UNIP
UNIVERSIDADE PAULISTA

Interativa

Banco de Dados

Autora: Profa. Sandra Muniz Bozolan
Colaboradores: Prof. Angel Antonio Gonzalez Martinez
Prof. Tiago Guglielmeti Correale

Professora conteudista: Sandra Muniz Bozolan

É doutoranda e mestre (2016) em Tecnologias da Inteligência e Design Digital pela Pontifícia Universidade Católica de São Paulo. Também é especialista em Segurança da Informação e graduada em Tecnologia da Informação. É professora e coordenadora auxiliar nos cursos de Análise e Desenvolvimento de Sistemas e Gestão de Tecnologia da Informação da Universidade Paulista. É professora titular no Instituto Tecnológico de Barueri. Atua como consultora em tecnologia educacional e tem fluência em diversas linguagens de programação na área de computação.

Dados Internacionais de Catalogação na Publicação (CIP)

B793b Bozolan, Sandra Muniz.

Banco de Dados / Sandra Muniz Bozolan. – São Paulo: Editora Sol, 2021.

204 p., il.

Nota: este volume está publicado nos Cadernos de Estudos e Pesquisas da UNIP, Série Didática, ISSN 1517-9230.

1. Dados. 2. Ferramentas. 3. Licença. I. Título.

CDU 681.3.07

U511.55 – 21

© Todos os direitos reservados. Nenhuma parte desta obra pode ser reproduzida ou transmitida por qualquer forma e/ou quaisquer meios (eletrônico, incluindo fotocópia e gravação) ou arquivada em qualquer sistema ou banco de dados sem permissão escrita da Universidade Paulista.

Prof. Dr. João Carlos Di Genio
Reitor

Prof. Fábio Romeu de Carvalho
Vice-Reitor de Planejamento, Administração e Finanças

Profa. Melânia Dalla Torre
Vice-Reitora de Unidades Universitárias

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Pós-Graduação e Pesquisa

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Graduação

Unip Interativa – EaD

Profa. Elisabete Brihy
Prof. Marcello Vannini
Prof. Dr. Luiz Felipe Scabar
Prof. Ivan Daliberto Frugoli

Material Didático – EaD

Comissão editorial:
Dra. Angélica L. Carlini (UNIP)
Dr. Ivan Dias da Motta (CESUMAR)
Dra. Kátia Mosorov Alonso (UFMT)

Apoio:
Profa. Cláudia Regina Baptista – EaD
Profa. Deise Alcantara Carreiro – Comissão de Qualificação e Avaliação de Cursos

Projeto gráfico:
Prof. Alexandre Ponzetto

Revisão:
Irana Magalhães
Willians Calazans

Sumário

Banco de Dados

Unidade IV

APRESENTAÇÃO

O objetivo desta disciplina é fornecer os conceitos fundamentais de bancos de dados e apresentar as principais técnicas de modelagem conceitual, lógica e gerenciamento de dados. O aluno aprenderá a programar um banco de dados na linguagem SQL, que permite definir e manipular dados. O conteúdo deste material ensina sobre a modelagem de banco de dados, a criação de diagramas de entidade-relacionamento, a normalização de modelos de banco de dados, a criação de consulta, os procedimentos armazenados, visões e funções, e também como administrar os bancos de dados criados. Além disso, apresentaremos a plataforma Microsoft SQL Server, que será utilizada como ferramenta-padrão nesse aprendizado.

Ao ler este livro-texto, espera-se que o aluno compreenda toda a complexidade que envolve a modelagem, a criação e o gerenciamento de banco de dados. Para isso, o livro foi dividido em quatro unidades. Na primeira unidade, serão apresentados o período histórico do surgimento dos bancos de dados e a sua importância nas organizações. Discutiremos os princípios da modelagem dos dados iniciais até a exemplificação do modelo físico a partir da modelagem.

A segunda unidade tem como foco os elementos fundamentais de um modelo entidade-relacionamento (ER), generalização, especialização, cardinalidade e relacionamentos, dependência, além da normalização e suas formas.

Na terceira unidade, avançaremos em conceitos mais detalhados do modelo lógico, da geração de *scripts* e veremos como funciona a criação do banco de dados com a linguagem SQL.

Na quarta unidade, aprofundaremos o conceito de sistema de gerenciamento de banco de dados – SQLServer, sua administração, tipos de *backup*, bem como as noções sobre tecnologias de acesso a banco de dados ODBC e JDBC. Junto a isso, entenderemos quais são as melhores soluções de banco de dados para cada necessidade e, por fim, o dimensionamento de *hardware*.

Esperamos que você tenha um bom estudo e sinta-se motivado a ler e conhecer mais sobre bancos de dados.

INTRODUÇÃO

A era da informação nos cerca em todos os contextos com uma quantidade e velocidade de dados sem precedentes. A revolução da informação está modificando o perfil da indústria mundial e muitas dessas mudanças estão sendo criadas a partir de informações sobre o comportamento das pessoas e suas organizações. Isso ocorre no ritmo dos 7 Vs do *big data*: volume, velocidade, variedade, veracidade, variabilidade, visualização e valor. Esses termos são conhecidos por nós e, frequentemente, são utilizados por empresas para capturar as nossas informações. Diante desses conceitos, é importante entender o que é um banco de dados e como lidar com esse volume de informações.

Banco de dados é uma estrutura computacional compartilhada e integrada que armazena um conjunto de dados de um usuário final, ou seja, dados brutos que poderão ser manipulados no momento em que essa informação for necessária. Sem um banco de dados, uma aplicação de *software*, seja a mais simples ou a mais complexa, não teria utilidade. Sendo assim, os sistemas de informação e de comunicações e a própria inteligência artificial necessitam de um banco de dados.

Unidade I

1 INTRODUÇÃO AO BANCO DE DADOS

O volume de informações que acessamos e sua disponibilidade estão alcançando níveis inimagináveis e, por conta disso, muitas organizações têm passado a prestar mais atenção no valor dos dados e de que forma esses dados podem ser geridos. Para obter a maior parte de seus grandes e complexos conjuntos de dados, os usuários necessitam de ferramentas que simplifiquem as tarefas de gerenciamento dos dados e a extração de informações úteis de forma oportuna. Caso contrário, os dados podem se tornar um passivo, cujo custo de aquisição e gerenciamento excede muito o valor por eles produzido.

Um banco de dados é uma estante de coleção de dados que, tipicamente, descreve as atividades de uma ou mais organizações relacionadas. Por exemplo, um banco de dados de uma universidade poderia conter informações sobre:

- **Entidades**: alunos, professores, cursos e turmas.
- **Relacionamentos entre as entidades**: a matrícula dos alunos nos cursos, os cursos ministrados pelos professores, o uso de salas por cursos etc.

Um sistema de gerenciamento de banco de dados, ou SGBD, é um *software* projetado para auxiliar na manutenção e utilização de vastos conjuntos de dados. A necessidade de tais sistemas, assim como o seu uso, tem crescido rapidamente. A alternativa para não se usar um SGBD é armazenar os dados em arquivos e escrever um código específico do aplicativo para gerenciá-los. O uso de um SGBD tem diversas vantagens importantes, como veremos a seguir.

Como gerenciar esses dados

Os bancos de dados e suas tecnologias têm influenciado na forma como usamos nossos dispositivos eletrônicos e suas soluções, entre eles, os computadores, *smartphones*, os sistemas da informação e os mais diversos aplicativos, dentre os quais estão as redes sociais que, munidas de grandes algoritmos, têm influenciado as pessoas a consumir produtos. Grande parte dessas estratégias só são possíveis pois existe uma enorme base de dados. Diante desse novo paradigma da informação, modelos de como gerenciar essas informações são fundamentais e desempenham um papel crucial em quase todas as áreas em que os computadores são usados, incluindo negócios, comércio eletrônico, áreas de engenharia, medicina, biomedicina, direito, educação, biblioteconomia e genética. Diversos tipos de sistemas de gerenciamento de banco de dados estão em uso, mas, neste material, nos concentraremos nos **sistemas de banco de dados relacionais (SGBDRs)**, que constituem o tipo dominante de SGBD atualmente.

Elmasri e Navathe (2011) definem que um banco de dados é uma coleção de dados relacionados. Dados são traduzidos em fatos conhecidos que podem ser registrados e possuem significado implícito. Por exemplo, considere os nomes, números de telefone e endereços das pessoas que você conhece. Você pode ter registrado esses dados em uma agenda ou, talvez, os tenha armazenado em um disco rígido, usando um computador pessoal e um *software* como Microsoft Access ou Excel. Essa coleção de dados relacionados, com um significado implícito, é um banco de dados.

Lembrete

Podemos usar recursos de um SGBD para gerenciar os dados de forma robusta e eficiente. Conforme o volume de dados e o número de usuários aumentam, centenas de *gigabytes* de dados e milhares de usuários passam a consumir recursos do SGBD, para tanto é preciso que o *software* tenha suporte para gerenciar esse volume de dados.

Observação

Um *kilobyte* (KB) representa 1024 *bytes*, um *megabyte* (MB) representa 1024 KBs, um *gigabyte* (GB) representa 1024 MBs, um *terabyte* (TB) representa 1024 GBs e um *petabyte* (PB) representa 1024 *terabytes*.

Um banco de dados normalmente é mais restrito e tem funcionalidades específicas no âmbito do gerenciamento. Entre essas funcionalidades temos as propriedades implícitas a seguir:

- Representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de dados. As mudanças no minimundo podem ser representadas em forma de dados.
- Representa uma coleção lógica, pois os dados possuem significado inerente. Existe uma variedade aleatória desses dados, que pode ser corretamente chamada de **banco de dados**.
- Projetado, construído e populado com dados para uma finalidade específica. Possui grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados.

1.1 Histórico

Na década de 1960, os sistemas de informação já dominavam os sistemas de arquivos e, no início dos anos 1970, as organizações passaram a utilizar os sistemas de gerenciamento de banco de dados (SGBD) de forma gradativa. Para manter os SGBDs em funcionamento, muitas organizações criaram o cargo DBA (DBA vem do inglês *database administrator*), que seria o administrador de banco de dados. Além disso, novos departamentos de administração de banco de dados foram criados para supervisionar e controlar as atividades de seu ciclo de vida nos sistemas. Isso se repetiu nos departamentos de tecnologia da informação (TI) e de gestão de recursos de informação, que têm sido reconhecidos por grandes organizações como sendo fundamentais para o gerenciamento comercial bem-sucedido. Os dados estão

sendo considerados e utilizados no meio corporativo e, consequentemente, o gerenciamento e controle desses dados são importantes para o trabalho eficaz da organização.

Em outras palavras, podemos dizer que a computação tem ganhado muitas funções nas organizações e, com isso, houve um aumento na necessidade de manter grande volume de dados disponíveis em um estado atualizado a cada minuto. Com a crescente complexidade desses dados e das aplicações, relacionamentos complexos entre os dados precisam ser modelados e mantidos. Devido a essas mudanças, a consolidação de recursos de informação se tornou uma tendência em muitas organizações.

Os sistemas de banco de dados se tornaram componentes integrais nos sistemas de informação e a sua utilização é imprescindível para que os dados possam ser gerenciados. A seguir, vemos alguns dos principais recursos que eles oferecem:

- Tudo em um único banco de dados, com a integração de dados em várias aplicações.
- Desenvolvimento de novas aplicações possuem suporte da forma rápida, usando linguagens de alto nível, como a SQL.
- Gerentes possuem suporte casual para navegação e consulta, além de oferecer suporte para o processamento principal de transação em nível de produção para os clientes.

Sistemas computadorizados precisam de recursos que incluem os próprios dados, o *software* de SGBD, o *hardware* do sistema de computação e o meio de armazenamento, o pessoal que usa e gerencia os dados (DBA, usuários finais e assim por diante), os programas de aplicação (*software*) que acessam e atualizam os dados e, claro, os programadores que desenvolvem essas aplicações. Isso nos faz perceber que o sistema de banco de dados é parte de um sistema de informação organizacional muito maior.

Para se projetar um banco de dados precisamos analisar o ciclo de vida típico de um sistema de informação e como os bancos serão encaixados nele. O ciclo de vida do sistema de informação tem sido chamado de **ciclo de vida macro**, enquanto o ciclo de vida do sistema de banco de dados tem sido chamado de **ciclo de vida micro**. O ciclo de vida macro, normalmente, inclui as seguintes fases:

1) Estudo de viabilidade: nessa etapa, realizamos uma análise das aplicações em potencial, identificamos a economia da coleta e disseminação de informações, realizando estudos preliminares de custo-benefício, determinando, assim, a complexidade de dados e processos, e estabelecendo prioridades entre as aplicações.

2) Levantamento e análise de requisitos: detalhamos os requisitos do sistema, definindo a interação com os usuários em potencial e identificando seus problemas e necessidades em particular. Todos os procedimentos de dependências entre aplicações, comunicação e relatório serão identificados.

3) Projeto: será dividido em dois aspectos – o projeto do sistema de banco de dados e o projeto dos sistemas de aplicação (programas), os quais utilizarão o processamento de banco de dados por meio de recuperações e atualizações.

4) Implementação: o sistema de informação é implementado, o banco de dados é carregado e as transações deste são implementadas e testadas.

5) Validação e teste de aceitação: nessa etapa, a aceitação do sistema deve atender aos requisitos dos usuários e os critérios de desempenho são validados. O sistema será testado conforme os critérios de desempenho e especificações de comportamento.

6) Implantação, operação e manutenção: antes que o novo sistema seja implantado, a validação precedida pela conversão de usuários do sistema mais antigo deve ser acompanhada, bem como o treinamento do usuário. A operacionalização começa quando todas as funções do sistema estiverem em funcionamento e forem validadas.

Novas requisições e/ou novas aplicações devem passar pelas fases anteriores até que sejam validadas e incorporadas ao sistema. O monitoramento do desempenho do sistema e a sua manutenção são atividades importantes durante a fase operacional.

O ciclo de vida do sistema de aplicação de banco de dados é composto pelo projeto e sua implementação. O problema do projeto de banco de dados pode ser declarado da seguinte forma:

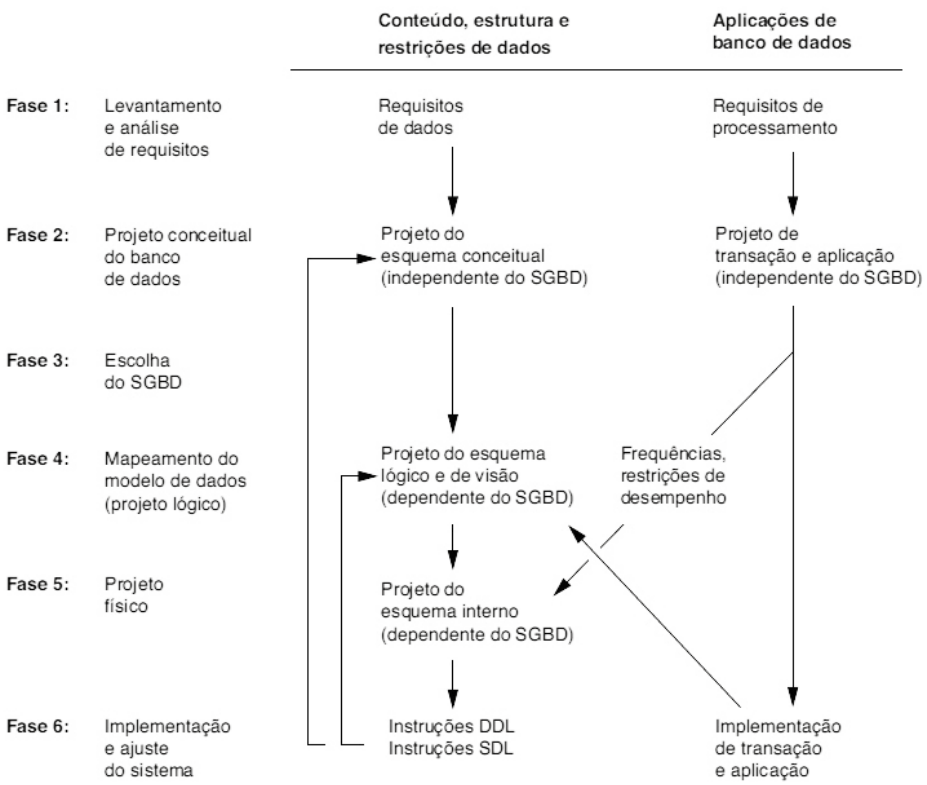

Figura 1 – Fases de projeto e implementação para grandes bancos de dados

As partes lógica e física devem ser projetadas em um ou mais bancos de dados para colecionar todas as informações necessárias dos usuários da instituição para um conjunto definido de sistemas. Os objetivos do projeto de banco de dados são múltiplos:

- Satisfazer os requisitos de conteúdo de informação dos usuários e aplicações especificadas.
- Oferecer a estruturação da informação de forma natural e fácil de entender.
- Definir suporte aos requisitos de processamento bem como outros objetivos de desempenho, como tempo de resposta, tempo de processamento e espaço de armazenamento.

Há um grau de dificuldade em realizar e medir, e isso envolve uma escolha inerente. Por exemplo, para tentar obter mais naturalidade e compreensibilidade do modelo, pode ter custos no desempenho. O problema é agravado porque o processo de projeto de banco de dados, normalmente, começa com requisitos informais e incompletos. De modo contrário, o resultado da atividade de projeto é um esquema rígido, que não pode ser facilmente modificado quando o banco de dados é implementado. Podemos identificar seis fases principais do processo geral de projeto e implementação do banco de dados:

1) Levantamento e análise de requisitos.

2) Projeto conceitual do banco de dados.

3) Escolha de um SGBD.

4) Mapeamento do modelo de dados (também chamado de projeto lógico do banco de dados).

5) Projeto físico do banco de dados.

6) Implementação e ajuste do sistema de banco de dados.

Conforme pode ser visto na figura anterior, as seis fases não prosseguem estritamente em sequência. Em muitos casos, podemos ter que modificar o projeto de uma fase mais antiga para outra mais recente. Esses ciclos de retorno entre as fases — e também dentro delas — são comuns.

Saiba mais

Para conhecer mais sobre as fases do ciclo de vida de um projeto de implementação de banco de dados, recomendamos a leitura a seguir:

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*: fundamentos e aplicações. 4. ed. São Paulo: Pearson, 2011.

Para aprofundar o seu estudo sobre os conceitos gerais de sistemas de gerenciamento de banco de dados, recomendamos a leitura a seguir:

RAMAKRISHNAN, R.; GEHRKE, E. J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: McGraw-Hill, 2011.

1.2 Importância dos sistemas de bancos de dados nas organizações

Um SGBD deve ser considerado como fator fundamental para o sucesso de uma organização onde o gerenciamento dos dados deve ser administrado como um bem maior no meio corporativo. Os gerentes devem compreender o valor das informações, ou seja, dos dados processados. Existem empresas cujo produto é a informação e cujo sucesso é uma função exclusiva do gerenciamento de informações. Em contrapartida, é preciso pensar sobre as formas que vários agentes se comunicam (usuários ou programas), com o objetivo de atender às necessidades de dados de diversas aplicações, permitir o desenvolvimento de aplicações que utilizem um BD, assim como possibilitar que aspectos de performance possam ser otimizados, conforme a demanda de acesso a dados pelas aplicações.

Os dados são utilizados por pessoas diferentes, em departamentos variados e por diversos motivos. Portanto, os agentes de integração de gerenciamento de dados devem tratar o conceito de dados compartilhados. Os agentes de integração de sistemas se enquadram nos seguintes itens:

- A interpretação e a apresentação de dados em formatos úteis, transformando os dados brutos em informação.
- A distribuição de dados e informações para as pessoas certas no momento certo.
- A preservação dos dados e o monitoramento de sua utilização por períodos adequados.
- O controle da duplicação e da utilização de dados, tanto interna como externa.

Independentemente do tipo de organização, o papel predominante do banco de dados é dar suporte para a tomada de decisões gerenciais em todos os níveis da organização e preservar a privacidade e a segurança dos dados.

A estrutura gerencial de uma organização pode ser dividida em três níveis: alto, médio e operacional. O gerenciamento de alto nível toma decisões estratégicas, o de nível médio toma as decisões táticas e o gerenciamento operacional toma decisões operacionais. Estas últimas são de curto prazo e afetam apenas as operações diárias, a exemplo da decisão de alterar o preço de um produto para acabar com seu estoque. As decisões táticas envolvem um período de tempo maior e afetam operações de maior escala; por exemplo, alterar o preço de um produto em resposta às pressões competitivas. As decisões estratégicas são as que afetam o bem-estar da empresa a longo prazo, ou, mesmo, a sua sobrevivência; por exemplo, alterar a estratégia de precificação das linhas de produtos para capturar uma parcela maior do mercado.

O SGBD deve oferecer ferramentas que forneçam, em cada nível de gerenciamento, uma visão útil dos dados, que deem o suporte necessário para a tomada dessas decisões. Veremos, a seguir, as atividades que são comuns em cada nível.

No alto nível de gerenciamento, o banco de dados deve ser capaz de:

- Fornecer as informações necessárias para a tomada de decisões estratégicas, o planejamento estratégico, a formulação de políticas e a definição de metas.

- Fornecer acesso aos dados externos e internos para identificar oportunidades de crescimento e traçar a direção do crescimento. A direção refere-se à natureza das operações. Por exemplo, se a empresa se tornará uma organização de serviços, de fabricação ou a combinação de ambas.

- Fornecer um modelo de definição à aplicação de políticas organizacionais. (Lembre-se de que essas políticas são traduzidas em regras de negócios nos níveis inferiores da organização.)

- Aumentar a probabilidade de retorno positivo sobre o investimento na empresa, buscando novos modos de reduzir custos e/ou impulsionar a produtividade.

- Fornecer *feedback* para monitorar a empresa e auxiliá-la a atingir suas metas.

No nível médio de gerenciamento, o banco de dados deve ser capaz de monitorar se a empresa está atingindo as suas metas:

- Oferecer os dados necessários para decisões táticas e de planejamento.

- Monitorar e controlar a alocação e utilização dos recursos da empresa, além de avaliar o desempenho dos diferentes departamentos.

- Fornecer um modelo de aplicação e garantir a segurança e privacidade dos dados no banco. Segurança, nesse caso, significa proteger os dados da utilização acidental ou intencional por usuários não autorizados. A privacidade lida com os direitos individuais e da organização de determinar "quem, o que, quando, onde e como" em relação ao acesso sobre esses dados.

No nível operacional de gerenciamento, o banco de dados deve ser capaz de:

- Representar e dar suporte às operações da empresa do modo mais fiel possível. O modelo de dados deve ser flexível o suficiente para incorporar todos os dados necessários, presentes e esperados.

- Produzir resultados de consultas dentro de níveis especificados de desempenho. Lembre-se de que as exigências de desempenho aumentam em níveis inferiores de gerenciamento e operação. Assim, o banco de dados deve dar suporte a respostas rápidas para um número maior de transações no nível de gerenciamento operacional.

- Aprimorar a capacidade operacional de curto prazo da empresa, fornecendo informações oportunas para o suporte ao cliente e para o desenvolvimento de aplicações e operações computacionais.

O objetivo geral de qualquer banco de dados é fornecer um fluxo contínuo de informações para toda a empresa.

O banco de dados da empresa também é conhecido como **banco corporativo** ou **empresarial**. O banco de dados empresarial pode ser definido como a representação dos dados de uma empresa que fornece suporte a todas as operações presentes e esperadas no futuro. A maioria das organizações bem-sucedidas, hoje em dia, depende do banco de dados empresarial para fornecer suporte a todas as operações do projeto, considerando a implementação, das vendas aos serviços e da tomada diária de decisões ao planejamento estratégico.

1.3 Banco de dados em níveis organizacionais

Ter um sistema de gerenciamento de banco de dados computadorizado não garante que os dados sejam utilizados adequadamente para fornecer as melhores soluções exigidas pelos gerentes. O SGBD é uma ferramenta de gerenciamento de dados. Como qualquer ferramenta, deve ter a capacidade de permitir o gerenciamento de dados e ser utilizada de modo eficiente para produzir os resultados desejados. Considere a seguinte analogia: nas mãos de um carpinteiro, o martelo pode ajudar a produzir móveis, porém a mesma ferramenta, se colocada nas mãos de uma criança, pode causar danos. A solução para os problemas de uma empresa não é a mera existência de um sistema de computadores ou de um banco de dados, mas a eficiência de seu gerenciamento e utilização.

Uma organização é descrita como um processo que inclui três aspectos importantes:

- **Tecnológico**: *software* e *hardware* do SGBD.
- **Gerencial**: funções administrativas.
- **Cultural**: resistência da corporação à mudança.

O **aspecto tecnológico** inclui a seleção, a instalação, a configuração e o monitoramento do SGBD, para assegurar que ele trate de modo eficiente o armazenamento, o acesso e a segurança de dados. As pessoas encarregadas de cuidar dos aspectos tecnológicos da instalação do SGBD devem ter as habilidades técnicas necessárias para fornecer e assegurar o suporte adequado aos diferentes usuários desse sistema: programadores, gerentes e usuários finais.

O **aspecto gerencial** da introdução do SGBD em uma organização exige planejamento cuidadoso, para criar uma estrutura organizacional adequada e acomodar as pessoas responsáveis pela administração do sistema. A estrutura organizacional também deve ter excelentes habilidades interpessoais e de comunicações, combinadas com a ampla compreensão dos negócios da organização.

O gerenciamento de alto nível deve estar comprometido com o novo sistema, além de definir e dar suporte às funções de administração de dados, metas e papéis no interior da organização.

O **aspecto cultural** da introdução de um sistema de banco de dados deve ser avaliado cuidadosamente. A existência do SGBD, provavelmente, terá um efeito sobre as pessoas, funções e interações. Por exemplo, é possível que novas pessoas sejam adicionadas, novos papéis sejam atribuídos ao pessoal existente e o desempenho dos funcionários seja avaliado por meio de novos padrões.

1.4 Uma breve introdução sobre banco de dados relacionais e NoSQL

Os bancos de dados relacionais são, há décadas, os modelos de persistência mais usados na maioria dos aplicativos de *software*. Do ponto de vista dos benefícios dos bancos de dados relacionais, podemos ressaltar:

- **Persistência de dados**: uma das missões mais importantes dos bancos de dados é poder manter enormes quantidades de dados persistentes. Na maioria das aplicações, é necessário que se tenha um depósito de dados que atue como um *backup*. Esses depósitos podem ser estruturados de várias maneiras, por exemplo, um arquivo no sistema de arquivos do sistema operacional. No entanto, esses dados oferecem mais flexibilidade do que o sistema de arquivos e isso permite obter conjuntos de informações de uma maneira rápida e fácil.

- **Concorrência**: nas aplicações normais, em geral, muitas pessoas estão trabalhando nos mesmos dados e podem estar modificando os mesmos dados. Essa situação obriga a coordenar as diferentes interações nos mesmos dados para evitar inconsistências neles. O gerenciamento direto de simultaneidade é complicado e existem muitas possibilidades de cometer erros.

- **Integração**: os aplicativos, geralmente, exigem interação com outros aplicativos de maneiras que compartilham os mesmos dados e ambos usam os dados que foram modificados pelo restante dos aplicativos. Uma forma de implementar essa interação é usar uma integração com base no compartilhamento de um banco de dados, ou seja, os diferentes aplicativos que interagem armazenam seus dados em um banco de dados único. O uso de um único banco de dados permite que todos os aplicativos usem os dados mais facilmente. Além disso, o sistema de controle de concorrência do banco de dados permite a existência de vários aplicativos usando o mesmo banco de dados.

- **Padronização**: bancos de dados relacionais tiveram sucesso, porque fornecem muitos benefícios de constituir um sistema padrão. Uma pessoa pode obter conhecimentos gerais sobre o modelo relacional que são comuns a diferentes bancos de dados relacionais e aplicá-los em qualquer sistema.

No entanto, bancos de dados relacionais não são perfeitos. Um dos principais problemas refere-se à diferença entre o modelo relacional e as estruturas de dados de memória, que alguns autores denominam como "incompatibilidade". O modelo relacional organiza os dados em uma estrutura de tabela e linha (também chamado relações e tuplas). No modelo relacional, uma tupla é um conjunto de pares nome-valor e uma relação é um conjunto de tuplas. Todas as operações no SQL consomem e retornam relacionamentos de acordo com a álgebra relacional. O uso de relacionamentos proporciona certa elegância e simplicidade, mas introduz um conjunto de limitações. Em particular, valores em uma tupla relacional precisam ser simples (eles não podem ter como estrutura um registro aninhado ou uma lista). Essa limitação não é dada nas estruturas de dados da memória, onde existe um conjunto de tipos de estruturas muito mais ricos que as relações. Como consequência, para usar uma estrutura de dados mais complexos que o relacionamento, eles devem ser traduzidos em uma representação relacional, para poder armazená-los em disco.

O termo NoSQL, "não relacional" ou "não SQL", foi primeiramente utilizado em 1998 (entretanto, o termo "banco de dados não relacionais" existe desde 1960) como o nome de um banco de dados não relacional de código aberto e pode ser explicado como um termo genérico que representa os bancos de dados não relacionais. Trata-se de uma classe definida de banco de dados que fornecem um mecanismo para armazenamento e recuperação de dados que são modelados de formas diferentes das relações tabulares usadas nos bancos de dados relacionais. Em outras palavras, quanto maior o tamanho exigido, maior será a quantidade de equipamento e, por consequência, de mão de obra especializada para a sua correta manutenção.

Existem várias famílias de produtos NoSQL e cada família compartilha de um mesmo modelo de armazenamento. O mais tradicional, em vez de possuir uma tabela com um número fixo de colunas tipadas, trabalha com o conceito de chave-valor, ou KVS (acrônimo para *key-value store*). Isso significa que em vez de incluir um conjunto de atributos, a operação insere apenas uma chave e um valor e nada mais. Algumas implementações do tipo KVS são Couchbase, Kyoto Cabinet, Redis e Amazon DynamoDB.

Já no modelo de armazenamento de colunas ordenadas, baseados no modelo Bigtable da Google, em vez de o dado ser orientado por linha, ele é orientado por coluna. São implementações desse modelo HBase, mantido pela fundação Apache, e HyperTable da Cloudata.

Algumas das características compartilhadas pelos bancos de dados do NoSQL são:

- Eles não usam SQL como uma linguagem de consulta, no entanto, alguns deles usam linguagens de consulta semelhantes à SQL, como CQL, no Cassandra.

- Em geral, esses são projetos de código aberto.

- Muitos dos bancos de dados NoSQL nasceram da necessidade de executar em um *cluster*, o que influenciou tanto seu modelo de dados como sua abordagem à consistência. Bancos de dados relacionais usam transações ACID para gerenciar a consistência do banco de dados, no entanto, isso colide com um ambiente em *cluster*, portanto eles oferecem várias opções para implementar consistência e distribuição. Contudo, nem em todos os bancos de dados o NoSQL é orientado para executar em um *cluster*.

- Os bancos de dados NoSQL não possuem um esquema fixo.

De todos os recursos dos bancos de dados NoSQL, destaca-se a inexistência de um esquema. Quando guardar dados em um banco de dados relacional, a primeira coisa a fazer é definir um esquema que indica quais tabelas existem, quais são as colunas de cada tabela e que tipo de dados cada coluna possui.

No entanto, os bancos de dados NoSQL operam sem esquema, o que permite adicionar livremente campos aos registros do banco de dados sem precisar redefinir a estrutura.

Devido às limitações mencionadas na seção anterior, algumas empresas decidiram usar alternativas aos bancos de dados relacionais. Foi o caso da Google e da Amazon, que desenvolveram sistemas de

armazenamento baseados no uso de *clusters*: grandes tabelas da Google e Dynamo da Amazon. Esses sistemas permitem gerenciar grandes quantidades de dados em ambientes distribuídos e realizam seu processamento, razão pela qual são considerados a origem dos chamados **bancos de dados NoSQL**.

Saiba mais

Para ver mais detalhes sobre banco de dados noSQL, visite o *site*:

Disponível em: https://bit.ly/3xHQ4QK. Acesso em: 5 maio 2021.

2 MODELAGEM DE DADOS

Quando falamos de modelagem, precisamos ter em mente que a escolha de um modelo de dados, muitas vezes, é feita para resolver um problema específico. Alguns mecanismos de *software* executam essa tarefa com todos os elementos necesssários, ou seja, capturam arquivos do sistema, transformando-os em item de dados, que é a menor unidade de dados identificável e que possui significado sabido. Por exemplo: caso necessite descrever dados de uma pessoa, podemos resgistrar esses dados como: nome, sobrenome, n. de CPF, idade etc. Um grupo de itens de dados relacionados, tratados como uma unidade isolada por uma aplicação, é chamado de **registro**. Alguns exemplos de tipos de registros são: pedido, vendedor, cliente, produto e departamento. Um arquivo pode ser definido com uma estrutura de campos de registro com a mesma tipicidade. Os sistemas de banco de dados tomam como referência os arquivos que possuem descrição mais detalhada. Em um banco de dados relacional, esse dado é entendido como uma coluna ou atributo, que pode ainda ser chamado de linha ou tupla; e um arquivo chamado de tabela.

Segundo Ramakrishnan e Gehrke (2011), um banco de dados é um objeto mais complexo, é uma coleção de dados armazenados e inter-relacionados, que atende às necessidades de vários usuários dentro de uma ou mais organizações, ou seja, coleções inter-relacionadas de muitos tipos diferentes de tabelas. As motivações para usar bancos de dados em vez de arquivos incluem acesso mais amplo a um conjunto diversificado de usuários, a integração de dados para facilitar o acesso e a atualização de transações complexas, além da menor redundância de dados.

De acordo com os autores, definido como um sistema de *software* genérico para manipular dados, o sistema gerenciador de banco de dados (SGBD) admite uma visão lógica (esquema, subesquema), visão física (métodos de acesso, *clustering* de dados), linguagem de definição de dados, linguagem de manipulação de dados e utilitários importantes, como gerenciamento de transação e controle de concorrência, integridade de dados, recuperação de falhas e segurança. Os SGBDs relacionais, tipo dominante de sistemas que apoiam bancos de dados de negócios bem definidos, também fornecem maior grau de independência de dados que os SGBDs hierárquicos e de rede (CODASYL), mais antigos. A independência de dados é a capacidade de fazer mudanças na estrutura lógica ou física do banco de dados sem exigir reprogramação dos programas de aplicação. Ela também facilita bastante a conversão

e a reorganização do banco de dados. Os SGBDs relacionais fornecem um grau de independência de dados muito mais elevado que os sistemas anteriores e são o foco da nossa discussão sobre modelagem de dados.

A concepção do projeto de banco de dados e sua implementação vai dar suporte ao sistema que será desenvolvido, que passa por diferentes estágios de abstração. Existe muita pressão para que a equipe de desenvolvimento entregue o projeto o quanto antes, para que novos sistemas entrem em funcionamento. Assim, não é incomum tomar conhecimento de que as etapas de levantamento de requisitos e modelagem foram negligenciadas e jogadas em segundo plano. Esse é um erro comum que costuma ter efeito financeiro, em geral, muito oneroso ao projeto, principalmente nas etapas seguintes do processo de desenvolvimento. Vale fazermos, aqui, a recomendação de nunca negligenciar nenhuma das etapas ao projetar um banco de dados. A figura a seguir mostra uma possível representação do modelo de banco de dados de produto/cliente na mente do usuário final.

Dada a complexidade das informações, o projeto de banco de dados deve seguir três níveis de abstração. Conforme veremos a seguir.

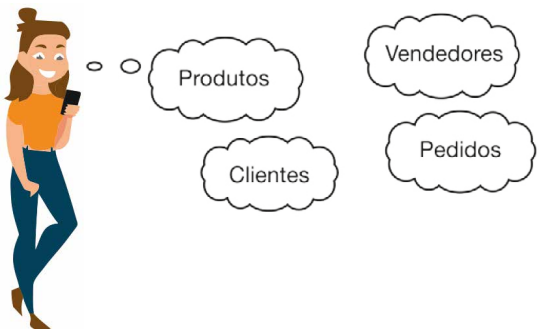

Figura 2 – Análise de requisitos (realidade)

2.1 Modelo conceitual (DER)

Conhecido como **diagrama entidade-relacionamento**, trata-se de um modelo de dados abstrato que descreve a estrutura de um banco de dados independente de sua implementação. O modelo conceitual registra quais dados podem aparecer no banco de dados, mas não como esses dados devem ser armazenados em nível de SGBD. A técnica de modelagem conceitual mais difundida é a abordagem entidade-relacionamento (ER). Essa técnica permite criar um modelo conceitual que é muito conhecido como diagrama entidade-relacionamento (DER). A figura a seguir simula um exemplo de dados *produto* e a entidade associada à categoria.

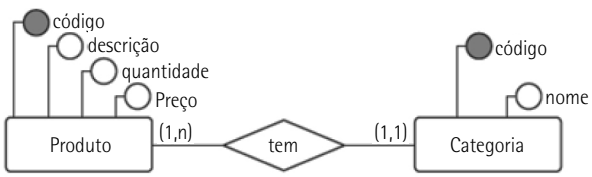

Figura 3 – Exemplo de um modelo conceitual

2.2 Modelo lógico (esquema do BD)

O modelo lógico tem como objetivo transformar o modelo conceitual em um modelo que define como o banco de dados será implementado em um SGBD específico. Deve representar relações e restrições do modelo de dados que representa a estrutura de um BD e o esquema do banco de dados. A figura a seguir exemplifica o modelo lógico a partir do exemplo de modelo conceitual da figura 3.

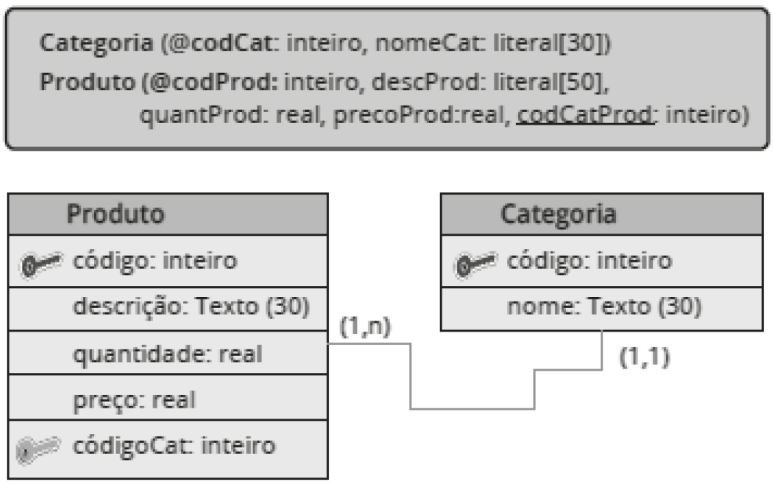

Figura 4 – Exemplo de um modelo lógico

2.3 Modelos físicos

Modelos físicos são popularmente conhecidos como ***Script* do BD em SQL**. Representam o modelo do banco de dados enriquecido com detalhes que influenciam no desempenho, mas não interferem em sua funcionalidade. *Script* do banco de dados em SQL representa os detalhes dos dados internamente ao BD (campo, tipo/domínio, restrições).

```
--criação da tabela TB_PRODUTO
CREATE TABLE TB_PRODUTO (
codigo INTEGER PRIMARY KEY,
quantidade REAL,
preco REAL,
descricao  VARCHAR(30),
codigocat INTEGER
);

--criação da tabela TB_CATEGORIA
CREATE TABLE TB_CATEGORIA (
codigocat INTEGER PRIMARY KEY,
descricao VARCHAR(30)
);
ALTER TABLE TB_PRODUTO ADD FOREIGN KEY(codigocat) REFERENCES
TB_CATEGORIA (codigocat);
```

Figura 5 – Exemplo de um modelo físico

Introdução ao MER (modelo entidade-relacionamento)

Uma breve introdução à modelagem de dados entidade-relacionamento (ER) irá permitir descrever os dados que uma empresa possui e como os objetos do mundo real e seu relacionamento serão utilizados amplamente para desenvolver um projeto inicial de banco de dados. A modelagem fornece conceitos úteis que nortearão o mover de uma descrição informal do que os usuários desejavam de seu banco de dados para uma descrição mais detalhada e precisa que pode ser implementada em um SGBD.

2.4 Projeto de banco de dados e diagramas ER

O projeto de banco de dados observa que esta é, tipicamente, apenas uma parte, embora seja uma parte central nos aplicativos que fazem uso intensivo de dados, em um projeto maior de sistema de *software*. Nosso foco primário é o projeto do banco de dados e, por enquanto, não discutiremos outros aspectos do projeto de *software*.

A produção de um projeto de banco de dados é dividida em seis etapas. O modelo ER é o mais relevante nas três primeiras etapas.

Segundo Ramakrishnan e Gehrke (2011), podemos seguir as etapas dispostas aqui:

1) Análise de requisitos: fase inicial de um projeto de banco de dados, que permite compreender quais dados devem ser armazenados, quais informações os aplicativos devem manipular, quais são as operações mais importantes e quais estão sujeitas a requisitos de desempenho. Deve-se descobrir o que os usuários desejam do banco de dados. Gerida por reuniões informais que, geralmente, envolvem discussões com grupos de usuários, estudo do ambiente operacional em vigor, visando definir quais alterações serão trabalhadas, a análise de toda a documentação disponível sobre os aplicativos existentes que se deseja substituir ou complementar com o banco de dados e assim por diante. Muitas metodologias são propostas com o foco de organizar e apresentar as informações coletadas, quase sempre são usadas ferramentas automatizadas.

2) Projeto conceitual do banco de dados: nessa etapa, as coletas de dados da fase de análise de requisitos serão utilizadas para fornecer uma descrição dos dados a serem armazenados, levando em consideração as informações mais importantes. No modelo entidade-relacionamento (ER), disponibilizam-se diversos modelos de dados semânticos, ou de alto nível, utilizados no projeto de banco de dados. Tem como objetivo criar uma descrição simples dos dados que melhor corresponda à visão ou ideia que os usuários e desenvolvedores têm em relação aos dados (e às pessoas e processos a serem representados nos dados).

3) Projeto lógico de banco de dados: a partir da conversão do projeto conceitual de banco de dados, um SGBD será escolhido para implementar as informações de um esquema ER para um esquema de banco de dados relacional.

As três etapas restantes do projeto de banco de dados

Nessa fase, uma análise mais minuciosa tem como objetivo refinar o esquema lógico, o qual, a partir da etapa 3, será direcionado para o novo esquema. A partir do esquema lógico, novas funções devem receber atenção, como o desempenho e os protótipos do esquema físico, seguido por aspectos de segurança, que assegurem que os usuários sejam capazes de acessar os dados de que eles precisam, mas não os dados que desejamos ocultar deles.

4) Refinamento do esquema: nessa etapa, o projeto de banco de dados consiste em analisar a coleção de relações em nosso esquema de banco de dados relacional para identificar problemas em potencial e refiná-los. Contrasta com as etapas de análise de requisitos e projeto conceitual, mas é importante lembrar que elas são essenciais ao esquema de refinamento.

5) Projeto físico de banco de dados: nessa etapa, todas as alterações devem suportar um refinamento ainda maior do projeto de banco de dados, para assegurar que este satisfaça os critérios de desempenho desejados. A criação de índices em algumas tabelas e o agrupamento de tabelas serão inicializados nessa fase, podendo envolver um reprojeto substancial de partes do esquema de banco de dados obtido nas etapas anteriores do projeto.

6) Projeto de aplicativos e segurança: a partir dessa etapa, novas ferramentas e metodologias podem ser implementadas ao SGBD, devendo considerar aspectos do aplicativo que vão além do banco de dados propriamente dito. Nessa fase deve-se identificar as entidades (por exemplo, usuários, grupos de usuários, departamentos) e os processos envolvidos no aplicativo. Deve-se descrever o papel de cada entidade em cada processo que é refletido em alguma tarefa de aplicativo, como parte de um fluxo dessa tarefa, identificando as partes do banco de dados que devem ser acessíveis e as partes que não devem ser acessíveis, e quais medidas devem ser criadas para assegurar que essas regras de acesso sejam obedecidas.

Para Ramakrishnan e Gehrke (2011), uma divisão do processo de projeto deve ser vista como algo benéfico. Embora possamos começar com um processo com as seis etapas esboçadas aqui, o projeto completo de banco de dados provavelmente exigirá uma fase de sintonização (*tuning*) subsequente, na qual todos os seis tipos de etapas de projeto são intercalados e repetidos até que o projeto esteja satisfatório.

2.5 Entidades, atributos e conjuntos de entidades

Uma **entidade** é um objeto do mundo real, que pode ser descrita por qualquer pessoa, seja ela concreta ou abstrata. Como exemplos, podemos citar: um carro popular, a concessionária, o gerente da loja, filiais que vendem o mesmo modelo do veículo. Normalmente, é útil identificar uma coleção de entidades semelhantes.

Uma coleção é chamada de **conjunto de entidades**. Um conjunto de entidades não precisa ser disjunto. Por exemplo, o conjunto de funcionários da concessionária e o conjunto de funcionários da manutenção de veículos, ambos podem conter o funcionário Roberto Marrotti (que pode trabalhar em ambos os departamentos). Poderíamos também definir um conjunto de entidades chamado Funcionarios que contém os conjuntos de funcionários dos dois departamentos: de vendas e manutenção.

Um **conjunto de atributos** é representado por uma entidade. Essas entidades representam um determinado conjunto de atributos com as mesmas características. É necessário escolher quais atributos refletem o nível de detalhes com os quais desejamos representar as informações sobre as entidades. Por exemplo, no conjunto de entidades Funcionarios poderíamos usar: nome, código de pessoa física (CPF) e vaga de estacionamento como atributos. Nesse caso, armazenaremos o nome, CPF e o número da vaga do estacionamento para cada funcionário. Entretanto, não armazenaremos o endereço de um funcionário (ou o sexo ou a idade).

Os **atributos** são características de entidades. Por exemplo, a entidade ALUNO pode incluir, entre outros atributos, ALU_NOME, ALU_SOBRENOME, ALU_INICIAL. Na notação de Chen original, os atributos são representados por elipses e conectados ao retângulo da entidade por uma seta. Cada elipse contém o nome dos atributos que representa. Na notação de pé de galinha, os atributos são escritos na caixa de atributos.

Atributos necessários e opcionais

Um atributo necessário é aquele que deve apresentar valor. Em outras palavras, não pode ser deixado vazio. Como apresentado na figura a seguir, há dois atributos em negrito na notação pé de galinha. Isso indica que é necessária uma entrada de dados.

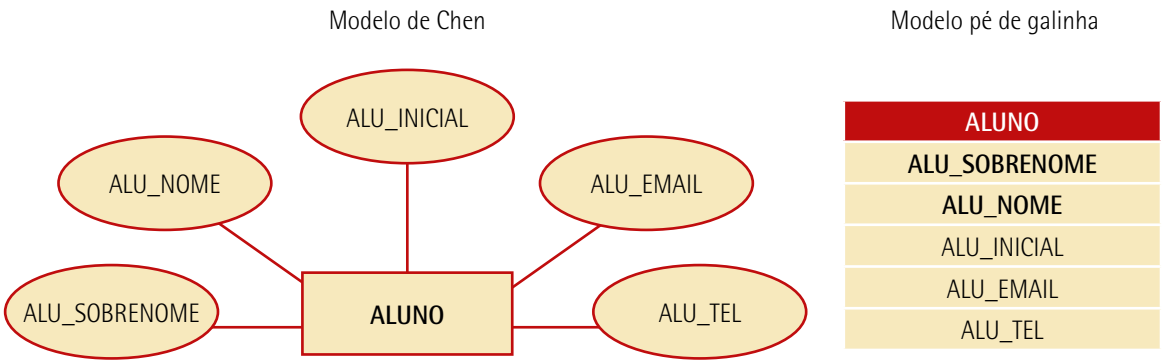

Figura 6 – Atributos da entidade ALUNO

O diagrama ER, conhecido pela notação pé de galinha, é amplamente utilizado no modelo entidade-relacionamento e foi definido por Peter Chen em 1976.

Saiba mais

A notação pé de galinha foi desenvolvida por Peter Chen na década de 1970. Para conhecer mais sobre esse diagrama, acesse:

LUCIDCHART. *O que é um diagrama entidade relacionamento?* [s.d.]. Disponível em: https://bit.ly/3uzFfxL. Acesso em: 7 abr. 2021.

No exemplo dado aqui, ALU_SOBRENOME e ALU_NOME exigem entradas de dados, devido ao pressuposto de que todos têm nome e sobrenome. Mas os alunos podem não ter um nome do meio e isso, talvez, não seja apresentado em negrito na caixa de entidades. Um atributo opcional é aquele que não exige um valor e, portanto, pode ser deixado vazio.

Lembrete

O conceito de força de relacionamento não faz parte do ER original. Ele se aplica diretamente aos diagramas pé de galinha. Como esses diagramas são amplamente utilizados no projeto de banco de dados relacionais, é importante compreender como a força do relacionamento afeta a implementação do banco de dados

Observação

A notação do DER de Peter Chen é orientada para a modelagem conceitual e, portanto, não distingue entre relacionamentos fortes e fracos.

Domínios

Um domínio é o conjunto de valores possíveis de um determinado atributo. Por exemplo, deseja-se conhecer a média das notas atribuídas a um aluno, que é apresentada como (2.4); para esse dado, sabemos que o resultado da média representa um número real, onde o menor valor possível da média é 2, enquanto o maior é 3, dependendo do sistema de conversão. O domínio do atributo de sexo consiste apenas nas duas possibilidades: M ou F (ou outro código equivalente). O domínio para o atributo de data de contratação de uma empresa consiste em todas as datas que se ajustem a uma determinação de faixa (por exemplo, da fundação da empresa à data atual).

Os atributos podem compartilhar um domínio. Por exemplo, o endereço de um aluno e o de um professor compartilham o mesmo domínio de todos os endereços possíveis. De fato, o dicionário de dados pode permitir que um novo atributo declarado herde as características de um existente, se for utilizado o mesmo nome de atributo. Por exemplo, cada uma das entidades PROFESSOR e ALUNO pode ter um atributo chamado ENDEREÇO e, portanto, é possível compartilharem um domínio.

Identificadores (chaves primárias)

O ER utiliza identificadores, ou seja, um ou mais atributos que identifiquem de modo exclusivo cada instância de entidade. No modelo relacional, esses identificadores são mapeados para chaves primárias (PKs) de tabelas. No DER, eles aparecem sublinhados. Os atributos chave também são sublinhados em uma notação simplificada da estrutura de tabela, utilizada conforme o seguinte formato:

NOME DA TABELA (**ATRIBUTO_CHAVE1**, ATRIBUTO2, ATRIBUTO3, ATRIBUTOK)

Por exemplo, uma entidade EQUIPAMENTO pode ser representada por:

EQUIPAMENTO (COD_EQUIPAMENTO, DESCR_EQUIPAMENTO, COD_CATEGORIA, QUANT_EQUIPAMENTO)

Cada EQUIPAMENTO é identificado por um único código de equipamento ou COD_EQUIPAMENTO.

Identificadores compostos

A princípio, um identificador de entidade é composto de um único atributo. Por exemplo, na tabela que aparece a seguir, utilizaremos uma chave primária de um único atributo chamado Cod_Classe (código da classe). No entanto, é possível utilizar um identificador composto, ou seja, uma chave primária composta de mais de um atributo. Por exemplo, o administrador do banco de dados do Colégio Big Brain pode decidir por identificar cada instância (ocorrência) da entidade TURMA utilizando a chave primária composta pela combinação Cod_Disciplina (código da disciplina) e Secao_Classe (seção da classe) em vez de utilizar Cod_Classe. Ambas as abordagens identificam de modo exclusivo cada instância de entidade. Dada a estrutura atual da tabela TURMA, que corresponde à tabela 1, Cod_Classe é a chave primária e a combinação entre Cod_Disciplina e Secao_Classe é uma chave candidata (Cod_Disciplina e Secao_Classe) que se torna uma chave primária composta aceita.

Se o Cod_Classe for utilizado como chave primária, a entidade TURMA pode ser representada de modo simplificado como:

TURMA (COD_CLASSE, COD_DISCIPLINA, SECAO_CLASSE, HORARIO_CLASSE, COD_QUARTO, PROF_NUM)

Por outro lado, se o Cod_Classe for excluído e a chave primária composta for a combinação de Cod_Disciplina e Secao_Classe, a entidade TURMA será representada por:

TURMA (**COD_DISCIPLINA**, **SECAO_CLASSE**, HORARIO_CLASSE, COD_QUARTO, PROF_NUM)

Observe que ambos os atributos de chave são sublinhados na notação da entidade.

Tabela 1 – Componentes e conteúdo da tabela (entidade) TURMA

Cod_Classe	Cod_Disciplina	Secao_Classe	Horario_Classe	Cod_Quarto	Prof_Num
10012	ACCT-211	1	8h às 8:50h	BUS311	342
10025	ACCT-211	2	9h às 9:50h	BUS201	342
10030	ACCT-212	3	10h às 10:50h	BUS111	105
10010	ACCT-210	2	7h às 8h	BUS100	105
10015	CIT-2001	2	7h às 8:40h	KLR207	22105
10017	CIT-2001	1	7h às 7:50h	KLR277	22106

Adaptada de: Rob e Coronel (2011).

Atributos simples e compostos

Os atributos são classificados como simples ou compostos. Atributo composto é aquele que pode ser subdividido de modo a se obter mais atributos. Por exemplo, o atributo endereço pode ser dividido em rua, cidade, estado, CEP. De modo similar, o atributo Numero_Telefone pode ser subdividido em código de área e prefixo. Um atributo simples é aquele que não pode ser subdividido. Por exemplo, idade, sexo e estado civil seriam classificados como atributos simples. Para facilitar consultas detalhadas, os atributos compostos devem ser convertidos em uma série de atributos simples.

Atributos monovalorados

Um atributo monovalorado é aquele que pode ter apenas um valor. Por exemplo, uma pessoa pode ter apenas um número de Seguro Social e uma peça fabricada tem um único número de série. Tenha em mente que um atributo monovalorado não é necessariamente um atributo simples. Por exemplo, o número de série de uma peça como SE-009-003-1988001 possui um único valor, mas é um atributo composto, pois pode ser subdividido em: região em que cada peça foi produzida (SE), planta dessa região (009), divisão dessa planta (003) e número da peça (1988001).

Atributos multivalorados

Atributos multivalorados são aqueles que possuem muitos valores. Por exemplo, uma pessoa pode ter vários graus de ensino, uma família pode ter diversos números de telefone. Ou, ainda, a cor de um carro, que pode ser subdividida em várias.

Para cada atributo associado a um conjunto de entidades, devemos identificar um domínio de possíveis valores. Por exemplo, o domínio do atributo nome de funcionários poderia ser o conjunto de cadeias de 20 caracteres. Como outro exemplo, se a empresa avalia funcionários em uma escala de 1 a 10 e armazena as avaliações em um campo chamado avaliação, o domínio associado consiste em números inteiros de 1 a 10. Além disso, para cada conjunto de entidades, escolhemos uma chave, que consiste em um conjunto mínimo de atributos cujos valores identificam unicamente uma entidade do conjunto. Pode haver mais de uma chave candidata e, se houver, designamos uma delas como a chave primária. Por enquanto, consideramos que cada conjunto de entidades contém, no mínimo, um conjunto de atributos que identifica unicamente uma entidade no conjunto inteiro, ou seja, o conjunto de atributos contém uma chave.

O conjunto de entidades **Funcionario** com atributos *CPF*, *nome* e *vaga* é ilustrado na próxima figura. Um conjunto de entidades é representado por um retângulo e cada atributo é representado por um símbolo oval ou circular. Cada atributo da chave primária é sublinhado. A informação do domínio poderia ser listada juntamente com o nome do atributo, mas omitimos isso para manter as figuras compactas. A chave é *CPF*.

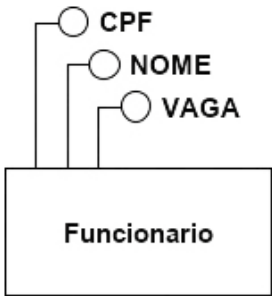

Figura 7 – Conjunto de entidades Funcionario

2.6 Relacionamentos e conjuntos de relacionamentos

Segundo Ramakrishnan e Gehrke (2011), um relacionamento é uma associação entre duas ou mais entidades. Por exemplo, podemos ter o relacionamento em que Roberto trabalha no departamento de farmácia. Como acontece com as entidades, podemos desejar reunir um conjunto de relacionamentos semelhantes em um conjunto de relacionamento. Um conjunto de relacionamentos pode ser considerado um conjunto de n-tuplas:

$$\{(e_1, ..., e_n) \mid e_1 \in E_1, ..., e_n, \in E_n\}$$

Em n-tupla, é visto um relacionamento envolvendo n entidades e_1 a e_n, sendo que a entidade e_i pertence ao conjunto de entidades. Na figura seguinte, ilustramos o conjunto de relacionamentos Trabalha_em, no qual cada relacionamento indica um departamento em que um funcionário trabalha. Observe que diversos conjuntos de relacionamentos podem envolver os mesmos conjuntos de entidades. Por exemplo, poderíamos também ter um relacionamento Gerencia envolvendo Funcionario e Departamento.

Em um relacionamento, ainda podemos ter atributos descritivos, nos quais os dados são usados para registrar informações sobre o relacionamento entre uma das entidades participantes. Por exemplo, podemos desejar registrar que Roberto trabalha no departamento de vendas de uma loja de roupas a partir de março de 2000. Essa informação é registrada na figura a seguir, adicionando-se um atributo, *desde*, a Trabalha_em. Um relacionamento deve ser identificado unicamente pelas entidades participantes, sem referência aos atributos descritivos. No conjunto de relacionamento Trabalha_em, por exemplo, cada relacionamento Trabalha_em deve ser identificado univocamente pela combinação de *CPF* de funcionário e *id-depto* de departamento. Assim, para um determinado par funcionário-departamento, não podemos ter mais de um valor *desde* associado.

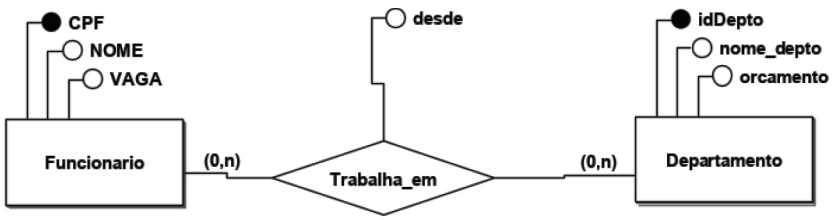

Figura 8 – Conjunto de relacionamentos Trabalha_em

Uma instância de um conjunto de relacionamentos é um conjunto de seus relacionamentos. Intuitivamente, uma instância pode ser considerada o "retrato" do conjunto de relacionamentos em determinado momento. Cada entidade Funcionário é denotada pelo seu CPF, e cada entidade Departamento é denotada pela sua *id-depto*, por simplicidade. O valor *desde* é ilustrado ao lado de cada relacionamento.

Os comentários "muitos-para-muitos" e "participação total", da figura a seguir, serão discutidos posteriormente, quando abordarmos as restrições de integridade.

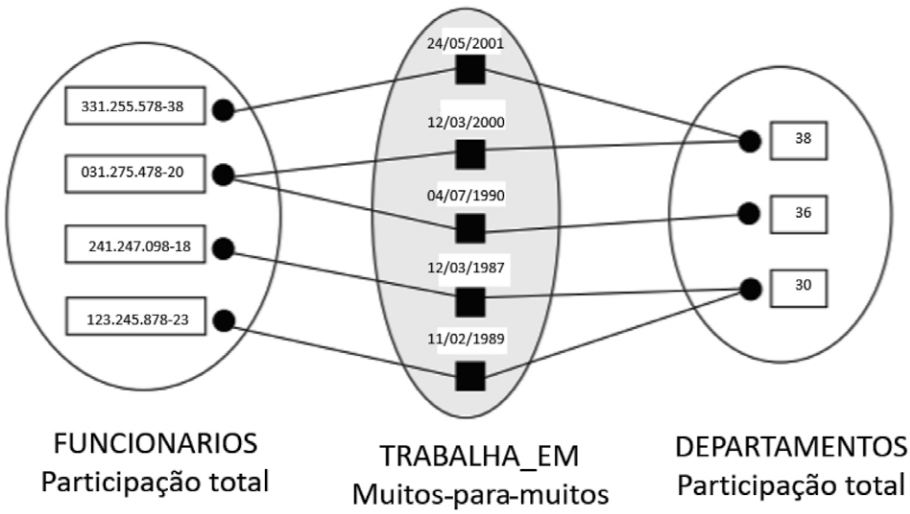

Figura 9 – Instância do conjunto de relacionamentos Trabalha_em

No diagrama ER, podemos verificar que cada departamento tem escritórios em diversas localidades e se deseja registrar as localidades nas quais cada funcionário trabalha. Esse tipo de relacionamento é chamado de relacionamento ternário, pois registra uma associação entre um funcionário, um departamento e uma localidade. O diagrama ER para essa variante de Trabalha_em, que denominamos Trabalha_em2, é ilustrado na figura a seguir.

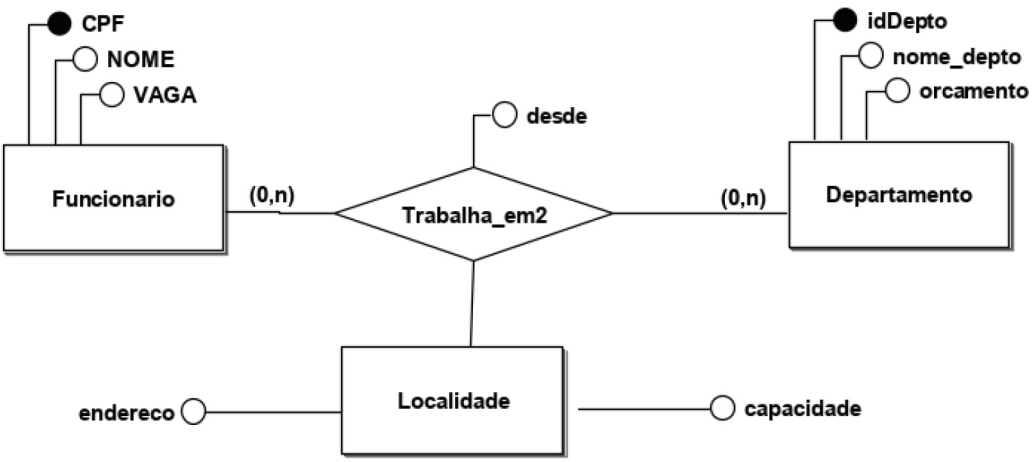

Figura 10 – Conjunto de relacionamento ternário

As entidades que fazem parte de um conjunto, que participam de relacionamentos, não precisam ser distintas; algumas vezes, um relacionamento pode envolver duas entidades no mesmo conjunto de entidades. Por exemplo, considere o conjunto de relacionamentos Reporta_a. Como os funcionários reportam a outros funcionários, todo relacionamento em Reporta_a é da forma (func1, func2), sendo que func1 e func2 são entidades de Funcionário. Entretanto, eles desempenham papéis diferentes: func1 reporta ao funcionário supervisor func1, o que é refletido nos indicadores de papel supervisor e subordinado. Se um conjunto de entidades desempenhar mais do que um papel, o indicador de papel concatenado a um nome de atributo do conjunto de entidades nos fornecerá um nome único para cada atributo do conjunto de relacionamentos. Por exemplo, o conjunto de relacionamentos Reporta_a tem os atributos correspondentes a CPF do supervisor e CPF do subordinado, e os nomes desses atributos são supervisor_CPF e subordinado_CPF, os quais poderiam ser facilmente verificados em uma sessão relatório no sistema de informação.

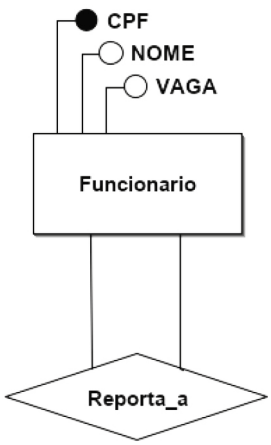

Figura 11 – Conjunto de relacionamento Reporta_a

Recursos adicionais do modelo ER

Construtores do modelo ER permitem descrever as principais propriedades das características dos dados. A descrição do modelo ER é uma boa explicação para seu uso tão difundido.

2.7 Restrições de chave

Tomamos como base a relação entre Trabalha_em, ilustrada na figura 10. Um funcionário pode trabalhar em diversos departamentos, e um departamento pode ter diversos funcionários, como ilustrado na instância Trabalha_em, mostrada na figura 9, com as informações do funcionário 031.275.478-20 trabalha no Departamento 36 desde 04/07/1990 e no Departamento 38 desde 12/03/2000. O Departamento 38 tem dois funcionários.

Por esse exemplo, agora temos outro conjunto de relacionamentos, chamado Gerencia, entre os conjuntos de entidades Funcionario e Departamento, onde cada departamento tenha no máximo um gerente, embora um mesmo funcionário possa gerenciar mais do que um departamento. A restrição de que cada departamento tem no máximo um gerente é um exemplo de restrição de chave e isso

implica que cada entidade Departamento apareça em, no máximo, um relacionamento Gerencia, em qualquer instância permitida de Gerencia. Essa restrição é indicada no diagrama ER da figura a seguir, utilizando uma seta de Gerencia a Departamento. Intuitivamente, a seta afirma que dada a entidade de Departamento, podemos determinar univocamente o relacionamento Gerencia no qual ela aparece.

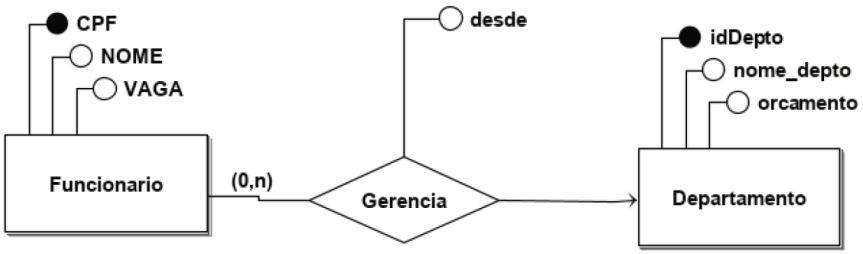

Figura 12 – Restrição da chave em Gerencia

Uma entidade do conjunto de relacionamentos Gerencia é ilustrada na figura 12. Ainda pode ser observado que uma entidade em potencial do conjunto de relacionamentos Trabalha_em viola a restrição de chave em Gerencia.

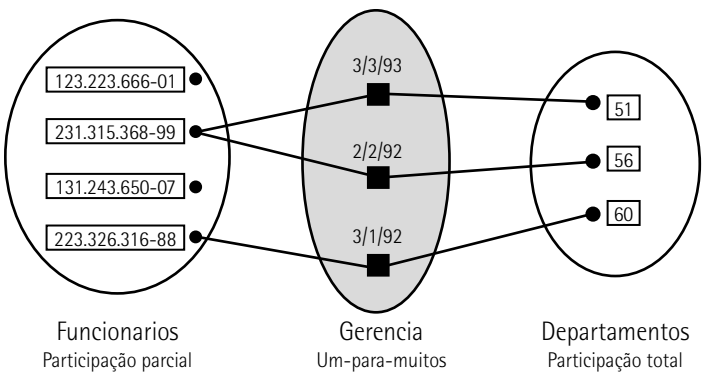

Figura 13 – Instância do conjunto de relacionamentos Gerencia

O tipo de relacionamento um-para-muitos indica que um funcionário pode estar associado a muitos departamentos (na qualificação de gerente). Nesse caso, cada departamento pode estar associado a, no máximo, um funcionário como seu gerente. Se verificado de outra forma, o relacionamento muitos-para-muitos é exemplificado pela relação de muitos funcionários que trabalham em muitos departamentos. Se adicionarmos ao conjunto de relacionamentos Gerencia a restrição de que cada funcionário pode gerenciar no máximo um departamento, sendo que essa restrição seria indicada acrescentando uma seta de Funcionarios a Gerencia, representando um conjunto de relacionamentos um-para-um.

2.7.1 Restrições de chave para relacionamentos ternários

Entende-se que há uma convenção – e o conceito subjacente de restrição de chave – nos conjuntos de relacionamentos envolvendo três ou mais conjuntos de entidades: se um conjunto de entidades E tiver uma restrição de chave em um conjunto de relacionamentos R, cada entidade em uma instância de E

aparecerá em, no máximo, um relacionamento em (uma instância correspondente de) R. Para indicar uma restrição de chave do conjunto de entidades E em um conjunto de relacionamentos R, desenhamos uma seta de E a R.

Um relacionamento ternário com restrições de chave é ilustrado na figura a seguir. Cada funcionário trabalha em, no máximo, um departamento e em uma única localidade. Existe uma instância do conjunto de relacionamentos Trabalha_em3. Observe que cada departamento pode estar associado a diversos funcionários e localidades e que cada localidade pode estar associada a diversos departamentos e funcionários; entretanto, cada funcionário está associado a um único departamento e localidade.

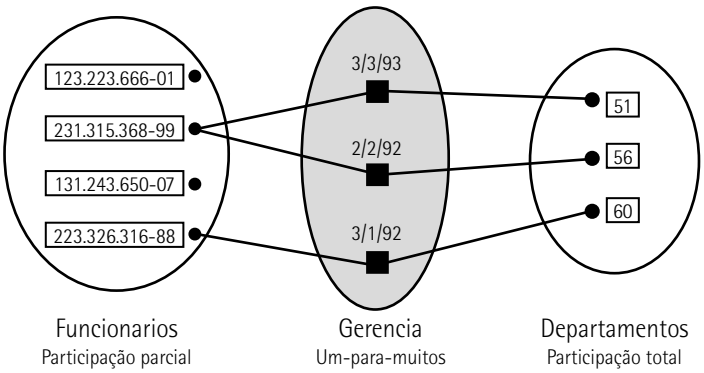

Figura 14 – Conjunto de relacionamentos ternário com restrições de chave

2.7.2 Restrições de participação

Representada por uma restrição de chave que, nesse caso, pode ser exemplificada em Gerencia e que nos informa que um departamento possui no máximo um gerente. Uma dúvida comum é questionar se todo departamento tem um gerente. Consideremos que é exigido que todo departamento tenha um gerente. Esse requisito é um exemplo de **restrição de participação** e mostra que a participação do conjunto de entidades Departamentos no conjunto de relacionamentos Gerencia é considerada **total**. Uma participação que não é total será chamada de **parcial**. Por exemplo, a participação do conjunto de entidades Funcionarios em Gerencia é parcial, uma vez que nem todo funcionário gerencia um departamento.

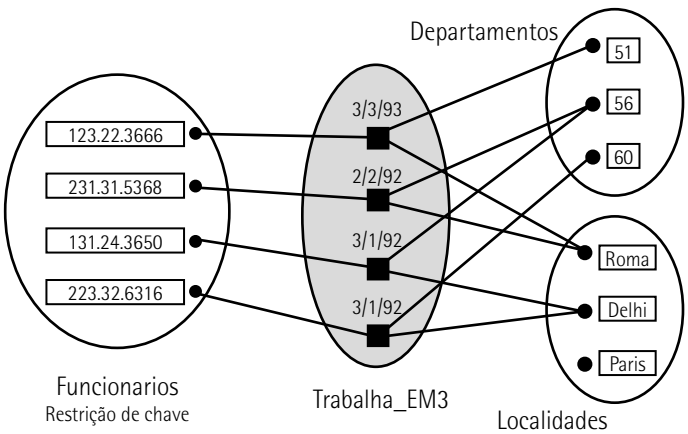

Figura 15 – Uma instância de Trabalha_em3

Quando pensamos no conjunto de relacionamentos Trabalha_em, é evidente que cada funcionário trabalhe em, no mínimo, um departamento e que cada departamento tenha, no mínimo, um funcionário. Isso significa que a participação tanto de Funcionarios quanto de Departamentos em Trabalha_em é total. É possível visualizar, na figura 16, a diagramação ER, na qual vemos os conjuntos de relacionamentos Gerencia e Trabalha_em e todas as restrições determinadas. Se a participação de um conjunto de entidades em um conjunto de relacionamentos for total, os dois são conectados por uma linha grossa; independentemente, a presença de uma seta indica uma restrição de chave. As instâncias de Trabalha_em e Gerencia ilustradas nas figuras 9 e 15 satisfazem a todas as restrições da figura 16.

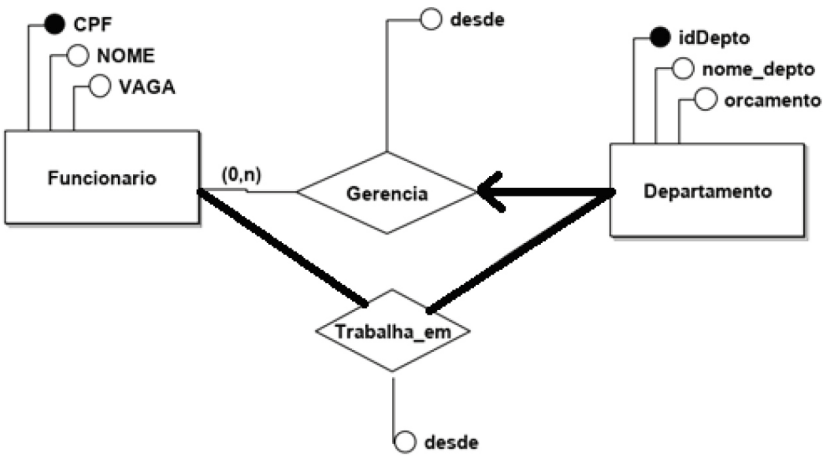

Figura 16 – Gerencia e Trabalha_em

2.7.3 Entidades fracas

A associação de atributos a um conjunto de entidades inclui uma chave. Nem sempre é possível essa suposição. Por exemplo, suponha que os funcionários possam adquirir planos de saúde para cobrir seus dependentes. Desejamos registrar as informações sobre esses planos de saúde, incluindo quem é coberto por cada plano, mas essa informação é realmente nosso único interesse sobre os dependentes de um funcionário. Se um funcionário se desligar da empresa, qualquer plano de saúde pertencente ao funcionário é extinto e é desejável excluir todas as informações relevantes sobre seu plano de saúde e seus dependentes do banco de dados.

Nessa situação, podemos escolher identificar um dependente apenas pelo nome, já que é razoável esperar que os dependentes de determinado funcionário tenham nomes diferentes. Assim, os atributos do conjunto de entidades Dependentes poderiam ser *nomed* e *idade*. O atributo *nomed* não identifica univocamente um dependente. Lembre-se de que a chave de Funcionarios é CPF, assim, poderíamos ter dois funcionários chamados Eduardo e cada um deles poderia ter um filho chamado Felipe. Dependentes é um exemplo de um conjunto de entidades fracas. Uma entidade fraca pode ser univocamente identificada apenas se considerarmos alguns dos seus atributos em conjunto com a chave primária de uma outra entidade, que é chamada de proprietária identificadora.

Segundo Ramakrishnan e Gehrke (2011), deve-se manter as seguintes restrições:

- O conjunto de entidades proprietárias e o conjunto de entidades fracas devem participar em um conjunto de relacionamentos um-para-muitos (uma entidade proprietária está associada a uma ou mais entidades fracas, mas cada entidade fraca tem uma única proprietária). Esse relacionamento é chamado **conjunto de relacionamentos identificadores** do conjunto de entidades fracas.

- O conjunto de entidades fracas deve ter participação total no conjunto de relacionamentos identificadores.

Por exemplo, uma entidade de Dependentes pode ser univocamente identificada apenas se considerarmos a chave da entidade proprietária Funcionarios e o *nomed* da entidade Dependentes. O conjunto de atributos de um conjunto de entidades fracas que identifica univocamente uma entidade fraca de uma determinada entidade proprietária é chamada chave parcial do conjunto de entidades fracas. Em nosso exemplo, *nomed* é uma chave parcial de Dependentes.

O conjunto de entidades fracas Dependentes e seu relacionamento com Funcionarios são ilustrados na próxima figura. A participação total de Dependentes em Apólice é indicada pela ligação entre eles com uma linha grossa. A seta de Dependentes a Apólice indica que cada entidade de Dependentes aparece em, no máximo, um (exatamente um, na realidade, em razão da restrição de participação) relacionamento Apólice. Para ressaltar o fato de que Dependentes é uma entidade fraca e Apólice é seu relacionamento identificador, desenhamos ambos com linhas grossas. Para indicar que *nome_dep* é uma chave parcial de Dependentes, sublinhamos o atributo usando uma linha tracejada. Isso significa que pode muito bem haver dois dependentes com o mesmo valor de *nome_dep*.

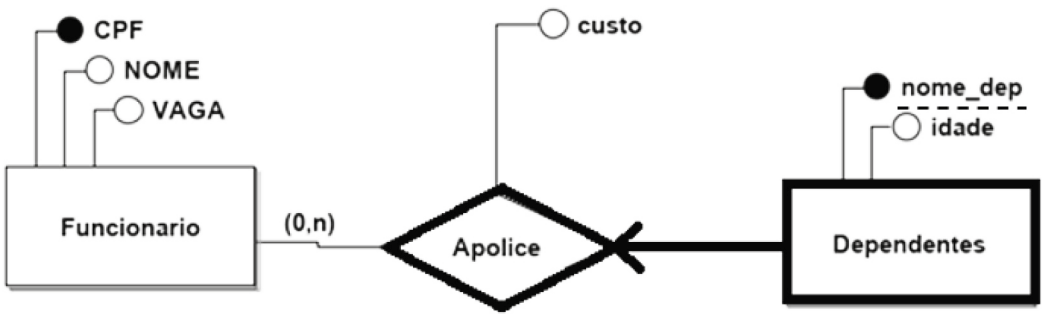

Figura 17 – Conjunto de entidades fracas

2.7.4 Hierarquias de classe

Classificar as entidades de um conjunto de entidades em subclasses é uma atividade muito comum. Por exemplo, poderíamos considerar um conjunto de entidades Funcionario_Horistas e um conjunto de entidades Funcionario_Contratados para distinguir a forma como os funcionários são pagos. Poderíamos ter os atributos Horas_Trabalhadas e Salário_Hora definidos para Funcionario_Horistas e um atributo Id_Contrato definido para Funcionario_Contratados.

Toda entidade deve possuir uma semântica em que um desses conjuntos seja também uma entidade de Funcionarios e, como tal, deve possuir todos os atributos definidos para Funcionarios. Assim, os

atributos definidos para uma entidade Funcionario_Horistas são os atributos de Funcionarios mais os de Funcionario_ Horistas. Os atributos do conjunto de entidades Funcionarios são herdados pelo conjunto de entidades Funcionario_Horistas, e Funcionario_Horistas é também considerado um empregado. Há uma restrição nas consultas sobre instâncias desses conjuntos de entidades: uma consulta que solicita todas as entidades Funcionários deve considerar também todas as entidades Funcionario_Horistas e Funcionario_Contratados, sua hierarquia pode ser observada na figura a seguir.

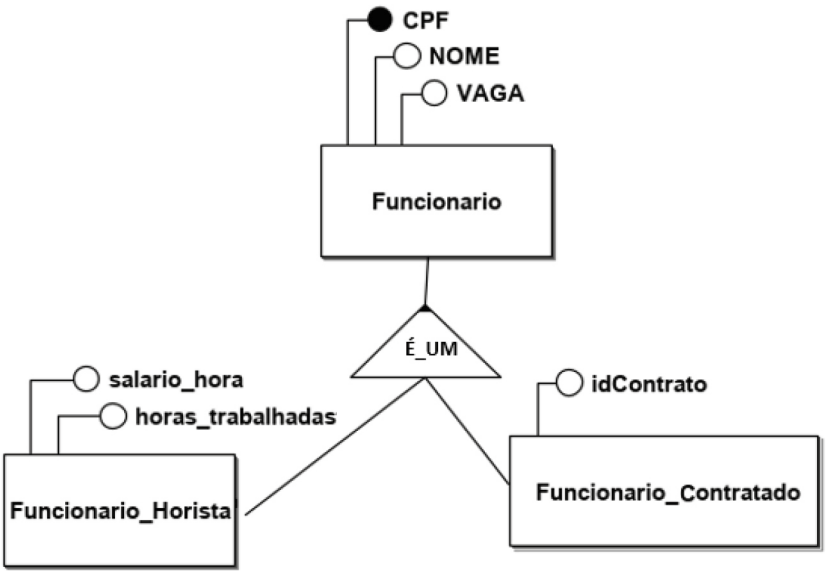

Figura 18 – Hierarquia de classe

Uma hierarquia de classe pode ser vista de duas formas:

- A superclasse Funcionarios é dividida em duas classes especializadas. A especialização é o processo de identificar subconjuntos de um conjunto de entidades (a superclasse) que compartilham algumas características distinguíveis. Tipicamente, a superclasse é definida primeiro, as subclasses são definidas em seguida, e os atributos específicos da subclasse e os conjuntos de relacionamentos são adicionados depois.

- Funcionario_Horistas e Funcionario_Contratados são generalizados em Funcionarios.

Temos outro exemplo, dois conjuntos de entidades Barcos_Motorizados e Carros podem ser generalizados em um conjunto de entidades Veículos_Motorizados. Como a generalização consiste em identificar quais são as características comuns de uma coleção de conjuntos de entidades e criar um novo conjunto de entidades que contenha as entidades possuindo essas características comuns, é recorrente que as subclasses sejam definidas primeiro, a superclasse seja definida em seguida e quaisquer conjuntos de relacionamentos que envolvam a superclasse sejam definidos depois.

Dois tipos de restrições com relação às hierarquias de restrições de sobreposição e de cobertura serão exemplificados a seguir. As restrições de sobreposição determinam se duas subclasses podem conter a mesma entidade. Por exemplo, NewsSoftware pode ser uma entidade de Funcionario_Horistas e também

Funcionario_Contratados? Intuitivamente, não. Pode ele ser tanto uma entidade Funcionario_Contratados como também Funcionario_Seniores? Intuitivamente, sim. Denotamos isso escrevendo <Funcionario_Contratados SOBREPÕE Funcionario_Seniores>. Na falta de tal afirmação, consideramos por padrão que os conjuntos de entidades têm restrições de não se sobrepor.

As restrições de cobertura determinam como as entidades das subclasses estão juntas a todas as entidades da superclasse. Por exemplo, toda entidade Funcionarios deve pertencer a uma de suas subclasses? Intuitivamente, não. Toda entidade Veículos_Motorizados deve ser ou uma entidade Barcos_Motorizados ou uma entidade Carros? Intuitivamente, sim; uma propriedade característica de hierarquias de generalização é que toda instância de uma superclasse é uma instância de uma subclasse. Denotamos isso escrevendo <Barcos_Motorizados E Carros COBREM Veículos_Motorizados>.

Podemos entender que veículos motorizados não são barcos motorizados. Isso gera um padrão que não há restrição de cobertura.

As subclasses podem ser identificadas em duas subclasses (por especialização ou generalização):

1) Acrescentando a descrição dos atributos que façam sentido apenas às entidades em uma subclasse. Por exemplo, Salário_Hora não faz sentido para uma entidade Funcionario_Contratados, cujo pagamento é determinado por um contrato individual.

2) A identificação do conjunto de entidades que participam em algum relacionamento. Por exemplo, poderíamos definir o relacionamento Gerencia de forma que os conjuntos de entidades participantes sejam Funcionario_Seniores e Departamentos, para assegurar que apenas os funcionários seniores possam ser gerentes. Como um outro exemplo, Barcos_Motorizados e Carros podem ter atributos descritivos diferentes (digamos, tonelagem e número de portas), mas como entidades de Veículos_Motorizados, eles devem ter licenciamento. A informação de licenciamento pode ser capturada por um relacionamento Licenciado_Para entre Veículos_Motorizados e um conjunto de entidades chamado Proprietários.

2.8 Agregação

É um tipo especial de associação onde se tenta demonstrar que as informações de um objeto precisam ser complementadas pelas informações de um objeto de outra classe. Esse tipo de associação tenta demonstrar uma relação todo ou parte entre os objetos associados. O conjunto de relacionamentos Financia captura essa informação. Um departamento que financia um projeto pode designar funcionários para monitorar o financiamento. Intuitivamente, Monitora deve ser um conjunto de relacionamentos que associa um relacionamento Financia (em vez de uma entidade Projetos ou Departamentos) a uma entidade Funcionarios. Entretanto, definimos relacionamentos para associar duas ou mais entidades.

Para exemplificar um conjunto de relacionamentos como Monitora, adicionamos um novo recurso ao modelo ER chamado **agregação**, o qual possibilita indicar que um conjunto de relacionamentos (identificado através de um quadro tracejado) participa de outro conjunto de relacionamentos. Isso é ilustrado na figura 19, com um quadro tracejado ao redor de Financia (e seus conjuntos de entidades

participantes) usado para denotar agregação. Algo assim efetivamente nos possibilita tratar Financia como um conjunto de entidades para propósitos de definição do conjunto de relacionamento Monitora.

Lembrete

Agregar significa compor parte de alguma coisa. O conceito de agregação significa que podemos entender classes de coisas como elemento de composição de outra classe maior. Ou seja, entender que um conjunto de várias entidades compõem uma entidade maior.

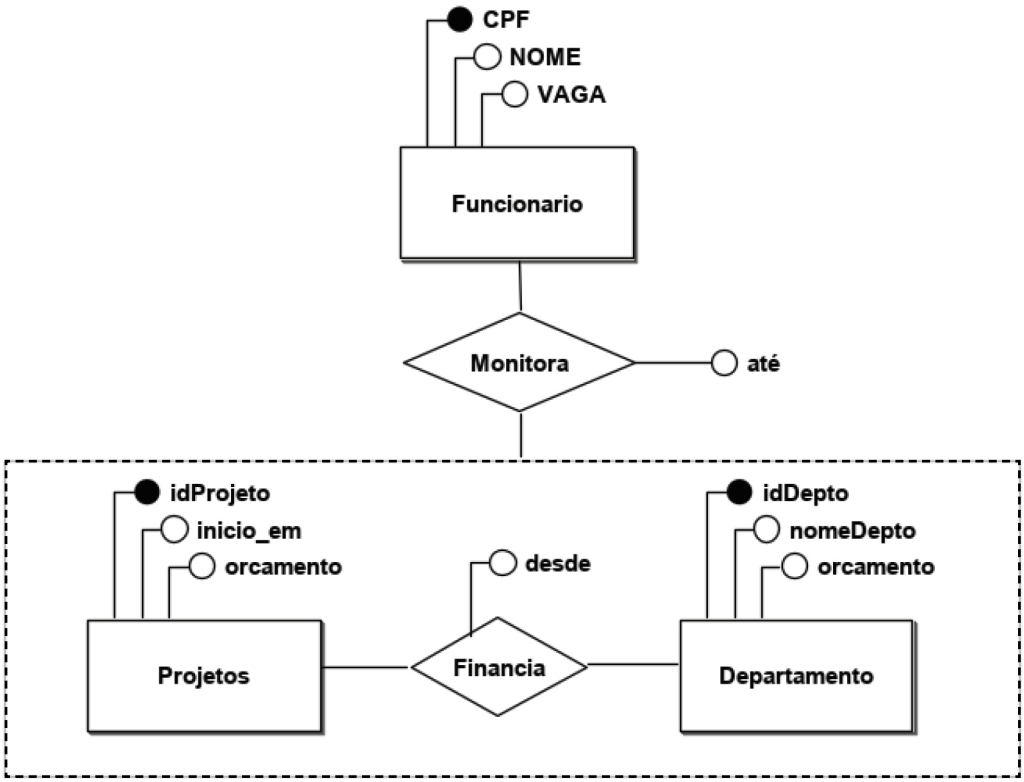

Figura 19 – Agregação

Resumo

Nesta unidade, vimos que o banco de dados pode ser definido como uma coleção de dados relacionados entre si, que, quando bem gerenciados, podem ajudar a garantir a eficiência e o bom funcionamento de uma organização ou empresa. Ao apresentar o desenvolvimento histórico dos bancos de dados, destacamos como em meados dos anos 1960, foram criados os fundamentos que deram origem ao que conhecemos atualmente como sistema de gerenciamento de banco de dados (SGBD) e como essa ferramenta foi aplicada, gradativamente, pelas organizações para melhorar a manutenção e eficiência dos dados utilizados.

Apresentamos os três aspectos dos níveis organizacionais: tecnológico, gerencial e cultural, e vimos, ainda, quais são as principais vantagens e desvantagens dos bancos de dados relacionais. Para isso, foram apresentados dois modelos utilizados entre os profissionais de desenvolvimento de *software* e administradores de banco de dados, o modelo relacional e o no SQL.

Junto dos conceitos mais importantes de sistemas de gerenciamento de banco de dados, foram apresentadas as etapas necessárias para a implementação de um projeto de banco de dados, além das terminologias básicas da área. Além disso, mostramos os níveis de abstração envolvidos na concepção do projeto de banco de dados: o modelo conceitual, o modelo lógico e os modelos físicos.

Expusemos o modelo relacional em detalhes e também as definições de entidade e de atributo, bem como as características relacionadas a cada um desses elementos. Para encerrar a unidade, exemplificamos, por meio dos conjuntos de relacionamentos, como funcionam as restrições de chave e a agregação.

Exercícios

Questão 1. Sabemos que a utilização de banco de dados relacionais oferece uma série de benefícios. Considere as ações a seguir relativas a esses benefícios.

I – Permitir que os dados sejam armazenados por quantidade arbitrária de tempo, normalmente bem longo, e de forma segura.

II – Permitir que os dados sejam acessados simultaneamente por grande número de aplicações.

III – Permitir que grande número de aplicações utilize os mesmos dados de forma consistente.

IV – Garantir uma forma de interpretar e trabalhar com os dados que seja suficientemente poderosa e, ao mesmo tempo, flexível, mas independente dos detalhes de uma implementação específica.

Nas alternativas a seguir, temos as quatro ações anteriores seguidas dos nomes de conceitos. Escolha a alternativa que associa corretamente o nome e o conceito indicado.

A) I, persistência de dados; II, integração; III, padronização; IV, concorrência.

B) I, integração; II, persistência de dados; III, padronização; IV, concorrência.

C) I, integração; II, concorrência; III, padronização; IV, persistência de dados.

D) I, persistência de dados; II, concorrência; III, integração; IV, padronização.

E) I, concorrência; II, integração; III, padronização; IV, persistência de dados.

Resposta correta: alternativa D.

Análise da questão

A questão aborda as quatro principais vantagens de utilização dos sistemas de banco de dados relacionais. O objetivo é ter uma clara visibilidade de quais são essas quatro vantagens e saber qual é o conceito correspondente a cada uma delas. Assim, temos as associações a seguir.

Persistência de dados refere-se à possibilidade de que os dados sejam armazenados seguramente por período arbitrário.

Concorrência refere-se à possibilidade de que os dados sejam acessados por elevada quantidade de aplicações ao mesmo tempo.

Integração refere-se à possibilidade de que elevada quantidade de aplicações use os mesmos dados de forma consistente.

Padronização refere-se à garantia de que haja uma maneira de interpretar e trabalhar com os dados que seja suficientemente poderosa, flexível e independente dos detalhes de uma implementação específica.

Questão 2. De acordo com a teoria de banco de dados relacionais, devemos utilizar vários modelos para obtermos um projeto adequado de um sistema. Esses modelos são frequentemente construídos em etapas, em um processo iterativo de detalhamento e de refinamento.

Nesse contexto, considere as afirmativas a seguir sobre os três modelos utilizados na modelagem de banco de dados relacionais.

I – Modelo que corresponde a um nível inicial da modelagem. Nesse nível, estamos preocupados em compreender a natureza do problema de forma abstrata, sem focarmos em como o banco de dados vai ser implementado fisicamente. No entanto, queremos entender como os aspectos relevantes para a implementação do sistema se inter-relacionam e quais são as suas características básicas.

II – Modelo que corresponde a um nível intermediário de detalhamento da modelagem, no qual aprofundamos vários aspectos do nível anterior, buscando entender a estrutura dos elementos criados anteriormente. Esse nível de modelagem deve ser independente da implementação, isto é, esse modelo ainda é abstrato. Além disso, esse modelo serve de entrada para uma etapa posterior, mais concreta.

III – Modelo que corresponde a uma etapa em que estamos preocupados em criar a estrutura que implementa os modelos anteriores em um banco de dados específico. Muitas vezes, esse modelo tem um mapeamento direto em *scripts*, visando à criação da sua estrutura.

Assinale a alternativa que associa corretamente os três conceitos citados anteriormente e os seus nomes.

A) I, modelo conceitual; II, modelo físico; III, modelo lógico.

B) I, modelo conceitual; II, modelo lógico; III, modelo físico.

C) I, modelo físico; II, modelo conceitual; III, modelo lógico.

D) I, modelo físico; II, modelo lógico; III, modelo conceitual.

E) I e II, modelo físico; III, modelo conceitual.

Resposta correta: alternativa B.

Análise da questão

O objetivo da questão é verificar a compreensão dos três conceitos utilizados na modelagem de sistemas de banco de dados relacionais. Para essa questão, é importante ser capaz de entender quais são os objetivos desses três modelos, para que a sua construção, em um projeto real, seja feita de forma consistente. Assim, temos as associações a seguir.

Modelo conceitual trata-se da etapa inicial da modelagem, na qual queremos compreender a natureza do problema de forma abstrata e entender como os aspectos fundamentais para a implementação do sistema se inter-relacionam e quais são as suas características básicas.

Modelo lógico trata-se da etapa intermediária da modelagem, na qual queremos compreender a estrutura dos elementos criados anteriormente, ainda em uma análise independente da implementação.

Modelo físico trata-se da etapa na qual focamos em criar a estrutura que implementa os modelos anteriores em um banco de dados específico.