

Unidade 1

Fundamentos de Bancos de Dados

Aula 1

Introdução aos Bancos de Dados

Introdução aos bancos de dados

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula, serão apresentados os conceitos introdutórios de banco de dados, cruciais na compreensão da modelagem de dados. Bancos de dados são coleções lógicas, coerentes e inter-relacionadas de arquivos que visam auxiliar e atender às necessidades de uma comunidade de usuários. Para ilustrar esta definição, vamos supor que uma pessoa se matriculará em um curso. Ela vai, primeiramente, até à secretaria da faculdade e fornece suas informações pessoais e acadêmicas, entre muitas outras que serão registradas em um banco de dados. Para criar boletos de pagamento, por exemplo, ela não precisará inserir seus dados novamente, porque o departamento financeiro já possui as informações concedidas na hora da matrícula. Dois sistemas diferentes, portanto, utilizam o mesmo banco de dados, o financeiro e o de cadastro. Pode-se afirmar que ambos se valem da mesma base de dados.

Logo, para saber mais sobre o mundo dos bancos de dados, concentre-se nesses primeiros conteúdos, os quais são elementares para um melhor aprofundamento em modelagem de dados. Bons estudos!

Vamos Começar!

Na perspectiva de Machado (2020, p. 14), modelagem de dados é o estudo das informações existentes em um contexto sob observação para a construção de um modelo de representação e entendimento de tal contexto. A modelagem de dados, assim, minera as informações que representam um contexto, estruturando-as em um conjunto denominado modelo lógico de dados. Uma das principais características da modelagem de banco de dados é que ela fornece níveis de abstração de dados que omitem do usuário final detalhes sobre o armazenamento dos dados. Não existe a preocupação com um banco de dados tecnologicamente falando. O modelo de dados, então, é um conjunto de conceitos que podem ser utilizados para descrever as estruturas lógicas e físicas de um banco de dados.

Ainda de acordo com o autor (2020, p. 18), um banco de dados pode ser definido como um conjunto de dados devidamente relacionados. Dados são os objetos conhecidos que podem ser armazenados e que possuem um significado implícito, porém o significado do termo banco de dados é mais restrito do que a definição dada anteriormente. Um banco de dados apresenta as seguintes características:

- É uma coleção organizada e lógica de dados que possui um significado inerente; dados dispostos de maneira caótica não podem ser considerados um banco de dados.
- É criado, projetado e preenchido com valores de dados para atender a um propósito específico, contendo um conjunto predefinido de usuários e aplicações.
- Representa uma parte do mundo real, denominada minimundo; qualquer modificação realizada no minimundo reflete-se automaticamente no banco de dados.

A fim de modelar um banco de dados, para Machado (2020, p. 18-20), três níveis de visão de dados devem ser considerados. São eles o conceitual, o lógico e o físico. O modelo conceitual, entendido como a primeira etapa, descreve a realidade do ambiente do problema, constituindo-se em uma perspectiva global dos principais dados e de seus relacionamentos (estruturas de informação), completamente independente dos aspectos de sua implementação tecnológica. Já o modelo lógico inicia-se somente após a criação do modelo conceitual, momento em que se considera uma das abordagens possíveis da tecnologia SGBD (relacional, hierárquica, rede ou orientada a objetos) para estruturação e estabelecimento da lógica dos relacionamentos existentes entre os dados definidos no modelo conceitual. Quanto ao modelo físico, ele será construído a partir do modelo lógico e descreve as estruturas físicas de armazenamento de dados conforme os requisitos de processamento e uso mais econômico dos recursos computacionais.

Deve-se lembrar que um conjunto de dados só pode ser considerado uma base de dados se existir uma relação coerente dentro da qual possam ser extraídas informações de interesse para

uma ou mais organizações.

Um sistema gerenciador de banco de dados (SGBD) é um software cuja finalidade é gerenciar as informações de um banco de dados (também chamada de base de dados). Segundo Alves (2020), em um sistema de banco de dados, os aplicativos (ou aplicações) não têm qualquer conhecimento dos mecanismos relacionados com as operações de gravação e leitura física dos dados. O que eles fazem é simplesmente se comunicar com o software de gerenciamento para recuperar ou armazenar as informações desejadas. Desta forma, diversos programas podem acessar um mesmo banco de dados e qualquer alteração em sua estrutura não pressupõe, necessariamente, modificações nos aplicativos. Assim, o uso de um SGBD adequado à situação ou realidade da organização é de extrema importância para o manuseio de suas bases de dados.

Quando nos referimos ao termo aplicação, estamos falamos de softwares que se beneficiarão dos dados inseridos em um banco de dados. Por exemplo, como mencionamos na introdução desta aula, o sistema do setor financeiro de uma faculdade que utiliza as informações arquivadas no banco de dados do sistema de controle acadêmico da secretaria; ou seja, pode-se afirmar que existe um banco de dados único e centralizado para diversas aplicações utilizarem, conforme mostra a Figura 1.



Figura 1 | Aplicações em um banco de dados.

Na Figura 1, podemos observar uma imagem central, o banco de dados. O acesso ao banco de dados por diversas aplicações necessita de regras específicas para garantir tanto a segurança quanto a integridade das informações inseridas.

Alves (2020) lembra que existem vários critérios que podemos utilizar para categorizar os bancos de dados. Entre esses critérios, estão o modelo de dados, o número de usuários suportados simultaneamente, a localização física e o método de acesso. A partir deles, tem-se os principais tipos de banco de dados: hierárquicos, relacionais e orientados a objetos.

Bancos de dados hierárquicos

É considerado o primeiro tipo de banco de dados de que se tem notícia. Ele foi desenvolvido graças à consolidação dos discos endereçáveis e, devido a essa característica, a organização de endereços físicos do disco é utilizada em sua estrutura.

Em sistemas de banco de dados hierárquicos, encontramos dois conceitos fundamentais: registros e relacionamentos pai-filho. O registro é uma coleção de valores que representam informações sobre uma dada entidade de um relacionamento. Quando temos registros do mesmo tipo, nós os denominamos tipos de registros, que são similares às tabelas/relações do sistema relacional. Os registros que antecedem outros na hierarquia têm a denominação pai e os registros que o sucedem são chamados filhos.

No relacionamento pai-filho, um tipo de registro do lado pai pode se corresponder com vários (ou nenhum) tipos de registro do lado filho.

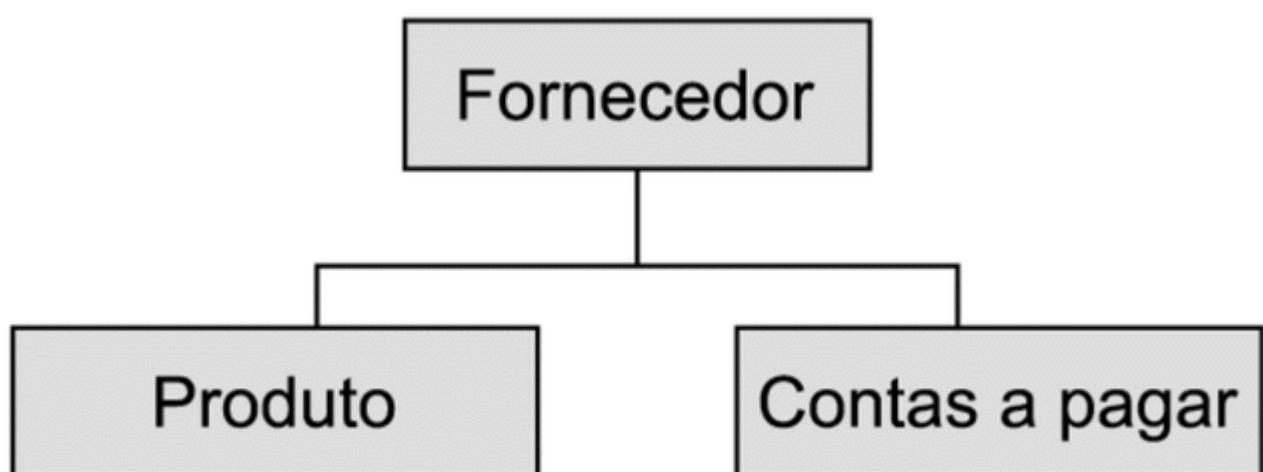


Figura 2 | Organização de registros em um banco de dados hierárquico. Fonte: Alves (2020).

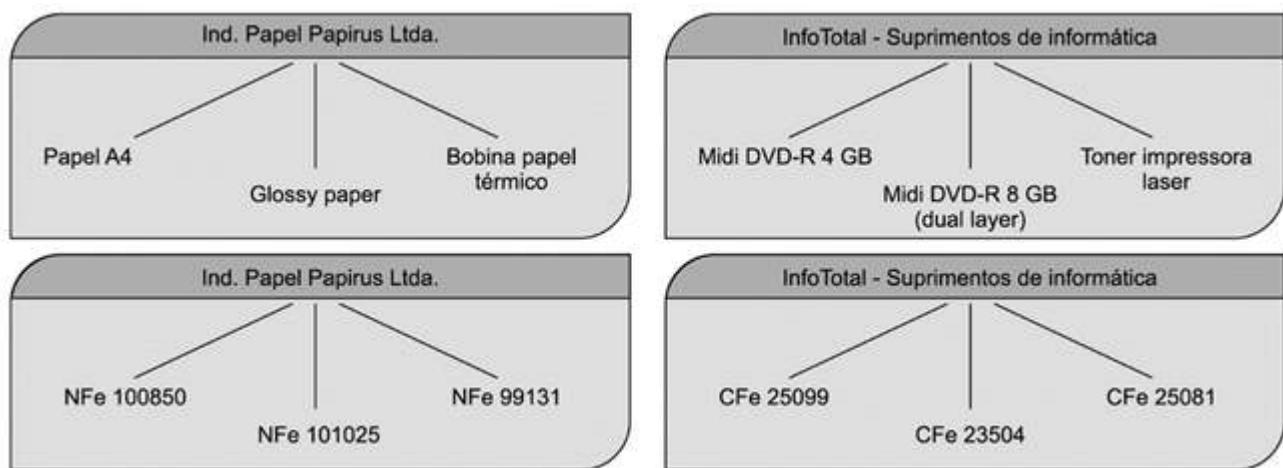


Figura 3 | Exemplo de relacionamento pai-filho em um banco de dados hierárquico. Fonte: Alves (2020).

Podemos perceber, por meio das Figuras 2 e 3, que o esquema hierárquico é estruturado em árvore e que o tipo de registro corresponde a um nó. Assim, temos nós pais e nós filhos.

Siga em Frente...

Bancos de dados relacionais

A maioria dos sistemas de gerenciamento de bancos de dados atualmente em uso pertence ao tipo relacional. Um banco de dados relacional se distingue por sua organização dos dados em relações, comumente chamadas de tabelas, as quais são compostas por linhas e colunas. Portanto, essas tabelas assemelham-se a conjuntos de elementos ou objetos, agrupando, de forma organizada, informações relacionadas a um tópico específico. Nossa atenção, nesta disciplina, estará voltada para o estudo desse tipo de banco de dados.

Da mesma forma que na matemática, podemos efetuar operações entre dois ou mais conjuntos, como, por exemplo, obter os elementos que são comuns a ambas tabelas/relações em um banco de dados relacional. Podemos, também, executar certas operações com elas, como ligar duas ou mais por meio de campos comuns. Quando uma operação de consulta é executada, o resultado é um conjunto de registros que pode ser tratado como uma tabela virtual (que só existe enquanto a consulta está ativa). Isso significa que não há comandos para efetuar uma navegação pelos registros, do tipo MOVE ou SKIP.

Edgard F. Codd (1923-2003) formulou os princípios básicos do sistema de banco de dados relacional em 1968, baseando-se na teoria dos conjuntos e da álgebra relacional. Provavelmente por ter sido um brilhante matemático, ele observou que certos conceitos da matemática poderiam ser aplicados ao gerenciamento de bancos de dados. Em 1985, propôs um conjunto de doze regras para que um banco de dados relacional fosse admitido como tal:

- Regra de informações.
- Regra de acesso garantido.
- Tratamento de valores nulos.
- Catálogo relacional ativo.
- Inserção, exclusão e alteração em bloco.
- Linguagem de manipulação de dados abrangente.
- Independência física dos dados.
- Independência lógica dos dados.
- Regra de atualização de visões.
- Independência de integridade.
- Independência de distribuição.
- Regra não subversiva.

Bancos de dados orientados a objetos

Esse tipo de banco de dados, de acordo com Alves (2020), surgiu em meados de 1980, em virtude da necessidade de armazenamento de dados que não podiam ser guardados pelos sistemas relacionais tradicionais devido às limitações destes. Podemos citar, como exemplo, os sistemas de geoprocessamento GIS (sigla de *Geographic Informations Systems* – Sistemas de Informações Geográficas) e CAD/CAM/CAE (*Computer Aided Design/Computer Aided Manufacturing/Computer Aided Manufacturing*), que são baseados em tipos de dados complexos.

Basicamente, o modelo de dados orientado a objetos é caracterizado pela definição de bancos de dados por meio de objetos, com suas propriedades e operações. Isso significa que um registro é mais parecido com uma classe definida na linguagem C++, C# ou Java, por exemplo. Nesse contexto, o Grupo de Gerenciamento de Dados Objetos (ODMG, em inglês) definiu um padrão de estrutura para bancos de dados orientados a objetos.

São indiscutíveis as vantagens de trabalharmos com orientação a objetos, pelo alto índice de reutilização de código, pela simplicidade de criação de aplicações e pela alta velocidade de desenvolvimento, mas, também, é indiscutível a contínua necessidade de executar a modelagem de dados para que as duas técnicas se complementem na execução de sistemas eficazes, que utilizem tecnologia Java, por exemplo, com um banco de dados relacional.

Vamos Exercitar?

Nesta aula, vimos alguns conceitos relacionados a banco de dados. Aprendemos que, em resumo, um banco de dados é nada mais que um aglomerado de informações organizadas de forma lógica que podem ser dispostas de diferentes maneiras a depender da finalidade de como se quer acessá-los. Assim, para o exemplo das informações de matrícula do estudante, o melhor banco de dados para satisfazer as necessidades da instituição é o relacional, contendo tabelas que se relacionam entre si. Para uma melhor compreensão do problema, recomenda-se ao estudante seguir o modelo conceitual do caso a se resolver, pois, com isso, será possível enxergar a melhor solução e qual tipo de banco de dados é o mais indicado.

Saiba mais

Para mais informações sobre conceitos básicos de banco de dados e seus tipos, recomenda-se a leitura do primeiro capítulo dos livros [Banco de dados: projeto e implementação](#), de Felipe N. R. Machado, e [Banco de dados: teoria e desenvolvimento](#), de William P. Alves, disponíveis na Minha Biblioteca.

Para um maior aprofundamento no assunto, indicamos os artigos da revista [Database Trends and Applications](#), ISSN 1547-9897, disponível na ProQuest.

Referências

ALVES, W. P. **Banco de dados: teoria e desenvolvimento**. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

Aula 2

Sistemas Gerenciadores de Bancos de Dados (SGDB)

Sistema gerenciadores de bancos de dados (SGBD)

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula, iremos estudar os sistemas gerenciadores de banco de dados, mais conhecidos pela sigla SGBD, pois um banco de dados pode ser criado e mantido por um conjunto de aplicações desenvolvidas especialmente para esta tarefa ou por um tipo de SGBD. Os SGBDs, como veremos, surgiram para atender à necessidade de armazenamento e de recuperação de grandes volumes de informações, propiciando um ambiente seguro e adequado para empresas e organizações consultarem a qualquer momento seus dados. Com objetivo de aprofundar ainda mais o conhecimento em modelagem de dados, é necessário, pois, também entender por qual SGBD o banco de dados modelado será gerenciado. Bons estudos!

Vamos Começar!

Um sistema de gerenciamento de banco de dados (SGBD) é um software que executa os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações, incluindo módulos para consulta, atualização e interfaces entre o sistema e o usuário. De acordo com Elmasri e Navathe (2018), o objetivo geral de um banco de dados é centralizar as informações em um computador específico (servidor ou servidores), permitindo, assim, o compartilhamento de dados entre uma ampla variedade de sistemas.

Quando vários usuários acessam os dados, é possível que isso ocorra de maneira simultânea. Como exemplo, temos um banco de dados de eletrodomésticos indicando que o refrigerador mais recente de uma determinada marca está em estoque. Dois vendedores acessam a caixa registradora ao mesmo tempo e vendem uma geladeira aos seus clientes. O cliente definitivamente não teria mais geladeira, o que causaria muitos problemas tanto para ele quanto para a loja. Identificamos esse tipo de controle de simultaneidade de eventos como um dos objetivos fundamentais de um SGBD e, segundo Silberschatz, Korth e Sudarshan (2020), são técnicas utilizadas para garantir propriedades de isolamento de transações que são realizadas ao mesmo tempo.

Em resumo, há diversos programas executam essas garantias conhecidos como SGBD. Observe, na Figura 1, como é o esquema do SGBD em relação às aplicações e ao banco de dados.

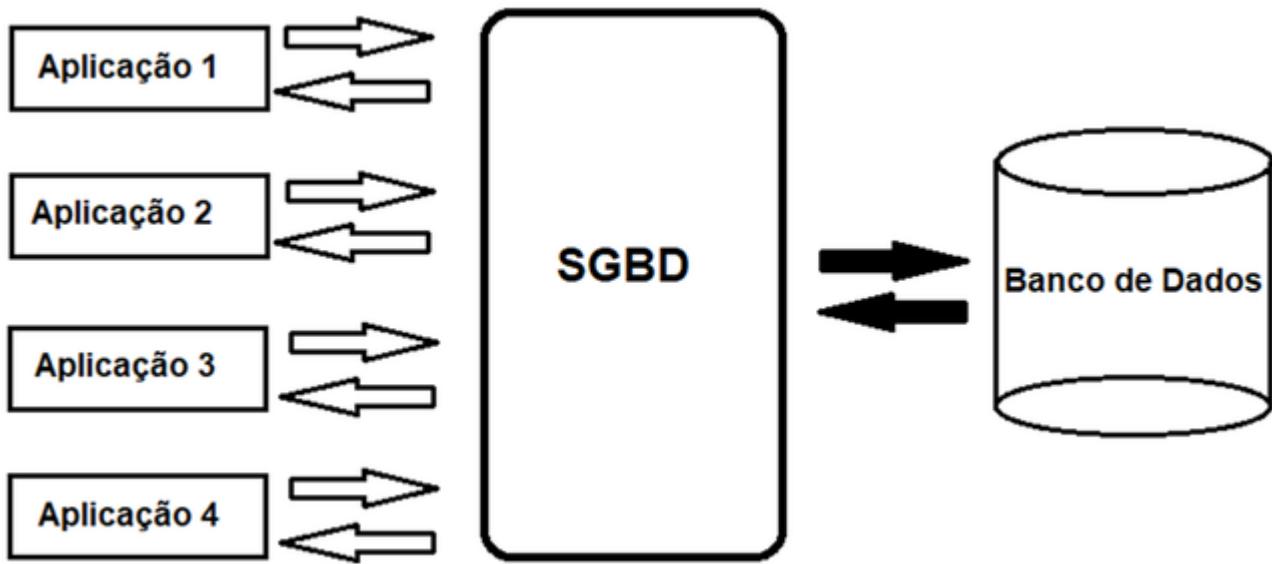


Figura 1 | SGBD em um banco de dados.

Logo, para Silberschatz, Korth e Sudarshan (2020), um sistema de gerenciamento de banco de dados consiste em um conjunto de dados que está vinculado a um conjunto de programas que pode acessá-los e, então, realizar várias operações básicas, tais como: inclusão, pesquisa, atualização, impressão e ordenação. Um SGBD é projetado para processar grandes quantidades de informações, até 1.152.921.504.606.846.976 bytes ou exabytes. O objetivo do SGBD é garantir a segurança das informações inseridas no banco de dados e protegê-las de acessos não autorizados ou de problemas causados por erros de software ou hardware. O SGBD pode ser implantado em vários computadores, no mesmo local ou até mesmo em locais diferentes (salas, cidades, países). Quando estão localizados em locais físicos diferentes, cada SGBD é chamado de nó, e as operações realizadas no banco de dados podem acontecer em um ou mais nós. Segundo Elmasri e Navathe (2018), os computadores e seus SGBD se comunicam utilizando diferentes protocolos de comunicação.

De acordo com Alves (2020), os SGBDs, para interagirem com os programas aplicativos dos usuários, oferecem dois recursos importantes, compostos por um conjunto de instruções da

linguagem SQL (*Structured Query Language* – Linguagem estruturada de consulta). O primeiro é conhecido como *Data Definition Language* – DDL (Linguagem de definição de dados), e compreende instruções que o usuário pode utilizar para criar ou alterar tabelas, definir *Stored Procedures* (procedimentos armazenados) e índices. Os comandos desse grupo normalmente não são executados pelos programas aplicativos, mas pelo administrador de banco de dados, ou DBA (*Database Administrator*). Já os comandos para manipulação/gerenciamento de dados (DML – *Data Manipulation Language*) podem ser executados interativamente, a partir de uma ferramenta de gerenciamento ou por um programa aplicativo. Esses comandos permitem que sejam extraídos dados da base, inseridos novos registros, excluídos registros antigos etc.

Elmasri e Navathe (2018) afirmam que além de um SGBD possuir as funções de permitir aos seus usuários a pesquisa em um banco de dados para recuperar uma determinada informação, alterar e gerar relatórios das informações, eles também têm outros papéis a destacar, como a proteção e a recuperação dos dados quando houver problemas de hardware ou software, a segurança a acessos indevidamente autorizados, a possibilidade de compartilhamento de dados, a administração da redundância e a restrição de integridade dos componentes do banco.

Siga em Frente...

O conjunto de requisitos de um SGBD recebe o nome de ACID, dos termos em inglês *Atomicity, Consistency, Isolation, Durability* ou, respectivamente, atomicidade, consistência, isolamento e durabilidade. O SGBD escolhido pela empresa deve possuir os fatores ACID para garantir que uma transação no banco de dados seja realizada com sucesso. Antes de analisarmos cada uma das quatro características de um SGBD, precisamos compreender o real significado de uma transação. Segundo Elmasri e Navathe (2018), uma transação é um processo ou um determinado programa que pode incluir vários bancos de dados ou somente uma parte deles, realizando atividades como consultas, alterações e até exclusão de informações da base de dados. Já para Silberschatz, Korth e Sudarshan (2020), é uma consequência da efetivação de um programa (ou uma rotina) que acessa e possivelmente atualiza vários itens de dados. A transação é o resultado da execução de um programa de usuário escrito em uma linguagem de manipulação de alto nível ou em uma linguagem de programação, como Java, C# ou SQL, entre outras.

Agora que sabemos o que é uma transação, podemos compreender os requisitos de um SGDB:

1. A **atomicidade** garante que nenhuma ou a totalidade das operações da transação sejam realizadas com sucesso. Suponha que estamos aumentando os salários dos funcionários (este aumento é uma alteração em uma tabela e, neste caso é uma transação) e que durante a atualização faltou luz. Somente uma parte dos funcionários receberá o aumento no salário, caso não haja a verificação de atomicidade. Conforme Silberschatz, Korth e Sudarshan (2020), a ideia por trás da garantia de atomicidade é que o sistema de banco de dados mantenha um registro (em disco) dos antigos valores de quaisquer dados a serem alterados. Caso aconteça algum problema durante a realização da transação, o SGBD reestabelece os dados antigos, como se nunca tivessem sido modificados.

2. A **consistência** preserva as regras impostas no banco de dados. Assim que a transação for finalizada, todos os dados devem estar íntegros. Um exemplo seria a soma de dois valores. Após uma transação, os valores iniciais não podem ser alterados, mas, claro, se esta for a regra determinada no banco de dados. A consistência é a garantia de manter a integridade dos dados durante e na finalização da transação realizada no banco de dados.
3. O **isolamento** é a segurança de que uma transação não interfira no trabalho de outra. Somente após o término de uma transação, ela estará liberada para receber outras. Silberschatz, Korth e Sudarshan (2020) afirmam que, mesmo asseguradas as propriedades de atomicidade e consistência para cada transação, a intercalação das operações de várias transações concorrentes pode resultar em inconsistências (erros nos resultados e/ou nos dados). Alterações feitas por transações simultâneas precisam ser isoladas das alterações feitas por qualquer outra transação simultânea.
4. A **durabilidade** é a certeza de que após uma transação ser realizada com sucesso, os resultados fiquem gravados no banco de dados, mesmo se algum problema ocorrer, como a queda do sistema. A durabilidade ou persistência (como também é conhecida) em um meio de armazenamento confiável e seguro é um dos requisitos mais importantes de um sistema gerenciador de banco de dados.

Atualmente, é difícil criar um projeto de banco de dados para uma única aplicação. Por mais que isso ocorra, cabe ao analista de sistema pensar e deixar o banco de dados modelado para futuras mudanças e adaptações. As principais características do uso de um banco de dados, conforme Elmasri e Navathe (2018), são as seguintes:

- Natureza auto descritiva do SGBD.
- Isolamento entre os programas, os dados e a abstração dos dados.
- Suporte a diversas visões dos dados inseridos no banco de dados.
- Transações para diversos usuários do banco e a possibilidade de compartilhar os dados da base de dados.

Uma característica essencial de um SGBD é possuir uma ampla gama de possibilidades para definir a estrutura da base de dados e poder aplicar restrições no banco. Os programas de aplicação que irão acessar a base de dados devem ser criados independentemente da estrutura do banco. De acordo com Elmasri e Navathe (2018), um SGBD oferece aos usuários uma representação conceitual de dados, omitindo vários detalhes, por exemplo, como os dados realmente são guardados ou como as transações são realizadas no banco de dados. Essa representação de modelo de dados é informalmente conhecida como abstração de dados.

Os SGBDs que têm sido mais utilizados recentemente são:

- SGBDs livres, como o MySQL, que tem o código fonte aberto e foi desenvolvido para ser uma opção aos SGBDs corporativos proprietários. Embora tenha crescido nos últimos anos e tenha ganhado grande popularidade em aplicativos voltados para a web, ainda não é amplamente utilizado por empresas de grande porte, que preferem confiar seus dados a aplicações mais "maduras" e com maior suporte. Existem outros SGBDs livres que seguem a linha do MySQL, como o PostgreSQL, por exemplo.

- Microsoft Access (relacionado ao pacote Microsoft Office), melhor para bancos de dados pessoais (uso doméstico) e menos robustos (pequenas aplicações de uso não crítico).
- Base (relacionado ao pacote BrOffice/LibreOffice), também mais voltado para uso doméstico.
- SGBDs comerciais e proprietários para uso corporativo, como o SQL Server e o Oracle, utilizados em projetos mais volumosos que envolvem bancos de dados corporativos (de grandes empresas). Outros SGBDs podem ser destacados, como: SyBase, Adabas, DB2 etc.

É comum profissionais da área fazerem referência aos SGBDs como banco de dados. Até certo ponto, o banco de dados é muito parecido a um tipo de arquivo eletrônico com conteúdo muito bem-organizado com a ajuda de um software, que é exatamente o sistema de gerenciamento de banco de dados. Porém, os SGBDs são softwares, já os bancos de dados conceitualmente não são considerados como tal. Isto é, os SGBDs sozinhos não possuem relevância alguma; os bancos de dados armazenados em um SGBD são quem verdadeiramente possui algum significado e são importantes para a organização que os mantém.

Vamos Exercitar?

Nesta aula, vimos alguns conceitos relacionados aos sistemas gerenciadores de banco de dados (SGBD) e que o melhor SGBD a ser usado para um determinado banco de dados é aquele que melhor atende às necessidades do cliente e/ou organização. É necessário levar em conta fatores como a infraestrutura, os recursos disponíveis e, principalmente, as reais necessidades do contexto. Para as organizações, o primeiro ponto que você deve considerar para implementação de um SGBD é o porte dela e suas movimentações e/ou transações de dados. Essa informação impacta diretamente os custos de software e hardware. Com os conhecimentos necessários, você será, então, capaz de realizar orientações e, até mesmo, manipular bem tanto um software quanto um hardware referente a bancos de dados.

Saiba mais

Neste artigo, você pode conhecer mais sobre o termo *Big Data* e compreender os novos desafios dos profissionais de informática com o grande volume de dados gerados e utilizados por grandes corporações e que deverão ser gerenciados por SGBDs.

SILVEIRA, M.; MARCOLIN, C. B.; FREITAS, H. M. R. [O big data e seu uso corporativo: uma revisão de literatura](#). In: SIMPÓSIO INTERNACIONAL DE GESTÃO DE PROJETOS, INOVAÇÃO E SUSTENTABILIDADE, 4., 2015, São Paulo. **Anais** [...]. São Paulo: SINGEP, 2015.

Referências

ALVES, W. P. **Banco de dados: teoria e desenvolvimento**. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

Aula 3

Dados como apoio a tomada de decisão

Dados como apoio à tomada de decisão

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Grandes organizações têm uma estrutura organizacional interna bastante complexa e, por essa razão, diferentes dados podem estar presentes em diferentes locais ou em diferentes sistemas operacionais, ou sob diferentes esquemas. Por exemplo, os dados de problema de manufatura e os dados de reclamação do cliente podem ser armazenados em diferentes sistemas de banco de dados. As organizações, com frequência, compram dados de fontes externas, como listas de discussão, usadas para promover produtos, ou pontuação de crédito fornecida por escritórios de crédito para decidir a capacidade de endividamento dos clientes. Os que tomam decisões corporativas exigem o acesso a informações de todas essas fontes. A preparação de consultas sobre fontes individuais é desajeitada e ineficaz. Ademais, as

fontes de dados só podem armazenar dados atuais, enquanto os que tomam decisões podem precisar de acesso a dados do passado; informações sobre a mudança nos padrões de compra nos últimos anos, por exemplo, poderiam ser de grande importância para uma organização. Depósitos de dados (*Data Warehouse*) oferecem uma solução para esses problemas. Assim, convidamos você, caro estudante, a entender melhor como os dados servem de apoio à tomada de decisão. Bons estudos!

Vamos Começar!

Segundo Silberschatz, Korth e Sudarshan (2020), um depósito de dados (*Data Warehouse*) é um repositório (ou arquivamento) de informações colhidas de várias origens, armazenadas sob um esquema unificado, em um único local. Uma vez reunidos, os dados são armazenados por muito tempo, permitindo o acesso a dados históricos. Assim, os depósitos de dados oferecem ao usuário uma única interface consolidada para os dados, facilitando a escrita de consultas de apoio à decisão. Além disso, acessando informações a partir de um depósito de dados, quem toma decisões pode garantir que os sistemas de processamento de transação on-line não serão afetados pela carga de trabalho de apoio à decisão.

A Figura 1 mostra a arquitetura de um depósito de dados típico e ilustra a coleta, o armazenamento e o suporte da consulta e análise de dados. Entre as questões a serem enfrentadas na montagem de um depósito de dados, estão as seguintes:

- **Quando e como coletar dados:** em uma arquitetura controlada pela fonte para a coleta de dados, as fontes de dados transmitem novas informações, seja continuamente (quando ocorre o processamento da transação) ou periodicamente (à noite, por exemplo). Já em uma arquitetura controlada por destino, o depósito de dados envia, periodicamente, solicitações para novos dados às fontes.

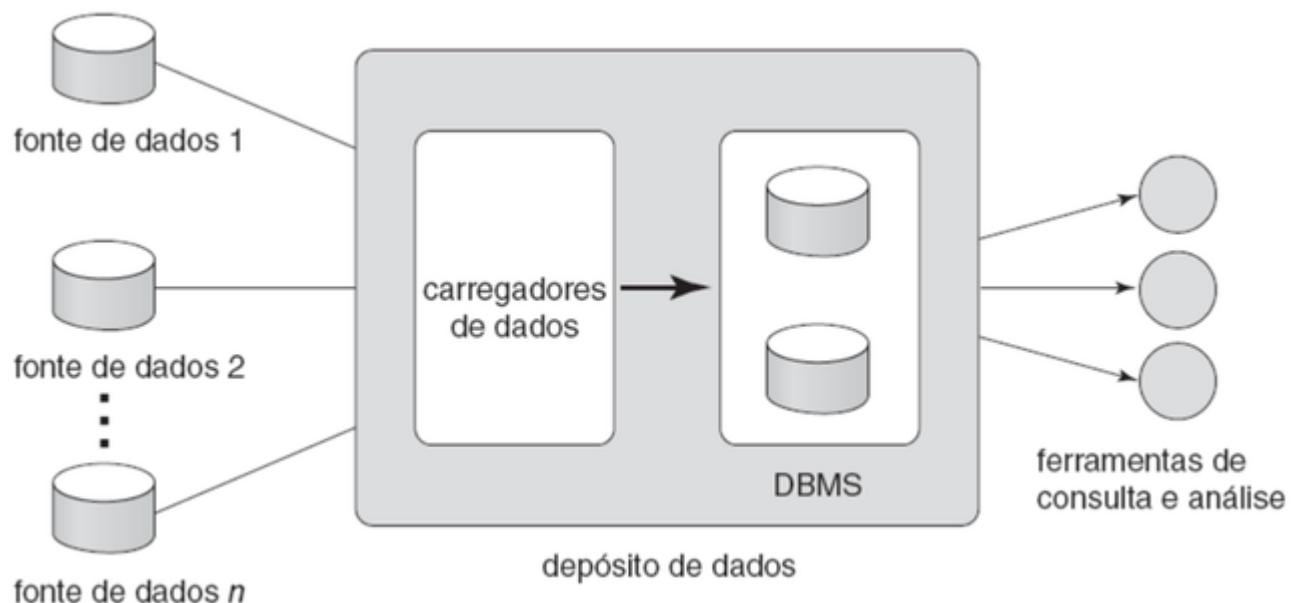


Figura 1 | Arquitetura de um Data Warehouse. Fonte: Silberschatz, Korth e Sudarshan (2020).

Ainda conforme Silberschatz, Korth e Sudarshan (2020), a menos que as atualizações nas fontes sejam replicadas no depósito de forma síncrona, este nunca estará muito atualizado. A replicação síncrona pode ser dispendiosa, de modo que os depósitos de dados normalmente não a utilizam, realizando consultas somente sobre dados que são antigos o bastante para terem sido completamente replicados. Tradicionalmente, os analistas se satisfaziam com os dados do dia anterior, de tal forma que os depósitos de dados poderiam ser carregados com dados até o final do dia anterior. Porém, cada vez mais organizações desejam dados mais atualizados.

Os requisitos de atualidade dos dados variam conforme a aplicação. Dados de até algumas horas atrás podem ser suficientes para algumas aplicações; outras, que exigem respostas em tempo real a determinados eventos, podem usar a infraestrutura de processamento *streaming* no lugar de depender de uma infraestrutura de depósito de dados.

- **Que esquema utilizar:** as fontes de dados que foram construídas de forma independente provavelmente terão diferentes esquemas. Na verdade, elas podem até mesmo usar diferentes modelos de dados. Parte da tarefa de um depósito é realizar a integração de esquema e converter dados para o esquema integrado antes que eles sejam armazenados. Como resultado, tem-se que os dados armazenados no depósito não são apenas uma cópia dos dados das fontes. Em vez disso, eles podem ser imaginados como uma visão materializada dos dados nas fontes.
- **Transformação e limpeza de dados:** a tarefa de corrigir e pré-processar dados é chamada de limpeza de dados. As fontes de dados normalmente entregam dados com diversas inconsistências menores, que podem ser corrigidas. Por exemplo, os nomes normalmente possuem erros de digitação, e os endereços podem ter erros em campos de rua/bairro/cidade ou códigos postais informados incorretamente. Eles podem ser corrigidos até certo ponto consultando-se um banco de dados de nomes de rua e códigos postais em cada cidade. A combinação aproximada de dados exigidos para essa tarefa é considerada como pesquisa difusa (*fuzzy lookup*).

As listas de endereços coletadas de várias fontes podem ter duplicatas que precisam ser eliminadas em uma operação *merge-purge* (operação também conhecida como eliminação de duplicidade). Os registros para os vários indivíduos em uma casa podem ser agrupados de modo que apenas uma correspondência seja enviada para a residência; essa operação é chamada de *householding*.

Os dados podem ser transformados de outras maneiras além da limpeza, como a mudança de unidades de medida ou a conversão para um esquema diferente, pela junção de dados de várias relações de origem. Vale dizer que os depósitos de dados normalmente possuem ferramentas gráficas para dar suporte à transformação. Essas ferramentas permitem que a transformação seja especificada como caixas, e que linhas possam ser criadas entre elas para indicar o fluxo dos dados. As caixas condicionais podem, assim, direcionar os dados para um próximo passo apropriado na transformação.

- **Como propagar atualizações:** as atualizações sobre as relações nas fontes de dados precisam ser propagadas para o depósito de dados. Se as relações no depósito de dados forem exatamente as mesmas daquelas na fonte de dados, a propagação será direta. Se não forem, o problema de propagação de atualizações é, basicamente, o problema de manutenção de visão (*view-maintenance*).
- **Quais dados resumir:** os dados brutos gerados por um sistema de processamento de transação podem ser muito grandes para serem armazenados on-line. Porém, podemos responder a muitas consultas mantendo apenas dados de resumo obtidos pela agregação sobre uma relação, em vez de manter a relação inteira. Por exemplo, em lugar de armazenar dados sobre cada venda de roupas possível, podemos armazenar o total de vendas de roupas por nome e categoria do item.

As diferentes etapas envolvidas na obtenção de dados para um depósito de dados são chamadas de tarefas de extração, transformação e carga (ETL); a extração refere-se à obtenção de dados das fontes, enquanto a carga diz respeito à carga dos dados no depósito de dados. Nos depósitos de dados de última geração, que possuem suporte para funções definidas pelo usuário ou *frameworks* MapReduce, os dados podem ser extraídos, carregados para o depósito e, depois, transformados. A abordagem ELT permite, pois, o uso de *frameworks* de processamento paralelo para a transformação de dados.

Para Alves (2020), quando se fala em *Data Warehouse*, é comum ouvirmos expressões como OLAP, OLTP e *Data Mining*. OLAP é a sigla em inglês para *On-Line Analytical Processing* (Processamento Analítico On-Line) e significa que as informações são processadas para uma análise complexa. Já o OLTP - *On-Line Transaction Processing* (Processamento de Transação On-Line) refere-se aos sistemas com os quais trabalhamos normalmente, ou seja, qualquer operação (inserção, alteração ou exclusão) executada de imediato no banco de dados utilizando-se transações.

De acordo com Silberschatz, Korth e Sudarshan (2020), o termo mineração de dados (ou *Data Mining*) relaciona-se, em geral, ao processo de analisar grandes bancos de dados para encontrar padrões úteis. Assim como a descoberta de conhecimento na inteligência artificial (também chamada aprendizado de máquina) ou na análise estatística, a mineração de dados tenta descobrir regras e padrões a partir dos dados. Porém, esta difere daquelas porque lida com grande volume de dados, armazenados principalmente em disco. Ou seja, a mineração de dados ocupa-se da descoberta de conhecimento nos bancos de dados (KDD).

Siga em Frente...

Alguns tipos de conhecimento descobertos em um banco de dados podem ser representados por um conjunto de regras. Eis um exemplo de uma regra citada informalmente: “mulheres jovens com renda anual maior que \$ 50.000 são as pessoas mais prováveis de comprar carros esportivos”. É claro que essas regras não são universalmente verdadeiras e possuem graus de “suporte” e “confiança”, conforme veremos. Outros tipos de conhecimento são representados por equações relacionando diferentes variáveis entre si. De um modo geral, o conhecimento

descoberto pela aplicação de técnicas de aprendizado de máquina em instâncias do passado em um banco de dados é representado por um modelo que é, então, usado para prever resultados para novas instâncias. Características ou atributos das instâncias são entradas para o modelo, e a saída de um modelo é uma previsão.

Existem diversos padrões possíveis que podem ser úteis, e diferentes técnicas são utilizadas para encontrá-los. Estudaremos, a seguir, alguns exemplos de padrões e veremos como eles podem ser derivados automaticamente de um banco de dados.

Normalmente, existe um componente manual para a mineração de dados, consistindo no pré-processamento dos dados para um formato aceitável para os algoritmos e no pós-processamento de padrões descobertos para encontrar outros que poderiam ser úteis. Também pode haver mais de um tipo de padrão que pode ser recuperado a partir de determinado banco de dados, e a interação manual pode ser necessária para selecionar aqueles aproveitáveis. Por esse motivo, a mineração de dados é, na realidade, um processo semiautomático na vida real. Porém, em nossa descrição, nos concentraremos em seu aspecto automático.

Segundo Alves (2020), os sistemas OLAP são projetados para atender às consultas que surgem em função das necessidades dos usuários no momento. Nessa tecnologia, os dados brutos são transformados em informações consistentes para tornar fácil sua compreensão por parte do usuário. Os depósitos de dados (*Data Warehouses*), assim, normalmente possuem esquemas que são projetados para análise de dados usando ferramentas do tipo OLAP. As relações em um esquema de depósito de dados, em geral, podem ser classificadas como tabelas de fatos e tabelas de dimensão. Tabelas de fatos registram informações sobre eventos individuais e, em geral, são muito grandes. Uma tabela registrando informações de vendas para um comércio varejista, com uma tupla para cada item vendido, é um exemplo típico de tabela de fatos. Os atributos na tabela de fatos podem ser classificados como atributos de dimensão ou atributos de medição. Os atributos de medição armazenam informação quantitativa, que pode ser agregada; em uma tabela vendas, eles incluiriam o número de itens vendidos e o preço dos itens. Já os atributos de dimensão são dimensões sob as quais os atributos de medição, e os resumos destes, são agrupados e visualizados. Em uma tabela vendas, eles incluiriam um identificador de item, a data em que ele foi vendido, em que local (loja) foi comprado, qual cliente o comprou, e assim por diante.

No sistema OLTP, os dados são acumulados a partir de transações diárias da empresa. São dados que se encontram em seu estado “puro”, sem o devido tratamento para análise. Somente consultas preestabelecidas são possíveis nesse sistema. Desta forma, ele é definido como a fonte de dados para o *Data Warehouse*. Existe também outro termo com o qual nos deparamos frequentemente nesse contexto, o ODS – *Operational Data Store* (Depósito de Dados Operacional), que se refere a uma espécie de repositório de dados, similar a um *Data Warehouse*, mas que não coloca à disposição as informações para uma tomada de decisão.

Imagine um laboratório farmacêutico cujo gerente de vendas precisa ter em mãos informações referentes aos produtos/medicamentos que mais têm saída em uma determinada época do ano (inverno, por exemplo). Com base nessas informações, ele deve decidir em qual linha vai atuar mais. Ele também precisa passar essas informações ao gerente de produção para

que ele tome as providências necessárias para produzir em maior quantidade os produtos/medicamentos adequados; caso contrário, o fornecimento será prejudicado.

A área responsável pelo transporte também deve ter conhecimento desse aumento na produção/venda para poder administrar os processos de entrega aos clientes (se for necessário, contratar mais transportadoras). Podemos, desta forma, perceber a importância de uma informação de boa qualidade. As informações devem ter um grau de precisão alto, pois podem interferir não apenas em um, mas em vários processos de gestão ou setores de uma empresa.

Em resumo, a principal característica dos *Data Warehouses* é que eles são verdadeiros depósitos de dados integrados originados de várias fontes. Formam, assim, um modelo de dados multidimensional. Esse modelo é bem adequado às tecnologias disponíveis para suporte à tomada de decisão. Os dados que podem ser modelados usando atributos de dimensão e atributos de medida são chamados de dados multidimensionais.

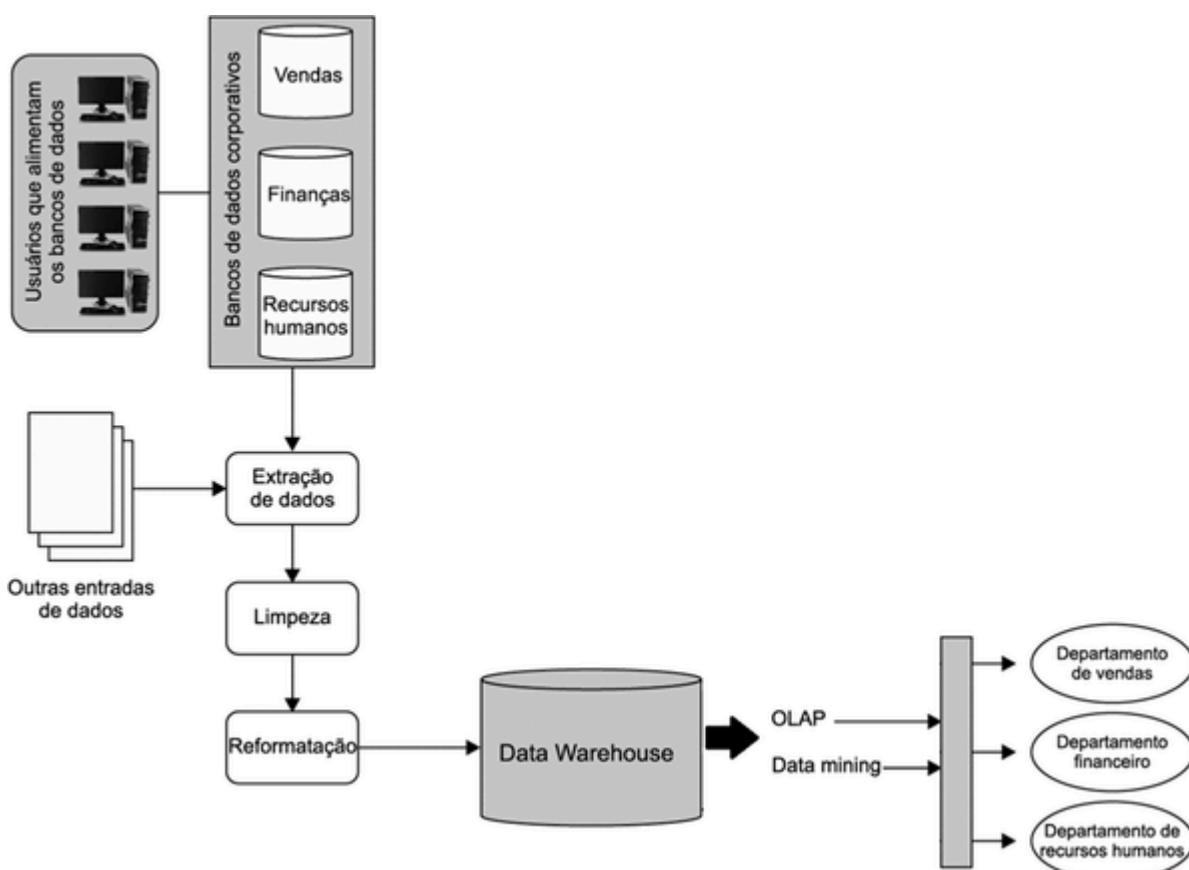


Figura 2 | Processos básicos envolvidos em um Data Warehouse. Fonte: adaptada de Alves (2020).

Vamos Exercitar?

Nesta aula, estudamos alguns conceitos relacionados ao *Data Warehouse*, que em grande parte faz uso de bancos de dados distribuídos. Outros assuntos abordados foram o OLAP, OLTP e *Data*

mining, todos eles fatores que devem ser considerados na implantação de um *Data Warehouse*. Vimos que ferramentas de processamento analítico on-line (OLAP) ajudam os analistas a verem dados resumidos de diferentes maneiras, de modo que possam ter ideia do funcionamento de uma organização; aprendemos, também, que a mineração de dados é o processo de analisar, de maneira semiautomática, grandes bancos de dados para encontrar padrões úteis. Por fim, concluímos que existem diversas aplicações da mineração de dados, como a previsão de valores com base em exemplos passados, a descoberta de associações entre compras e agrupamento automático de pessoas e filmes.

Saiba mais

Para mais informações sobre os conceitos básicos de banco de dados e seus tipos, recomendamos a leitura do capítulo 11 do livro [*Sistema de banco de dados*](#), de Elmasri e Navathe, e do capítulo 13 do livro [*Banco de dados: teoria e desenvolvimento*](#), de William Alves, disponíveis na Minha Biblioteca.

Para um maior aprofundamento no assunto, indicamos os artigos da revista [*Database Trends and Applications*](#), ISSN 1547-9897, disponível na ProQuest.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

Aula 4

Tópicos relevantes sobre banco de dados

Tópicos relevantes sobre banco de dados

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Em qualquer segmento de atuação, a maioria das empresas enfrenta uma transformação digital mais profunda. Isso quer dizer que seus gestores passaram a ter que lidar com novas demandas, com destaque para a coleta, tratamento e armazenamento de dados. Nesse sentido, a redundância de banco de dados se torna fundamental.

Devido ao aumento gradativo do volume de informações gerenciadas, as empresas tiveram que buscar soluções de recuperação para eventos programados ou inesperados, como ataques cibernéticos. Assim, conhecer e implantar sistemas redundantes efetivos se tornou prioridade, bem como políticas de autenticação e acesso a banco de dados ou *Data Warehouses* de forma mais restritiva se transformaram em um padrão comum nas organizações, principalmente em grandes Fintechs que lidam com dados sensíveis de clientes e fornecedores. De maneira geral, a redundância em banco de dados e a adoção de protocolos de segurança da informação garante a continuidade operacional de qualquer tipo de negócio. A LGPD (Lei Geral de Proteção de Dados Pessoais) aumentou ainda mais a responsabilidade das companhias pelos dados que estão sob seu controle. Assim, convidamos você, caro estudante, a compreender melhor como os dados servem como apoio à tomada de decisão. Bons estudos!

Vamos Começar!

A redundância de dados é a criação de cópias adicionais de informações, armazenando-as em bancos de dados diferentes ou usando mecanismos diferentes. O objetivo desta prática de backup redundante é garantir a preservação e recuperação de dados em caso de falhas, erros ou incidentes como defeitos de hardware, ataques cibernéticos ou desastres naturais.

Mas o que o torna importante? Em primeiro lugar, a disponibilidade contínua de dados, que permite, por exemplo, às instituições financeiras garantirem o acesso constante a informações críticas mesmo em situações adversas.

Se uma fonte de dados ficar inacessível ou falhar, cópias redundantes podem ser usadas para manter as operações em dia. Elas também aceleram o processo de recuperação em situações de falha ou desastre. Ao implantar cópias atualizadas em locais distintos, as instituições financeiras, por exemplo, podem restaurar rapidamente os sistemas e minimizar o impacto das interrupções, assegurando a continuidade das operações e a satisfação do cliente. Outro fator a ser destacado é que essas instituições estão mais bem preparadas para lidar com perdas accidentais de informações. Se um conjunto de dados for corrompido ou excluído, cópias redundantes poderão ser usadas para recuperar informações perdidas para evitar perdas financeiras e manter a confiança do cliente. Além disso, existem regulamentações rigorosas neste setor que exigem uma proteção de dados adequada.

Em resumo, a redundância de dados tem o papel de auxiliar as instituições a cumprirem esses padrões, fornecendo uma camada adicional de segurança e garantindo a integridade e confidencialidade das informações dos clientes. Para que tudo funcione de forma harmoniosa, então, o banco de dados também deve passar por padrões e métodos de política de segurança de acesso de usuários.

Para Alves (2020), a segurança de um sistema de banco de dados está relacionada diretamente com sua integridade e com a proteção das informações nele armazenadas. Ao se trabalhar com segurança, é importante levar em consideração algumas questões, sendo as principais:

- O direito (ou não) de acesso a determinadas informações tidas como confidenciais ou sigilosas, como salários de funcionários, avaliações de desempenho ou gratificações que não podem ser acessadas por pessoas não autorizadas.
- O nível em que a segurança deve trabalhar, ou seja, as funções de segurança que devem ser tratadas no nível físico, no nível operacional ou no nível do sistema de gerenciamento do banco de dados.

Em aplicações do tipo monousuário, como um aplicativo para controle de orçamento doméstico (contas a pagar e a receber), a segurança não precisa necessariamente ser um fator crítico e merecedor de muita atenção. Isso, todavia, não ocorre quando se trata de sistemas multiusuários de uma empresa. Nesse caso, é imprescindível haver técnicas que controlem o acesso por parte de grupos de usuários, fornecendo-se permissão apenas a seções específicas do banco de dados.

Atualmente, todos os SGBDs relacionais possuem um subsistema de controle de acesso que é responsável pelo gerenciamento de usuários, pela definição de níveis de acesso e pela seleção das operações que podem ser executadas pelos usuários, bem como pelas informações que podem ser acessadas. Já vimos, superficialmente, como esse recurso trabalha.

Outra importante função desse sistema é a criptografia dos dados armazenados no banco, cujo objetivo é protegê-los (por serem sigilosos, como números de cartões de crédito).

Cabe ao administrador do banco de dados (DBA) a responsabilidade de definir privilégios de acesso aos usuários do sistema de acordo com as políticas de segurança adotadas pela empresa/organização. Ele possui uma conta de usuário especial, normalmente denominada

conta de superusuário ou conta do administrador. Os bancos de dados relacionais, assim, permitem o gerenciamento de privilégios dos usuários em dois níveis:

- **Nível de conta de usuário:** cada conta ou usuário individual possui um tipo de privilégio específico, independentemente das relações/tabelas existentes no banco de dados.
- **Nível de relação/tabela:** é possível definir para cada tabela do banco de dados um privilégio específico para acesso e manipulação dos dados.

Em sistemas padrão SQL, a segurança é baseada no conceito de direitos ou privilégios, por meio dos quais os usuários têm ou não permissão para executar determinadas operações.

Atualmente, o padrão ANSI/ISO define quatro privilégios: **SELECT** (consulta/extracção de dados), **INSERT** (inclusão de novos registros), **UPDATE** (atualização de registros) e **DELETE** (exclusão de registros). Se um usuário com privilégio somente de consulta (SELECT) tentar incluir um registro utilizando o comando “**INSERT INTO**”, o servidor SQL retornará uma mensagem de erro.

Conforme vimos anteriormente, em SQL existem dois comandos para o gerenciamento de privilégios de usuários: **GRANT**, para atribuir a um usuário um determinado privilégio, e **REVOKE**, para revogar (remover) um privilégio anteriormente estabelecido. Para que possamos conceder privilégios a um usuário, é preciso que ele seja previamente criado. Isso pode ser feito com o comando **CREATE USER** pelo administrador de banco de dados.

Suponhamos que exista um banco de dados com uma tabela de cadastro de clientes denominada **Clientes**. Para atribuir privilégios de inclusão e exclusão ao usuário identificado como **user001**, teríamos de executar o seguinte comando SQL:

```
GRANT INSERT, DELETE ON CLIENTES TO user001;
```

Se posteriormente fosse necessário revogar o privilégio de exclusão, usaríamos o comando:

```
REVOKE DELETE ON CLIENTES FROM user001;
```

Siga em Frente...

Os sistemas SQL atuais possuem ambientes gráficos que tornam mais fácil e intuitiva a tarefa de gerenciar contas, usuários e privilégios, uma vez que tudo é executado de forma visual, sem a necessidade de digitação de comandos.

Como lembram Silberschatz, Korth e Sudarshan (2020), é possível traçar uma trilha de auditoria que, por sua vez, é um registro (ou *log*) de todas as mudanças (inserções, exclusões e atualizações) nos dados da aplicação de banco de dados, juntamente com informações como qual usuário realizou a mudança e quando a mudança foi realizada. Se a segurança da aplicação for comprometida, ou mesmo se isso não acontecer, mas alguma atualização for executada

erroneamente, uma trilha de auditoria pode ajudar a descobrir o que aconteceu e quem pode ter executado as ações e a reparar o dano causado pelo furo de segurança ou atualização indevida.

Por exemplo, se descobrimos que a nota de um aluno está incorreta, o registro de auditoria pode ser examinado para localizar quando e como a nota foi atualizada, além de identificar qual usuário executou a atualização. A universidade pode, então, usar também a trilha de auditoria para rastrear todas as atualizações realizadas por esse usuário, a fim de descobrir outras alterações incorretas ou fraudulentas e depois corrigi-las.

As trilhas de auditoria também podem ser usadas para detectar furos de segurança em que a conta de um usuário é comprometida e acessada por um intruso. Por exemplo, toda vez que um usuário efetua o login, ele pode ser informado sobre as atualizações na trilha de auditoria que foram feitas a partir desse acesso no passado recente; se o usuário encontrar uma atualização que não executou, provavelmente a conta foi comprometida.

Ainda de acordo com Silberschatz, Korth e Sudarshan (2020), é possível criar uma trilha de auditoria em nível de banco de dados definindo *triggers* apropriadas nas atualizações da relação (usando variáveis definidas pelo sistema que identifiquem o nome do usuário e a hora). Porém, muitos sistemas de banco de dados oferecem mecanismos internos para criá-las que são muito mais convenientes de se usar. Os detalhes de como essas trilhas variam entre os sistemas de banco de dados, e você deverá consultar os manuais de cada sistema para obter tais informações.

Em geral, as trilhas de auditoria em nível de banco de dados são insuficientes para aplicações, pois normalmente não conseguem rastrear quem foi o usuário final da aplicação. Além disso, as atualizações são registradas em nível de linha, em termos de atualizações a tuplas de uma relação, em vez de em um nível mais alto, em termos da lógica do negócio. As aplicações, portanto, comumente criam uma trilha de auditoria de nível mais alto, registrando, por exemplo, que ação foi executada, por quem, quando e de qual endereço IP a solicitação foi originada.

Uma questão relacionada é a da proteção da própria trilha de auditoria contra modificação ou exclusão por usuários que quebram a segurança da aplicação. Uma solução possível consiste em copiar a trilha de auditoria para uma máquina diferente, à qual o intruso não teria acesso, com cada registro sendo copiado assim que fosse gerado. Uma solução mais forte é usar técnicas de *blockchain*, que armazenam logs em diversas máquinas e usam um mecanismo de *hashing* e tornam muito difícil para um intruso modificar ou excluir dados sem que seja detectado.

Para Alves (2020) e Silberschatz, Korth e Sudarshan (2020), o consenso sobre privacidade de dados armazenados em aplicações de banco de dados ou *Data Warehouses* está na ideia de que em um mundo onde uma quantidade cada vez maior de dados pessoais está disponível on-line, as pessoas estão mais preocupadas com a privacidade. Por exemplo, a maioria deseja que seus dados médicos pessoais fossem mantidos em segredo, e não revelados publicamente. Porém, esses dados precisam estar disponíveis para médicos e técnicos de emergência que tratam do paciente. Muitos países possuem leis sobre a privacidade desses dados que definem quando e para quem eles podem ser revelados, e sua violação pode resultar em penalidades

criminais. As aplicações que acessam esses dados privados, portanto, precisam ser criadas com cuidado, considerando as leis de privacidade.

No entanto, os dados privados agregados podem desempenhar um papel importante em muitas tarefas, como a detecção de efeitos colaterais de drogas ou de epidemias. Como tornar esses dados disponíveis aos pesquisadores que executam tais tarefas, sem comprometer a privacidade dos indivíduos, é um problema importante no mundo real. Como exemplo, suponhamos que um hospital oculte o nome do paciente, mas ofereça a um pesquisador a data de nascimento e o CEP (código postal) dele. Em muitos casos, apenas essas duas informações podem ser usadas para identificar com exclusividade o paciente (usando informações de um banco de dados externo), comprometendo, deste modo, sua privacidade. Nessa situação em particular, uma solução seria dar o ano do nascimento, mas não a data exata, junto com o código postal, o que pode ser suficiente para o pesquisador, mas não para identificar a maioria dos indivíduos de forma exclusiva.

Os sites, vale destacar, normalmente também coletam dados pessoais como endereço, telefone, e-mail e informações de cartão de crédito. Esses dados podem ser necessários para a realização de uma transação, como a compra de um item de uma loja, porém o cliente pode não querer que eles se tornem disponíveis para outras organizações ou desejar que parte da informação (como os números de cartão de crédito) seja apagada após algum período de tempo, como meio de impedir que caia em mãos não autorizadas, no caso de uma falha na segurança. Muitos sites permitem que os clientes especifiquem suas preferências de privacidade e, portanto, precisam garantir que essas preferências sejam respeitadas.

Vamos Exercitar?

Nesta aula, compreendemos alguns conceitos relacionados à segurança em bancos de dados. Vimos que os mecanismos de autorização do SQL são menos minuciosos e possuem valor limitado para aplicações que tratam de uma grande quantidade de usuários. Hoje, os programas de aplicação implementam uma autorização minuciosa, em nível de tupla, lidando com muitos usuários da aplicação, completamente fora do sistema de banco de dados. Estão sendo desenvolvidas, assim, extensões de banco de dados para fornecer controle de acesso em nível de tupla e para lidar com a grande quantidade de usuários da aplicação, mas elas ainda não estão padronizadas. A proteção da privacidade dos dados é uma tarefa importante para as aplicações de banco de dados. Muitos países possuem requisitos legais sobre a proteção de certos tipos de dados, como informações de cartão de crédito ou registros médicos. É importante, de fato, ter uma política de proteção e preservação dos dados contra acessos indevidos ou corrupção com risco de perda. Para isso, é preciso que o administrador dos dados atribua permissões de acesso a usuários e saiba utilizar o firewall e outras ferramentas como forma de proteger o sistema (e toda a infraestrutura da empresa) contra acesso externo não autorizado.

Saiba mais

MODELAGEM DE DADOS

Para mais informações sobre conceitos básicos de banco de dados e seus tipos, recomendamos a leitura do capítulo 9 do livro [*Sistema de banco de dados*](#), de Elmasri e Navathe, e do capítulo 15 do livro [*Banco de dados: teoria e desenvolvimento*](#), de William Alves, disponíveis na Minha Biblioteca.

Para um maior aprofundamento no assunto, indicamos os artigos da [*Journal of Database Management*](#), ISSN 1063-8016, disponível na ProQuest.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Chegada

Olá, estudante! Para desenvolver a competência desta unidade, isto é, compreender a importância dos bancos de dados e dos SGBDs, bem como de toda a parte de segurança envolvida, é necessário que você realize a leitura de vários conteúdos relacionados ao tema em paralelo a este material. Estudamos somente a parte introdutória do mundo dos banco de dados, passo este que é fundamental no decorrer da disciplina de Modelagem de Dados. Para um maior aprofundamento neste tema, você deverá, primeiramente, aprender algumas nomenclaturas e saber diferenciá-las, como ACID, OLAP, OLTP, SQL etc. É necessário, também, que você entenda bem sobre o contexto em que está inserido para, posteriormente, saber identificar oportunidades de melhorias em banco de dados, seja na parte fundamental ou relacionadas à segurança dos dados.

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dona Amélia, proprietária de uma empresa de bolos, doces e salgados, está com problemas no gerenciamento de seu negócio e quer informatizar sua rotina de trabalho, porém está receosa quanto ao valor de investimento. No último final de ano, ela deixou de atender vários clientes, o que causou prejuízos. A empresa precisa sair do processo manual e informatizar tudo, da encomenda até a entrega e cobrança. Como Dona Amélia não está mais conseguindo gerenciar as encomendas e precisa de um software, procurou a nossa empresa pedindo ajuda e conselhos e querendo saber no que e quanto terá que investir. De uma coisa ela tem certeza: quer que todas as informações fiquem na empresa e de forma bem segura.

Você, como analista de sistemas, deverá auxiliar a Dona Amélia. Após ter aprendido um pouco sobre os SGBDs, qual seria o mais indicado para a empresa dessa cliente? Como poderemos determinar o tipo de SGBD ideal para uma pequena, média ou grande base de dados? Formule algumas perguntas para compreender melhor o contexto e poder sugerir a melhor solução neste caso.

Verifique os conceitos, caso haja dúvidas. Após estudar sobre os SGBDs, você entenderá que, como analistas, devemos ser cautelosos ao indicar um sistema desnecessário às necessidades do cliente.

Faça algumas reflexões:

- É comum pensar e até mesmo associar banco de dados a um software. Reflita por que isso acontece.
- Quais são as vantagens de se realizar a redundância em banco de dados?
- Qual é a diferença entre banco de dados relacional e *Data Warehouse*?

Para a resolução do problema da empresa de Dona Amélia, precisamos verificar alguns pontos que devem ser levados em consideração para a análise de suas necessidades:

1. Qual é a quantidade de clientes da empresa? É uma empresa pequena, portanto automaticamente descartamos o SGBD Oracle. A versão SQL Server somente é viável se a opção for *freeware*, mas essa decisão depende diretamente da ferramenta de desenvolvimento.
2. O sistema é todo manual, então haverá a necessidade de investimento em infraestrutura com a compra de equipamentos eletrônicos. Além disso, uma máquina deverá ser configurada para o servidor. Lembre-se de que Dona Amélia deseja que as informações fiquem seguras na própria empresa.
3. Como a empresa é de encomendas de bolos e salgados e tudo é feito de forma manual, por que não indicar um sistema integrado à internet? Nesta situação, o MySQL pode perfeitamente ser utilizado.
4. Outra opção seria um serviço em nuvem. Neste caso, teríamos que demonstrar as vantagens e desvantagens deste tipo de serviço, como preço *versus* segurança.

Com os conhecimentos adquiridos nesta unidade, você deverá escolher e indicar o melhor SGBD em termos de performance e segurança. Existem outras opções disponíveis no mercado, mas o fundamental é ter o conhecimento para não sugerir algo que inviabilize a realização do software que o cliente deseja. Conquistar o cliente é uma das características desejadas de um analista de sistemas.

Caro estudante, a seguir apresentamos um breve infográfico contendo um resumo dos principais temas abordados nesta unidade de forma simples e objetiva. Reserve um tempo para leitura desse material e reflita.

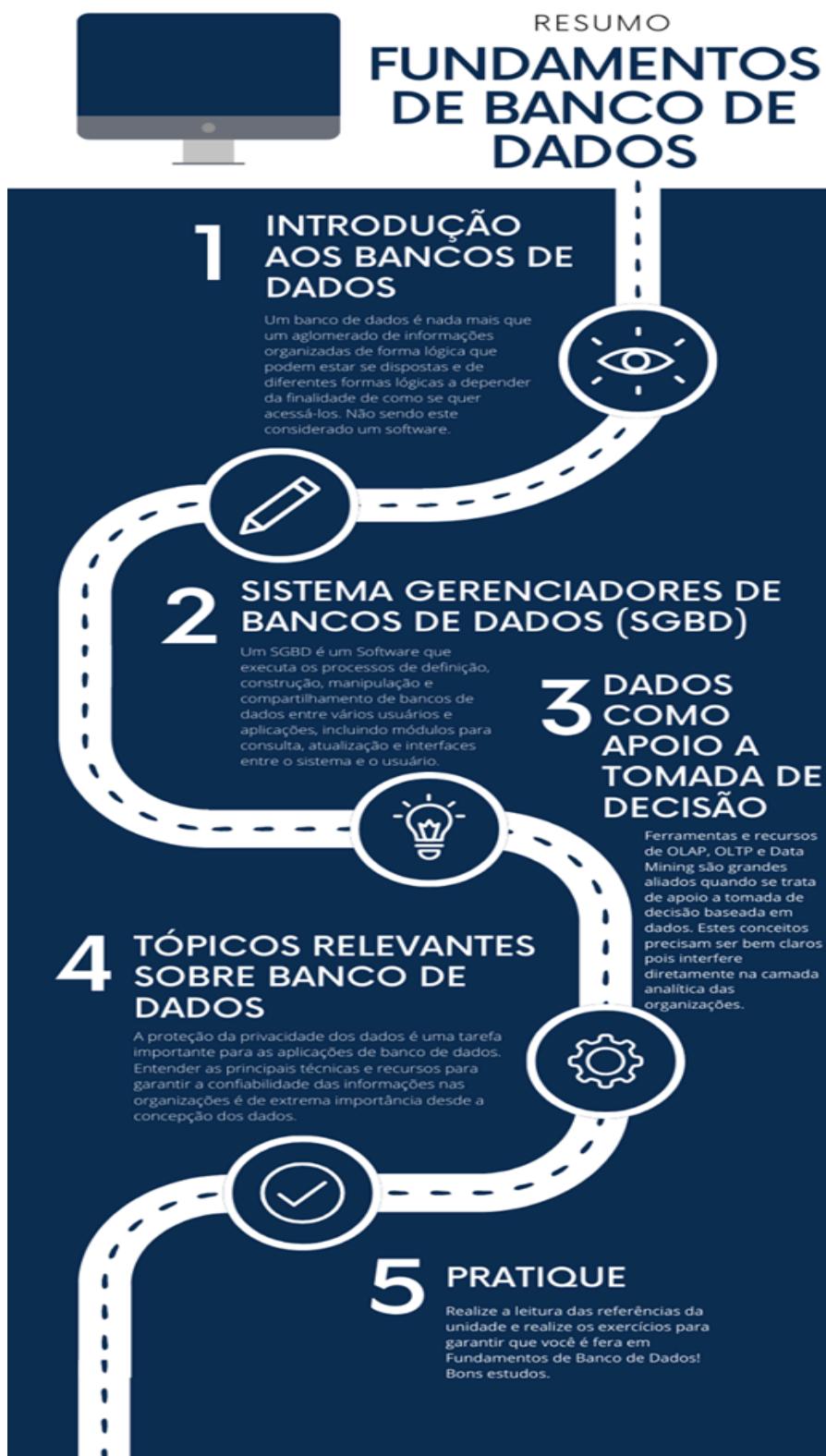


Figura | Fundamentos de banco de dados.

ALVES, W. P. **Banco de dados: teoria e desenvolvimento.** 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

Unidade 2

Modelos de Dados

Aula 1

Modelagem de dados com o modelo Entidade-Relacionamento (ER)

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula, estudaremos conceitos muito importantes em modelagem de dados. A modelagem de dados é uma metodologia utilizada para especificar os requisitos que um determinado software deverá possuir. Ela faz parte, portanto, do processo desenvolvimento de um software. Modelar dados envolve uma série de aplicações teóricas e práticas visando construir um esquema banco de dados que possa ser utilizado em qualquer sistema gerenciador

de banco de dados (SGBD). Para tanto, nesta unidade você aprenderá as diferenças entre os modelos conceitual, lógico e físico, além de conhecer com detalhes os elementos usados na construção da modelagem como o modelo ER (entidade-relacionamento). Fique atento aos conceitos que serão apresentados, pois eles serão essenciais para a compreensão da disciplina. Assim, convidamos você, caro estudante, a prestar muita atenção nesta parte inicial; você perceberá que precisará dela para se aprofundar mais no tema. Bons estudos!

Vamos Começar!

De acordo com Sordi (2019), a modelagem de dados é um processo que se repetirá de forma gradual, começando com a compreensão inicial de um problema. À medida que este problema for mais bem compreendido, o nível de detalhes do modelo também aumentará. Uma das etapas mais importantes no desenvolvimento de um software, assim, é a elaboração de um projeto de banco de dados.

Criar um banco de dados sem um estudo e de forma apressada pode causar diversos problemas, como um desempenho abaixo do esperado, consultas difíceis e, ainda, resultados equivocados, o que afeta negativamente a performance do software. Um modelo de dados é um detalhamento dos tipos de dados que serão armazenados em um banco de dados. Em sua futura elaboração, as linguagens utilizadas serão qualificadas, de acordo com a forma de apresentar modelos, em textual ou gráfica. Esta forma de representação de um modelo de dados, por meio de uma linguagem de modelagem de dados, é conhecida como esquema de banco de dados, segundo Silberschatz, Korth e Sudarshan (2020).

Quando pretendemos fazer uma viagem, a primeira preocupação é planejá-la cuidadosamente e escolher o que vamos levar nas malas, o roteiro a ser feito e o meio de transporte a ser utilizado (carro, avião, ônibus etc.). Da mesma forma, quando estamos empenhados no desenvolvimento de um sistema para computador (seja de grande ou pequeno porte), devemos planejar todas as suas etapas e dedicar atenção especial ao projeto e estruturação do banco de dados. Para isso, utilizamos uma técnica chamada modelagem de dados, cujo objetivo é transformar uma ideia conceitual em algo que possa ser traduzido em termos computacionais. Para Alves (2020), com a modelagem de dados podemos refinar um modelo conceitual durante as fases que compõem o projeto, eliminando redundâncias ou incoerências que possam inevitavelmente surgir.

Sem esse planejamento prévio, a manutenção do sistema certamente se tornará uma tarefa mais complexa e frequente. Não podemos, no entanto, acreditar que todo o projeto de banco de dados deva ser realizado apenas pela equipe de TI. Os próprios utilizadores devem também participar de sua elaboração nas fases mais críticas, como recolha de dados, testes de usabilidade e validação. Não há dúvida de que até os profissionais precisam saber como funciona um negócio, caso contrário, as consequências podem ser desastrosas.

De acordo com Alves (2020), durante a fase inicial de modelagem conceitual dos dados, o profissional precisa observar atentamente tudo o que for relevante no mundo real e que deva ser “transportado” para o sistema que está sendo projetado. Com essas informações, ele já pode

criar um esboço, representando de forma gráfica o processo. A esse esboço damos o nome de abstração ou modelo abstrato. Nele, podemos encontrar três componentes muito importantes: o modelo conceitual, o modelo lógico e o modelo físico.

Siga em Frente...

O **modelo conceitual** é a primeira etapa do projeto, na qual se representa a realidade mediante uma visão global e genérica dos dados e de seus relacionamentos. Seu objetivo é conter todas as informações dessa realidade que serão armazenadas, sem que se retratem aspectos relativos ao banco de dados que será utilizado. Para Alves (2020), essas informações podem aparecer no formato de uma lista descritiva das operações executadas pelos usuários e dos dados que eles devem manipular.

O **modelo lógico** é a segunda etapa e compreende a descrição das estruturas que serão armazenadas no banco de dados. Resulta em uma representação gráfica dos dados de maneira lógica, inclusive já nomeando os componentes e as ações que exercem um sobre o outro, como vemos na Figura 1. Nesse momento também se define a abordagem de banco de dados que será utilizada: hierárquica, de rede ou relacional.

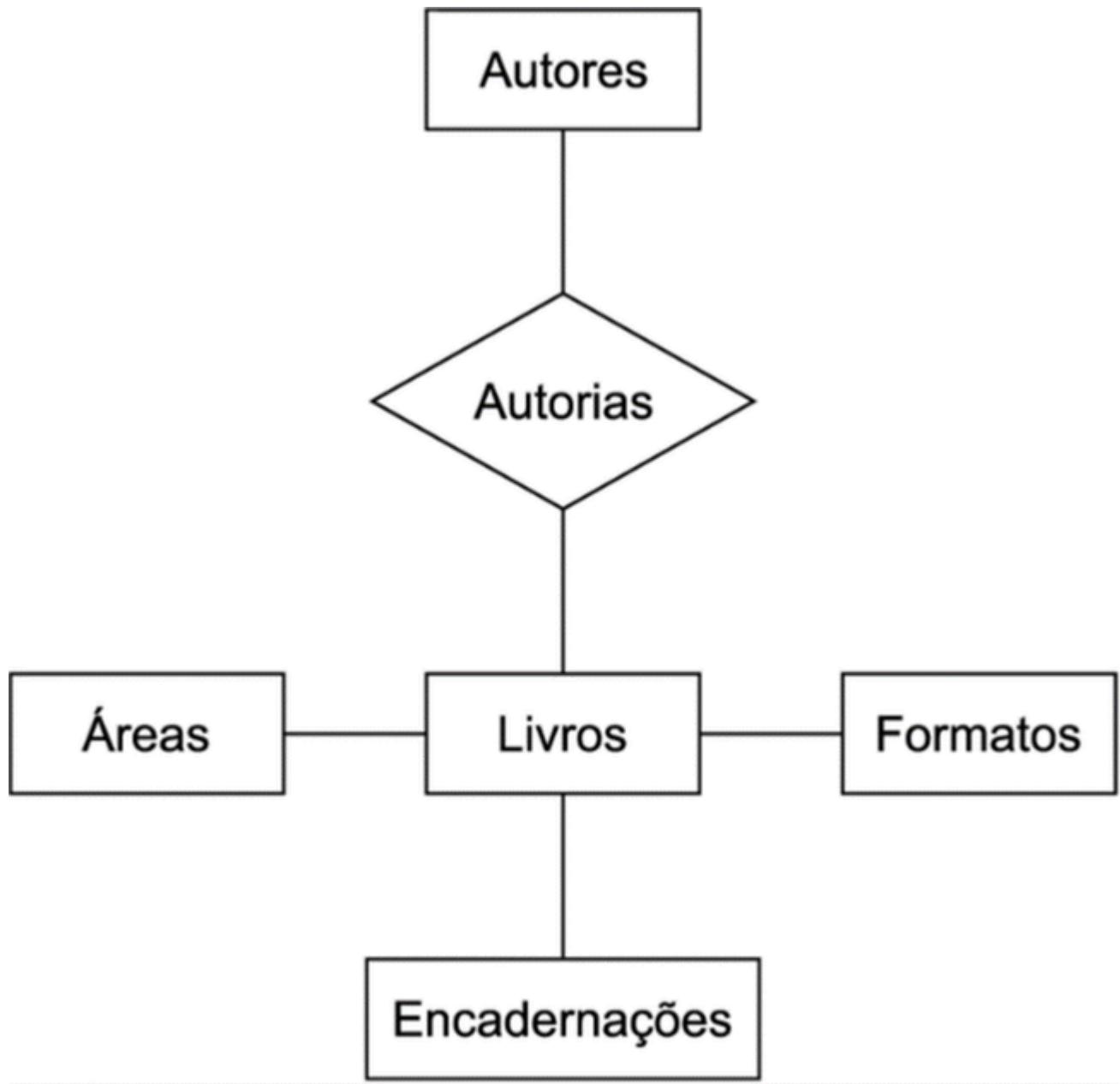


Figura 1 | Diagrama do modelo lógico do banco de dados. Fonte: Alves (2020).

Do modelo lógico, podemos derivar o **modelo físico**, no qual se encontram detalhados os componentes de estrutura física do banco de dados, como tabelas, campos, tipos de valores, índices etc. Quando chegarmos a esse ponto, estaremos prontos para a criação propriamente dita do banco de dados utilizando o sistema gerenciador que mais se adequar às nossas necessidades.

Uma vez que a realidade muda de uma empresa para outra, é preciso estabelecer uma forma padrão para se estruturar um banco de dados, independentemente do tipo de ambiente ou negócio. Assim, definiu-se o que normalmente conhecemos como metamodelo, sendo o mais utilizado o tipo entidade-relacionamento (ER). Um modelo de dados ER é composto de três

classes de objetos: entidades, relacionamentos e atributos. Segundo Silberschatz, Korth e Sudarshan (2020), o modelo ER também tem associada a ele uma representação esquemática, o diagrama de ER. Um diagrama ER pode expressar, de forma gráfica, a estrutura lógica geral de um banco de dados. Os diagramas ER são simples e claros – qualidades que podem ser responsáveis, em grande parte, pelo uso generalizado do modelo ER.

Objetos ou eventos do mundo real, isto é, atividades que geram atributos que precisam ser mantidos, podem ser modelados e convertidos em entidades (tabelas), como clientes, empresas, funcionários, produtos, reservas, serviços, aluguéis etc.

Vamos Exercitar?

Nesta aula, estudamos alguns conceitos relacionados à definição de modelagem de dados e os tipos de modelagem mais utilizados, como o modelo conceitual, cujo alvo é a definição do problema, e não a sua solução, que mostra o que precisará existir no banco de dados de uma forma geral, sem se preocupar de que forma isso será realizado no SGBD; o modelo lógico, que, por sua vez, é mais complexo que o modelo conceitual, necessita de mais detalhes de como os dados se relacionam; e o modelo físico, que se preocupa em implementar os modelos descritos anteriormente em algum SGBD. Também vimos que, para um problema que desejamos resolver, o passo inicial é criar um modelo conceitual e observar os requisitos levantados, por meio dos quais poderemos retirar várias informações. Assim, de forma visual e prática, conseguiremos avançar ainda mais a solução do problema com ajuda do modelo lógico e, por fim, implementar nosso banco modelado em algum SGBD apropriado.

Saiba mais

Para uma melhor compreensão dos assuntos abordados em aula, recomendamos que você visite a seguinte página na web:

[MODELAGEM de dados: modelo conceitual, modelo lógico e físico](#). Blog Utilidade Pública: Tecnologia, Educação e Cidadania.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.

Aula 2

Elementos do modelo Entidade-Relacionamento (ER) - I

Elementos do modelo entidade-relacionamento (ER) - I

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula, avançaremos ainda mais sobre o tema modelagem entidade-relacionamento (MER). A base do modelo relacional é um conceito matemático denominado relação, no qual dois conjuntos de números possuem termos relacionados entre si. No modelo conceitual, um conjunto é chamado de entidade; já no modelo lógico, é chamado de tabela. Cada tabela é definida por um conjunto de atributos, também conhecidos como campos, que descrevem suas características particulares. A representação gráfica da modelagem relacional é a forma de representação dos componentes do modelo lógico de um banco de dados. Essa representação é uma parte muito importante da compreensão do esquema do banco de dados. É preciso, então, estudar mais a fundo as entidades, os relacionamentos e os atributos dentro da MER, dado que esta é a seção primordial para continuarmos nossos estudos em modelagem de dados. Bons estudos!

Vamos Começar!

Para Sordi (2019), **entidade** pode ser conceituada como algo que existe física ou virtualmente, identificável unicamente por intermédio de suas características (seus atributos). Um sistema de informação pode manipular dados de diversas entidades. Imaginemos um sistema educacional que monitore o desempenho dos alunos em cada disciplina. Há muitos alunos em instituições e eles podem ser representados por um tipo de entidade que chamamos de Aluno. Da mesma forma, muitas disciplinas podem ser representadas por um tipo e uma entidade (tipo entidade) denominada Disciplina. Um registro contendo vários dados (atributos) de um estudante é uma instância da entidade Aluno. Uma entidade (tipo entidade), assim, pode armazenar dados de vários alunos, ou seja, de múltiplas instâncias.

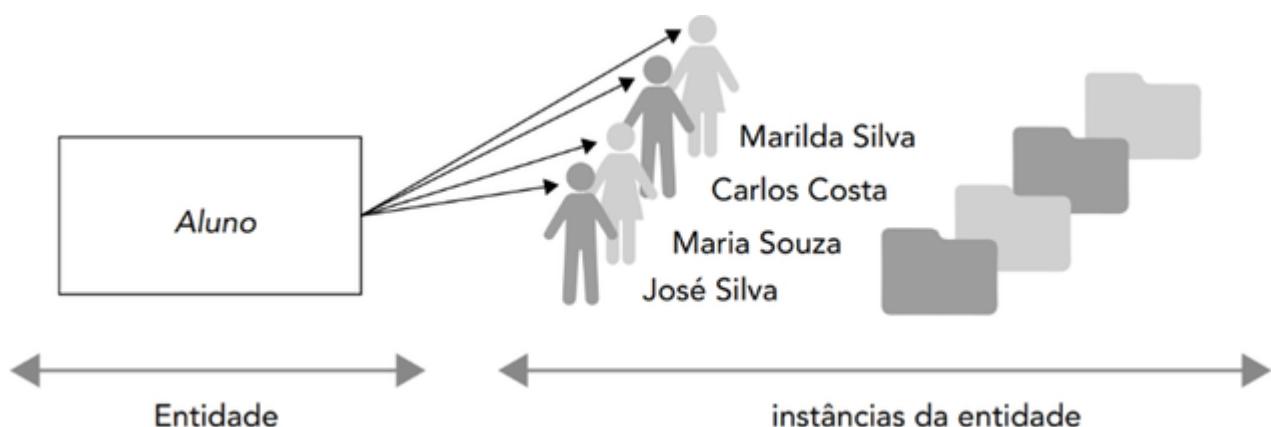


Figura 1 | Conceito de entidade e instâncias da entidade. Fonte: adaptada de Sordi (2019).

Ainda segundo Sordi (2019), as entidades (*entity types*) podem ser de três tipos: forte (ou fundamental), fraca (ou atributiva) e associativa. A entidade forte é a mais comum; existe independentemente de qualquer outra entidade, e por essa razão geralmente a chamamos apenas de entidade. Já uma entidade fraca depende de outra entidade (forte). Por exemplo, a entidade dependente de funcionários só existirá se houver instâncias de funcionários que declarem filhos ou outros entes como dependentes. Portanto, a entidade dependente é fraca por depender da entidade Colaborador. Observe que, por padrão, os nomes das entidades são usados sempre no singular: Colaborador, Dependente, Cliente. Por fim, temos, ainda, a entidade associativa, que caracteriza o relacionamento entre duas ou mais entidades.

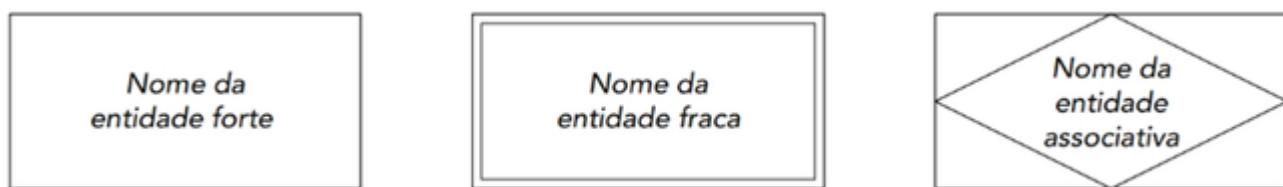


Figura 2 | Notações gráficas para os três tipos de entidades. Fonte: adaptada de Sordi (2019).

De acordo com Silberschatz, Korth e Sudarshan (2020), um relacionamento é uma associação entre várias entidades. Podemos, a título de exemplo, definir um relacionamento mentor que

associa o instrutor Katz ao aluno Shankar. Esse relacionamento especifica que Katz é o mentor do aluno Shankar. Um conjunto de relacionamento é, pois, um conjunto de relacionamentos do mesmo tipo.

Considere os dois conjuntos de entidades, Instrutor e Aluno. Definimos o conjunto de relacionamentos mentor para indicar a associação entre os instrutores e os alunos que atuam como seus mentoreados. A Figura 3 ilustra essa associação, porém, para simplificar, somente alguns atributos dos dois conjuntos de entidades foram mostrados.

Uma instância de relacionamento em um esquema ER representa uma associação entre as entidades nomeadas nas empresas reais que estão sendo modeladas. Como ilustração, a entidade individual *instrutor Katz*, que possui o ID de instrutor 45565, e a entidade *aluno Shankar*, que possui o ID de aluno 12345, participam de uma instância de relacionamento de mentor. Ela representa que, na universidade, o instrutor Katz está aconselhando o aluno Shankar.

Um conjunto de relacionamento é representado em um diagrama ER por um losango, que é ligado por meio de linhas a uma série de conjuntos de entidades (retângulos) diferentes. O diagrama ER da Figura 4 mostra os dois conjuntos de entidades, Instrutor e Aluno, relacionados por meio de um conjunto de relacionamento binário mentor.

Siga em Frente...

Como outro exemplo, considere os dois conjuntos de entidades Aluno e Seção, em que seção indica a oferta de um curso. Podemos definir o conjunto de relacionamento para indicar a associação entre o aluno e o curso no qual ele está matriculado.

Embora, nos casos anteriores, cada conjunto de relacionamento fosse uma associação entre dois conjuntos de entidades, em geral, um conjunto de relacionamento pode indicar a associação de mais de dois conjuntos de entidades.

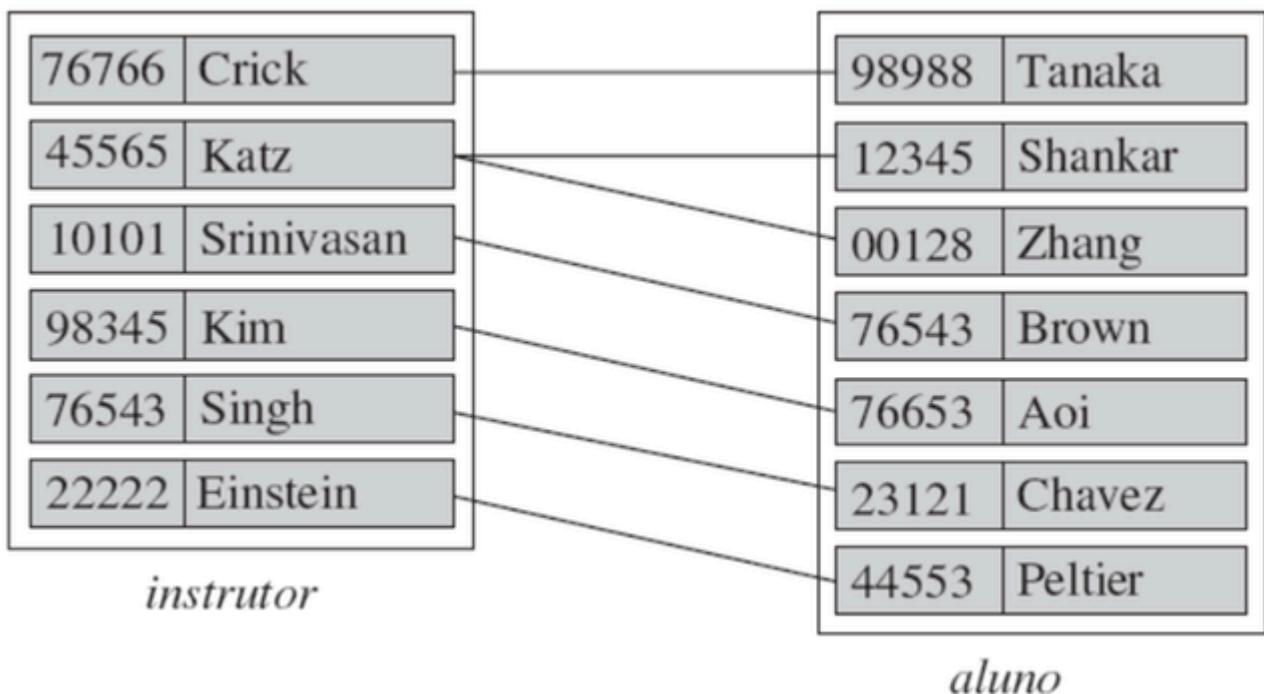


Figura 3 | Conjunto de relacionamento mentor. Fonte: adaptada de Silberschatz, Korth e Sudarshan (2020).

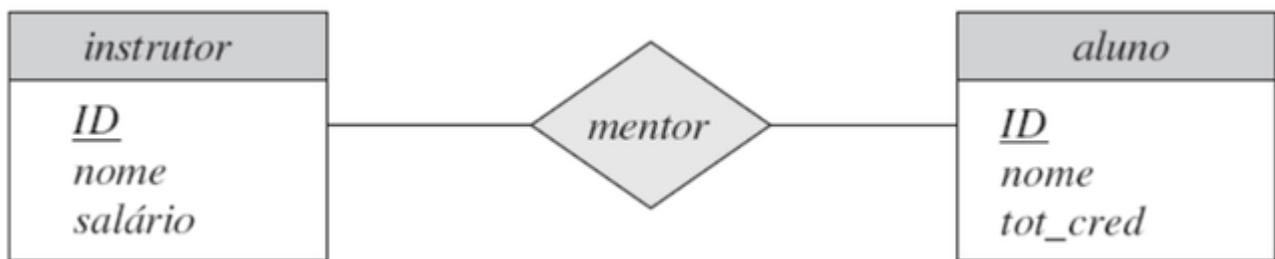


Figura 4 | Diagrama ER e o conjunto de relacionamento mentor. Fonte: adaptada de Silberschatz, Korth e Sudarshan (2020).

Para Sordi (2019), pela análise da definição de entidade, podemos inferir que atributo é sinônimo de característica da entidade. De fato, muitos autores o definem como uma característica ou propriedade desta. Por exemplo, a entidade Aluno pode ser descrita pelos atributos número da Matrícula (ID), nome, data de nascimento, sexo, endereço, telefone e e-mail. O termo ID, nesse contexto, significa identificador, ou seja, atributo único e diferenciador de uma instância perante todas as demais instâncias da entidade.

Já para Silberschatz, Korth e Sudarshan (2020), o atributo de um conjunto de relacionamento é representado em um diagrama ER por um retângulo não dividido. Ligamos o retângulo por uma linha tracejada até o losango que representa esse conjunto de relacionamento. A Figura 5 mostra, de maneira exemplar, o conjunto de relacionamento “realiza” entre os conjuntos de entidades Seção e Aluno. O atributo descriptivo “nota” é ligado ao conjunto de relacionamento “realiza”. Um conjunto de relacionamento, vale dizer, pode ter vários atributos descriptivos; por exemplo, também podemos armazenar um atributo para crédito com o conjunto de relacionamento

“realiza” a fim de registrar se um aluno realizou a seção para crédito ou se está auditando (ou apenas assistindo) o curso.

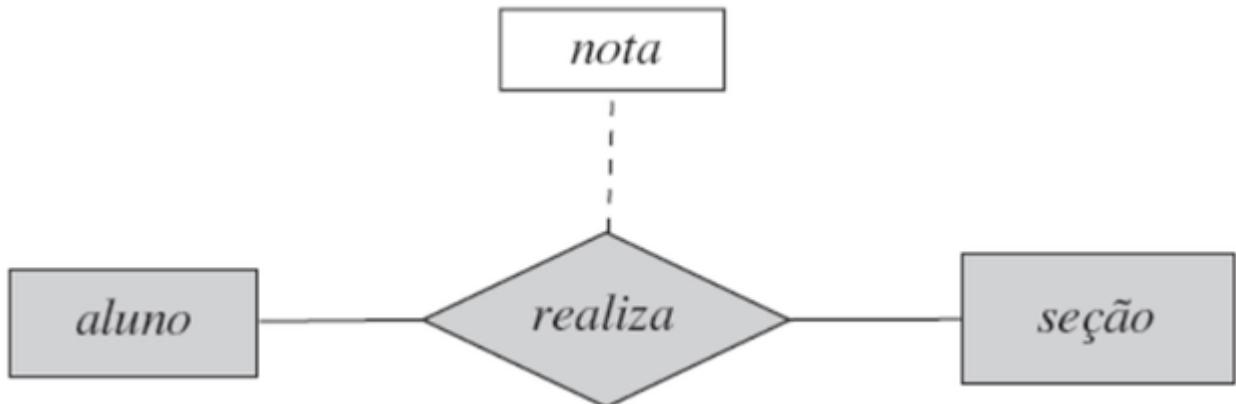


Figura 5 | Diagrama ER com um atributo ligado a um conjunto de relacionamento. Fonte: adaptada de Silberschatz, Korth e Sudarshan (2020).

Vamos Exercitar?

Nesta aula, aprendemos mais alguns conceitos relacionados à modelagem de dados. Com a introdução dos temas entidades, relacionamentos e atributos, temos, agora, mais alguns subsídios para o início do processo de modelagem de dados. Entender suas diferenças e como identificá-los é muito importante, pois através deles é que conseguimos construir um banco de dados e deixá-lo funcional e pronto para uso. Como vimos, com ajuda dos diagramas de entidade-relacionamento (DER) é possível visualizar as entidades, os relacionamentos entre elas e os atributos ligados.

Para a continuação dos estudos sobre MER, caro estudante, sugerimos que comece a praticar. Na literatura, podemos encontrar várias bibliografias com exemplos e exercícios relacionados ao tema. Ponha a mão na massa e avance ainda mais seus estudos teóricos aliados à prática.

Saiba mais

Para uma melhor compreensão dos assuntos abordados em aula, recomendamos que você visite a seguinte página na web:

[MODELAGEM de dados: modelo conceitual, modelo lógico e físico](#). Blog Utilidade Pública: Tecnologia, Educação e Cidadania.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

Aula 3

Elementos do modelo Entidade-Relacionamento (ER) - II

Elementos do modelo entidade-relacionamento (ER) - II

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula, estudaremos conceitos avançados sobre entidades, atributos e relacionamentos do MER (modelagem entidade-relacionamento). O modelo entidade-relacionamento (MER) foi desenvolvido para melhorar o design do banco de dados e permitir a especificação de um modelo conceitual, segundo Silberschatz, Korth e Sudarshan (2020). Este é o modelo mais comumente usado em sistemas de gerenciamento de banco de dados. Foi

desenvolvido por Edgar F. Codd em 1970, mas somente em 1987 é que as empresas de software começaram a usá-lo.

É preciso destacar, nesse contexto, que a generalização e a especialização são conceitos empregados na representação de objetos do mundo real que compartilham atributos semelhantes e podem ser classificados em uma hierarquia, exibindo as relações de dependência entre as entidades de uma categoria específica. Considere, pois, uma companhia de seguros que comercializa apólices para uma clientela composta por indivíduos e empresas. Neste exemplo, podemos observar que a entidade Cliente pode ser subdividida em duas outras entidades que compartilham atributos em comum, como Pessoa Física e Pessoa Jurídica.

Logo, convidamos você, caro estudante, para refletirmos sobre os conceitos de especialização/generalização em modelagem de dados bem como sobre outros tópicos pertinentes ao tema. Bons estudos!

Vamos Começar!

O modelo de dados entidade-relacionamento, concebido por Peter Chen, tem sido aplicado para a comunicação com os utilizadores finais, mostrando entidades e relacionamentos. No entanto, quando empregado para unir múltiplos modelos conceituais, cada um com diferentes utilizadores finais, ele enfrenta restrições significativas até que se recorra a um conceito de abstração de dados chamado generalização.

Segundo Machado (2020), a generalização ocorre quando se define um subconjunto de relacionamentos entre duas ou mais classes de dados.

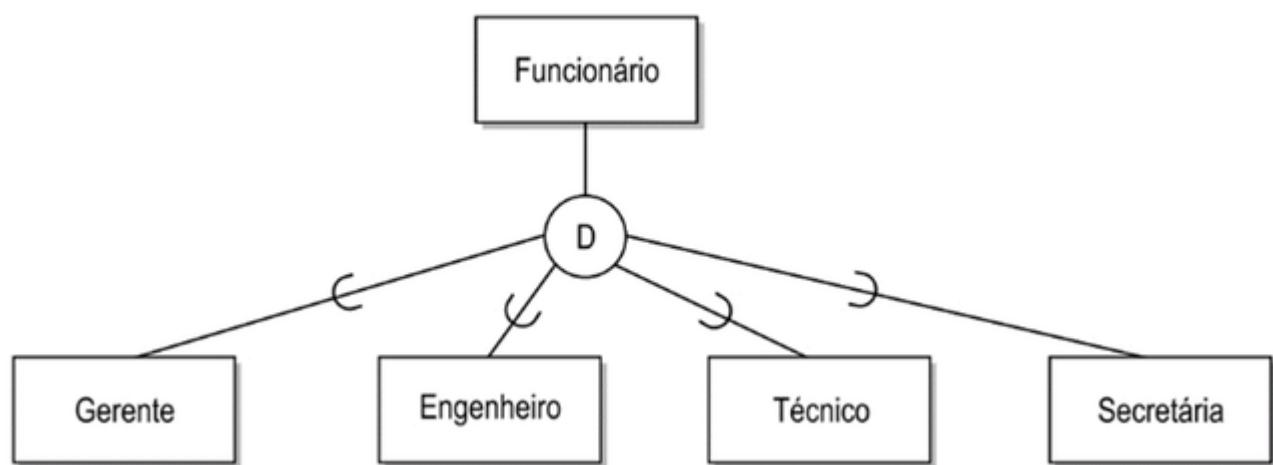


Figura 1 | Exemplo de generalização. Fonte: adaptada de Machado (2020).

Ainda para o autor (2020), quando examinamos a entidade Funcionário sob a perspectiva de diferentes grupos de utilizadores finais, é notório que a classe de dados que identificamos como funcionários representa uma abstração que engloba outras classes, tais como gerentes, engenheiros, secretárias, técnicos de sistemas e assim por diante.

Em geral, a generalização ocorre quando entidades que possuem atributos em comum são generalizadas em alto nível como uma entidade só, como uma entidade genérica ou uma superclasse de dados. De acordo com Machado (2020), as entidades de nível mais baixo que fazem parte desse supertipo são denominadas subtipos e refletem a especialização de partes da entidade supertipo.

A simbologia para representar o relacionamento entre a entidade supertipo e a entidade subtipo é muito variada. Em geral, utilizamos um pequeno círculo na intersecção das linhas que levam o supertipo até os subtipos, com uma indicação em seu interior.

Entidades associativas são aquelas que surgem devido ao tipo de conexão que existe entre as tabelas. O nome desse tipo de tabela deve ser descritivo, como Contrato ou Histórico, e é comum que seja uma combinação dos nomes das tabelas envolvidas. Em termos de requisitos de banco de dados, esse tipo de tabela geralmente se relaciona a ações ou eventos, como atender, contratar ou prescrever. Quando ocorre uma relação entre duas entidades, o número de vezes que uma se associa à outra define a intensidade do relacionamento ou cardinalidade entre as tabelas.

Para Sordi (2019), a cardinalidade de uma instância de relação é uma maneira aproximada de medir quantas instâncias de uma ou mais entidades estão conectadas à própria instância ou a outra(s) entidade(s). No modelo de dados, usamos dois símbolos para representar essa cardinalidade: o valor mínimo e o valor máximo. Para o valor mínimo da cardinalidade, temos duas escolhas: zero (0) ou um (1). O valor zero indica que a conexão é opcional, enquanto o valor um implica que a conexão é obrigatória, o que é fundamental na modelagem das regras de negócios. Quanto ao valor máximo da cardinalidade, também temos duas opções: um (1) ou muitos (M). Na representação gráfica, qualquer valor maior que um é o que importa, indicando que existem várias conexões, embora não forneça um número exato.

Siga em Frente...

Segundo Silberschatz, Korth e Sudarshan (2020), na notação do diagrama ER, indicamos as restrições de cardinalidade em um relacionamento que pode ser representado por um desenho de uma linha direcionada (⤠) ou uma linha não direcionada (⤠) entre o conjunto de relacionamentos e o conjunto de entidades em questão. Especificamente, para o exemplo da universidade, tem-se:

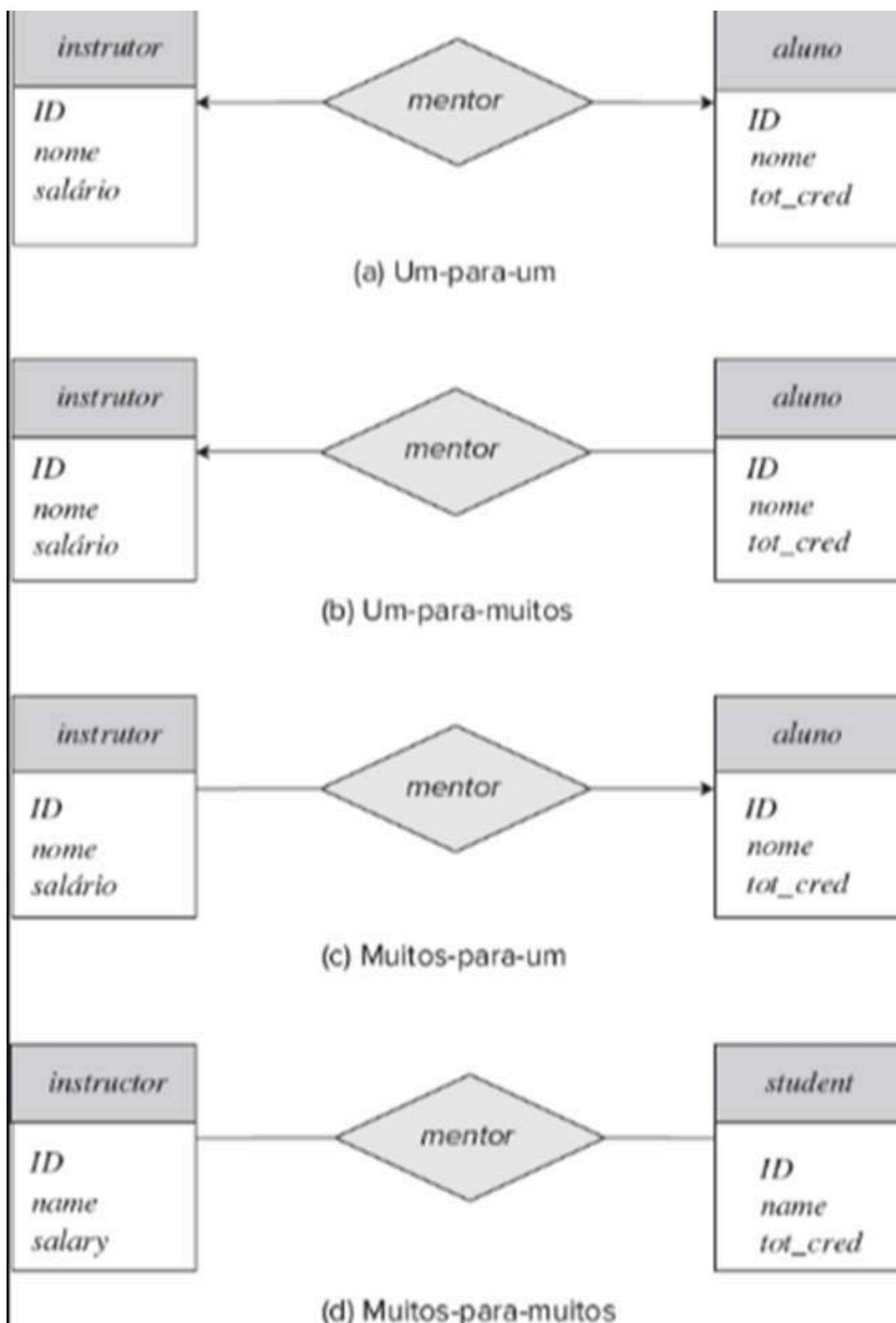


Figura 2 | Cardinalidades de relacionamento. Fonte: adaptada de Silberschatz, Korth e Sudarshan (2020).

Ainda de acordo com Sordi (2019), a presença de um grupo de entidades E em um conjunto de relacionamentos R é denominada participação total quando todas as entidades em E estão envolvidas em pelo menos um relacionamento em R. Se apenas algumas das entidades em E participam dos relacionamentos em R, então a presença do grupo de entidades E no relacionamento R é considerada parcial.

Em uma universidade, assim, pode ser necessário que cada aluno tenha pelo menos um orientador; na modelagem ER, isso implica que cada entidade aluno deve estar ligada a pelo menos um instrutor por meio do relacionamento de orientação. Portanto, a participação dos alunos no conjunto de relacionamentos de orientação é total. No entanto, um instrutor não necessariamente precisa orientar um aluno. Assim, é possível que apenas algumas das entidades instrutor estejam relacionadas com o grupo de entidades aluno por meio do relacionamento de orientação, tornando a participação dos instrutores no conjunto parcial.

Para determinar a cardinalidade entre as tabelas, é necessária muita atenção. Perguntamos sempre se existe a possibilidade de repetição, e se existir, será N; mas não havendo essa possibilidade, será 1. Por exemplo, um médico pode atender mais de um paciente? Claro que sim! E o paciente? Pode ser atendido por mais de um médico? Sim, e em várias ocasiões. Fica claro que o relacionamento entre as tabelas Médico e Paciente terá a cardinalidade de N para N. Agora, se o paciente precisar informar o seu tipo sanguíneo? Ele não poderá informar mais de um tipo, mas esse mesmo tipo sanguíneo pode se repetir em outros pacientes? Com certeza! Pronto! Teremos um relacionamento N para 1 entre as tabelas Paciente e Tipo Sanguíneo. Observe, na Figura 3, como fica o diagrama de entidade-relacionamentos (DER) dessa situação descrita: Médico, Paciente e Tipo Sanguíneo.

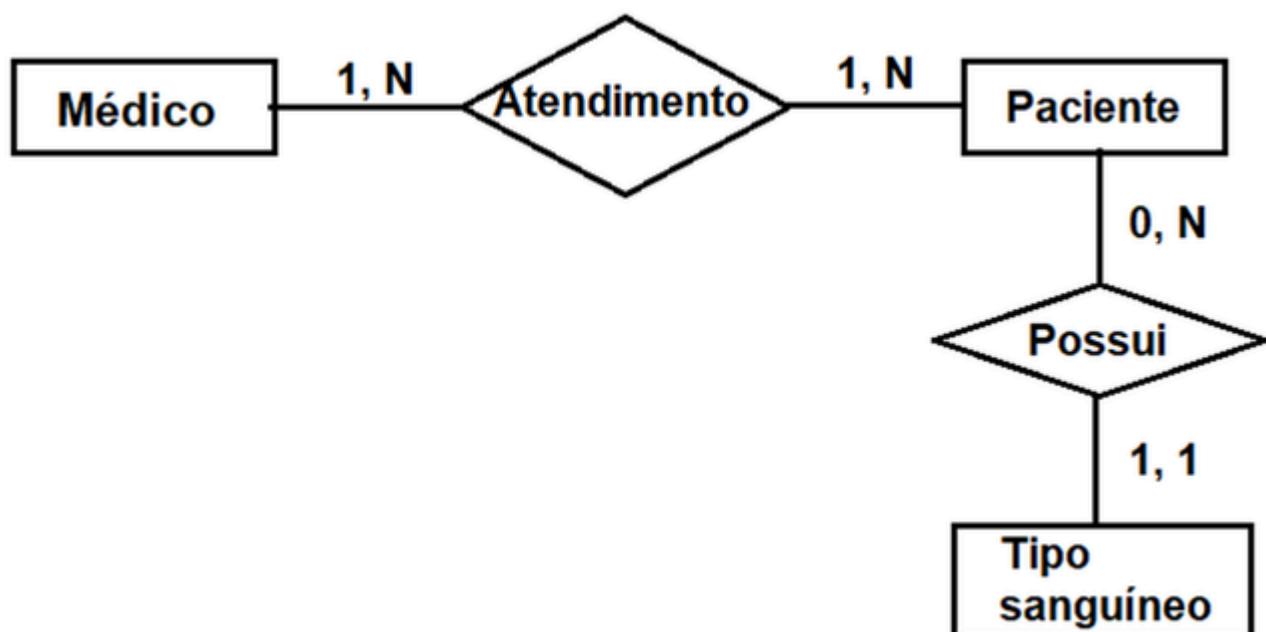


Figura 3 | Exemplo de DER.

A determinação da cardinalidade é um aspecto crítico para a eficácia de um banco de dados. Após concluir a modelagem desse banco, é aconselhável compartilhar o diagrama com outras pessoas, uma vez que suas opiniões podem identificar questões que foram inadvertidamente negligenciadas durante o processo. Além disso, em situações em que a cardinalidade não está claramente definida, é essencial consultar o cliente para obter mais esclarecimentos.

Vamos Exercitar?

Com os conhecimentos adquiridos nesta aula, você, estudante, poderá detalhar ainda mais suas modelagens utilizando-se de cardinalidades nos relacionamentos, bem como do conceito de entidades associativas e de generalização/especialização. Com ajuda dos diagramas de entidade-relacionamento (DER), é possível visualizar as entidades, os relacionamentos entre elas e as cardinalidades, além das generalizações, como citado anteriormente no exemplo da entidade Cliente. Uma sugestão para a continuação dos estudos sobre MER é começar a praticar. Na literatura, podemos encontrar várias bibliografias com exemplos e exercícios relacionados ao tema. Coloque a mão na massa e avance ainda mais seus estudos teóricos aliados à prática.

Saiba mais

Para uma melhor compreensão dos assuntos abordados em aula, recomendamos a leitura das referências bibliográficas e a visita à seguinte página na web:

PRATA, J. F. [Um roteiro para facilitar o ensino e o aprendizado na elaboração de projetos conceituais de bancos de dados](#). *Exacta*, São Paulo, v. 4, n. 1, p. 113-121, jan./jun. 2006.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados**: estudos de casos abrangentes da concepção lógica à implementação. 1. ed. São Paulo: Érica, 2019.

Aula 4

Elementos do modelo Entidade-Relacionamento (ER) - III

Elementos do modelo entidade-relacionamento (ER) - III

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula nos debruçaremos sobre mais alguns componentes importantes de entidades e cardinalidade na modelagem MER e iremos introduzir o conceito de chave. Um banco de dados consiste em múltiplas tabelas que estão conectadas entre si. É fundamental que cada uma delas possua um nome único e representativo. Por exemplo, uma tabela destinada a armazenar informações sobre carros pode ser nomeada Garagem, em vez de usar um nome genérico como Tabela_A. Além disso, tabelas possuem certas características distintas. Uma tabela é percebida como uma estrutura composta por filas e colunas (bidimensional). Cada fila ou registro representa uma única instância da entidade no conjunto de entidades. Cada coluna representa um atributo distinto e tem um nome único em relação aos outros. Cada ponto de interseção entre uma fila e uma coluna contém um valor singular que corresponde aos dados da tabela. Todos os valores em uma coluna devem seguir o mesmo formato. Vale dizer que a ordem das colunas e filas não tem relevância para um sistema de gerenciamento de banco de dados (SGBD). E cada tabela deve representar um atributo (também conhecido como chave, como discutiremos melhor posteriormente) ou uma combinação de atributos que identifique de maneira única cada fila.

Assim, para lidar com essas características, acompanhe esta aula que lidará com esses e outros assuntos pertinentes ao MER. Bons estudos!

Vamos Começar!

Alves, em seu livro *Banco de dados: teoria e desenvolvimento* (2020), ressalta que pode existir uma categoria de entidades desprovida de atributos-chave, o que implica que não é possível distinguir uma entidade individual dentro desse conjunto, uma vez que entidades idênticas podem existir. Essas entidades são chamadas de entidades fracas. Já as entidades que possuem atributos-chave são conhecidas como entidades fortes.

Em outras palavras, para Silberschatz, Korth e Sudarshan (2020), cada tabela que representa uma entidade no modelo conceitual pode ser categorizada como uma entidade forte ou uma entidade fraca. Essa diferenciação é estabelecida com base na análise de duas condições fundamentais: a dependência de existência e a dependência de identificador. Uma entidade é considerada fraca se pelo menos uma dessas formas de dependência estiver presente no relacionamento entre duas entidades. Conforme Elmasri e Navathe (2018) explicam, uma entidade fraca sempre implica em uma restrição de participação total e uma dependência de existência em relação a uma entidade específica. Os autores classificam os tipos de entidades da seguinte maneira:

- **Entidades fortes:** tabelas independentes que não requerem outras para a sua existência. Exemplos incluem as entidades Aluno, Curso, Cliente, Empresa e Paciente. Na análise de requisitos, essas entidades são facilmente identificadas, pois são substantivos robustos e de significado claro.
- **Entidades fracas ou dependentes:** tabelas que dependem de outras entidades para existir e só são viáveis devido à presença de uma entidade forte. Entidades fracas são frequentemente representadas por retângulos com bordas duplas (embora, em muitos modelos atuais, sejam representadas apenas por retângulos). Por exemplo, a tabela Dependente só existe porque a tabela Funcionário está presente; para cada dependente registrado, deve haver um funcionário que esteja vinculado a ele. Sem a existência da tabela Funcionário, a tabela Dependente não teria razão de existir.

Entidades fortes e fracas podem ter um relacionamento. Por exemplo, a entidade Aluno pode estar relacionada à entidade Curso. Desta forma, é preciso entender também as variações do número de relacionamentos, ou melhor, a cardinalidade pelas quais as entidades se relacionam, pois, como vimos no exemplo anterior, a entidade Curso pode estar relacionada a várias entidades Aluno ao mesmo tempo, porém, nem sempre a entidade Aluno pode estar relacionada a várias entidades Curso.

Assim, segundo Sordi (2019), há na literatura menções a quatro tipos de variações de cardinalidades pelas quais as entidades se relacionam: um para um (1:1), um para muitos (1:M); muitos para um (M:1) e muitos para muitos (M:N). Como a relação muitos para um é a leitura inversa da relação um para muitos, a maioria dos autores acaba por excluí-la da tipologia. Essas designações têm um caráter mais geral, sendo menos detalhadas e precisas. Elas se referem aos valores máximos das cardinalidades em ambos os sentidos da relação. Por exemplo, a relação um para muitos (1:M) indica que a instância de uma entidade está relacionada a várias instâncias de outra entidade, sem entrar em pormenores sobre as cardinalidades mínimas de ambas no que

diz respeito ao relacionamento. A seguir, descrevemos cada uma das quatro variações em mais detalhes.

Um para um (1:1): na relação de cardinalidade um para um, cada instância de uma entidade está ligada a uma única instância de outra entidade. Para ilustrar o cenário, considere uma empresa que fornece um dispositivo de comunicação (um celular) a cada um de seus funcionários de vendas. A Figura 1 mostra um extrato do modelo de entidade-relacionamento (MER) relacionado ao sistema de alocação de ativos dessa empresa. Neste caso, cada vendedor possui exatamente um telefone celular, e cada celular está atribuído a um único vendedor. Assim, a relação de um para um é válida nos dois sentidos, indicando a quantidade máxima de relações permitidas em ambos os lados da interligação.

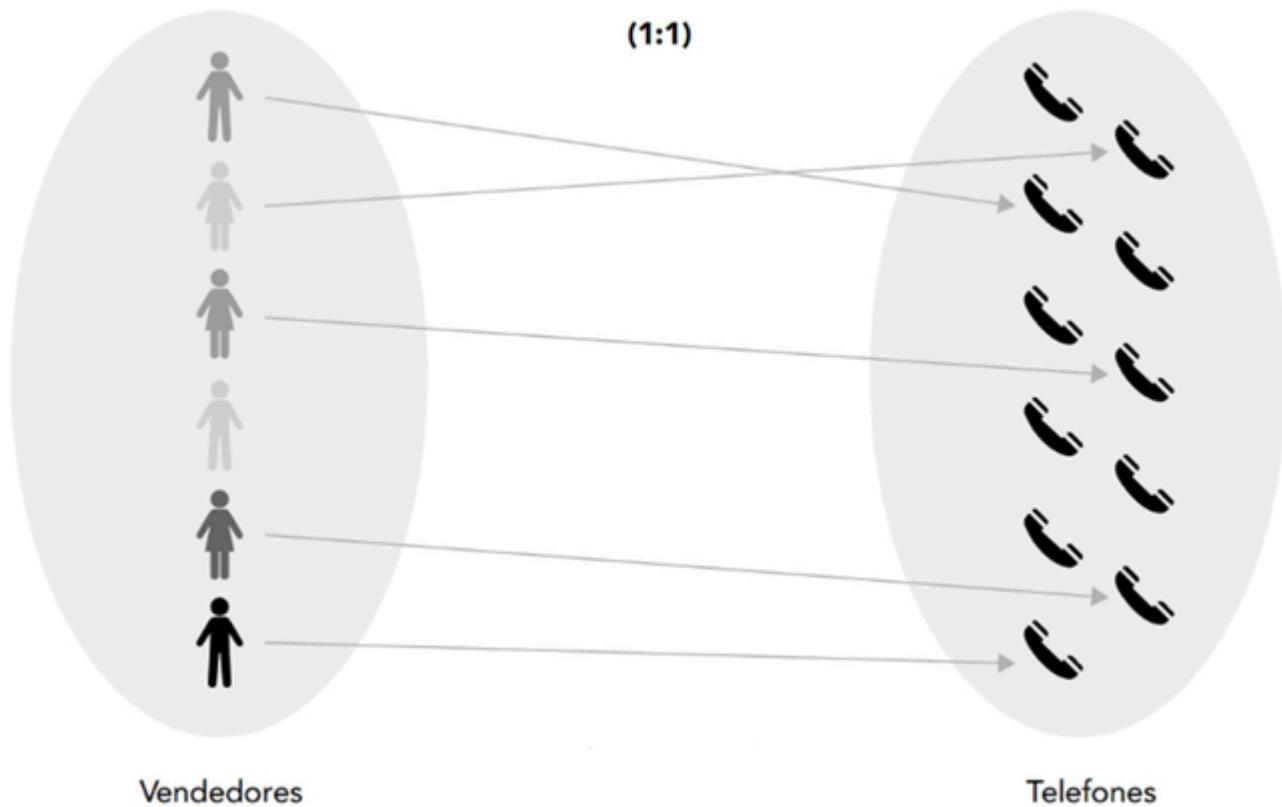


Figura 1 | Exemplo de instância de relacionamento com cardinalidade 1:1. Fonte: adaptada de Sordi (2019).

Um para muitos (1:M): Na relação de cardinalidade um para muitos, a instância de uma entidade está associada a várias instâncias de outra entidade. Por exemplo, consideremos o caso de uma escola em que cada criança deve ter o nome e o número de telefone de um adulto responsável. Dado que um pai ou mãe pode ter mais de uma criança matriculada na mesma escola, é possível que um responsável cuide de várias crianças na instituição. Dessa forma, uma única instância da entidade Responsável pode estar relacionada com várias instâncias da entidade Aluno. No sistema de gerenciamento de matrículas da escola, o modelo de entidade-relacionamento (MER) pode ser representado como na Figura 2, o que permite que cada responsável esteja vinculado a mais de um aluno no sistema de registro da escola.

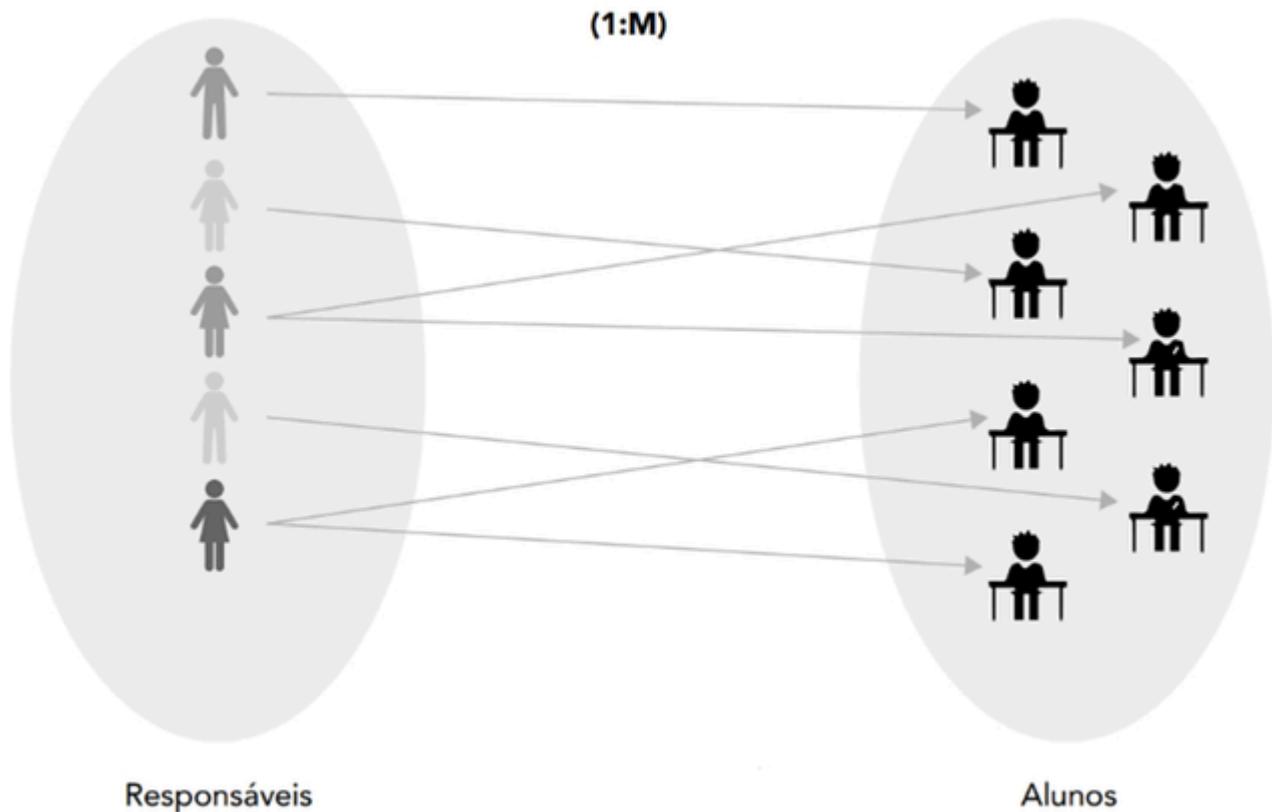


Figura 2 | Exemplo de instâncias de relacionamento com cardinalidade 1:M. Fonte: adaptada de Sordi (2019).

Muitos para um (M:1): a relação muitos para um é a inversa da relação um para muitos. Conforme ilustrado na Figura 2, na parte superior, a leitura ocorre da esquerda para a direita, o que significa que vários alunos podem ter o mesmo responsável, mas cada um deles possui, no máximo, um responsável. Ao ler o modelo de entidade e relacionamento (MER) da esquerda para a direita, ou seja, no contexto de muitos para um, observamos que um Aluno está associado a um único Responsável.

Muitos para muitos (M:N): na relação muitos para muitos, a instância de uma entidade pode estar relacionada com várias instâncias de outra entidade, e vice-versa. Na Figura 3, apresenta-se um cenário típico, no qual uma instância da entidade Aluno está matriculada em várias disciplinas, estabelecendo relacionamentos com diversas instâncias da entidade Disciplina.

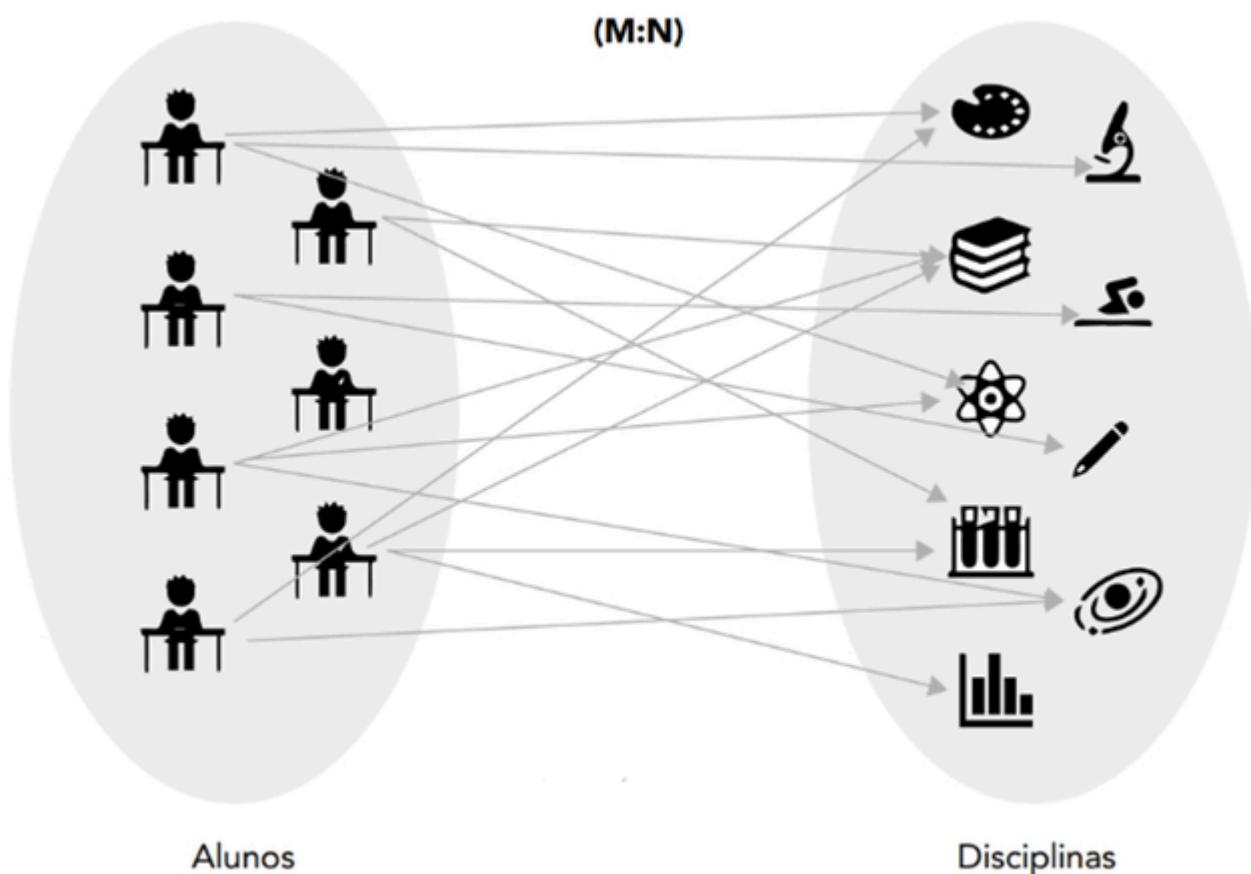


Figura 3 | Exemplo de instâncias de relacionamento com cardinalidade M:N. Fonte: adaptada de Sordi (2019).

Siga em Frente...

Para Silberschatz, Korth e Sudarshan (2020), é preciso haver uma maneira de especificar as entidades dentro de determinado conjunto de entidades e relacionamentos. Para tanto, se faz necessário o uso de chaves. Segundo Alves (2020), chave é o componente de uma relação que pode ser formada por um ou mais atributos, cuja função é permitir identificar uma linha na relação ou registro em uma tabela. Uma chave pode ser classificada como superchave, chave primária e chave estrangeira.

Uma **superchave** é uma restrição que impede a ocorrência de valores idênticos em atributos de duas ou mais entidades distintas. Em termos simples, isso possibilita a identificação única de uma linha dentro de uma relação. Em algumas situações, é possível que uma superchave contenha atributos que não sejam necessários para essa identificação única da linha. Por exemplo, em uma tabela de registros de clientes, a combinação do código e do número de CPF pode formar uma superchave. No entanto, dado que o código é exclusivo para cada cliente, o CPF não se faz necessário para identificá-lo de forma única.

Já uma **chave primária** é um atributo ou conjunto de atributos em uma tabela que serve para identificar exclusivamente seus registros. Além disso, desempenha um papel semelhante ao de

um índice, automaticamente ordenando os registros, o que determina a ordem de exibição. As chaves primárias podem ser formadas por um ou mais campos, e, neste último caso, são conhecidas como chaves compostas. Uma chave primária garante que não haja registros duplicados, ou seja, nenhum valor pode se repetir nos campos que a compõem, o que otimiza a velocidade de pesquisa. Por exemplo, ao procurar um registro em um banco de dados de clientes com base no número de CPF, se esse campo for uma chave primária, a busca será mais rápida.

Ao escolher os campos para definir uma chave primária, dois critérios principais devem ser considerados. Em primeiro lugar, deve-se optar por campos de tamanho reduzido para acelerar as atualizações e consultas da chave primária. Da mesma forma, quando se trata de uma chave composta, deve-se minimizar o número de campos incluídos. O segundo critério diz respeito à estabilidade do valor armazenado no campo da chave primária. Esse valor deve ser constante, ou seja, não deve ser alterado frequentemente, porque os bancos de dados usam a chave primária para estabelecer relações entre tabelas.

No entanto, em algumas situações, um campo que faz parte da chave primária pode precisar ser atualizado. Por exemplo, em uma tabela de cadastro de produtos na qual o campo de código do produto contém o código de barras da embalagem, pode ser necessário alterá-lo. Nesse caso, o procedimento correto seria adicionar um novo registro e desabilitar o antigo para não perder o histórico do produto. Os sistemas de bancos de dados relacionais também oferecem a opção de atualização em cascata, que atualiza automaticamente os valores nas tabelas relacionadas quando um campo de chave primária é alterado. No entanto, isso pode afetar o desempenho, especialmente com um grande número de registros para atualizar.

Idealmente, os campos de chaves primárias devem ser compostos por valores sequenciais, calculados pelo sistema, seja por meio do SGBD ou por rotinas do aplicativo que utiliza o banco de dados. É importante notar que os campos que compõem chaves primárias devem ser obrigatoriamente preenchidos.

Quanto às **chaves estrangeiras**, elas têm a função de estabelecer relações entre os registros de uma tabela com os registros de outra por meio da chave primária desta última. Para ilustrar esse conceito, considere duas tabelas com as seguintes características:

1. Uma tabela de registro de áreas de livros, na qual há um campo de código de área (Codigo_Area) designado como chave primária.
2. Uma tabela de registro de livros que inclui um campo de código de área ao qual o livro pertence (Codigo_Area).

É notável que o campo chamado Codigo_Area está presente nas duas tabelas, mas na tabela de registro de livros, ele não é a chave primária. No entanto, é utilizado para estabelecer uma relação com a tabela de áreas, na qual o campo Codigo_Area é a chave primária. Esse campo, nesse contexto, é conhecido como chave estrangeira.

Ainda segundo Alves (2020), além da função principal de estabelecer relações entre tabelas, as chaves estrangeiras também são amplamente utilizadas para fornecer valores de outra tabela, a qual contém a chave primária. Por exemplo, podemos incorporar uma caixa de seleção na tela de

cadastro de um aplicativo de gerenciamento, permitindo aos usuários escolher a área de um livro, em vez de digitarem manualmente o código da área. Isso não apenas simplifica a entrada de informações, como também reduz a probabilidade de erros nos dados inseridos. Embora seja comum que o nome do campo (ou conjunto de campos) que define a chave primária seja o mesmo da chave estrangeira, não há impedimento para que eles sejam diferentes.

Vamos Exercitar?

Com toda bagagem adquirida até aqui, já possível construir nossas próprias modelagens através do MER. Com os aprendizados de hoje, você, caro estudante, saberá diferenciar entidades fracas e fortes, os tipos de cardinalidades em relacionamentos entre entidades bem como os tipos de chaves. Continue avançando seus estudos sobre MER e revise o conteúdo visto até aqui, procurando colocá-lo em prática. Lembre-se de que participar do processo de modelagem de dados visando, no futuro, a concepção de um banco de dados é muito enriquecedor, pois agrega experiência à vida profissional.

Saiba mais

Para uma melhor compreensão dos assuntos abordados em aula, recomendamos a leitura das referências e, como complemento, do artigo *Um roteiro para facilitar o ensino e o aprendizado na elaboração de projetos conceituais de bancos de dados*, de José Ferreira Prata, que propõe um roteiro para a elaboração de modelos conceituais de banco de dados, abordando a busca por tabelas e estabelecendo relacionamentos entre elas.

PRATA, J. F. [Um roteiro para facilitar o ensino e o aprendizado na elaboração de projetos conceituais de bancos de dados](#). Exacta, São Paulo, v. 4, n. 1, p. 113-121, jan./jun. 2006.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados**: estudos de casos abrangentes da concepção lógica à implementação. 1. ed. São Paulo: Érica, 2019.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Chegada

Olá, estudante! Para desenvolver a competência desta unidade, isto é, compreender e ser capaz de elaborar uma modelagem de dados utilizando a modelagem entidade-relacionamento (MER) e seus elementos, é necessário que você pratique a leitura de vários conteúdos relacionados ao tema em paralelo a este material. Estudamos, nesta unidade, um conteúdo rico em teoria que foi tratado cuidadosamente para que pudesse ser consultado na hora de praticar as modelagens e de resolver os exercícios. Para um maior aprofundamento dos estudos neste tema, você deverá, primeiramente, ter claras as diferenciações de alguns conceitos, como de entidade fraca, forte e associativa, de generalizações/especializações de entidades, seus atributos e relacionamentos, bem como de sua cardinalidade, dentre outros tópicos que foram abordados ao decorrer da unidade. É de suma importância estar ciente de em qual contexto se está inserido para posteriormente saber identificar oportunidades de melhorias em modelagem de dados.

É Hora de Praticar!

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Flores Brasil é uma floricultura que trabalha com entregas de flores e presentes. Em datas especiais, o volume de pedidos aumenta consideravelmente. Para otimizar esse processo, será desenvolvido um software baseado na web. Sua tarefa, então, consistirá em criar um modelo conceitual para o banco de dados da floricultura.

Aqui estão os requisitos essenciais já identificados:

- É necessário registrar informações de clientes, pedidos, locais de entrega e produtos.
- Um cliente pode fazer vários pedidos, mas cada pedido está vinculado a um único cliente.
- Cada pedido pode conter diversos produtos.
- Os produtos são categorizados por tipo, como flores, chocolates, presentes, cartões etc.
- Cada pedido deve estar associado a um local de entrega.
- O controle de pagamento não é necessário, pois será tratado via cartão de crédito.

Com base nos requisitos mencionados, você deve criar um modelo conceitual visual e responder às seguintes perguntas:

- Quais tabelas farão parte do modelo conceitual?

Como será representado graficamente o relacionamento entre essas tabelas no modelo conceitual?

Algumas perguntas para reflexão:

- Na sua opinião, é possível criar o modelo lógico sem ter criado o modelo conceitual antes?
- O que significa, na prática, um relacionamento um para muitos? E muitos para muitos?
- Qual é a diferença principal entre chave primária e chave estrangeira?

Uma possível representação de um modelo conceitual pode ser visualizada na figura a seguir, na qual se tem uma visão geral do funcionamento da floricultura com base nos requisitos fornecidos. Esse modelo servirá como ponto de partida para a próxima fase de desenvolvimento, que é a criação do modelo lógico.

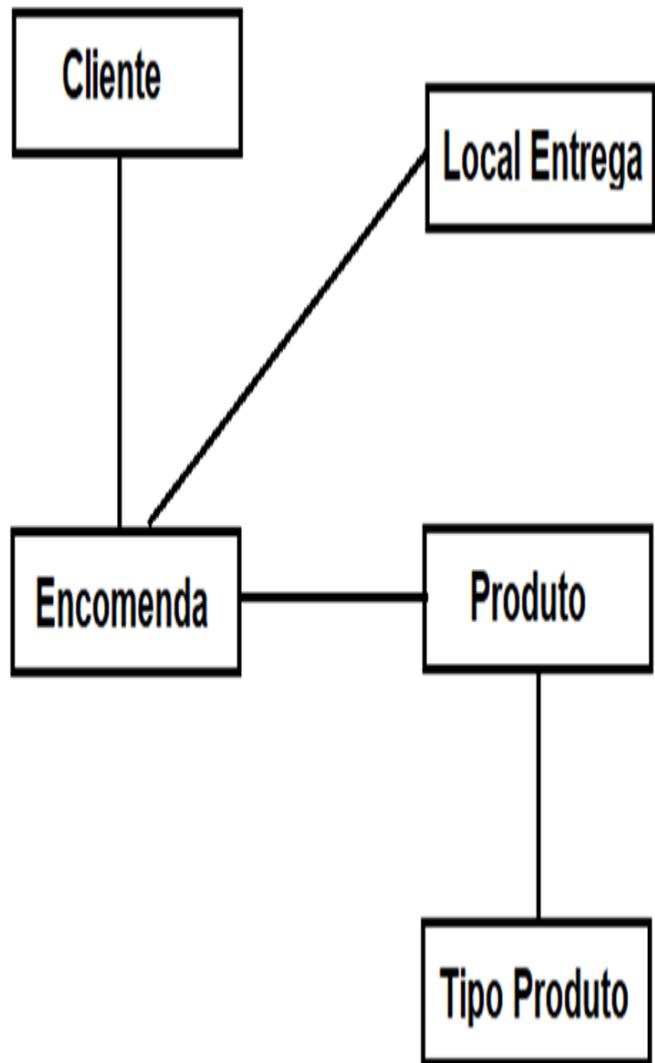


Figura 1 | Modelo conceitual.

Caro estudante, a seguir apresentamos um infográfico contendo um resumo dos principais temas abordados nesta unidade de forma simples e objetiva. Reserve um tempo para leitura desse material e reflita.



Figura 2 | Modelos de dados.

ALVES, W. P. **Banco de dados: teoria e desenvolvimento.** 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.

Unidade 3

Diagramas e Ferramentas de Modelagem

Aula 1

Diagrama Entidade-Relacionamento (DER)

Diagrama Entidade-Relacionamento (DER)



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Constantemente surgem inovações tecnológicas que trazem consigo novas perspectivas de emprego e empreendedorismo. Um profissional na área de tecnologia da informação deve, pois, realizar pesquisas, manter-se atualizado e permanecer vigilante em

relação às mudanças e às demandas do mercado. No entanto, uma habilidade verdadeiramente essencial é a capacidade de se adaptar a novas situações, às necessidades dos clientes e da empresa, já que esta será um fator determinante para o sucesso. Um projeto modesto e de pequena escala pode se transformar em uma excelente oportunidade de empreender. Logo, é importante estar sempre atento às modelagens que você realizará, pois elas podem dar origem a novas ideias de negócios que proporcionarão excelentes perspectivas para a sua carreira.

Nesta aula, abordaremos estratégias de modelagem de banco de dados e a importância de se documentar os diagramas criados. Lembre-se de que o diagrama de entidade-relacionamento (DER) faz parte do processo de modelagem de banco de dados e que, portanto, é uma etapa essencial na definição de padrões de desenvolvimento que permitirão a implementação eficiente do banco de dados ao final do processo. Bons estudos!

Vamos Começar!

Uma vez estudados os conceitos essenciais em modelagem entidade-relacionamento (MER), estamos prontos para criar um projeto conceitual de banco de dados e, posteriormente, seu diagrama entidade relacionamento (DER). Utilizaremos, aqui, um exemplo clássico oferecido por Alves (2020). Considere que, como designer de sistemas de banco de dados, você tenha sido encarregado de desenvolver um sistema para uma editora especializada em livros técnicos. Após conduzir entrevistas com todas as partes envolvidas no gerenciamento da editora, incluindo vendedores, diagramadores e gerentes de produção, você chegou à conclusão de que o banco de dados deve incluir as seguintes entidades:

- **Áreas:** códigos de área e descrições da área.
- **Formatos:** códigos de formato, descrições de formato e dimensões (altura e largura da página).
- **Encadernações:** códigos de encadernação e descrições de encadernação.
- **Autores:** códigos de autor, nomes dos autores, endereços completos (nome da rua, número, bairro, cidade e estado), CPF, RG, números de telefone, datas de nascimento, gênero, estado civil e locais de trabalho.
- **Livros:** códigos ISBN, títulos, formatos, tipos de encadernação, números de páginas, peso, custos, preços de venda, números de edição, anos de edição, números de reimpressão e números de contrato.

Durante o processo de coleta de requisitos, também foram identificados os seguintes tipos de relacionamentos:

1. Livro pertence à área

Razão de cardinalidade M:1, uma vez que um livro pertence a uma área específica, enquanto uma área pode ser atribuída a vários livros.

De um lado, há a restrição de participação total da entidade Livros, pois um livro deve, necessariamente, pertencer a uma área. Por outro lado, na entidade Áreas a restrição é parcial, pois pode haver áreas sem qualquer vínculo com algum livro.

2. Livro possui formato

Razão de cardinalidade M:1, dado que um livro somente pode possuir um formato de publicação, enquanto um mesmo formato pode ser atribuído a vários livros.

De um lado, há a restrição de participação total da entidade Livros, pois um livro deve, necessariamente, possuir um formato. Por outro, na entidade Formatos, a restrição é parcial, pois pode haver formatos sem qualquer vínculo com algum livro.

3. Livro possui encadernação

Razão de cardinalidade M:1, uma vez que um livro somente pode possuir um tipo de encadernação, enquanto um mesmo tipo de encadernação pode ser atribuído a vários livros.

Há a restrição de participação total da entidade Livros, pois um livro deve, necessariamente, possuir um tipo de encadernação. Já na entidade Encadernações, a restrição é parcial, pois pode haver um tipo de encadernação sem qualquer vínculo com algum livro.

4. Autor escreve livro

Razão de cardinalidade é M:N, pois podemos ter vários autores que escrevem vários livros, embora o mais comum seja um autor escrever vários livros. A restrição de participação de Autores é total, pois não se pode ter no cadastro de autores pessoas que não tenham escrito algum livro ou mesmo não ter assinado um contrato de edição. De igual modo, não se pode cadastrar um livro sem que haja ao menos um autor vinculado a ele. Assim, sua participação também é total.

Vamos, agora, transformar esse material descritivo em algo mais técnico e de fácil entendimento pelos profissionais responsáveis pelo projeto de banco de dados. Os relacionamentos entre as entidades estão, assim, representados pelos diagramas das Figuras 1 a 4.



Figura 1 | Relacionamento “livro pertence à área”. Fonte: Alves (2020).

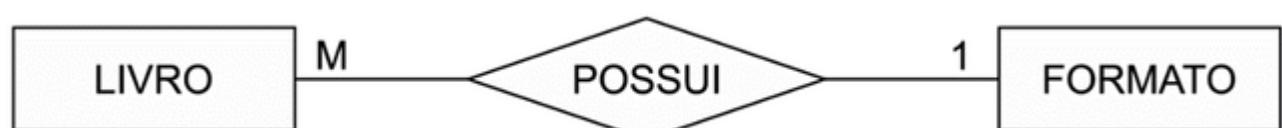


Figura 2 | Relacionamento “livro possui formato”. Fonte: Alves (2020).



Figura 3 | Relacionamento “livro possui encadernação”. Fonte: Alves (2020).



Figura 4 | Relacionamento “autor escreve livro”. Fonte: Alves (2020).

Segundo Alves (2020), para criar o diagrama entidade-relacionamento (DER), são requeridos diversos símbolos gráficos, de maneira semelhante ao que ocorre quando se pretende elaborar um fluxograma para representar uma rotina de programação.

A Figura 5 oferece uma visão geral dos principais símbolos empregados na elaboração dos diagramas entidade-relacionamento.

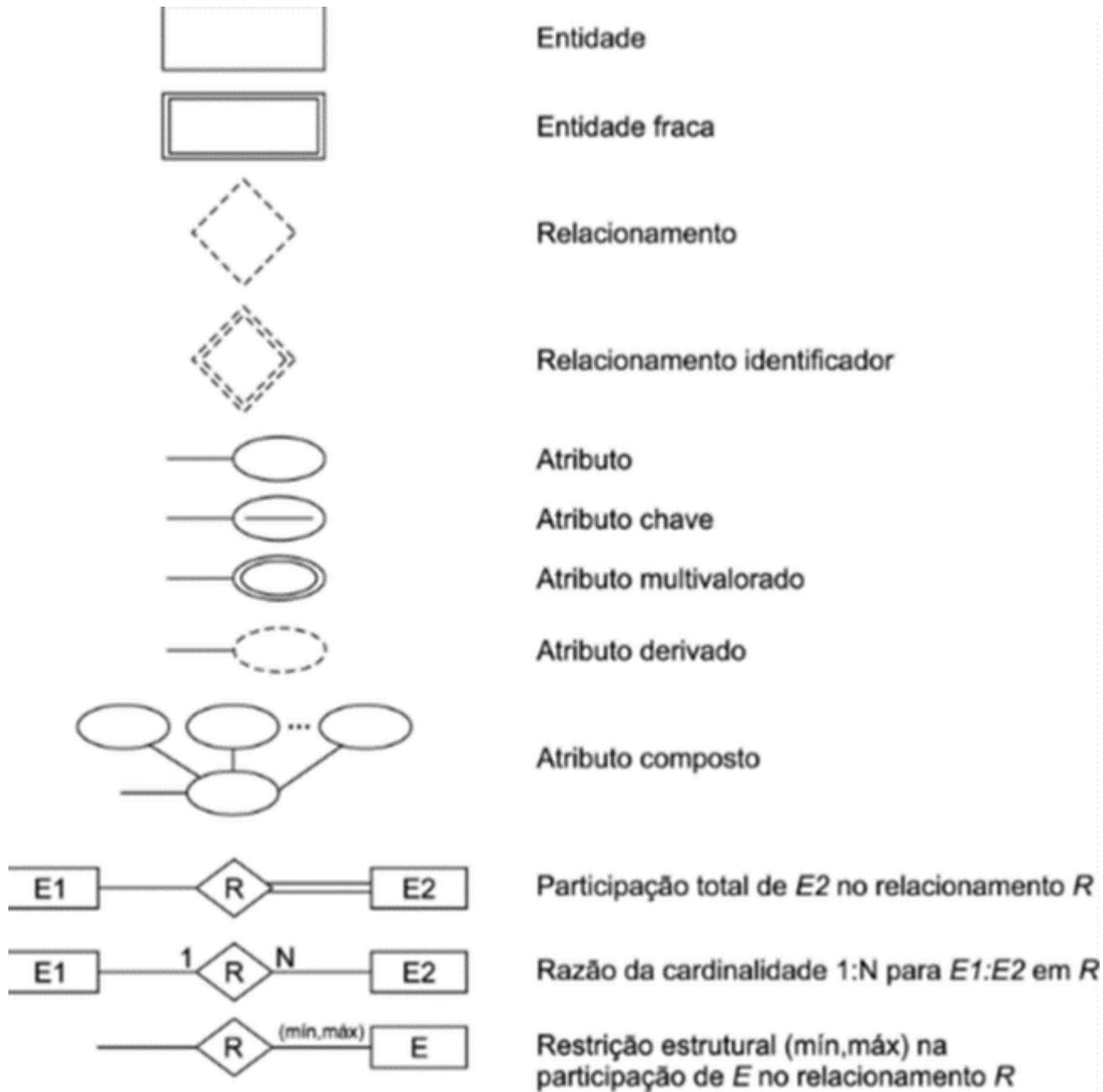


Figura 5 | Principais símbolos utilizados em diagramas entidade-relacionamento. Fonte: Alves (2020).

Já na Figura 6, podemos ver o diagrama completo das entidades e relacionamentos apresentados no tópico anterior.

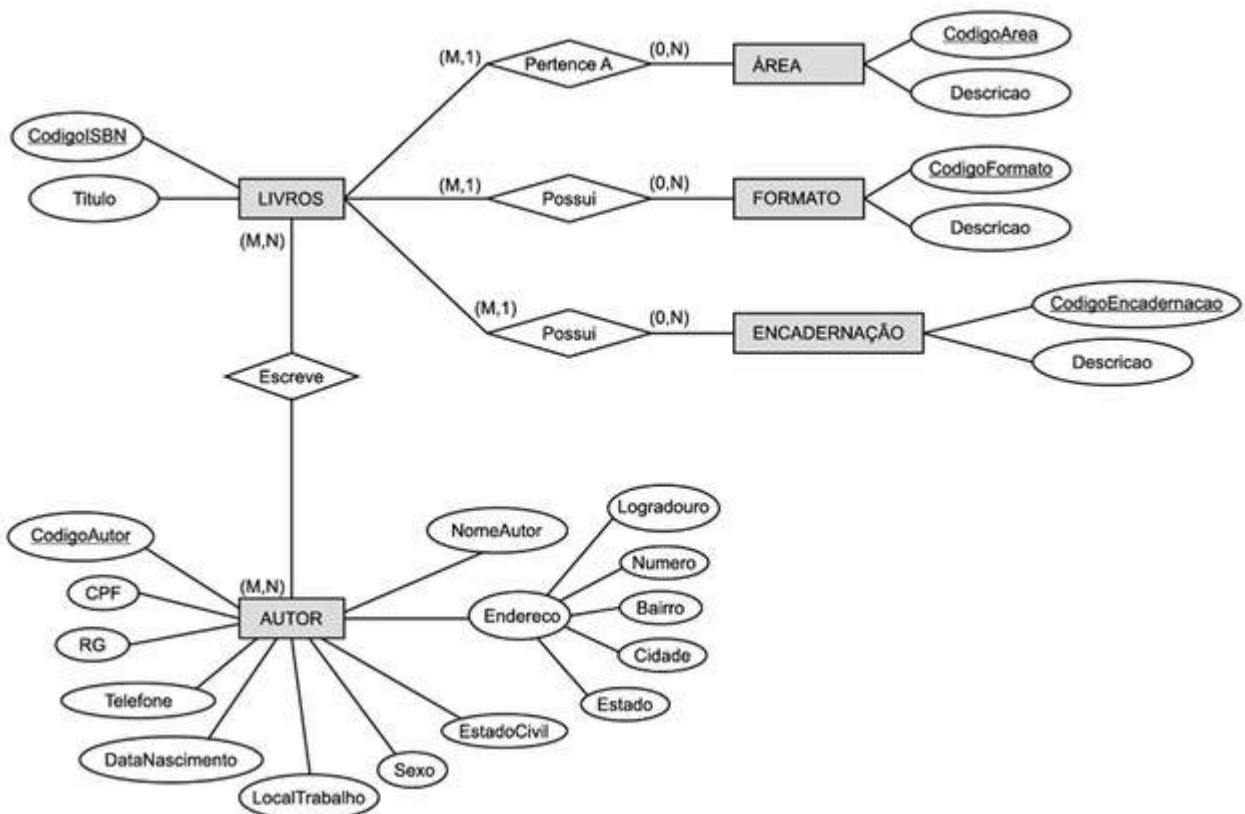


Figura 6 | DER do projeto de publicação de livro. Fonte: Alves (2020).

Repare que a figura anterior segue a notação de Peter Chen, na qual a cardinalidade das relações entre entidades é exposta com números (como 0, N, por exemplo) em cima das linhas, porém existem outros tipos de notações que podem ser expressas no DER. A Figura 7 mostra um resumo das principais cardinalidades e de como usá-las.

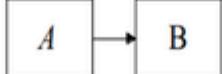
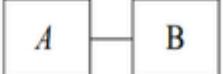
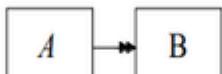
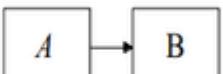
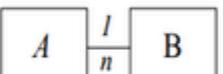
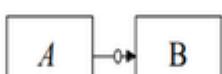
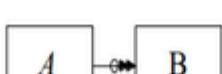
Grau Associação \ Notação	Crow's foot Pé de Galinha	Ross	Bachman	Chen
Um A está associado com um B				
Um A está associado com zero ou um ou mais B's				
Um A está associado com zero ou um B				
Um A está associado com zero, um ou mais B's				

Figura 7 | Resumo de notações de cardinalidade.

Siga em Frente...

A notação desenvolvida por Bachman é amplamente empregada em muitos aplicativos para a concepção de modelos lógicos de sistemas de banco de dados. Com o tempo, essa notação passou por modificações e evoluiu para o formato conhecido como notação de setas, que é adotado por diversas ferramentas de software.

Cardinalidade	Notação Original de Bachman	Notação de Setas
1:1	—	↔
1:N	→	↔
N:1	←	↔
M:N	↔	↔

Figura 8 | Notação de Bachman vs Notação de setas.

A notação desenvolvida por James Martin, notadamente seu renomado diagrama de pé-de-galinha, desfruta de grande popularidade entre as ferramentas que permitem criar modelos gráficos para representar esquemas de bancos de dados. Na Figura 8, apresentamos como são simbolizadas as cardinalidades nessa notação.

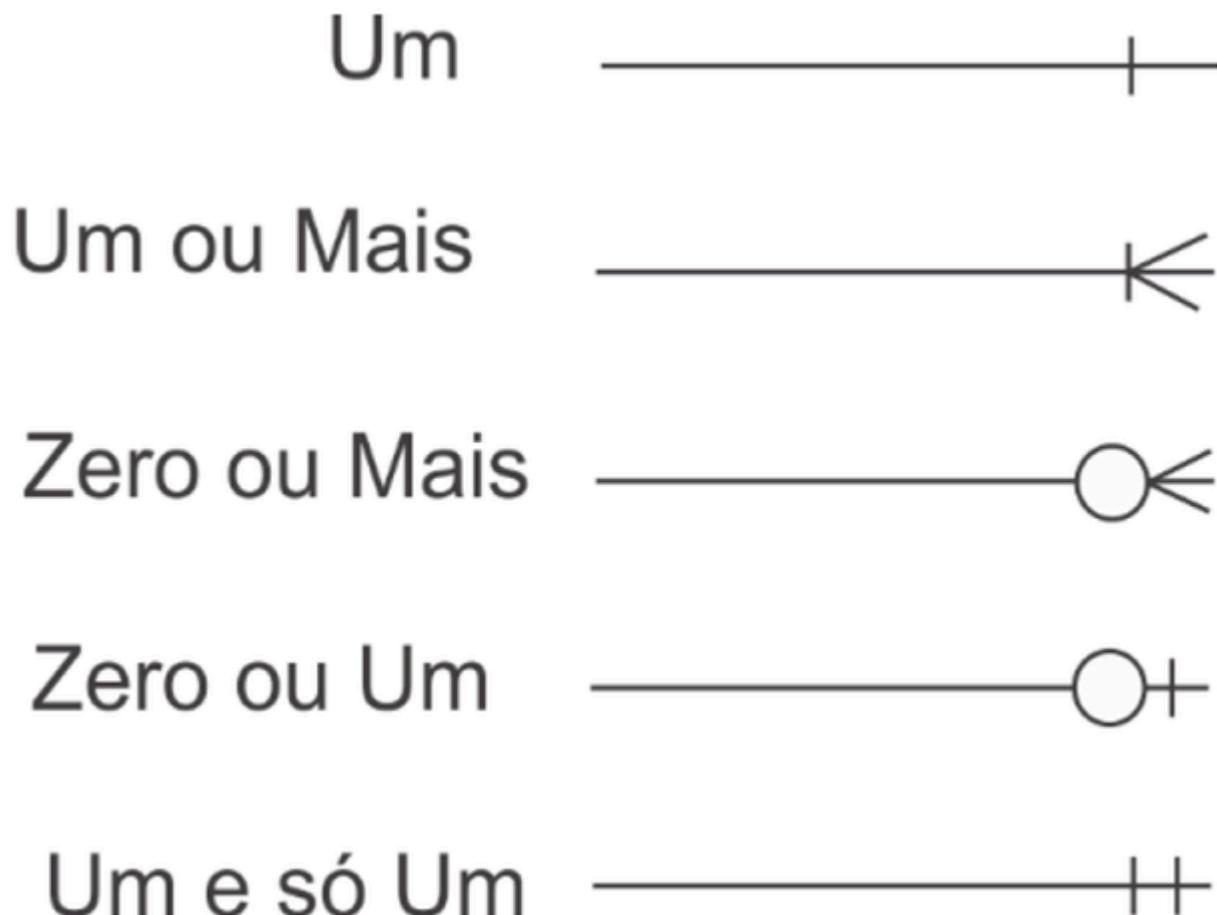


Figura 9 | Notação Pé de galinha.

Em resumo, para Machado (2020), a criação de um DER é uma etapa fundamental no processo de modelagem de dados, pois ajuda a representar de forma clara e visual como os dados estão organizados e como as entidades se relacionam umas com as outras. Com base no DER, é possível projetar o esquema do banco de dados e determinar como os dados serão armazenados e consultados no sistema. É, portanto, uma ferramenta essencial para profissionais que trabalham com design de sistemas de banco de dados.

Vamos Exercitar?

Nesta aula, vimos alguns conceitos relacionados à definição de diagrama entidade-relacionamento para modelagem em banco de dados. Analisamos exemplos de modelagens e construímos um DER utilizando conceitos já aprendidos. Aprendemos que há várias ferramentas disponíveis para a criação de DER, cada uma delas com suas próprias variações de notação. Uma distinção fundamental entre os conjuntos de entidades em um DER e os esquemas de relação derivados dessas entidades é a inclusão dos atributos dos relacionamentos entidade-relacionamento nos esquemas relacionais correspondentes. Algumas ferramentas de modelagem de dados permitem aos projetistas optarem por duas perspectivas diferentes da mesma entidade, uma delas representando-a sem esses atributos e a outra apresentando uma

visão relacional que inclui tais atributos. Nas próximas aulas, esses tópicos serão abordados mais a fundo.

Saiba mais

Para mais informações sobre DER e outros assuntos pertinentes à modelagem de banco de dados, recomendamos a leitura do quinto e oitavos capítulos dos livros [Banco de dados: projeto e implementação](#), de Felipe Nery R. Machado, e [Banco de dados: teoria e desenvolvimento](#), de William Alves, disponíveis na Minha Biblioteca.

Para um maior aprofundamento do tema, indicamos os artigos da revista [Database Trends and Applications](#), ISSN 1547-9897, disponível na ProQuest.

Referências

ALVES, W. P. **Banco de dados: teoria e desenvolvimento**. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.

Aula 2

Aspectos de projeto em banco de dados

Aspectos de projeto em banco de dados

Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo



para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Por um longo período, houve uma concepção de que o processo de projetar bancos de dados era uma disciplina independente, uma atividade distinta e, em certa medida, isolada dentro do ciclo de vida de um sistema. Essa atividade teria sua própria identidade e era conduzida com base em princípios e conceitos exclusivos da técnica de modelagem de dados. Para representar visualmente a etapa lógica da modelagem, sabe-se que o uso do diagrama entidade-relacionamento (DER) é extremamente interessante. Logo, estudaremos, nesta aula, algumas estratégias para modelagem em um DER. Aqui, vale lembrar que, assim que o modelo lógico começar a ser implementado, o modelo conceitual servirá de apoio à construção do esquema do banco de dados. Bons estudos!

Vamos Começar!

Banco de dados refere-se a qualquer software criado e implantado com o objetivo de gerenciar arquivos de dados em formato digital e organizado. O propósito fundamental de qualquer software de banco de dados é efetivamente lidar com grandes volumes de dados, o que torna essencial a observação e documentação cuidadosas de seu desenvolvimento e implementação. Esse processo é formalmente reconhecido como o ciclo de vida do banco de dados, que abrange cinco etapas distintas. Cada uma dessas fases representa os estágios pelos quais o programa é planejado, desenvolvido, avaliado e implantado em ambientes do mundo real. Adicionalmente, caso ocorra a identificação de problemas ou lacunas em qualquer fase, o ciclo de vida pode ser retrocedido ou reiniciado para eliminá-los, assegurando a criação de um software de banco de dados eficaz. As etapas, nesta ordem, geralmente são:

- **Planejamento:** engloba a coleta de informações necessárias e preparação de um bloco teórico sobre os requisitos mínimos para o qual o programa de banco de dados será desenvolvido.
- **Análise:** engloba a avaliação crítica do planejamento de desenvolvimento de banco de dados realizada na etapa anterior.
- **Desenvolvimento:** todos os parâmetros do projeto lógico de operações de banco de dados são finalizados e o desenvolvimento de software é feito em um período predeterminado.

- **Implementação:** o programa criado é colocado em prática em relação a um conjunto de dados (informações) específico, e seus parâmetros de operação são então ativados para executar as tarefas predefinidas.
- **Manutenção:** essa etapa costuma ser de longa duração, abrangendo todo o período destinado ao teste, gerenciamento, resolução de problemas e manutenção das funcionalidades do programa de banco de dados que foi desenvolvido.

De acordo com Machado (2020), a criação e manutenção de um dicionário de dados é um componente essencial da arquitetura de informações da empresa e desempenha um papel crucial no ciclo de vida de um banco de dados e na interpretação dos dados que descrevem a natureza do negócio. Um dos benefícios notáveis de manter um dicionário de dados bem elaborado é a promoção da consistência entre os itens em diferentes tabelas. Portanto, a sua constante atualização assume uma grande importância na área da administração de dados. Quando uma empresa estabelece um dicionário de dados de alcance corporativo, o objetivo deve ser fornecer definições semânticas precisas que se aplicam em toda a organização. Isso envolve tanto a precisão na criação do significado dos elementos de dados (componentes semânticos) quanto as definições que explicam como os elementos de dados são armazenados em formato digital (definições de representação).

A atualização contínua do dicionário de dados ocorre em resposta a qualquer solicitação relacionada à administração de dados que envolva a criação ou manutenção das estruturas de dados nos bancos de dados de produção da empresa.

De acordo com Alves (2020), as informações que fornecem descrições da estrutura do banco de dados são denominadas metadados. Esses metadados são armazenados no que é comumente referido como catálogo do sistema. O SGBD (ou o usuário) gerencia o catálogo do sistema quando há modificações na estrutura do banco de dados. Cada banco de dados mantém suas próprias informações no catálogo do sistema, o qual é consultado pelo SGBD para determinar o formato a ser empregado no acesso aos dados. Alguns sistemas, vale mencionar, costumam utilizar o termo “dicionário de dados” para se referir aos conjuntos de metadados.

Siga em Frente...

Em muitos projetos de banco de dados envolvendo um grande número de tabelas e campos, é crucial estabelecer padrões de desenvolvimento para evitar conflitos de nomenclatura de atributos, entre outros problemas. Em situações em que dois ou três analistas ou programadores estão trabalhando em uma modelagem e não existe um padrão definido, é possível que o mesmo campo seja criado e referenciado com nomes diferentes, o que pode complicar consultas e manutenções futuras. Portanto, é primordial criar um dicionário de dados para estabelecer padronização e documentação abrangente para cada tabela desenvolvida. Em síntese, segundo Silberschatz, Korth e Sudarshan (2020), um dicionário de dados é uma descrição detalhada dos dados armazenados em uma tabela contendo metadados. Um exemplo simplificado de dicionário de dados pode ser observado no Quadro 1.

Tabela: funcionário				
	Campo	Descrição	Tipo	Tamanho
PK	cd_func	Código do funcionário	VARCHAR	20
	nm_func	Nome do funcionário	VARCHAR	100
	cpf_func	CPF do funcionário	VARCHAR	15
	dt_nasc_func	Data de nascimento funcionário	DATE	-
FK	id_cidade	Cidade do funcionário	INTEGER	-

Quadro 1 | Dicionário de dados da tabela funcionário.

Cada empresa de desenvolvimento de software estabelece seu próprio formato de dicionário de dados. No entanto, Silberschatz, Korth e Sudarshan (2020) destacam que um dicionário de dados deve incluir informações básicas, como:

- Descrições dos nomes de tabelas, relacionamentos e atributos.
- Especificações de tipos de dados (domínio) e seus respectivos tamanhos.
- Detalhes minuciosos sobre as chaves usadas.
- Registros dos nomes dos usuários e suas permissões relacionadas às tabelas.

O nível de detalhamento do dicionário de dados é variável, mas acaba se tornando um documento essencial, especialmente quando o banco de dados enfrenta problemas que requerem manutenção. No contexto da modelagem de dados via DER, a maioria das fontes destaca duas estratégias de modelagem clássicas e tradicionais que podem ser aplicadas em um diagrama de entidade-relacionamento:

- **Estratégia *top-down*:** tem início com a identificação dos conjuntos de dados e, posteriormente, com a definição dos elementos em cada um desses conjuntos. Isso envolve a nomeação de diversos tipos de entidades e a especificação de seus atributos. Geralmente, essa abordagem é usada em bancos de dados de grande porte.
- **Estratégia *bottom-up*:** tem início com a identificação dos elementos de dados, ou seja, os itens individuais, e, em seguida, com o agrupamento desses itens para formar os conjuntos de dados. Neste caso, os atributos são identificados primeiramente e, ao agrupá-los, as tabelas são criadas. Essa técnica é comum em bancos de dados menores.

Frequentemente, as abordagens *top-down* e *bottom-up* se complementam, dado que muitos analistas ou projetistas aplicam ambas as técnicas ao mesmo banco de dados a ser modelado. Essa mescla resulta no desenvolvimento de uma abordagem híbrida conhecida como *middle-up-down*.

A etapa de modelagem conceitual em um projeto de banco de dados é considerada de alto nível, pois seu principal objetivo é promover uma compreensão fácil e eficaz entre os envolvidos no projeto, como destacam Silberschatz, Korth e Sudarshan (2020). Nessa abordagem, o foco está

em detalhar e discutir o funcionamento dos aspectos do negócio do cliente, sem se aprofundar na implementação de tecnologias específicas. Essa abordagem, portanto, deixa de lado os detalhes sobre como as informações serão armazenadas e recuperadas no banco de dados.

Vamos Exercitar?

Nesta aula, tivemos contato com novas metodologias para aperfeiçoar o desenvolvimento das modelagens de banco de dados. Você teve a oportunidade de explorar alguns aspectos adicionais a esse processo, como as estratégias de modelagem *top-down* e *bottom-down*, e de compreender a importância da documentação no desenvolvimento de um diagrama de entidade-relacionamento. À medida sua carreira avançar, você será apresentado a diversos modelos que são adaptados pelas empresas de acordo com suas necessidades de desenvolvimento. Sua capacidade de se manter atualizado com as novas tecnologias será um fator determinante para o seu sucesso profissional.

Nas aulas a seguir, então, discutiremos os padrões de modelagem usando a UML, uma linguagem de modelagem amplamente utilizada na programação orientada a objetos.

Saiba mais

Para uma melhor compreensão dos assuntos abordados em aula, indicamos a leitura do artigo de Ávila e Mello a seguir, o qual apresenta uma ferramenta de apoio ao processo de normalização de tabelas visando contribuir, principalmente, com projetos *bottom-up* de bancos de dados relacionais. Por meio da análise de dados, a ferramenta descobre dependências funcionais existentes que devem ser removidas.

ÁVILA, M. L. de; MELLO, R. S. [Uma ferramenta de apoio à normalização de tabelas relacionais baseada na análise de dados](#).

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação.** 1. ed. São Paulo: Érica, 2019.

Aula 3

Notação UML para modelagem de dados

Notação UML para modelagem de dados

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Desenvolver um software envolve um conjunto abrangente de habilidades, e uma das áreas de crescimento mais notável em TI é a abordagem de desenvolvimento orientada a objetos. Essa abordagem tem como princípio fundamental a reutilização de componentes em diferentes softwares. A reutilização é uma consideração essencial no campo de desenvolvimento de sistemas, pois permite economizar recursos. Para que isso se torne uma realidade, é necessário um planejamento sólido e a padronização dos processos de desenvolvimento de software.

Nesta seção, exploraremos uma ferramenta de padronização conhecida como UML, amplamente adotada por analistas de sistemas, que também pode ser aplicada ao processo de modelagem de banco de dados. A UML enfatiza a criação dos diagramas de casos de uso como o ponto de partida em um processo metodológico para o desenvolvimento de aplicações. No entanto, essa abordagem é uma herança dos conceitos da análise estruturada e análise essencial, nos quais os processos sempre receberam mais destaque e importância do que os dados.

Atente-se, pois, caro estudante, aos conceitos que apresentaremos aqui sobre modelagem de dados através do modelo entidade-relacionamento usando UML. Bons estudos!

Vamos Começar!

A UML (*Unified Modeling Language*), que se traduz como linguagem unificada de modelagem, é uma linguagem padrão amplamente utilizada para modelar sistemas orientados a objetos. Ela surgiu da combinação de três abordagens de destaque: BOOCH, OMT (Rumbaugh) e OOSE (Jacobson). A UML não é uma metodologia de desenvolvimento, mas uma linguagem que auxilia na visualização e comunicação de conceitos em sistemas orientados a objetos. Ela oferece uma maneira padronizada de representar o trabalho dos desenvolvedores em diagramas, sendo especialmente útil na criação de modelos de sistemas de software.

Além de fornecer as ferramentas necessárias para apoiar a prática de engenharia de software orientada a objetos, a UML também se destaca como uma linguagem de modelagem padrão para sistemas concorrentes e distribuídos. Ela utiliza uma variedade de técnicas de notação gráfica para criar modelos visuais de sistemas de software complexos, combinando as melhores práticas de modelagem de dados, processos de negócios, objetos e componentes. A UML é uma linguagem de modelagem única, comum e amplamente aplicável.

No entanto, vale ressaltar que a UML, sozinha, não define um processo específico para desenvolver softwares. Ela se concentra em responder "o que fazer" e "como fazer" ao representar o software por meio de modelos orientados a objetos, mas não aborda questões como "quando fazer" e "por que deve ser feito". Portanto, é essencial complementar a UML com um dicionário de dados completo, que descreva todas as entidades envolvidas, a fim de refinar os requisitos funcionais do software.

Segundo Silberschatz, Korth e Sudarshan (2020), a representação esquemática de um modelo de dados em uma aplicação desempenha um papel crucial no desenvolvimento de um esquema de banco de dados. Criar um esquema de banco de dados envolve não apenas a expertise de modelagem de dados, mas também o conhecimento de especialistas no domínio que compreendam as necessidades da aplicação, mesmo que não sejam necessariamente autoridades em modelagem de dados. Logo, comprehende-se uma representação esquemática intuitiva é de extrema importância, pois facilita a comunicação de informações entre esses dois grupos de especialistas.

Diversas notações alternativas para modelagem de dados foram propostas, sendo que as mais amplamente adotadas incluem a notação de diagramas ER e a notação de diagramas de classes UML. É importante observar que não há um padrão universal para a notação de diagramas ER, e diferentes livros e softwares de diagramas ER podem utilizar notações distintas. Vale, do mesmo modo, salientar que existem algumas das notações alternativas de diagramas ER, bem como a notação de diagramas de classes UML.

De acordo com Fowler (2004), a linguagem UML é composta por vários tipos de diagramas, dentre os quais podemos destacar os seguintes:

- **Diagrama de classes:** é o diagrama mais amplamente utilizado na linguagem UML, pois permite representar conjuntos de classes e suas relações.
- **Diagrama de objetos:** ilustra como as informações de um objeto são armazenadas em uma classe na prática.
- **Diagrama de caso de uso:** complementa o diagrama de classes, sendo particularmente útil na fase de especificação de requisitos do sistema, pois mostra os usuários e as funcionalidades do software.
- **Diagrama de sequência:** oferece uma visão orientada pelo tempo da colaboração entre objetos, destacando a ordem temporal em que as mensagens são trocadas.
- **Diagrama de atividades:** descreve o fluxo de tarefas que podem ser executadas pelo software ou por um ator.
- **Diagrama de estados:** representa os estados que um objeto pode assumir e os eventos que podem acionar a transição do objeto de um estado para outro.
- **Diagrama de componentes:** revela os componentes do software e suas relações, que podem incluir bibliotecas, arquivos de ajuda e classes que podem ser incorporadas ao software.

A cardinalidade empregada em um diagrama de classe é análoga àquela presente nos diagramas de entidade-relacionamento. No entanto, é possível especificar o número exato de ocorrências, por exemplo, estipulando um mínimo de uma ocorrência e um máximo de cinco. Se no diagrama de entidade-relacionamento a letra "N" representa "muitos", no diagrama de classes, utiliza-se o símbolo de asterisco (*) para a mesma finalidade. Note que, na Figura 1, a classe "Estado" possui dois atributos do tipo *string* e um método, enquanto a classe "Cidade" tem três atributos, sendo dois inteiros e um do tipo *string*, além de um método. É importante destacar que a quantidade de atributos e métodos pode variar de projeto para projeto, pois o desenvolvedor deve modelar a classe conforme considerar mais apropriado.



Figura 1 | Exemplo de relacionamento entre duas classes.

Quando uma classe é criada como uma instância, um objeto é concebido na memória do computador. Nesse cenário, os atributos da classe são inicializados com dados específicos. Além do diagrama de classes, há também o diagrama de objetos, que é usado para representar as informações que serão armazenadas na classe. A Figura 2, assim, ilustra como os atributos exibidos no diagrama da Figura 1 serão organizados no objeto.

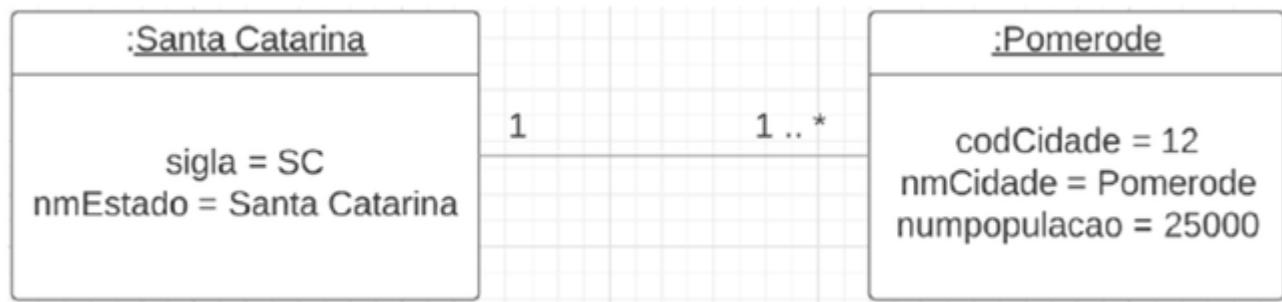


Figura 2 | Exemplo de diagrama de objetos.

Quando uma classe é criada como uma instância, as informações a ela associadas são retidas na memória RAM do computador. Se for necessário armazenar essas informações para uso futuro, é primordial gravá-las em arquivos ou em um banco de dados. Esse procedimento é conhecido como persistência. Uma classe de persistência tem como principal objetivo preservar as informações em um local duradouro, de modo que possam ser recuperadas em um momento posterior.

De acordo com Fowler (2005), na programação orientada a objetos, é possível estabelecer relações entre classes por meio de hierarquias. Uma classe derivada tem a capacidade de herdar as propriedades de outra, que neste contexto é chamada de classe base ou superclasse. Na estrutura de herança, as classes compartilham as funcionalidades e características que são comuns a todas elas, enquanto as subclasses têm a capacidade de adquirir características distintas adicionais.

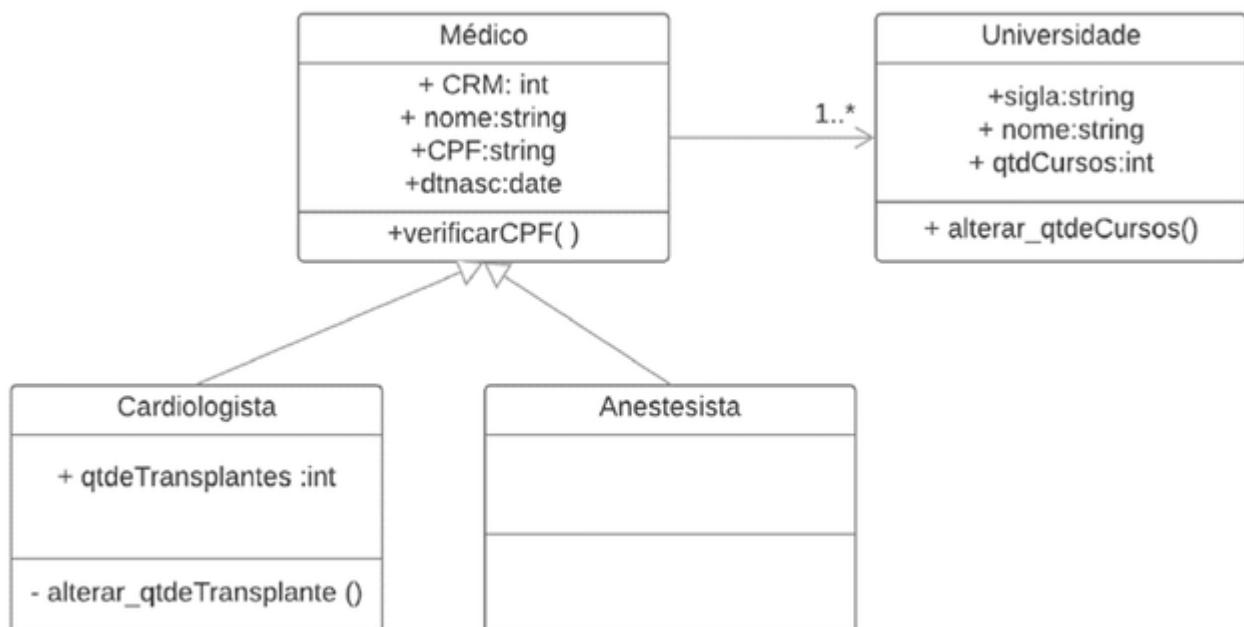


Figura 3 | Exemplo de hierarquia de herança entre as classes.

Na Figura 3, a classe "Médico" atua como a classe base e suas duas subclasses ("Cardiologista" e "Anestesista") recebem a herança dos atributos e métodos da classe mãe. Esse fato implica que os atributos como CRM, nome, CPF e data de nascimento não precisam ser especificados nas subclasses, uma vez que são automaticamente herdados da classe principal. No caso da classe "Anestesista", ela não apresenta atributos ou métodos adicionais além dos herdados, uma vez que é possível defini-los conforme necessário (contudo, a classe deve ser criada no diagrama).

A utilização da herança entre classes é uma característica da programação orientada a objetos que pode ser aplicada na modelagem de dados relacionais com o propósito de aprimorar o suporte a relacionamentos entre estruturas de classes (superclasses e subclasses). O conceito fundamental por trás da herança é a capacidade de criar subclasses que podem herdar as propriedades da classe principal. Essa analogia é igualmente aplicada aos modelos de entidade-relacionamento, e o processo pelo qual várias entidades (ou tabelas) são agrupadas em uma entidade genérica, chamamos de generalização. Já a especialização corresponde ao processo inverso, no qual novas entidades são criadas com atributos que acrescentam detalhes à entidade genérica.

Silberschatz, Korth e Sudarshan (2020) observam que, em um diagrama de entidade-relacionamento, a generalização e a especialização representam um tipo de relação entre entidades no qual uma engloba a outra. Isso significa que uma entidade superior contém um ou mais conjuntos de entidades inferiores.

Siga em Frente...

Um exemplo prático da aplicação da UML em um diagrama de entidade-relacionamento envolve a implementação de conceitos de generalização e especialização, que basicamente consistem em empregar o princípio da herança no modelo. Consideremos o cenário de uma universidade que necessita armazenar informações sobre as disciplinas ministradas por professores e as disciplinas cursadas por alunos. Além delas, outras informações relacionadas às tabelas precisam ser registradas:

- Na tabela "Funcionário", temos dados como nome, endereço, RG, CPF, data de nascimento, data de admissão, data de demissão, salário, nome da mãe e nome do pai.
- Na tabela "Professor", encontramos informações como nome, endereço, RG, CPF, data de nascimento, nome da mãe, nome do pai, valor da hora/aula e quantidade de horas/aula.
- A tabela "Aluno" contém dados como nome, endereço, RG, CPF, data de nascimento, data de entrada na faculdade, data de formatura, salário, nome da mãe e nome do pai.

Nota-se que há atributos comuns entre as tabelas, como nome, endereço, RG, CPF, data de nascimento, nome da mãe e nome do pai. Para evitar tal redundância, é possível criar uma entidade denominada "Pessoa" e incluir nela os atributos compartilhados pelas demais tabelas. Na Figura 4, apresentamos um exemplo de como esse modelo conceitual será organizado.

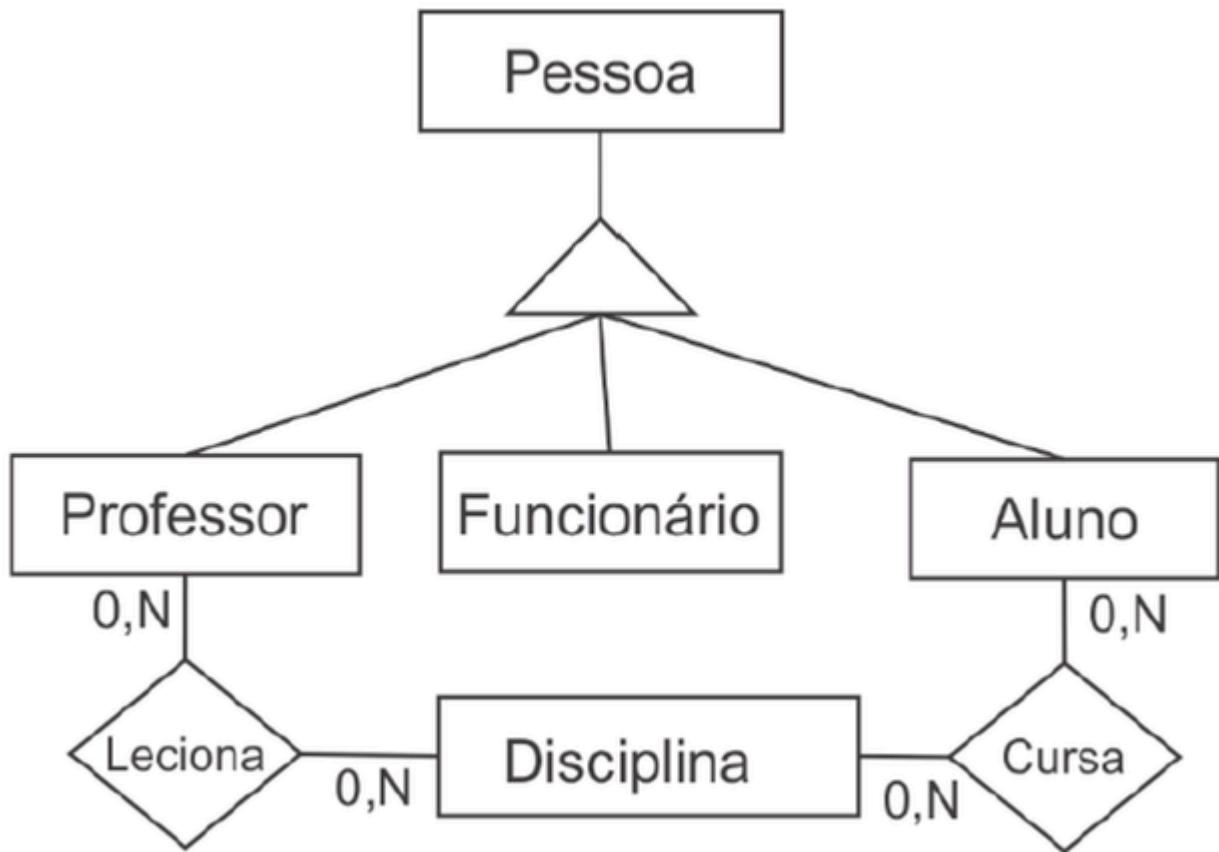


Figura 4 | Exemplo do modelo conceitual com generalização e especialização.

Silberschatz, Korth e Sudarshan (2020) explicam que o diagrama de classes pode servir como uma representação gráfica para construir o modelo lógico de um banco de dados. A Figura 5, assim, ilustra como o diagrama de entidade-relacionamento (DER) será estruturado utilizando a notação UML.

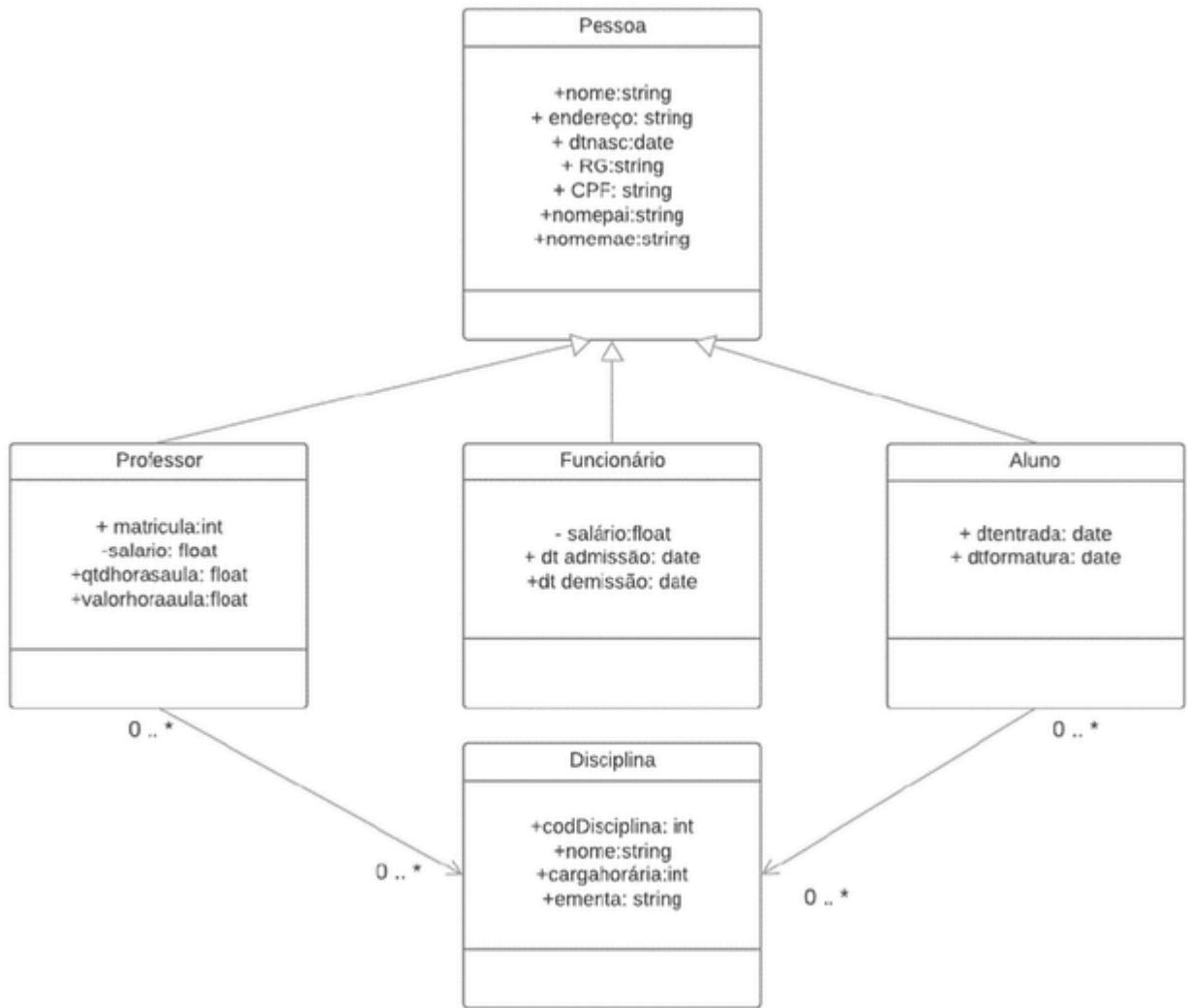


Figura 5 | Exemplo do DER com notação UML.

A modelagem de dados incorporou os princípios da programação orientada a objetos como uma maneira de se manter atualizada com os conceitos mais recentes em programação. Nesta seção, exploramos a aplicação de conceitos como herança, incluindo generalização e especialização, no contexto do modelo de entidade-relacionamento. Como uma atividade reflexiva de aprendizado, recomendamos uma revisão das unidades anteriores em busca de diagramas que possam acomodar esse novo tipo de relacionamento entre tabelas

O aspecto crucial consiste em reconhecer que a presença frequente de atributos repetidos é o indicativo mais evidente de que é necessário aplicar os conceitos de generalização e especialização entre as entidades.

Vamos Exercitar?

Nesta aula, apresentamos uma nova abordagem para estabelecer relacionamentos, denominada generalização e especialização, um processo análogo ao conceito de herança na programação orientada a objetos. Vimos, igualmente, que a notação UML pode ser empregada como uma ferramenta auxiliar na modelagem de bancos de dados. Uma prática comum nas empresas é criar o modelo lógico com essa notação, especialmente quando alguns softwares utilizam esse padrão em seus projetos. Ademais, constatamos que o diagrama de entidade-relacionamento guarda notável semelhança com o diagrama de classes, e que a transição entre ambos pode ser facilmente realizada.

Saiba mais

Para saber mais sobre a modelagem de dados utilizando o UML, recomendamos a leitura do artigo de Tanaka *et al.*, o qual apresenta a modelagem de uma aplicação web demonstrando como as classes persistentes definidas na UML podem ser traduzidas no modelo relacional de um banco de dados.

TANAKA, S. A. *et al.* [Modelagem de uma aplicação web a partir de um framework de agenda de tarefas](#). In: I CONGRESSO SUL CATARINENSE DE COMPUTAÇÃO, 1., 2005, Criciúma. **Anais** [...]. Criciúma: UNESC, 2005.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

FOWLER, M. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3. ed. Porto Alegre: Bookman, 2005.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados**: estudos de casos abrangentes da concepção lógica à implementação. 1. ed. São Paulo: Érica, 2019.

Aula 4

Ferramentas CASEs de modelagem de dados

Ferramentas CASEs de modelagem de dados



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula vamos introduzir a utilização de ferramentas para auxílio no processo de modelagem de dados. Você conhecerá ferramentas tipo CASE, cujo objetivo é a criação do modelo gráfico do diagrama entidade-relacionamento estudado até aqui. Na maior parte dos empreendimentos de banco de dados destinados a aplicativos de médio e grande porte, diversas equipes de analistas e desenvolvedores podem colaborar no projeto como um todo ou em suas partes individuais. A tarefa de conceber e criar um banco de dados exige rigor e método. É essencial estabelecer diretrizes de desenvolvimento, ou seja, definir regras para garantir que todos os participantes adotem uma abordagem consistente na modelagem. Nesse cenário é que a ferramenta CASE entra em cena. Então, bons estudos!

Vamos Começar!

As ferramentas CASE (*Computer Aided Software Engineering* ou, em português, engenharia de software auxiliada por computador) são recursos que oferecem uma variedade de serviços para facilitar o processo de desenvolvimento de software, podendo reduzir o tempo necessário para criar o programa planejado. Segundo Elmasri e Navathe (2018), as primeiras ferramentas CASE surgiram nos anos 80 e eram categorizadas em dois tipos:

- **Lower CASE:** fornece suporte nas fases de análise e projeto de sistemas.
- **Upper CASE:** oferece assistência nas fases de construção e análise de sistemas.

Atualmente, as ferramentas CASE são classificadas como *Integrated CASE*, combinando os recursos do *Lower CASE* e *Upper CASE*, abrangendo praticamente todas as etapas de um projeto de sistemas. Independentemente de seu tipo, todas as ferramentas CASE compartilham a capacidade de representar graficamente elementos do projeto, como diagramas de entidade-relacionamento, diagramas de classes, casos de uso, entre outros. Existem diversas ferramentas CASE disponíveis, desde opções gratuitas (com funcionalidades básicas) até proprietárias (pagas), que oferecem uma ampla gama de recursos.

As ferramentas CASE são empregadas com o propósito de automatizar diversas atividades, tais como:

- **Produção de código:** elas têm a capacidade de gerar automaticamente o código com base no diagrama gráfico.
- **Documentação automática:** facilitam a padronização dos processos de documentação. **Realização de testes:** permitem a validação das especificações formais de desenvolvimento.
- **Elaboração de relatórios:** possibilitam o acompanhamento do planejamento e da gestão do projeto de forma eficiente.

Nas atividades de engenharia de software, as ferramentas CASE desempenham um papel crucial, fornecendo assistência em todos os procedimentos. Exemplos incluem Rational Rose, Astah, Genexus, Multicase, Clarify, entre outras. Embora elas possam ser usadas para modelagem de bancos de dados, existem ferramentas CASE específicas para a criação de diagramas de entidade-relacionamento, como Oracle Designer, DBDesigner, Erwin, Embarcadero e MySQL Workbench. Uma tendência notável são as ferramentas CASE online, como Draw.IO ou Lucidchart. A maioria dessas ferramentas, incluindo as versões online, oferece edições gratuitas com algumas limitações, adequadas para diagramas de menor complexidade. Para aqueles interessados, é possível adquirir versões oficiais com funcionalidades expandidas.

As ferramentas CASE para bancos de dados apresentam as seguintes características, como descrevem Silberschatz, Korth e Sudarshan (2020):

- Suporte para elaboração de representações gráficas.
- Utilização de alguma forma de notação para modelagem de bancos de dados.
- Capacidade de gerar scripts SQL (*Structured Query Language - Linguagem de Consulta Estruturada*).
- *Forward engineer*: permite, a partir do modelo de diagrama de entidade-relacionamento (DER), automatizar a conexão com o banco de dados e gerar automaticamente o modelo físico.
- *Reverse engineer*: possibilita criar o modelo gráfico (DER) do banco de dados a partir do modelo físico já existente.
- **Geração de documentação**: à medida que os atributos são criados nas tabelas, a ferramenta CASE cria automaticamente um dicionário de dados.

Astah é uma ferramenta CASE que se concentra na criação de diagramas UML e está disponível em diversas versões, tais como a *Community*, gratuita para projetos UML (com algumas

limitações), e a *Professional*, completa e paga (ou disponível de forma *trial*). É altamente adequada para desenvolvedores Java, uma vez que é capaz de gerar scripts em Java, agilizando significativamente o processo de desenvolvimento de software. Na versão profissional, oferece a capacidade de criar diagramas de entidade-relacionamento, empregando a notação IDEF1X, como ilustrado na Figura 1, em que a bola preta representa a relação "N de muitos". Para obtê-la, você pode fazer o download diretamente no [site oficial](#).

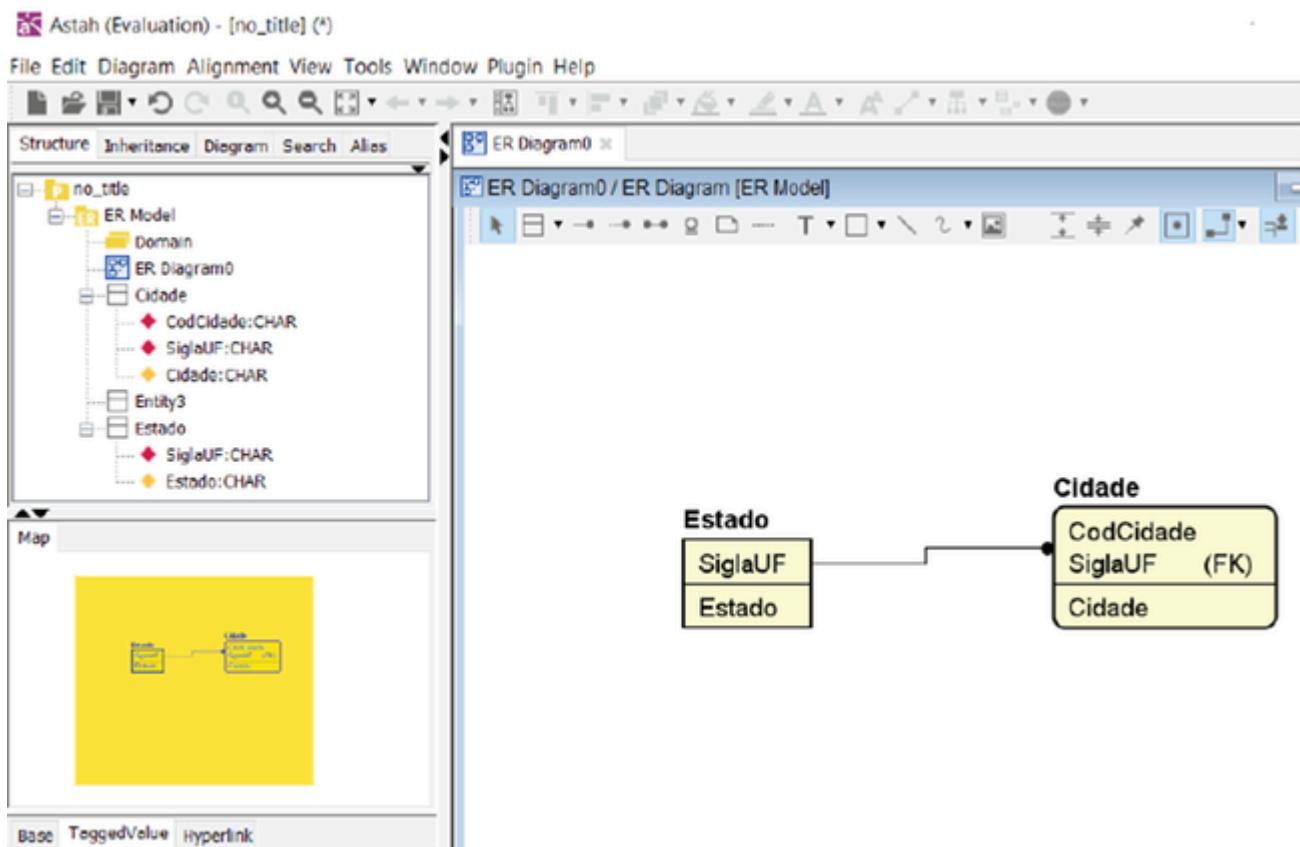


Figura 1 | Exemplo ferramenta CASE Astah. Fonte: captura de tela do Astah.

Uma das vantagens oferecidas pela ferramenta Astah é a capacidade de criar o dicionário de dados de maneira automática. Basta exportar os dados para a ferramenta Microsoft Excel, e o dicionário de dados será gerado, contendo todas as informações das tabelas, como ilustra a Figura 2. Para realizar essa exportação, você pode seguir os seguintes passos: vá ao menu no Astah, selecione "Tools", accese a opção "ER Diagram", e escolha a quarta opção, "Export Entity Definition Report". Em seguida, selecione as tabelas das quais deseja criar o dicionário de dados e salve o arquivo com um nome específico.

H1	A B C D E F G H I J K L M N O P Q R S U V W X Y Z AAAA AD AE AF AG AH AI AJ AJAN									
1 Entity	Logic Name Cidade Physical Na									
3 Kind										
4	Definition									
5										
6										
7										
8 No. Attribute(Logic)	Attribute(Physical)	PK	FK	AK	IE	NN	Data Type	Length&Precision	Initialvalue	Definition
9 1 CodCidade			Y			Y	CHAR	40		Chave Primária da Tabela
10 2 SiglaUF	Sigla		Y	Y		Y	CHAR	2		Chave Estrangeira da Tabela Estado
11 3 Cidade							CHAR	50		Nome da Cidade
12										

Figura 2 | Exemplo dicionário de dados gerado no Astah. Fonte: captura de tela do Microsoft Office Excel.

Siga em Frente...

O MySQL Workbench® pertence à empresa Oracle e é uma ferramenta CASE gratuita especializada em gerar *scripts* destinados ao sistema de gerenciamento de banco de dados MySQL. Sua ênfase está na modelagem física do banco de dados, proporcionando uma aceleração significativa no processo de criação da base de dados. Para obter essa ferramenta, você pode acessar o [site oficial da empresa](#). Note que, na Figura 3, o relacionamento é representado utilizando a notação pé de galinha.

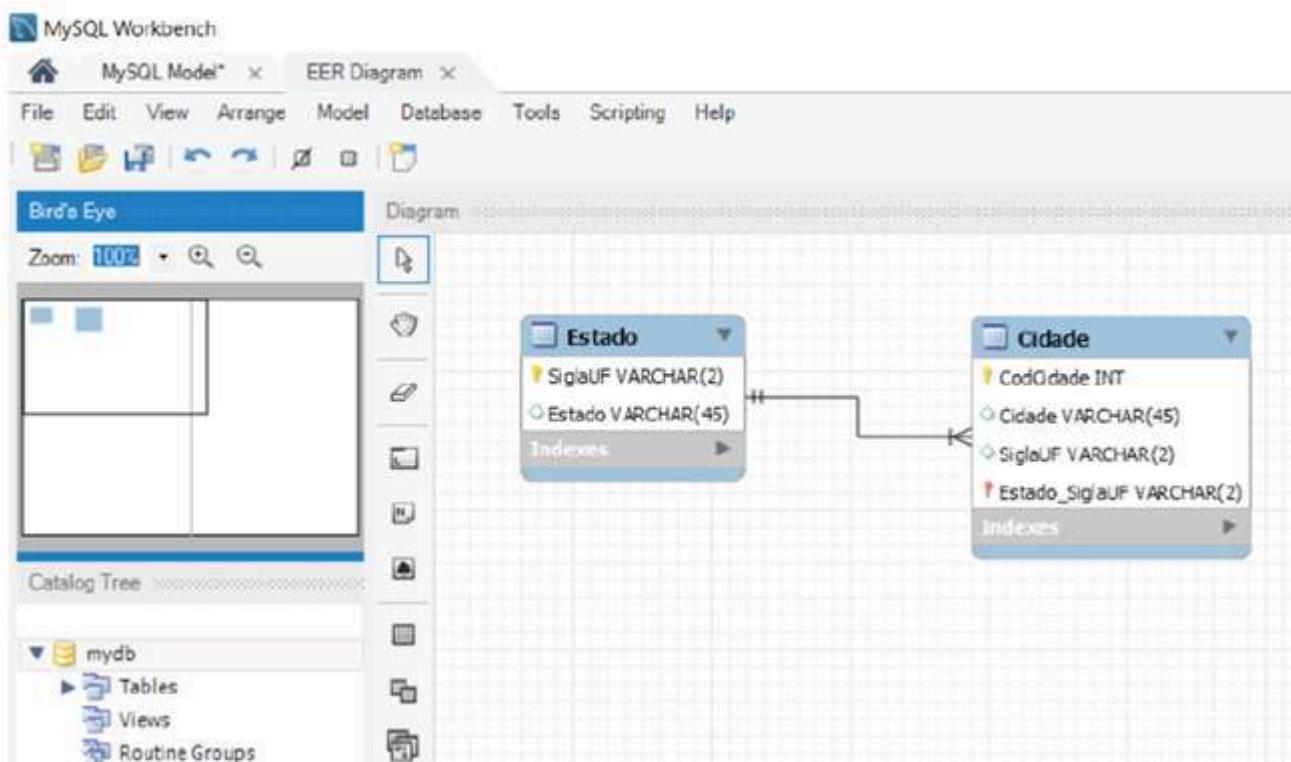


Figura 3 | Exemplo ferramenta CASE MySQL Workbench. Fonte: captura de tela do MySQL.

Uma alternativa para criar modelos gráficos de desenvolvimento de software é utilizar ferramentas online. O Draw.IO é uma opção simples que requer apenas uma conexão com a Internet e está disponível em seu [site oficial](#). Essa ferramenta oferece uma variedade significativa de modelos de exemplo para auxiliar na modelagem. Na Figura 4, como vemos, foi realizada uma modelagem com a notação de setas.

A utilização do Draw.IO é bastante intuitiva. À esquerda, você encontra categorias de diagramas e pode selecionar a categoria "entidade-relacionamento" para acessar as opções de tabelas, e adicioná-las à área de trabalho. Os campos podem ser editados diretamente na tabela adicionada. Para estabelecer relações entre as tabelas, escolha as opções de setas e conecte a chave primária diretamente à chave estrangeira.

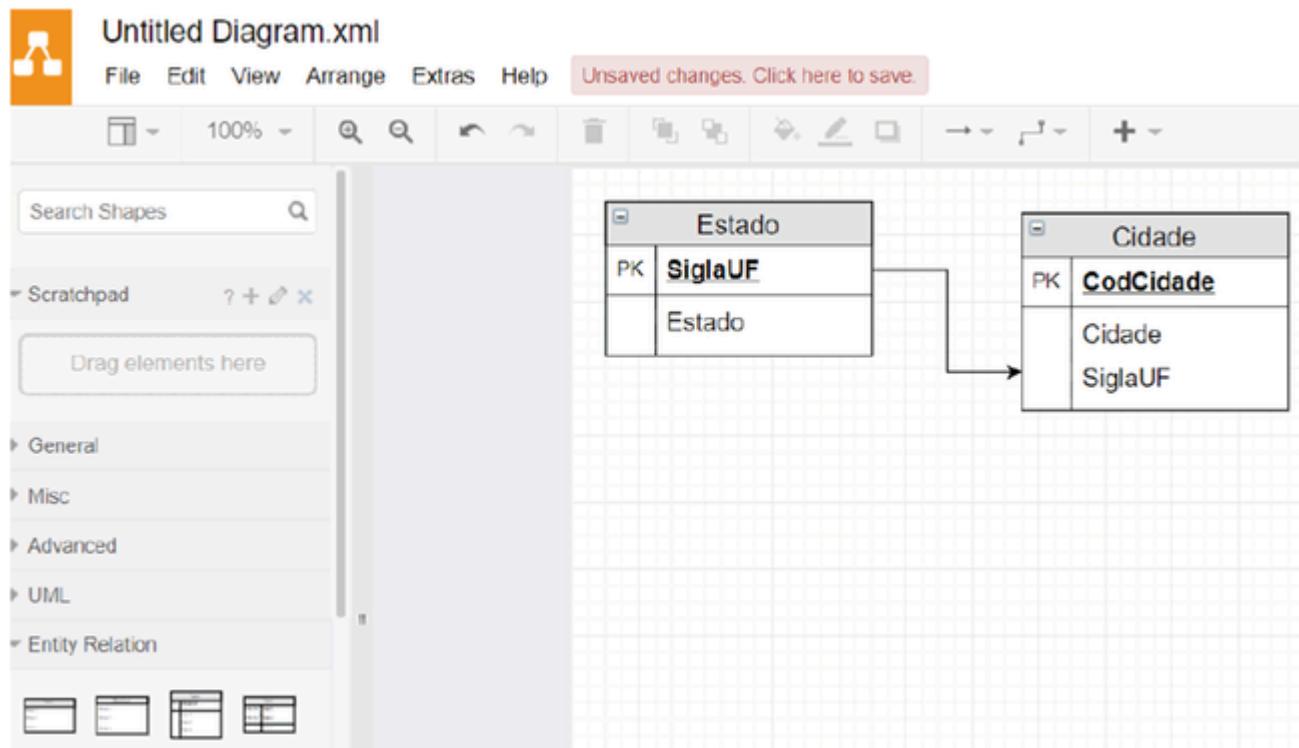


Figura 4 | Exemplo ferramenta CASE online Draw.IO. Fonte: captura de tela do Draw.IO.

Outra opção de ferramenta CASE disponível online é o Lucidchart, que pode ser acessado também em seu [site oficial](#). No site, é possível criar um modelo gratuitamente com até 60 objetos. Para projetos que excedem esse limite, é necessário adquirir uma versão paga. Essa ferramenta é particularmente adequada para projetos de pequeno e médio porte. Um exemplo ilustrativo encontra-se na Figura 5, na qual a notação pé de galinha foi aplicada.

Dentro da ferramenta, você pode fazer uso dos recursos disponíveis nas categorias padrão e entidade-relacionamentos. Ao trabalhar com a categoria de entidade-relacionamentos, basta selecionar o modelo de tabela desejado e inserir diretamente os campos em sua representação

gráfica. Para estabelecer relações entre as tabelas, utilize a ferramenta "seta" na categoria, sempre conectando a chave primária à chave estrangeira correspondente na outra tabela.

Uma das características notáveis dessa ferramenta é a capacidade de criar *scripts* para os seguintes sistemas de gerenciamento de banco de dados: MySQL, PostgreSQL, SQL Server e Oracle, conforme é observamos na Figura 6. Para gerar esse *script*, basta acessar a opção "Exportar" no menu localizado à esquerda do software, dentro da categoria de entidade-relacionamentos. Quando essa opção for selecionada, você terá acesso à tela apresentada na Figura 6, na qual poderá escolher qual banco de dados deseja utilizar para gerar o *script*. Além disso, é possível exportar o diagrama de entidade-relacionamento (DER) como uma imagem, selecionando "Arquivo" → "Baixar como" e escolhendo a opção desejada.

Exporte Para SQL X

Qual sistema de gerenciamento de banco de dados (DBMS) você está usando?

Vamos alterar a sintaxe dos nossos comandos SQL para corresponder ao seu DBMS.

MySQL
 PostgreSQL
 SQL Server
 Oracle

Exporte para SQL

Copie estes comandos para aplicá-los em seu próprio banco de dados. Antes de usar os comandos gerados, pode ser necessário adicionar tipos de dados, índices e chaves estrangeiras.

Copiar para a área de transferência

```
CREATE TABLE [Cidade] (
    [CodCidade] <type>,
    [Cidade] <type>,
    [SiglaUF] <type>,
    PRIMARY KEY ([CodCidade])
);

CREATE INDEX [FK] ON [Cidade] ([SiglaUF]);

CREATE TABLE [Estado] (
    [SiglaUF] <type>,
```

Concluir

Figura 6 | Exemplo exportando para SQL no Lucidchart. Fonte: captura de tela do Ludichart.

Um diagrama de entidade-relacionamento (DER) costuma envolver várias entidades, e o uso de uma ferramenta CASE para gerar as tabelas e estabelecer os relacionamentos simplifica consideravelmente as tarefas do desenvolvedor.

Vamos Exercitar?

Nesta aula, vimos que a utilização de ferramentas CASE na criação de modelos gráficos de bancos de dados é primordial no processo de desenvolvimento de qualquer software. Esse fato é particularmente relevante na apresentação ao cliente, por exemplo, uma vez que esses diagramas, quando bem elaborados, geralmente causam uma impressão muito favorável. Para a equipe de desenvolvimento, a vantagem reside na capacidade de facilitar a comunicação (todos podem compreender o que está sendo modelado) e acelerar a geração dos *scripts* necessários para criar fisicamente o banco de dados em um sistema de gerenciamento de banco de dados (SGBD). Conforme avançamos para a nossa última aula sobre este tema, fica evidente que não basta ter apenas um modelo gráfico atraente; ele deve ser completamente preciso e coerente com os padrões do mercado.

Saiba mais

Para saber mais sobre a modelagem de dados utilizando o UML com ferramentas CASE, recomendamos a leitura do artigo Análise de ferramentas CASE quanto às boas práticas de modelagem de software com UML.

CUNHA, W. S.; COSTA, H.; PARREIRA JR, P. A. [Análise de ferramentas CASE quanto às boas práticas de modelagem de software com UML](#). In: XV SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 15., 2016, Maceió. *Anais* [...]. Porto Alegre: SBConOpenLib, 2016, p. 51-63.

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados**: estudos de casos abrangentes da concepção lógica à implementação. 1. ed. São Paulo: Érica, 2019.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Chegada

Olá estudante! Para desenvolver a competência desta unidade, isto é, compreender todo o processo e saber implementar diagramas de entidade-relacionamento a partir de ferramentas especializadas para a criação de banco de dados, é necessário que você leia com atenção e profundidade as referências citadas nas aulas e também consuma conteúdos relacionados ao tema além deste material. Estudamos diversos tipos de notações e formas de abordagens do DER e aprendemos como implementá-los utilizando ferramentas do tipo CASE. Vimos, também, que a depender do grau de dificuldade da modelagem de dados, o tipo de notação para exemplificar o DER proposto pode ser melhor do que outros. Lembre-se de que um DER bem-feito é aquele que é coerente e preciso; não priorize a estética, mas sim a qualidade e integridade da modelagem do problema. Com os avanços do nosso estudo sobre MER, estamos, finalmente, chegando à etapa final.

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Os habitantes de sua vizinhança possuem uma horta comunitária em um terreno fornecido pela administração municipal. Cada residente que demonstrou interesse real em cultivar hortaliças

recebeu um terreno. Os tamanhos dos lotes variam: famílias maiores recebem espaços maiores, embora haja flexibilidade para que um morador possa optar por um lote menor, talvez apenas para o cultivo de temperos. Cada morador tem a liberdade de plantar o que preferir, contanto que o item seja registrado e aprovado, um controle necessário para evitar problemas futuros. A única regra rígida é a proibição absoluta de pesticidas químicos.

A equipe encarregada de gerenciar e fiscalizar os lotes requer um banco de dados que possa:

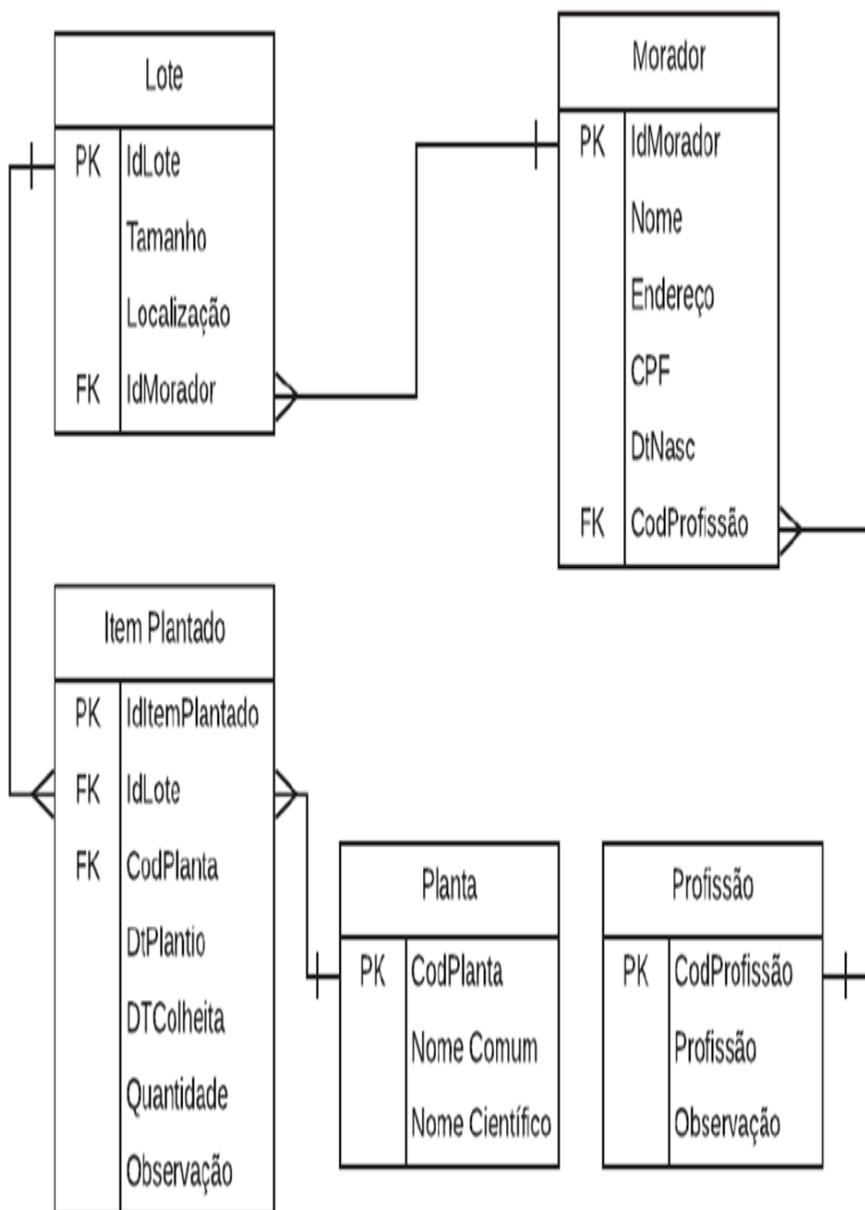
- Identificar o morador responsável por cada lote, considerando que um morador pode ser responsável por mais de um lote. É relevante, também, conhecer a profissão de cada pessoa, pois essa informação pode ser útil em situações em que um morador necessite da ajuda de alguém com uma habilidade específica.
- Registrar as informações assim que o residente fizer plantações no lote, incluindo o cadastro de cada item de hortaliça cultivado.
- Manter um registro das datas de plantio e colheita, contribuindo para a manutenção de estatísticas para a horta.

Como você pode colaborar na criação de um modelo de banco de dados para essa situação?

Algumas perguntas para reflexão:

- Quais são os benefícios da utilização de herança (generalização e especialização) em um diagrama de entidade-relacionamentos? Como podemos detectar essa necessidade em nossos modelos?
- Qual é a diferença entre as estratégias *top-down* e *bottom-up* em um DER?
- Qual é a real importância, dentro da modelagem de banco de dados, de se construir um dicionário de dados?

Para resolver essa situação-problema, devemos nos atentar às solicitações da equipe que gerencia a horta. Com essas informações em mãos, já podemos identificar as seguintes tabelas: Morador, Profissão, Lote, Plantas e Itens plantados. E podemos utilizar uma ferramenta CASE para criar o DER. Observe, na figura abaixo, o DER elaborado na ferramenta CASE online Lucidchart.



Lembre-se de que o PK representa a chave primária e o FK representa a chave estrangeira da tabela.

Caro estudante, a seguir apresentamos um infográfico contendo um resumo dos principais temas abordados nesta unidade de forma simples e objetiva. Reserve um tempo para leitura desse material e reflita.

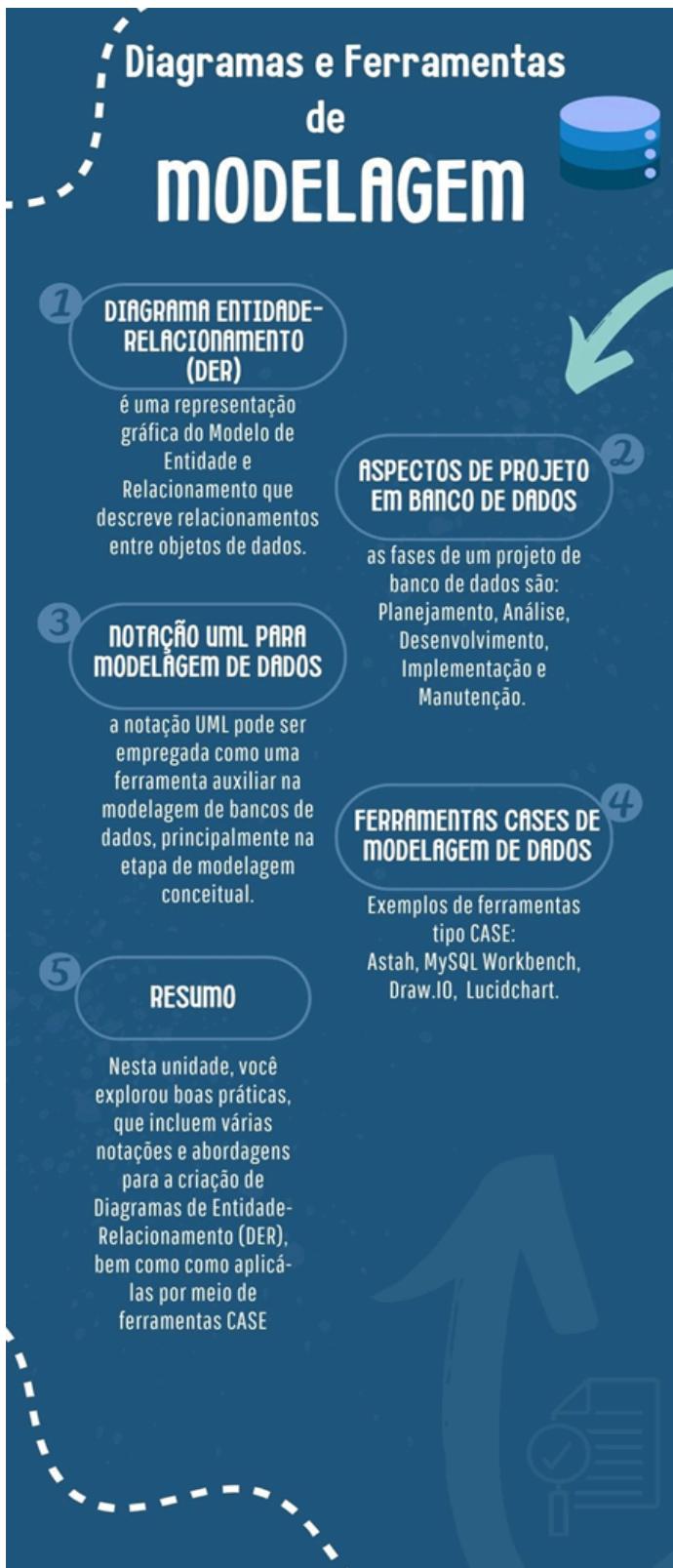


Figura | Diagramas e ferramentas de modelagem.

- ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.
- ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.
- SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.
- SORDI, J. O. de. **Modelagem de dados**: estudos de casos abrangentes da concepção lógica à implementação. 1. ed. São Paulo: Érica, 2019.

Unidade 4

Normalização de Dados

Aula 1

Normalização de dados

Normalização de dados

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Em um projeto de banco de dados, é imprescindível seguir um princípio fundamental: evitar a combinação de informações distintas dentro de uma única tabela. Deve-se combater com firmeza dois elementos: a redundância de dados e a discrepância de informações.

Para compreender tais diretrivas, nesta seção, exploraremos os conceitos relacionados à normalização de dados e à dependência funcional. Um especialista em tecnologia da informação deve estar familiarizado com eles e, além disso, deve saber aplicá-los ativamente ao criar estruturas de bancos de dados.

Portanto, é crucial que você tenha uma compreensão abrangente dos diversos setores de mercado. É também certo que as necessidades do cliente nem sempre estão claramente definidas, e que as entidades não são obrigatoriamente estabelecidas por meio de um estudo de caso completo. Nesse contexto, você, como analista, precisará visitar a empresa em questão e conduzir entrevistas com vários membros da equipe. Além disso, é de grande importância coletar cópias de documentos em uso na empresa, tais como faturas, pedidos de compra, registros de estoque, formulários de admissão de funcionários, entre outros. Esses documentos contêm informações valiosas que frequentemente passam despercebidas, inclusive pelo próprio cliente que solicita o desenvolvimento de software. É preciso analisar a fundo, assim, o conceito de normalização de dados e seus objetivos em modelagem de dados. Bons estudos!

Vamos Começar!

A essência da modelagem de dados reside na otimização de procedimentos. Esse fato implica que há sempre espaço para aprimorar o que já foi realizado. Mas uma vez que tenhamos elaborado o diagrama entidade-relacionamento (DER), podemos simplesmente implementá-lo em um sistema de gerenciamento de banco de dados (SGBD)? Não. A abordagem correta é revisá-lo minuciosamente, identificar imperfeições e aprimorar o projeto, antecipando eventuais problemas.

O principal desafio em bancos de dados é, frequentemente, a redundância, já que pode acarretar consequências significativas que podem passar despercebidas até que o banco de dados esteja em uso pela organização. O dano mais crítico causado pela redundância é a repetição da mesma informação em diversas tabelas, o que, além de criar duplicidade, pode levar a erros em relatórios.

Segundo Alves (2020), a meta mais importante a ser alcançada em um projeto de banco de dados é obter um resultado altamente preciso na representação dos dados, incluindo os relacionamentos entre eles e as restrições que devem ser obedecidas. Para atingir esse resultado, além da modelagem entidade-relacionamento, podemos empregar uma técnica chamada normalização de dados. Através dela, realizamos uma análise minuciosa dos relacionamentos entre os atributos dentro de uma entidade (e não entre entidades), aplicando uma série de testes conhecidos como formas normais.

O autor (2020) ainda destaca que a normalização, que pode ser aplicada em qualquer fase do projeto de banco de dados, tem como objetivo criar um conjunto de entidades que satisfaçam os requisitos do projeto, respeitando as seguintes características:

- Mínimo de atributos nas entidades.

- Atributos na entidade com relações estreitas.
- Redução máxima de redundância de dados, o que implica na eliminação de atributos que desempenham a mesma função de armazenar informações idênticas, com exceção das chaves estrangeiras.

Já para Machado (2020), o propósito da normalização é evitar as potenciais falhas no projeto do banco de dados e eliminar a "mescla de tópicos" e as redundâncias de informações desnecessárias. Uma diretriz fundamental durante o planejamento de um banco de dados orientado ao modelo relacional é evitar a combinação de diferentes tópicos em uma única tabela. O procedimento de normalização impõe um conjunto de regulamentos às tabelas de um banco de dados com o objetivo de avaliar se estão devidamente estruturadas. Embora existam cinco formas normais, na prática, costumamos aplicar principalmente três delas. A criação das tabelas de um banco de dados relacional é derivada de um modelo entidade-relacionamento (MER), e frequentemente, durante essa transição, surgem desafios relacionados ao desempenho, à integridade e à manutenção dos dados.

Normalmente, após a implementação das formas normais, algumas tabelas acabam sendo subdivididas em duas ou mais, resultando em um número maior do que inicialmente previsto. Esse processo simplifica os atributos de cada tabela e contribui significativamente para a estabilidade do modelo de dados, reduzindo as demandas de manutenção de forma considerável.

Alves (2020) destaca um conceito importante para se considerar no processo de normalização de dados: a dependência funcional. A dependência funcional possui grande relevância em bancos de dados relacionais, estabelecendo uma restrição entre dois conjuntos de atributos pertencentes à mesma entidade ou relação. Ela descreve a relação entre esses atributos, ou seja, em que medida um atributo depende de outro para a sua existência.

A dependência funcional é frequentemente representada pela notação $X \rightarrow Y$, em que X e Y designam subconjuntos de atributos dentro de uma relação específica. Essa notação estabelece uma restrição que sugere que um elemento Y em um registro depende de um valor presente no conjunto X, ou seja, é por ele determinado. De forma recíproca, os valores em X exercem influência exclusiva sobre os valores em Y. Em resumo, Y é funcionalmente dependente de X.

Siga em Frente...

Para ilustrar esse cenário, considere o exemplo de uma relação de livros cujo esquema relacional é o seguinte:

```
LIVROS(CodigoISBN, Titulo, Area, Formato, TipoEncadernacao, NumeroPaginas, Peso,  
ValorCusto, ValorVenda, NumeroEdicao, AnoEdicao, NumeroReimpressao, NumeroContrato,  
EstoqueMinimo, EstoqueMaximo, EstoqueAtual, DataUltimaEntrada)
```

Neste exemplo, o valor do atributo **Titulo** pode ser recuperado apenas sabendo-se o valor do atributo **CodigoISBN**. Dessa forma, o atributo **CodigoISBN** determina o valor único de **Titulo**, ou seja, não temos títulos de livros diferentes com o mesmo código ISBN. Tecnicamente falando, **Título** é dependente funcional de **CodigoISBN**, como é possível ver na Figura 1.

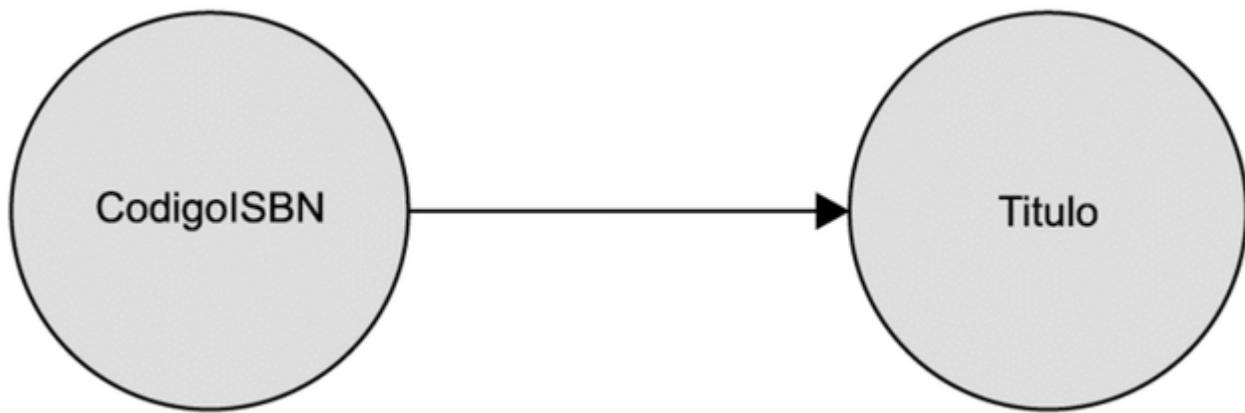


Figura 1 | Diagrama da dependência funcional de **Titulo** em relação a **Codigo ISBN**. Fonte: Alves (2020).

O atributo do qual parte a seta nesse tipo de diagrama é denominado determinante. Seguindo o raciocínio, Alves (2020) destaca que, como cada livro está vinculado a uma área, o atributo **CodigoISBN** também pode determinar o valor do atributo **Area**. Porém, neste caso, se procurarmos na relação por todas as tuplas que possuem o mesmo valor no atributo **Area**, encontraremos várias instâncias, uma vez que podemos ter vários livros pertencentes à mesma área. Em outras palavras, **CodigoISBN** determina um único valor para **Area**, mas **Area** pode determinar mais de um valor para **CodigoISBN**. O relacionamento entre **CodigoISBN** e **Area** é de 1:1 (um para um), enquanto o oposto, entre **Area** para **CodigoISBN**, é de 1:M (um para muitos), como vemos na figura a seguir.

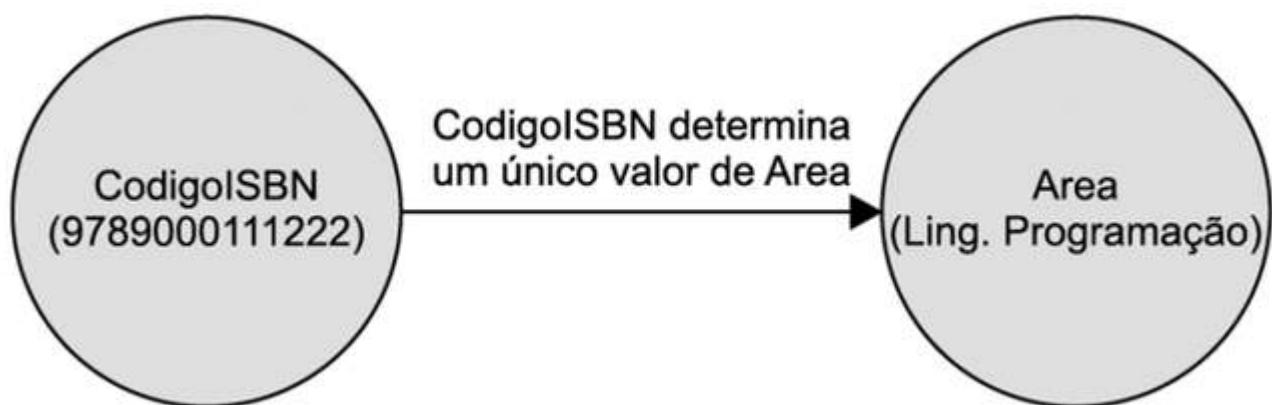


Figura 2 | Relacionamento entre atributos **CodigoISBN** e **Area** onde **Area** determina mais de um **CodigoISBN**. Fonte: Alves (2020).

Podemos categorizar a dependência funcional em três tipos distintos: total (ou completa), parcial e transitiva. Na primeira categoria, a dependência existe somente se a chave primária, composta

por diversos atributos, for capaz de determinar de forma única um atributo ou um conjunto de atributos. No nosso exemplo, o atributo **CodigoISBN** é a chave primária que determina o valor de todos os outros atributos na tupla.

Quando um atributo ou conjunto de atributos depende apenas de parte dos valores da chave primária, nos deparamos com uma dependência parcial. Em relação à dependência transitiva, ela ocorre quando um atributo ou conjunto de atributos depende de outro atributo ou conjunto de atributos que não fazem parte da chave primária.

É fundamental observar que apenas o valor do atributo **CodigoISBN** é necessário para determinar o título do livro, caracterizando, assim, uma dependência parcial. Agora, consideremos uma relação que abrange os itens de um pedido de venda com o seguinte esquema:

ITENS_PEDIDO(Numeropedido,CodigoISBN,Quantidade,ValorUnitario,ValorTotal,Desconto)
--

Neste caso, para sabermos a quantidade referente a um determinado livro dentro do pedido, não basta conhecermos apenas o número desse pedido. É necessário que também tenhamos o código ISBN do livro. Isso significa que o atributo **Quantidade** é dependente funcional da combinação de dois outros atributos, a saber, o **Numeropedido** e o **CodigoISBN**.

{Numeropedido,CodigoISBN} ↳ {Quantidade}
--

Por fim, Alves (2020) lembra que, neste caso, temos uma dependência funcional total, já que são necessários os dois atributos para identificar univocamente o valor de outro.

Em síntese, a normalização de tabelas constitui um conjunto de procedimentos destinados a identificar falhas em um projeto de banco de dados, detectando inconsistências que podem envolver dados duplicados ou dependências funcionais mal definidas ou mal organizadas. A ação de normalização consiste em decompor uma tabela em tabelas de menor complexidade, reduzindo gradualmente o número de redundâncias e dependências funcionais. É importante notar que o objetivo primordial da normalização não é a completa eliminação das inconsistências, mas sim o seu controle. Nesse sentido, a identificação das dependências funcionais nas tabelas representa o primeiro passo para reconhecer a necessidade de normalizar as tabelas em um banco de dados.

Vamos Exercitar?

Nesta aula, aprendemos que existe um outro processo de verificação de qualidade da modelagem de dados: a normalização das tabelas. Como desafio, você, enquanto proprietário de uma empresa, será responsável por atender às necessidades de um clube de futebol. Nesse contexto, será atribuída a tarefa de automatizar um formulário de registro de jogos que, até o

momento, era preenchido manualmente. A meta é informatizar esse formulário e armazenar as informações em um banco de dados.

Para solucionar esse problema, é crucial, inicialmente, analisar minuciosamente as informações fornecidas pelo formulário. Mas como identificar os campos das tabelas? Uma abordagem consiste em elaborar um inventário de todas as informações que podem ser inseridas, incluindo itens como número do jogo, data da partida, nome do oponente, local do jogo, entre outros. E no que se refere às tabelas, como podemos identificá-las? Nesse ponto, a intuição desempenha um papel relevante (considerando que ainda não discutimos as técnicas de normalização). Alguns exemplos de tabelas a serem consideradas incluem jogadores, cidades, estados etc. Utilize a teoria aprendida nesta aula e reflita sobre o caso. Lembre-se que a prática leva à perfeição; então, vamos exercitar?

Saiba mais

Para compreender melhor o assunto abordado nesta aula, recomendamos a leitura dos capítulos 9 em diante do livro *Banco de dados: teoria e desenvolvimento*, de William P. Alves. Disponível na Biblioteca Virtual.

ALVES, W. P. [Banco de dados: teoria e desenvolvimento](#). 2. ed. São Paulo: Érica, 2020.

Referências

ALVES, W. P. **Banco de dados: teoria e desenvolvimento**. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.

Aula 2

Formas Normais I

Formas normais I



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la? Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula iniciaremos o processo de normalização das tabelas de um banco de dados. Algumas das diretrizes iniciais da normalização já foram aplicadas no início da disciplina. Com elas, você terá as regras disponíveis para auxiliá-lo caso surjam dúvidas durante o processo de criação de tabelas e campos. Quando uma empresa investe na criação de um software, ela espera não apenas que ele funcione, mas também que forneça resultados rapidamente. As solicitações de consultas ao banco de dados são frequentes e constantemente surgem novas necessidades de consulta. Um banco de dados mal projetado pode resultar em longas esperas por resultados, e o pior cenário possível ocorre: erros duplicados e imprecisões.

A normalização de tabelas, assim, é um método para avaliar e corrigir a estrutura das tabelas com o propósito de reduzir redundâncias de dados, e, portanto, a probabilidade de erros e anomalias nas tabelas. Para Alves (2020), a normalização envolve um procedimento de aprimoramento do esquema do banco de dados que visa remover redundâncias entre as entidades, resolver questões de dependência parcial entre atributos e minimizar as ocorrências de problemas na inclusão, modificação e exclusão de dados. Aprofundemo-nos, então, nos conceitos de normalização de dados e suas formas normais. Bons estudos!

Vamos Começar!

O processo de normalização de dados, de acordo com Alves (2020), é dividido em múltiplas etapas, conhecidas como formas normais, que envolvem vários testes para garantir a

conformidade do esquema com as condições específicas de cada etapa. Com base nesses testes, as relações podem ser desmembradas em relações menores, conforme necessário.

As formas normais de uma relação indicam o nível de normalização atingido por ela. Existem, academicamente, cinco formas normais, embora as três primeiras sejam geralmente suficientes para definir adequadamente a estrutura do banco de dados. Ao final do processo de normalização, obtemos a resposta para uma das questões essenciais colocadas no início de um projeto: quantas tabelas são necessárias no banco de dados?

A aderência às regras de normalização possibilita a criação e a modificação de bancos de dados robustos e eficientes. Quando elas são seguidas cuidadosamente, o sistema como um todo (banco de dados e aplicativo) se torna altamente flexível, confiável e de fácil manutenção. Existem, pois, duas abordagens/metodologias que podem ser empregadas no processo de normalização de um banco de dados:

- **Abordagem de cima para baixo (*Top-Down*):** envolve a organização de atributos em relações que são definidas a partir do projeto conceitual. Uma análise é realizada nessas relações, descompondo-as em entidades e relacionamentos até que as propriedades desejadas para a implementação física sejam alcançadas.
- **Abordagem de baixo para cima (*Bottom-Up*):** é uma abordagem reversa à anterior, na qual os relacionamentos entre os atributos são o ponto de partida para o processo, e, igualmente, usados na construção das relações. Também é chamada de projeto por síntese.

Ted Codd originalmente propôs apenas três formas normais, denominadas primeira, segunda e terceira forma normal. Posteriormente, em conjunto com Raymond Boyce, introduziu uma nova definição chamada de forma normal de Boyce-Codd (ou FNBC). Todas essas formas normais se baseiam na dependência funcional entre os atributos de uma entidade no banco de dados e nas chaves primárias. Atualmente, são reconhecidas também a quarta (4FN) e a quinta forma normal (5FN), embora, na maioria dos casos, a normalização seja efetivamente realizada com base nas três primeiras.

O primeiro passo é garantir que não existam campos que contenham mais de um valor (campos não atômicos). Por exemplo, um campo destinado ao endereço deve conter apenas o nome do logradouro e, no máximo, o número do imóvel. Armazenar informações como bairro, cidade e estado violaria essa primeira regra.

Para atingir os objetivos da normalização, de acordo com Sordi (2019), as tabelas devem:

- Abordar exclusivamente um único tema; por exemplo, uma tabela contendo informações sobre medicamentos não deve conter detalhes sobre médicos.
- Evitar o armazenamento redundante do mesmo campo em múltiplas tabelas, reduzindo a necessidade de atualização de todas.
- Garantir que seus campos dependam exclusivamente de sua chave primária e não de outros campos.

- Ser projetadas de forma a evitar anomalias durante operações de inserção, atualização e exclusão, garantindo a integridade e a consistência dos dados. Por exemplo, na tabela de Clientes, em vez de permitir que os usuários informem a cidade de nascimento livremente, deve-se restringir a escolha entre opções previamente cadastradas ou oferecer a possibilidade de busca via CEP para obter o endereço completo.

Para aplicar as regras de normalização, é fundamental observar os atributos presentes nas tabelas, segundo Silberschatz, Korth e Sudarshan (2020):

- **Atributo simples ou atômico:** refere-se a atributos indivisíveis que possuem um único significado, como o RG ou o CPF de uma pessoa, e que não podem ser subdivididos em campos menores.
- **Atributo composto:** são atributos que podem ser desmembrados em partes menores, como o atributo de endereço, o qual pode ser dividido em rua, número, complemento e bairro.
- **Atributo monovalor:** denota atributos que têm apenas um valor válido na tabela, como a matrícula de um aluno, que não deve se repetir na mesma tabela.
- **Atributo multivvalorado:** é um tipo de atributo que pode conter múltiplas informações. Um exemplo típico é o atributo "telefone", que pode acomodar vários números diferentes.
- **Atributo derivado:** esse tipo de atributo possui um valor que é derivado de outra tabela ou de campos existentes. Por exemplo, a idade de um paciente pode ser calculada com base na data de nascimento e na data da consulta médica, fator especialmente relevante para um cardiologista.
- **Atributo chave:** é um atributo escolhido ou criado para identificar exclusivamente um registro (linha) na tabela.

A primeira forma normal (1FN) estabelece a regra fundamental de que uma tabela estará na 1FN se, e somente se, todos os seus atributos forem atômicos, ou seja, não contenham grupos repetitivos ou colunas com múltiplos valores. Para atender aos requisitos da 1FN, os seguintes passos devem ser seguidos:

- Identificação da chave primária da tabela.
- Identificação da coluna com dados repetidos.
- Remoção da coluna com dados repetidos.
- Criação de uma nova tabela para armazenar os dados repetidos.
- Estabelecimento de um relacionamento entre a tabela em processo de normalização e a nova tabela auxiliar.

O Quadro 1 exibe o exemplo de um quadro de funcionários que ainda não passou pelo processo de normalização e contém diversos campos.

Nome	Idade	Valor da Hora	Cidade	Departamento	Data de Admissão
Carlos Augusto	25	R\$ 18,54	Curitiba	Contabilidade	15/01/2018
Roberto César	19	R\$ 16,70	São Paulo	Produção	21/11/2017
Marta Maria	22	R\$ 20,15	Santo André	RH	03/04/2018
Olivia Costa	31	R\$ 21,19	Rio de Janeiro	Tecnologia da Informação	29/05/2018

Quadro 1 | Quadro não normalizado de funcionário.

Para que o Quadro "Funcionário" esteja em conformidade com a primeira forma normal (1FN), precisamos começar por examinar os campos existentes. Podemos questionar se algum desses campos pode servir como chave primária. No entanto, uma abordagem preferível é criar um novo campo para atuar como chave primária. Uma sugestão viável é utilizar um campo, por exemplo, "matrícula" ou "código do funcionário", como chave primária. A seguir, apresentamos a tabela "Funcionário" na forma textual:

Funcionário: (#matrículaFunc, nome, idade, data de admissão, valor da hora, cidade, departamento).

É possível melhorar este cenário identificando os campos "cidade" e "departamento" como candidatos à normalização do Quadro "Funcionário" para atender à 1FN. Para tanto, procedemos da seguinte maneira: primeiro, criamos uma tabela denominada "Cidade"; em seguida, inserimos uma chave estrangeira da tabela "Cidade" na tabela "Funcionário".

Cidade (#idCidade, Cidade).
 Funcionários (#matrículaFunc, nome, idade, valordahora, dtadmissão, Departamento, &idCidade).

matrículaFunc	Nome	Idade	Valor da Hora	idCidade	Departamento	Data de Admissão
123	Carlos Augusto	25	R\$ 18,54	12	Contabilidade	15/01/2018
456	Roberto César	19	R\$ 16,70	18	Produção	21/11/2017
789	Marta Maria	22	R\$ 20,15	19	RH	03/04/2018
270	Olivia Costa	31	R\$ 21,19	21	Tecnologia da Informação	29/05/2018

Quadro 2 | Funcionário na 1FN.

idCidade	Cidade
12	Curitiba
18	São Paulo
19	Santo André
21	Rio de Janeiro

Quadro 3 | Cidade na 1FN.

Observando novamente os campos do quadro, é possível notar a presença do campo "idade". Embora não seja incorreto armazenar esse dado, toda vez que um funcionário tiver uma alteração na idade, será necessário realizar uma atualização na tabela. Essa, de longe, não é uma solução prática, nem a mais adequada para um banco de dados. Como podemos resolver essa

questão? Uma alternativa é substituir o campo "idade" pelo campo "data de nascimento." Nesse caso, a tabela terá o seguinte formato:

Funcionários (#matrículaFunc, nome, dtNascimento, valordahora, dtadmissão, Departamento, &idCidade).

A segunda forma normal, denominada 2FN, segue a regra da seguinte forma: uma tabela estará em 2FN se, e somente se, estiver na 1FN e todas as colunas que não fazem parte da chave primária dependerem exclusivamente de toda a chave primária, e não apenas de parte dela. Para alcançar a 2FN, precisamos realizar as seguintes ações:

- Identificar as colunas que não têm dependência funcional com a chave primária da tabela.
- Remover essas colunas da tabela e criar uma nova tabela para armazenar esses dados.

Tomando como exemplo a tabela "Funcionário", temos:

Funcionários (#matrículaFunc, nome, idade, valordahora, dtadmissão, Departamento, &idCidade).

Para obtermos a 2FN, é necessário criar uma tabela chamada "Departamento" e inserir a chave estrangeira correspondente na tabela "Funcionário". O formato das tabelas ficará assim:

FuncionárioDepartamento (#codDepart, Departamento).
 Funcionário (#matrículaFunc, nome, dtNascimento, valordahora, dtadmissão, &codDepart, &idCidade).s (#matrículaFunc, nome, idade, valordahora, dtadmissão, Departamento, &idCidade).

matrículaFunc	Nome	dtNascimento	Valor da Hora	idCidade	codDepart	dtAdmissao
123	Carlos Augusto	02/01/1998	R\$ 18,54	12	13	15/01/2018
456	Roberto César	17/06/2004	R\$ 16,70	18	21	21/11/2017
789	Marta Maria	05/12/2002	R\$ 20,15	19	25	03/04/2018
270	Olivia Costa	20/04/1992	R\$ 21,19	21	16	29/05/2018

Quadro 4 | Funcionário na 2FN.

codDepart	Departamento
13	Contabilidade
21	Produção
25	RH
16	Tecnologia da Informação

Quadro 5 | Departamento.

Siga em Frente...

A Figura 1 ilustra o diagrama entidade-relacionamento obtido após a implementação da 1FN e 2FN. Anteriormente, tínhamos apenas uma tabela denominada "Funcionário". Agora, dispomos de três tabelas distintas.

MODELAGEM DE DADOS

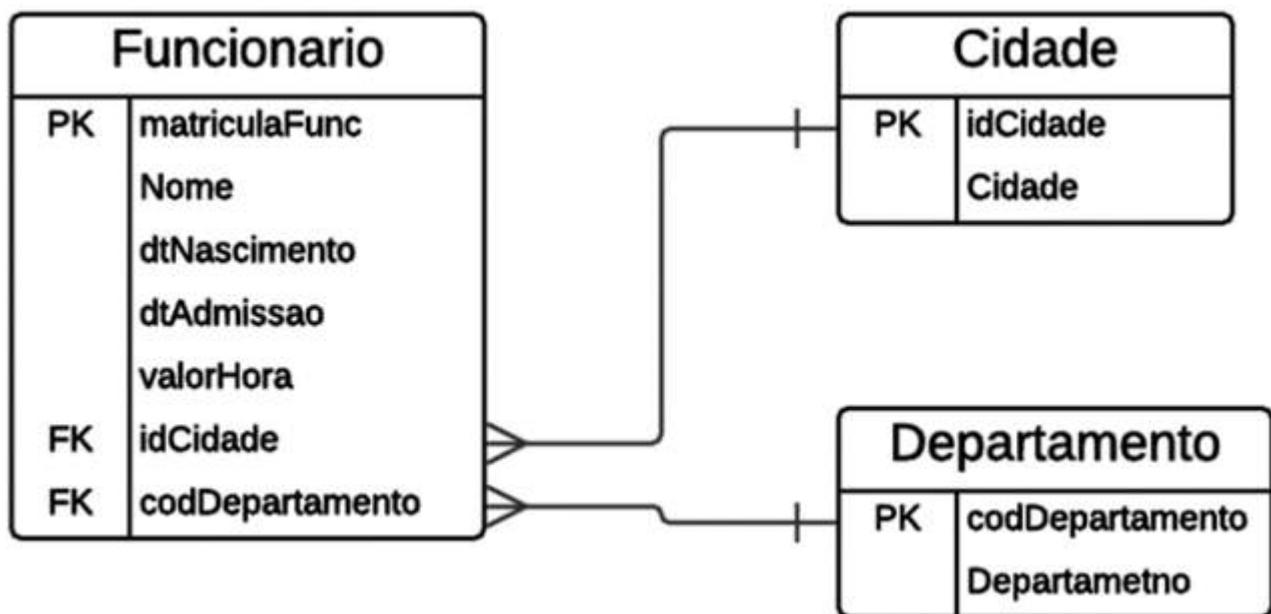


Figura 1 | DER Funcionários na 1FN e na 2FN.

Uma tabela alcançará a terceira forma normal (3FN) apenas quando estiver na segunda forma normal e todos os campos forem independentes, o que significa que não deve haver dependências funcionais entre os campos e todos os campos devem depender exclusivamente da chave primária da tabela. Para atender aos critérios da terceira forma normal, é preciso:

- Identificar os campos que têm dependência funcional com outras colunas não chaves.
- Remover as colunas dependentes.

De acordo com Alves (2020), para tornar uma tabela compatível com a terceira forma normal, é necessário eliminar todas as dependências transitivas, ou seja, remover todos os campos que dependem de outras tabelas.

O Quadro 6 apresenta a tabela "Funcionário", na qual se observa que um dos campos é a "Descrição", mas descrição de quê? É a descrição do cargo que o funcionário ocupa, e, nesse contexto, é necessário aplicar a terceira forma normal à tabela.

#codFuncionario	Nome	idCargo	descCargo
148-9	Jane Anne	191	Analista Contábil I
721-4	Klaus Lins	323	Assistente de Produção II
673-2	Sandra Costa	101	Auxiliar de DP
502-1	Octávio Neto	254	Analista de TI III

Quadro 6 | Funcionário.

Lembre-se de que uma dependência funcional transitiva ocorre quando o valor de uma coluna é dependente de outra coluna que não compõe a chave primária, e, portanto, indiretamente, de outra chave primária. No Quadro 6, o campo "Descrição" depende do "Cargo" (que é outra tabela), e, nesse caso, há uma dependência funcional transitiva. Ao aplicar a 3FN na tabela "Funcionário", retira-se o campo "Descrição", afinal esse é um campo que detalha o cargo e deve estar em uma tabela apropriada, no caso, a tabela "Cargo". O Quadro 8 mostra o quadro Funcionário após aplicarmos a 3FN.

#codFuncionario	Nome	&idCargo
148-9	Jane Anne	191
721-4	Klaus Lins	323
673-2	Sandra Costa	101
502-1	Octávio Neto	254

Quadro 7 | Funcionário normalizado.

O Quadro Cargo, ilustrado no Quadro 8, agora possui a descrição do cargo do funcionário. O relacionamento entre as duas tabelas ("Funcionário" e "Cargo") é de 1 para N. Observe que no Quadro Funcionário há um campo com o sinal &, o que indica que o campo é uma chave estrangeira e a tabela está relacionada com o Quadro Cargo.

#idCargo	descCargo
191	Analista Contábil I
323	Assistente de Produção II
101	Auxiliar de DP
254	Analista de TI III

Quadro 8 | Cargo.

As três primeiras formas normais, 1FN, 2FN e 3FN, nesta aula explicadas, já foram incorporadas implicitamente durante o desenvolvimento da disciplina. Sempre que realizamos o processo de modelagem de dados, é crucial ter o cuidado de adicionar somente os campos que pertencem verdadeiramente à cada tabela. Não misturar campos de diferentes tabelas é uma regra fundamental e essencial na modelagem de dados.

Vamos Exercitar?

Caro estudante, nesta aula você aprendeu que a normalização desempenha um papel crucial na identificação de questões na estrutura do banco de dados. Às vezes, não estamos cientes de que um campo pode ser mais bem representado como uma tabela separada. É importante salientar que as consequências de uma modelagem inadequada só se tornam evidentes quando o banco de dados começa a manifestar uma série de problemas.

Embora o processo de normalização abranja um conjunto de sete formas normais, muitas delas não são pertinentes às atividades do projetista do banco de dados (DBA). Para os responsáveis pela concepção lógica dos dados (modelagem lógica de dados), que desenvolverão o MER, basta conhecer e aplicar as três primeiras. Na próxima aula, conheceremos outras formas normais (não menos importantes) a fim aplicá-las em problemas reais. Utilize a teoria aprendida nesta aula e reflita sobre as importâncias de cada etapa de normalização no caso estudado.

Saiba mais

Para compreender melhor o assunto abordado nesta aula, recomendamos a leitura do artigo de Ávila e Mello, o qual apresenta uma ferramenta de apoio ao processo de normalização de tabelas visando contribuir, principalmente, com projetos *bottom-up* de bancos de dados relacionais. Por meio da análise de dados, a ferramenta descobre dependências funcionais existentes que devem ser removidas.

ÁVILA, M. L. de; MELLO, R. S. [Uma ferramenta de apoio à normalização de tabelas relacionais baseada na análise de dados](#).

Referências

ALVES, W. P. **Banco de dados**: teoria e desenvolvimento. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

Aula 3

Formas Normais II

Formas normais II

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! Nesta aula, concluiremos a apresentação das regras de normalização, as quais desempenham um papel fundamental na identificação de inconsistências na estrutura do banco de dados. Após praticar a aplicação dessas regras várias vezes, você desenvolverá a capacidade de realizar a modelagem de forma mais intuitiva e eficaz. Na maioria dos casos, como no exemplo final da aula anterior, as entidades se encontram normalizadas na 3FN, mas pode ocorrer de, às vezes, uma entidade conter um ou mais fatos multivalorados, ou seja, pode ser que ainda exista relacionamentos muitos para muitos e muitos para um.

Segundo Machado (2020), na maioria dos casos, as entidades normalizadas até a 3FN são fáceis de entender, atualizar e de se recuperar dados, mas às vezes podem surgir problemas com relação a algum atributo não chave, que recebe valores múltiplos para um mesmo valor de chave. Essa nova dependência recebe o nome de multivalorada, e existe somente se a entidade contiver

no mínimo três atributos. Trataremos, pois, nesta seção, desse aspecto, que é justamente o que se estuda nos outros três tipos de formas normais, a saber, a quarta forma normal (4FN), a quinta forma normal (5FN) e a forma normal de Boyce-Codd (FNBC). Bons estudos!

Vamos Começar!

Como vimos na aula anterior, a normalização é um procedimento cujo objetivo principal é minimizar a redundância de dados presente no banco de dados. Sua abordagem central consiste em identificar e reduzir gradualmente as possíveis anomalias que possam surgir nas tabelas ou nos relacionamentos do banco. Em termos gerais, esse processo envolve a extração de um ou mais campos de uma tabela existente e a criação de novas tabelas para acomodar os campos retirados. Conforme destacam Elmasri e Navathe (2018), a normalização proporciona ao modelador de banco de dados uma série de ações que podem ser executadas para aprimorar a qualidade e a eficiência dele, tais como:

- Garantir uma estrutura organizada para examinar minuciosamente as conexões entre tabelas, considerando suas chaves primárias e estrangeiras, bem como as relações funcionais entre seus campos.
- Definir um conjunto de critérios e procedimentos nas formas normais que devem ser aplicados a cada esquema de relação, de modo a normalizar o modelo de banco de dados até o nível considerado mais apropriado para a modelagem.

De acordo com Machado (2020), as definições de 2FN e 3FN, propostas por Codd, não abordavam determinados cenários. Raymond Boyce identificou essa lacuna em 1974. Tais situações não contempladas pelas definições de Codd surgem, assim, apenas quando três condições específicas se combinam:

1. A entidade possui várias chaves candidatas.
2. As chaves candidatas são compostas por múltiplos atributos.
3. As chaves concatenadas compartilham pelo menos um atributo comum.

Já para Silberschatz, Korth e Sudarshan (2020), uma das formas normais mais desejáveis que podemos obter é a forma normal de Boyce-Codd (FNBC). Ela elimina toda redundância que possa ser descoberta com base nas dependências funcionais, porém podendo restar outros tipos de redundância. Logo, um projeto de banco de dados está na FNBC se cada membro do conjunto de esquemas de relação que constitui o projeto estiver na FNBC.

Na realidade, a forma normal de Boyce-Codd (FNBC) estende a 3FN e aborda anomalias que não eram resolvidas pela abordagem anterior. O problema surgiu porque tanto a 2FN quanto a 3FN lidavam apenas com casos de dependência parcial e transitiva de atributos que não faziam parte de nenhuma chave. No entanto, quando o atributo em questão fazia parte de uma chave (seja primária ou candidata), não era considerado pelas verificações da 2FN e 3FN. A definição da FNBC, pois, estabelece o seguinte critério: uma entidade estará na FNBC se, e somente se, todos os seus determinantes forem chaves candidatas. É importante observar que essa definição se

MODELAGEM DE DADOS

baseia nesse tipo de chave, não nas chaves primárias. Imaginemos, a título de exemplo, a entidade **filho** com os seguintes atributos:



Figura 1 | Atributos da entidade Filho. Fonte: Machado (2020).

Considerando uma situação hipotética em que um professor possa estar associado a mais de uma escola e sala de aula, suponhamos que tanto a chave candidata concatenada, que é composta por NOME-DA-ESCOLA + SALA-DA-ESCOLA, quanto a chave candidata concatenada de NOME-DA-ESCOLA + NOME-DO-PROFESSOR possam atuar como determinantes. Essa entidade cumpre, portanto, as três condições mencionadas anteriormente:

1. As chaves candidatas para a entidade Filho incluem NOME-DO-FILHO + ENDERECO-DO-FILHO, NOME-DO-FILHO + NUMERO-DA-SALA e NOME-DO-FILHO + NOME-DO-PROFESSOR.
2. As três chaves consistem em múltiplos atributos concatenados.
3. As três chaves compartilham um atributo comum, que é o NOME-DO-FILHO.

No exemplo apresentado, NOME-DO-PROFESSOR não é totalmente dependente do NUMERO-DA-SALA, e vice-versa. Em outras palavras, NOME-DO-PROFESSOR é completamente dependente da chave candidata concatenada NOME-DO-FILHO + NUMERO-DA-SALA, ou NUMERO-DA-SALA é completamente dependente da chave candidata concatenada NOME-DO-FILHO + NOME-DO-PROFESSOR.

Para aplicar a FNBC, a entidade Filho deve ser dividida em duas entidades distintas. Uma delas conterá todos os atributos que descrevem o filho, enquanto a outra englobará os atributos que identificam um professor em uma escola específica e o número de uma sala de aula.

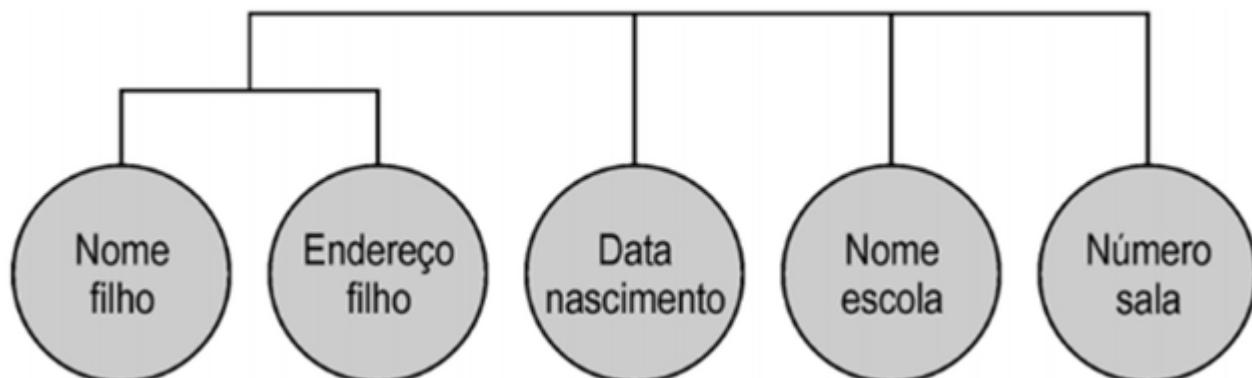


Figura 2 | Entidade FILHO. Fonte: Machado (2020).

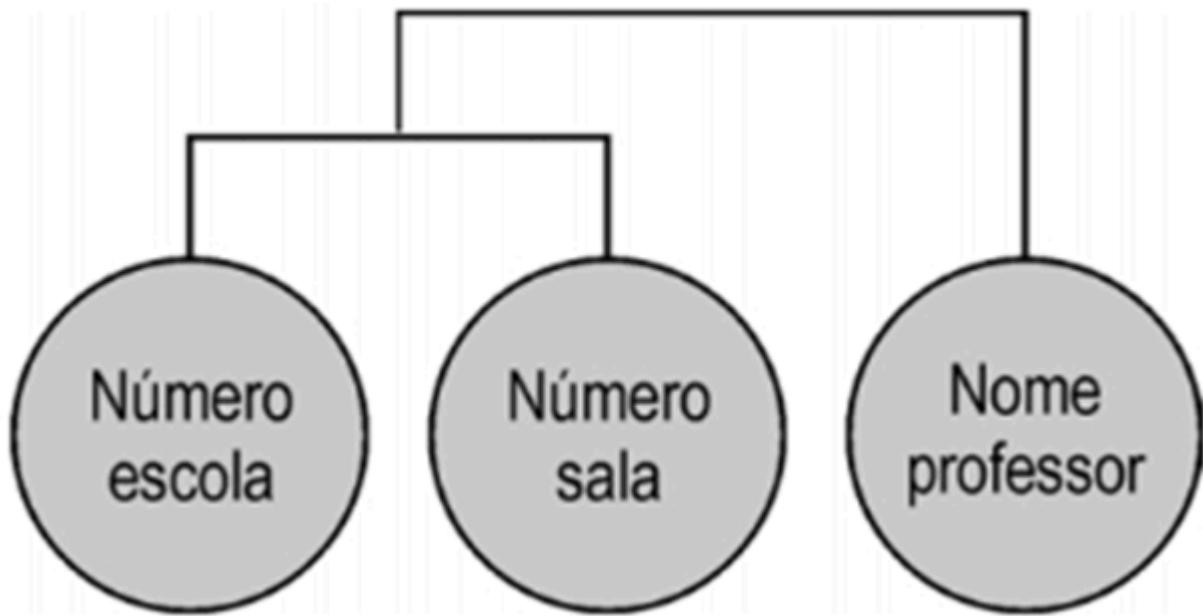


Figura 3 | Entidade SALA. Fonte: Machado (2020).

Os casos para aplicação dessa forma normal são mais raros de encontrar na prática, pois derivam de erros de modelagem realizados quando da estruturação dos atributos de uma entidade.

Para normalizar uma tabela na quarta forma normal (4FN), é necessário que ela já esteja na terceira forma normal (3FN). A tabela alcançará a 4FN quando não apresentar dependências multivaloradas, que ocorrem quando informações repetitivas na tabela podem causar redundância de dados. Para evitar esse tipo de problema, é aconselhável dividir a tabela e eliminar essa dependência. Para Elmasri e Navathe (2018), em uma tabela na 4FN, além de satisfazer os critérios da 3FN, todos os campos devem ser considerados indivisíveis, ou seja, não podem ser subdivididos em múltiplos campos.

Os passos para atingir a 4FN em uma tabela incluem:

1. Identificar os campos que causam dependências multivaloradas, resultando em repetições.
2. Criar uma nova tabela para cada conjunto multivalorado.
3. Estabelecer uma chave primária para cada nova tabela.
4. Inserir chaves estrangeiras na tabela que está sendo normalizada (na 4FN) para estabelecer os relacionamentos necessários entre as tabelas.

De acordo com Machado (2020), uma entidade que está na 3FN também está na 4FN se ela não contiver mais do que um fato multivalorado a respeito da entidade descrita. Essa dependência não é o mesmo que uma associação M:N entre atributos, geralmente descrita desta forma em algumas literaturas, mas ocorre quando consideramos a existência de relacionamentos, por exemplo, ternários. Imaginemos, assim, o conteúdo da tabela “Compra” na figura a seguir:

CodFornecedor	CodProduto	CodComprador
101	BA3	01
102	CJ10	05
110	88A	25
530	BA3	01
101	BA3	25

Figura 4 | Entidade Compra. Fonte: Machado (2020).

Como podemos notar, esta entidade procura abranger duas ocorrências multivaloradas: os vários produtos adquiridos e as diversas categorias. Esse fato resulta em uma relação multivalorada entre **CodFornecedor** e **CodProduto**, bem como entre **CodFornecedor** e **CodComprador**, como vemos na Figura 5.

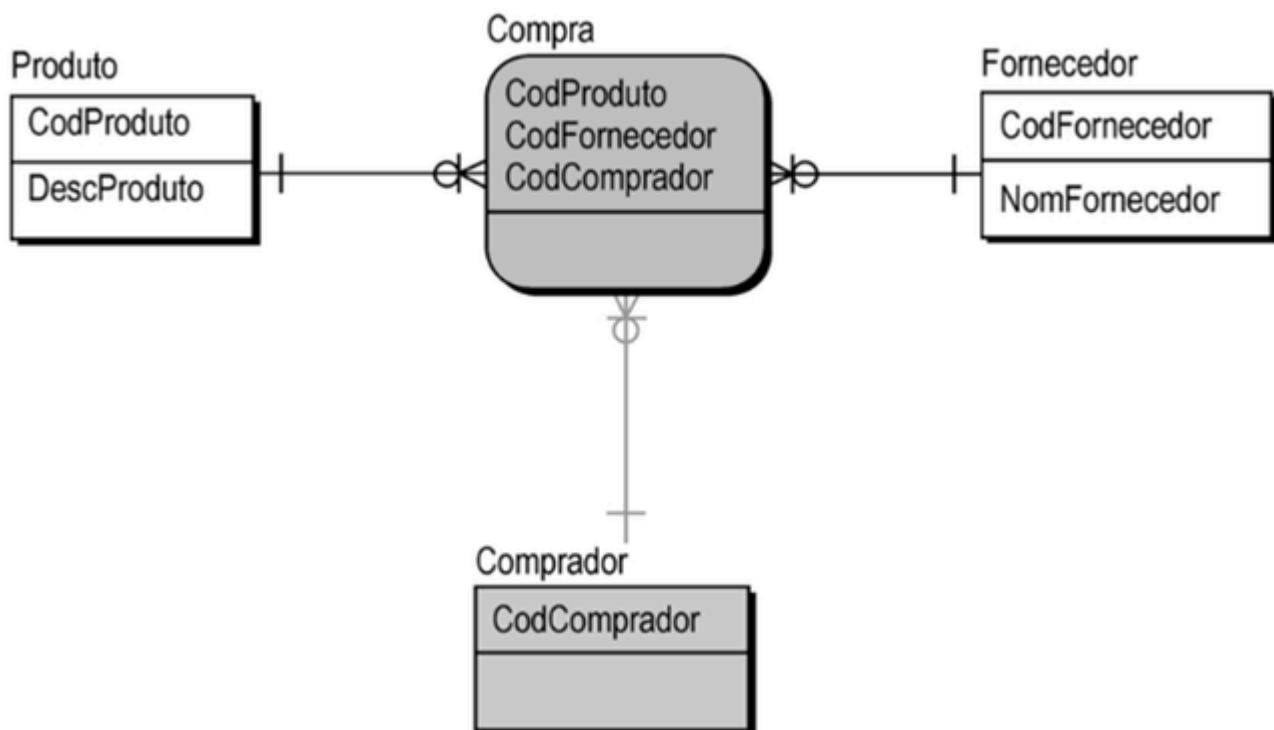


Figura 5 | Exemplo de DER. Fonte: Machado (2020).

Siga em Frente...

Embora esteja em conformidade com a terceira forma normal (3FN), uma vez que não há dependências transitivas, a inclusão de múltiplos fatos multivalorados torna a tabela muito complexa de ser atualizada. Além disso, há o risco de problemas relacionados ao espaço de armazenamento devido ao uso desnecessário de memória primária ou secundária, o que pode resultar em situações críticas que exigiriam mais espaço para outras aplicações. Para migrar a entidade anterior para a quarta forma normal (4FN), é necessário dividir a entidade original em duas novas entidades, ambas compartilhando a chave **CodFornecedor**, concatenadas com os atributos **CodProduto** e **CodComprador**.

CodFornecedor	CodProduto
101	BA3
102	CJ10
110	88A
530	BA3
101	BA3

Figura 6 | Entidade contendo o atributo CodProduto. Fonte: Machado (2020).

CodProduto	CodComprador
BA3	01
CJ10	05
88A	01
BA3	25

Figura 7 | Entidade contendo os atributos CodProduto e CodComprador. Fonte: Machado (2020).

Observando o modelo de dados na Figura 8, podemos entender melhor o exemplo:

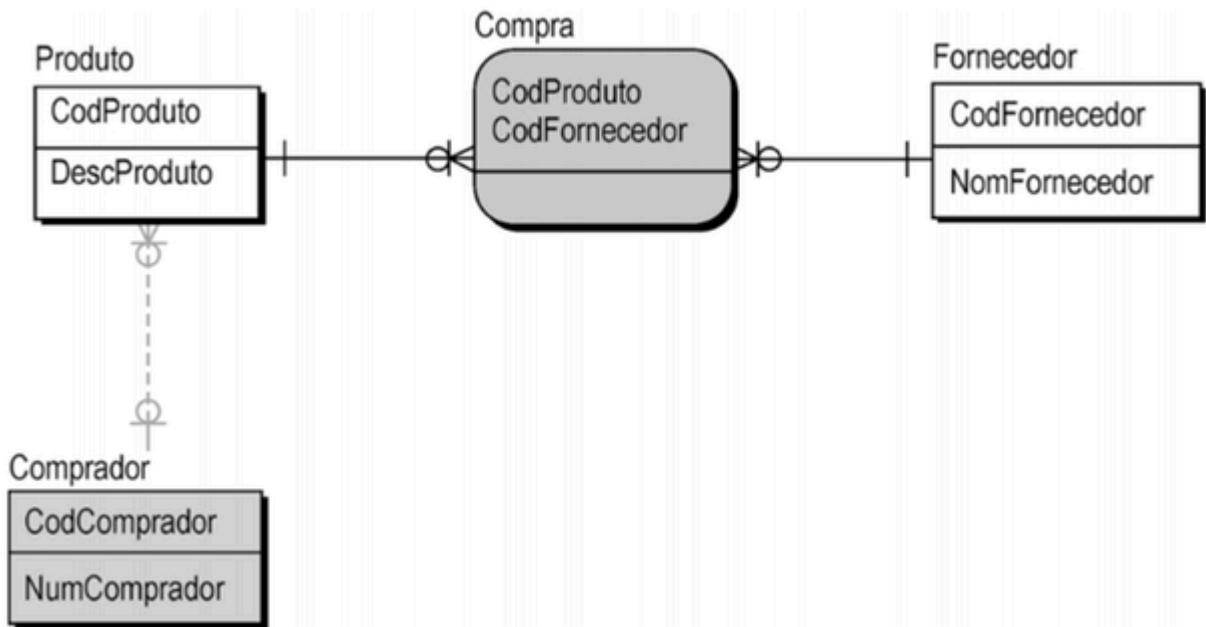


Figura 8 | DER na 4FN. Fonte: Machado (2020).

Acrescentamos, decerto, mais atributos ao modelo para fins ilustrativos, visto que eles não interferem e permitem uma compreensão mais ampla do contexto. Segundo Machado (2020), a aplicação da quarta forma normal (4FN) nesse cenário foi necessária para corrigir um erro na

modelagem de dados, pois no primeiro modelo usamos um relacionamento ternário de maneira desnecessária, associando o comprador exclusivamente ao processo de compra em vez de vinculá-lo aos produtos que ele adquire.

A conectividade apresentada permanece a mesma em ambos os casos: um comprador adquire vários produtos, e um produto é adquirido por um único comprador. No entanto, a anomalia permitiria a violação dessa regra de negócio devido à estruturação incorreta na 4FN no modelo inicial. Um modelo de dados conceitual e lógico pode parecer correto à primeira vista, mas é crucial validar as possibilidades de dados dentro de suas tabelas relacionais para garantir sua precisão e integridade.

Uma condição essencial para que uma tabela alcance a quinta forma normal (5FN) é que ela já esteja na quarta forma normal (4FN). De maneira concisa, podemos afirmar que uma tabela atinge essa forma quando não é mais possível subdividi-la em tabelas menores sem perder informações.

Essa abordagem gera um grande número de tabelas no banco de dados, o que pode impactar negativamente o desempenho do sistema. Por essa razão, ela não é amplamente adotada e, portanto, não nos concentraremos nela nesta aula.

A normalização desempenha um papel crucial na identificação de problemas na estruturação de um banco de dados. Em muitos casos, não é imediatamente evidente que um simples campo poderia ser mais bem representado como uma tabela separada. As consequências de uma modelagem inadequada geralmente só se tornam visíveis quando o banco de dados começa a manifestar uma série de problemas. Logo, é essencial aplicar preferencialmente as três formas normais e, somente em alguns casos, analisar a quarta e a quinta formas e/ou a forma normal de Boyce-Codd (FNBC), para assim garantir uma estrutura de banco de dados eficaz e livre de anomalias.

Vamos Exercitar?

Antes de formularmos qualquer conclusão, é importante notar que as formas normais são, em sua essência, restrições de integridade que se tornam progressivamente mais rigorosas à medida que avançamos na normalização. A utilidade dessas restrições pode variar de acordo com o sistema de gerenciamento de banco de dados relacional (SGBD) utilizado.

A abordagem da normalização exerce papéis distintos ao longo do ciclo de vida de um projeto de banco de dados. Ela pode ser mais eficaz na fase inicial de desenvolvimento (*bottom-up*), que se inicia a partir da normalização da documentação existente no ambiente analisado e dos arquivos utilizados em processos automatizados. No contexto do desenvolvimento *top-down*, no qual um modelo de dados é construído a partir de uma visão conceitual da realidade, a normalização ajuda a aprimorá-lo, tornando-o menos suscetível a redundâncias e inconsistências. Nesse cenário, ela se torna uma aliada valiosa na fase de implementação física do modelo.

Com base na experiência prática, reforçamos o fato de que a elaboração de um modelo de dados naturalmente conduz a entidades e relacionamentos que atendem à terceira forma normal (3FN), deixando a aplicação das formas normais FNBC, 4FN e 5FN para refinamentos e otimizações posteriores.

É importante notar que a simples criação de modelos de dados a partir da normalização de documentos e arquivos não é a abordagem mais recomendada. Isso porque tal abordagem focaliza o problema sem propor uma solução adequada. Nesse caso, projetamos estruturas de dados que se baseiam na situação atual, muitas vezes caótica, e que não atendem efetivamente às necessidades do ambiente analisado. Porém, ao criarmos um modelo de dados que reflete a realidade em estudo (o mundo real), desenvolvemos naturalmente uma base de dados que se alinha com a visão da realidade e, como resultado, solucionamos os problemas de informações.

Nossa experiência profissional tem demonstrado que a modelagem de dados, com a técnica de normalização, é uma ferramenta valiosa que pode ser aplicada tanto na fase inicial de levantamento de informações (documentos e arquivos) quanto na otimização de modelos de dados existentes. Esse fato é particularmente relevante se consideradas as restrições associadas à implementação física nos sistemas de gerenciamento de banco de dados tradicionais.

Saiba mais

Para melhor compreender o assunto abordado nesta aula, recomendamos a leitura do capítulo 8 do livro de Felipe Nery R. Machado, Banco de dados: projeto e implementação, no qual é possível encontrar exemplos completos e roteirizações de modelagem aplicando a normalização de dados.

MACHADO, F. N. R. [**Banco de dados: projeto e implementação**](#). 4. ed. São Paulo: Érica, 2020.

Referências

ALVES, W. P. **Banco de dados: teoria e desenvolvimento**. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.

Aula 4

Engenharia Reversa de bancos de dados

Engenharia Reversa de bancos de dados

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Partida

Olá, estudante! A engenharia reversa de bancos de dados relacionais é um processo fundamental na área de gerenciamento de dados e sistemas de informação. Esse processo envolve a análise e reconstrução da estrutura de um banco de dados a partir de um banco de dados existente, muitas vezes sem documentação adequada. Imagine, para ilustrar esse conceito, uma situação em que uma organização tenha um sistema de gerenciamento de recursos humanos em funcionamento há vários anos. Esse sistema mantém informações valiosas sobre funcionários, folhas de pagamento, benefícios e outras informações críticas. No entanto, com o passar dos anos, as informações sobre a estrutura do banco de dados e sua documentação foram perdidas ou nunca foram bem elaboradas.

É neste ponto que atua a engenharia reversa de banco de dados relacionais e que um profissional de banco de dados ou engenheiro de software qualificado assume o desafio de analisar o banco de dados existente e compreender sua estrutura subjacente. Ele busca identificar as tabelas, relacionamentos, chaves primárias, chaves estrangeiras e outros elementos que constituem o banco de dados. Para tanto, esse profissional pode usar ferramentas de engenharia reversa específicas que se conectam ao banco de dados existente e ajudam a recuperar informações sobre sua estrutura. Ao aplicar a engenharia reversa, portanto, ele poderá criar um modelo lógico

do banco de dados que refletira sua organização, facilitando a compreensão e a futura manutenção.

Em síntese, a engenharia reversa de banco de dados relacionais é essencial quando se planeja realizar atualizações, melhorias ou migrações do sistema existente. Ter uma compreensão completa da estrutura do banco de dados permite tomar decisões informadas e garantir que as mudanças sejam implementadas de maneira eficiente e sem impactar negativamente os dados e processos existentes. Mergulhemos, pois, neste assunto. Bons estudos!

Vamos Começar!

Muitas organizações, na atualidade, passam constantemente por migrações de sistemas e, até mesmo, de bancos de dados. Por exemplo, migrações de SGBDs, migração de banco de dados *on-premise* para banco de dados em nuvem etc. Silberschatz, Korth e Sudarshan (2020), nesse contexto, destacam algumas observações sobre os sistemas legados, sistemas de aplicação obsoletos mas que ainda continuam em uso. Essa persistência ocorre devido aos altos custos e riscos associados à sua substituição. Em alguns casos, inclusive, os sistemas legados podem usar tecnologias desatualizadas, incompatíveis com os padrões e sistemas modernos. Alguns deles têm décadas de uso e baseiam-se em tecnologias ultrapassadas, como bancos de dados hierárquicos ou de rede, além de linguagens de programação obsoletas como COBOL.

A substituição de um sistema legado por um novo é um processo dispendioso em termos de tempo e recursos. Esses sistemas costumam ser extensos, contendo milhões de linhas de código desenvolvidas ao longo de muitos anos. Eles também armazenam uma grande quantidade de dados valiosos que precisam ser migrados para a nova aplicação, a qual pode seguir um esquema completamente diferente. Essa transição envolve treinamento de pessoal e deve ser realizada sem interrupções, com os dados do sistema antigo ainda disponíveis no novo sistema.

Para evitar a substituição direta, muitas organizações optam por manter a interoperabilidade entre sistemas legados e sistemas novos. Essa opção é alcançada criando uma camada, conhecida como *wrapper*, que permite que o sistema legado seja acessado como se fosse um banco de dados relacional. O *wrapper* oferece suporte a padrões de interconexão, como ODBC ou OLE-DB, para consultar e atualizar o sistema legado. Sua função principal é traduzir consultas e atualizações relacionais em comandos comprehensíveis pelo sistema antigo.

Quando a decisão de substituir o sistema legado por um novo acontece, o processo de engenharia reversa é adotado, o que envolve analisar o código do sistema legado para criar um novo projeto de esquema, como um modelo de dados ER ou orientado a objetos. A engenharia reversa, assim, também procura entender os procedimentos e processos implementados no sistema antigo, proporcionando uma visão de alto nível. Isso é necessário porque os sistemas legados geralmente carecem de documentação adequada.

Uma vez que um novo sistema é desenvolvido, ele precisa ser preenchido com os dados do sistema legado e todas as funcionalidades precisam ser reproduzidas. No entanto, a transição abrupta para um novo sistema, chamada de abordagem big-bang, traz riscos, incluindo a falta de familiaridade dos usuários com as novas interfaces e possíveis bugs não detectados durante os testes. Essa situação pode levar a grandes prejuízos para as empresas, pois afeta a capacidade de realizar transações críticas. Em casos extremos, o novo sistema pode ser abandonado em favor do sistema legado.

Uma alternativa, neste cenário, é a abordagem incremental, na qual a funcionalidade do sistema legado é substituída gradualmente. Novas interfaces de usuário podem ser utilizadas com o sistema antigo no back-end e vice-versa. Em última análise, os sistemas legado e novo coexistem por algum tempo, e é necessário o desenvolvimento de *wrappers* para interoperabilidade. No entanto, essa abordagem envolve custos adicionais de desenvolvimento.

Um exemplo interessante é citado por Alves (2020), no qual o autor destaca que o MySQL Workbench, por exemplo, além de possibilitar a geração de um banco de dados a partir de um diagrama entidade-relacionamento nele criado, também oferece um recurso bastante interessante, comumente conhecido como engenharia reversa. Esse recurso consiste na geração do diagrama entidade-relacionamento a partir de uma base de dados.

Siga em Frente...

Engenharia reversa em banco de dados relacionais, em síntese, é um processo que visa entender e documentar um banco de dados existente, muitas vezes legado, a partir do qual não se possui informações detalhadas. Isso é feito para criar uma representação estruturada do banco de dados, como um modelo de dados, um diagrama Entidade-Relacionamento (ER), ou um esquema facilitando a manutenção, atualização ou migração do sistema. A seguir, indicamos alguns dos passos necessários para realizar a engenharia reversa em um banco de dados relacional:

- 1. Coleta de informações:** o primeiro passo é reunir todos os recursos disponíveis para entender o banco de dados, o que inclui a coleta de documentação existente, como diagramas ER, esquemas, manuais, consultas SQL, procedimentos armazenados, entre outros. Também é importante conversar com os usuários e administradores do sistema para obter insights sobre como o banco de dados é usado na prática.
- 2. Análise do banco de dados:** uma vez que você tenha todas as informações disponíveis, comece a analisar o banco de dados em si. Essa ação envolve examinar as tabelas, colunas, chaves primárias e estrangeiras, índices, relacionamentos e restrições de integridade referencial. Você também deve verificar as regras de negócio que o banco de dados suporta.
- 3. Uso de ferramentas de engenharia reversa:** existem várias ferramentas de engenharia reversa disponíveis que podem ajudar nesse processo. Essas ferramentas podem se conectar diretamente ao banco de dados e extrair informações sobre sua estrutura. Elas podem gerar automaticamente diagramas ER, esquemas de banco de dados ou código SQL a partir dos metadados do banco.

4. **Documentação e modelagem:** com base nas informações coletadas e nas saídas das ferramentas de engenharia reversa, comece a criar uma documentação detalhada do banco de dados. Isso pode incluir um modelo de dados, diagramas ER, dicionário de dados e descrições das tabelas e colunas. A documentação deve ser clara e completa, de modo a refletir com precisão a estrutura e os relacionamentos do banco de dados.
5. **Validação e ajustes:** após a criação da documentação inicial, é importante validar sua precisão, o que pode ser feito comparando as informações com os usuários reais do sistema e testando o modelo em relação aos dados reais. É possível que ajustes sejam necessários à medida que novas informações são obtidas.
6. **Manutenção contínua:** a engenharia reversa não é um processo único. À medida que o banco de dados evolui, é fundamental manter a documentação atualizada. Alterações no esquema do banco de dados devem ser refletidas na documentação para garantir que ela permaneça útil e precisa.

A engenharia reversa em bancos de dados relacionais é uma abordagem fundamental para lidar com sistemas legados e para melhorar a compreensão e a manutenção de bancos de dados existentes. É uma tarefa desafiadora, mas o uso de ferramentas apropriadas e a colaboração com as partes interessadas adequadas tornarão o processo mais eficiente e preciso.

A Figura 1 apresenta uma visão geral do processo de engenharia reversa de arquivos convencionais. O processo parte das descrições dos arquivos que compõem o sistema existente. Portanto, é um processo que segue a estratégia ascendente de construção de modelos ER.

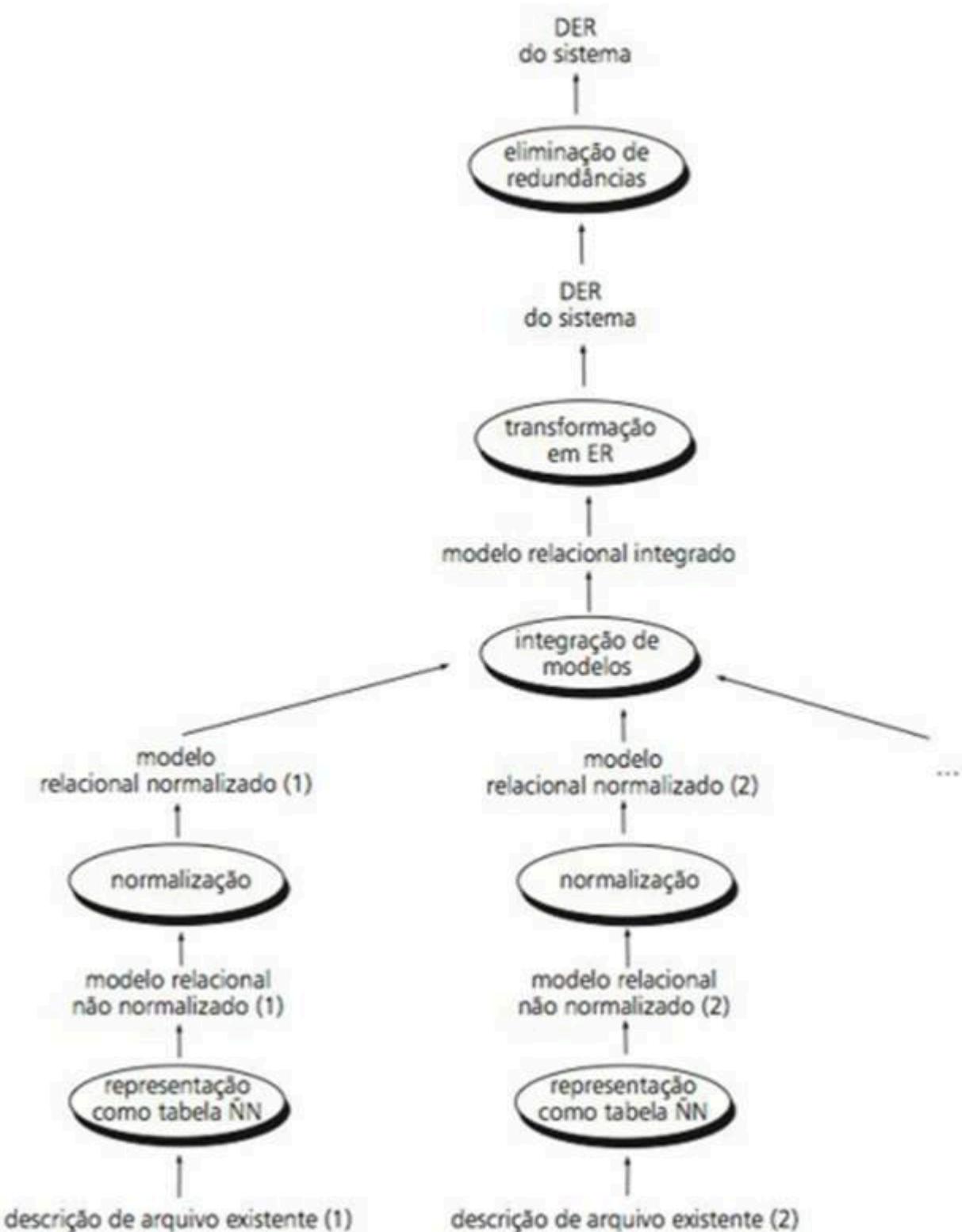


Figura 1 | Visão geral do processo de engenharia reversa de arquivos convencionais. Fonte: Heuser (2009).

O primeiro passo é a representação da descrição de cada arquivo existente na forma de um esquema de uma tabela relacional não-normalizada. Esse processo inicial visa adquirir uma

descrição que seja independente do tipo de arquivo em uso. A partir desse ponto, o procedimento concentra-se exclusivamente em tabelas relacionais, garantindo sua independência em relação ao formato do arquivo de entrada no processo de normalização.

Posteriormente, esse esquema de tabela não normalizada passa pelo processo de normalização, cujo propósito é duplo:

1. Reorganizar os dados de modo a eliminar quaisquer redundâncias presentes nos arquivos.
2. Reorganizar os dados de maneira a permitir a obtenção de um modelo entidade-relacionamento (ER).

Uma vez que todos os arquivos do sistema tenham sido normalizados, os vários esquemas relacionais resultantes da normalização são combinados para formar o esquema relacional do banco de dados do sistema. Nesse estágio, as informações comuns a diferentes arquivos são identificadas e representadas de forma unificada.

Por fim, a partir do esquema relacional obtido, seguindo as regras de engenharia reversa de um modelo relacional descritas no capítulo anterior, é possível obter o modelo entidade-relacionamento (ER) do sistema existente.

Vamos Exercitar?

A engenharia reversa de banco de dados relacionais, portanto, desempenha um papel crucial na preservação e otimização de sistemas de informação existentes. É uma habilidade valiosa para profissionais de banco de dados e engenheiros de software que desejam garantir que os dados de uma organização sejam gerenciados com eficácia e estejam alinhados com suas necessidades em constante evolução. Suponha, a título de ilustração, que em uma biblioteca de uma pequena cidade, o sistema de gerenciamento de empréstimos de livros era um legado antigo, construído há décadas. Esse sistema operava em um ambiente de banco de dados que usava modelos de dados hierárquicos, tornando-o obsoleto e difícil de manter. A equipe da biblioteca reconheceu a necessidade de modernizar seu sistema de controle, mas enfrentou o desafio de entender o funcionamento do sistema existente e de migrar seus dados para um formato mais contemporâneo. A solução foi realizar um processo de engenharia reversa no banco de dados legado contendo:

1. **Análise inicial:** a equipe identificou a necessidade de migrar para um sistema de banco de dados relacional moderno. Foram coletados documentos e arquivos relacionados ao sistema legado para entender a estrutura de dados e a lógica de negócios.
2. **Engenharia reversa:** com a documentação em mãos, a equipe mapeou a estrutura de tabelas e campos existentes no sistema hierárquico. Usando técnicas de normalização, as tabelas hierárquicas foram convertidas em um modelo de dados relacional mais claro, seguindo as melhores práticas. Um novo esquema relacional foi projetado, incorporando informações sobre livros, autores, empréstimos, membros da biblioteca e histórico de transações.

3. **Criação de wrappers:** para garantir uma transição suave, a equipe desenvolveu *wrappers* para permitir a interoperabilidade entre o sistema legado e o novo sistema relacional. Isso permitiu que as operações continuassem enquanto a migração estava em andamento.
4. **Migração de dados:** os dados do sistema hierárquico foram extraídos, transformados e carregados no novo banco de dados relacional, mantendo a consistência dos registros.
5. **Testes e implementação:** após a migração bem-sucedida, o novo sistema foi extensivamente testado. A equipe fez a transição gradual dos usuários para o novo sistema, oferecendo treinamento para garantir que todos estivessem confortáveis com a nova interface.
6. **Benefícios obtidos:** o novo sistema permitiu uma gestão de empréstimos de livros mais eficaz, com consultas e relatórios mais ágeis. A biblioteca conseguiu eliminar a dependência do sistema legado e reduzir os custos de manutenção.

Neste case, a engenharia reversa do banco de dados possibilitou a modernização do sistema de controle de biblioteca. Com a migração bem-sucedida para um banco de dados relacional, a biblioteca agora pode atender seus membros de forma mais eficaz e manter-se atualizada com as melhores práticas de gerenciamento de dados.

Saiba mais

Para compreender melhor o assunto abordado nesta aula, indicamos a leitura do capítulo 6 do livro de Heuser, *Projeto de banco de dados*, no qual são dados exemplos de ponta-a-ponta do processo de engenharia reversa e normalização de dados.

HEUSER, C. A. [Projeto de banco de dados](#). 6. ed. Porto Alegre: Bookman, 2009.

Referências

ALVES, W. P. **Banco de dados: teoria e desenvolvimento**. 2. ed. São Paulo: Érica, 2020.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

HEUSER, C. A. [Projeto de banco de dados](#). 6. ed. Porto Alegre: Bookman, 2009.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dica para você

Aproveite o acesso para baixar os slides do vídeo, isso pode deixar sua aprendizagem ainda mais completa.

Estudante, esta videoaula foi preparada especialmente para você. Nela, você irá aprender conteúdos importantes para a sua formação profissional. Vamos assisti-la?
Bons estudos!

Ponto de Chegada

Caro estudante, a normalização de dados e a engenharia reversa em banco de dados são tópicos essenciais no campo da gestão de informações. Essas competências são fundamentais para projetar, entender e otimizar bancos de dados, garantindo a eficiência e integridade dos dados. A seguir estão alguns pontos-chave para que você possa continuar sua jornada de aprendizado:

1. Compreenda os conceitos básicos: comece pelo básico. Entenda o que são bancos de dados, tabelas, registros e campos. Familiarize-se com os conceitos de chaves primárias e estrangeiras, bem como com a ideia de relacionamentos entre tabelas.
2. Formas normais: aprenda sobre as formas normais (1FN, 2FN, 3FN e assim por diante) e porque elas são importantes. Cada forma normal visa reduzir a redundância e as anomalias nos dados, tornando o banco de dados mais organizado e eficiente.
3. Processo de normalização: explore o processo de normalização, o qual envolve a divisão de tabelas em estruturas mais simples e relacionáveis. Essa ação reduz a redundância de dados e melhora sua integridade.
4. Engenharia reversa: descubra como a engenharia reversa é usada para analisar e compreender sistemas de banco de dados existentes. Aprenda a converter estruturas legadas em modelos de dados mais modernos.

5. Modelagem de dados: invista tempo na modelagem de dados. Entenda como projetar um banco de dados, criar esquemas, identificar chaves e relacionamentos e aplicar as formas normais. A modelagem é a base para qualquer sistema de banco de dados.
6. Ferramentas e tecnologias: explore ferramentas e tecnologias que facilitam a normalização e a engenharia reversa. Saiba como usar softwares de modelagem de dados e ferramentas de migração de banco de dados.
7. Prática e experiência: aplique seus conhecimentos na prática. Crie e normalize bancos de dados fictícios. Tente realizar a engenharia reversa em sistemas simples para ganhar experiência.
8. Aprenda com estudos de caso: estude casos reais de normalização e engenharia reversa. Eles oferecem insights valiosos sobre como lidar com sistemas complexos e legados.
9. Mantenha-se atualizado: a tecnologia está em constante evolução. Acompanhe as novas tendências em bancos de dados e práticas de normalização.

Lembre-se de que a normalização e a engenharia reversa são habilidades essenciais para quem trabalha com bancos de dados. Elas ajudam a manter a consistência e a qualidade dos dados, aspectos cruciais em qualquer cenário onde a informação desempenha um papel fundamental.

Continue explorando esses tópicos, pratique e nunca pare de aprender. À medida que você aprofunda seu conhecimento, estará preparado para lidar com desafios cada vez mais complexos no mundo dos bancos de dados. Boa jornada de aprendizado!

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Após fazer compras em uma pequena loja de materiais de construção, você percebeu que o controle dos produtos levados pelos clientes era cadastrado em um caderno com fichas coladas para serem preenchidas de forma manual. Somente clientes antigos e de confiança possuem a regalia de pegar os produtos e pagar depois. Ao saberem que você é da área de informática, pediram que fizesse um banco de dados para agilizar o processo de controle. Para tanto, você recebeu uma cópia da ficha de controle que permite a retirada de produtos da loja, como ilustra a Figura 1

FICHA DE CONTROLE DE RETIRADA DE MATERIAIS

Controle ficha nº:		Data:
Cliente:	RG:	CPF:
Endereço:	Cidade:	UF:

Produtos				
Código	Descrição	Quantidade	Preço unitário	Preço total
Valor total a pagar:				

Figura 1 | Ficha de controle de retirada de materiais.

Como ficará o processo de normalização das tabelas até a 4FN? Qual o DER resultante da modelagem do documento?

- Importância da normalização: como a normalização de dados pode impactar positivamente a eficiência e a integridade de um banco de dados? Que problemas ou desafios podem surgir quando os dados não são normalizados adequadamente?
- Engenharia reversa na prática: quais são os passos práticos envolvidos na aplicação da engenharia reversa em um banco de dados legado?
- Balanceando a normalização e o desempenho: em que situações a normalização excessiva pode levar a problemas de desempenho e como você os evitaria?

Essas questões incentivam a reflexão sobre as implicações práticas da normalização de dados e da engenharia reversa, bem como sobre os desafios que os profissionais de banco de dados enfrentam ao lidar com sistemas legados e a busca por estruturas mais eficientes.

Primeiramente, foi feita uma busca dos campos no documento e encontrados os seguintes elementos:

No de controle da ficha, Data da nota, Nome Cliente, RG Cliente, CPF Cliente, UF, Cidade, Código Produto, Descrição produto, Quantidade, Preço unitário, Preço item, Valor total da nota.

Na sequência, foram encontradas as tabelas: Ficha de Controle, Cliente, Produto, Cidade, Estado. Aplicamos a 1FN e a 2 FN nelas. Para aplicar a 1FN, devemos inserir a chave primária nas tabelas:

Ficha (#numControle, DtNota, valorTotal, Cidade, Estado)

Cliente (#CPF, Nome, RG, Endereco)

Produto (#codProduto, Descricao, precoUnitario, quantidade, precoTotal)

Para aplicar a 2FN, devemos criar as tabelas Cidade e Estado e inserir as chaves estrangeiras na tabela Ficha.

Ficha (#numControle, DtNota, valorTotal, &idCidade, &siglaEstado)

Cidade (#idCidade, Cidade)

Estado (#siglaEstado, Estado)

Para aplicar a 3FN, devemos eliminar todos os campos dependentes. Observando a tabela Produto, podemos retirar o campo precoTotal, pois ele pode ser calculado a partir da quantidade do produto e de seu preço unitário.

Para aplicar a 4FN, devemos verificar se não há campos multivalorados e não atômicos. Neste exemplo, temos o campo endereço, que poderia ser inserido em uma tabela, detalhando os campos que serão armazenados.

O DER resultante da modelagem do documento da figura anterior pode ser analisado na Figura 2 a seguir.

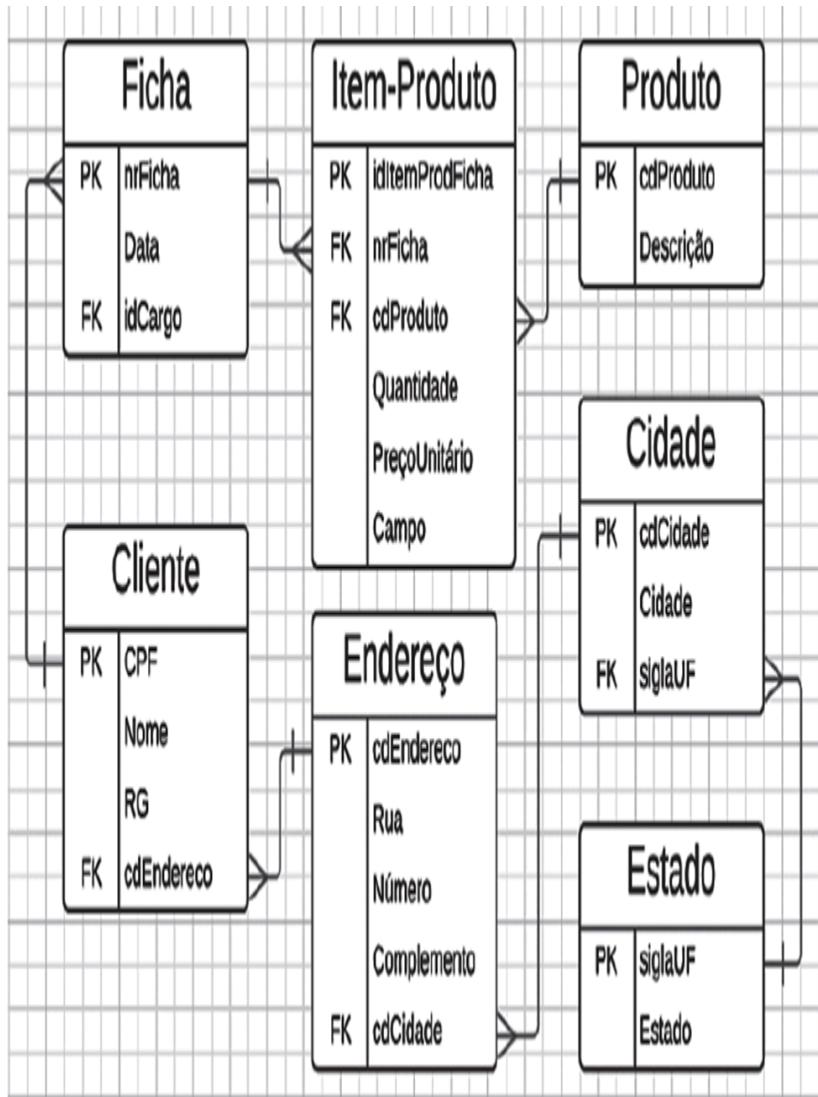


Figura 2 | DER resultante da modelagem da ficha de controle de retirada de materiais.

Caro estudante, a seguir apresentamos um breve infográfico contendo um resumo dos principais temas abordados nesta unidade de forma simples e objetiva. Reserve um tempo para leitura desse material e reflita.

DICAS PARA NORMALIZAÇÃO DE DADOS

1

ENTENDA OS CONCEITOS BÁSICOS

Antes de iniciar o processo de normalização, familiarize-se com os conceitos das Formas Normais, como 1NF, 2NF e 3NF. Compreender o que cada forma normal representa é fundamental.

2

IDENTIFIQUE CHAVES PRIMÁRIAS

Identifique corretamente as chaves primárias de suas tabelas, pois elas desempenham um papel crucial na normalização. Elas ajudam a determinar dependências funcionais e eliminar redundâncias.

3

EVITE REDUNDÂNCIAS DE DADOS

Elimine a redundância de dados, mantendo informações em um local centralizado. Isso simplifica as atualizações, reduz erros e economiza espaço de armazenamento.

4

USE FERRAMENTAS DE MODELAGEM

Utilize ferramentas de modelagem de banco de dados para visualizar e planejar a estrutura normalizada. Essas ferramentas facilitam a criação de esquemas e a identificação de relacionamentos.

5

MANTENHA O EQUILÍBRIO

Encontre um equilíbrio entre a normalização e o desempenho. A normalização rigorosa é essencial, mas em alguns casos, a desnortnalização controlada pode ser necessária para otimizar consultas complexas.

Figura | Dicas para normalização de dados.

ALVES, W. P. **Banco de dados: teoria e desenvolvimento.** 2. ed. São Paulo: Érica, 2020.

MODELAGEM DE DADOS

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: LTC, 2020.

SORDI, J. O. de. **Modelagem de dados: estudos de casos abrangentes da concepção lógica à implementação**. 1. ed. São Paulo: Érica, 2019.