

EAD FURB

BANCO DE DADOS

ALEXANDER ROBERTO VALDAMERI

LIVRO DA DISCIPLINA



Banco de Dados



Reitor

profa. Ma. Marcia Cristina Sardá Espíndola

Vice-Reitor

prof. Dr. João Luiz Gurgel Calvet da Silveira

**Pró-Reitora de Ensino de Graduação,
Ensino Médio e Profissionalizante**

Profª Drª Romeu Hausmann

Pró-Reitor de Administração

Prof. Jamis Antônio Piazza

**Pró-Reitoria de Pesquisa, Pós-Graduação,
Extensão e Cultura**

Prof. Dr. Oklinger Mantovaneli Junior

Professor Autor

Alexander Roberto Valdameri

Ambiente Virtual de Aprendizagem

Guilherme Legal de Oliveira

Design Instrucional

Clarissa Josgrilberg Pereira

Marcia Luci da Costa

Revisão Textual

Odair José Albino

Monitoria

Cibele Bohn

Produção de Mídia

Gerson Souza

Diagramação

Bárbara Marciniak

Design Gráfico

Amanda Ventura de Oliveira

Bianca Klegin Borges


João Pedro Roncalio

Vinícius Bretzke Amaral

Vitor Hugo Miranda

Ícones

No decorrer dos estudos desta disciplina, você irá interagir com diferentes caixas didáticas. Nelas você encontrará atividades, conteúdos extras e ações que dialogam com os conhecimentos específicos de cada ciclo. A identificação destas é realizada pelos seguintes ícones:

	Dica É um espaço com informações que podem auxiliar a compreensão e/ou aplicação do conteúdo abordado.
	Leitura Complementar É um espaço destinado à indicações de leituras complementares.
	QRCode É um recurso para você acessar materiais e conteúdos extras presentes nas caixas didáticas (utilizando o leitor de QR Code no celular ou clicando no código).
	Pratique É um espaço com sugestões para o desenvolvimento de alguma atividade ou experimento prático.
	Refleta É um espaço para estimular a reflexão, elaborar ou desenvolver alguma ideia, pensamento e/ou posicionamento perante determinado assunto.
	Glossário Explicação de um termo específico ou de uma palavra pouco conhecida.
	Saiba Mais É um espaço com informações extras que podem auxiliar a compreensão e/ou aplicação do conteúdo abordado.

Apresentação do Autor

Olá! Sou o professor Alexander Roberto Valdameri. Tenho graduação (1997) em Ciência da Computação pela Universidade Regional de Blumenau (FURB), mestrado (2001) pela Universidade Federal de Santa Catarina (UFSC) e especialização em Educação a Distância: Elaboração de Material e Tutoria (2013), pela Universidade Cruzeiro do Sul. Desde 2001 sou professor do quadro do Departamento de Sistemas e Computação (DSC) da FURB na área de Banco de Dados. Atualmente leciono disciplinas nesta área para os cursos de Ciência da Computação e Sistemas de Informação. Em outros tempos também lecionei disciplinas em outras áreas, como informática aplicada e programação.

Estive na coordenação do Colegiado do Curso de Ciência da Computação (2006-2009; 2012-2014), além de coordenador o curso de Pós-Graduação em Tecnologias para o Desenvolvimento de Aplicações Web (2005-2012).

Minha trajetória profissional na área de Tecnologia da Informação (TI) remete ao início do curso (Ciência da Computação) na FURB, quando atuei como estagiário (1993) na própria Instituição. Em seguida, atuei como monitor (1994-1995) do curso de Ciência da Computação e em paralelo atuei como programador de sistemas (1995-1996) e, após concluir o curso, como analista de sistemas (1997-2001), todos na FURB. Foi durante a graduação que me identifiquei fortemente com a área de banco de dados, razão que me fez buscar maior qualificação. No final da década de 90 obtive a certificação Oracle Certified Professional (OCP) para atuação como Database Administrator Oracle (DBA Oracle). Durante alguns anos (2000-2004) atuei como profissional autônomo para empresas da região do Vale do Itajaí, prestando serviços terceirizados na área de banco de dados.

Antes de me tornar professor na FURB tive a oportunidade de trabalhar em outras instituições de ensino superior, como a Asselvi (1999-2005), o SENAI (2008-2010) e o IEL (2009). Quer saber mais sobre minha formação e atuação? Acesse meu currículo lattes:



Sumário

Apresentação da Autora	5
Apresentação da Disciplina	7
2. Modelagem, normalização e projeto de banco de dados relacional.....	8
2.1 Modelagem de dados	9
2.1.1 A importância da modelagem de dados	9
2.1.2 Abstração na modelagem de dados	10
2.1.3 As etapas da modelagem de dados	11
2.2 O modelo entidade-relacionamento (MER).....	14
2.3 Normalização de dados	25
2.4 Projeto físico de dados	32
2.4.1 Derivação a partir do MER	32
2.4.2 Derivação a partir de um projeto orientado a objetos.....	35
Conclusão	38
Referências.....	39

Apresentação da Disciplina

Você encontrará neste livro, de forma concisa e objetiva, os principais conceitos e práticas envolvidos na temática Banco de Dados. Os conhecimentos deste guia contribuem para a formação de diferentes áreas da TIC (tecnologia da informação e comunicação). Este material se destina principalmente aos estudantes de graduação em informática, mas também se destina a outras áreas do conhecimento que buscam conhecer mais sobre armazenamento e tratamento de dados, como cursos da área de gestão (informação ou conhecimento), engenharias ou mesmo estatística. Como você verá ao longo da disciplina, os principais conceitos da tecnologia de banco de dados e suas aplicações foram divididos em três ciclos:

Ciclo 1

No primeiro Ciclo você terá a oportunidade de aprender o que é um banco de dados, a diferença entre dado e informação, por que os sistemas computacionais utilizam banco de dados, e um breve relato histórico sobre a evolução, as abordagens de bancos de dados existentes e as arquiteturas onde encontramos os bancos de dados.

Ciclo 2

No segundo Ciclo você será apresentado aos conceitos e técnicas de modelagem de dados relacional. Esta etapa será marcada por intensas atividades reflexivas acerca de cenários reais onde nosso objetivo será desenvolver aptidões para a leitura das demandas computacionais no tocante ao armazenamento de dados, de tal forma que possamos representar graficamente modelos de dados ideais para cada contexto estudado.

Ciclo 3

O terceiro Ciclo irá tratar os aspectos relacionadas à linguagem de manipulação dos bancos de dados relacionais, a SQL. Nesta etapa focaremos nos conceitos básicos da linguagem, comandos de criação e gerenciamento de estruturas, bem como comandos para manipulação e consulta dos dados.

Bons estudos!

2. Modelagem, normalização e projeto de banco de dados relacional

É chegada a etapa em que você será apresentado aos conceitos e técnicas de modelagem de dados relacional. Sim, relacional, que é o nosso foco exclusivo nesta disciplina. Esta etapa será marcada por intensas atividades reflexivas acerca de cenários reais onde nosso objetivo será desenvolver aptidões para a leitura das demandas computacionais no tocante ao armazenamento de dados, de tal forma que possamos representar graficamente modelos de dados ideais para cada contexto estudado. Além dos conteúdos, você será desafiado a participar de vários estudos de caso, onde juntos iremos discutir as melhores formas de representar a persistência (armazenamento) dos dados nos cenários que serão apresentados.

2.1 Modelagem de dados

Na primeira etapa da nossa disciplina você teve a oportunidade de conhecer conceitos, características e funcionalidades dos bancos de dados. Também tivemos a oportunidade de discutir brevemente sobre características de alguns dos principais SGBDRs disponíveis no mercado atualmente. Agora vamos conhecer as técnicas para modelagem de um projeto de banco de dados. Antes, vamos refletir:

A resposta não é simples, pois está revestida de vários argumentos, mas será dada gradualmente, à medida que avançarmos nos conteúdos. Por hora, adiantamos que esta etapa é fundamental, pois sem ela corremos um sério risco de não contemplar adequadamente as estruturas para a persistência dos dados, visto que precisamos conhecer, analisar, discutir e somente a partir deste momento representar graficamente como os dados serão armazenados.



Você deve estar se perguntando: por que não avançamos diretamente para a Linguagem SQL e criamos as estruturas (tabelas) necessárias para armazenar os dados da nossa aplicação?

2.1.1 A importância da modelagem de dados

Para um melhor entendimento da importância da modelagem de (ou dos) dados no processo de desenvolvimento de um sistema computacional, convidamos você a uma reflexão. Você já fez uma viagem, seja sozinho ou mesmo em família? A resposta provavelmente é sim. Pois bem, e para esta viagem houve a preocupação de planejá-la de forma a identificar e escolher o que deve ser levado na bagagem ou mesmo qual o roteiro a ser seguido? Da mesma forma, quando um profissional se propõe a desenvolver um sistema computacional, é preciso planejar todas as suas etapas e dedicar atenção especial ao projeto e estruturação do banco de dados. Isso vale para sistemas de pequeno e grande porte.

Para isso é utilizada uma técnica conhecida como modelagem de dados, cujo objetivo é transformar uma ideia conceitual em algo que possa ser traduzido em termos computacionais. Com a modelagem de dados é possível refinar um modelo conceitual durante as fases que compõem o projeto, eliminando redundâncias ou incoerências que possam inevitavelmente surgir (ALVES, 2014).

Há vários fatores que contribuem para o sucesso da modelagem de dados de um sistema computacional. Gostaria de destacar dois pontos: (i) o profissional de TI, que geralmente é o analista de sistemas, deve conhecer o cenário que será modelado/representado; e (ii) o usuário, por meio da sua participação em determinadas etapas, como levantamento de dados, testes de usabilidade e a validação. Veremos adiante mais detalhes sobre algumas destas etapas.

De acordo com Machado (2014), a modelagem de dados é o estudo das



Certo! Mas afinal o que é modelagem de dados? E como faremos a representação dos dados?

informações existentes em um contexto sob observação para a construção de um modelo de representação e entendimento de tal contexto. Assim, devemos considerar que a realidade, de forma geral, muda de uma empresa para outra e, portanto, precisamos estabelecer um formato padrão de representação para estruturar um banco de dados independentemente do tipo de negócio ou mesmo contexto em que ele está inserido.

Os detalhes dos elementos que compõem essa representação gráfica veremos mais adiante. Antes é importante que você compreenda que a modelagem de dados inicia pelo processo de observação dos objetos relevantes que existem em uma realidade ou contexto. A partir dessa observação iremos construir um modelo de compreensão e conceitos existentes nessa realidade. Chamamos de minimundo este primeiro modelo, sem nos atermos no momento à automatização ou processamento da informação desta realidade. A partir daí é que partimos para a análise de dados.

Esta etapa inicial exige do observador (analista/projetista) uma certa capacidade de abstração. É isso que discutiremos um pouco mais a seguir: a abstração no processo de modelagem de dados.

2.1.2 Abstração na modelagem de dados

Para Machado (2014, p. 28),

Abstração ou a capacidade de abstração é um processo, digamos, mental, que usamos quando selecionamos várias características e propriedades de um conjunto de objetos ou fatos, e excluimos outras características que não são relevantes. Em outras palavras, aplicamos abstração quando nos concentramos nas propriedades de um conjunto de objetos ou coisas que consideramos essenciais, e desprezamos outras que não consideramos importantes.

O autor exemplifica o conceito de abstração a partir da ilustração de uma bicicleta, conforme podemos ver na Figura 26. Para Machado (2014, p. 16), ao observamos a figura, “temos o resultado de um processo de abstração em que excluimos detalhes da estrutura de uma bicicleta, como os pedais, os freios, os mecanismos de tração e inclusive as possíveis diferenças entre várias bicicletas”. A partir disso, definimos o objeto como o que conceituamos e denominamos “bicicleta”.

No caso da figura, o conceito “bicicleta” é uma representação dessa prática de abstração. A partir da definição e exemplo do autor podemos assumir que a abstração é a maneira como enxergamos o minimundo sem considerar aspectos técnicos.



Figura 26: Case de abstração. Fonte: Machado (2014).

Procure ficar tranquilo quanto ao conceito de abstração durante o período de observação e captura de informações do cenário (minimundo). É comum nesse momento confundir abstração com superficialidade. Não podemos ser superficiais no olhar.

A seguir vamos apresentar as técnicas utilizadas para a modelagem de dados. Mas antes reforçamos uma vez mais a importância da racionalidade na identificação das necessidades de informação da realidade e sempre dentro do contexto desta realidade, de tal forma que ela contribua para que os indivíduos que atuam no contexto (minimundo) possam alcançar os objetivos em seu ambiente de negócios.



Devemos ser seletivos quanto aos aspectos que estamos observando para que nosso conjunto de requisitos represente minimamente as necessidades do usuário.

2.1.3 As etapas da modelagem de dados

Conhecer o problema é uma etapa indispensável para chegar à sua solução. Para que possamos compreender um problema, seja ele simples ou complexo, é necessário que o analisemos a partir de diferentes pontos de vista. A análise estruturada, assim como a orientada a objetos, focaliza um sistema computacional fundamentalmente a partir das funções que o compõem. Já a análise de dados foca nos dados a serem persistidos pelo sistema. Assim, fundamentalmente, ela trabalha com a visão dos dados. Observe na Figura 27 as visões “dados” e “funções” que compõem um sistema computacional.

No início, certamente não. Cada problema é único e, por mais que haja características comuns entre muitos casos, haverá sempre aspectos específicos a serem considerados. Neste caso, lembre-se do que vimos anteriormente sobre abstração!

AS VISÕES SE COMPLEMENTAM EM UM SISTEMA DE INFORMAÇÃO

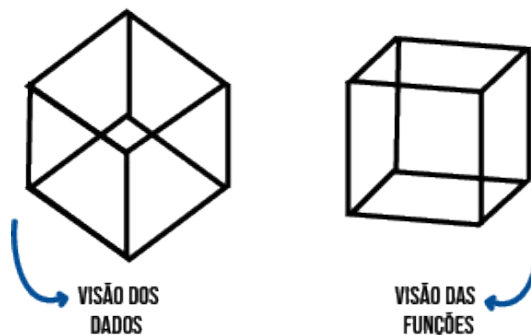


Figura 27: As visões de um sistema computacional.
Fonte: O autor, adaptado por DME/FURB (2019).



É fácil fazer a análise de dados?

Mas, afinal, por onde se deve começar? O primeiro passo é conhecer o problema. Não desanime se você não conseguir identificar e/ou diagnosticar “todos” os possíveis requisitos (funcionalidades do sistema) aparentes. Considere isso como sendo normal.

A análise de dados é um processo que exige refinamentos e, portanto, é contínuo até que se obtenha como resultado o modelo de dados mais próximo do que julgamos ideal. Há muita subjetividade no processo. É necessário ter paciência e persistência! Novamente, lembre-se da abstração, ou seja, atenha-se inicialmente a identificar aspectos menos técnicos.

A análise de dados fornece uma visão alternativa para o sistema. Um sistema é como se fosse um objeto de múltiplas dimensões. Dependendo do ponto de vista, alguns detalhes são destacados e outros ficam escondidos. As técnicas de especificação

de sistemas devem focalizar o problema sob diferentes ângulos.

Isoladamente, nenhuma delas fornece uma visão completa do sistema. Por isso, um bom analista/projetista deve conhecer as técnicas para a visualização e apuração dos dados e também das funções. A partir disso, você deve utilizá-las em conjunto para um melhor conhecimento do problema. O resultado da coleta da análise de dados será o conjunto de requisitos do sistema. A Figura 28 ilustra diferentes visões sobre elementos do problema que contribuem para um melhor entendimento seu.

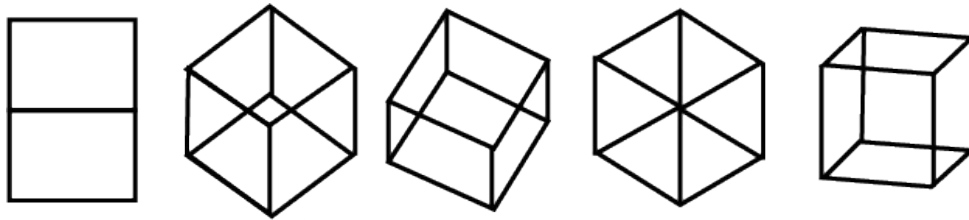


Figura 28: As diferentes visões e um melhor conhecimento do problema.
Fonte: O autor, adaptado por DME/FURB (2019).

A partir desse conjunto de requisitos funcionais, inicia-se a construção do modelo conceitual. Cabe destacar que nessa etapa há a independência do SGBD que iremos utilizar para implementar nosso banco de dados. O que isso significa? Na prática isso quer dizer que podemos construir um modelo de dados que poderá ser persistido a partir de um padrão de arquivos estruturados – como exemplos, XML ou JSON – ou mesmo adotarmos uma alternativa de serialização a partir de uma linguagem de programação. Mas calma, não vamos perder o foco: modelo de banco de dados relacional, ou seja, SGBD relacional!

A seguir temos a etapa onde, a partir das necessidades e características identificadas no cenário, definimos qual será o SGBD relacional que iremos utilizar para persistir nossos dados. Este momento, considerado abstrato, denominamos de modelo lógico.

Por fim, uma vez escolhido o SGBD relacional, partimos para a criação das estruturas de dados. Nessa etapa precisamos considerar todas as particularidades do SGBD escolhido, de forma a utilizar ao máximo as suas funcionalidades. Esta etapa é conhecida como modelo físico ou projeto físico do banco de dados.

De forma sintetizada, as etapas descritas do processo de modelagem de dados podem ser visualizadas na Figura 29.



Se você ficou interessado em conhecer mais sobre os padrões XML ou JSON, recomendo uma pesquisa em nossa Biblioteca. Se preferir, acesse o site da W3C



Mas por que SGBD relacional? Atenção! O foco da nossa disciplina é persistência em estruturas relacionais, lembra? Mesmo assim fique tranquilo, pois o esforço na identificação da estrutura de dados para persistência é válido, mesmo que em dado momento venhamos a nos convencer de que outra abordagem de armazenamento atenda nossa necessidade da mesma forma ou melhor. Vamos em frente!

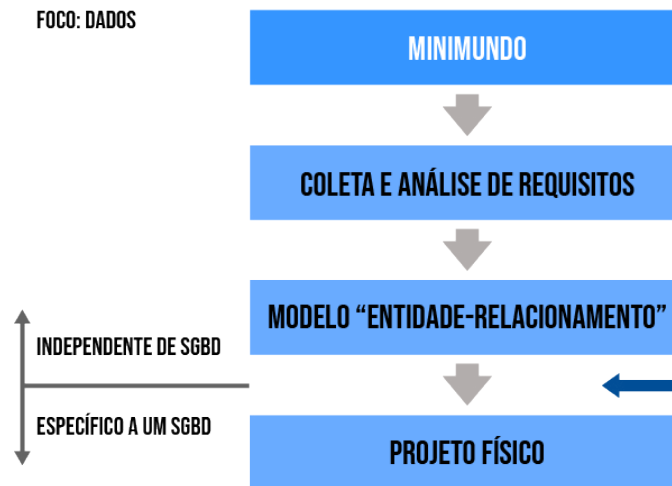


Figura 29: Etapas do processo de modelagem de dados.
Fonte: O autor, adaptado por DME/FURB (2019).



Neste momento todos os nossos olhares estão focados nas estruturas de armazenamento dos dados. Em outras disciplinas, como Engenharia de Software, esse olhar é mais amplo, com foco nas operações sobre os dados, por exemplo, sendo a persistência dos dados apenas uma das várias etapas do desenvolvimento de um *software*.

Muito bem! Já conhecemos as etapas que compõem o processo de modelagem de dados. Mas e agora, como faremos a representação gráfica do modelo conceitual? É chegado o momento de você conhecer as técnicas envolvidas no processo de representação gráfica desse modelo.

Basicamente, em se tratando da modelagem de dados para bancos de dados relacionais (ou SGBDs relacionais), vamos considerar a técnica de modelagem entidade-relacionamento (MER) e a técnica de normalização de dados.

2.2 O modelo entidade-relacionamento (MER)

Ao longo da história, foram feitas pesquisas e, dentre as muitas propostas, destacam-se métodos que nos auxiliam na criação de uma estrutura concisa com maior correção e facilidade para lidar com os dados. Para a representação formal dos dados, dentre os modelos propostos, tem destaque o modelo entidade-relacionamento, que é um padrão de modelagem conceitual e ponto de referência para as propostas de modelagem de objetos hoje, como a Unified Modeling Language (UML).

O modelo entidade-relacionamento (MER), também conhecido como diagrama entidade-relacionamento (DER), foi criado por Peter Chen em 1976 (CARDOSO, 2008). Peter Chen, por sinal, é uma das formas de representação gráfica bastante comum nas literaturas para ilustrar uma modelagem conceitual.

Atualmente, utiliza-se esse modelo, ou notação gráfica (Peter Chen), como o primeiro passo para a representação da estrutura de um banco de dados. Projetos que excluem esse processo apresentam muitos erros e falhas, que, ao longo de seu desenvolvimento e aplicação, causam inúmeros problemas. É necessário estar atento e não deixar de utilizar esse método, garantindo, assim, uma estrutura sólida, segura e com menor possibilidade de erros no projeto, seja de pequeno, médio ou grande porte (CARDOSO, 2008). A partir do MER é possível uma representação da estrutura lógica do projeto a partir de uma visão genérica. Sua estrutura é feita de forma clara e simples, possibilitando representar os dados do mundo real (minimundo) como objetos denominados “entidade” ligados uns aos outros (objetos) por meio de associações denominadas “relacionamento”. Daí a origem do nome “entidade-relacionamento”.

Agora que você conhece o seu conceito, por que ela deve ser utilizada e o cenário no qual o MER está inserida, é hora de conhecer os elementos básicos que a constituem.

O MER é constituído por:

- **entidades**
- **atributos**
- **relacionamentos**

Antes de abordarmos mais detalhadamente cada um desses elementos básicos, você deve saber como se dá a representação gráfica de cada um deles. Existem muitas notações para a construção do MER. Como já citada anteriormente, a **notação** original foi proposta por **Peter Chen** e é composta de entidades (retângulos), relacionamentos (losangos), atributos (círculos) e linhas de conexão (linhas) que indicam a ligação de uma entidade em um relacionamento, assim como a associação dos atributos com a entidade. Observe um exemplo na Figura 30:

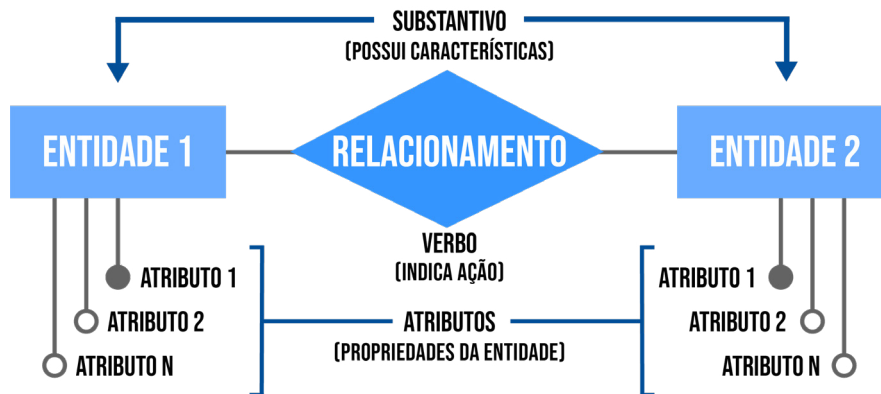


Figura 30: Representação gráfica dos elementos do MER.
 Fonte: O autor, adaptado por DME/FURB (2019).

Vejam os um exemplo: vamos considerar um cenário onde estão envolvidos sócios e títulos de um clube social e esportivo. No exemplo, um sócio é proprietário de um título. Veja como seria a representação gráfica do cenário apresentado na Figura 31.

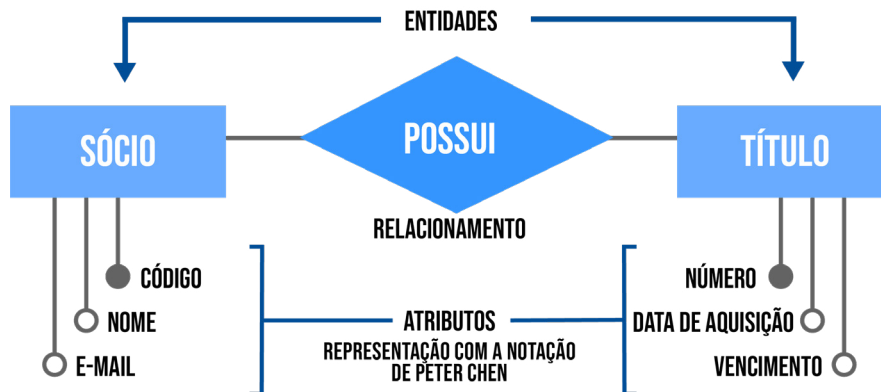


Figura 31: Exemplo de MER.
 Fonte: O autor, adaptado por DME/FURB (2019).

No exemplo apresentado podemos observar a presença dos elementos que compõem o MER. A seguir vamos explorar em detalhes cada um deles.

Os objetos ou “coisas” sobre os quais precisamos guardar dados são chamados de entidades. Quando analisamos um problema devemos ficar atentos aos substantivos e nomes que aparecem. Eles podem vir a ser entidades no nosso modelo.

Não considere isso uma regra. Na verdade, torna-se complicado afirmar a existência de “regras” para a definição das entidades, visto que cada cenário (minimundo) pode apresentar particularidades sensíveis ao contexto. No entanto, um bom começo para avaliar cada substantivo ou nome que julgamos ser uma entidade é fazer as seguintes perguntas:

- Há necessidade de guardar informações sobre cada objeto?
- Há mais de um objeto desse tipo?
- É possível identificar cada objeto unicamente?

Voltemos ao cenário ilustrado na Figura 31 como exemplo. O cenário envolve um sócio de um clube. Agora vamos responder às perguntas sugeridas:

Há necessidade de guardar informações sobre cada objeto?

– Sim!

Há mais de um objeto desse tipo?

– Sim!

É possível identificar cada objeto unicamente?

– Sim, através do número, por exemplo.

Diante das respostas positivas obtidas, podemos afirmar que sócio é de fato uma entidade que irá compor o modelo. Em caso de uma das perguntas apresentar uma resposta negativa, podemos desconsiderá-la como entidade.

O próximo passo é conhecer alguns conceitos envolvidos com as entidades. Os mais comumente utilizados são: **especialização**, **generalização**, **entidade fraca** e **entidade forte**. Para ajudá-lo a entender melhor esses conceitos, vamos mostrar a seguir sua aplicação através de exemplos.

Fazemos uma **especialização** quando existem atributos que só se aplicam a um subconjunto da entidade. A melhor forma de compreender esse conceito é através de um exemplo. Observe a Figura 32.

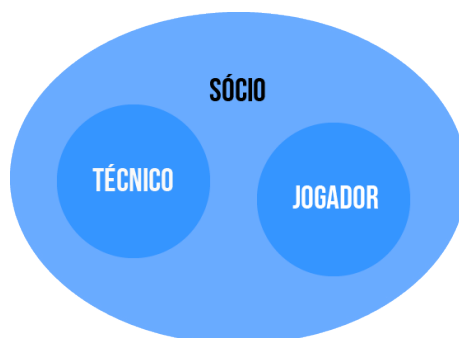


Figura 32: Exemplo de especialização.
Fonte: O autor, adaptado por DME/FURB (2019).

O clube organiza torneios de voleibol misto. Mas para participar, como jogador ou como técnico, é preciso ser sócio. Logo temos: um jogador é um sócio, assim como um técnico também. Portanto, uma entidade é considerada subentidade de outra se a primeira é um conjunto da segunda entidade. Ao criar uma subentidade, fazemos uma especialização.

Há também o processo inverso, no qual, ao examinarmos duas ou mais entidades, descobrimos que vários de seus atributos são comuns. Dizemos então que essas entidades são subconjuntos da mesma entidade. Esse processo é conhecido como **generalização**. Considere o seguinte exemplo ilustrado na Figura 33.

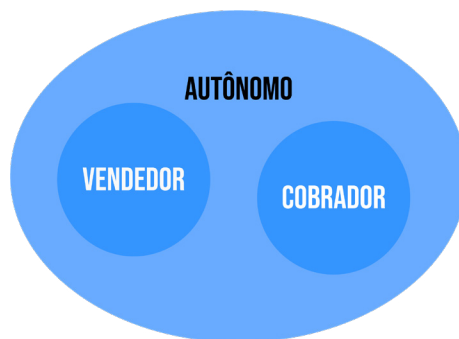


Figura 33: Exemplo de generalização.
Fonte: O autor, adaptado por DME/FURB (2019).

No exemplo temos: “autônomo” é uma entidade generalizada de vendedor e cobrador. Portanto, uma entidade generalizada contempla atributos comuns a duas ou mais entidades – estas, por sua vez, consideradas subentidades.



Observe que até o momento não estamos dando ênfase às propriedades que caracterizam as entidades. A razão é simples: identificar as entidades é um processo considerado mais complexo. Mas não se preocupe, tão logo você saiba mais sobre atributos, isso também fará parte dos seus estudos.

Consideramos uma **entidade fraca** aquela que depende de outra para sua existência. Para exemplificar, voltemos ao contexto do clube do qual você é sócio. Agora, porém, considere que você tem dependentes, os quais também frequentam o clube. Podemos representar isso por meio da Figura 34.

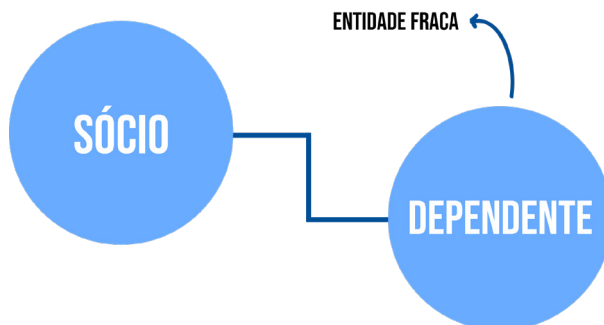


Figura 34: Exemplo de entidade fraca. Fonte: O autor, adaptado por DME/FURB (2019).

A identificação do dependente (propriedades) está associada à identificação do sócio, obrigatoriamente. Portanto, dependente é uma entidade fraca, pois depende da existência do sócio, uma vez que não existe dependente sem um sócio.

Você acabou de conhecer o conceito de entidade fraca. Toda vez que temos uma entidade fraca temos também uma entidade forte. No exemplo utilizado (Figura 35), que envolvia sócio e dependentes, a entidade “sócio” executa o papel de entidade forte, conforme podemos observar:

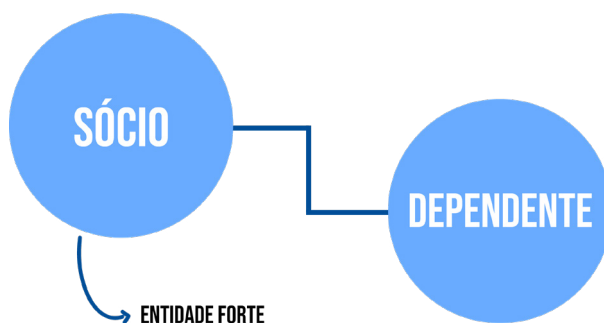


Figura 35: Exemplo de entidade forte. Fonte: O autor, adaptado por DME/FURB (2019).

Sempre que existir uma dependência, uma relação entre as entidades envolvidas deverá ser representada. Estamos falando do mesmo contexto envolvido com a entidade fraca, ou seja, dentre as propriedades do dependente haverá uma identificação do sócio, obrigatoriamente. Portanto, **sócio é uma entidade forte, pois sem a sua existência não há possibilidade da ocorrência de dependentes.**

Agora vamos falar sobre os atributos. Os atributos são as propriedades das entidades. Para Machado (2014), os atributos representam propriedades elementares de uma entidade ou relacionamento. Cada atributo está associado a um domínio particular, que é um conjunto de valores válidos para o atributo. Podemos afirmar

também que os atributos caracterizam as entidades, dando sentido à sua existência.

Na MER a denominação é “atributo”. Em breve, após conhecer o modelo físico de dados e, por consequência, a efetivação das estruturas projetadas em um SGBD, você vai descobrir que “atributo” é sinônimo de “coluna” ou “campo” em uma estrutura de banco de dados. Mas por ora vamos continuar a utilizar o termo **atributo**.

Vejamos alguns exemplos de atributos. Para a entidade “sócio”, entre outros, temos:

- número do sócio
- data de nascimento
- endereço de *e-mail*

Os atributos podem ser classificados de acordo com suas características. Alguns, por sua vez, podem ser classificados em mais de um tipo. Isso pouco vai influenciar o processo de modelagem de dados. Mesmo assim, vamos conhecer os principais tipos de atributos:

- **Atributo simples:** É o mais comum. Não possui qualquer característica especial. Ex.: gênero, data de nascimento, etc.
- **Atributo composto:** Pode ser dividido em partes. Ex.: nome (prenome, nome intermediário, sobrenome), endereço (logradouro, tipo de logradouro, bairro, cidade, estado e CEP), etc.
- **Atributo derivado:** Derivado de operações envolvendo outros atributos. Ex.: subtotal do item NF, total da NF, etc.
- **Atributo determinante:** Valor único para cada atributo instanciado. É sério candidato a chave primária da tabela. Ex.: código de identificação, número do CPF, número do PIS.

Os objetos do mundo real (entidades) não existem sozinhos. Eles se associam uns aos outros. Essa associação é denominada relacionamento.

A resposta é simples: os relacionamentos representados no MER serão responsáveis pelas regras de integridade referencial das estruturas propostas e, conseqüentemente, exercerão um papel importante na garantia da consistência dos dados que serão armazenados.

A importância dos relacionamentos estabelecidos no MER não pode ser avaliada simplesmente a partir da explanação acima. Você vai conhecer na prática a sua importância a partir do momento em que as estruturas conceituais são efetivadas juntamente com as regras (relacionamentos) e os dados passam a ser manipulados.

Antes de iniciarmos a exploração dos tipos de relacionamentos existentes no MER é fundamental que resgatem os dois conceitos já apresentados no primeiro Ciclo



Não há entidade sem atributo. Mesmo entidades com um único atributo são consideradas entidades. Mas atenção: é muito rara a existência de entidade com apenas um atributo.



Mas por que é importante um relacionamento em um modelo de banco de dados?

de nossa disciplina: chave primária e chave estrangeira.

O papel das chaves: a chave primária e a chave estrangeira são fundamentais para o modelo relacional. A partir deste ponto em que vamos discutir projetos de banco de dados relacionais, esses conceitos serão frequentemente utilizados. Por isso, vejamos:

- **Chave primária:** é a combinação de um ou mais atributos que identificam unicamente uma tabela. O valor deve ser único para cada registro (linha).
- **Chave estrangeira:** é a combinação de um ou mais atributos que referenciam outra tabela. Possui o valor da chave primária da tabela referenciada.



Com esses conceitos é possível afirmar que a chave primária e a chave estrangeira são os elementos que estabelecem o relacionamento entre as entidades. Lembre-se disso toda vez que falarmos em chave, seja ela primária, seja estrangeira.

Muito bem, resgatamos o conceito de chaves e possivelmente você ainda não está convencido do que precisamos projetar as estruturas de dados... Pois bem, o dado é parte fundamental em um sistema computacional. Uma estrutura de dados planejada e aperfeiçoada permite ao analista/projetista desenhar os processos, a interface do usuário, os relatórios e demais recursos relacionados ao desenvolvimento da aplicação sempre que necessário. Portanto, projetar as estruturas de armazenamento dos dados é um passo fundamental no desenvolvimento de sistemas computacionais, além de favorecer o entendimento do problema a ser solucionado.

É chegado o momento de tratarmos dos tipos de relacionamentos e as cardinalidades (ou conectividades). Os relacionamentos apresentam características e tipologias. Basicamente existem três tipos de relacionamentos. O primeiro, e mais comum, é conhecido como binário, pois envolve exclusivamente duas entidades. Veja a representação gráfica na Figura 36.



Figura 36: Exemplo de relacionamento binário. Fonte: O autor, adaptado por DME/FURB (2019).

O segundo é conhecido como recursivo binário ou autorrelacionamento, pois envolve uma entidade, sendo que esta exerce o papel de duas. Veja a representação gráfica da Figura 37.

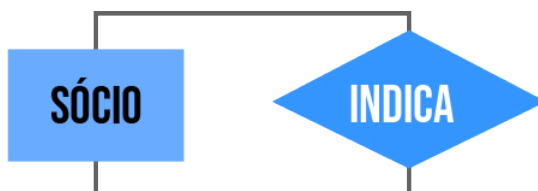


Figura 37: Exemplo de relacionamento recursivo binário. Fonte: O autor, adaptado por DME/FURB (2019).

O terceiro grupo de relacionamento é caracterizado pela ocorrência de três entidades, portanto é conhecido como ternário. Não é muito comum sua ocorrência

e, portanto, para exemplificar vamos considerar o cenário no qual uma empresa tem funcionários que atuam em diferentes projetos com habilidades específicas a cada projeto. Algo assim: João trabalha no projeto A como programador de sistemas, e no projeto B como analista de qualidade.

Agora veja a representação da Figura 38.

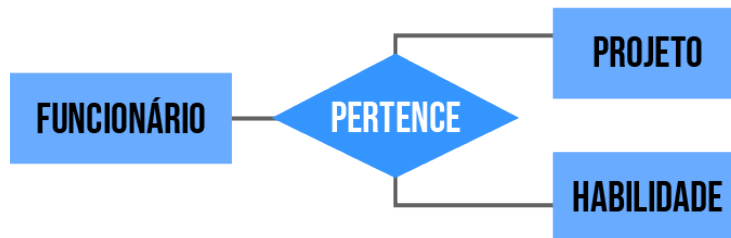


Figura 38: Exemplo de relacionamento ternário. Fonte: O autor, adaptado por DME/FURB (2019).

Vamos agora conhecer os três tipos de conectividades, ou cardinalidades, termo utilizado em algumas literaturas. Os tipos de conectividades representam a forma como uma entidade se associa a outra. Observe na Figura 39 como se aplica cada um dos tipos.

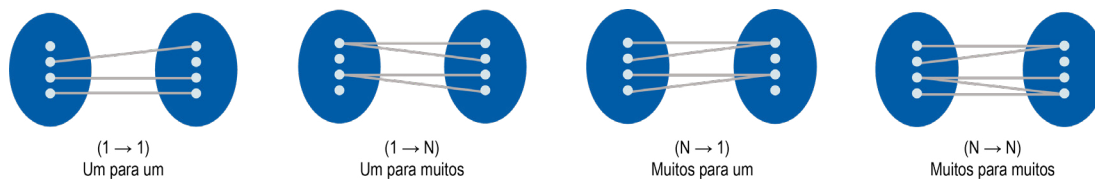


Figura 39: Tipos de conectividades. Fonte: O autor, adaptado por DME/FURB (2019).

Agora que conhecemos os tipos de conectividades, vamos entender como aplicá-los. Mas antes mesmo de convidar você a conhecer os tipos de conectividades, vamos entender como é o processo investigativo para descobrirmos qual tipo que se aplica a cada associação. Podemos utilizar a seguinte expressão:

Um(a) [ENTIDADE A] [RELACIONAMENTO] [preposição] **quantos(as)** [ENTIDADE B]?

Onde:

ENTIDADE A → coisa/objeto que faz o papel do sujeito na oração;

RELACIONAMENTO → verbo que indica a ação;

Preposição → estabelece a conexão (ex.: de, a, com, por, para, até, em);

ENTIDADE B → coisa/objeto que sofre a ação (relacionamento) provocada pelo sujeito.

É necessário que você faça a substituição dos elementos entre colchetes e em seguida realize o questionamento. É possível que pequenas adequações sejam



O relacionamento ternário apresenta uma particularidade. Tão logo você conheça os tipos de conectividades (ou cardinalidades) dos relacionamentos, será mais fácil seu entendimento e aplicação. Aguarde, estamos chegando lá!



O tipo de conectividade “um para muitos” e “muitos para um” são equivalentes. Isso porque a diferença entre eles limita-se à posição dos conjuntos de dados (entidades) na representação gráfica. Trataremos ambos como sendo “um para muitos”.



A resposta ao questionamento deverá sempre ser adicionada ao lado da entidade que sofreu a ação (pergunta)!

necessárias em razão dos verbos. Faça sem problemas. Fique certo de que isto não irá interferir no resultado de sua análise.

A seguir vamos exemplificar o uso da expressão acima. Antes, porém, é importante destacar que inicialmente iremos utilizar a notação gráfica do Peter Chen para representarmos o MER. Oportunamente falaremos mais sobre outros tipos de notações gráficas.

Considere parte de um cenário em que temos funcionário e cônjuge. A Figura 40 ilustra como seria a representação gráfica para um exemplo de conectividade “um para um”.



Figura 40: Exemplo de conectividade “um para um”. Fonte: O autor, adaptado por DME/FURB (2019).

Vejamos a aplicação da expressão que você aprendeu anteriormente:

Um(a) [ENTIDADE A] [RELACIONAMENTO] [preposição] **quantos(as)** [ENTIDADE B]?

Substituindo os termos:

Um FUNCIONÁRIO é CASADO com quantos CÔNJUGES?

Inverte-se a pergunta:

Um CÔNJUGE é CASADO com quantos FUNCIONÁRIOS?

A resposta de cada uma das questões feitas deve ser atribuída no lado da entidade que sofre a ação, ou seja, que foi questionada. Assim, temos um exemplo de conectividade “um para um”.

Este tipo de conectividade (“um para um”) é o menos comum dentre os três que você irá conhecer. Em geral é utilizado quando uma ocorrência de uma entidade equivalente a um registro em uma tabela se relaciona uma única vez com uma ocorrência de outra entidade. O exemplo está indicando que um funcionário estará relacionado com apenas um cônjuge, assim como um cônjuge também estará relacionado a apenas um funcionário. Nesse tipo de conectividade a ligação (chave) poderá ser adicionada a qualquer um dos lados, porém é fundamental o bom senso. No exemplo acima é recomendado adicionar nas propriedades da entidade cônjuge.

Passamos agora à conectividade “um para muitos”. O tipo de conectividade “um para muitos” é o mais utilizado entre entidades de um MER. Em geral, é utilizado quando um registro de uma entidade se relaciona mais de uma vez com registros de outras entidades. A Figura 41 apresenta parte de um cenário onde temos departamentos e funcionários.



Você está lembrado do tipo de relacionamento recursivo binário? Pois bem, este é um cenário em que poderíamos avaliar a possibilidade de o utilizarmos. Para isso teríamos que considerar que funcionário e cônjuge poderiam estar caracterizados como pessoa e, neste caso, seria possível a utilização deste tipo de relacionamento. Fica a dica. Iremos discutir isso nas atividades práticas.



A sugestão de incluir a chave de ligação nas propriedades do cônjuge se justifica pelo fato de que não necessariamente todo funcionário terá um cônjuge. Porém, todo cônjuge, uma vez que é cônjuge, terá um funcionário associado.



Figura 41: Exemplo de conectividade “um para muitos”. Fonte: O autor, adaptado por DME/FURB (2019).

Vejamos a aplicação da expressão que você aprendeu anteriormente:

Um(a) [ENTIDADE A] [RELACIONAMENTO] [preposição] **quantos(as)** [ENTIDADE B]?

Substituindo os termos:

Um DEPARTAMENTO lota quantos FUNCIONÁRIOS?

Inverte-se a pergunta:

Um FUNCIONÁRIO é lotado em quantos DEPARTAMENTOS?

Da mesma forma, a resposta de cada uma das questões feitas deve ser atribuída no lado da entidade que sofre a ação, ou seja, que foi questionada. Ao substituirmos os termos, temos: um departamento lota quantos funcionários? A resposta é muitos, portanto devemos colocar o “n” no lado do funcionário. Um funcionário é lotado em quantos departamentos? A resposta é “1” (um), portanto devemos colocar o “1” no lado do departamento. Veja que nessa pergunta é desprezado o fato de conter um “n” que indica muitos ao lado do funcionário, ou seja a pergunta investigativa independe da resposta anterior. Assim, temos um exemplo de conectividade “um para muitos”.

É importante destacar que houve a necessidade de adequar o uso do verbo para que a pergunta tivesse mais sentido. Isso é comum em razão do uso da linguagem natural no exercício da atividade.

Em se tratando da ligação (chave), temos que neste tipo de conectividade a regra é precisa e objetiva. A chave deverá ser adicionada ao lado “n” (muitos) do relacionamento. No caso do exemplo apresentado, estamos falando de adicionar uma chave nas propriedades da entidade “funcionário”.

Por fim, temos a conectividade “muitos para muitos”. O tipo “muitos para muitos” é comum em um MER. Geralmente é utilizado quando vários registros de uma entidade se relacionam com vários registros da outra entidade. Algumas literaturas afirmam que este tipo de relacionamento não existe, pois para que ele possa ser concebido é necessária a geração de uma terceira entidade, caracterizando assim dois relacionamentos “um para muitos” entre elas, que ganha o sentido de “muitos para muitos”. No entanto, devemos sim considerar este tipo, pois quando o modelo foi concebido, este é um princípio básico da criação. Além disso, ao assumirmos dois relacionamentos com conectividade “um para muitos”, algumas propriedades são perdidas e podem comprometer as regras de integridade referencial. Oportunamente falaremos mais sobre isso.



Se invertemos a posição das entidades, ou seja, à esquerda funcionário e à direita departamento, temos o exemplo transformado em “muitos para um”. Portanto, não há diferença entre “um para muitos” e “muitos para um”, exceto o fato da posição gráfica deles. Assim, o exemplo utilizado é adequado a ambos os casos.



No exemplo utilizado, você imagina por que devemos incluir a chave de ligação no lado do funcionário? Vejamos. Como o departamento pode lotar mais de um funcionário, se tivermos que criar chaves nas propriedades da entidade departamento, quantas seriam? Uma, duas, três, cinco... Não sabemos, pois há um indicativo de que pode ser “n” (muitos). Já ao considerarmos que um funcionário está lotado em apenas “1” (um) departamento, logo fica fácil estabelecer a ligação entre as entidades, acrescentando uma chave nas propriedades da entidade funcionário.



Figura 42: Exemplo de conectividade "muitos para muitos". Fonte: O autor, adaptado por DME/FURB (2019).

Vejamos a aplicação da expressão já conhecida:

Um(a) [ENTIDADE A] [RELACIONAMENTO] [preposição] **quantos(as)** [ENTIDADE B]?

Substituindo os termos:

Um ALUNO cursa quantas DISCIPLINAS?

Inverte-se a pergunta:

Uma DISCIPLINA é cursada por quantos ALUNOS?

Substituindo os termos, temos: um aluno cursa quantas disciplinas? A resposta é muitas, portanto devemos colocar um "n" ao lado da entidade disciplina. Com os devidos ajustes, inverte-se a pergunta. Uma disciplina é cursada por quantos alunos? A resposta é muitas, portanto devemos colocar um "n" também ao lado do aluno. Veja que nesta pergunta também é desprezado o fato de conter um "n" ao lado da entidade disciplina, ou seja, novamente a pergunta investigativa independe da resposta anterior.

Em uma conectividade "muitos para muitos" não é possível adicionar um elo (chave) em cada uma das propriedades das entidades envolvidas no relacionamento, à luz do que foi feito na conectividade tipo "um para muitos". Neste caso a regra é outra.

O exemplo está indicando que um aluno cursa uma ou mais disciplinas e que uma disciplina por sua vez também pode ser cursada por um ou mais alunos. O elo (chave) deve ser adicionado a uma nova estrutura, uma entidade de ligação que não irá conter apenas um, mas sim ambos os elos (chaves) de cada uma das entidades envolvidas no relacionamento. Você imagina por quê? Vejamos! Como não podemos criar chaves nas propriedades da entidade aluno e nem na entidade disciplina, pois não sabemos quantos seriam necessárias ("muitos para muitos"), a solução é a criação de uma nova estrutura que servirá de ligação.

Outro elemento importante da modelagem entidade-relacionamento diz respeito ao aspecto de condicionalidade. A condicionalidade nada mais é do que a análise da necessidade ou não de uma associação (relacionamento) de fato existir. Estamos falando da obrigatoriedade no relacionamento.

Para alguns, esse elemento é um aspecto que não interfere no projeto de banco de dados. Mas é um tremendo engano pensar assim. A obrigatoriedade no relacionamento será fundamental no momento da criação do projeto físico do banco de dados, pois ele indicará se um determinado campo chave terá ou não seu preenchimento obrigatório quando do cadastramento dos dados.



Mais detalhes sobre esta nova estrutura de ligação criada neste tipo de conectividade abordaremos no conteúdo sobre a derivação do modelo conceitual para o projeto físico.

Vamos a um exemplo. Observe a Figura 43, onde é apresentada parte de um cenário em que um departamento lota vários funcionários.



Figura 43: Exemplo de obrigatoriedade no relacionamento. Fonte: O autor, adaptado por DME/FURB (2019).

A representação gráfica indica que todo funcionário obrigatoriamente deve estar lotado em um departamento. Portanto, a ligação (chave) adicionada nas propriedades da entidade “funcionário” deverá ser preenchida (eis a obrigatoriedade), e esta, por sua vez, deverá conter um valor correspondente na entidade “departamento”.

O oposto da obrigatoriedade, por natureza óbvia, é o relacionamento de não obrigatoriedade. Ele é bastante comum em relacionamentos no MER. Para muitos, essa é a forma de omitir uma informação desconhecida ou imprecisa. Mas atenção: não se utilize deste subterfúgio para ignorar o aspecto da obrigatoriedade, pois, do contrário, os controles de preenchimento ou não do(s) campo(s) chave terão que ser feitos exclusivamente na camada da aplicação.

A Figura 44 ilustra um exemplo com o mesmo cenário apresentado no caso anterior.



Figura 44: Exemplo de não obrigatoriedade no relacionamento. Fonte: O autor, adaptado por DME/FURB (2019).

No exemplo acima, a representação gráfica indica que um funcionário não obrigatoriamente deve estar lotado em um departamento. Portanto, a ligação (chave) adicionada nas propriedades da entidade “funcionário” não necessariamente deverá ser preenchida (eis a não obrigatoriedade) e poderá ser nula (sem valor definido). A não obrigatoriedade também só tem sentido em uma relação do tipo “um para muitos”.

Agora que você conhece os elementos envolvidos no MER, suas especificidades e particularidades, é necessário avançarmos para o processo de normalização. A normalização é uma técnica relevante na obtenção de modelos de dados ideal, ou seja, que atenda na sua totalidade às necessidades do usuário considerando o cenário (minimundo).



A obrigatoriedade só tem sentido em uma relação do tipo “um para muitos”. Quanto avançarmos para as atividades práticas, isso ficará evidenciado.

2.3 Normalização de dados

Para Alves (2014), a normalização é um processo de refinamento do esquema de banco de dados que visa eliminar possíveis redundâncias (dados repetidos em entidades), sanar problemas de dependências parciais entre atributos e reduzir ao mínimo as anomalias de inclusão, alteração e exclusão de dados. O processo é dividido em várias etapas, que são denominadas tecnicamente de formas normais (FNs). Na sua aplicação são efetuados diversos testes com o objetivo de certificar-se de que o esquema satisfaz determinadas condições presentes na especificação de cada FN. A partir desses testes, as relações são decompostas em relações menores, conforme a necessidade. Como exemplo, uma relação pode ser dividida em duas ou mais, dependendo da situação.

A forma normal de uma relação indica o grau de normalização em que ela se encontra. Academicamente falando, existem cinco formas normais, embora apenas as três primeiras já sejam suficientes para se ter uma boa definição da estrutura do banco de dados. No fim da normalização tem-se a resposta à principal pergunta que surge logo no início de um projeto: quantas tabelas são necessárias em nosso banco de dados? As regras de normalização permitem que bancos de dados robustos e eficientes possam ser criados e facilmente alterados. Se essas regras forem seguidas com cuidado, o sistema todo (banco de dados e sistema) será bastante flexível e confiável.

Conforme antecipamos acima, três são as FNs mais usadas e aplicadas na modelagem de dados para sistemas computacionais. A partir de agora faremos uso de um exemplo para aplicar a definição de cada FN. Mas antes você vai conhecer a definição, o objetivo e a aplicação de cada uma delas.

É importante que você saiba de antemão que o fato de você conhecer conceitos de orientação a objetos favorecerá o seu entendimento. Mas isso não o dispensa da atenção e de cuidados, pois são aos pequenos detalhes que devemos nos ater quando aplicamos as FNs. A partir deste momento, nosso foco abrange também os atributos, e não mais somente as entidades e os relacionamentos.

O exemplo que iremos utilizar é considerado um minimundo completo por Alves (2014) e Heuser (2011), por apresentar os elementos necessários para a identificação das fragilidades e consequente demonstração da aplicação das FNs. Procure observar em detalhes o conteúdo da nota fiscal abaixo. No cabeçalho temos as informações da empresa e o número do cupom fiscal. Em seguida temos as informações do cliente, do vendedor e dos produtos contidos na nota, conforme ilustrado na Figura 45.



É possível que você estranhado o uso do termo “tabela” ao questionarmos: “quantas tabelas são necessárias em nosso banco de dados”? Adiante falaremos sobre a derivação do MER conceitual para o projeto físico do banco de dados. Em bancos relacionais, uma entidade é representada por uma tabela. Aos poucos vamos nos apropriando e se acostumando com estes termos.

Livraria Chega Mais Ltda. Av. Cosmos Bittencourt, 2345 - CEP 89902-123 Blumenau - Santa Catarina - Brasil				N.E.: 001234	
NOTA FISCAL - VENDA A CONSUMIDOR					
Cliente: Maria S. da Silva Endereço: Av. Martin Luther, s/n			Vendedor: Zé do Balcão		
			Data: 12/12/12		
Cód.	Descrição	UN.	QUANT.	Preço unitário	Preço total
123	Régua acrílico 30 cm	un.	1	0,80	0,80
234	Penal escolar ref. 1	un.	1	2,60	2,60
345	Grampo 0.08"	cx.	2	1,50	3,00
Total da nota					6,40

Figura 45: Cenário exemplo para a aplicação da normalização. Fonte: O autor, adaptado por DME/FURB (2019).

Difícilmente um analista/projetista iria propor a criação de uma tabela a partir do cenário apresentado como exemplo. Mesmo assim, vamos considerar que isso viesse a acontecer. Neste caso, uma descrição em linguagem natural iria possuir a seguinte estrutura, visualizada na Figura 46.

Tabela “nota fiscal”:

Número, NomeCliente, EndereçoCliente, NomeVendedor, DataEmissão, CodProd1, DescrProd1, UnProd1, QtdeProd1, PreçoProd1, PreçoTotal1, CodProd2, DescrProd2, UnProd2, QtdeProd2, PreçoProd2, PreçoTotal2, CodProd3, DescrProd3, UnProd3, QtdeProd3, PreçoProd3, PreçoTotal3, CodProd4, DescrProd4, UnProd4, QtdeProd4, PreçoProd4, PreçoTotal4, CodProd5, DescrProd5, UnProd5, QtdeProd5, PreçoProd5, PreçoTotal5, CodProd6, DescrProd6, UnProd6, QtdeProd6, PreçoProd6, PreçoTotal6, TotalNota

Figura 46: Tabela criada a partir do cenário exemplo. Fonte: O autor, adaptado por DME/FURB (2019).



Considerando a estrutura apresentada, será que o analista/projetista encontrará problemas na implementação e operacionalidade do sistema computacional?

Acredito que há dúvidas acerca do quão a estrutura proposta é ineficaz. Nela temos a possibilidade de apresentar a redundância nos dados, ausência ou excesso de atributos para manter os dados, entre outros. Vamos iniciar a aplicação das FNs. Observe com atenção o que estabelece cada uma das FNs, sua aplicação e o resultado obtido.

Uma tabela está na primeira forma normal e nenhum dos seus atributos tem domínio multivalorado. Entende-se como multivalorados os atributos que apresentam a possibilidade de receber um ou mais valores. No exemplo temos o conjunto de atributos que representam os itens (produtos) que compõem a “nota fiscal”. Para evitar a reserva de espaços que não serão utilizados, bem como a falta de espaços que serão necessários – pois vemos no exemplo que há a indicação da existência de seis itens

(produtos) na “nota fiscal” –, devemos projetar estes atributos multivalorados para outra tabela, levando um atributo como ligação.

Vamos identificar na estrutura proposta inicialmente pelo nosso analista/projetista onde estão os dados multivalorados. A Figura 47 ilustra o detalhe na tabela nota fiscal onde temos os atributos multivalorados.

Número, NomeCliente, EndereçoCliente, NomeVendedor, DataEmissao,
 CodProd1, DescrProd1, UnProd1, QtdeProd1, PrecoProd1, PrecoTotal1,
 CodProd2, DescrProd2, UnProd2, QtdeProd2, PrecoProd2, PrecoTotal2,
 CodProd3, DescrProd3, UnProd3, QtdeProd3, PrecoProd3, PrecoTotal3,
 CodProd4, DescrProd4, UnProd4, QtdeProd4, PrecoProd4, PrecoTotal4,
 CodProd5, DescrProd5, UnProd5, QtdeProd5, PrecoProd5, PrecoTotal5,
 CodProd6, DescrProd6, UnProd6, QtdeProd6, PrecoProd6, PrecoTotal6,
 TotalNota

Figura 47: Tabela criada a partir do cenário exemplo. Fonte: O autor, adaptado por DME/FURB (2019).

Ao aplicarmos a 1.^a FN, teremos a seguinte estrutura:

Tabela “nota fiscal”:

Número, NomeCliente, EndereçoCliente, NomeVendedor, DataEmissão,
 TotalNota

Tabela “item nota fiscal”:

Número, CodProd, DescrProd, UnProd, QtdeProd, PrecoProd, PrecoTotal

Observe que houve um desmembramento da tabela “nota fiscal” para uma derivada que denominamos de “item nota fiscal”. Nessa nova estrutura é possível observar o atributo “número” como ligação entre elas. Logo, é possível deduzir que houve um relacionamento entre estas tabelas. Adiante facilmente será possível identificar a conectividade entre as tabelas. Agora observe como ficam os dados armazenados após o redesenho da estrutura (Figura 48).

Numero	NomeCliente	EnderecoCliente	NomeVendedor	DataEmissao	TotalNota
001234	Maria de S. Silva	Av. Martin Luther, s/n	Zé do Balcão	12/12/12	6,40

Numero	CodProd	DescrProd	UnidProd	QtdeProd	PrecoProd	PrecoTotal
001234	123	Régua acrílico 30 cm	Un	1	0,80	0,80
001234	234	Penal escolar ref. 1	Un	1	2,60	2,60
001234	345	Grampo 0,08'	cx	2	1,50	3,00

Figura 48: Nova distribuição dos dados a partir da aplicação da 1.^a FN.
 Fonte: O autor, adaptado por DME/FURB (2019).



Temos dados multivalorados (ou domínio multivalorado) quando um atributo apresenta valores repetitivos ou tem mais de um valor, ou seja, não é atômico.



Observe que ao aplicarmos a 1.^a forma normal (FN) eliminamos a possibilidade de espaços que poderiam ficar ociosos e até mesmo a ausência de espaços para a inclusão de mais produtos à nota fiscal. Esse é o principal reflexo da aplicação da 1.^a FN.

Passamos agora para a aplicação da 2.^a FN. Esta prevê que projetemos os dados que dependem unicamente de um identificador (chave) para outra estrutura, mantendo esta ligação por meio do identificador. O objetivo desta FN é evitar que sejam mantidas informações sobre um conjunto que tem interseção com o conjunto representado na tabela, mas existência independente.

No exemplo utilizado podemos observar que a tabela “itens nota fiscal” apresenta conjuntos de dados relacionados diretamente ao “produto”. Assim, a aplicação da 2.^a FN estabelece que projetemos os atributos que dependem funcionalmente do “número” (chave) para fora da tabela, levando-a (a chave) para servir como ligação para refazer a ligação e recuperar o conteúdo da tabela original.

Vamos identificar, na estrutura ajustada após a aplicação da 1.^a FN onde ocorre a interseção de dados com existência própria, o seguinte:

Tabela “item nota fiscal”:

Número, **CodProd**, **DescrProd**, UnProd, QtdeProd, **PrecoProd**, PrecoTotal

Ao aplicar a 2.^a forma normal, teremos uma estrutura desmembrada em duas, ou seja, mantemos a “item nota fiscal” e criamos uma derivada que denominamos de “produto”, pois armazenará dados relacionados a ele, ou seja:

Tabela “item nota fiscal”:

Número, CodProd, QtdeProd, PrecoProd (estático), PrecoTotal

Tabela “produto”:

CodProd, DescrProd, UnProd, PrecoProd (dinâmico)

Agora observe como ficam os dados armazenados após o redesenho da estrutura (Figura 49).

Numero	CodProd	QtdeProd	PrecoProd	PrecoTotal
001234	123	1	0,80	0,80
001234	234	1	2,60	2,60
001234	345	2	1,50	3,00

CodProd	DescrProd	UnidProd	PrecoProd
123	Régua acrílico 30 cm	Un	0,80
234	Penal escolar ref. 1	Un	2,60
345	Grampo 0.08"	Cx	1,50



A aplicação da 2.^a forma normal possibilitará a economia de espaço (estrutura “item nota fiscal”) e a reutilização de dados já armazenados (estrutura “produto”). Esse é o principal reflexo da aplicação da 2.^a FN.

Figura 49: Nova distribuição dos dados a partir da aplicação da 2.^a FN.
Fonte: O autor, adaptado por DME/FURB (2019).

A aplicação da 3.^a FN é consolidada a partir do momento em que esteja na 2.^a FN e que não apresente dependência funcional transitiva entre seus atributos. No exemplo temos duas situações que ilustram esta dependência funcional. O primeiro envolve os atributos que representam o “cliente”. Observe que há atributos (NomeCliente, EndCliente) que dependem e se complementam para dar sentido ao “cliente”. Há também um atributo que representa o “vendedor”. Mesmo estando isolado no exemplo, este deve ser considerado para a aplicação da 3.^a FN. Esta afirmação ocorre pelo fato de haver, naturalmente, outros atributos associados ao “vendedor”, porém não presentes no cenário.

Assim, a aplicação da 3.^a FN prevê que separaremos subconjuntos insertos em um superconjunto e dessa forma podemos evitar a redundância nas informações. No exemplo, a redundância estaria na necessidade de cadastrarmos, repetidas vezes, os dados do “cliente” e do “vendedor”, no caso de eles estarem presentes em mais de uma “nota fiscal”.

Vamos identificar, na estrutura ajustada após a aplicação da 2.^a FN, onde ocorre a dependência funcional transitiva e/ou o atributo que possibilite a redundância nas informações, a partir das seguintes estruturas:

Tabela “nota fiscal”:

Numero, **NomeCliente**, **EndCliente**, **NomeVendedor**, DataEmissao, TotalNota

Ao aplicar a 3.^a FN, teremos a criação de novas estruturas como “cliente” e “vendedor”, conforme vemos:

Tabela “nota fiscal”:

Numero, CodCliente, CodVendedor, DataEmissao, TotalNota

Tabela “cliente”:

CodCliente, NomeCliente, EndCliente

Tabela “vendedor”:

CodVendedor, NomeVendedor

É importante observar que no exemplo não há um identificador (chave) para servir de ligação entre a estrutura “nota fiscal” e as novas criadas (cliente e vendedor). Por esta razão é que foram criados atributos que serviram de ligação (chave). Esta é diferença entre a aplicação da 2.^a FN e a 3.^a FN. Vamos observar como ficam os dados armazenados após o redesenho da estrutura (Figura 50):



Dependência funcional transitiva é a situação em que um atributo depende de outro, e este segundo depende de um terceiro.

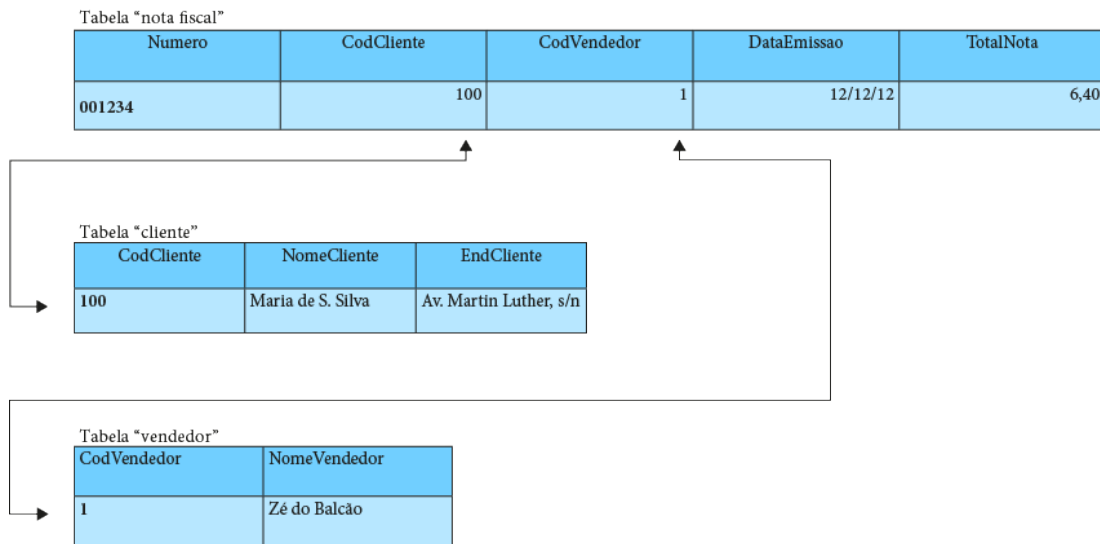


Figura 50: Nova distribuição dos dados a partir da aplicação da 3.ª FN.
Fonte: O autor, adaptado por DME/FURB (2019).

Após a aplicação das três primeiras e principais FNs, observe como ficou o redesenho da estrutura considerando o conjunto de dados inicial (Figura 51).

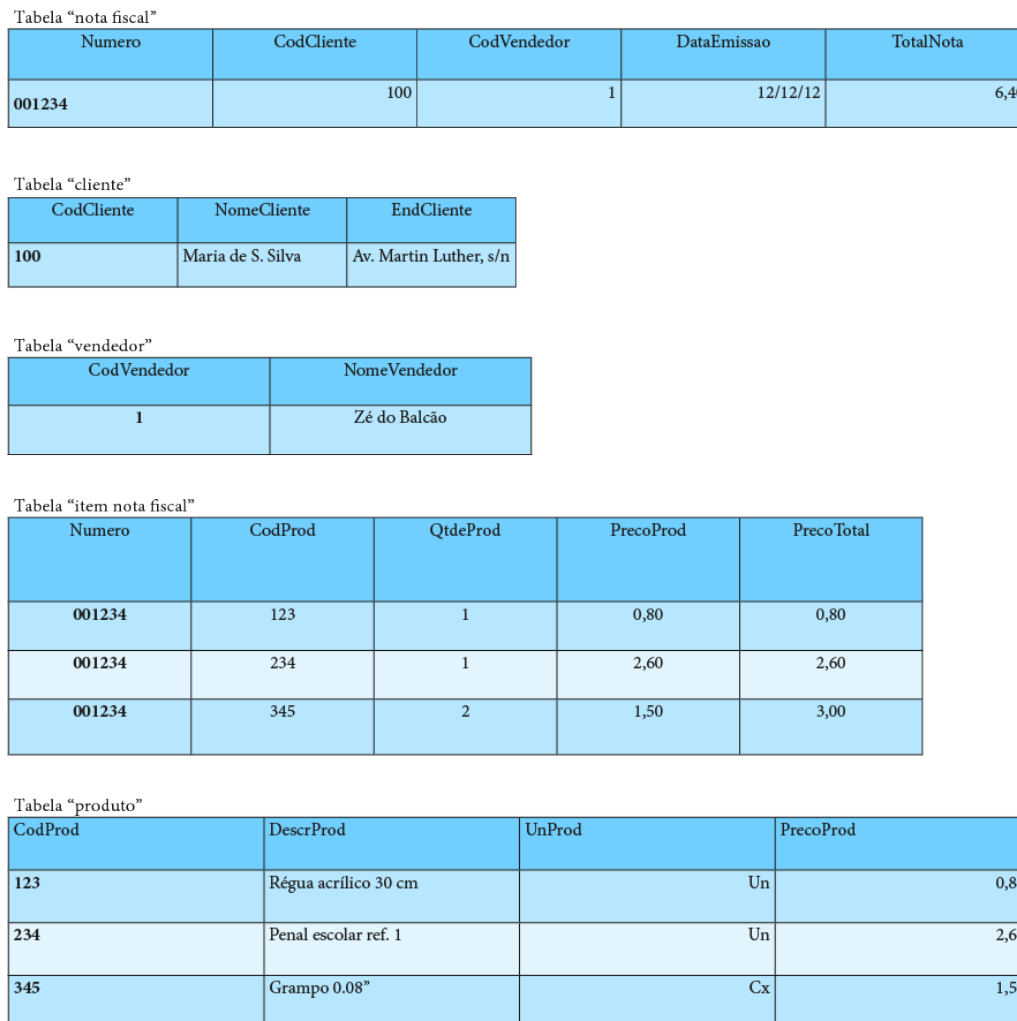


Figura 51: Nova distribuição dos dados a partir da aplicação da 3.ª FN.
Fonte: O autor, adaptado por DME/FURB (2019).

Você teve a oportunidade de conhecer as três primeiras FNs. Como já dito, essas são as mais importantes e aplicáveis. Porém, é necessário registrar que a literatura apresenta a 4.^a e a 5.^a FN, além da forma normal de Boyce Codd (FNBC). A aplicação destas é muito restrita e, portanto, desconsiderada no campo profissional. Mesmo assim, caso deseje saber mais sobre elas, recomendamos a leitura da obra de Chen (1990).

Como você pôde observar, a normalização é fundamental para a modelagem de dados relacional. Contudo, é importante que você faça uso do senso crítico para avaliar as circunstâncias pelas quais um atributo ou conjunto de atributos (ou até mesmo uma tabela) deve ou não ser segmentado através da normalização.

Lembre-se de que a normalização tem inúmeras vantagens, porém no longo prazo poderá requerer mais recursos computacionais para atender às requisições do sistema, pois o crescimento dos dados em um banco de dados é exponencial. Mas isso não deve ser, para você, ao menos neste momento, objeto de ponderação e preocupação, visto que tiraria o foco do tema proposto. Oportunamente você terá a chance de conhecer uma técnica já discutida e defendida por alguns profissionais, denominada “desnormalização”.

Agora convidamos você a avançar para o projeto físico de banco de dados. Trata-se da etapa em que iremos escolher qual SGBD Relacional iremos utilizar.

2.4 Projeto físico de dados

Nas seções anteriores tivemos a oportunidade de conhecer a modelagem “entidade-relacionamento” (MER) e a normalização de dados, ambas técnicas da análise de dados que compõem o modelo conceitual de um banco de dados. Agora vamos abordar o conceito e as características do projeto físico de banco de dados e conhecer as técnicas para transformar a MER, resultado do modelo conceitual, em um projeto físico de banco de dados. Abordaremos também aspectos relevantes sobre a derivação de um modelo orientado a objetos (diagrama de classes) para o projeto físico do banco de dados. O projeto físico é a última etapa antes da consolidação das estruturas de dados projetadas em estruturas internas (físicas) do banco de dados (tabelas e restrições de integridade).

Mas antes convém abordarmos brevemente o conceito de projeto lógico. O projeto lógico, também conhecido como modelo lógico, é a etapa intermediária abstrata na qual as definições conceituais representadas graficamente no MER são transformadas em estruturas físicas através do projeto físico de dados. Neste contexto podemos afirmar que o projeto lógico amplia a atenção do analista/projetista para os atributos e seus tipos de dados, assim como as regras de negócio do sistema, sobre os quais os dados serão submetidos. O resultado desta etapa é o esquema lógico de dados. Tomemos como exemplo projetos de banco de dados produzidos por uma ferramenta CASE – por exemplo, a DB Desinger Fork. Nessa ferramenta os atributos já estão vinculados aos tipos de dados, índices e restrições, entre outras propriedades do MySQL. Pois bem, podemos afirmar então que a ferramenta CASE DB Designer Fork trabalha com o projeto lógico e seu resultado é um esquema ou modelo lógico de dados para o MySQL.

Portanto, o modelo lógico é abstrato, ou seja, ocorre implicitamente no momento que definimos qual SGBD irá comportar as estruturas definidas no MER. A seguir abordamos a etapa do projeto físico do banco de dados, começando pela derivação a partir de um MER.

2.4.1 Derivação a partir do MER

O projeto físico dos dados é a etapa na qual as estruturas conceituais apuradas e representadas através do MER são criadas através dos princípios do modelo relacional e das características do SGBD escolhido. Portanto, é fundamental que você saiba que o projeto físico resultante de um MER pode apresentar características distintas em relação ao resultado da transformação do mesmo MER para outro SGBD. Isso ocorre, por exemplo, pelas diferenças entre os tipos de dados e índices utilizados.

Atualmente boa parte das ferramentas CASE, quando necessário, realizam a transformação do MER para o projeto físico do banco de dados definido. Independentemente disso, você terá a oportunidade de conhecer as principais regras de derivação do MER para o projeto físico.

A 1.^a regra indica que cada entidade do MER gerará uma tabela no projeto físico. Esta é a regra mais simples e básica do processo de derivação. A entidade é a unidade básica de armazenamento no projeto físico, que passa a se chamar de tabela. Observe a seguir a aplicação desta regra na Figura 52.

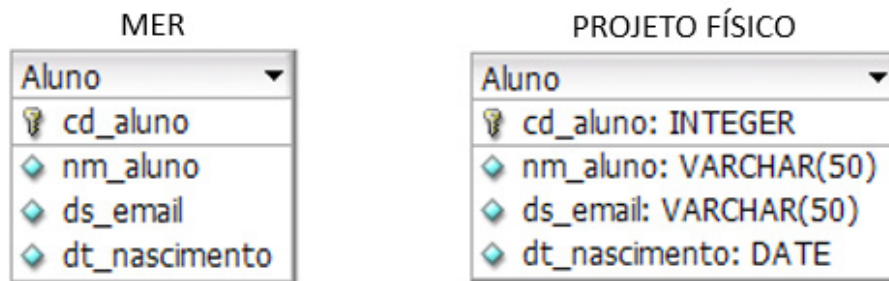


Figura 52: Representação da derivação da entidade em tabela.
Fonte: O autor, adaptado por DME/FURB (2019).



As ilustrações acima, bem como as demais que você verá nos próximos exemplos, foram geradas na ferramenta CASE DBDesigner Fork.

Cada subentidade também gerará uma entidade, cuja chave primária é a chave da superentidade correspondente. Neste caso, o dado que identifica o tipo de subentidade deve ser incluído na tabela da superentidade como se fosse uma chave estrangeira. Observe a aplicação desta 2.^a regra através da ilustração da Figura 53.

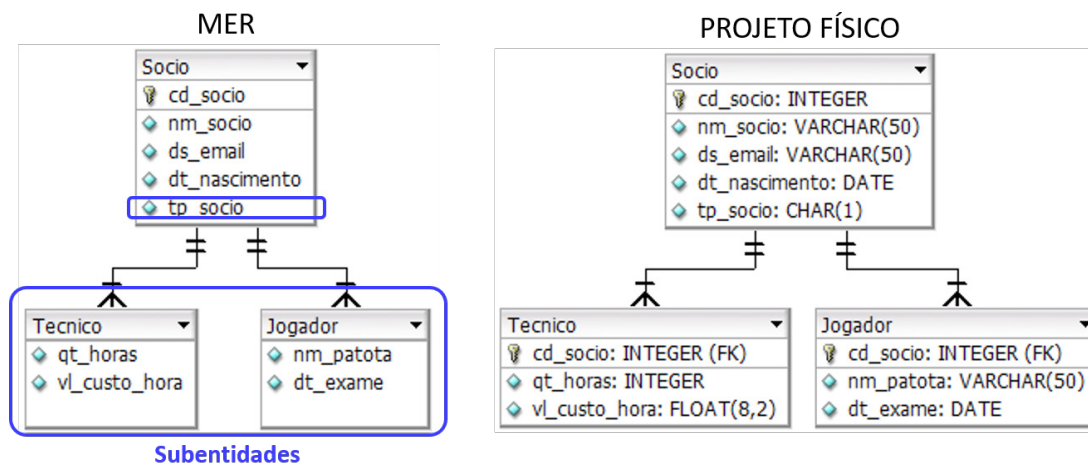


Figura 53: Representação da derivação da entidade com dependência.
Fonte: O autor, adaptado por DME/FURB (2019).

Outra regra (4.^a) que de alguma forma você já deve a oportunidade de conhecer está associada ao relacionamento com conectividade “muitos para muitos”. Nesta, a estrutura resultante do relacionamento necessariamente deve gerar uma nova

tabela, cuja chave primária será a concatenação das chaves primárias das entidades relacionadas. Confira a aplicação desta regra (Figura 54).

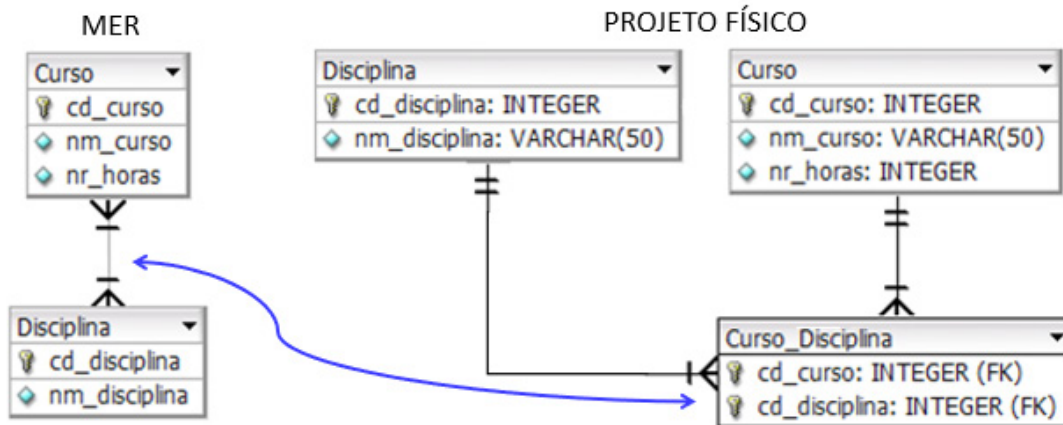


Figura 54: Representação da derivação de relacionamento “muitos para muitos”.
Fonte: O autor, adaptado por DME/FURB (2019).

Como estamos tratando da derivação envolvendo relacionamentos, passamos agora para a regra (5.^a) envolvida no relacionamento com conectividade “um para muitos”. Nesta devemos incorporar junto à tabela resultante da entidade que ocupa o lado “n” (muitos) do relacionamento uma chave estrangeira equivalente à chave primária da tabela do lado “1” (um). Confira a aplicação desta regra ilustrada na Figura 55.

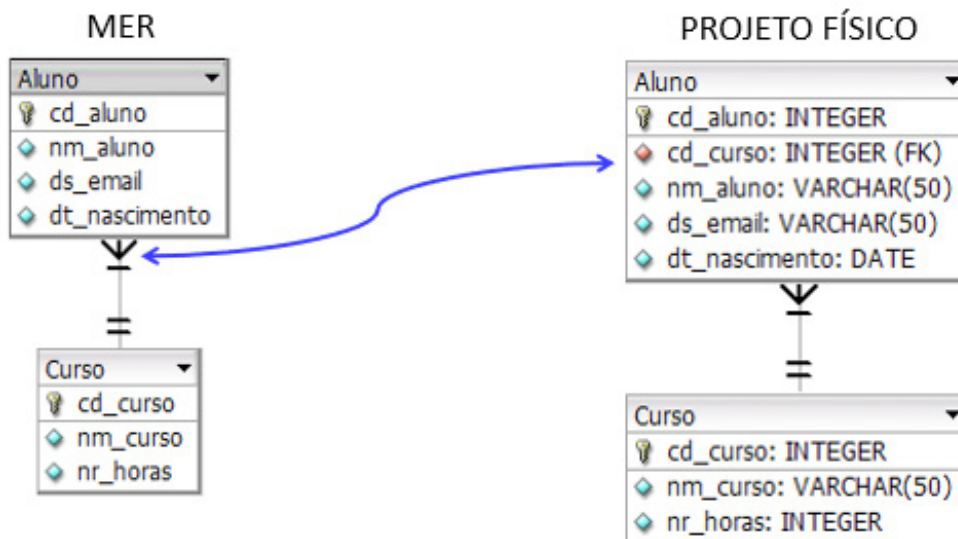


Figura 55: Representação da derivação de relacionamento “um para muitos”.
Fonte: O autor, adaptado por DME/FURB (2019).

Temos, ainda, a derivação envolvendo o relacionamento de conectividade “um para um”. Neste é necessária a inclusão de uma chave estrangeira equivalente à chave primária da outra entidade que resultará em tabela. Neste caso fica a critério do analista/projetista a escolha do lado em que receberá esta chave. Uma dica é observar a estrutura

que demandará menor volume de dados. Ela será uma séria candidata a receber o elemento de ligação (chave). Confira a aplicação desta regra (6.^a) ilustrada na Figura 56.

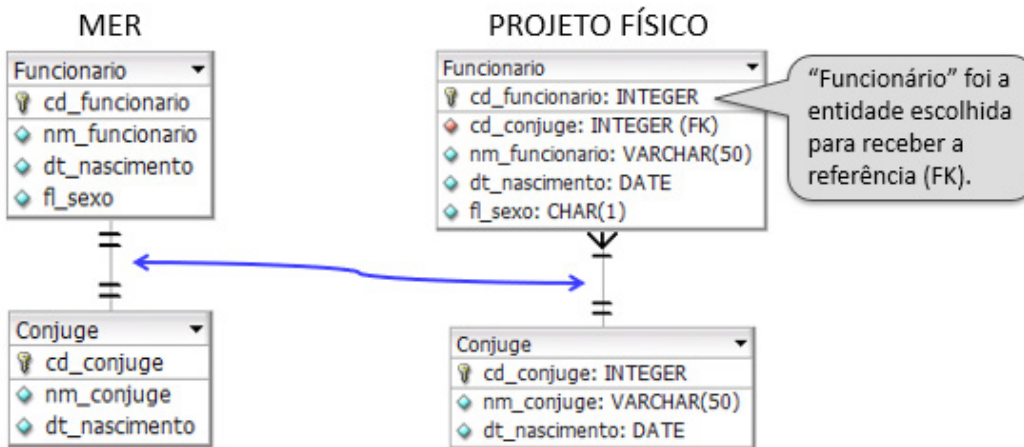


Figura 56: Representação da derivação de relacionamento “um para um”.
Fonte: O autor, adaptado por DME/FURB (2019).



Observe que a ilustração gerada a partir da ferramenta CASE DBDesigner em que temos um relacionamento de conectividade “um para um” é transformado, automaticamente, em relacionamento de conectividade “um para muitos”. Na prática isso ocorre pela necessidade da indicação de qual tabela irá receber o atributo de ligação. Neste momento, podemos concluir que, implicitamente, ocorreu uma mudança conceitual da conectividade do relacionamento.

Por fim destacamos ainda que ao utilizar a ferramenta CASE DBDesigner Fork você poderá alternar a visualização do modelo de dados entre conceitual e físico com facilidade, visto que ela faz essa transformação de forma automática para você. Lembre-se: na ferramenta CASE DBDesigner, por padrão, todo projeto é construído tendo como SGBD o MySQL.

2.4.2 Derivação a partir de um projeto orientado a objetos

Ao longo da sua trajetória no curso você pôde perceber que é cada vez maior a especificação de sistemas computacionais através dos conceitos das orientações a objetos. É possível então que você se pergunte: por que não utilizamos a *Linguagem de Modelagem Unificada* (UML) para a definição da estrutura de dados que será persistida? Existem literaturas que apontam para esse caminho, mas ele ainda é pouco explorado. E por que isso ocorre? A explicação é simples. A representação conceitual de um modelo de dados, através de um diagrama de classes, por exemplo, é insuficiente quanto às reais estruturas a serem derivadas para o modelo de banco de dados relacional. É justamente isso que vamos explorar a partir de agora. A seguir vamos indicar, por meio de exemplos, algumas diretrizes que poderão ajudá-lo no momento da transformação do diagrama de classes em um modelo “entidade-relacionamento” (MER).

É importante ressaltar que nosso foco agora não é discutir a modelagem do diagrama de classe, tampouco o da especificação através da UML. Vamos utilizar o diagrama de classes como modelo orientado a objetos que desejamos transformar em modelo para o banco de dados relacional.

Vamos conhecer agora algumas diretrizes básica para que possamos derivar um diagrama de classe para um projeto de banco de dados relacional. A primeira indica que uma classe se torna uma tabela. A partir desta definição temos que:

- o atributo de uma classe torna-se uma coluna em uma tabela;
- o tipo de atributo de uma classe torna-se um tipo de coluna em uma tabela;
- se o atributo de uma classe indicar marcação {nullable}, a coluna da tabela poderá ser nula; do contrário, será de preenchimento obrigatório;
- se o atributo de uma classe tiver valor inicial, a coluna na tabela deverá apresentar a propriedade default;
- caso não haja definição de atributos identificadores (únicos) em classes sem generalização, deve-se criar uma coluna que servirá de chave primária da tabela;
- se o atributo possuir marcador {alternate}, deve-se usar a restrição de UNIQUE KEY na tabela.



Alguns conceitos apresentados aqui você verá na prática quando conhecer em detalhes a linguagem SQL.

Dando sequência às diretrizes, temos os cenários que envolvem classes e subclasses, ligações através de associações por sinal, a mais utilizada representação de relações entre classes de dados e que você irá encontrar com maior frequência:

- nos casos envolvendo classes e subclasses, deve-se acrescentar uma coluna que servirá como elo entre as tabelas;
- acrescentar a cláusula *check* para restrições explícitas;
- criar chaves estrangeiras em classes com cardinalidade 0..1 e 1..1 na associação;
- criar chave primária para agregação composta com chave estrangeira para a tabela agregada;
- otimizar classes de associação binária;
- criar tabelas para associação n..n e ternárias;
- criar as restrições de chaves primárias e estrangeiras nas associações n..n e ternárias.

Considere parte de um cenário em que temos uma classe “nota fiscal” e uma classe “cliente”. Entre elas há uma relação de associação. A representação do diagrama de classes pode ser vista na Figura 57.

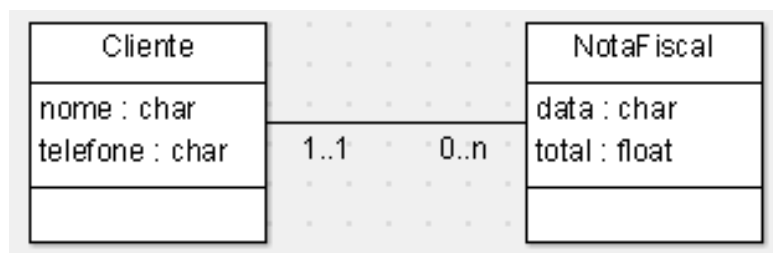


Figura 57: Exemplo de diagrama de classe com associação.
Fonte: O autor, adaptado por DME/FURB (2019).

Ao aplicarmos as diretrizes para a transformação para um projeto físico de dados temos a representação gráfica da Figura 58.

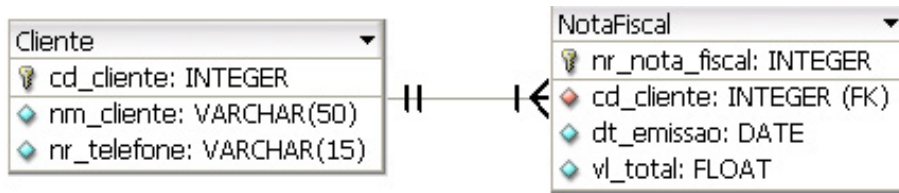


Figura 58: Exemplo de associação em projeto físico de banco de dados.
Fonte: O autor, adaptado por DME/FURB (2019).



A associação no diagrama de classes (UML) é equivalente ao relacionamento no modelo “entidade-relacionamento”.

Seguimos agora com um exemplo envolvendo composição. Considere o exemplo envolvendo “nota fiscal” e “item de nota fiscal”. A representação do diagrama de classes correspondente é apresentada na Figura 59.

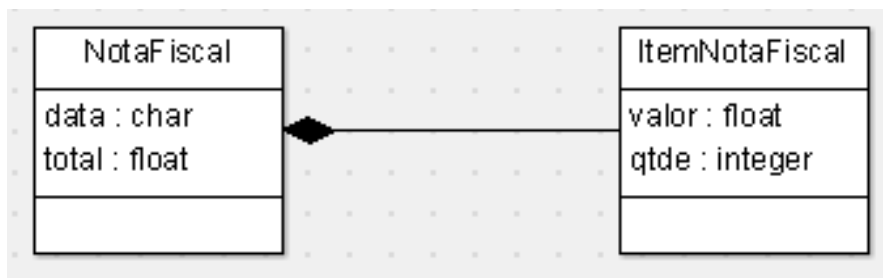


Figura 59: Exemplo de diagrama de classe com composição.
Fonte: O autor, adaptado por DME/FURB (2019).

A aplicação das diretrizes para a transformação de uma composição para um projeto físico de dados é representada da Figura 60.

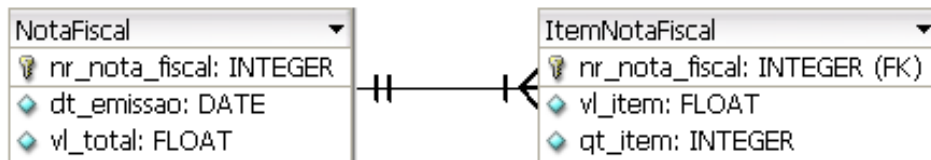


Figura 60: Exemplo de associação em projeto físico de banco de dados.
Fonte: O autor, adaptado por DME/FURB (2019).

No exemplo anterior observamos que foi criada uma chave primária na tabela “nota fiscal”. Esta, por sua vez, foi utilizada como chave estrangeira para servir de ligação com a tabela “item nota fiscal”, assim como parte da chave primária desta última tabela. É fundamental ressaltar que, possivelmente, outros atributos possam ser adicionados à tabela “item nota fiscal”, inclusive como parte da chave primária.

Conclusão

Neste Ciclo você teve a oportunidade de conhecer conceitos, princípios básicos e técnicas para construção de modelos entidade-relacionamento (MER). A partir do MER você foi apresentado às diretrizes para derivar um modelo para um projeto físico de banco de dados relacional.

Em dado momento, falamos brevemente do uso de ferramenta CASE. As ferramentas CASE são parte do processo de construção de modelos de dados. Elas estarão inseridas nas atividades práticas da disciplina. Aliás, serão muitas as atividades práticas.

Assim como não se aprende programação lendo, exclusivamente, livros ou mesmo assistindo a vídeos nas redes sociais, modelar projetos de bancos de dados também requer experimentação. Por meio do exercício constante e intensivo você irá potencializar o desenvolvimento de aptidões para o sucesso nessa atividade.

Referências

ALVES, William Pereira. **Banco de dados**. São Paulo: Erica, 2014.

CARDOSO, Virginia M. **Sistemas de banco de dados**. São Paulo: Saraiva, 2008.

CHEN, Peter. **Gerenciando banco de dados**: a abordagem entidade-relacionamento para projeto lógico. São Paulo: McGraw-Hill, 1990.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. Porto Alegre: Bookman, 2011.

MACHADO, Felipe Nery Rodrigues. **Banco de dados**: projeto e implementação. 3. ed. São Paulo: Erica, 2014.