversa, 113  
  mono-valorada, 76, 99  
  multit-valorada, 76, 99  
  multi-valorada, 123  
  nome de, 89  
  obrigatória, 80  
  opcional, 80, 88, 96, 102  
  redundante, 108  
compartilhamento de dados, 3  
consulta a banco de dados, 82

## D

dependência funcional, 129  
indireta. *Consulte* dependência funcional:  
transitiva  
multi-valorada, 137  
parcial, 130, 177  
transitiva, 133, 177  
DER. *Consulte* diagrama entidade-relacionamento  
diagrama de ocorrências, 13, 15, 158  
diagrama entidade-relacionamento, 6, 11  
exemplo, 22  
símbolos, 34  
variantes de notação, 59  
dicionário de dados, 23, 63  
domínio de campo. *Consulte* campo, domínio  
domínio de coluna, 80. *Consulte* coluna, domínio

## E

Engenharia de Informações  
notação, 60  
engenharia reversa, 8, 64, 85, 108  
de arquivo, 119, 120, 178, 179, 180, 181, 186,  
187, 188, 189  
de banco de dados, 109  
de BD relacional, 85  
de modelos relacionais, 109, 119, 175, 176, 184  
entidade, 7, 12, 89, 162, 163, 165, 167  
associativa, 32  
atributo, 23  
fraca, 26  
identificador de. *Consulte* identificador  
implementação na abordagem relacional, 89  
papel em relacionamento, 14  
quando usar, 48  
entidade-relacionamento  
abordagem. *Consulte* abordagem entidade-  
relacionamento  
diagrama. *Consulte* diagrama entidade-  
relacionamento  
modelo. *Consulte* modelo entidade-  
relacionamento  
ER. *Consulte* entidade-relacionamento  
especialização. *Consulte*  
generalização/especialização  
esquema de banco de dados, 5, 34  
relacional, 81  
notação, 82

## F

forma normal, 125  
não-normalizada, 139  
primeira, 125  
quarta, 136, 144  
segunda, 130, 144, 177, 178  
terceira, 133, 144, 177, 178  
fusão de tabelas, 93, 96, 97

## G

generalização/especialização, 28, 158, 161, 162  
exclusiva, 30  
herança múltipla, 30  
implementação na abordagem relacional, 100

não exclusiva, 31, 160  
parcial, 29  
quando usar, 49  
total, 29  
vários níveis, 30  
gerenciador de comunicação, 4  
gerenciador de interface de usuário, 4

## H

herança. *Consulte* generalização/especialização

## I

identificador  
atributo, 25  
composto de vários atributos, 25  
de entidade, 24, 26, 164, 166, 168  
de relacionamento, 27  
minimalidade, 26  
relacionamento, 25, 91  
implementação na abordagem relacional, 90  
inconsistência de dados, 3  
índice, 88  
instância. *Consulte* ocorrência  
integração de modelos, 141, 182, 189  
tabelas com a mesma chave, 142  
tabelas com chaves contidas, 143  
integração de visões. *Consulte* integração de  
modelos  
integridade referencial, 81

## J

junção, 82, 87

## L

linguagem de modelagem de dados, 5  
linguagem de terceira geração, 120  
linha, 76, 87

## M

MERISE  
notação, 61  
modelagem conceitual, 8, 11  
modelo  
conceitual, 6, 85, 109, 119, 120  
de dados, 5  
físico, 7  
lógico, 6, 8, 85, 109, 119, 120  
modelo conceitual, 11  
modelo entidade-relacionamento, 11, 85, 122  
construção, 63, 162, 163  
*bottom-up*. *Consulte* engenharia reversa  
engenharia reversa. *Consulte* engenharia  
reversa  
estratégia, 63  
*inside-out*, 65  
*top-down*, 64  
equivalência, 46  
exemplo, 22  
propriedades, 44  
aspecto temporal, 55, 164

capacidade limitada de expressão, 44  
correção, 53  
não ambíguo, 44  
representação gráfica. *Consulte* diagrama  
entidade-relacionamento  
representação textual, 34  
variantes, 59

## N

**n** (cardinalidade máxima), 16  
**n:n** (relacionamento), 17  
ÑN. *Consulte* tabela, não-normalizada  
normalização, 119, 121  
limitações, 145  
*null* (valor de campo), 80

## O

ocorrência  
de entidade, 12  
de relacionamento, 13

## P

papel em relacionamento, 14  
primeira forma normal. *Consulte* forma normal,  
primeira  
projeto  
de banco de dados, 8  
lógico, 8, 85, 109  
passos, 88  
visão geral, 86  
projeto lógico, 174

## Q

quarta forma normal. *Consulte* forma normal, quarta

## R

redigitação, 3  
redundância de dados, 3  
controlada, 3  
não controlada, 3  
redundância em modelos ER. *Consulte* atributo,  
redundante. *Consulte* relacionamento,  
redundante  
relação. *Consulte* tabela  
relacionamento, 13, 92, 93, 157, 162, 163, 165, 167  
**1**  
**\n**, 97  
**1:1**, 17, 93, 95, 96  
**1:n**, 17, 18, 19  
atributo, 24  
auto, 19  
auto-, 14

binário, 17  
cardinalidade, 15  
em relacionamento ternário, 20  
máxima, 16, 93, 163  
**1**, 16, 93  
**n**, 16  
mínima, 20, 22, 93, 161, 165  
**0**, 20  
**1**, 20  
de grau maior que dois, 19, 99  
identificador. *Consulte* identificador  
identificador de, 27  
implementação na abordagem relacional, 91, 93  
mútua exclusão de participação em, 105  
**n:n**, 17, 19, 99  
transformação em entidade, 47, 99, 159  
papel de entidade, 14  
recursivo, 46, 161  
redundante, 54  
temporal, 56  
ternário, 19, 99, 158  
restrição de integridade, 65, 80, 166, 168, 171  
semântica, 81

## S

segunda forma normal. *Consulte* forma  
normal:segunda  
SGBD. *Consulte* sistema de gerência de banco de  
dados  
sintonia de banco de dados, 7  
sistema de gerência de banco de dados, 4, 6  
pré-relacional, 120  
relacional, 6, 75, 85, 87, 88, 89  
linguagem, 82, 89  
sistema legado, 120  
SQL, 82

## T

tabela, 6, 76, 87, 89, 92, 93, 100  
aninhada, 123, 125  
engenharia reversa, 110  
não-normalizada, 122  
não-primeira-forma-normal. *Consulte* tabela  
não-normalizada  
própria, 92  
propriedades, 76  
tabela própria, 95, 98, 99  
terceira forma normal. *Consulte* forma normal,  
terceira  
tupla. *Consulte* linha

## V

vazio (valor de campo), 80, 81, 88

Este livro é distribuído GRATUITAMENTE pela equipe DIGITAL SOURCE e VICIADOS EM LIVROS com a intenção de facilitar o acesso ao conhecimento a quem não pode pagar e também proporcionar aos Deficientes Visuais a oportunidade de apreciar mais uma manifestação do pensamento humano.

Se quiser outros títulos nos procure.

Será um prazer recebê-lo em nosso grupo.



[http://groups-beta.google.com/group/Viciados\\_em\\_Livros](http://groups-beta.google.com/group/Viciados_em_Livros)

<http://groups-beta.google.com/group/digitalsource>