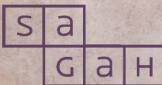


# BANCO DE DADOS

Ednilson da Silva Vida



SOLUÇÕES EDUCACIONAIS INTEGRADAS

# Projeto de banco de dados: modelos conceitual, lógico e físico

## Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir os modelos conceitual, físico e lógico.
- Converter modelos de banco de dados conceitual, lógico e físico.
- Ilustrar a modelagem de banco de dados relacional com SQL.

## Introdução

O conhecimento acerca dos modelos entidade-relacionamento (MERs) possibilita a criação de pequenos e grandes projetos lógicos de bancos de dados e de modelagens de alta qualidade, propiciando o sucesso dos projetos de *software* ou aplicação. Assim, neste capítulo, você vai estudar os tipos e modelos de bancos de dados, com destaque para o modelo relacional, e vai aprender sobre o projeto desses bancos de dados. Você também vai compreender como ocorre o processo de transformação do MER para o modelo relacional e vai verificar os modelos conceitual, físico e lógico, compreendendo o processo de conversão entre esses modelos. Por fim, você vai estudar a modelagem de banco de dados utilizando a linguagem de consulta estruturada (SQL, do inglês *structured query language*).

## 1 Modelagem e projeto de banco de dados

Um dos momentos mais críticos no processo de desenvolvimento de um *software* é a **modelagem de banco de dados**, pois o produto deve atingir os objetivos estabelecidos pelo requisitante. Segundo Heuser (2009), previamente à construção de bancos de dados, são utilizados padrões em textos e gráficos

para modelar, sendo propostos três níveis de abstração de dados: modelo conceitual, modelo lógico e modelo físico. Como muitos usuários de banco de dados são leigos nas técnicas de informática, faz-se necessário simplificar em um projeto a sua estrutura, para oferecer uma visão geral dos dados com os aspectos de interesse, possibilitando bancos de dados flexíveis (COUGO, 1997).

Os requisitos podem ser descritos graficamente por **diagramas**, com declarações sobre as funções que o sistema deve oferecer de forma abstrata de alto nível. Leva-se também em consideração a **engenharia de requisitos**, que é um processo que engloba todas as atividades que contribuem para a elaboração de um documento, com todos os requisitos, e para a sua manutenção. A etapa de engenharia de requisitos é a fase de descobrir, analisar e verificar funções e restrições (GIMENES; HUZITA, 2005).

Um erro durante a modelagem compromete a usabilidade do sistema final e acarreta a necessidade de retrabalho, o que aumenta o custo do processo de desenvolvimento. Para que isso não ocorra, a seguir serão apresentados os passos fundamentais do processo de modelagem de um banco de dados, conforme as fases apresentadas na Figura 1.



**Figura 1.** Etapas de modelagem de dados.

Fonte: Adaptada de Cougo (1997).

## Identificação do problema (levantamento de requisitos)

Nesta etapa, é realizado um estudo detalhado das atividades em questão. Quando não há conhecimento prévio sobre o negócio, entrevistas podem levantar informações relevantes sobre as necessidades dos futuros usuários. Os administradores de dados se reúnem com os usuários para entender e documentar seus requisitos.

Os requisitos são a base de todos os produtos de *software*. Sua elucidação, seu gerenciamento e seu entendimento são problemas comuns a todas as metodologias de desenvolvimento. Segundo Pressman e Maxim (2011), a tarefa de **análise de requisitos** é um processo de descoberta, refinamento, modelagem e especificação. A análise de requisitos proporciona ao projetista de *software* uma representação da informação e da função, que pode ser traduzida no projeto procedural, arquitetônico e de dados, oferecendo ao desenvolvedor e ao cliente os critérios para avaliar a qualidade logo que o sistema for construído.

### Modelagem conceitual (alto nível)

A **modelagem conceitual** é a representação que considera exclusivamente o ponto de vista do usuário criador dos dados, levando em consideração fatores técnicos para sua implementação. O nível conceitual especifica como os dados são armazenados e relacionados, independentemente de como serão implementados no banco de dados.

Para Heuser (2009, p. 25), “[...] a técnica de modelagem conceitual mais difundida é a abordagem **entidade-relacionamento**. Nessa técnica, um modelo conceitual é usualmente representado através de um diagrama”. O MER utiliza elementos gráficos para descrever o modelo de dados de uma aplicação com alto nível de abstração (CALIARI, 2007), identificando **entidades, atributos e relacionamentos**. Peter Chen, em 1976, idealizou uma notação para realizar a modelagem de dados para ambientes relacionais.

### Modelagem lógica (representativa ou de implementação)

O **modelo lógico** só deve ser inicializado após a conclusão do modelo conceitual. Diferentemente do modelo conceitual, o modelo lógico será criado com base em um tipo de banco de dados, como SQL Server, Oracle, MySQL, dentre outros.

Muitos analistas não aceitam que a etapa do modelo conceitual seja importante, acreditando que ela é desnecessária. Devido aos prazos curtos dos projetos, tais analistas não criam o modelo conceitual e iniciam o projeto com o desenvolvimento do modelo lógico. Porém, no fim, muitos se dão conta de que nem todo requisito ou solicitação ficou completo ou foi atendido corretamente, o que poderia ser facilmente criado e interpretado na elaboração do modelo conceitual.

De acordo com Heuser (2009) e Machado (2014), o modelo lógico descreve e mapeia as estruturas que estarão presentes no banco de dados, de acordo com as características da abordagem. Evitam-se:

- muitas tabelas;
- tempo longo de resposta nas consultas e atualizações de dados;
- desperdício de espaço;
- muitos controles de integridade no banco de dados; e
- muitas dependências entre dados.

### **Modelagem física (baixo nível)**

O **modelo físico** é concebido por meio do modelo lógico. É nesse modelo que serão definidos os tipos de dados que serão armazenados, e ocorre a implementação da estrutura lógica em um **sistema gerenciador de banco de dados** (SGBD), que administra fisicamente os dados armazenados.

Esse modelo se resume à SQL, que é a linguagem necessária para gerenciar um banco de dados relacional (OLIVEIRA, 2002). Nele, são detalhados os componentes da estrutura física do banco, como tabelas, campos, tipos de valores, índices etc. Entram em cena os detalhes técnicos do projeto, atendendo à necessidade do cliente e já implantando a política de cópia e segurança. Nessa fase, há a geração das instruções em código SQL, que vão criar a base de dados do sistema. Por isso, nesse ponto, a tecnologia aplicada assume lugar primordial, pois a parte de negócios já foi definida e estabelecida.

## **2 Modelo entidade-relacionamento**

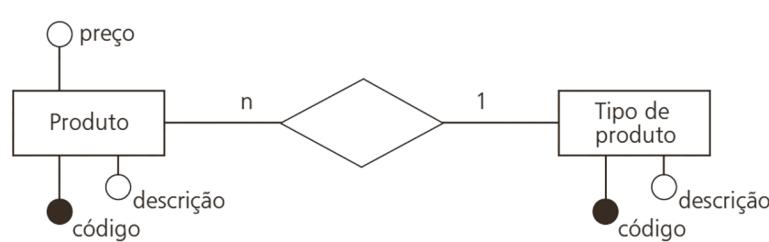
O modelo conceitual tem como objetivo solucionar problemas do mundo real, criando elementos globais e interligando-os por meio de estruturas de informação. Para a criação de um banco de dados, devemos iniciar primeiramente pelo modelo conceitual. Mas, por que o modelo conceitual é o primeiro a ser criado? Justamente porque ele tem por obrigação demonstrar ao usuário final, que nem sempre terá um conhecimento de banco de dados, quais são a estrutura e as regras de negócios que serão implementadas no banco. O modelo conceitual deve atender a todo o processo de solicitações cotidianas, para que se possa ter esses dados estruturados fisicamente.



## Fique atento

É importante destacar que o modelo conceitual não é associado e não faz parte de nenhum SGBD. O modelo conceitual é útil para identificar e analisar se as regras de negócio estão bem definidas e, dessa forma, evitar erros futuros.

Dentre as técnicas de modelagem de dados existentes, as mais conhecidas e utilizadas são a do modelo entidade-relacionamento (MER) (conceitualmente) e do diagrama entidade-relacionamento (DER) (visualmente). No entanto, independentemente do tipo de representação do modelo, faz-se necessário o entendimento de alguns conceitos referentes aos componentes, como: entidade, atributos, relacionamento, cardinalidades, entre outros. A Figura 2 demonstra de forma gráfica a criação de um modelo conceitual utilizando um DER.



**Figura 2.** Modelagem conceitual utilizando um diagrama entidade-relacionamento.

*Fonte:* Heuser (2009, p. 26).

## Entidades

A **entidade**, também conhecida como **tabela**, é uma representação gráfica de um conjunto ou objeto de informações. Segundo Heuser (2009), entidade é um conjunto modelado de objetos da realidade sobre os quais se deseja manter informações no banco de dados. Dessa forma, o banco de dados é separado por entidades (tabelas). Para identificarmos uma entidade, devemos considerar os objetos, as coisas ou algo que seja relevante no levantamento de dados.

Uma entidade é representada em um modelo conceitual por meio de um **retângulo**, com o nome da tabela ao centro dele. Essa entidade terá uma ou várias informações. Cada ocorrência dessas informações é chamada de **instância** e vai representar um conjunto exclusivo dos dados. Veja a seguir algumas definições.

- **Entidade forte:** são aquelas cuja existência não depende de outras entidades, ou seja, elas já possuem total sentido de existir. Em um sistema de notas, a entidade Alunos, por exemplo, independe de quaisquer outras para existir.
- **Entidade fraca:** ao contrário das entidades fortes, as fracas são aquelas que dependem de outras entidades para existirem, pois individualmente elas não fazem sentido.

Cabe ressaltar que, nos nomes atribuídos às entidades, devem ser levados em consideração os substantivos da frase, principalmente no caso de esse substantivo ser relevante ao sistema, de modo a transformá-lo em uma entidade.

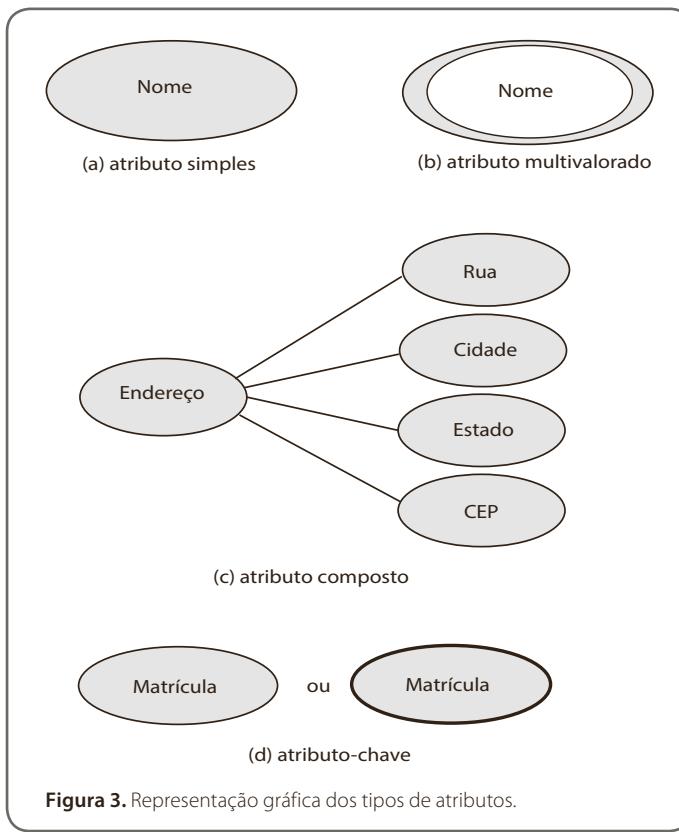
## Atributos

Podemos definir **atributos** como informações ou descrições das entidades. Podemos dizer ainda que eles são as **colunas** da tabela, já que cada atributo se atém ao armazenamento de uma informação específica. Heuser (2009) define atributo como um dado que é associado a cada ocorrência de uma entidade ou relacionamento.

Podemos classificar os atributos como simples, multivalorados e compostos. Em DERs, utilizamos principalmente os atributos simples e multivalorados. Nesse sentido, definimos **atributos simples** (Figura 3a) como aqueles que contêm apenas um valor para cada elemento da entidade. Por exemplo, no caso de uma entidade chamada Alunos, cada registro terá o nome de um aluno.

Os **atributos multivalorados** (Figura 3b) são aqueles que suportam vários registros. Eles são a solução do problema quando, por exemplo, têm-se vários telefones para um aluno. Já os **atributos compostos** (Figura 3c) nos permitem indicar um atributo que pode ser dividido em outros. Por exemplo, no caso de um endereço, pode-se dividi-lo em rua, cidade, estado e CEP.

Um **atributo-chave** (Figura 3d) ocorre quando, entre os atributos, faz-se necessário informar qual será o atributo de identificação, sendo este único em toda a tabela e nunca nulo — isto é, com preenchimento vazio. Como exemplo, pode-se dizer que a informação Matrícula é um atributo-chave. A representação gráfica de cada um dos tipos de atributos é apresentada na Figura 3.



**Figura 3.** Representação gráfica dos tipos de atributos.

Para Heuser (2009, p. 49), “[...] os atributos também podem possuir uma cardinalidade, de maneira análoga a uma entidade em um relacionamento. A cardinalidade de um atributo define quantos valores deste atributo podem ser associados a uma ocorrência de entidade/relacionamento a qual ele pertence”. Outro conceito muito importante a ser abordado em relação aos atributos é referente ao domínio. As **restrições de domínio** especificam que o valor de cada atributo deve ser um valor atômico; para cada atributo criado, deve-se associar um tipo a ele.

Além do atributo-chave, é necessário entender os conceitos relacionados aos atributos que se tornam **chaves estrangeiras** (do inglês *foreign key*). Essa chave consiste em um campo que aponta para a chave primária (atributo-chave, ou *primary key*) de outra tabela. No entanto, nessa relação entre linhas de duas entidades, o objetivo da chave estrangeira é garantir a integridade dos dados referenciais, pois, nesses casos, serão permitidos valores que não aparecerão na base de dados.



### Saiba mais

Após estabelecer uma chave estrangeira, o atributo marcado não permitirá a exclusão, a inserção ou a modificação de dados em tabelas que estejam dependentes uma das outras, exigindo maior atenção dos administradores do banco de dados.

A Figura 4 ilustra um exemplo da aplicação de uma chave estrangeira entre duas tabelas. Podemos notar que o campo *Aluno\_Curso* é a chave estrangeira na tabela *Aluno*.

Tabela: Curso	
CURSO_COD	CURSO_DESCRÍÇÃO
1	GEOGRAFIA
2	BIOLOGIA
3	AGRONOMIA

↓

ALUNO_MATRÍCULA	ALUNO_NOME	ALUNO_CURSO	ALUNO_ESTADO
1000	RICARDO VIEIRA	1	SÃO PAULO
1010	LUCAS SOUZA	2	BAHIA
1110	WILLIAM GOMES	3	ACRE

**Figura 4.** Aplicação de chave estrangeira entre duas tabelas.

## Relacionamentos

Resgatando-se os conceitos do **modelo relacional**, nota-se que as entidades não podem ficar isoladas, uma vez que as informações serão organizadas para o acesso de forma integrada. Assim, para que essa organização não tenha perda de conteúdo, as entidades precisam estar integradas entre si. A forma de ligação entre as entidades é por meio de **relacionamentos**. Heuser (2009) define relacionamento como um conjunto de associações entre ocorrências de entidades. Em um DER, um relacionamento é representado por um **losango**, ligado por linhas aos retângulos representativos das entidades que participam do relacionamento (HEUSER, 2009).

Ao se realizar a modelagem de dados, nem sempre se torna claro o nome de um relacionamento entre duas entidades. Uma das formas de facilitar essa escrita é por meio da utilização de **verbos**, pois eles permitem correlacionar as informações entre duas entidades e, de certa forma, associá-las.

Podemos encontrar vários tipos de relacionamentos na literatura, porém, estes normalmente são classificados de acordo com a quantidade de entidades associadas a eles. Ou seja, o número de entidades que participam em um conjunto de relacionamentos é o que determina o grau desse conjunto. Os tipos de relacionamentos existentes são, segundo Heuser (2009): autorrelacionamentos, relacionamento binário e relacionamento ternário e relacionamento contendo um atributo.

O **autorrelacionamento** (Figura 5a), também denominado de **relacionamento recursivo**, refere-se a um relacionamento composto de apenas uma entidade. Segundo Heuser (2009), no caso do relacionamento de casamento da Figura 5a, uma ocorrência de pessoa exerce o papel de marido e a outra ocorrência exerce o papel de esposa.

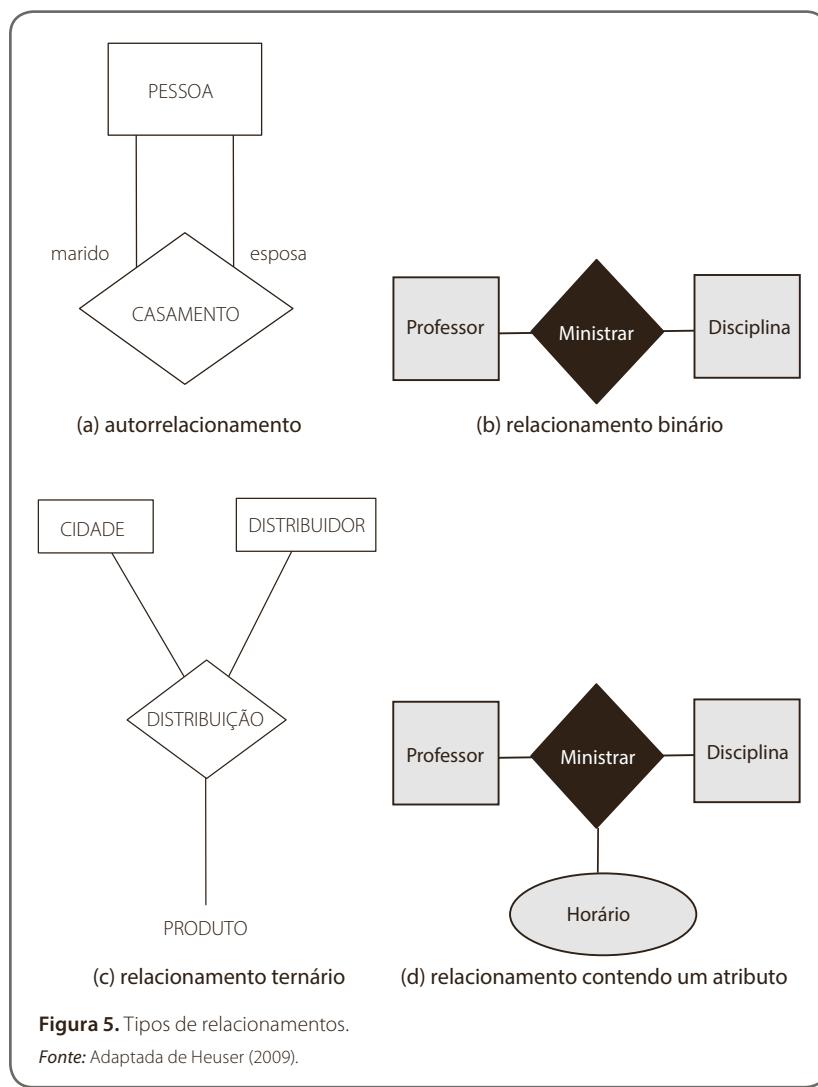
O **relacionamento binário** (Figura 5b) é definido como de grau dois, devido a este ter duas entidades. Já o **relacionamento ternário** (Figura 5c) é chamado relacionamento de grau três, pois apresenta três entidades associadas no relacionamento. Para Heuser (2009), cada ocorrência do relacionamento Distribuição, da Figura 5c, associa três ocorrências de entidade: um produto a ser distribuído, uma cidade na qual é feita a distribuição e um distribuidor.



### Fique atento

Cabe ressaltar que, entre duas entidades, também pode ocorrer mais de um relacionamento — ou seja, uma entidade pode estar associada a outra por mais de um relacionamento.

Outra particularidade de um relacionamento é que este **pode conter atributos** (Figura 5d). Estes não fazem parte obrigatória das propriedades das entidades; porém, quando é inserido um atributo, este é associado a um relacionamento e deve ser comum às duas entidades. Assim, notamos que, na Figura 5d, o atributo Horário é parte comum às entidades associadas no relacionamento e informa em que horário o professor ministra a referida disciplina.



**Figura 5.** Tipos de relacionamentos.

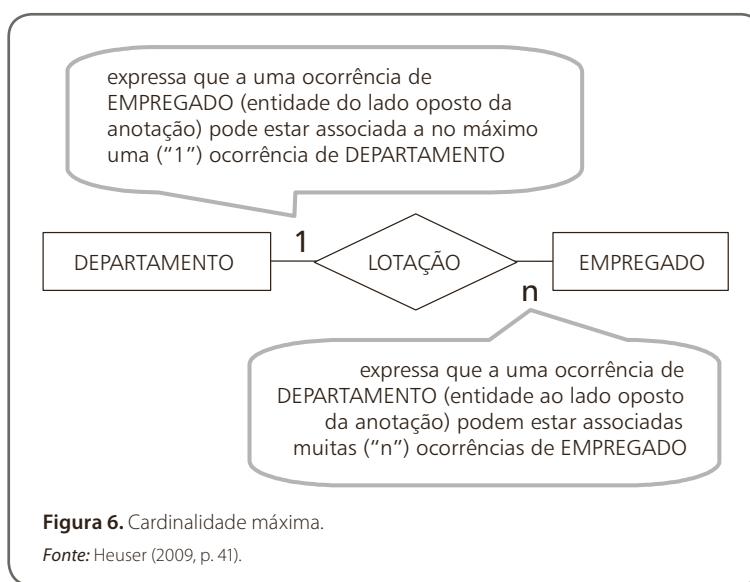
Fonte: Adaptada de Heuser (2009).

## Cardinalidades

Por meio da **cardinalidade**, é possível expressar o número de ocorrências com que uma entidade pode tomar parte em um relacionamento. Ela possibilita também expressar as possibilidades e as restrições de associações entre uma entidade e outras. Heuser (2009, p. 39) diz que “[...] uma propriedade importante de um relacionamento é a de quantas ocorrências de uma entidade podem estar associadas a uma determinada ocorrência através do relacionamento”.

Essa propriedade é denominada de cardinalidade de uma entidade, podendo ser classificada de duas formas: cardinalidade mínima e cardinalidade máxima. Ainda para Heuser (2009, p. 39), “[...] a cardinalidade (mínima ou máxima) de entidade em um relacionamento é o número (mínimo ou máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento”.

A Figura 6 demonstra um exemplo de cardinalidade máxima em um DER.



A **cardinalidade máxima** aborda o limite máximo de ocorrências de uma entidade em relação à outra, da seguinte forma:

- um para um (1:1);
- um para muitos (1:N);
- muitos para muitos (N:N ou N:M).

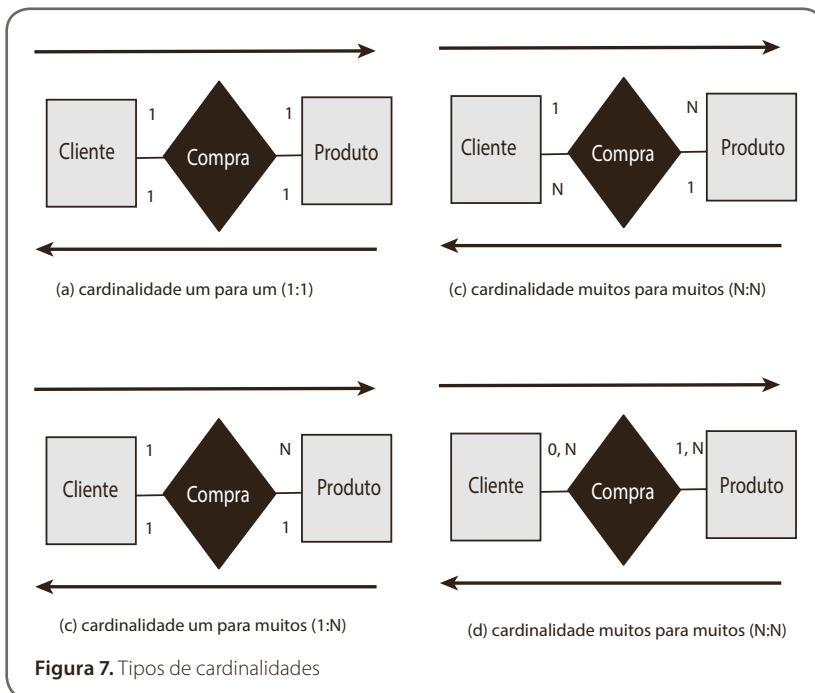
A cardinalidade 1:1 (Figura 7a) acontece quando a ocorrência de uma entidade se relaciona com (no máximo) uma ocorrência de outra e vice-versa. Já na cardinalidade 1:N (Figura 7b), a ocorrência de uma entidade se relaciona com (no máximo) muitas ocorrências de outra; porém, a ocorrência de outra entidade se relaciona com (no máximo) uma ocorrência da primeira.

Na cardinalidade N:N (Figura 7c), a ocorrência de uma entidade se relaciona com (no máximo) muitas ocorrências de outra entidade e vice-versa. Cabe ressaltar que, quando a leitura é feita 1:N, em ambos os sentidos das entidades, o resultado é apresentado como N:N.

Podemos identificar que a **cardinalidade mínima** abrange apenas o mínimo de ocorrências de uma entidade em relação à outra. Ela é classificada conforme descrito a seguir.

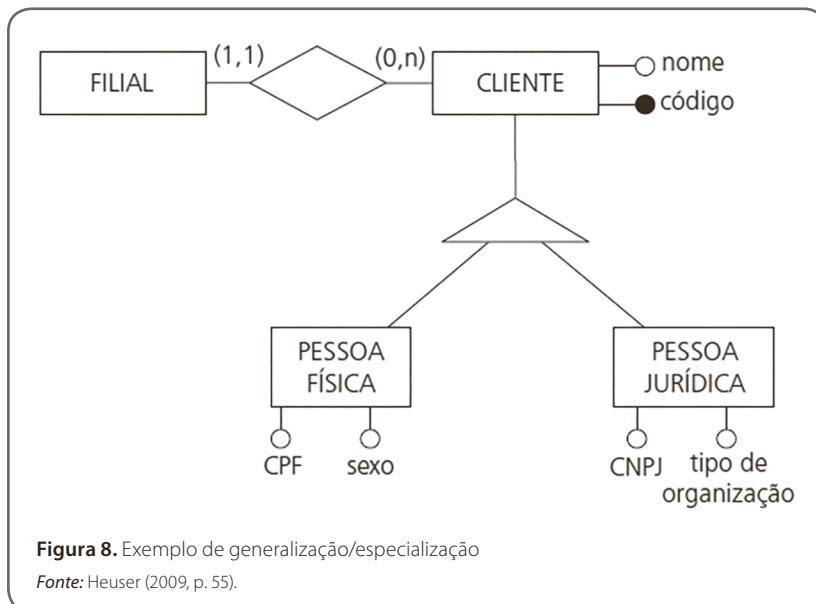
- **Opcional (0):** uma ocorrência se relaciona com (no mínimo) nenhuma outra entidade e pode ser representada como:
  - **0:1** — a representação textual se dá por “no mínimo, nenhuma ocorrência em uma entidade, para, no máximo, uma ocorrência na outra entidade”.
  - **0:N** — a representação textual se dá por “no mínimo, nenhuma ocorrência em uma entidade, para, no máximo, muitas ocorrências na outra entidade”.
- **Obrigatória (1):** uma ocorrência se relaciona com (no mínimo) uma de outra entidade.

No exemplo da Figura 7d, notamos que a regra de negócio, na cardinalidade mínima, marcada como zero, significa que o cliente não é obrigatório no momento da venda do produto e que o produto é comprado por, no máximo, um cliente.

**Figura 7.** Tipos de cardinalidades

Conforme visto anteriormente nos relacionamentos, podemos ter, além de relacionamentos binários, também os ternários; dessa forma, a cardinalidade se faz necessária. A forma de atribuição de cardinalidade em associações ternárias segue as mesmas características que as binárias, porém, com algumas considerações.

Para Heuser (2009, p. 54), “[...] além de relacionamentos e atributos, propriedades podem ser atribuídas a entidade através do conceito de generalização/especialização. Por meio da generalização/especialização é possível atribuir propriedades particulares a um subconjunto das ocorrências (especializadas) de uma entidade genérica”. A Figura 8 ilustra um exemplo de **generalização/especialização**, em que podemos encontrar todas as propriedades da ocorrência da entidade CLIENTE correspondentes às entidades PESSOA FÍSICA e PESSOA JURÍDICA. Dessa forma, a generalização/especialização proporciona agregar todos os atributos da entidade CLIENTE (entidade genérica) aos atributos da entidade PESSOA FÍSICA ou PESSOA JURÍDICA (entidade especializada).



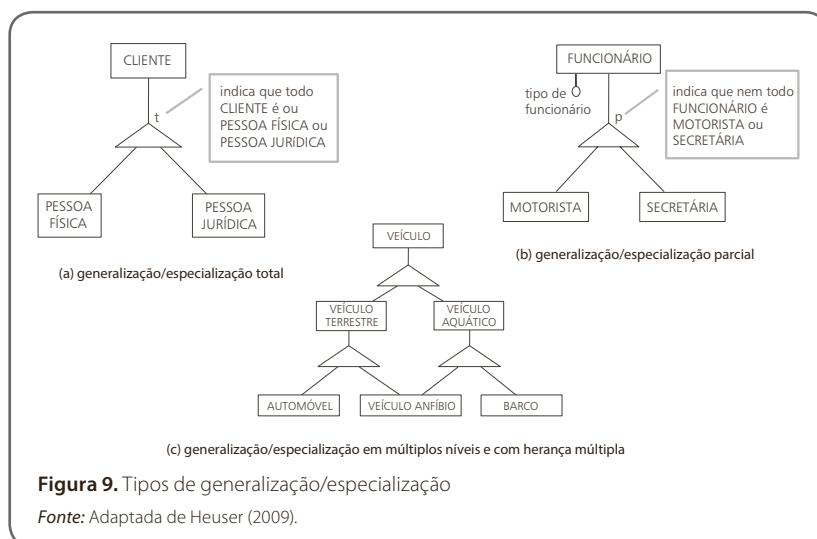
**Figura 8.** Exemplo de generalização/especialização

Fonte: Heuser (2009, p. 55).

No entanto, ao utilizarmos generalização/especialização, precisamos identificar se esta é **parcial ou total**. Isso é feito de acordo com a obrigatoriedade (ou não) de uma ocorrência da entidade especializada ser correspondida pela ocorrência da entidade genérica. Para Heuser (2009, p. 56), “[...] em uma generalização/especialização total para cada ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas”, conforme ilustrado na Figura 9a, em que toda ocorrência da entidade CLIENTE corresponde a uma ocorrência em uma das duas especializações. Por sua vez, em uma generalização/especialização parcial, observa-se, pela Figura 9b, que nem toda ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada.

Além das generalizações/especializações total e parcial, também se encontra a classificação compartilhada e exclusiva. A **generalização/especialização exclusiva**, segundo Heuser (2009), significa que, em uma hierarquia de generalização/especialização, uma ocorrência de entidade genérica é especializada no máximo uma vez, nas folhas da árvore de generalização/especialização. Observe a Figura 9b, em que uma instância FUNCIONÁRIO aparece uma vez somente nas entidades especializadas (MOTORISTA ou SECRETÁRIA), já que um funcionário ou é motorista ou é secretária, mas não ambas as funções ao mesmo tempo.

Já a **generalização/especialização compartilhada** indica que “[...] em uma hierarquia, uma ocorrência de entidade genérica pode aparecer em várias entidades nas folhas da árvore de generalização/especialização” (HEUSER, 2009, p. 57). Um exemplo de generalização/especialização compartilhada é quando consideramos o conjunto de pessoas vinculadas a uma universidade. “Neste exemplo, a especialização é compartilhada, já que a mesma pessoa pode aparecer em múltiplas especializações, isto é, uma pessoa pode ser professor e aluno ou funcionário e aluno ao mesmo tempo” (HEUSER, 2009, p. 57). No entanto, uma entidade pode ser especializada em qualquer número de entidades, inclusive em uma única, conforme ilustra a Figura 9c.



**Figura 9.** Tipos de generalização/especialização

## Entidade associativa

Para Heuser (2009), um relacionamento é uma associação entre entidades. No entanto, na modelagem entidade-relacionamento, não foi prevista a possibilidade de associar uma entidade com um relacionamento ou de associar dois relacionamentos entre si. Na prática, ao construir um novo MER, ou, então, ao modificar um MER existente, surgem algumas situações em que é desejável permitir a associação de uma entidade a um relacionamento.

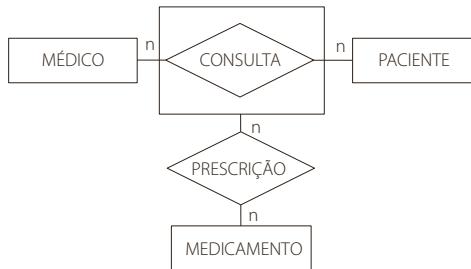
Considerando o MER ilustrado na Figura 10a, notamos que é necessário saber quais medicamentos existem, além da informação de quais medicamentos tiveram prescrição em cada consulta. Nesse ponto, precisamos questionar qual entidade existente deve estar relacionada a essa nova entidade (HEUSER, 2009). A necessidade de solucionar essas questões provocou a criação do conceito de **entidade associativa**. Heuser (2009, p. 60) define entidade associativa como “[...] a redefinição de um relacionamento, que passa a ser tratada como se fosse também uma entidade”.

Podemos verificar a representação gráfica de uma entidade associativa na Figura 10b. Pode-se notar que o “[...] retângulo desenhado ao redor do relacionamento CONSULTA indica que este relacionamento passa a ser visto como uma entidade associativa, já que é baseada em um relacionamento” (HEUSER, 2009, p. 61). Note que, caso não se desejasse usar o conceito de entidade associativa, seria preciso transformar o relacionamento CONSULTA em uma entidade, que então poderia ser relacionada a MEDICAMENTO. Veja a representação na Figura 10c.

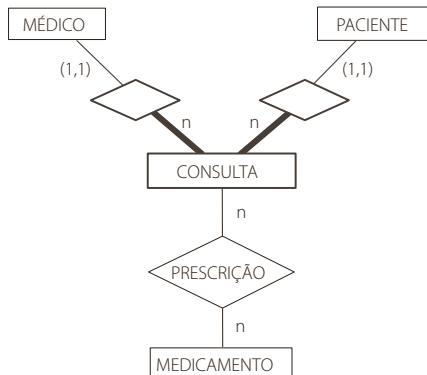
Nota-se que o diagrama da Figura 10b é semelhante ao da Figura 10c, pois ambos geram o mesmo banco de dados relacional. Em MERs, é importante destacar a forma de representação, isto é, como esse modelo pode ser representado de forma gráfica e textual.



(a) modelo entidade-relacionamento ao ser modificado



(b) entidade associativa



(c) substituindo relacionamento por entidade.

**Figura 10.** Necessidade, utilização e substituição de uma entidade associativa.

*Fonte:* Adaptada de Heuser (2009).

### 3 Modelagem de banco de dados relacional

Um **banco de dados relacional** é composto de tabelas, também conhecidas como relações. No entanto, a terminologia **tabela** é mais utilizada nos produtos comerciais e na prática, enquanto o termo **relações** é mais adotado na área acadêmica. Para Heuser (2009), uma tabela é um conjunto não ordenado de **tuplas** ou linhas, e estas são compostas de uma série de campos, ou seja, **valores de atributos**.

Quando nos referimos a um banco de dados relacional, este é composto de tabelas, chaves (primárias, estrangeiras e alternativas), domínios, valores vazios e restrições de integridade. É por meio das chaves que há a identificação de linhas. Além disso, as chaves são responsáveis pelas relações entre as linhas, as tabelas e o banco de dados. As chaves mais conhecidas e utilizadas em um banco de dados relacional são as chaves primárias, as chaves estrangeiras e as chaves alternativas.

Como chave primária, segundo Heuser (2009, p. 122), define-se “[...] uma coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela”. A Figura 11 ilustra um exemplo de chave primária na tabela Empregado. Nota-se, nesse exemplo, que o campo *CodigoEmp* é a chave primária da tabela — ou seja, essa coluna se distingue das demais. No entanto, as chaves primárias podem ser definidas como únicas, ou seja, apenas um campo da tabela, como também podem ser compostas.

Dept	
CodigoDepto	NomeDepto
D1	Compras
D2	Engenharia
D3	Vendas

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

**Figura 11.** Tabelas com chaves primária e estrangeira.

*Fonte:* Heuser (2009, p. 121).

Assim, para qualquer combinação de colunas, para as chaves primárias compostas, faz-se necessário que estas sejam mínimas. Conforme Heuser (2009, p. 123), “[...] uma chave mínima é quando todas as colunas forem efetivamente necessárias para garantir o requisito de unicidade de valores da chave”. Ainda segundo Heuser (2009, p. 123), “[...] 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 de banco de dados relacional”. A Figura 11 já apresentada ilustra um exemplo de chave estrangeira. Nele, a coluna CódigoDeptº da tabela Emp é uma chave estrangeira em relação à chave primária da tabela Deptº.

Em alguns casos, mais de uma coluna ou combinações de colunas podem servir para diferenciar uma linha das outras. Assim, uma das colunas pode ser escolhida como chave primária, e as demais são definidas como chaves alternativas, conforme demonstrado na Figura 12. No exemplo da Figura 12, são definidas duas colunas que podem ser utilizadas para representar as demais linhas — no entanto, a coluna CódigoEmp foi escolhida para ser chave primária.

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

**Figura 12.** Chave alternativa (coluna CPF).

Fonte: Heuser (2009, p. 126).

Os domínios e valores vazios são utilizados em um banco de dados relacional para especificar um conjunto de valores (numéricos, alfanuméricos, entre outros) que os campos da respectiva coluna podem assumir. Esse conjunto de valores é chamado **domínio da coluna** ou **domínio do campo**. Outra característica importante é definir se os campos devem ser vazios ou não. Quando o campo é definido como vazio, isso significa que ele não recebeu valor de seu domínio. Já em relação às restrições de integridade, entende-se que elas são primordiais para um SGBD, pois permitem manter o controle e a manutenção de um banco de dados. A **integridade** de um banco de dados significa dizer que os dados refletem corretamente a realidade representada no banco e que eles são consistentes entre si.

Dentre as diferentes representações de um modelo de banco de dados relacionais, encontram-se o **esquema textual** de banco de dados relacional e o esquema diagramático de banco de dados relacional. Sobre a primeira notação, diz-se que é incompleta, mas compacta, e é útil para exemplos como os mostrados neste capítulo, assim como para discussões sobre a estrutura geral do banco de dados, quando não se deseja entrar no maior nível de detalhe. A **representação diagramática** é composta por retângulos que simbolizam as tabelas e, dentro deles, apresentam os atributos ou colunas que representam as tabelas. Assim, um esquema textual do banco de dados relacional é apresentado na Figura 13.

		Dept			
		CodigoDepto	NomeDepto		
		D1	Compras		
		D2	Engenharia		
		D3	Vendas		

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

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

**Figura 13.** Esquema textual do banco de dados referente às tabelas Dept e Emp.

Fonte: Adaptada de Heuser (2009).

Com base nesse esquema textual, torna-se possível, também, transformar um **modelo relacional em modelo lógico**. Por exemplo, para criar uma tabela CLIENTE, teríamos a seguinte representação em esquema textual: CLIENTE (CPF, nome, rua, bairro, cidade, estado, sexo, data de nascimento). Para essa tabela, temos a opção de definir apenas o campo CPF como chave primária, pois é o único atributo que temos certeza de que jamais se repetirá para clientes diferentes, pois não é possível que pessoas diferentes tenham CPFs iguais. A criação dessa tabela utilizando o modelo relacional e utilizando a linguagem SQL é apresentada na Figura 14.

```
1 CREATE TABLE CLIENTE (
2   cpf NUMBER(11) PRIMARY KEY,
3   nome VARCHAR(500) NOT NULL,
4   rua VARCHAR(250),
5   bairro VARCHAR(250),
6   cidade VARCHAR(150),
7   estado VARCHAR(2),
8   sexo CHAR(1),
9   dataNascimento DATE
10 )
```

**Figura 14.** Instrução em SQL referente à criação da tabela CLIENTE.

Na Figura 14, o comando `CREATE TABLE` serve para a criação de uma nova tabela no banco de dados. Nesse comando, o primeiro parâmetro a ser definido é o nome da tabela que está sendo criada, seguido dos atributos, de seus respectivos tipos e das eventuais restrições do atributo. Conforme descrito anteriormente, ao criar uma tabela, é necessário identificar os atributos e os seus respectivos tipos.



### Fique atento

Note que, em linguagens de programação, ao criarmos uma variável, devemos identificar o seu respectivo tipo de dado. Na linguagem SQL isso também ocorre, porém, os tipos existentes são mais restritos se comparados à quantidade disponibilizada na linguagem de programação.

Ainda no exemplo da Figura 14, pudemos ver que será obrigatório o preenchimento dos campos CPF e nome. O CPF é obrigatório por ser a chave primária, e o nome, por ter a cláusula `NOT NULL` (não nula) em sua descrição. As chaves primárias, por padrão, já são não nulas, mas, para os demais campos, devemos utilizar a cláusula `NOT NULL`. No caso de restrições com chaves primárias e integridades referenciais, são adotados os comandos `PRIMARY KEY` e `UNIQUE`, respectivamente, conforme apresentado no exemplo da Figura 15.

```
1 CREATE TABLE EMAIL(
2   id INT PRIMARY KEY,
3   email VARCHAR(60) NOT NULL UNIQUE,
4   contato_fk INT,
5   FOREIGN KEY (contato_fk) REFERENCES contato (id)
6 );
7 CREATE TABLE TELEFONE (
8   id INT PRIMARY KEY,
9   telefone VARCHAR(20) NOT NULL,
10  contato_fk INT,
11  FOREIGN KEY (contato_fk) REFERENCES contato (id)
12 );
```

**Figura 15.** Instrução em SQL referente à criação das tabelas EMAIL e TELEFONE.

Na Figura 15, a cláusula PRIMARY KEY representa o atributo como a chave primária. Já a cláusula UNIQUE é utilizada para especificar chaves únicas, ou seja, chaves que não são primárias em uma determinada tabela. A restrição definida por integridade referencial está relacionada às chaves estrangeiras e são definidas por meio da cláusula FOREIGN KEY, conforme demonstrado nas tabelas EMAIL e TELEFONE.

## Transformação do modelo entidade-relacional para o modelo relacional

Um MER pode ser implementado por diversos modelos relacionais que contêm as informações especificadas pelo próprio DER. Para isso, faz-se necessária a transformação do MER para o modelo relacional. Assim, a transformação de um MER para um relacional deve seguir dois objetivos básicos, segundo Heuser (2009):

- obter um banco de dados que permita bom desempenho de instruções de consulta e alteração do banco de dados;
- obter um banco de dados que simplifique o desenvolvimento e a manutenção de aplicações.

Nesse sentido, para que se possam alcançar tais objetivos, as regras foram definidas tendo como base os princípios descritos a seguir.

- **Evitar junções:** significa ter os dados necessários a uma consulta em uma única linha.
- **Diminuir o número de chaves:** para a implementação eficiente do controle da unicidade da chave primária ou chave alternativa, o SGBD usa normalmente uma estrutura de índice.
- **Evitar campos opcionais:** são os campos que possibilitam receber valores vazios.

### Implementação inicial de entidades e respectivos atributos

Essa implementação é bastante sugestiva, isto é, cada entidade é traduzida para uma tabela, assim como cada atributo da entidade define uma coluna dessa tabela. Por sua vez, os atributos identificadores da entidade definem as colunas que compõem a chave primária da tabela, conforme ilustrado na Figura 16.



Esquema relacional correspondente:

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

**Figura 16.** Transformação de entidade em tabela.

Fonte: Heuser (2009, p. 141).

Cabe destacar, nessa transformação de entidades para tabelas, que não é aconselhável apenas transcrever os nomes dos atributos para os nomes das colunas. Isso porque os nomes de colunas são normalmente referenciados em programas, e os caracteres acentuados, o espaço e outros caracteres especiais ficam limitados ou mais difíceis de serem referenciados em certas linguagens de programação. No exemplo ilustrado na Figura 16, nota-se que o atributo código é transformado em campo, e a este é atribuído o nome CodigoPess. Essa alternativa é utilizada para diferenciar os atributos que apresentam o mesmo nome, porém, em entidades diferentes.

### **Implementação de relacionamentos e respectivos atributos**

Ao trabalharmos com **tradução de relacionamentos**, o fator importante que precisa ser levado em consideração é a cardinalidade mínima e máxima das entidades que participam do relacionamento. A implementação de relacionamentos pode ser dividida em três formas: tabela própria, adição de colunas e fusão de tabelas de entidades.

Para Heuser (2009), a implementação de relacionamentos em uma **tabela própria** é realizada por meio das colunas correspondentes aos identificadores das entidades relacionadas e correspondentes aos atributos do relacionamento. A chave primária dessa 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 à tabela que implementa a entidade referenciada.

Conforme ilustrado na Figura 17 e exemplificado em Heuser (2009, p. 145), “[...] 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)”. As tabelas ENGENHEIRO e PROJETO se tornam chaves estrangeiras das tabelas relacionadas.



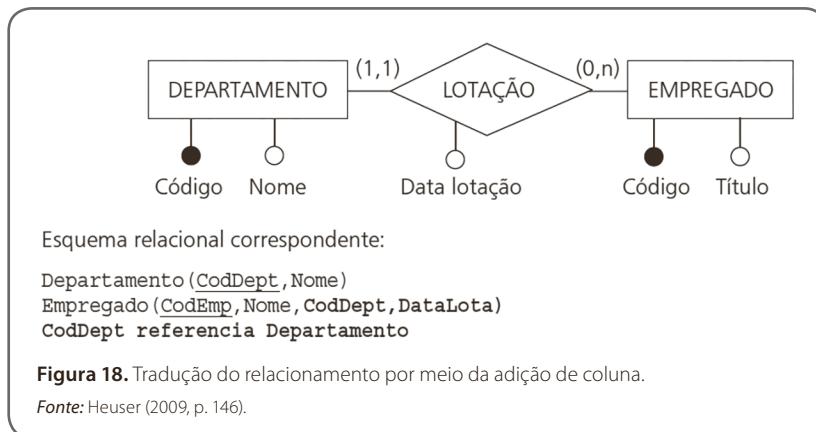
Esquema relacional correspondente:

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

**Figura 17.** Tradução do relacionamento por meio de tabela própria.

Fonte: Heuser (2009, p. 145).

Uma alternativa para a transformação de relacionamentos é a **adição de colunas** em uma das tabelas correspondentes às entidades participantes do relacionamento, conforme ilustra a Figura 18.



Observa-se, na Figura 18, que a tradução consiste em inserir na tabela correspondente à entidade com cardinalidade máxima uma das seguintes colunas:

- coluna correspondente ao identificador da entidade relacionada — essa coluna forma uma chave estrangeira em relação à tabela que implementa a entidade relacionada (coluna CodDept);
- coluna correspondente aos atributos do relacionamento — coluna DataLota.

Já a **fusão de tabelas** de entidades implementadas em um relacionamento é dada conforme a Figura 19.



Esquema relacional correspondente:

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

**Figura 19.** Tradução do relacionamento por meio da fusão de tabelas.

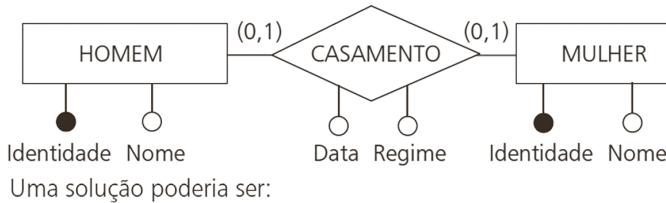
Fonte: Heuser (2009, p. 147).

Conforme descrito anteriormente, na tradução dos relacionamentos, o que deve ser levado em consideração são as cardinalidades mínimas e máximas, pois é por meio delas que os esquemas relacionais serão implementados. Para isso, existem algumas regras que devem ser consideradas em relação aos tipos de relacionamentos 1:1, 1:N e N:N.

Em relação aos relacionamentos 1:1, é necessário ter atenção às seguintes regras:

- ambas as entidades têm participação opcional;
- uma entidade tem participação opcional e a outra tem participação obrigatória;
- ambas as entidades têm participação obrigatória.

Para os relacionamentos 1:1, a primeira regra de implementação é dada pelas entidades que apresentam participação opcional no relacionamento — ou seja, a participação de ambas as entidades pode ser opcional, conforme ilustrado na Figura 20. Para esse tipo de relacionamento, a alternativa preferida é a adição de colunas.



Uma solução poderia ser:

**Mulher** (IdentM, Nome, IdentH, Data, Regime)

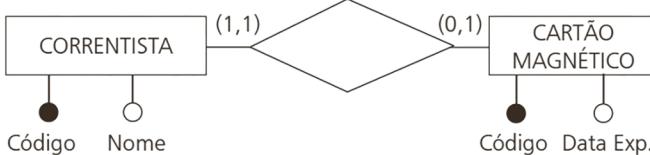
**IdentH** referencia Homem

**Homem** (IdentH, Nome)

**Figura 20.** Implementação de relacionamento 1:1 com participação opcional de ambas as entidades.

**Fonte:** Heuser (2009, p. 149).

Já para o outro tipo de relacionamento, conforme mostra a Figura 21, em que uma entidade tem participação opcional e a outra tem participação obrigatória, nota-se que a regra de implementação preferida é a fusão de tabelas.



Esquema relacional correspondente:

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

**Figura 21.** Implementação de relacionamento 1:1 com participação obrigatória de uma entidade e participação opcional da outra.

**Fonte:** Heuser (2009, p. 151).

No caso de as entidades terem participação obrigatória, a tradução preferida é a fusão das tabelas, conforme demonstrado na Figura 22.



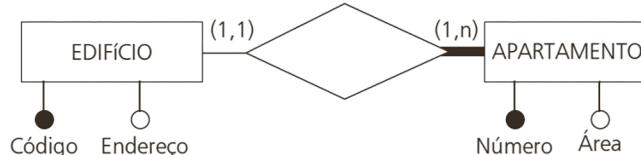
Esquema relacional correspondente:

Conferência(CodConf, Nome, DataInstComOrg, EnderComOrg)

**Figura 22.** Implementação de relacionamento 1:1 com participação obrigatória de ambas as entidades.

Fonte: Heuser (2009, p. 152).

No caso do relacionamento 1:N, a alternativa preferida de implementação é a adição de colunas. Um exemplo que ilustra a tradução dessa implementação é demonstrado na Figura 23, em que a coluna CódigoEd da tabela APARTAMENTO, além de ser chave estrangeira, é também parte da chave primária.



Esquema relacional correspondente:

Edifício(CódigoEd, Endereço)

Apartamento(CódigoEd, NúmeroAp, ÁreaAp)

CódigoEd referencia Edifício

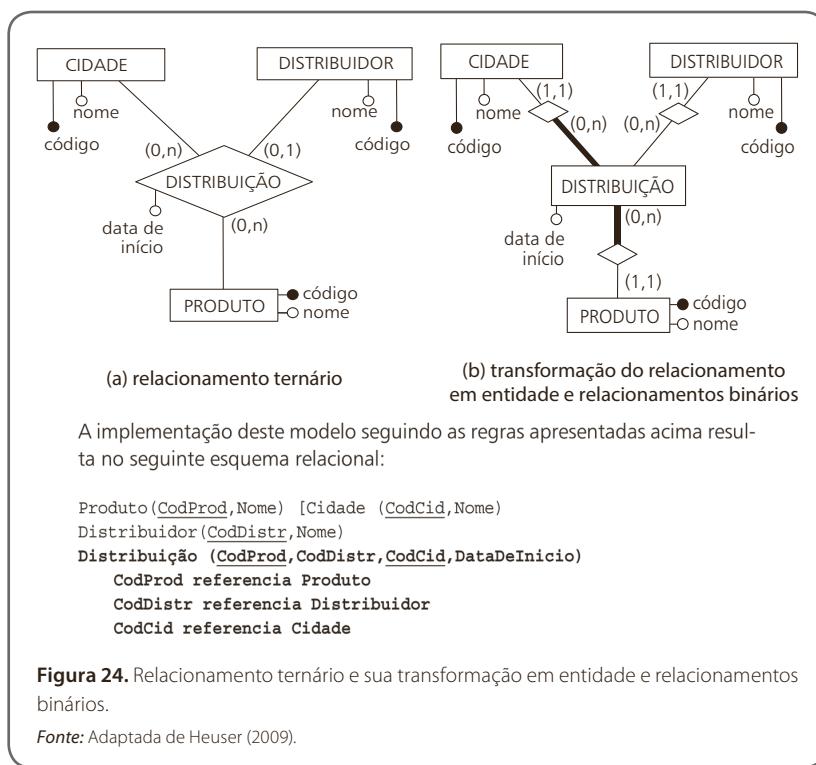
**Figura 23.** Tradução de relacionamentos 1:N pela adição de colunas.

Fonte: Heuser (2009, p. 152).

No caso dos relacionamentos N:N, independentemente da cardinalidade mínima, estes são sempre implementados por meio de tabela própria, conforme mostra a Figura 24. Cabe ressaltar que, muitas vezes, um MER apresenta relacionamentos de entidades com grau maior do que dois — isto é, quando se tem três entidades e um único relacionamento. Nesses casos, a tradução de implementação ocorre por meio dos passos descritos a seguir.

1. O relacionamento é transformado em uma entidade. Essa nova entidade é ligada por meio de um relacionamento binário a cada uma das entidades que participavam do relacionamento original.
  2. As regras de implementação de entidades e relacionamentos binários, apresentadas anteriormente, são aplicadas às entidades e aos relacionamentos binários criados.

A Figura 24 ilustra um exemplo de um MER com relacionamento ternário e como ele é transformado em um esquema relacional correspondente.





## Referências

CALIARI, F. M. *Método para construção de ontologias a partir de diagramas entidade-relacionamento*. 2007. Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial) — Universidade Tecnológica Federal do Paraná, Curitiba, 2007. Disponível em: <https://bit.ly/2l9G0Yb>. Acesso em: 19 mar. 2020.

COUGO, P. S. *Modelagem conceitual e projeto de banco de dados*. Rio de Janeiro: Elsevier, 1997.

GIMENES, I. M. S.; HUZITA, E. H. M. *Desenvolvimento baseado em componentes: conceitos e técnicas*. São Paulo: Ciência Moderna, 2005.

HEUSER, C. A. *Projeto de banco de dados*. Porto Alegre: Bookman, 2009. (Série Livros Didáticos Informática UFRGS).

MACHADO, F. N. R. *Banco de dados: projeto e implementação*. São Paulo: Érica, 2014.

OLIVEIRA, C. H. P. *SQL: curso prático*. São Paulo: Novatec, 2002.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 7. ed. Porto Alegre: AMGH, 2011.



## Fique atento

Os *links* para *sites da web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

