

# CURSOS TÉCNICOS **SENAI**

Saiba fazer e aconteça no mercado.



**Curso Técnico em Informática**  
**Banco de Dados I**

Curso Técnico em Informática

## **Banco de Dados I**

**Robson Braga de Andrade**  
Presidente da Confederação Nacional da Indústria

**Rafael Lucchesi**  
Diretor do Departamento Nacional do SENAI

**Regina Maria de Fátima Torres**  
Diretora de Operações do Departamento Nacional do SENAI

.....

**Alcantaro Corrêa**  
Presidente da Federação da Indústria do Estado de Santa Catarina

**Sérgio Roberto Arruda**  
Diretor Regional do SENAI/SC

**Antônio José Carradore**  
Diretor de Educação e Tecnologia do SENAI/SC

**Marco Antônio Dociatti**  
Diretor de Desenvolvimento Organizacional do SENAI/SC



Confederação Nacional da Indústria  
Serviço Nacional de Aprendizagem Industrial

Curso Técnico em Informática

## Banco de Dados I

Adriano Gonçalves Polidoro

Florianópolis/SC  
2011

É proibida a reprodução total ou parcial deste material por qualquer meio ou sistema sem o prévio consentimento do editor.

**Autor**

Adriano Gonçalves Polidoro

**Fotografias**

Banco de Imagens SENAI/SC  
<http://www.sxc.hu/>  
<http://office.microsoft.com/en-us/images/>  
<http://www.morguefile.com/>  
<http://www.bancodemidia.cni.org.br/>

Ficha catalográfica elaborada por Luciana Effting CRB14/937 - Biblioteca do SENAI/SC Florianópolis

P766b

Polidoro, Adriano Gonçalves

Banco de dados I / Adriano Gonçalves Polidoro. – Florianópolis :  
SENAI/SC, 2011.

99p. : il. color; 28 cm.

Inclui bibliografias.

1. Banco de dados. 2. Organização de arquivos (Computação). 3. Banco de dados relacionais. 4. Exploração de dados. I. SENAI. Departamento Regional de Santa Catarina. II. Título.

CDU 004.65

SENAI/SC — Serviço Nacional de Aprendizagem Industrial

Rodovia Admar Gonzaga, 2.765 – Itacorubi – Florianópolis/SC

CEP: 88034-001

Fone: (48) 0800 48 12 12

[www.sc.senai.br](http://www.sc.senai.br)

# Prefácio

Você faz parte da maior instituição de educação profissional do estado. Uma rede de Educação e Tecnologia, formada por 35 unidades conectadas e estrategicamente instaladas em todas as regiões de Santa Catarina.

No SENAI, o conhecimento a mais é realidade. A proximidade com as necessidades da indústria, a infraestrutura de primeira linha e as aulas teóricas, e realmente práticas, são a essência de um modelo de Educação por Competências que possibilita ao aluno adquirir conhecimentos, desenvolver habilidade e garantir seu espaço no mercado de trabalho.

Com acesso livre a uma eficiente estrutura laboratorial, com o que existe de mais moderno no mundo da tecnologia, você está construindo o seu futuro profissional em uma instituição que, desde 1954, se preocupa em oferecer um modelo de educação atual e de qualidade.

Estruturado com o objetivo de atualizar constantemente os métodos de ensino-aprendizagem da instituição, o **Programa Educação em Movimento** promove a discussão, a revisão e o aprimoramento dos processos de educação do SENAI. Buscando manter o alinhamento com as necessidades do mercado, ampliar as possibilidades do processo educacional, oferecer recursos didáticos de excelência e consolidar o modelo de Educação por Competências, em todos os seus cursos.

É nesse contexto que este livro foi produzido e chega às suas mãos. Todos os materiais didáticos do SENAI Santa Catarina são produções colaborativas dos professores mais qualificados e experientes, e contam com ambiente virtual, mini-aulas e apresentações, muitas com animações, tornando a aula mais interativa e atraente.

Mais de 1,6 milhões de alunos já escolheram o SENAI. Você faz parte deste universo. **Seja bem-vindo e aproveite por completo a Indústria do Conhecimento.**



SÉRGIO ROBERTO ARRUDA  
Diretor Regional SENAI/SC



# Sumário

**Conteúdo Formativo** 9

**Apresentação** 11

## 14 Unidade de estudo 1

O que é Banco de Dados I?

15 **Seção 1** - Definição de dado x informação

15 **Seção 2** - Definição de banco de dados

16 **Seção 3** - Exemplos de banco de dados

17 **Seção 4** - Utilização e benefícios

## 20 Unidade de estudo 2

Sistema Gerenciador de Banco de Dados

21 **Seção 1** - Definição

22 **Seção 2** - Vantagens da utilização de um SGBD

23 **Seção 3** - Tipos de usuários de um SGBD

24 **Seção 4** - Quando não utilizar um SGBD

24 **Seção 5** - SGBD x Sistemas de arquivos

## 26 Unidade de estudo 3

A História do Banco de Dados

27 **Seção 1** - Década de 1950 – o início

27 **Seção 2** - Década de 1960 – banco de dados hierárquico

28 **Seção 3** - Década de 1960 e início de 1970 – banco de dados em rede

28 **Seção 4** - Década de 1970 – banco de dados relacionais

29 **Seção 5** - Década de 1980 – banco de dados orientados a objetos

29 **Seção 6** - Década de 1990 – bancos de dados objetos-relacionais

30 **Seção 7** - Anos 2000 – bancos de dados avançados

## 32 Unidade de estudo 4

Conceitos de Modelagem de Dados

33 **Seção 1** - O que é modelagem de dados

33 **Seção 2** - Modelagem conceitual

34 **Seção 3** - Modelagem lógica

34 **Seção 4** - Modelagem física

## 36 Unidade de estudo 5

Modelo de Entidade-Relacionamento

37 **Seção 1** - O que é modelo de entidade-relacionamento

37 **Seção 2** - Entidades

38 **Seção 3** - Atributos

39 **Seção 4** - Relacionamentos

40 **Seção 5** - Cardinalidade

## 44 Unidade de estudo 6

Projeto de Banco de Dados

45 **Seção 1** - O que é um projeto?

45 **Seção 2** - Projeto de banco de dados

## 46 Unidade de estudo 7

Coleta de Requisitos e Regras de Negócio

47 **Seção 1** - Conceitos

47 **Seção 2** - Como levantar os requisitos e as regras de negócios

## 48 Unidade de estudo 8

Modelagem Conceitual Baseada no MER

49 **Seção 1** - Definindo as entidades

49 **Seção 2** - Definindo os relacionamentos

50 **Seção 3** - Definindo as cardinalidades

52 **Seção 4** - Contextualização

## 54 Unidade de estudo 9

Modelagem Lógica Baseada no DER Conceitual

55 **Seção 1** - Construindo um modelo lógico

55 **Seção 2** - Mapeamento das entidades, relacionamentos, cardinalidade e atributos

57 **Seção 3** - Atributos de chave primária

58 **Seção 4** - Atributos de chave estrangeira

## 60 Unidade de estudo 10

Modelagem Física Baseada no DER Lógico

61 **Seção 1** - Construindo um modelo físico

61 **Seção 2** - Estudos e tabelas

62 **Seção 3** - Estudo e caracterização dos atributos

64 Seção 4 - Sugestões para uma boa confecção do DER físico

## 66 Unidade de estudo 11

Banco de Dados MySQL

67 **Seção 1** - O que é MySQL

67 **Seção 2** - O aplicativo EasYPHP

68 **Seção 3** - O funcionamento do MySQL

69 **Seção 4** - Pormenores

## 70 Unidade de estudo 12

Utilização de Ferramenta case – MySQL Workbench

71 **Seção 1** - Utilidade de uma ferramenta *case*

71 **Seção 2** - MySQL Workbench

72 **Seção 3** - Criando o modelo físico (estrutura)

74 **Seção 4** - Inserindo os dados

75 **Seção 5** - Criando o banco de dados MYSQL

76 **Seção 6** - Sincronizando o DER com o banco de dados MySQL

## 80 Unidade de estudo 13

SQL

81 **Seção 1** - O que é SQL

81 **Seção 2** - Breve histórico

81 **Seção 3** - O funcionamento

81 **Seção 4** - DDL

83 **Seção 5** - DML

85 **Seção 6** - DCL

86 **Seção 7** - Vantagens na utilização do SQL

# Sumário

## 88 Unidade de estudo 14

Transações de Banco de Dados

89 **Seção 1** - Conceitos

89 **Seção 2** - Na prática

90 **Seção 3** - Controle transacional

## 92 Unidade de estudo 15

Elementos Complementares – Normalização e Integridade Referencial

93 **Seção 1** - Normalização

93 **Seção 2** - Integridade referencial

## 96 Unidade de estudo 16

Arquiteturas de Bancos de Dados

97 **Seção 1** - Sistemas de computadores pessoais

97 **Seção 2** - Plataformas centralizadas

98 **Seção 3** - Sistemas cliente-servidor

98 **Seção 4** - Sistemas Distribuídos

## 100 Unidade de estudo 17

Tópicos Especiais – Ferramentas de Suporte à Decisão

101 **Seção 1** - *Data warehouse*

101 **Seção 2** - *Business Intelligence*

101 **Seção 3** - *Data Mining*

102 **Seção 4** - Utilização

Finalizando

105

Referências

107



# Conteúdo Formativo

## Carga horária da dedicação

→ Carga horária: 30 horas

## Competências

→ Aplicar as normas de qualidade no desenvolvimento de sistemas computacionais.

## Conhecimentos

- Acessibilidade;
- adaptabilidade;
- clareza;
- coerência;
- compatibilidade;
- conceito de qualidade;
- conforto;
- direitos autorais/propriedade intelectual;
- ergonomia;
- expressividade;
- flexibilidade;
- interatividade;
- normas de qualidade de *software*;
- obediência;
- portabilidade;
- qualidade de *software*;
- sistemas da qualidade;
- usabilidade.

## Habilidades

- Identificar critérios de usabilidade e qualidade no desenvolvimento de *softwares*;
- Utilizar normas de ergonomia;
- Utilizar normas de qualidade de *software*;
- Aplicar princípios de acessibilidade.

## Atitudes

- Organização e zelo na utilização de equipamentos;
- Foco no conteúdo trabalhado;
- Acesso a *sites* relacionados ao tema trabalhado;
- Organização e limpeza dos ambientes coletivos;
- Dedicação e empenho nas atividades curriculares e extracurriculares;
- Capacidade de abstração;
- Trabalho em equipe;
- Apresentação de novas soluções para situações-problemas;
- Cumprimento de prazos;
- Análise crítica de suas produções.

# Apresentação

Seja bem-vindo à unidade curricular Banco de Dados I!

Esta unidade curricular está voltada ao entendimento, planejamento e pré-requisitos para uma boa implementação de um banco de dados, competências que estão diretamente relacionadas à sua profissão.

É possível que você esteja se perguntando: “Por que eu preciso estudar e entender os bancos de dados?” A resposta é muito simples: a maioria das atividades que você desempenhará como técnico de informática remete-se a banco de dados. Afinal, todos os dados produzidos pelos sistemas de uma empresa precisam ser armazenados organizadamente em algum lugar, e esse lugar é o banco de dados.

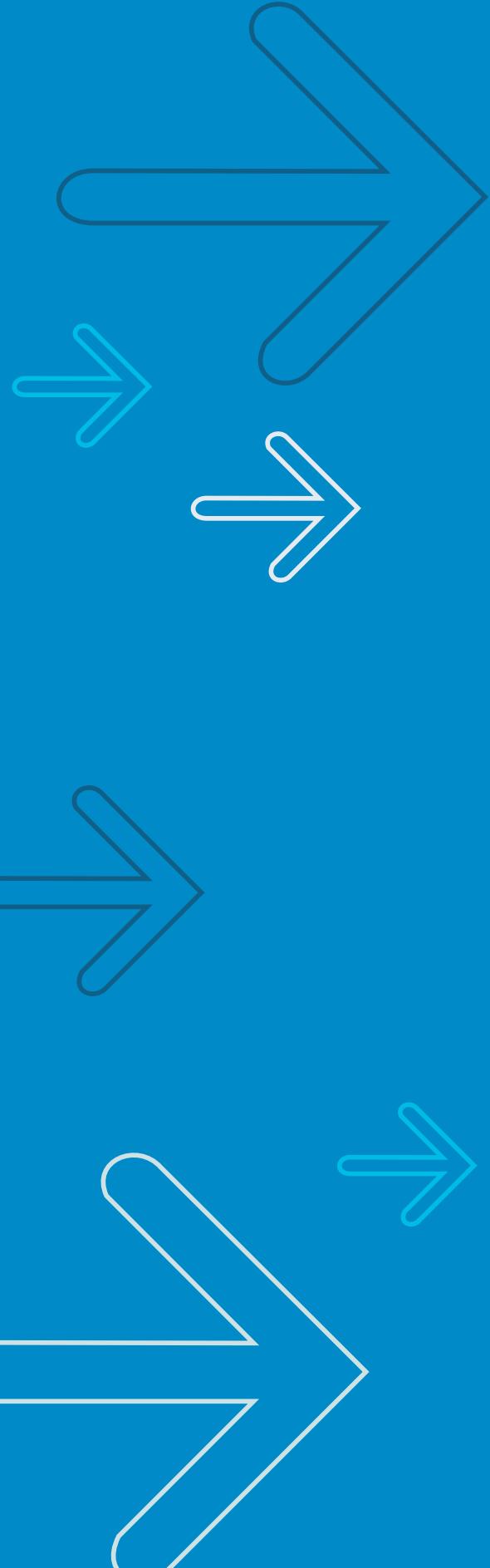
Se você não possui nenhum conhecimento em bancos de dados, não se preocupe, este material contém os ensinamentos básicos para modelar e criar um banco de dados e contribuirá para que você entenda esse conceito e desenvolva as habilidades necessárias para se preparar para o mercado de trabalho.

Esta unidade curricular contempla ainda a explanação das arquiteturas de armazenamento de dados, a extração de informações para desenvolver conhecimentos para tomada de decisões e a demonstração dos comandos triviais para iniciar a confecção de um banco de dados.

Vale lembrar que este livro acompanha um caderno de exercícios que devem ser feitos ao final de cada unidade para aprimorar seu aprendizado. Pronto para começar os estudos? Então, mãos à obra!

 Adriano Gonçalves  
Polidoro

É graduado em Sistemas de Informação (FURB) e atua há nove anos na área de Tecnologia de Informação. Possui experiência em docência no SENAI de Itajaí/SC e em empresas de grande porte, como Seara Alimentos, Cargill S. A. e Bunge Alimentos. Atualmente é analista de tecnologia na Tigre S. A. – Tubos e Conexões, em especial na análise de projetos de segurança da informação. Também leciona no SENAI de Joinville/SC. É certificado pelo ITIL Foundation pela Exin, e em 2010 recebeu o prêmio *Security Leaders*, promovido pela revista *Risk Report*, com o *case* de Segurança da Informação de maior sucesso em sua respectiva categoria.



# **Unidade de estudo 1**

## **Seções de estudo**

**Seção 1 – Definição de dado x informação**  
**Seção 2 – Definição de banco de dados**  
**Seção 3 – Exemplos de banco de dados**  
**Seção 4 – Utilização e benefícios**

# O que é um Banco de dados?

## SEÇÃO 1

### Definição de dado x informação

Antes de avançar sobre quaisquer conceitos específicos sobre banco de dados, é preciso entender o que é dado e o que é informação.

Imagine que você escreva o número dez em um papel ou no computador. Esse número, sozinho, nada representa para quem o lê, assim como uma palavra qualquer, por exemplo, “roda”, ou até mesmo uma frase em língua desconhecida. Tudo isso são considerados dados, pois não trazem consigo nenhum significado real. A informação, por sua vez, carrega um significado que pode ser compreendido facilmente: por exemplo, as frases “Isto custa R\$ 10,00” e “Esta **roda** é do meu carro” são consideradas informações, pois trazem consigo um significado real.

Conforme Abreu e Machado (2004), a informação deve ser encarada como um bem patrimonial de uma empresa, como um prédio, máquinas, mesas, entre outros. A informação deve ser utilizada de maneira estratégica pelos colaboradores da empresa e armazenadas em locais seguros e confiáveis, como um banco de dados. Já Silva (2001) vai mais longe, afirmando que a informação é mais importante do que quaisquer máquinas, influenciando diretamente todo nosso cotidiano.

Adicionalmente, alguns autores citam os conceitos de conhecimento, que representa o entendimento do significado daquela informação, e o conceito de sabedoria, que representa a competência necessária para produzir ganhos com o respectivo conhecimento.

Agora, prepare-se para conhecer banco de dados!

## SEÇÃO 2

### Definição de banco de dados

Como o próprio nome já diz, banco de dados é um conjunto de dados quaisquer, armazenados de maneira organizada, de onde podem ser extraídos a qualquer momento para trazer informações a quem é de direito.

Um banco de dados apenas trata dados, nunca informações. As informações são vistas pelos usuários por meio de consulta aos dados armazenados, sendo que a união desses dados extraídos é que apresenta a respectiva informação aos usuários.

Os bancos de dados têm o intuito de manter organizadas as informações de uma empresa, organização ou pessoa. Eles são utilizados para armazenar ordenadamente todos os tipos de dados possíveis e, com a ajuda de um sistema de informação, extrair informações de acordo com as necessidades. São projetados para uma necessidade específica e representam o mundo real, refletindo suas alterações. Por exemplo, se um supermercado utiliza um banco de dados para armazenar informações sobre seus produtos, inclusive preços, em uma situação real de aumento de preço, é necessário que uma atualização seja feita no banco de dados para refletir a situação real.

Adicionalmente, é importante saber que todo banco de dados é acessado por usuários que possuem privilégios específicos para tal, sendo que é necessária a utilização de um nome de usuário e uma senha para executar o acesso.

## SEÇÃO 3

### Exemplos de banco de dados

A seguir, são mostrados alguns exemplos de uso comum dos bancos de dados que não são virtuais, ou seja, não usam a tecnologia para armazenar dados:



Figura 1: Exemplos de bancos de dados não computacionais

Você pode ter utilizado poucas vezes alguns desses itens, que nem por isso deixam de ser bancos de dados, já que representam dados agrupados de maneira organizada de onde podem ser extraídos a qualquer momento em formato de informações.

Os bancos de dados utilizados na informática são similares a esses, porém são computacionais. Eles são utilizados, geralmente, por empresas que desejam que suas informações fiquem ordenadas e protegidas, sendo armazenadas em forma de dados agrupados que podem ser extraídos, modificados, incrementados e recuperados a qualquer momento com a ajuda de sistemas.

Vários são os bancos de dados do meio computacional atualmente existentes. Veja os mais utilizados:



Figura 2: Exemplos de bancos de dados computacionais

## SEÇÃO 4

### Utilização e benefícios

Percebe-se que fica cada dia mais difícil escapar da quantidade de informações geradas e disponibilizadas nos mais variados meios de comunicação, principalmente nos virtuais, que utilizam a tecnologia para se manter: internet, sistemas integradores, televisão, entre outros. Nessa linha, é inadmissível para uma empresa com grande volume de **transações** utilizar papéis, arquiveiros ou quaisquer outros métodos sem um banco de dados computacional para controlar seus negócios. Essa prática seria inviável, tamanho o volume e a velocidade em que as informações percorrem atualmente. Por esse motivo, você se depara com bancos de dados em todo lugar, mesmo indiretamente.

➤ **Transação** é uma unidade de execução de programa que acessa e possivelmente atualiza alguns dados (KORTH; SILVERSCHATZ; SUDARSHAN, 1999). Você verá mais sobre transações na unidade 14.



Figura 3: Exemplos de lugares onde encontram-se bancos de dados

Os sistemas dessas empresas devem ser bem confeccionados, utilizando um banco de dados estruturado para auxiliar na organização dos dados, tornando possível sua extração em forma de informações consistentes que darão suporte à tomada de decisões da empresa.

A extração dos dados e sua visualização em forma de informações estruturadas não é o único benefício da utilização de um banco de dados. Ele permite também que você faça acessos múltiplos às informações, atualização, exclusão e inclusão de dados, garantindo sempre sua consistência e solidez, permitindo *backup* e recuperação, caso seja necessário.

Um banco de dados, geralmente, é acessado, manipulado e gerenciado por meio de um sistema gerenciador de banco de dados, e este será o assunto da próxima unidade. Até lá!



# Unidade de estudo 2

## Seções de estudo

**Seção 1 – Definição**

**Seção 2 – Vantagens da utilização de um SGBD**

**Seção 3 – Tipos de usuários de um SGBD**

**Seção 4 – Quando não utilizar um SGBD**

**Seção 5 – SGBD x sistemas de arquivos**

# Sistema Gerenciador de Banco de Dados

## SEÇÃO I

### Definição

Um sistema gerenciador de banco de dados (SGBD) nada mais é do que um conjunto de *softwares* que gerencia os dados presentes nos bancos de dados. Ele permite criar o banco e é responsável por mantê-lo, proporcionando flexibilidade, segurança, recuperação e integridade.

Canto, Medeiros e Valdo (1997, p. 2) definem SGBD como “[...] um programa de computador especializado em armazenar e manter diferentes bancos de dados.”

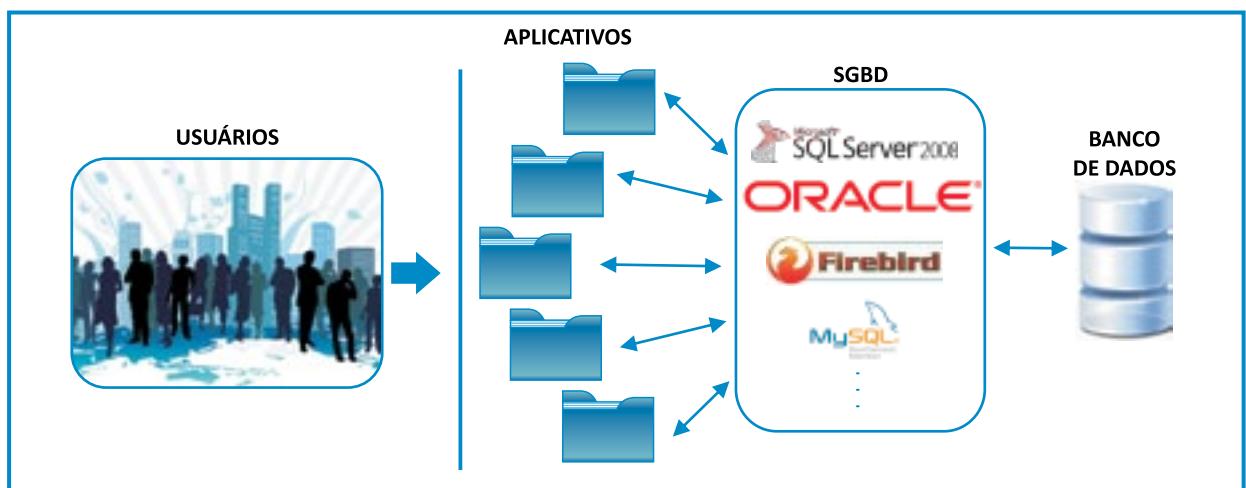


Figura 4: Composição de um sistema de acesso a um banco de dados por meio do SGBD

Um banco de dados não precisa ter, necessariamente, um SGBD ativo para estar em operação, porém a não utilização de um SGBD pode resultar em sérios problemas, como erros nas informações apresentadas e, inclusive, perda de dados importantes da empresa, já que não há controle sobre as informações ali inseridas. As vantagens da utilização de um SGBD para a maioria das empresas superam quaisquer custos ou tempo de implementação.

## SEÇÃO 2

### Vantagens da utilização de um SGBD

Acompanhe agora um detalhamento sobre algumas das vantagens da utilização de um SGBD em conjunto com um banco de dados.

#### ➤ Segurança

O SGBD é responsável pela definição de quem acessará determinada informação dentro do banco de dados. Além disso, gerencia esse acesso, permitindo apenas ações de usuários dos quais ele possui autorização. Por exemplo: uma pessoa da área comercial não pode visualizar dados da folha de pagamento de sua empresa. Já uma pessoa da área de recursos humanos pode. Seguindo o mesmo exemplo, é entendível que a maioria das pessoas da área de recursos humanos pode visualizar, mas nem todas podem alterar os dados da folha de pagamento da empresa. É nessa hora que o SGBD entra em ação novamente, permitindo ou não a inserção, atualização ou exclusão de dados.

#### ➤ Recuperação

Recursos como *backup* e recuperação de dados compõem o SGBD.

Isso permite que as informações armazenadas no banco de dados sejam recuperadas em caso de falhas nos aplicativos ou até mesmo de *hardware*.

Por exemplo: se um usuário está realizando uma transação nas informações da área financeira no banco de dados com auxílio de um *software* e esse *software* é interrompido no meio do processamento por algum motivo, seja por problemas físicos no servidor ou até mesmo por problemas de sistemas operacionais, o SGBD tem a responsabilidade de restaurar a base de dados em um estado anterior a essa falha ou, até mesmo, solicitar automaticamente que a transação seja reiniciada do exato ponto onde parou. Isso permite que as informações da área financeira não sejam corrompidas ou tornem-se não confiáveis.

Essa recuperação é possível graças ao sistema de gravação utilizado pelo SGBD, que grava as transações na memória e somente depois as repassa para o disco, produzindo inclusive maior velocidade de gravação, já que o acesso à memória é mais rápido que o acesso ao disco rígido. Um *log* é utilizado para gravar todas as alterações feitas enquanto as transações estão sendo gravadas na memória e em casos de acontecimento das falhas descritas anteriormente. Esse *log* é utilizado, então, pelo SGBD para que seja possível retornar a uma situação prévia à falha.

#### ➤ Integridade

Para cada dado gravado no banco de dados, é gravado outro dado relacionado, que pode ser em formato texto ou número.

Conforme já visto, um dos intuios da utilização dos bancos de dados é favorecer a organização dos dados. Para que isso seja possível, o SGBD possui uma funcionalidade que permite que as informações fiquem sempre organizadas e íntegras, e realiza isso da seguinte forma:

Exemplo: gravação dos dados de faturamento de uma empresa:

- Nome da empresa: JC representações Ltda.
- Faturamento anual: R\$ 800.000,00
- Quantidade de funcionários: 38
- Quantidade de filiais: 2

Todos esses dados fazem parte do mesmo conjunto de informações: as informações da empresa JC Representações Ltda. Em caso de exclusão de alguns desses dados, a informação sobre a respectiva empresa ficaria incompleta. Faria sentido possuir todas as informações da empresa, com exceção de seu nome? Veja mais claramente no exemplo abaixo:

- Faturamento anual: R\$ 800.000,00
- Quantidade de funcionários: 38
- Quantidade de filiais: 2

Embora você saiba o que significam essas informações, elas não têm importância se você não puder associá-las a uma empresa. Nesse caso, no que lhe auxiliaria possuir essas três informações descritas?

O SGBD trabalha de uma forma específica com o intuito de manter dados íntegros, não permitindo a exclusão da informação “Nome da Empresa” se houver dados sobre ela em qualquer outra parte do banco.

## Acessos simultâneos ou concorrentes

Imagine que, em uma empresa que conta com uma área contábil de cerca de cinco funcionários, ao início do mês, deseja-se ver o balanço mensal do mês anterior. É comum que mais de uma pessoa tenha interesse em fazer essa consulta, e duas dessas pessoas tentam acessar a informação ao mesmo tempo. Com a utilização de um SGBD, é possível que essa operação se realize, ou seja, mais de uma pessoa consiga visualizar a mesma informação ao mesmo tempo.

Em uma situação mais crítica, as duas pessoas que estão lendo as mesmas informações contábeis ao mesmo tempo resolvem alterá-las. Você pode imaginar os problemas que poderiam ser gerados a partir dessas operações simultâneas? O SGBD não deixa que esses problemas aconteçam: prioriza as alterações feitas pelo primeiro usuário, fazendo com que a segunda pessoa interessada não consiga alterar os mesmos dados e precise aguardar até que a primeira pessoa finalize suas operações.

 **Registros** são conjuntos de dados armazenados em algum local, nesse caso, em um banco de dados.

## SEÇÃO 3

### Tipos de usuários de um SGBD

Existem quatro tipos de usuários de um SGBD, cada um responsável por algumas funções. Conheça mais detalhadamente esses usuários e suas responsabilidades:

- Administradores

São mais conhecidos como DBAs, do inglês *Data Base Administration*. Eles gerenciam o funcionamento do banco de dados, programam as devidas manutenções, tornam o ambiente de consultas e gravações de dados performático e aplicam as devidas correções de *software*, se for necessário. Trabalham no nível interno do banco de dados, preocupando-se com os **registros** e métodos de armazenamento.

- Desenvolvedores

Confeccionam os *softwares* que se conectarão ao banco de dados e darão condições para que os usuários possam extrair, alterar ou gravar informações no banco. Trabalham no nível conceitual, em que há preocupações com a engenharia do banco de dados e seus métodos de conexões.

- Usuários avançados

São pessoas que conseguem extrair informações do banco de dados por meio de programas específicos, possibilitando atender às requisições gerenciais da empresa. Alguns utilizam comandos diretos no banco de dados para uma busca mais direcionada de informações.

- Usuários simples

São usuários que se conectam ao banco por meio de *softwares* feitos pelos desenvolvedores para efetuar operações simples de alteração, consulta ou gravação de dados. Trabalham no nível externo do banco, em que a preocupação limita-se à parte operacional dos processos da empresa.

## SEÇÃO 5

### SGBD x sistemas de arquivos

Em menor escala de complexidade e de investimentos, existe também o sistema de arquivos, que não pode ser confundido com o SGBD.

O sistema de arquivos, segundo Ferreira, Italiano e Takai (2005), deve ser utilizado para suportar bancos de dados de baixa escala, como o de um acervo pessoal descrito no exemplo anterior. Não possui controle rígido sobre a redundância de dados e padronização, depende de uma aplicação específica para acesso a cada informação, além de representar os dados fisicamente, ao contrário do SGBD, que auxilia na eliminação de redundâncias, é capaz de permitir diversas aplicações acessando o banco de dados e representa conceitualmente os dados em conjunto com seus programas.

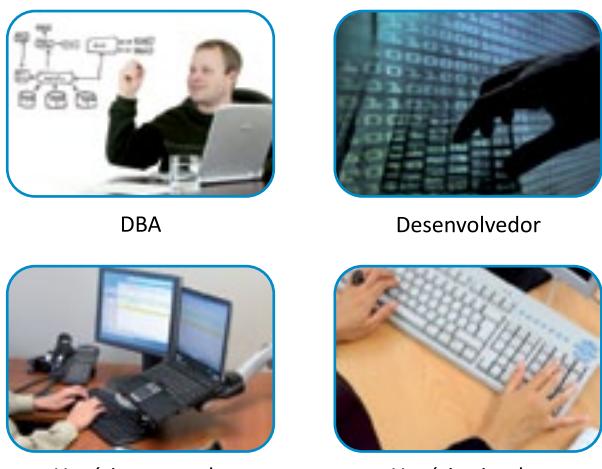


Figura 5: Tipos de usuário de um SGBD

## SEÇÃO 4

### Quando não utilizar um SGBD

A implantação de um SGBD exige um bom *hardware*, mão de obra especializada para configurar os devidos controles de segurança e controles de integridade e acessos simultâneos, o que acarreta em consumo de tempo de implementação e gera altos investimentos. Além disso, se houver má administração do SGBD, tanto a segurança quanto a integridade do banco poderão ficar comprometidas (UNIVERSIDADE DE MARÍLIA, [s.d.]).

Nessa linha, não podemos considerar a implementação de um SGBD para um sistema que não seja vital para o andamento de um negócio, não possua grande número de transações, tenha baixo número de usuários interagindo ou não tenha expectativas de mudanças.

Por exemplo, um banco de dados desenvolvido para controlar um acervo pessoal de livros, contendo 50 itens, sendo consultado por uma pessoa apenas e em uma frequência aproximada de um acesso por semana. Nesse caso, o banco de dados desenvolvido não necessita obrigatoriamente de um SGBD ativo. Por ser considerado um processo tradicional, o custo de implementação provavelmente não justificaria o benefício oferecido, sendo mais viável o uso de um sistema de arquivos.

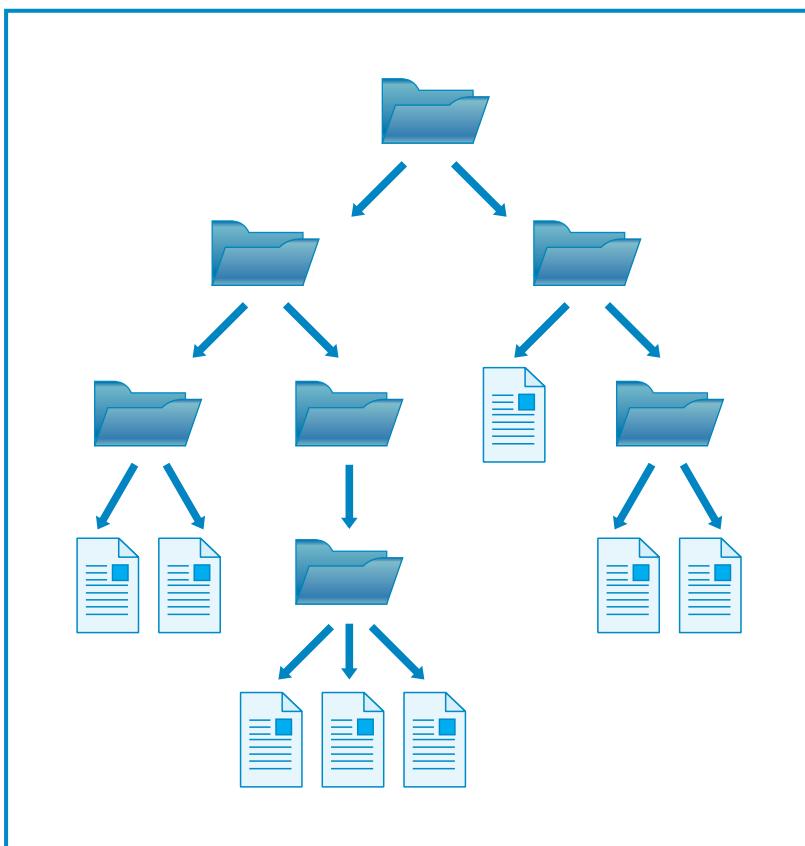
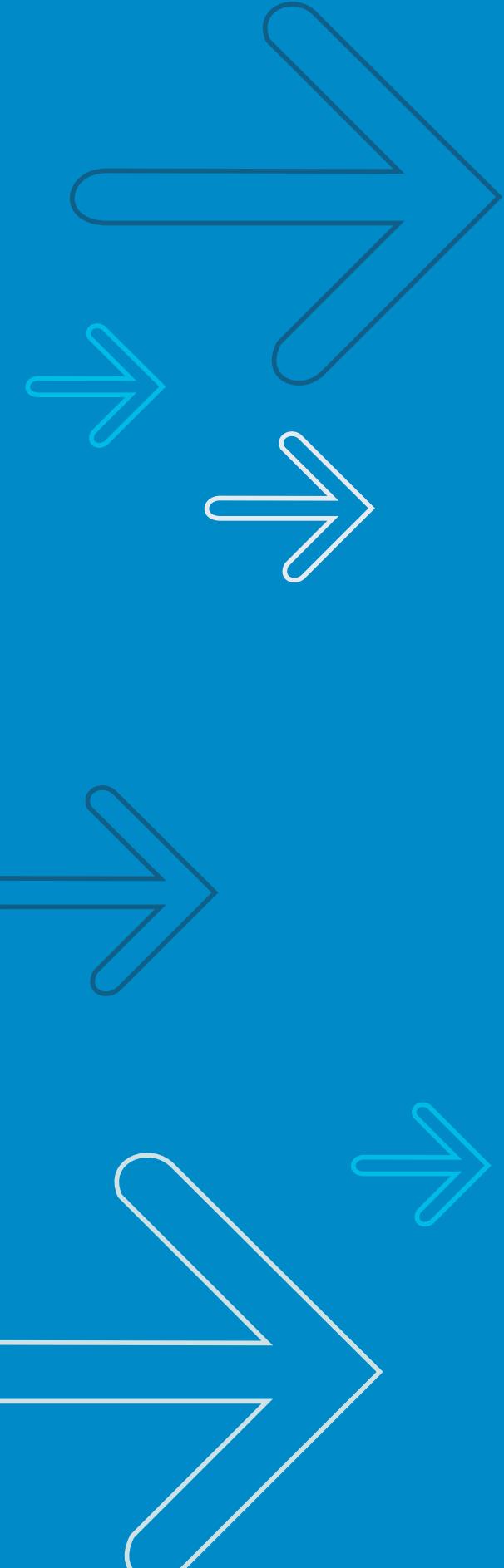


Figura 6: Representação de um sistema de arquivos

Como o sistema de arquivos não é a melhor forma de armazenar e gerenciar os dados de uma empresa, possui poucos recursos e falta controle efetivo de seus processos e usuários, este livro terá foco no estudo do banco de dados por meio de SGBDs, e não dos sistemas de arquivos



# Unidade de estudo 3

## Seções de estudo

- Seção 1 – Década de 1950 – O início**
- Seção 2 – Década de 1960 – banco de dados hierárquico**
- Seção 3 – Década de 1960 e início de 1970 – banco de dados em rede**
- Seção 4 – Década de 1970 – banco de dados relacionais**
- Seção 5 – Década de 1980 – banco de dados orientados a objetos**
- Seção 6 – Década de 1990 – bancos de dados objetos-relacionais**
- Seção 7 – Anos 2000 – bancos de dados avançados**

# A História do Banco de Dados

## SEÇÃO 1

### Década de 1950 o início

As primeiras formas de armazenagem de dados foram registradas no final da década de 1950, período em que se deu o início da informática em grande escala. Era necessário guardar tudo que era produzido em um computador, e os meios encontrados na época foram o armazenamento em cartões perfurados e fitas magnéticas.

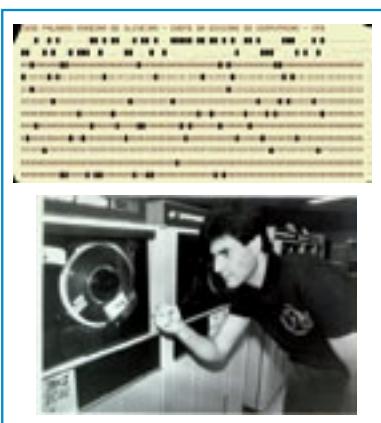


Figura 7: Exemplos de cartões perfurados e fitas magnéticas

Com o crescimento da informática, houve um impulso quantitativo de redes de computadores que geraram uma grande quantidade de processamento de dados (BARBOSA; INGRID, LABORDE, 2010), necessitando assim de uma armazenagem de dados estruturada e organizada, o que tornou obsoletas as fitas magnéticas. A seguir você verá a linha do tempo de evolução dos modelos de bancos de dados.

## SEÇÃO 2

### Década de 1960 – banco de dados hierárquico

Na década de 1960, com o surgimento dos discos rígidos e o crescimento da capacidade de armazenamento, iniciava-se o armazenamento de dados computacionais estruturado, no formato hierárquico, chamado de banco de dados hierárquico.

Nesse tipo de banco, os dados eram organizados em hierarquias, como um conjunto de árvores, utilizando-se de registros para representar os dados que eram relacionados entre si por meio de *links*. Veja a seguir outras características do modelo de dados hierárquico:

- O acesso aos dados era feito percorrendo-se as árvores, até os últimos “ramos”, onde estava o registro pretendido;
- O acesso às informações era feito de maneira sequencial;
- Os registros eram manipulados um de cada vez;
- As consultas eram complexas e demoradas.

A figura a seguir demonstra um diagrama representativo do modelo hierárquico de dados.

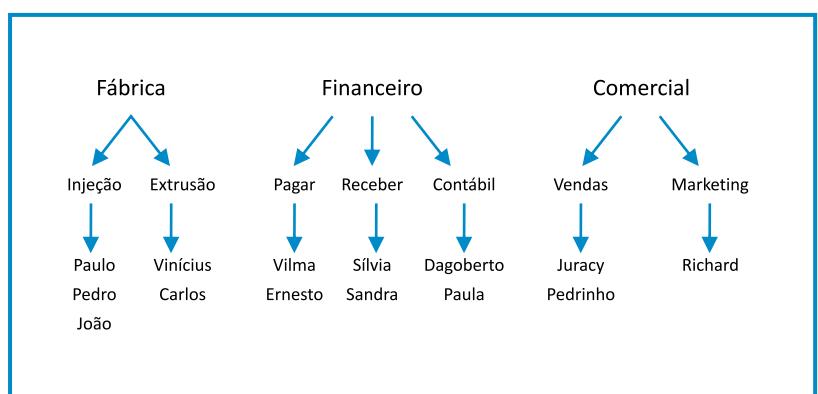


Figura 8: Organização dos dados em um banco de dados hierárquico

## SEÇÃO 3

### Década de 1960 e início de 1970 – banco de dados em rede

Ainda na década de 1960, surgiu, por meio de Charles Bachman, o banco de dados em rede, que é composto de uma estrutura mais completa do que o modelo hierárquico, sendo considerado, inclusive, uma extensão deste. Algumas características do modelo de dados em rede são:

- os dados são representados por uma coleção de registros que possui várias associações, eliminando o conceito de hierarquia;
- o acesso aos dados é dado por meio de relacionamentos;
- as consultas continuam sendo complexas, forçando o desenvolvedor dos sistemas a pensar com como percorrer os *links* até chegar ao seu destino.

Veja, na figura a seguir, um modelo de dados em rede:

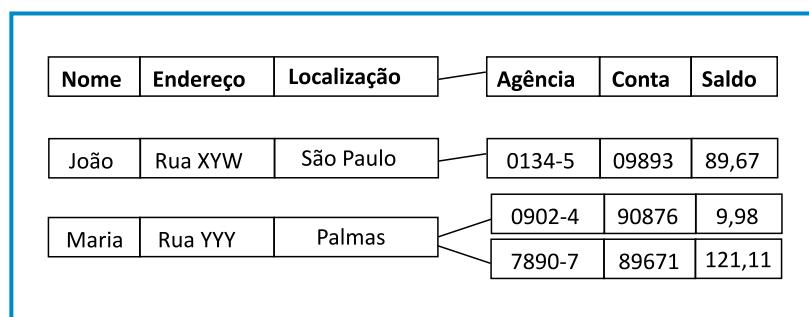


Figura 9: Organização dos dados em um banco de dados em rede

## SEÇÃO 4

### Década de 1970 – banco de dados relacionais

Nos anos de 1970, Edgar Codd apresenta o modelo relacional, que se tornou um diferencial entre os bancos de dados já existentes, sendo considerado o sucessor dos modelos em rede e hierárquicos, contribuindo decisivamente para a utilização em grande escala dos bancos de dados. Nesse tipo de banco de dados, os dados são armazenados em tabelas interligadas por meio de relacionamentos, estrutura que viabiliza a pesquisa de dados no banco, simplifica o armazenamento e facilita a visão humana sobre sua estrutura e as informações armazenadas nele.

A figura a seguir demonstra graficamente como os dados estão armazenados em um banco de dados do tipo relacional:

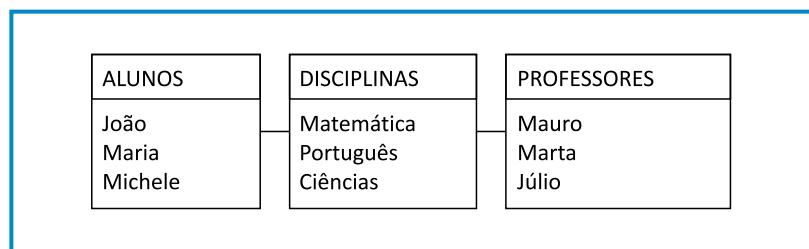


Figura 10: Organização dos dados em um banco de dados relacional

Por esses motivos, apesar de ter surgido na década de 1970 e somente ser utilizado em massa a partir da década de 1980, ele é o banco mais utilizado até hoje nas empresas e será o foco de aprendizado neste livro.

Veja, a seguir, outras vantagens na utilização de bancos de dados relacionais (BDR), segundo Abreu e Machado (2004).

- Visão múltipla de dados.
- Melhor comunicação entre armazenagem de dados e usuários.
- Redução acentuada no desenvolvimento de aplicações e tempo gasto na manutenção.
- Melhoria na segurança dos dados.
- Agilidade nas consultas aos dados, gerando informações de qualidade na tomada de decisões da empresa.

Ainda conforme Abreu e Machado (2004), o conceito principal dos bancos de dados relacionais vem da teoria de que não importa para os usuários saber onde estão armazenados os objetos e nem como eles são. Basta lidar com os objetos contando com um conjunto de funções de alto nível e linguagem de programação de banco de dados, como é o *Structure Query Language* (SQL), que será visto na unidade 13.

## SEÇÃO 5

### Década de 1980 – bancos de dados orientados a objetos

Em meados da década de 1980, iniciaram os estudos sobre bancos de dados orientados a objetos (BDOO), que visavam a criar um tipo de banco de dados que pudesse interagir mais eficazmente com aplicações desse tipo, ou seja, aplicações que utilizam linguagem de programação orientada a objetos, como: Java, C++ ou *Smalltalk*, que armazenam objetos complexos, como textos, imagens e gráficos. Esse tipo de banco procura armazenar os dados como objetos, somente podendo ser manipulados pelos métodos da classe à qual esses objetos pertencem, trazendo maior segurança e agilidade, já que um objeto (dado) pode ser encontrado seguindo os ponteiros diretamente, sem necessidade de buscas por meio das tabelas, como é feito no modelo relacional.

Seu SGBD é diferente e conhecido como SGBDOO, ou seja, sistema gerenciador de banco de dados orientado a objetos, que auxilia na modelagem das estruturas complexas utilizadas no BDOO.

Além de privilegiar o armazenamento de dados em formato de objetos e trazer maior agilidade na busca desses objetos dentro do banco, o BDOO possui também as seguintes vantagens:

- armazena não somente dados em tabela, mas também dados em outros formatos, como imagens, documentos e sons;
- é utilizado em aplicações complexas, como de projetos de arquitetura, engenharia, empresas de telecomunicações ou de multimídia;
- manipula os objetos de forma consistente e segura;
- padroniza dos dados.

Observe, a seguir, uma representação gráfica do modelo de dados orientado a objetos:

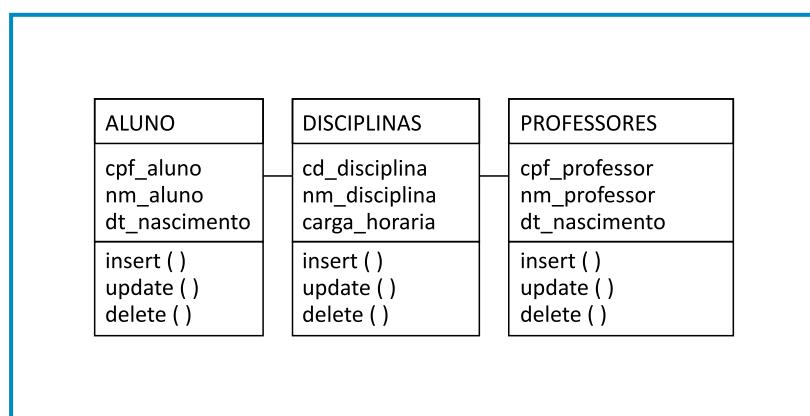


Figura 11: Modelo de dados orientado a objetos

A desvantagem na utilização desse tipo de banco está no alto custo de aquisição de novas tecnologias e no investimento a ser feito em treinamentos para capacitar profissionais na administração e desenvolvimento de aplicações especificadas para o BDOO.

## SEÇÃO 6

### Década de 1990 – bancos de dados objetos-relacionais

Com o aumento da interconectividade, outros tipos de dados são armazenados em bancos de dados, o que levou à necessidade de reestruturação dos bancos de dados relacionais para tratar essa demanda, criando-se então os bancos de dados objetos-relacionais.

A seguir, você tem uma breve explanação desse tipo de banco de dados e dos motivadores de seu surgimento.

#### ⇒ A explosão da web

Na metade dos anos 1990, viu-se uma explosão na utilização da internet, que impulsiona o mercado de alta disponibilidade visando a prover acesso em tempo real 24 horas por dia, sete dias na semana.

Antes mesmo de finalizar essa década, já se evidenciava um alto crescimento na utilização da web. Segundo Oliveira e Santos (1998?), em 1998, havia 30 milhões de servidores web, entretanto a previsão para 1999 era de que existissem cem milhões de servidores web espalhados pelo mundo, cerca de 320 vezes mais do que no início da mesma década. Todo esse crescimento fez alavancar todos os ramos da informática, desde os componentes eletrônicos e sua capacidade de processamento até os softwares e seu poder de automatização. Logicamente, os bancos de dados acompanharam esse crescimento e, mais que isso, até influenciaram essa ascensão, suportando o grande volume de consultas na web, a gravação de dados multimídia e permitindo maior segurança nas transações

de mercado eletrônico, como a verificação de dados financeiros em uma compra com cartão de crédito.

## » A utilização de bancos de dados objetos-relacionais

Para suprir as carências do modelo relacional proporcionando naturalidade em modelar objetos complexos sem abrir mão de toda sua carga tecnológica adquirida em anos de pesquisas e desenvolvimentos, surgem os bancos de dados objetos-relacionais (BDOR) (VIDAL, 200-?) que continuam oferecendo a mesma interface relacional à qual os desenvolvedores já estão acostumados e permitem trabalhar outros objetos, além de dados em tabelas.

Você já deve ter percebido a ligação entre os BDOR e a utilização da internet, certo? Se você acha que essa ligação se dá por conta da necessidade de armazenamento de objetos como documentos, sons e imagens, você está certo!

O BDOR consegue tratar objetos complexos sem a necessidade de utilização de um BDOO, o qual exigiria altos investimentos em atividade humana e em *hardwares*, objetivando boa *performance* no ambiente.

Os bancos de dados objetos-relacionais mesclam as ideias dos BDR e BDOR satisfazendo aos dois quesitos. Conforme Korth, Silberschatz e Sudarshan (1999, p. 273), “[...] modelos de dados relacionais-objetos estendem o modelo de dados relacional fornecendo um tipo de sistema mais rico, incluindo orientação a objeto e acrescentando estruturas a linguagens de consulta relacionais [...]”.

Além das vantagens apresentadas, vale destacar que a utilização dos BDOR não afeta o desempenho do sistema. Ao contrário, consegue tratar os objetos com a mesma *performance* que os bancos de dados orientados a objetos.

DOCUMENTOS		
TÍTULO	AUTOR	TAMANHO
Aceite do Cliente	José Pereira	102
Concessão de Crédito	Marlene Souza	55

Figura 12: Modelo de banco de dados objetos-relacionais

## SEÇÃO 7

### Anos 2000 – bancos de dados avançados

A entrada do terceiro milênio não altera as tendências de comunicações instantâneas e trocas de arquivos *on-line*. Pelo contrário, a utilização da internet continua influenciando a sociedade em seus hábitos e as empresas em seus métodos de armazenamento. Bancos de dados móveis, pessoais, de multimídia, espaciais e geográficos destacam-se nesse período, em que a mobilidade e o nível de qualidade das imagens armazenadas aumentam exponencialmente. A seguir, acompanhe uma breve explicação de cada um desses modelos segundo Korth, Silberschatz e Sudarshan (1999).

- **Móveis e pessoais**

Ao contrário dos bancos de dados comerciais de grande porte, estes, por conta da expansão de dispositivos pessoais móveis e da internet sem fio, armazenam os dados diretamente nos dispositivos móveis de seus proprietários, que se tornam os administradores desses bancos.

- **Multimídia**

São responsáveis pelo armazenamento de arquivos como sons, imagens e vídeos favoravelmente em grande quantidade, trazendo possibilidades de buscas por meio dos reais significados desses objetos e de maior confiabilidade de armazenamento.

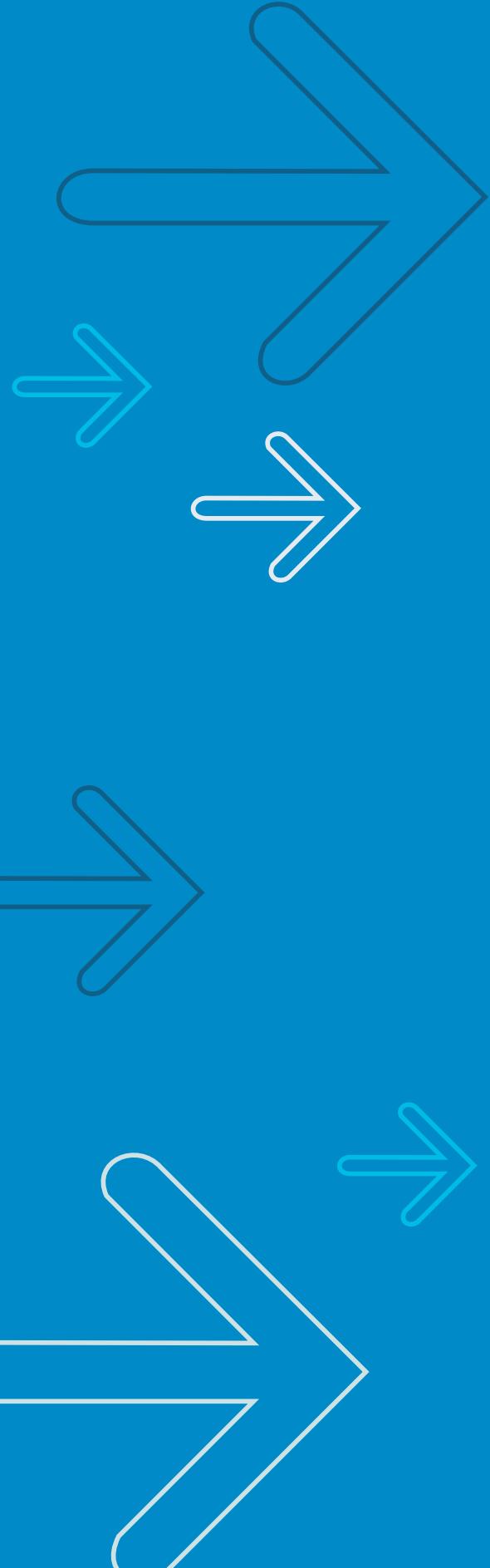
- **Espaciais e geográficos**

Armazenam dados relacionados a localizações espaciais e oferecem suporte para consultas com base nessas localizações.



Figura 13: Simbolização de bancos de dados móveis, multimídias e geográficos

Na próxima unidade, você entenderá o significado de cada tipo de modelagem baseado em conceitos teóricos e pequenos exemplos, preparando você para as unidades 8, 9 e 10, nas quais a modelagem de dados será vista em um nível mais avançado. Até mais.



# Unidade de estudo 4

## Seções de estudo

- Seção 1 – O que é modelagem de dados
- Seção 2 – Modelagem conceitual
- Seção 3 – Modelagem lógica
- Seção 4 – Modelagem física

# Conceitos de modelagem de dados

## SEÇÃO 1

### O que é modelagem de dados

Em uma empresa existem vários processos formados por conjuntos de ações de seus funcionários, que fazem com que os negócios caminhem dentro da normalidade. Provavelmente, você já ouviu falar que a falta de processos em uma empresa pode gerar certos danos, não ouviu? No início dos estudos em informática, você aprendeu que os sistemas não criam novos processos, apenas automatizam e melhoram os processos já existentes em uma organização.

Modelagem de dados é uma técnica utilizada para planejar o armazenamento das informações de uma organização em um banco de dados, ou seja, com a modelagem de dados, é possível transformar os processos da vida real de uma organização para o mundo virtual e sistêmico. Seu resultado é conhecido como modelo de dados, que simplesmente descreve a forma como os dados estão sendo armazenados dentro do banco de dados.

A modelagem de dados propicia o desenvolvimento de um projeto de dados e é dividida em três grupos, os quais você verá nas seções seguintes.

## SEÇÃO 2

### Modelagem conceitual

A modelagem conceitual retrata puramente uma transformação do mundo real para o mundo virtual. Nesse tipo de modelagem, pode-se afirmar que estamos modelando o mundo daquela empresa, pessoa, organização, enfim, o proprietário do respectivo sistema, transformando essa realidade em um “minimundo” desenhado.

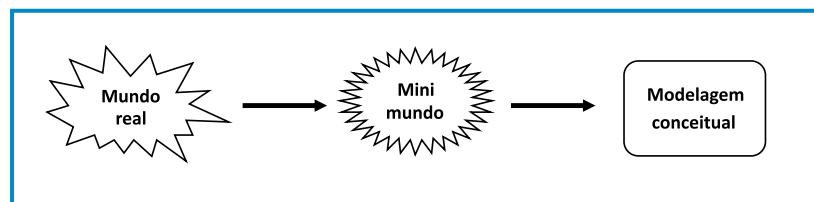


Figura 14: Mundo real x mundo virtual

O responsável por confeccionar esse desenho, chamado então de modelo conceitual, é o analista de sistemas, e o responsável por executá-lo e implementá-lo efetivamente em um banco de dados é o desenvolvedor, aquele mesmo que você estudou na unidade de estudo 2, lembra?

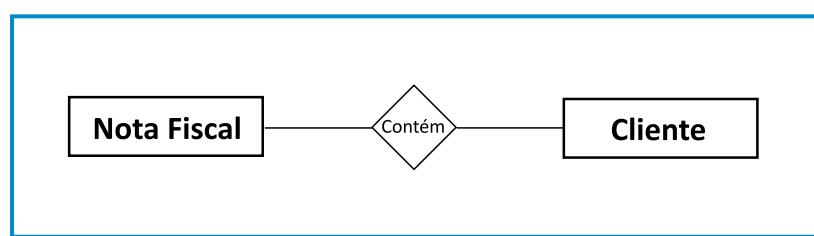


Figura 15: Representação básica de um modelo conceitual

É importante observar que a confecção desse modelo não depende de nenhum tipo de tecnologia ou fabricante de SGBDs. É um modelo de alto nível e, mesmo depois de pronto, somente deverá ser utilizado para o entendimento do negócio da respectiva organização.

## SEÇÃO 3

### Modelagem lógica

➤ **Modelo Lógico relacional** modelo de dados que utiliza a abordagem relacional para ser confeccionado

➤ **Campos** no caso de modelo de dados, campos relacionais são sinônimos de atributos, porém fala-se campos quando já se está construindo o modelo físico de dados. A denominação “colunas” também pode ser usada para simbolizar os campos.

A modelagem lógica tem seu início a partir da modelagem conceitual e considera os diferentes tipos de abordagem de dados, por exemplo, as abordagens hierárquica, de rede, relacional e orientada a objetos, vistas na unidade de estudo 3.

Em outras palavras, para se construir um modelo lógico, não é relevante considerar as características do SGBD onde será implementado, mas é necessário saber qual abordagem será utilizada (no caso deste livro, será sempre utilizada a abordagem relacional, devido à sua ampla aceitação).

Um **modelo lógico relacional** descreve as estruturas que estarão contidas no banco de dados de acordo com as possibilidades permitidas pela abordagem escolhida (ABREU; MACHADO, 2004). Quem o ajuda a implementar é o analista de negócios, também conhecido como usuário avançado (que você conheceu na unidade de estudo 2)

Levando em consideração a abordagem relacional, são definidas, nessa etapa, as tabelas, os padrões de nomenclatura e são definidas as chaves primárias e estrangeiras que serão estudadas na unidade 9.

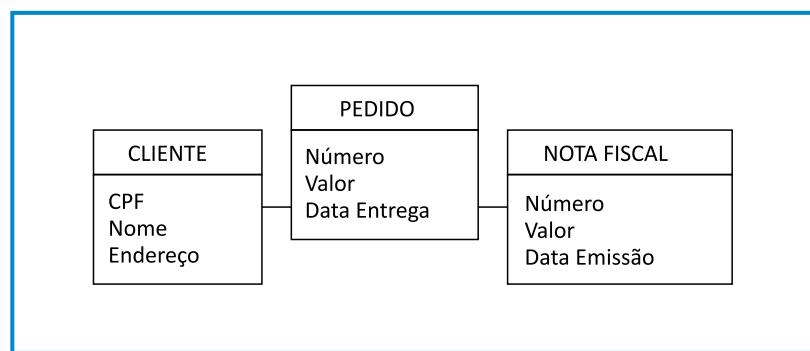


Figura 16: Representação básica de um modelo lógico relacional

## SEÇÃO 4

### Modelagem física

Ao contrário das modelagens conceitual e lógica, a modelagem física leva em consideração o SGBD que será utilizado para gerenciar o banco de dados e todas as suas limitações. Um modelo físico é derivado de um modelo lógico e descreve como os dados serão realmente armazenados dentro do banco de dados, demonstrando os tamanhos dos **campos** e de que tipo são, se são do tipo números, textos, datas ou outros.

O modelo físico é o fim do processo de modelagem e é desenvolvido pelo desenvolvedor juntamente com o administrador do banco de dados (DBA) de acordo com as regras do SGBD escolhido para suportar o sistema. Nessa etapa, o DBA tem preocupações com espaço físico, tipos de registros com a forma como eles afetarão a *performance* do banco.

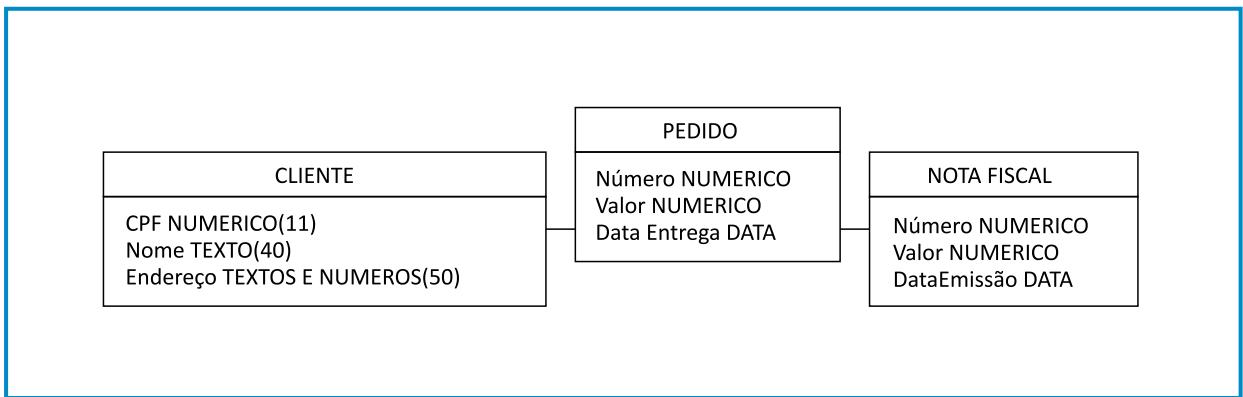
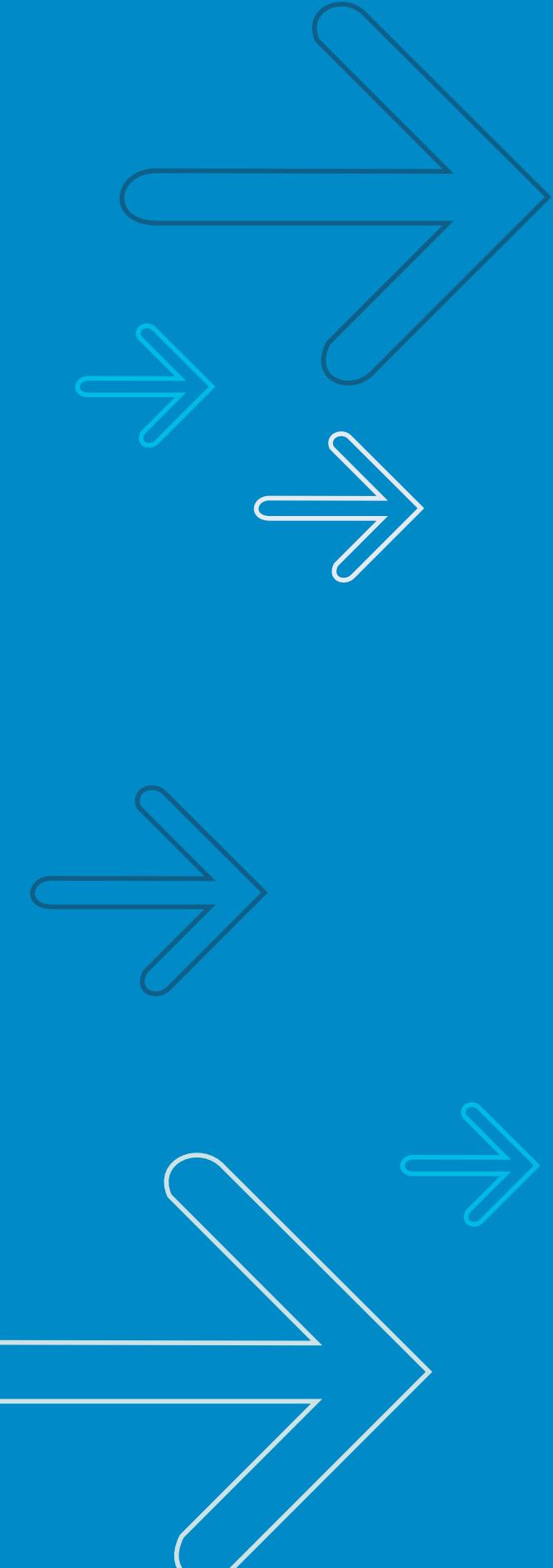


Figura 17: Modelo físico relacional

A essa altura você deve estar se perguntando: “já que um modelo é a sucessão do outro, como eu faço então para fazer a transição entre os modelos, confeccionar um projeto de banco de dados, enfim, gerar um banco de dados real?” Calma! Isso você verá a partir da unidade 8. Antes disso, conheça o modelo de entidade-relacionamento, na próxima unidade.



# **Unidade de estudo 5**

## **Seções de estudo**

- Seção 1 – O que é modelo de entidade-relacionamento?**
- Seção 2 – Entidades**
- Seção 3 – Atributos**
- Seção 4 – Relacionamentos**
- Seção 5 – Cardinalidade**

# Modelo de Entidade-Relacionamento

## SEÇÃO I

### O que é modelo de entidade-relacionamento?

Cada sistema é construído para sanar uma necessidade de negócio de uma organização que, para isso, deve fazer parte da elaboração do sistema e ser conhecida pelo analista de sistemas, pelo desenvolvedor e também pelo conhecedor de negócios da empresa na qual o sistema será implantado. Esses profissionais terão a tarefa de adequar toda a realidade da organização em um modelo virtual que represente as atividades do sistema e que, inicialmente, pode ser feito até mesmo em uma folha de papel. Esse modelo é conhecido como modelo de entidade-relacionamento, ou simplesmente MER.

O MER tem a finalidade de, conceitualmente, descrever os dados armazenados em um banco de dados e suas interligações.

Na unidade anterior, você conheceu os tipos de modelagem de dados e, consequentemente, viu que a modelagem conceitual visa a virtualizar o mundo real. Essa virtualização pode ser facilitada quando se cria o MER do conjunto de processos da respectiva organização, mostrando de que forma as informações constantes nas tabelas estão organizadas, porém, sem demonstrar o fluxo dessas informações entre as tabelas.

“A modelagem [no caso, através do MER] é como a arte fotográfica, prepara-se a câmera e tira-se a foto, sem se importar com os movimentos” (ABREU; MACHADO, 2004, p. 29).

Desse processo origina-se o diagrama de entidade-relacionamento (DER). Para o DER, é preciso criar as entidades que são compostas por atributos e ligá-las por meio de relacionamentos. Veja um exemplo:

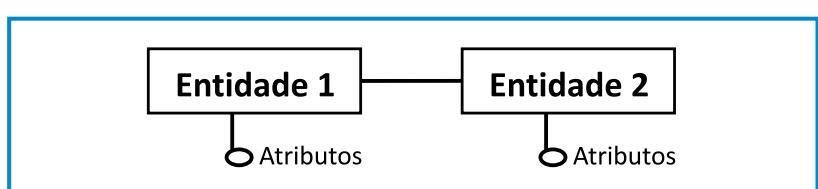


Figura 18: Definição do MER

A partir da próxima seção, você entenderá melhor como montar um MER. Acompanhe.

## SEÇÃO 2

### Entidades

O primeiro grande passo, e talvez o mais difícil, para se montar um modelo MER é definir as entidades que serão a base para a modelagem de entidade-relacionamento. Mas o que são entidades?

Uma entidade nada mais é do que:

- algo que existe no mundo real dentro de uma organização, um objeto ou **uma** pessoa;
- algo com relevância para os processos de negócio;
- algo cujas informações se deseja armazenar;
- algo que representa um padrão específico de características.
- Por exemplo: em um supermercado há produtos, notas fiscais, clientes e fornecedores.

Todos esses itens podem ser considerados entidades, pois existem na vida real, deseja-se armazenar informações sobre eles e possuem padrões de características (todo cliente possui um nome e endereço, todo produto tem um código e um preço e, em toda nota fiscal há valores e produtos).



Figura 19: Interpretação de entidades reais

Cada entidade é representada por um retângulo dentro do DER, conforme você poderá ver na figura 20.

Mas você deve estar se perguntando: “como essas entidades poderão ser identificadas sem que ocorram maiores erros?” Segundo Abreu e Machado (2004), mesmo sendo este um tema de difícil formalização, há processos que podem ser seguidos para que se identifiquem corretamente as entidades de um sistema.

O primeiro passo é imaginar um sistema a ser desenvolvido. Então, para exemplificar, imagine um sistema de vendas em varejo. Nesse sistema, deve-se imaginar o fluxo do processo juntamente com as pessoas ou objetos reais que participam dele. Por exemplo, o vendedor vende ao cliente algum produto e gera uma nota fiscal.

Você concorda que **vendedor**, **cliente**, **produto** e **nota fiscal** representam padrões específicos de características, são importantes para os negócios da empresa proprietária do sistema e, consequentemente, deseja-se armazenar informações sobre eles, certo? Então, estão definidas as primeiras entidades do sistema: vendedor, cliente, produto e nota fiscal.

Para completar a definição de todas as entidades do sistema, basta executar a mesma tarefa para os outros processos da área de vendas, como pedidos, formas de pagamento, verificação de crédito, entre outras.

## SEÇÃO 3

### Atributos

Cada entidade possui seus atributos, que nada mais são do que os dados que são associados a cada ocorrência de uma entidade ou de um relacionamento (HEUSER, 2004), ou seja, as características comuns das entidades.

**Exemplo 1:** um cliente, do sistema de vendas em varejo, possui CPF, nome, endereço e data de nascimento. Esse conjunto de características pode ser aplicado para todas as pessoas, portanto é considerado **atributo da entidade Cliente**.

**Exemplo 2:** para a entidade Produtos, temos outro conjunto de características em comum: marca, código, nome, tipo da embalagem, data de validade e nome do fornecedor. Cada uma dessas características é considerada **atributo da entidade Produto**.

Um atributo é representado no DER por meio de uma elipse logo abaixo da sua respectiva entidade.

**Vendedor**    **Cliente**    **Produto**    **Nota Fiscal**

Figura 20: Entidades representadas no DER

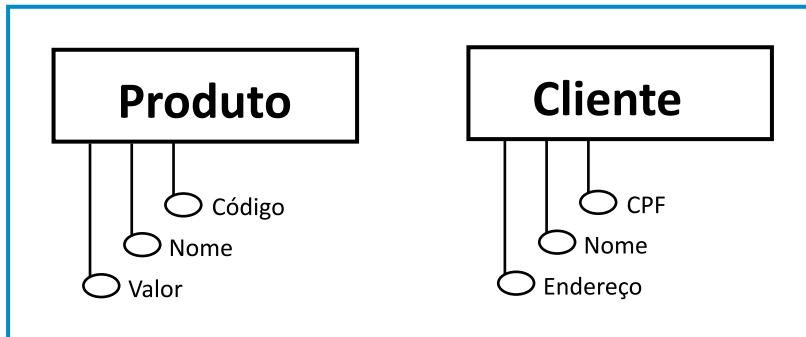


Figura 21: Representação das entidades Produto e Cliente com seus atributos

Os valores de cada atributo, como nome do cliente = João da Silva, preço do produto = R\$ 3,00 é que formam a maior quantidade de dados armazenados no banco de dados e possibilitam o funcionamento do sistema de uma empresa. Nas unidades 12 e 13 você aprenderá mais sobre como atribuir dados para cada atributo de cada entidade.

## SECÃO 4

### Relacionamentos

É de fundamental importância para a concretização de trabalhos de modelagem o entendimento dos conceitos de relacionamentos e sua aplicação pelas pessoas que participaram, e principalmente por quem não participou do processo de modelagem, mas, que ao vê-lo, consegue compreender o funcionamento do sistema.

Os relacionamentos representam o conjunto de associações entre ocorrências de entidades, sendo responsáveis por unir as entidades que possuem alguma relação entre si, possibilitando que as entidades compartilhem informações e evitando que dados sejam gravados em duplicidade.

Portanto, para que um relacionamento ocorra, uma entidade deve se “interessar” pelas informações de outra, utilizando aquela informação que somente a outra entidade possui.

Para facilitar a compreensão desses conceitos, será utilizado o mesmo exemplo do sistema de vendas de varejo apresentado nas unidades 2 e 3, onde foram definidas as entidades e seus respectivos atributos.

Se você imaginar as entidades **nota fiscal** e **cliente**, perceberá que elas têm certa relação. Você saberia informar que relação há entre essas entidades?

Acompanhe: um cliente deseja receber a nota fiscal de sua compra, e essa nota fiscal deve conter o nome do cliente, certo? Está aí a relação! Se uma nota fiscal possui o nome do cliente, esta tem interesse em ir até a entidade Cliente e buscá-lo, sendo que a entidade Cliente deverá compartilhar essa informação com a entidade nota fiscal e, se necessário, também com outras.

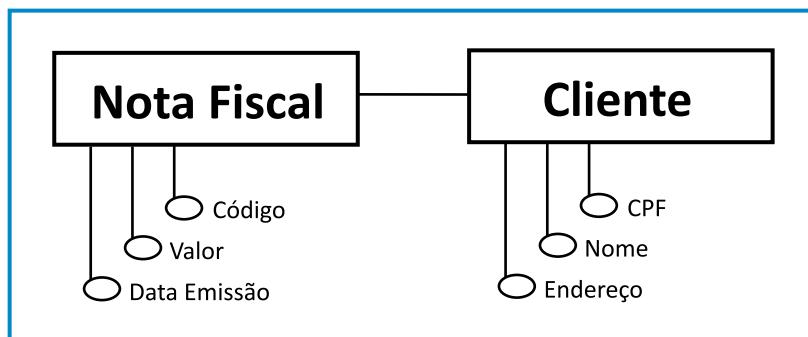


Figura 22: Relacionamento entre entidades nota fiscal e cliente

Você já reparou nos verbos que explicam os relacionamentos? São apenas fatos reais. Preste atenção:

**Cliente recebe** a nota fiscal, nota fiscal **possui** o nome do cliente, vendedor **emite** a nota fiscal, nota fiscal **contém** produtos etc.

Os relacionamentos de entidades nos bancos de dados reproduzem os relacionamentos reais que há nas entidades das organizações e são identificados simplesmente por verbos. A figura a seguir representa os relacionamentos e suas ações. Observe.

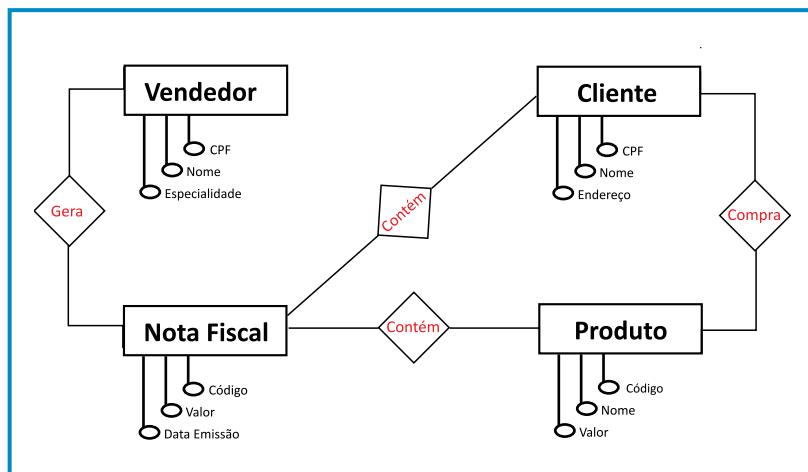


Figura 23: Parte do DER do módulo comercial de uma empresa de vendas de varejo

Os verbos representam expressões dos fatos reais e são representados no diagrama de entidade-relacionamento por meio de um losango, conforme visto na figura 23.

Cada relacionamento possui um tipo de restrição, que representa o número de vezes que uma entidade pode se relacionar com outra. Essa restrição chama-se cardinalidade e é assunto da próxima seção.

## SEÇÃO 5

### Cardinalidade

A definição literal de cardinalidade é representada pelo número (mínimo, máximo) de ocorrências de uma entidade associada a uma ocorrência de outra entidade por meio do relacionamento (HEUSER, 2004).

Em outras palavras, significa o número de ações que uma entidade pode executar em outra entidade relacionada. Por exemplo, quantas ocorrências pode haver entre a entidade Cliente e a entidade Nota fiscal? Ou seja, quantas notas fiscais o cliente poderá receber e quantos clientes poderão estar contidos em uma nota fiscal?

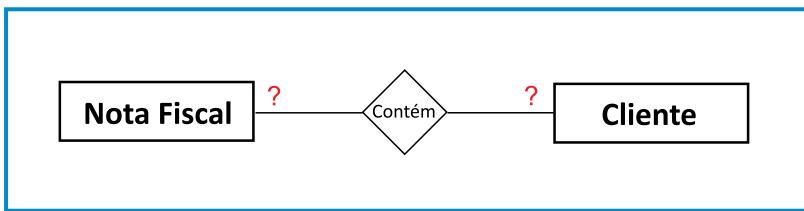


Figura 24: Início da representação da cardinalidade

Sempre existirá esse grau de relacionamento entre entidades que se relacionam entre si, e deverá ser definido de acordo com os requisitos do sistema. Há três tipos de cardinalidade existentes: um para um, um para muitos e muitos para muitos.

### ➤ Relacionamento um para um

O relacionamento um para um é bastante raro no meio corporativo. Significa que cada ocorrência da entidade A acontece apenas uma vez com a entidade B, assim como o inverso também é verdadeiro.

**Exemplo:** a entidade departamento e a entidade gerente relacionando-se entre si. O funcionário denominado gerente pode gerenciar apenas um departamento, assim como um departamento de uma empresa pode ser gerenciado por apenas um gerente, então o relacionamento entre departamento e gerente é de um para um.

Para cada entidade relacionada, deve-se representar seu grau de relacionamento por meio do seu símbolo ao lado interno da entidade, preferencialmente acima da linha que representa o próprio relacionamento.

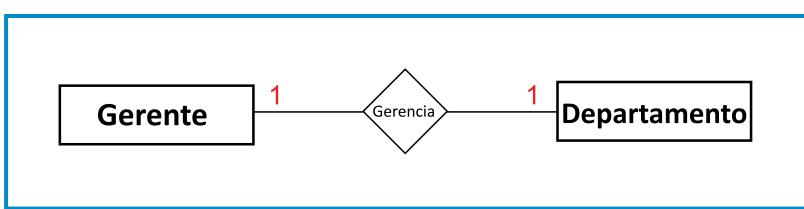


Figura 25: Um gerente gerencia um departamento e um departamento somente pode ser gerenciado por um gerente

### ➤ Relacionamento de um para muitos ou de muitos para um

Esse tipo de cardinalidade é o mais comumente encontrado no mundo real e, consequentemente, nas modelagens dos sistemas. É também denominado de relacionamento básico entre entidades. Indica que uma ocorrência da entidade A pode se relacionar com muitas ocorrências da entidade B, no entanto o inverso não é verdadeiro.

Utilizando o exemplo do sistema de vendas, é possível afirmar que um vendedor pode gerar várias notas fiscais para os clientes ( $N$  notas fiscais), porém cada nota fiscal possui apenas um cliente descrito, não sendo possível que a mesma nota fiscal seja gerada para dois clientes ao mesmo tempo.

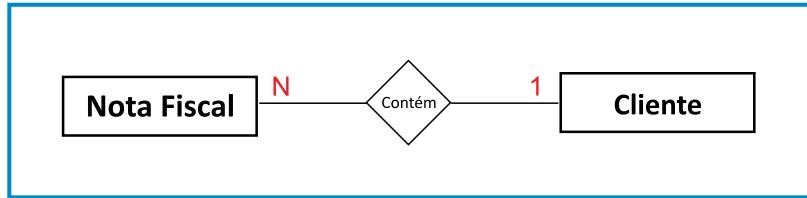


Figura 26: Relacionamento do tipo muitos para um entre Nota fiscal e Cliente

Dessa forma, esse relacionamento é considerado de um para muitos. Observe que o símbolo para muitos é a letra “N”.

Todo relacionamento deve ser lido nos dois sentidos. Portanto, na figura anterior, lê-se: Um cliente gera várias notas fiscais, mas cada nota fiscal é gerada para um cliente.

### ➤ Relacionamento muitos para muitos

Um relacionamento do tipo muitos para muitos representa que várias ocorrências da entidade A relacionam-se com várias ocorrências da entidade B, sendo que, consequentemente, o inverso é verdadeiro.

Para facilitar o entendimento, veja um exemplo conforme nosso sistema de vendas alocado em um supermercado, que possui as entidades Produto e Cliente. Em geral, um cliente pode comprar vários produtos, assim como um determinado produto pode ser adquirido por vários clientes. Tem-se então um relacionamento N para N.

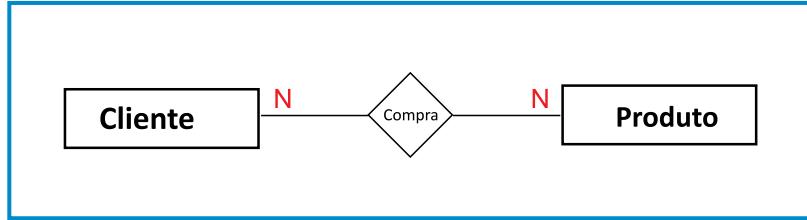


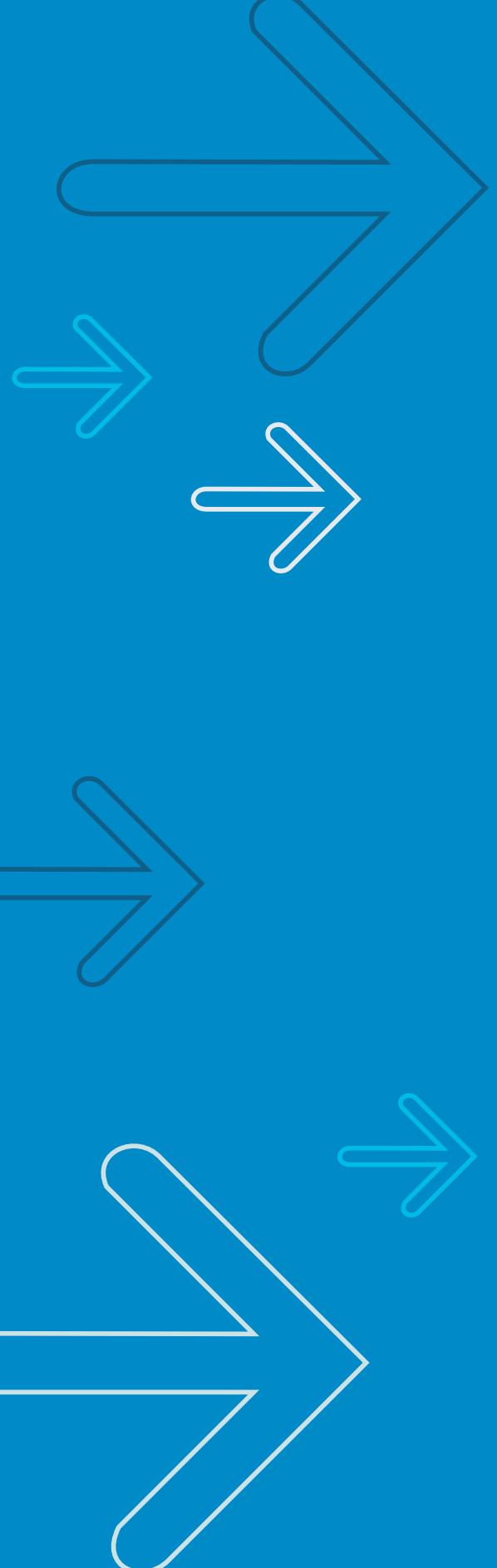
Figura 27: Relacionamento N para N entre Cliente e Produto

Muitos autores também consideram as expressões N para M, M para N ou ainda M para M. Neste livro, você verá apenas a expressão N para N, porém todas elas possuem o mesmo significado: um relacionamento de muitos para muitos.

No relacionamento N para N, há uma particularidade importante: a cada relacionamento desse tipo, deve ser criada uma tabela auxiliar entre as duas tabelas principais. Se a entidade Cliente se relaciona de N para N com a entidade Produto, então deverá ser criada uma nova tabela **Cliente\_Produto**, que trará informações das duas tabelas demonstrando, por exemplo, o valor que cada cliente pagou em um determinado produto em uma determinada data.

Para exemplificar melhor, considere um sistema de uma universidade em que a entidade Disciplina se relaciona com a entidade Aluno, de N para N, e gera uma tabela auxiliar contendo as notas de cada aluno e sua média final.

A tabela auxiliar se relaciona de forma N para um com as outras duas tabelas e contém as chaves primárias de cada uma das outras duas. Calma! Não se preocupe. Esse conceito você verá detalhadamente na unidade 9. Enquanto isso, veja como se faz um projeto de banco de dados.



# Unidade de estudo 6

## Seções de estudo

Seção 1 – O que é um projeto?

Seção 2 – Projeto de banco de dados

# Projeto de Banco de Dados

## SEÇÃO 1

### O que é um projeto?

Por definição, projeto é um

"[...] esforço temporário empreendedor para criar um produto, serviço ou outros tipos de resultados

(PROJECT MANAGEMENT INSTITUTE, 2008, p. 11).

Todo projeto é temporário, tem começo, meio e fim, possui prazos, orçamentos de despesas ou investimentos e é realizado e controlado por pessoas.

Segundo Araújo e Matos (2010), um projeto pode ser realizado em todos os níveis estratégicos da organização, inclusive na área de tecnologia da informação. Deve ter um gerente (do projeto), que tem a incumbência de aplicar todo o seu esforço para alcançar os objetivos propostos, sabendo identificar as necessidades, balancear as demandas de **escopo** e gerenciar as expectativas do cliente, podendo este ser uma organização qualquer ou até mesmo outra área da mesma organização.

Agora que você já sabe o que é um projeto, prepare-se para aprender a confeccionar um projeto de banco de dados!

## SEÇÃO 2

### Projeto de banco de dados

A confecção de um projeto de banco de dados é muito importante para a consistência das informações após o banco implementado. Assim como em qualquer projeto, primeiro, planeja-se quantas vezes e quanto tempo for necessário, para, depois disso, executar as ações de implementação do banco de dados.

Um banco de dados bem planejado permite consultas rápidas e eficazes à base de dados, ao contrário de um banco de dados que não teve planejamento e pode gerar erros nos retornos das consultas, apresentando informações não confiáveis, além de degradar a *performance* do sistema.

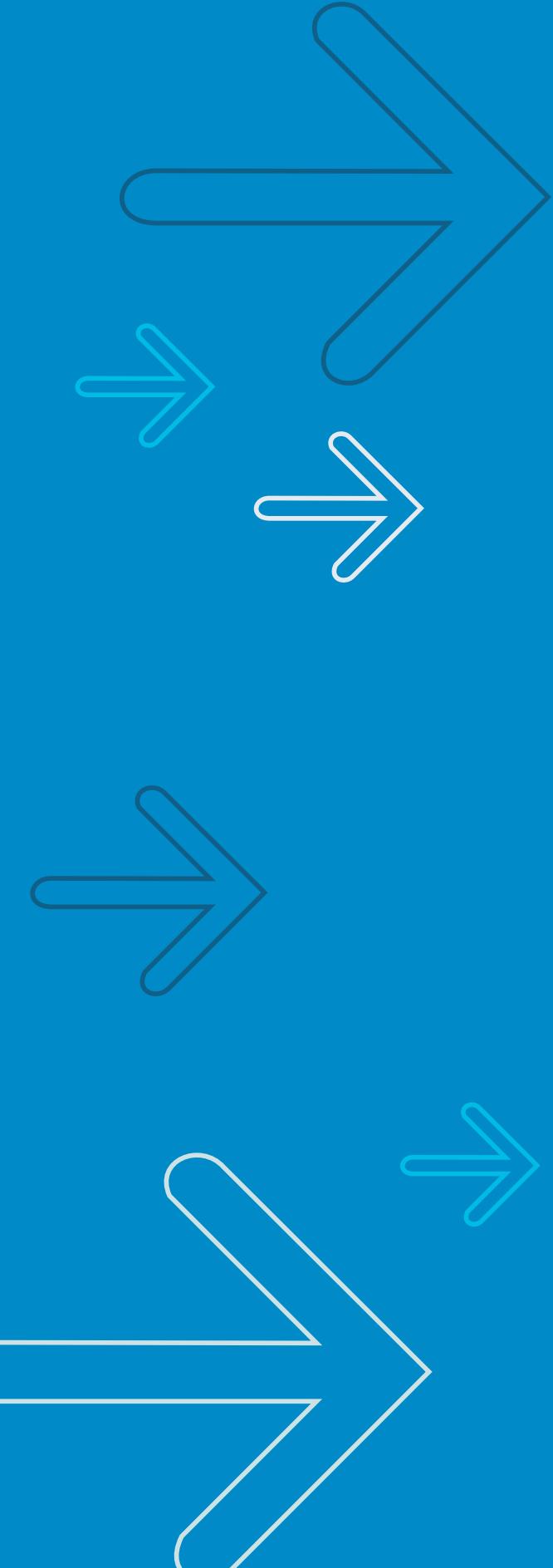
Portanto, sempre que criar um banco de dados, planeje, planeje, planeje e somente depois execute!



Há vários passos na confecção de um projeto de banco de dados:

- coletar os requisitos e as regras de negócio;
- confeccionar o modelo conceitual;
- confeccionar o modelo lógico;
- confeccionar o modelo físico;
- gerar os *scripts* para implementação do banco.

Cada um desses passos serão detalhados a seguir, entre as unidades 8 e 13, sendo as unidades 11 a 13 citações de exemplos com base em plataformas.



# Unidade de estudo 7

## Seções de estudo

Seção 1– Conceitos

Seção 2– Como levantar os requisitos e as  
regras de negócios

# Coleta de Requisitos e Regras de Negócio

## SEÇÃO I

### Conceitos

A coleta de requisitos é uma busca prévia das funções que o sistema executará. Se o sistema deve possuir a opção para cadastro de cliente, então este deve ser considerado um requisito, assim como o sistema pode executar uma venda mesmo não havendo o produto no estoque ou oferecer desconto especial a um determinado grupo de clientes.

Já as regras de negócios representam uma busca prévia das informações necessárias para determinar como o sistema funcionará. Por exemplo: a tabela de produtos deve ser atualizada periodicamente? Ou: o envio das notas fiscais deve ser feito automaticamente ou manualmente? Juntos, os requisitos e as regras de negócio demonstram as características do aplicativo, suas funcionalidades e infraestrutura.

Muitas pessoas se confundem e coletam todos os requisitos de acordo com o que o usuário pediu. Isso não pode ser considerado uma boa prática, já que quase sempre o cliente ou usuário não possui o entendimento suficiente de banco de dados ou de sistema a ponto de apontar características técnicas do ambiente.

Esse entendimento é muito importante para mapear corretamente os requisitos, permitindo a criação eficaz do modelo conceitual. O usuário é responsável por entregar todos os requisitos e regras do negócio, mas não por definir como regras e requisitos serão internalizadas no sistema.

Essas regras e requisitos foram claramente informados pelo cliente ou usuário do novo sistema, pois não seria possível que você, técnico em informática, pudesse prever com exatidão essas premissas. Porém, a partir daí, a construção do modelo conceitual por meio da técnica do MER é com você! Você é responsável por criar o melhor modelo de banco de dados e prever a melhor forma para armazenar os dados do sistema e relacionar as entidades. Apenas demonstre os resultados dos relacionamentos falando a linguagem do negócio ao seu cliente ou usuário.



## SEÇÃO 2

### Como levantar os requisitos e as regras de negócios

É necessário entender e mapear tudo o que o cliente ou usuário necessita. Segundo Carvalho (no prelo), grande parte dos erros nos softwares são resultados do não entendimento dos requisitos e das regras de negócios informados pelo cliente.

Acompanhe um exemplo de requisito e de regra de negócio:

- Regra de Negócio 1: a verificação de estoque deve ser feita a cada novo pedido.
- Requisito Funcional 5: o sistema deverá permitir desconto por produto para cada cliente.

Lembre-se de que um levantamento falho de requisitos poderá prejudicar todo o restante do projeto, como a modelagem conceitual, lógica e física e, consequentemente, a implementação do banco de dados.

Na próxima unidade, você conhecerá a modelagem conceitual baseada em MER. Sempre que tiver alguma dúvida, consulte o seu professor em sala de aula.



# **Unidade de estudo 8**

## **Seções de estudo**

- Seção 1 – Definindo as entidades**
- Seção 2 – Definindo os relacionamentos**
- Seção 3 – Definindo as cardinalidades**
- Seção 4 – Contextualização**

# Modelagem Conceitual Baseada no MER

Você já tem bastante informação sobre modelagem conceitual e MER. Agora chegou a hora de juntar as informações e aprender a construir um modelo conceitual. Nesta unidade, você acompanhará os passos para a criação de um modelo conceitual de banco de dados para a empresa X.

Com base no que você aprendeu nos capítulos anteriores deste livro sobre modelagem conceitual e MER, será explanado, nesta unidade, como construir o modelo conceitual para o banco de dados de uma empresa X. Todos os passos aqui apresentados deverão ser realizados com base no aprendizado das unidades 4, 5 e 6.

## SEÇÃO 1

### Definindo as entidades

A definição das entidades é o primeiro passo para a construção de um modelo conceitual baseado em entidade-relacionamento. Defina todas as entidades para que seja possível iniciar o modelo de entidade-relacionamento. Acompanhe o exemplo de um sistema de uma simples padaria, onde as entidades que podem ser definidas são as seguintes:

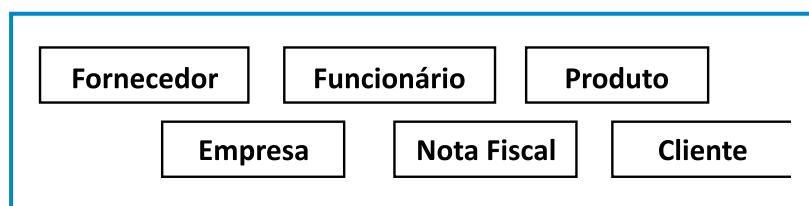


Figura 28: Entidades definidas

## SEÇÃO 2

### Definindo os relacionamentos

Essas entidades “soltas”, como estão, nada representam em um MER, portanto, é preciso interligá-las utilizando os relacionamentos. Visualizando essas entidades, pode-se definir o seguinte:

- Nota fiscal se relaciona com a empresa (a própria padaria), pois em uma nota fiscal há o nome da empresa que vendeu o produto.
- Nota fiscal se relaciona com o produto, pois em uma nota fiscal sempre há a descrição dos produtos comprados.
- A empresa se relaciona com seus clientes e também com seus funcionários, pois, para ter funcionário, é preciso ter uma empresa, e o mesmo acontece para o cliente, quando também é necessária a existência de uma empresa.

- A empresa se relaciona com seus produtos, pois, para existirem, é necessário que a padaria os compre.
- Se a empresa compra o produto, o compra de um fornecedor, representado por uma entidade. Você já deve ter previsto, então, que o produto também se relaciona com o fornecedor, certo? Diante dessas afirmações, você consegue descrever com que outras entidades a nota fiscal se relaciona? Pense um pouco no trâmite do negócio na vida real e tente imaginar os relacionamentos. Que dados ou informações são necessárias para gerar a nota fiscal que é emitida para o cliente? Facilmente podemos afirmar que os dados dos produtos, da padaria e do cliente são necessários para gerar essa nota.

Mas e a nota fiscal da compra dos produtos? Podemos afirmar que as informações pertinentes para a nota de entrada seriam as informações do fornecedor, da padaria e, logicamente, do produto. Dessa forma:

- A nota fiscal se relaciona com o fornecedor, a padaria (empresa), o cliente e os produtos. Deve-se, preferencialmente, identificar a entidade que mais possui relacionamentos e colocá-la no meio do diagrama, pois o conjunto de relacionamentos pode atrapalhar no entendimento do MER. Então, qual é a entidade que mais possui relacionamentos? Nesse caso, são três as entidades que

mais possuem relacionamentos: a Nota fiscal, a Empresa e o Produto. Elas possuem o maior número de relacionamentos (4) e uma delas, no caso é definida pela entidade Nota Fiscal, fica no meio do MER, pois a probabilidade de ela ter mais relacionamentos posteriormente é grande. Após todas essas afirmações, já é possível construir o MER com os relacionamentos. Veja na figura a seguir como ficou o MER com os relacionamentos:

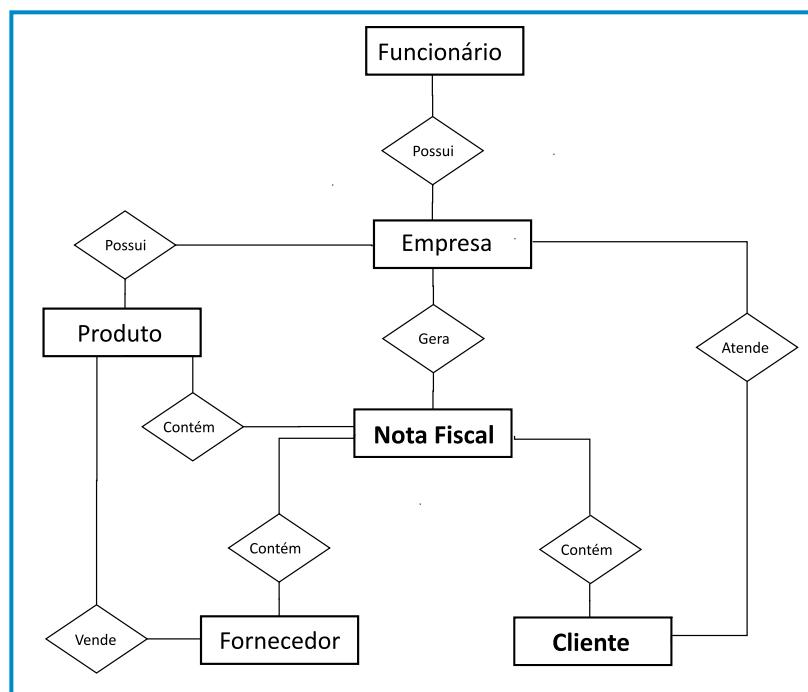


Figura 29: MER simulando um banco de dados de uma padaria

Está criando forma de diagrama, não é mesmo? Porém, ele ainda não está finalizado! Falta utilizar os conceitos de cardinalidade para informar as ocorrências existentes entre as entidades.

## SEÇÃO 3

### Definindo as cardinalidades

No primeiro relacionamento, você pode ver que a Empresa se relaciona com a Nota fiscal. Quantas ocorrências você acha que há nesse ponto? Pense que a padaria pode emitir várias notas fiscais, porém cada nota fiscal tem apenas um emissor (a padaria), portanto o relacionamento entre empresa e nota fiscal é do tipo um para muitos (possibilidade de existir apenas uma empresa em cada nota fiscal e muitas notas fiscais para cada empresa).

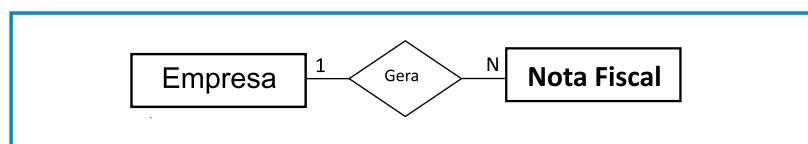


Figura 30: Relacionamento entre Empresa e Nota fiscal

Pensando agora no relacionamento Nota fiscal e Produto: quantos produtos podem ser inseridos em uma Nota fiscal (apenas um ou vários?). É isso mesmo! Vários produtos podem ser inseridos em uma Nota fiscal. O mesmo vale para a leitura contrária, ou seja, esses mesmos produtos também podem estar em outras Notas fiscais, afinal, você pode vender um suco e um salgado para vários clientes, certo? Então esse relacionamento é de N para N ou muitos para muitos.

## SEÇÃO 4

### Contextualização

Esse é o início efetivo do projeto de um banco de dados, demonstrando as entidades e os relacionamentos básicos de um negócio.

Perceba que esse conjunto básico de entidade e relacionamento pode ser aplicado não somente em uma padaria, mas em quaisquer empresas que possuam requisitos e regras de negócios similares, como uma loja de autopeças ou outro comércio com as mesmas características de relacionamentos, por exemplo.

Vale lembrar que este é apenas um núcleo do modelo conceitual. A partir daí, várias outras entidades surgirão e caracterizarão um negócio em específico, por exemplo, as entidades de acervo e status de filme para uma videolocadora ou de descrição dos serviços para uma oficina mecânica.

Você já aprendeu a construir um modelo conceitual e vai ver que agora ficará mais fácil construir um modelo lógico. Acompanhe, na próxima unidade, os passos para a modelagem lógica baseada no DER conceitual.

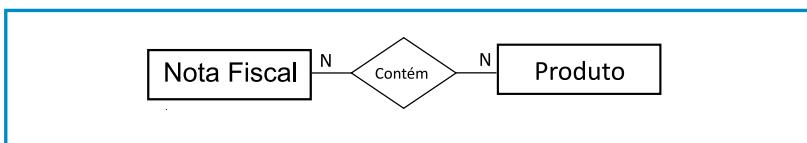


Figura 31: Lê-se: uma nota fiscal contém vários produtos, assim como um produto pode estar contido em mais de uma nota fiscal

Agora está fácil. Imagine o restante dos relacionamentos para construir as cardinalidades.

- A Empresa se relaciona com seus Clientes de forma um para N (nesse caso, não importa se os clientes comprarão em outras padarias, por isso, está indicado que o cliente é somente dessa padaria).
- A Empresa se relaciona com seus Produtos de forma um para N (nesse caso, não importa se esses produtos serão vendidos pelo fornecedor para outras padarias, por isso está indicado que o produto somente está nessa padaria).
- A entidade Fornecedor se relaciona com Produto de forma N para N, pois um produto pode ser comprado de mais de um fornecedor e esses fornecedores podem fornecer mais de um produto.
- A Empresa se relaciona com seus Funcionários de forma um1 para N.

Veja agora como ficou o diagrama de entidade-relacionamento (DER) conceitual referente ao banco de dados de uma padaria com os relacionamentos e suas respectivas cardinalidades:

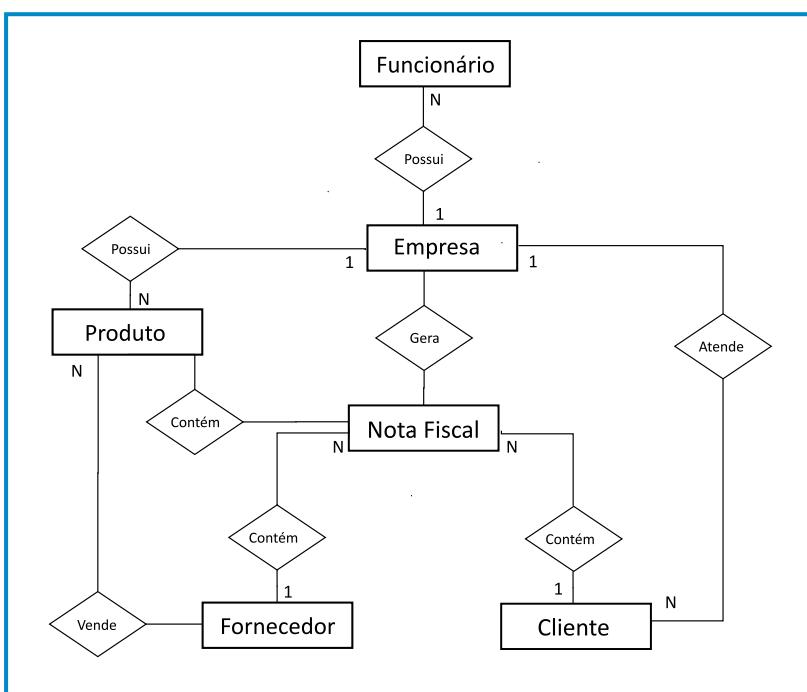
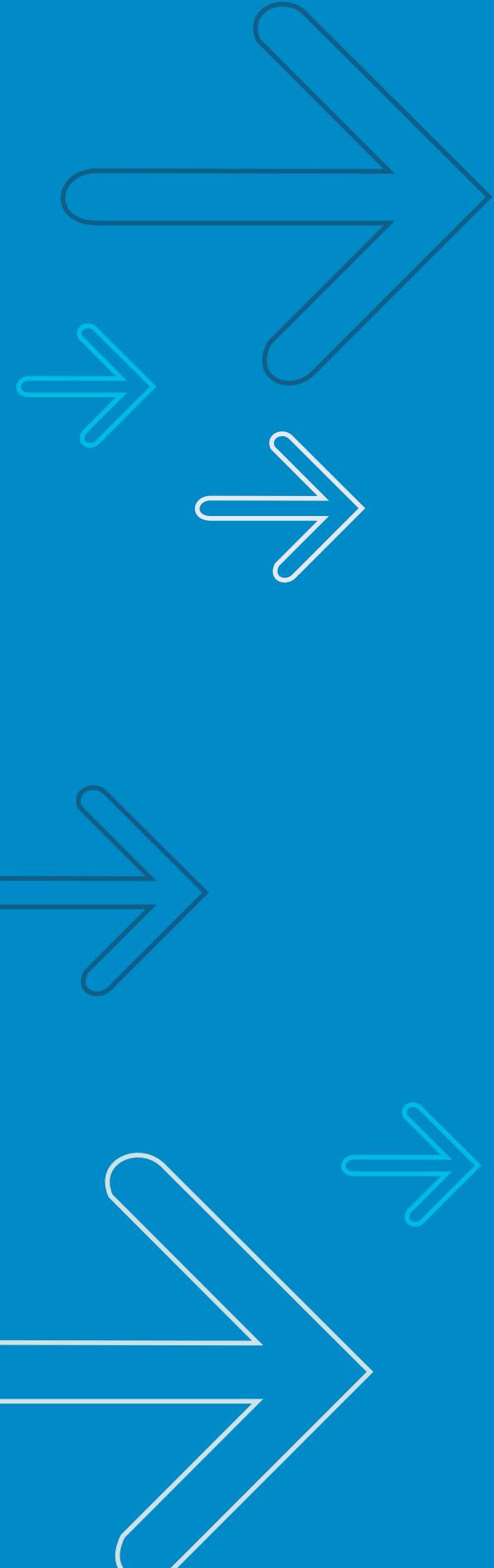


Figura 32: Exemplo de um DER Conceitual de uma padaria



# **Unidade de estudo 9**

## **Seções de estudo**

- Seção 1 – Construindo um modelo lógico**
- Seção 2 – Mapeamento das entidades, dos relacionamentos, da cardinalidade e dos atributos**
- Seção 3 – Atributos de chave primária e chave estrangeira**

# Modelagem Lógica Baseada no DER Conceitual

## SEÇÃO I

### Construindo um modelo lógico

O modelo lógico exige a definição de uma abordagem de dados, e, neste livro, a abordagem utilizada será a abordagem relacional de dados.

A migração de um modelo conceitual para um modelo lógico relacional é feita por meio do repasse das visões e dos conceitos pelo desenvolvedor. Todas as entidades identificadas no modelo DER conceitual passarão a ser chamadas de tabelas no DER lógico, utilizando a abordagem relacional de dados.

## SEÇÃO 2

### Mapeamento das entidades, dos relacionamentos, das cardinalidades e dos atributos

Seguindo no mesmo projeto de confecção de um banco de dados para uma padaria, traga para cá todas as entidades do DER e altere o formato dessas entidades para representar uma tabela, por exemplo, as entidades Produto e Nota fiscal:

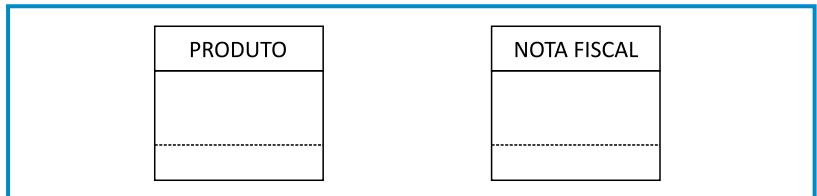


Figura 33: Representação lógica da entidade Produto e da entidade Nota fiscal

Nessa linha, você deverá transformar todas as entidades em tabelas e fazer o mapeamento de relacionamentos, cardinalidade e atributos. Veja esses passos a seguir.

Executar os relacionamentos entre as tabelas requer atenção, pois eles são primordiais para o bom funcionamento do banco de dados.

Você se lembra dos relacionamentos aplicados anteriormente, no modelo conceitual? Agora você fará o mesmo procedimento no modelo lógico, aplicando os relacionamentos entre as tabelas, com as devidas cardinalidades.

Neste modelo, você pode abolir a utilização do losango com os verbos para representar as ações, afinal, você já consegue supor qual é a ação executada entre os relacionamentos apenas visualizando as entidades relacionadas.

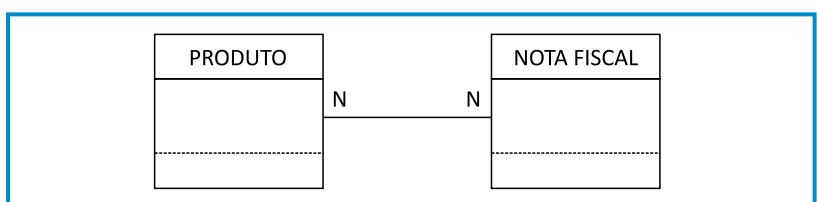


Figura 34: Tabelas relacionadas com suas devidas cardinalidades

Feitos os relacionamentos, é preciso preencher a tabela, afinal uma tabela vazia nada representa em um banco de dados. Preencha os **campos** das tabelas definindo os atributos para cada uma. Para fazer isso, verifique as características comuns dentro de cada entidade, indagando-se: “O que descreve essa entidade?”. Seguindo o exemplo da unidade 5, têm-se para a entidade Cliente os seguintes atributos (ou campos):

- número do CPF;
- nome do cliente;
- endereço;
- data de nascimento.

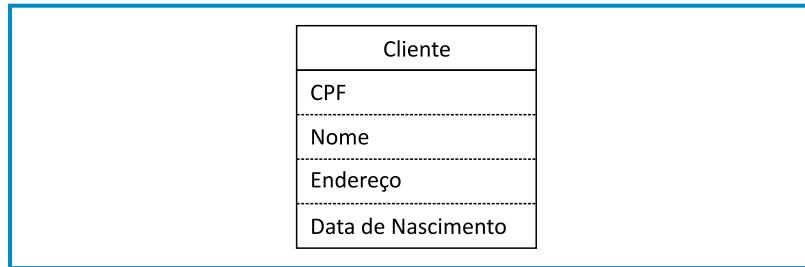


Figura 35: Entidade Cliente com seus atributos, ou Tabela Cliente com seus Campos

Definidos os campos para todas as tabelas, deve-se, então, relacioná-las entre si conforme a necessidade, identificando para cada relacionamento a sua cardinalidade. Veja como está ficando o nosso exemplo:

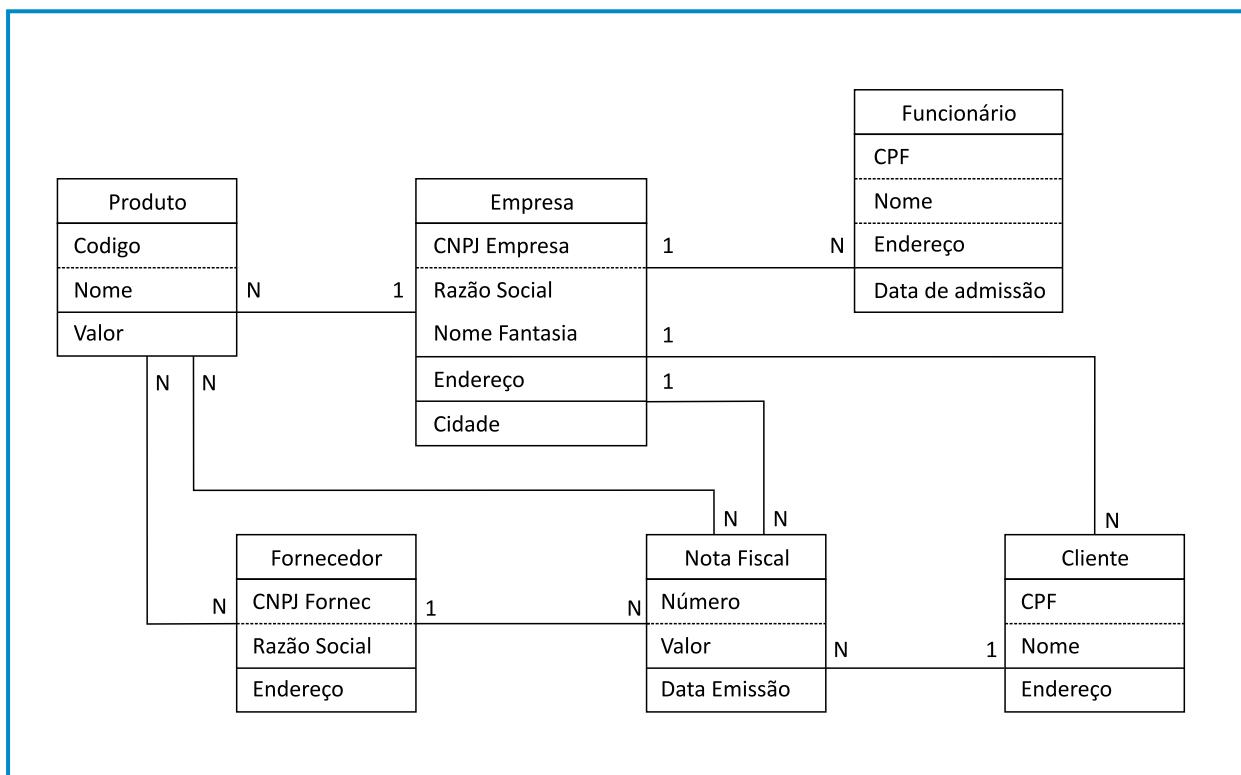


Figura 36: DER lógico com suas entidades, atributos e relacionamentos

## SEÇÃO 3

### Atributos de chave primária

Chegou a hora de definir os identificadores das tabelas.

Os atributos de chave primária são aqueles que nunca se repetem dentro de uma entidade e têm a função de atuar como identificadores de cada **instância** da entidade, ou seja, é o atributo ou conjunto de atributos concatenados que identifica uma única ocorrência dentro da tabela (ABREU; MACHADO, 2004).

→ **Instância** várias explicações técnicas e complexas poderiam ser dadas para o termo instância, porém, para facilitar o aprendizado, considera-se que instância significa o conjunto de valores inseridos nos campos de cada tabela em um banco de dados.

**Se o valor de uma chave primária identifica uma ocorrência dentro de uma entidade, por exemplo, um cliente dentro da entidade Cliente, esse valor nunca pode ser nulo,** em outras palavras, ao cadastrar um cliente no banco de dados, o atributo definido como chave primária nunca pode ser deixado em branco, pois, nesse caso, o cliente não poderia ser identificado.

Nesse mesmo exemplo da entidade Cliente, com os quatro atributos já definidos a ela, é fácil identificar qual atributo poderia ser a chave primária. Pense um pouco: um atributo que não pode se repetir para os demais clientes é o CPF, certo? Então, como o CPF não pode ser cadastrado para mais de um cliente, ele é o atributo ideal para ser a chave primária da tabela cliente.

Para representar uma chave primária dentro de uma tabela, será utilizado o símbolo PK, do inglês: *Primary Key*. Alguns livros simbolizam a chave primária com o desenho de uma chave, e outros com o cifrão (\$). Definições à parte, o importante é que a chave primária esteja sempre identificada.

Cliente
PK CPF
Nome
Endereço
Data de Nascimento

Figura 37: Identificação da chave primária

Adicionalmente, vale reforçar que mais de uma chave primária pode ser definida, como por exemplo, o CPF e o RG. Se esses dois campos forem definidos como chave primária, o SGBD considerará a junção dos dois campos como a chave primária da tabela, podendo um desses campos se repetir ao longo dos registros, mas nunca os dois campos serão iguais ao mesmo tempo. Adicionalmente, vale reforçar que mais de uma chave primária pode ser definida, como por exemplo, o CPF e o RG. Se esses dois campos forem definidos como chave primária, o SGBD considerará a junção dos dois campos como a chave primária da tabela, podendo um desses campos se repetir ao longo dos registros, mas nunca os dois campos serão iguais ao mesmo tempo.

O analista pode definir a chave primária ao final de toda a modelagem, deixando marcados os campos que pretende avaliar posteriormente. Esses campos são chamados de chaves candidatas e permanecem até que sua definição seja feita (chave primária ou atributo comum).

## SEÇÃO 4

### Atributos de chave estrangeira

Os atributos de chave estrangeira especificam as relações entre as entidades. Também conhecidas como chave de relação, elas especificam as relações que cada entidade possui e apontam sempre para o atributo de chave primária de outra tabela.

Segundo Abreu e Machado (2004, p. 62), chave estrangeira é “[...] um dado colocado em uma entidade que em outra é o identificador único (chave primária).”

A partir da criação de uma chave estrangeira, também conhecida como FK (do inglês *Foreign Key*), passa a existir um relacionamento entre a tabela em que ela está inserida e a tabela onde está a chave primária para qual ela está apontando.

Nesse relacionamento, a chave estrangeira vai sempre do lado de cardinalidade N (muitos).

Continuando o exemplo da padaria, em que a entidade Empresa se relaciona com a entidade Funcionário de um para N, tem-se uma chave primária para a Empresa, que pode ser o seu CNPJ, e uma chave primária para Funcionário, o seu CPF. Para efetivar esse relacionamento, deve-se criar um campo de chave estrangeira na tabela de funcionários que será o mesmo campo de chave primária da tabela Empresa, ou seja, o campo: “Número do CPF”.

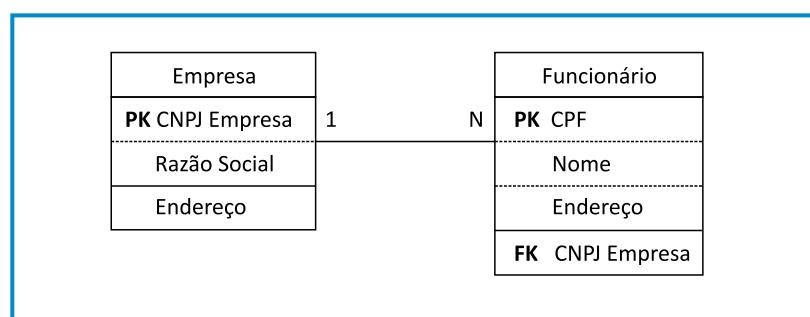


Figura 38: Definição da FK na tabela Funcionário

Agora está fácil finalizar o DER lógico! Insira todos os atributos nas tabelas que já estão relacionadas, lembrando sempre das chaves primárias e estrangeiras, e o fato de os relacionamentos N para N gerarem uma tabela auxiliar que se relaciona com as duas tabelas principais de N para um.

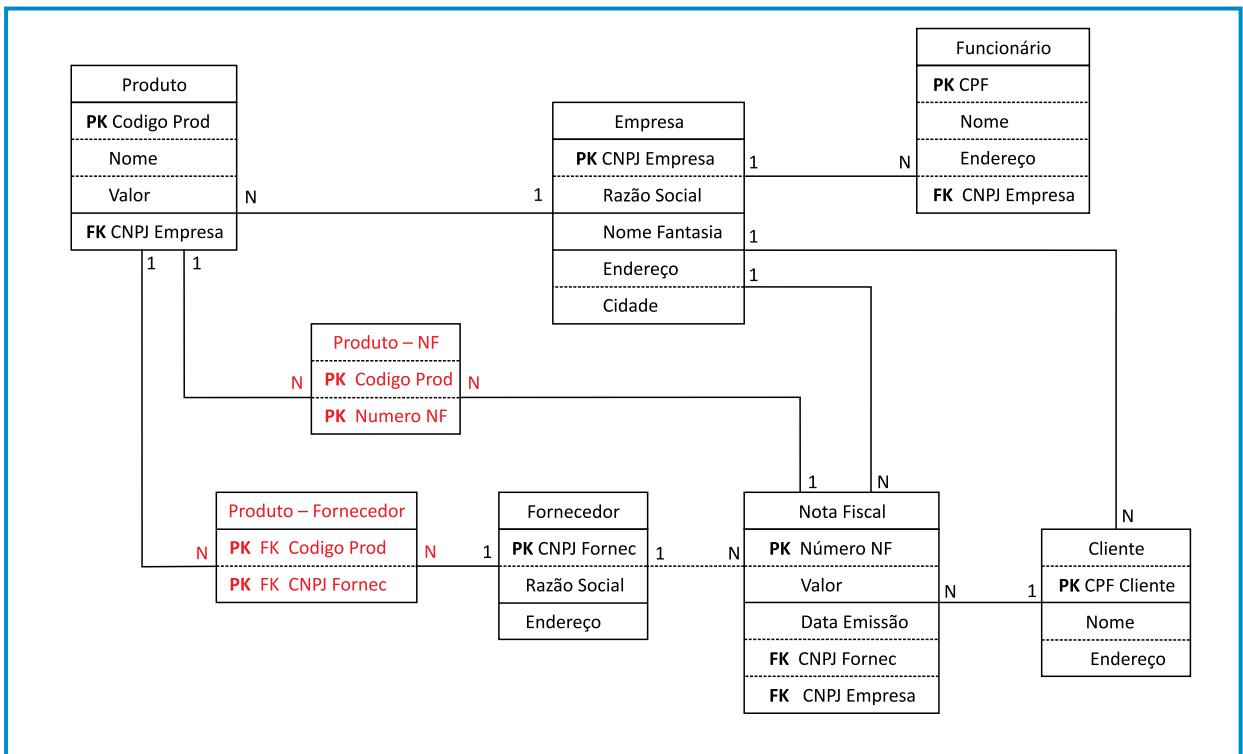
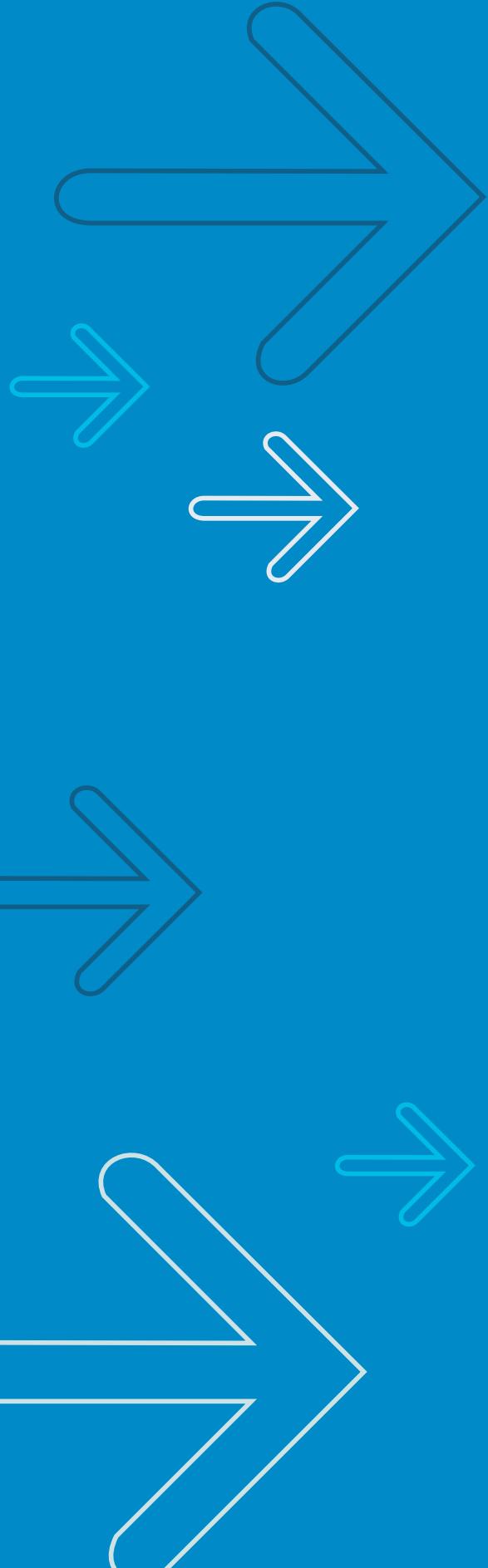


Figura 39: DER lógico completo

Agora que você já sabe construir um modelo conceitual e um modelo lógico, aprenda, na próxima unidade, como construir um modelo físico.



# **Unidade de estudo 10**

## **Seções de estudo**

- Seção 1 – Construindo um modelo físico**
- Seção 2 – Migração do DER lógico relacional para o modelo físico**
- Seção 3 – Estudo das tabelas**
- Seção 4 – Definição de padrões nas nomenclaturas dos campos**
- Seção 5 – Estudo e caracterização dos atributos**
- Seção 6 – Sugestões para a confecção**

# Modelagem Física Baseada no DER Lógico

## SEÇÃO 1

### Construindo um modelo físico

Essa etapa leva em consideração as características do SGBD ao qual será aplicado o modelo e criado o banco de dados. O desenvolvedor, juntamente com o DBA, implementa esse modelo e se preocupa, nessa etapa, com o tamanho dos campos de cada tabela, as formas de buscas de informações e o tipo dos registros que serão inseridos em cada campo, ou seja, qual o formato do registro, se são números, datas, textos ou outros.

A migração do modelo lógico relacional para o modelo físico representa a reta final no planejamento do projeto de construção de um banco de dados, e é possível definir algumas regras para que essa migração seja bem-sucedida.

## SEÇÃO 2

### Estudo das tabelas

Nesse ponto, o SGBD utilizado já é conhecido, portanto, devem-se estudar as tabelas a fim de verificar se elas estão aptas a serem utilizadas no SGBD escolhido. Por exemplo, dependendo do SGBD, uma tabela definida no modelo lógico pode gerar mais de uma tabela no modelo físico, ou até mesmo desaparecer, caso a inserção de um campo em outra tabela abstraia a necessidade de criação desta.

Por isso, para facilitar a pesquisa, criação e alteração de campos das tabelas, é importante definir padrões de nomeclatura para os campos constantes na tabela. Uma tabela funcionário pode ter, por exemplo, os seguintes campos:

- CPF do funcionário;
- nome completo;
- data de nascimento;
- grau de instrução.

Como campos de tabela, esses nomes não são adequados e podem dificultar uma futura **query** e até mesmo a confecção de relacionamentos entre tabelas, já que as chaves estrangeiras apontam para as chaves primárias de outras tabelas, buscando-as pelos nomes. Portanto, pode-se definir um padrão para cada tipo de campo. Aqui será dada a seguinte sugestão:

- Campos com números iniciam-se com **nr\_XXX**. Exemplo: **nr\_cpf**.
- Campos que definem nomes iniciam-se com **nm\_XXX**. Exemplo **nm\_cliente**.
- Campos que definem descrições de algo iniciam-se com **ds\_XXX**. Exemplo: **ds\_endereco** ou **ds\_grau\_de\_instrucao**.

➤ **Query** termo muito utilizado entre os tipos de usuários de banco de dados, que representa simplesmente uma consulta ao banco.

- Campos de datas iniciam-se com **dt\_XXX**. Exemplo: dt\_nascimento ou dt\_pedido.
- Mais que uma sugestão, uma obrigatoriedade, **campos com acentos não são aceitos**, dessa forma, nome do funcionário é alterado para: nm\_funcionario. Seguindo esses padrões, a tabela de clientes fica:

Cliente
<b>PK</b> nr_cpf
nm_cliente
ds_endereco
dt_nascimento

Figura 40: Tabela cliente com os campos no padrão de nomenclatura

## SEÇÃO 3

### Estudo e caracterização dos atributos

Após a definição das tabelas e da nomenclatura dos campos, é preciso estabelecer os tipos dos registros que serão inseridos nos

campos, se são do tipo numérico, data, texto ou outros. Dependendo do tipo do SGBD, pode haver diferenças na nomenclatura dos campos, por isso utilizaremos aqui os tipos aceitos pelo MySQL (banco de dados que será abordado na unidade 11 deste livro).

Há vários tipos de campos de dados, como:

- INT e INTEGER quando os campos forem receber registros em forma de números inteiros;
  - TEXT quando os registros forem ser do tipo texto;
  - CHAR e VARCHAR quando os registros forem ser do tipo números, textos e/ou caracteres especiais;
  - DATE, DATETIME e SMALLDATETIME quando os registros têm a forma de data. Eles são inseridos ao lado do nome do campo, separados pelo caractere: (dois pontos).
- Exemplo:** nm\_cliente:TEXT

Entre os tipos de dados citados, para os tipos INT, CHAR, VARCHAR e TEXT, é possível definir o tamanho máximo do registro que será inserido no respectivo campo inserindo o número de ca-

racteres máximo entre parênteses ao lado da descrição do tipo.

Dessa forma, para a mesma tabela Cliente, os tipos dos campos são:

- nr\_cpf: CHAR(11);
- nm\_cliente: TEXT(50);
- ds\_endereco: VARCHAR(70);
- dt\_nascimento: SMALLDATETIME;

Assim, a tabela de clientes no modelo físico fica:

Cliente
<b>PK</b> nr_cpf: CHAR(11)
nm_cliente: TEXT(50)
ds_endereco: VARCHAR(70)
dt_nascimento: SMALLDATETIME

Figura 41: Tabela cliente no modelo físico

Pronto! Depois de todo o aprendizado até então, já é possível construir o diagrama físico relacional que dará origem, enfim, ao banco de dados de um sistema – nesse caso, o sistema de uma padaria. Transformando o DER lógico relacional para físico, ele ficará como na figura seguinte.

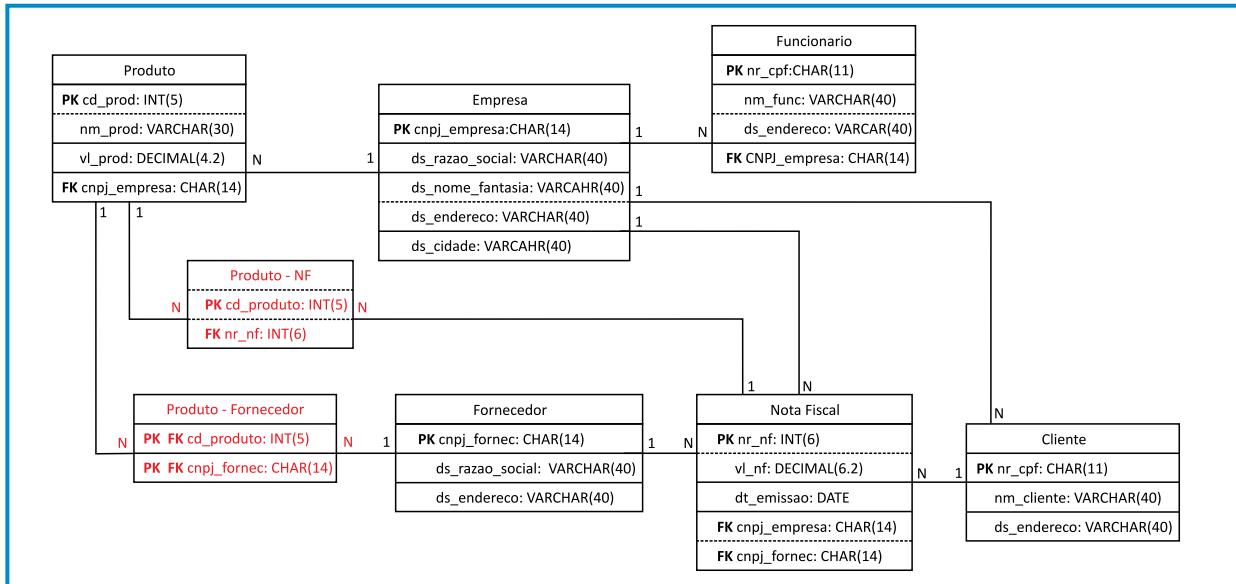


Figura 42: Diagrama físico relacional

## ➤ A importância dos tipos de campos

A definição dos tipos dos campos é muito importante para o correto armazenamento dos dados e para a *performance* nas consultas. Por exemplo, os tipos de campos CHAR e VARCHAR armazenam os mesmos tipos de dados e poderiam ser utilizados no campo ds\_endereco. Por que, então, foi preferível a utilização do campo VARCHAR? Por um simples motivo: o tipo de campo VARCHAR armazena apenas os caracteres que o usuário inserir no campo.

**Exemplo:** Nesse campo, há espaço para um registro de 70 caracteres, mas se o usuário inserir apenas 20, sendo do tipo VARCHAR, o banco reservará espaço para apenas 20 caracteres, ao contrário do tipo CHAR, que, apesar de trazer mais rapidez nas consultas, reservaria espaço para os 70 caracteres, mesmo que os outros caracteres não fossem utilizados.

Então, pensando dessa forma, por que, para o campo nr\_cpf, utilizou-se o tipo CHAR em vez do VARCHAR? A resposta está no tamanho do campo. Um CPF sempre tem 11 dígitos e, se algum tiver menos que 11 dígitos, é completado com zeros à esquerda (038.479...). Dessa forma, não há variação do tamanho dos dados, então, a utilização do CHAR é adequada, pois não sobrecregaria o tamanho do banco de dados e traria maior rapidez nas consultas das tabelas.

Imaginando que, como o nr\_cpf é a chave primária da tabela cliente, receberá muitas consultas, então, é vantajosa a utilização de um tipo de dados de tamanho fixo, o qual agilizará as pesquisas nessa tabela. Para traduzir isso tudo para o seu real cotidiano, lembre-se da última vez que você ligou para uma operadora de celular, de cartão de crédito ou até mesmo para instituições financeiras. Qual é a primeira informação solicitada? Isso mesmo, o CPF, o que reforça o fato de considerar o CPF a chave primária da tabela clientes e definir o valor como CHAR, agilizando as consultas por meio desse campo.

### SAIBA MAIS

É importante que você faça uma pesquisa ao final desta unidade sobre os tipos de campos de alguns bancos de dados. Você pode utilizar os sites <[www.microsoft.com](http://www.microsoft.com)> e <<http://dev.mysql.com>> para se aprofundar no assunto.

Lembre-se: o alinhamento com o DBA anteriormente à criação do banco de dados é muito importante, pois é ele que vai administrá-lo e prever melhorias pós-implementação.

Bom, até aqui, você já aprendeu a confeccionar os modelos conceituais, lógico e físico, completando todo o planejamento de um banco de dados, porém falta criá-lo efetivamente. Já tem ideia de como isso pode ser feito? Não se preocupe, é esse o assunto que será abordado a partir de agora.

Há duas formas de criar o banco de dados a partir do modelo físico relacional de dados:

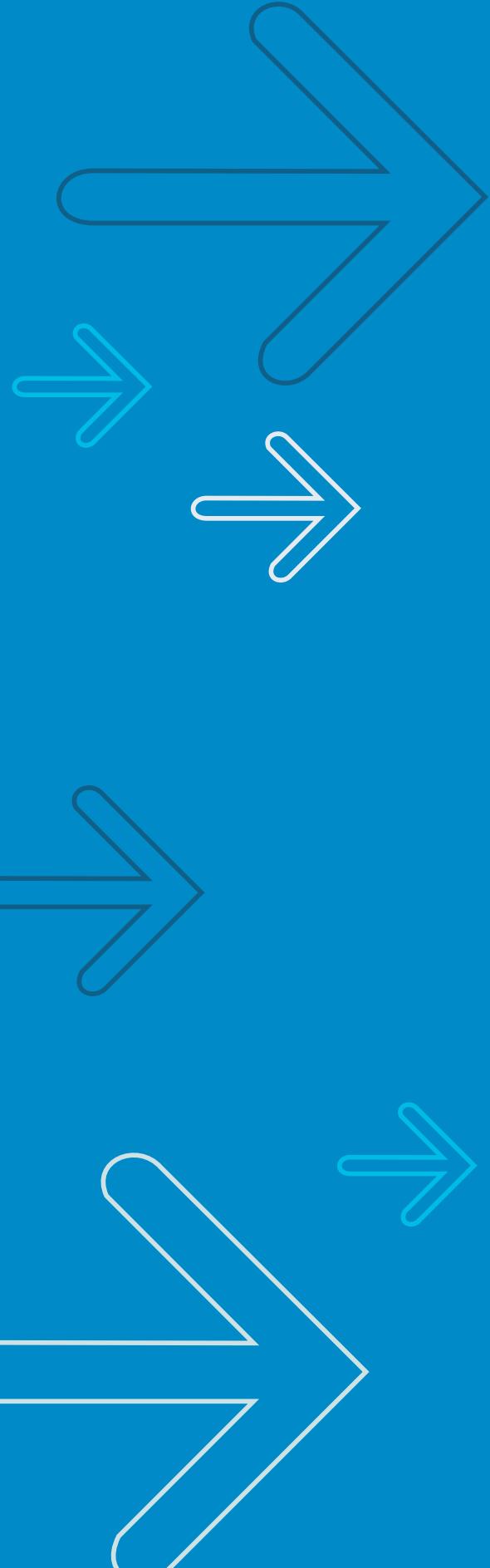
- Pela criação do modelo físico em uma ferramenta *case* que permita a migração das tabelas e relacionamentos diretamente para o banco de dados; ou
- Pela criação manual de todas as tabelas e relacionamentos via comandos SQL.

Para começar esse assunto, você conhecerá, na próxima unidade, um pouco da ferramenta de banco de dados MySQL, e nas unidades seguintes, acompanhará um detalhamento dessas duas formas de criar banco de dados. Siga em frente!

## SEÇÃO 4

### Sugestões para uma boa confecção do DER físico

Um formato interessante que visa a facilitar a confecção do DER físico é executar essa migração do modelo lógico para o físico em conjunto com o DBA, onde o desenvolvedor apresenta o modelo lógico ao administrador sugerindo os formatos e tamanhos dos campos e este verifica o diagrama *versus* suas consequências em termos de armazenamento de dados, administração e *performance* do banco.



# Unidade de estudo 11

## Seções de estudo

- Seção 1 – O que é MySQL
- Seção 2 – O aplicativo EasYPHP
- Seção 3 – O funcionamento do MySQL
- Seção 4 – Pormenores

# Banco de Dados MySQL

## SEÇÃO 1

### O que é MySQL

Segundo MySQL (2010a), MySQL é um SGBD com larga utilização mundial, que utiliza a linguagem SQL como interface para comunicação. É considerado o banco de dados mais popular no mundo, possuindo mais de 10 milhões de instalações atualmente, entre versões gratuitas e pagas.



Figura 43: Logotipo MySQL

Por ter uma interface prática com relação a esse conteúdo, por possuir uma versão gratuita, proporcionar uma boa compatibilidade com sistemas *web* e possibilitar ótima integração com a linguagem PHP (*Hypertext Processor*) é que o MySQL foi escolhido para estudarmos.

#### SAIBA MAIS

Atualmente, o MySQL encontra-se na versão 5.1 e pode ser disponível para *download* no endereço: <[www.mysql.com](http://www.mysql.com)>.

## SEÇÃO 2

### O aplicativo EasyPHP

Você pode fazer o *download* do MySQL, instalá-lo e configurá-lo diretamente ou fazer o *download* do aplicativo EasyPHP que, além do MySQL, traz consigo o aplicativo PHP, o *web server* Apache e o gerenciador da base de dados, o PhpMyAdmin, que poderão ser utilizados futuramente em um projeto de um novo *software* integrado com o MySQL.

Segundo EasyPHP (2010), o EasyPHP é um pacote móvel completo que traz os principais serviços para o desenvolvimento de um sistema *web*, podendo ser instalado no computador ou em uma mídia removível qualquer, possibilitando o desenvolvimento dos sistemas necessários de qualquer computador.



Figura 44: EasyPHP

➤ **Script** comando executado dentro de um programa para executar determinadas ações.

A vantagem da instalação do EasyPHP sobre a instalação unicamente do MySQL é que, além de vir com os aplicativos mencionados nesta seção, o EasyPHP não exige maiores configurações durante e após a instalação, sendo facilmente instalado e configurado em um tempo menor, exigindo inclusive mínimos conhecimentos técnicos para tal.

#### SAIBA MAIS

O EasyPHP encontra-se atualmente na versão 5.3 e está disponível no endereço: <[www.easypht.org](http://www.easypht.org)>.

## SEÇÃO 3

### O funcionamento do MySQL

O MySQL, independentemente do modo como é instalado, cria estruturas dos bancos e manipula os dados via comandos de linguagem SQL, podendo estes serem gerados via programas de apoio como o MySQL Workbench, que será visto na unidade 12, phpMyAdmin ou por meio de códigos SQL gerados manualmente, opção que será vista na unidade 13 e é a mais utilizada por desenvolvedores experientes em empresas de grande porte.

Lembre-se de que essas formas de execução de *scripts* de linguagem SQL são apenas meios, mas apontam para um mesmo fim, que é gerenciar as informações que estão dentro do banco de dados.

## SEÇÃO 4

### Pormenores

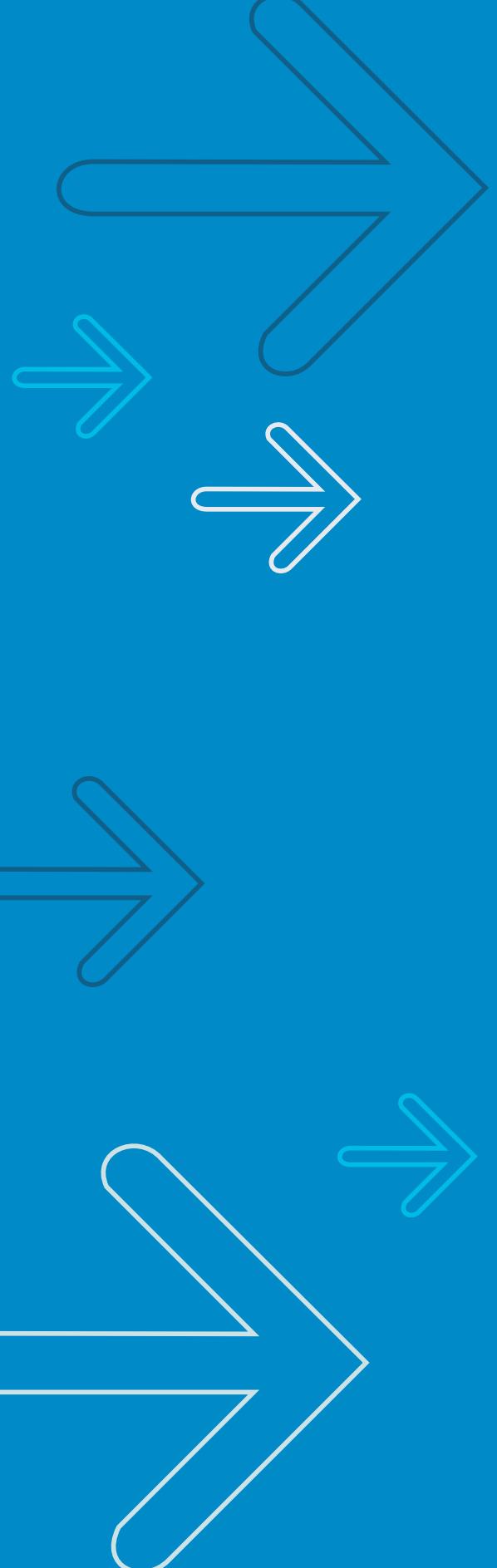
Silva (2001) afirma que, se há algum ponto fraco no MySQL, este poderia ser atribuído ao fato de a interface para o usuário ser executada por meio de linhas de comando que podem desalentar os usuários ou desenvolvedores que estão iniciando no tema e acostumados com as interfaces gráficas do Microsoft Windows.

Porém, essa deficiência é compensada com a instalação de um aplicativo do tipo GUI (*Graphics Users Interface*) para Windows. Se você instalar o EasyPHP descrito na seção 2 desta unidade, você automaticamente terá instalado também uma ferramenta de interface gráfica, que é o PhpMyAdmin. Silva (2001) complementa ainda que o acesso e a manipulação dos dados pela interface gráfica são bem mais versáteis que a interação com o DOS.

#### SAIBA MAIS

Outras ferramentas GUI podem ser encontradas no site do fabricante do SGBD pelo endereço: <[www.mysql.com](http://www.mysql.com)>.

Você já sabe criar modelos de diagramas de dados. Na próxima unidade, você começará a parte prática. Prepare-se!



# Unidade de estudo 12

## Seções de estudo

- Seção 1 – Utilidade de uma ferramenta *case*
- Seção 2 – MySQL Workbench
- Seção 3 – Criando o modelo físico (estrutura)
- Seção 4 – Inserindo os dados
- Seção 5 – Criando o banco de dados MySQL
- Seção 6 – Sincronizando o DER com o banco de dados MySQL

# Utilização de Ferramenta *case* – MySQL Workbench

## SEÇÃO 1

### Utilidade de uma ferramenta *case*

Até o momento, você viu modelos e diagramas de dados que podem e devem ser feitos em uma simples folha de papel. Primeiramente, indica-se rascunhar os modelos em uma folha ou caderno, para depois serem passados para o computador.

Há alguns *softwares* que suportam a confecção do modelo físico dos dados e até mesmo interagem com o banco de dados, criando automaticamente todas as tabelas, atributos e relacionamentos no banco. Esses *softwares* são chamados de ferramenta *case*, que significa *Computer-Aided Software Engineering*. Traduzindo: Engenharia de *Software* Auxiliada por Computador.

Conforme Schaefer (2010), as ferramentas *case* servem para automatizar atividades de gestão de projetos de *software*, podendo aumentar a produtividade, diminuir o tempo e o custo de desenvolvimento de um projeto, mantendo um alto nível de qualidade.

Ainda segundo Schaefer (2010), as ferramentas *case* automatizam uma grande variedade de tarefas, entre elas, a geração de códigos. Em outras palavras, com uma das várias ferramentas *case* disponíveis no mercado, você pode confeccionar o diagrama físico dos dados e gerar o código para criar todas as tabelas, atributos e relacionamentos dentro de um banco de dados.

Entre as várias ferramentas disponíveis, neste livro, será utilizada a MySQL Workbench, por ser gratuita e apresentar alto grau de usabilidade e intuitividade, ideal para você que está começando a se familiarizar com bancos de dados.

Ela é a evolução da ferramenta DbDesigner, anteriormente fabricada e distribuída pela empresa Fabforce, descontinuada em 2006.

## SEÇÃO 2

### MySQL Workbench

Segundo MySQL (2010b), a MySQL Workbench é uma ferramenta *case* ou *software* de modelagem de dados da empresa MySQL que provê, ao DBA e a desenvolvedores, um conjunto de ferramentas integradas para modelar um banco de dados, desenvolver em SQL e administrar uma base de dados já criada.



Figura 45: Logomarca MySQL Workbench

#### SAIBA MAIS

Atualmente, a mais nova versão do *software* MySQL Workbench é a 5.2, que pode ser adquirida gratuitamente na página <<http://wb.mysql.com>>.

Neste curso, serão informados apenas os conceitos macros para confeccionar o DER físico na ferramenta MySQL Workbench. Esses pequenos procedimentos não podem ser confundidos com o manual do *software*. Se você desejar saber mais detalhes sobre sua utilização, deve fazer o *download* do manual de utilização no mesmo site.

## SEÇÃO 3

### Criando o modelo físico (estrutura)

Para criar o modelo físico no MySQL Workbench, abra-o e clique no ícone “*create a new EER model*” e posteriormente em “*Add diagram*”.

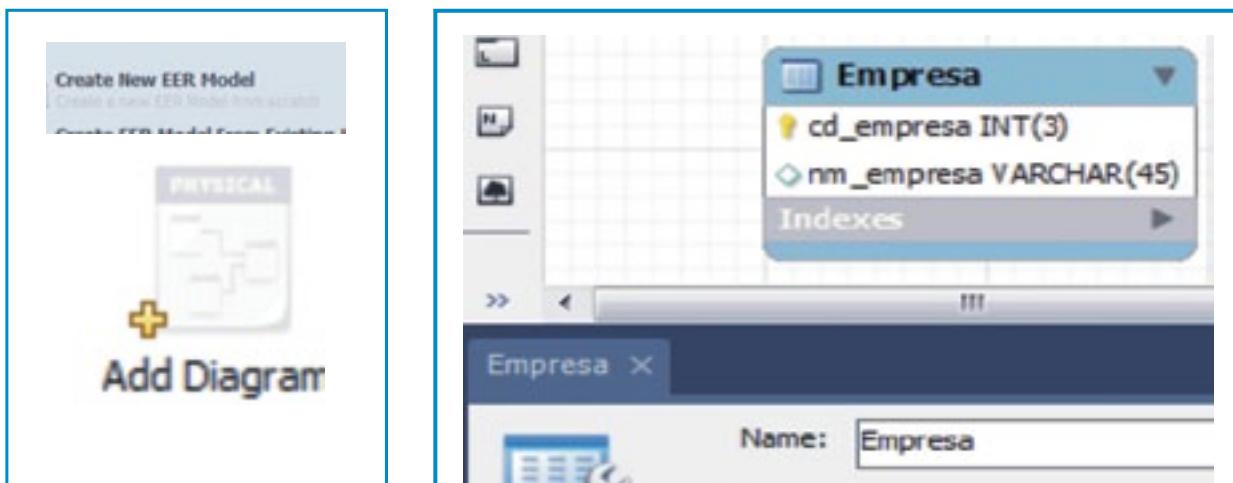


Figura 46: Iniciando um novo DER físico no MySQL Workbench

Utilize a barra de ferramentas do lado esquerdo para criar o modelo físico que você confeccionou na unidade de estudo 10.

Lembre-se de criar os relacionamentos corretamente e repare que o lado N dos relacionamentos é representado por um símbolo parecido com um “pé de galinha”, representação normal nas ferramentas *case* que você pode utilizar, também, para fazer o modelo lógico e físico no papel anteriormente à modelagem utilizando *softwares*.

Clicando duas vezes em cima das tabelas criadas, você poderá alterar seus nomes na aba: “Table” e adicionar os atributos necessários na aba: “Columns”.

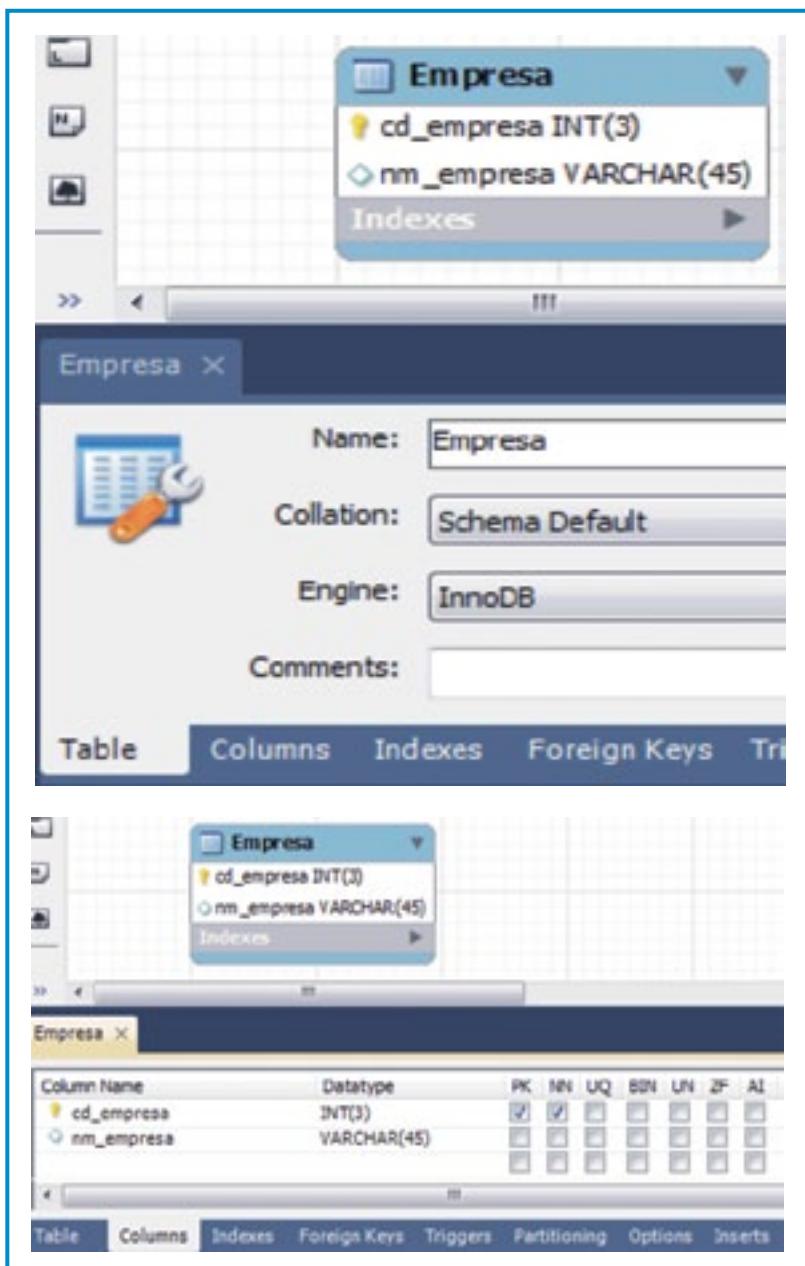


Figura 47: Editando as configurações da tabela

Sempre crie o nome das tabelas e dos atributos de acordo com um padrão intuitivo.

**Não** crie nomes como: “tabela1”, “atributo3”, “XyV456” etc. Lembre-se da lição aprendida na unidade 10, sobre os padrões de nomenclatura.

Repare na figura a aba “Columns”. Existem algumas opções que caracterizam cada atributo, entre eles, a opção PK, que indica a chave primária de cada tabela, podendo ser escolhida mais de uma, conforme já estudado.

A opção “NN” significa “*Not Null*”, ou seja, não nulo. Com essa opção marcada em algum atributo, o SGBD impede que, ao incluir um registro na tabela, esse atributo fique nulo, ou seja, sem nenhum registro inserido. Lembre-se de que um campo de chave primária nunca pode ser nulo. Então, se você definir um campo como chave primária, deve automaticamente marcar também a opção de “NN” para ela.

Outra opção importante a ser definida para cada atributo é a opção de “AI”, do inglês “*Auto Incremental*”. Essa opção autoincrementa um campo. Mesmo que você não insira nenhum valor, o SGBD completa esse campo a cada novo registro. Ela é muito útil para as tabelas que não têm uma predefinição de PK. No caso da tabela Nota fiscal, por exemplo, pode-se definir que o campo nr\_nota\_fiscal é a PK com o campo determinado para autoincrementar a cada novo registro, sendo que, a cada nova Nota fiscal cadastrada, esse campo é autocompletado gradativamente (Nota fiscal 01, Nota fiscal 02, Nota fiscal 03...).

Conforme você estudou anteriormente, quando há um relacionamento de cardinalidade N para N entre as tabelas, é necessário criar uma tabela auxiliar contendo as chaves primárias das duas tabelas que a originaram. No MySQL Workbench, a tabela auxiliar é criada automaticamente, inclusive com as duas chaves primárias necessárias.

Essa tabela gerada se relaciona com as duas tabelas que a originaram de forma N para um e deve ser considerada uma tabela normal, inclusive podendo se relacionar com outras e acrescentando os atributos necessários ou, até mesmo, ter seu nome alterado para o que melhor lhe convier.

➤ **Povoada** quando se diz que uma tabela está povoada, significa que uma tabela está cheia de dados nela inseridos.

## SEÇÃO 4

### Inserindo os dados

Após criada a estrutura, é hora de inserir os dados necessários, um processo simples e rápido.

Quando você está editando uma tabela, há uma aba chamada “*Inserts*”. Ao clicar nessa aba, visualizam-se todos os campos na horizontal, em formato de colunas. Insira manualmente os dados desejados na primeira linha, começando pela PK. Ao terminar, pula para a segunda linha, e assim por diante, até completar todos os registros desejados.

Fazendo isso para todas as tabelas, elas (na modelagem) estarão **povoadas** de dados, e o seu banco, depois de criado, já virá com esses registros.



Figura 48: Criação da tabela auxiliar em um relacionamento N para N

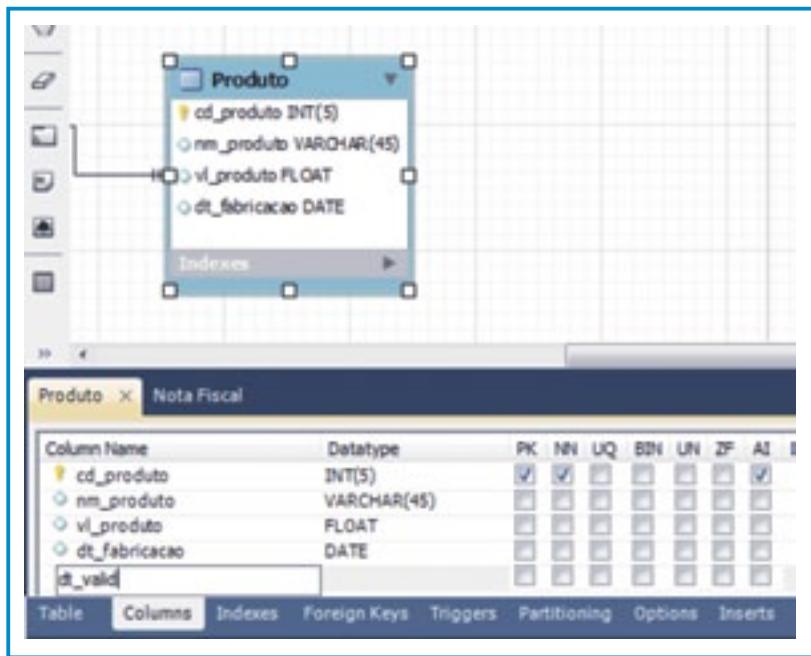


Figura 49: Inserção de dados nas tabelas

Depois da estrutura criada e dos dados inseridos, é hora de criar o banco de dados e sincronizar a modelagem com ele.

## SEÇÃO 5

### Criando o banco de dados MySQL

Existem várias formas de você acessar o MySQL e criar um banco de dados ou verificar sua estrutura. Mas, supondo que você instalou o EasyPHP, você pode utilizar o PhpMyAdmin, programa que está inserido no pacote EasyPHP, por meio do endereço: <<http://localhost/home/mysql>> para executar tal ação.

Na tela que se abrirá, aparecerá uma mensagem em inglês perguntando se você deseja criar um novo banco de dados (“Create new database”). Escolha o melhor nome para banco de dados e clique em “create”. Como estamos continuando o exemplo inicial de um *software* de controle de uma padaria, vamos chamar esse banco de dados de “padaria-silva”.

Ao lado esquerdo da tela, você verá o seu novo banco criado, possivelmente com mais um banco já pré-criado, o mydb, banco padrão que é criado na instalação do EasyPHP.

## SEÇÃO 6

### Sincronizando o DER com o banco de dados MySQL

Para conectar-se ao banco de dados MySQL por meio do MySQL Workbench, você deve seguir basicamente três passos:

- criar uma conexão com o banco de dados;
- sincronizar a modelagem;
- confirmar a criação e verificar a estrutura.

Prepare-se para criar efetivamente a estrutura de seu banco de dados!

#### ➤ Criar uma conexão com o banco de dados

Após criar o seu banco de dados conforme instruções da seção 6, é necessário, então, via MySQL Workbench, criar uma conexão para que possamos migrar a modelagem criada para esse novo banco de dados.

Para isso, deve-se, com o seu DER físico aberto, clicar em: “Database” e depois em “Manage Connection”. Crie uma nova conexão clicando em: “New” e insira os parâmetros de seu banco, lembrando que, se você instalou o EasyPHP no mesmo computador que está desenvolvendo sua modelagem, essa conexão deverá apontar para “localhost” ou para o IP “127.0.0.1”, que representa seu próprio computador. Se você criou uma senha para o seu banco de dados quando o criou no PhpMyAdmin, você também deve inseri-la nessa tela.

Seus parâmetros de conexão ficarão parecidos com a figura a seguir:

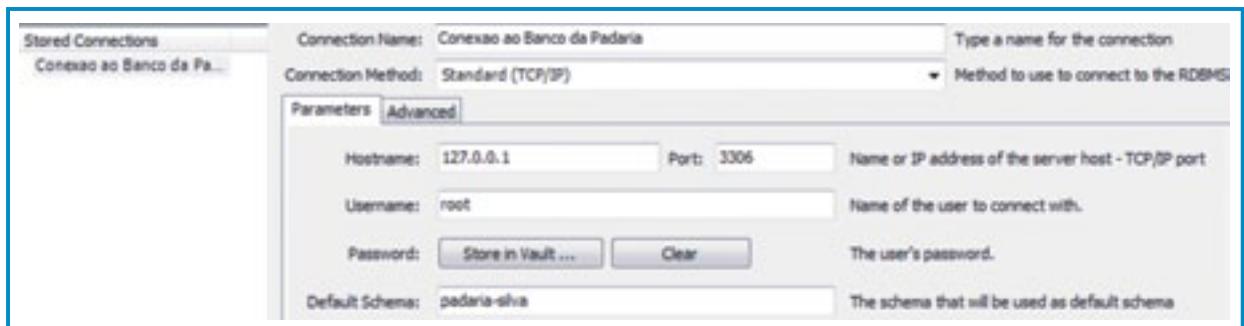


Figura 50: Parâmetros de conexão com o banco de dados

## ➤ Sincronizar a modelagem

É preciso, então, sincronizar a modelagem com o banco MySQL para que os *scripts* de criação e manipulação sejam criados e automaticamente executados no banco, criando assim todas as tabelas, campos, definindo as chaves primárias e estrangeiras e, consequentemente, os relacionamentos.

Para sincronizar a modelagem, você deve clicar no menu “*Database*” e em “*Synchronize Model*”. Uma nova tela se abrirá e nela você deverá selecionar sua conexão, criada anteriormente, e seguir adiante.

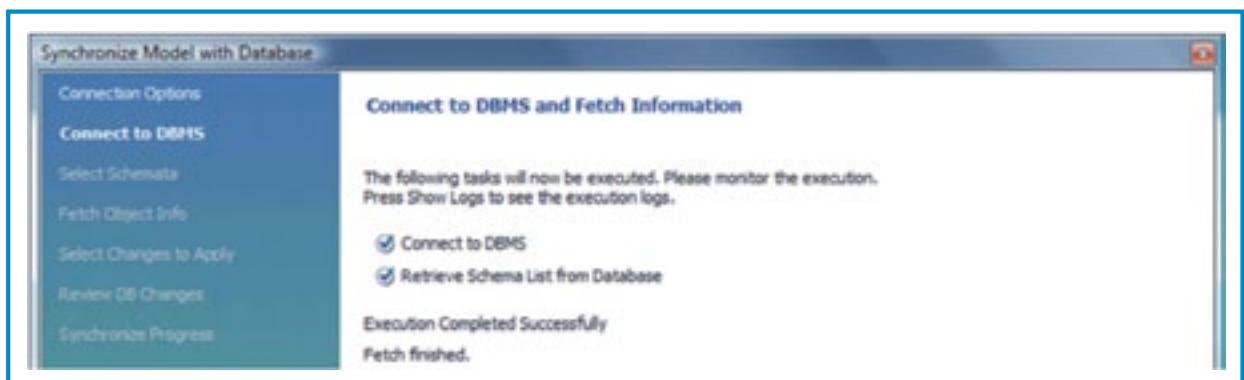


Figura 51: Tela de confirmação de conexão com o banco

Selecione seu banco de dados, que foi criado no PhpMyAdmin, e siga em frente.

Uma tela de confirmação aparecerá e, após esse passo, você deve confirmar o sentido da sincronização da sua modelagem, que será do modelo para o destino: banco de dados.

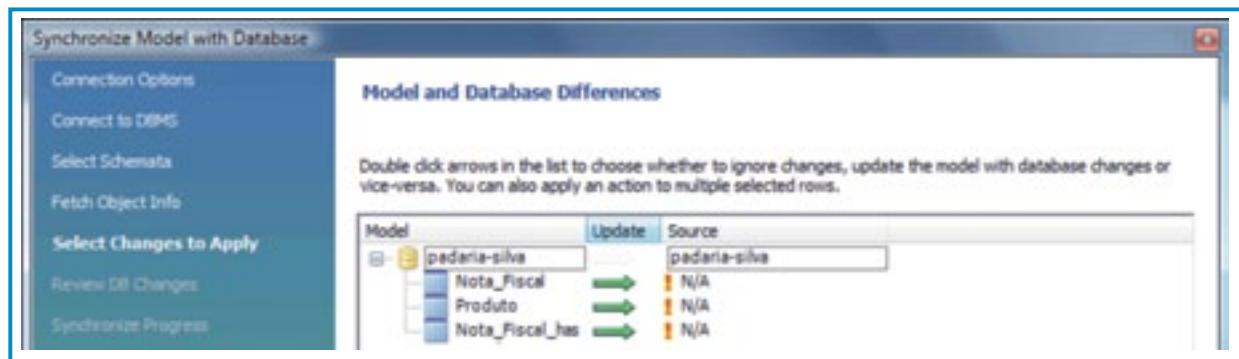


Figura 52: Tela de seleção do banco

Observe que na próxima tela é mostrado todo o *script* SQL que está sendo criado automaticamente pelo MySQL Workbench para gerar tabelas, campos e relacionamentos no banco.

Você deverá, então, clicar em executar e, quando a tela de confirmação do sincronismo aparecer, finalizar o processo, criando a estrutura para seu banco de dados.

Pronto! Você já está com sua estrutura de banco de dados montada, mas, logicamente, sem nenhuma informação armazenada. Para verificar sua estrutura do banco, vá diretamente ao MySQL por meio do PhpMyAdmin e confirme se o sincronismo foi efetuado com sucesso.

## » Confirmar a criação e verificar a estrutura

Acesse o endereço <http://localhost/home/mysql>. Do lado esquerdo, você verá o(s) banco(s) de dado(s) já criado(s). Clique no banco de dados que você criou – se você seguiu o exemplo deste livro, o banco de dados é o “padaria-silva”.

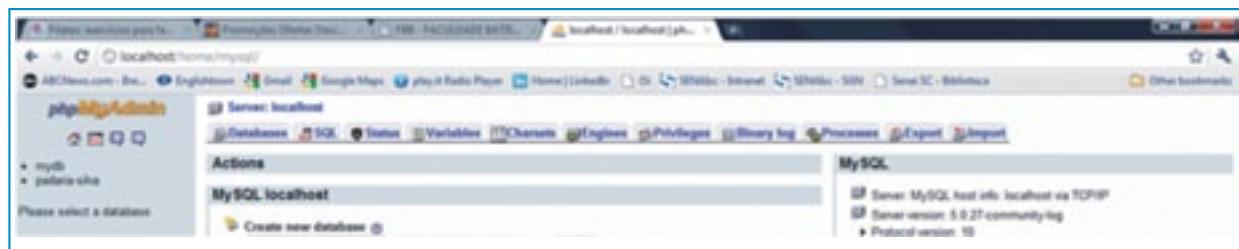


Figura 53: Visão dos bancos de dados do MySQL por meio do PhpMyAdmin

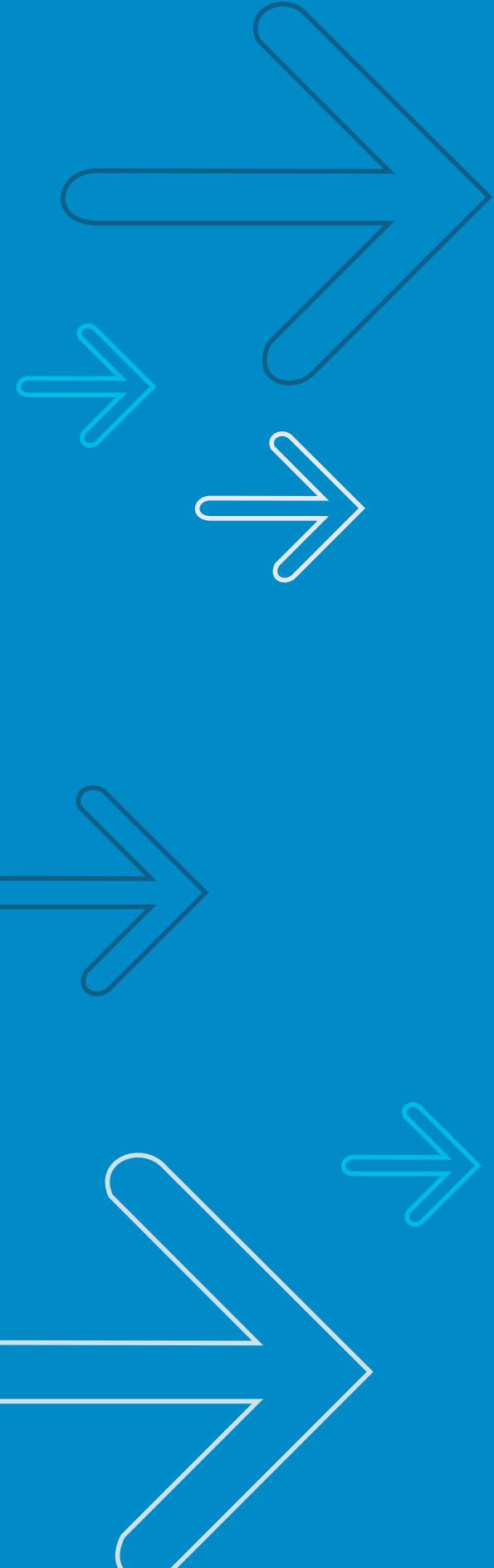
Aparecerão, então, todas as tabelas que você criou na modelagem. Para mostrar seus respectivos campos, clique no ícone “Structure” e, para mostrar os valores que estão inseridos nos campos, clique em “Browse”. Ambos os botões localizam-se ao lado de cada tabela.

Pelo PhpMyAdmin, você também pode alterar a estrutura do banco, tabelas e inserir ou excluir dados e, posteriormente, fazer a **engenharia reversa** para a modelagem no MySQL Workbench, mas sugere-se executar essas alterações sempre em lugar único, na ferramenta *case* que você desejar e, posteriormente, sincronizá-las com o banco de dados.

Conforme você já aprendeu, o SGBD MySQL gerencia os bancos de dados por meio de *scripts* SQL. O que o MySQL Workbench faz nada mais é do que transformar a modelagem e os dados inseridos nesses *scripts* para que sejam entendíveis pelo MySQL, facilitando seu trabalho e evitando que você tenha de confeccionar o *script* com todos os comandos manualmente.

Repare que esse processo poderá ser executado a qualquer momento, mesmo depois de ter sido executado pela primeira vez. Por exemplo, se depois de criada a estrutura e inseridos os dados você precisar alterá-los, faça isso na modelagem, exclua ou inclua tabelas e campos, altere relacionamentos, chaves primárias, insira, altere ou exclua dados e sincronize-os novamente. Você perceberá que suas alterações se refletirão no banco de dados.

Mas, além de consultas, é possível manipular dados e definições de estrutura e de segurança do banco de dados utilizando SQL. Esse é o tema da próxima unidade. Aproveite as aulas para trocar ideias com o professor. Não fique com dúvidas.



# **Unidade de estudo 13**

## **Seções de estudo**

- Seção 1 – O que é SQL**
- Seção 2 – Breve histórico**
- Seção 3 – O funcionamento**
- Seção 4 – DDL**
- Seção 5 – DML**
- Seção 6 – DCL**
- Seção 7 – Vantagens na utilização do SQL**

## SEÇÃO 1

### O que é SQL

SQL significa *Structure Query Language*, em português, Linguagem de Consulta Estruturada. Foi desenvolvida especificamente para o padrão relacional de dados e permite não somente consultas, como o nome indica, mas também manipulação de dados e definições de estrutura e de segurança do banco de dados. Além do MySQL, há vários SGBDs atualmente que utilizam a linguagem SQL como padrão para a manutenção e gerenciamento de seus bancos de dados, entre eles Firebird, Microsoft SQL Server, PostgreSQL, ORACLE.

Segundo Korth, Silverschatz e Sudarshan (1999), a SQL utiliza uma combinação de construtores em álgebra e cálculo relacional e é a linguagem comercial mais utilizada do mercado, já que apresenta certa facilidade para seus usuários. A intenção deste livro, assim como no caso das ferramentas *case* e do MySQL, não é apresentar um tutorial ou guia completo para os usuários, desenvolvedores ou administradores do banco, mas apresentar conceitos e sintaxes básicas para um breve entendimento dos fundamentos dessa linguagem. Conceitos mais avançados dessa linguagem são detalhados no livro de Banco de Dados II.

## SEÇÃO 2

### Breve histórico

A primeira versão do SQL foi desenvolvida no início dos anos 1970, com o patrocínio da empresa IBM. Fundamentava-se no modelo relacional de dados e chamava-se SEQUEL, que significa *Structure English Query Language*. Segundo Abreu e Machado (2004), a partir de 1976, essa linguagem foi revisada e ampliada e, por razões jurídicas, teve seu nome alterado para SQL.

Por volta de 1986, os padrões para a SQL foram renovados pela International Standards Organization (ISO) e pelo American National Standards Institute (ANSI) e publicados no mercado como SQL-86. A partir daí, uma série de revisões foram feitas por essas duas entidades e por outras.

A empresa IBM personalizou a então versão em 1989 e lançou seu próprio padrão, nomeado de SQL-89. A partir de 1992, a ANSI e a ISO revisaram diversas vezes suas versões e lançaram no mercado o padrão SQL-92 em 1992, o SQL3 em 1999 e os padrões SQL: 2003 e SQL: 2008 em 2003 e 2008, respectivamente. Nessas últimas versões, foram adicionadas algumas características de orientação a objetos, melhorados os métodos de consultas com a utilização de consultas recursivas e iniciada a utilização de gatilhos.

## SEÇÃO 3

### O funcionamento

Ainda que a SQL tenha sido padronizada pelo ANSI e pela ISO, diversos fabricantes de SGBDs a adaptaram para seus próprios aplicativos. Dessa forma, alguns conceitos apresentados aqui podem variar de SGBD para SGBD, mas foco estará nos padrões atuais da ANSI, que se aplicam entre outros, ao MySQL.

Dentro da SQL, há várias funções e comandos que são divididos em grupos. Os mais conhecidos e utilizados grupos de comandos são DDL, DML e DCL, assunto para as três próximas seções, nas quais serão observadas suas características e sintaxes.

## SEÇÃO 4

### Comandos DDL

DDL significa *Data Definition Language*, em português, Linguagem de Definição de Dados. Representa o conjunto de comandos que permite criação, alteração e exclusão da estrutura das tabelas do banco (SILVA, 2001), ou seja, define como os dados são estruturados.

Alguns exemplos de comandos DDL são: CREATE, ALTER e DROP.

## ➤ CREATE

O comando CREATE cria um objeto, que pode ser uma tabela ou até mesmo um banco de dados. Sua sintaxe para criação de novos bancos de dados é: CREATE DATABASE “nome do banco de dados” Onde “nome do banco de dados”, que deve ser substituída pelo nome do seu banco de dados, sem aspas. Exemplo:

```
CREATE DATABASE padaria-silva
```

Já a criação de novas tabelas dentro do banco de dados é representada com a seguinte sintaxe: CREATE TABLE “nome da tabela” (

“Descrição do campo 1”  
“Descrição das chaves” )

**Exemplo:**

```
CREATE TABLE produto (  
cd_produto INT not null,  
nm_produto VARCHAR(20),  
PRIMARY KEY (cd_produto)  
)
```

Onde cd\_produto e nm\_produto representam os nomes dos campos, INT e VARCHAR o tipo do campo, nesse caso, respectivamente campos dos tipos números inteiros e do tipo letras e números com no máximo 20 caracteres. Primary Key representa a definição da chave primária que, no caso dessa tabela, é o atributo cd\_produto (código do produto).

Agora será observada a criação de duas tabelas que possuem relacionamentos de N para 1, nas quais, ao lado N, haverá uma chave estrangeira (FK), que, conforme você já aprendeu, é sempre a PK da outra tabela que representa o lado 1 do relacionamento.

Serão criadas as tabelas empresa e cliente do banco de dados da então padaria-silva. Você já sabe que empresa se relaciona com os clientes de 1 para N, então será criado primeiramente o lado 1 dessa relação, que é a tabela empresa, com apenas três atributos:

```
CREATE TABLE empresa (  
cd_empresa INT not null,  
nm_empresa VARCHAR(40),  
ds_endereco VAR-  
CHAR(50),  
PRIMARY KEY (cd_produto)  
)
```

Após o lado 1, cria-se o lado N do relacionamento. Para isso, é preciso adicionar mais um atributo, que será a FK, acrescentando a sintaxe a seguir na parte de definição das chaves:

```
CONSTRAINT “nome da constraint” FOREIGN KEY (“nome do atributo”)  
REFERENCES “nome da tabela a qual este atributo é a PK (“nome desse campo dentro dessa tabela”).
```

Dessa forma, toda a sintaxe de criação dessa tabela fica:

```
CREATE TABLE cliente (  
cd_cliente INT not null,  
nm_cliente VARCAHR(20),  
dt_nascimento DATE,  
cd_empresa INT,  
PRIMARY KEY (cd_produto),  
CONSTRAINT c1 FOREIGN KEY  
(cd_empresa) REFERENCES  
empresa(cd_empresa))
```

Repare que o atributo de FK da tabela cliente deve ser do mesmo tipo que está configurado na tabela empresa.

Agora você acabou de criar um relacionamento de N para 1 entre as tabelas Cliente e Empresa.

A sintaxe CONSTRAINT significa restrição e impede que o usuário insira no campo de chave estrangeira um valor que seja diferente de qualquer valores inseridos na PK da tabela empresa.

Observe também que o nome c1 não representa nenhum atributo ou tabela: é simplesmente o nome dado para a restrição criada.

Exemplificando: se há uma empresa “Padaria Silva” de código 01, cadastrada na tabela empresa, quando for cadastrado o cliente, o usuário nunca poderá inserir no campo cd\_empresa da tabela cliente o código 02, pois isso significaria que o cliente é da empresa de código 02, a qual não existe.

## ➤ DROP

Já que foi possível criar o banco de dados e as tabelas via comandos SQL, logicamente é possível excluí-las também via comandos SQL. Para isso, utiliza-se o comando DROP, que possui a sintaxe a seguir:

DROP DATABASE “nome do banco de dados”  
DROP TABLE “nome da tabela”  
Exemplos:

```
DROP DATABASE padaria-silva  
(para excluir o banco de dados  
padaria-silva)  
DROP TABLE produto (para ex-  
cluir a tabela produto)
```

## ➤ ALTER

Veja a sintaxe do comando ALTER, que altera a estrutura de uma tabela:

ALTER TABLE “nome da tabela”(  
ADD COLUMN “nome do campo”);  
Exemplo:

```
ALTER TABLE produto (  
ADD COLUMN ds_produto  
VARCHAR(50));
```

Nesse caso, o campo ds\_produto está sendo adicionado na tabela produto, porém é possível excluir campos substituindo o ADD pelo DROP, renomear campos e tabelas substituindo o ADD pelo RENAME, alterar colunas substituindo pelo CHANGE etc.

## SEÇÃO 5

### Comandos DML

DML significa *Data Manipulation Language*, em português, Linguagem de Manipulação de Dados. Engloba os comandos necessários para alterações, inserções e exclusões de dados dentro das tabelas (KORTH; SILVERSCHATZ; SUDARSHAN, 1999).

Alguns exemplos de comandos DML são: INSERT, DELETE e SELECT.

## ➤ INSERT

O comando INSERT insere um registro dentro dos campos das tabelas. Sua sintaxe para inserção de registros é:

INSERT INTO “nome da tabela” (“nome do campo”) VALUES (“valor do campo”);

Onde “nome da tabela”, “nome do campo” e “valor do campo” devem ser substituídos respectivamente pelo nome da tabela desejada, nome do atributo que deseja inserir os dados e o valor a ser inserido, sem aspas. Exemplo:

```
INSERT INTO produto (cd_pro-  
duto) VALUES (1);
```

Nesse caso, você está inserindo o primeiro produto na tabela, que possui código 1. Mas e os outros campos? De acordo com os exemplos anteriores, na linguagem DDL, a tabela produtos possui três campos: o cd\_produto, o nm\_produto e o ds\_produto. Apenas foi inserido o código do produto no campo cd\_produto, que é a PK dessa tabela. Então, é necessário inserir o restante dos dados completando o registro, e você pode fazer isso de uma só vez, ou seja, inserir dados em mais de um campo com apenas um comando – INSERT – separando os campos por vírgulas. Veja:

```
INSERT INTO produto (nm_pro-  
duto, ds_produto) VALUES  
("refrigerante", "bebida com  
gás 300ml");
```

Repare que os valores devem estar na mesma ordem que os campos. Essa é a sintaxe básica do comando INSERT para atribuir valores para cada atributo, mas você pode inserir registros para todos os campos de uma só vez, sem a necessidade de descrever todos os campos, apenas inserindo os valores na mesma ordem dos campos. Como exemplo, será inserido o segundo item na tabela produto:

```
INSERT INTO produto VA-  
LUES (2,"arroz", "pacote de  
arroz branco com 5 kg");
```

Nesse caso, foi inserido o produto arroz, que possui código 2 no banco de dados, e assim segue, sucessivamente, até inserir todos os registros desejados.

## ➤ DELETE

Se você deseja excluir alguns registros, o comando é tão simples quanto o INSERT. Basta utilizar o DELETE juntamente com a cláusula WHERE.

DELETE FROM “nome da tabela” WHERE “descrição dos parâmetros”.

Onde os parâmetros são as condições necessárias que determinarão qual registro será excluído. Exemplo:

```
DELETE FROM produto  
WHERE cd_produto = 1
```

Nesse caso, está sendo excluído o produto 1 da tabela produto que, de acordo com o que foi inserido anteriormente, é o produto “refrigerante”.

## » SELECT

Você se lembra de que o SQL, mesmo não sendo apenas uma linguagem de consulta de dados, tem seu nome ligado a essa função, certo? Portanto, serão apresentados alguns parâmetros do comando mais comumente utilizados em um banco de dados, segundo Abreu e Machado (2004): o SELECT, que tem a função de extrair dados de uma tabela.

A sintaxe básica do comando SELECT é:

SELECT “nome do campo” FROM “nome da tabela”.

Assim como no comando INSERT, você também pode selecionar vários campos para serem pesquisados separando-os por vírgula:

```
SELECT “nome do campo 1”,  
“nome do campo 2” FROM  
“nome da tabela”
```

No caso anterior, são listadas, em uma tabela, apenas as linhas das colunas desejadas. Exemplo:

Com o comando:

```
SELECT cd_produto,nm_produto  
FROM produto
```

Devem ser apresentadas apenas as colunas cd\_produto e nm\_produto conforme resultado a seguir:

cd_produto	nm_produto
1	Refrigerante
2	Arroz

Figura 54: Resultado apresentado com o SELECT cd\_produto e nm\_produto

Se você desejar listar todos os campos de uma tabela, não é necessário descrever todos. **Basta colocar o \*** (asterisco) no lugar dos atributos. Assim:

```
SELECT * FROM produto
```

Com essa sintaxe, deve ser apresentado o seguinte resultado:

cd_produto	nm_produto	ds_produto
1	Refrigerante	Bebida com gás 300ml
2	Arroz	Pacote de arroz branco com 5kg

Figura 55: Lista de todos os campos da tabela

Caso houvesse mais campos inseridos na tabela, mais colunas apareceriam em sua respectiva ordem.

### SELECT com cláusulas e operadores lógicos

Em uma tabela com grande número de registros, os comandos anteriores, que mostram todos os registros inseridos, demandariam recursos do servidor, podendo prejudicar sua *performance*. Para limitar as consultas, trazer resultados mais “limpos” e onerar menos os recursos do servidor, utilizam-se a cláusula WHERE e operadores lógicos, como AND, OR e LIKE, operadores aritméticos, como +, – e / e também operadores de comparação, como =, < e !=.

Por exemplo, se for necessário trazer apenas o nome do item de código 1, pode-se utilizar a sintaxe:

```
SELECT nm_produto WHERE cd_produto = 1
```

Devendo apresentar o resultado:

nm_produto
Refrigerante

Figura 56: Item de código 1

Para utilizar operações lógicas como parâmetros, veja a seguir:

```
SELECT ds_produto WHERE cd_produto = 2 AND nm_produto LIKE “arroz”
```

Essa sintaxe trará a descrição de um produto que possui código 02 e cujo nome é arroz, apresentando o seguinte resultado:

ds_produto
Pacote de arroz branco com 5kg

Figura 57: Descrição do produto com código 2

Existem muitas sintaxes e operações lógicas do comando SELECT e dos outros aqui apresentados, as quais devem ser pesquisadas para melhor entendimento.

## SEÇÃO 6

### DCL

DCL significa *Data Control Language*, em português, Linguagem de Controle de Dados. Como o nome já diz, possui comandos responsáveis pelo controle de acesso ao banco de dados, como o GRANT e REVOKE que, respectivamente, concedem e revogam o acesso de usuários específicos a um banco de dados. Esse processo é extremamente importante para manter a segurança das informações constantes no banco de dados.

Segundo Abreu e Machado (2004), cada usuário tem uma determinada necessidade em relação aos dados armazenados. De acordo com as definições feitas no projeto de banco de dados, alguns usuários somente podem consultar dados, outros podem atualizar alguma informação, outros inserir, e assim por diante. Na maioria dos casos, os acessos são controlados por funções e departamentos ou, tecnicamente falando, por tabelas, sendo que as permissões de acesso podem se dar para um conjunto de dados.

Exemplo:

Um usuário simples da área comercial pode consultar os dados referentes aos pedidos do mês, mas não pode alterá-los e tampouco consultar ou alterar dados de outro setor como o setor contábil. Porém, um usuário avançado do mesmo setor comercial pode, além de consultar, alterar os dados dos pedidos, mas continua sem ler ou alterar os dados do setor contábil.

Para que todas essas restrições sejam efetuadas, os comandos GRANT e REVOKE são primordiais. É com eles que se executam essas permissões.

#### ➤ GRANT

O comando GRANT, que concede acesso a uma tabela específica a um determinado usuário, possui a seguinte sintaxe:

GRANT “lista de privilégios” ON “Nome da tabela” TO “usuário”

Se for necessário dar acesso para o usuário Vinicius para consultar a tabela de produtos, por exemplo, a sintaxe para executar esse comando ficaria:

```
GRANT SELECT ON pedido TO Vinicius
```

Se preferíssemos dar acesso de gravação de novos registros à mesma tabela para o usuário Maurício, a sintaxe seria:

```
GRANT INSERT ON pedido TO Mauricio
```

Assim, temos outras sintaxes, como UPDATE, que permite atualizar registros já inseridos, DELETE, que permite excluir registros, e ALL, que concede todos os privilégios, entre outras. Se você ainda desejar oferecer acesso total a um usuário a uma tabela e também conceder acesso para que ele possa controlar o acesso de outras pessoas, basta adicionar o comando WITH GRANT OPTION ao final da sintaxe. Exemplo:

```
GRANT ALL ON pedido TO Marta WITH GRANT OPTION
```

Pronto, Marta agora possui poderes para executar quaisquer ações na tabela pedidos e também para conceder ou revogar acesso de outros usuários nessa tabela.

## REVOKE

Assim como um administrador ou usuário avançado pode dar acesso a outro usuário, ele também pode revogá-lo, utilizando o comando REVOKE, que possui uma sintaxe similar ao comando GRANT. Para revogar o acesso de gravação de Paulo à tabela de clientes, utiliza-se a sintaxe a seguir:

```
REVOKE INSERT ON cliente FROM Paulo
```

Caso se desejar retirar os acessos de seleção e inserção de Felipe à mesma tabela, pode-se fazer isso no mesmo comando, no qual é necessário separar os dois privilégios por vírgulas:

```
REVOKE INSERT, SELECT ON cliente FROM Paulo
```

Agora, além de você já saber como criar o banco, as tabelas e inserir os registros, também sabe como manter o mínimo necessário de segurança de acesso para garantir a segurança das informações existentes no banco de dados.

## SEÇÃO 7

### Vantagens na utilização do SQL

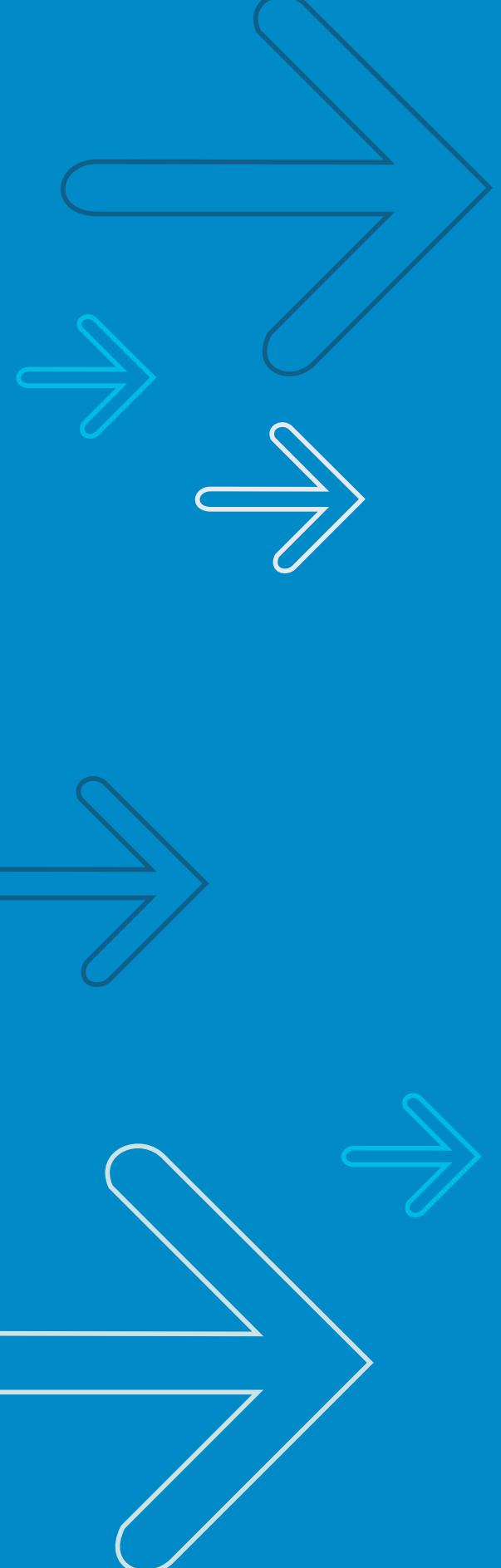
Abreu e Machado (2004) apontam algumas vantagens da utilização da linguagem SQL:

- Independência entre fabricantes.
- Portabilidade entre computadores.
- Redução dos custos de treinamento.

Isso tudo quer dizer que, mesmo que alguns fabricantes de SGBDs tenham personalizado os métodos de desenvolvimento de estruturas do banco e de manipulação de dados, a imensa maioria dos comandos padronizados pela ANSI e ISO é respeitada fazendo com que usuários avançados, administradores e desenvolvedores tenham conhecimentos equivalentes nos mais diversos fabricantes de SGBD aqui mencionados, não necessitando altos investimentos em caso de troca de um dos SGBDs por outro também baseado em SQL.

Com certeza, a SQL não é perfeita e apresenta também algumas desvantagens. Uma delas, também apresentada por Abreu e Machado (2004), é que o fato de ser tão padronizada pode levar a uma natural inibição da criatividade, pois o desenvolvedor retrai-se a soluções únicas já predispostas na linguagem. Porém, diante de todas as vantagens que possui, com certeza, esse e outros possíveis pontos ruins são minoria e não apresentam relevância na escolha da linguagem manipuladora do banco de dados.

Como a SQL é a linguagem mais utilizada mundialmente em termos de usuários e de quantidades de SGBDs atualmente disponíveis, é claro que você precisa conhecer como são feitas as transações de bancos de dados. Então, fique atento! Esse é o tema da próxima unidade.



# **Unidade de estudo 14**

**Seções de estudo**

**Seção 1 – Conceitos  
Seção 2 – Na prática  
Seção 3 – Controle transacional**

# Transações de Banco de Dados

## SEÇÃO 1

### Conceitos

Você já ouviu falar em uma transação financeira? Provavelmente sim, e de várias formas, como, por exemplo, execução de transações na bolsa de valores e execução de empréstimos ou transferências bancárias. Isso são transações, ou seja, são ocorrências que resultam, geralmente, em atualização de sua conta bancária, seja um débito ou um crédito.

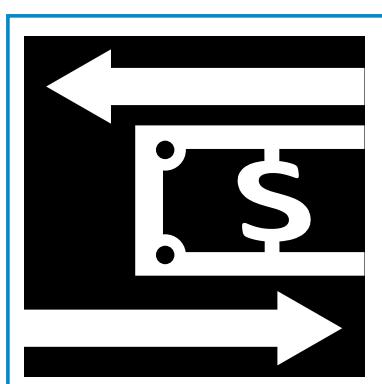


Figura 58: Transação bancária

Uma transação em banco de dados tem conceitos similares.

É considerado uma transação todo o conjunto de operações que resultam em uma consulta, podendo atualizar dados constantes em uma tabela específica.

Segundo Korth, Silberschatz e Sudarshan (1999), uma transação é uma unidade de execução de programa que acessa e [...] atualiza vários itens de dados [...], geralmente, é o resultado da execução de um programa de usuário escrito em uma linguagem de manipulação de dados de alto nível ou em uma linguagem de programação, como por exemplo, SQL, COBOL, C ou PASCAL.

## SEÇÃO 2

### Na prática

Por exemplo, para efetuar o pagamento de um boleto pela internet, o banco de dados do banco faz várias operações em ordem cronológica, que juntas podem ser consideradas uma transação. Nesse exemplo, a ordem das operações que o banco de dados da instituição financeira executa após o *login* no *internet banking* pode ser:

1. Verificação do código de barras do boleto para confirmação de sua veracidade.
2. Confirmação da existência de alguma regra que poderia não permitir ao correntista pagar esse tipo de conta.
3. Verificação do saldo atual da conta.
4. Verificação da senha para pagamento.
5. Efetivação do pagamento.
6. Atualização do saldo da conta para refletir esse pagamento.

Todo esse conjunto de operações pode ser considerado uma transação dentro do banco de dados da instituição financeira, não devendo acontecer em ordem diferente. Em caso de erro em alguma dessas operações, as outras operações não devem continuar. Se houver lentidão em uma delas, a próxima obrigatoriamente deverá aguardar até que esta se conclua.

## SEÇÃO 3

### Controle transacional

Internamente, uma transação pode ser executada de diversas maneiras, sempre iniciando quando uma instrução SQL é executada e terminando quando o comando COMMIT é emitido.

Você pode estar se perguntando então: “por que as criações de banco de dados, de tabelas, campos e inserções de dados, considerados transações, não necessitavam de COMMIT, ainda que cada um daqueles comandos sejam transações?”.

Bom, para esses comandos, o COMMIT já é feito automaticamente, sendo considerado um AUTO COMMIT. A SQL entende que cada operação é uma transação. Para você declarar que deseja fazer uma transação com vários comandos, é necessário iniciá-la com a sintaxe: BEGIN (“conjunto de comandos SQL”) END

O final de uma transação também é representado por COMMIT ou ROLLBACK, onde COMMIT informa ao banco que todas as operações ali incluídas foram finalizadas com êxito e ROLLBACK informa que as operações resultaram em algum erro e o restante da transação não pode continuar. Aborta-se, então, o processo e volta-se ao estado anterior ao início da transação.

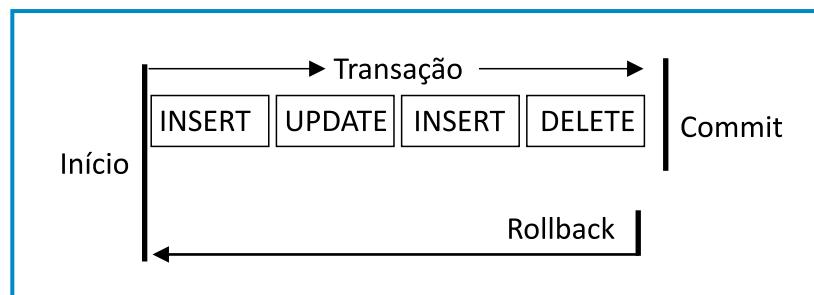


Figura 59: Controle das transações

Há casos em que, se houver algum erro, é importante que, mesmo assim, uma parte da transação seja executada. Nesses casos, é possível inserir pontos de salvamento, os *savepoints*, e configurar o ROLLBACK para retornar a um ponto específico da transação, não voltando, então, ao estado anterior a essa transação, mas a um estado anterior ao acontecimento do erro.

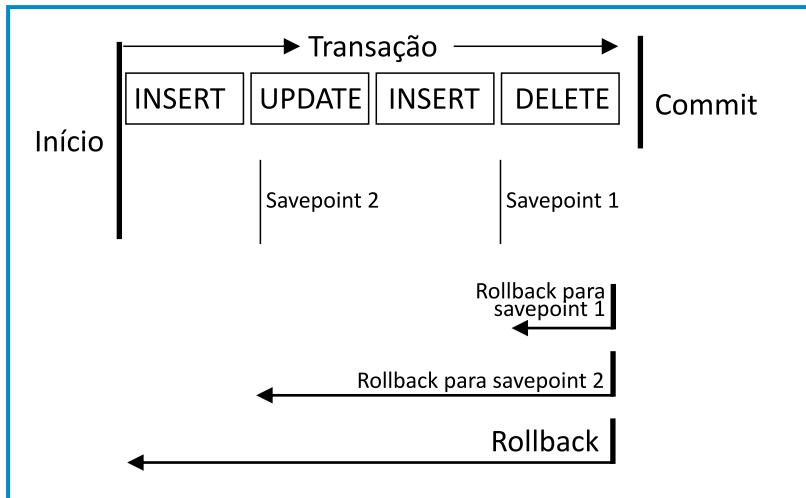
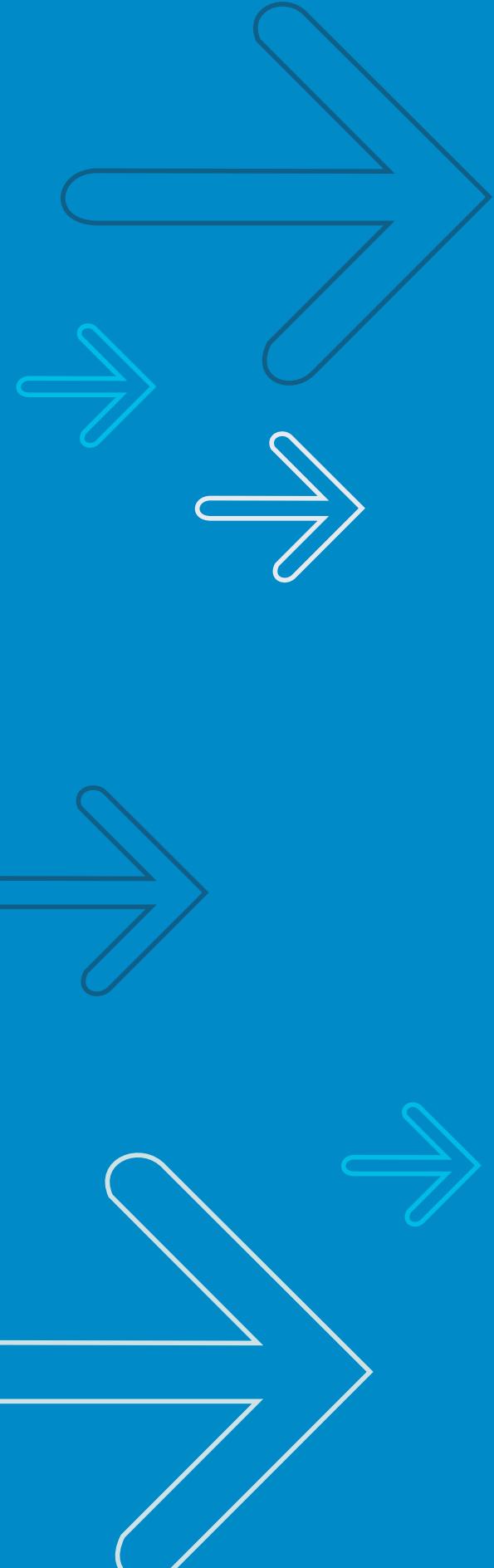


Figura 60: Controle das transações com *savepoints*

Na próxima unidade, você conhecerá a normalização e a integridade referencial, que padronizam e complementam os bancos de dados. Até lá!



# **Unidade de estudo 15**

**Seções de estudo**

**Seção 1 – Normalização  
Seção 2 – Integridade referencial**

# Elementos Complementares – Normalização e Integridade Referencial

## SEÇÃO 1

### Normalização

Normalização em banco de dados é um processo que busca padronização e consistência na armazenagem, evitando redundâncias de dados e permitindo acessos consistentes às informações ali armazenadas.

O conceito de normalização foi introduzido em 1970 por E. F. Codd, um dos mentalizadores, na mesma época, da abordagem relacional de dados. Essa técnica permite substituir um conjunto de registros repetitivos por entidades relacionadas que apresentarão dados únicos e padronizados (ABREU, MACHADO, 2004).

#### SAIBA MAIS

Há várias formas normais atualmente estabelecidas, mas não é foco deste livro abordar cada uma delas. Você poderá consultá-las no livro Sistemas de Bancos de Dados, cuja referência se encontra ao final deste livro.

## SEÇÃO 2

### Integridade referencial

A integridade referencial é um conjunto de regras que fornecem a garantia de que muitas mudanças feitas no banco de dados por usuários autorizados não resultem em perda de consistência de dados (KORTH; SILVERSCHATZ; SUDARSHAN, 1999). Dessa forma, é possível afirmar que a integridade protege o banco de dados de alterações indevidas acidentais.

Vejamos um exemplo prático em que no banco de dados de uma padaria há tabelas e classes de produtos. Se o produto é refrigerante, e a classe desse produto é bebidas, pode-se afirmar que o produto está dentro da classe de bebidas. Dessa forma, essas duas tabelas se relacionariam de N para 1, conforme figura a seguir.

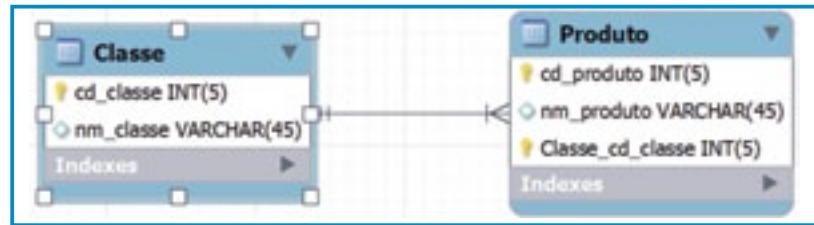


Figura 61: Tabelas de produto e classe e seu relacionamento

Para cadastrar um produto, é necessário sempre associá-lo a uma classe, por exemplo: refrigerante a bebidas, queijo a derivados de leite, pão francês à classe pães. Lembre-se de que as classes não podem ser excluídas caso exista algum produto cadastrado para elas.

Classe		Produto		
cd_classe	nm_classe	cd_produto	nm_produto	cd_classe
1	Paes	1	Pao Frances	1
2	Bebidas	2	Refrigerante	2
3	Carnes	3	Pao Integral	1

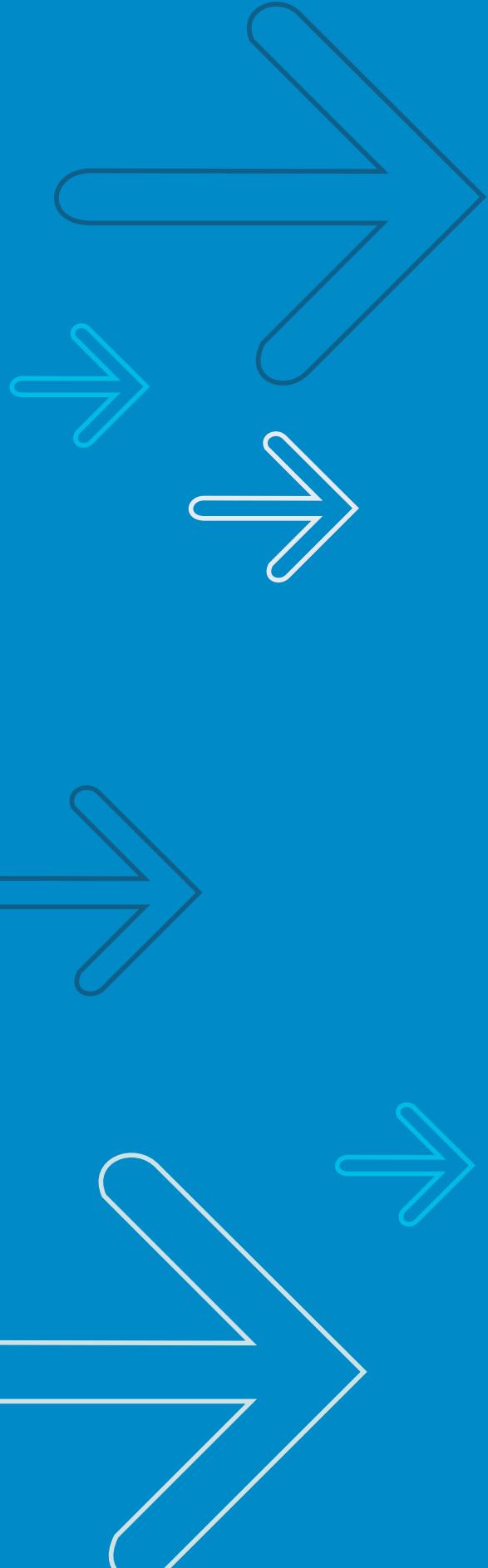
Figura 62: Registros das tabelas produto e classe

Na figura anterior não podemos excluir a categoria pães, pois existe o item pão francês cadastrado e associado a essa categoria.

Isso é um resumo do que é integridade referencial. Você verá mais sobre esse tema no livro de Banco de Dados II.

Agora que você já viu todos os conceitos de bancos de dados, SGBDs, modelagens, ferramenta *case* e comandos SQL, você está apto para projetar, modelar e criar um banco de dados, assim como inserir e manipular seus registros, conceder e revogar permissões de acesso. Conhecimento nunca é demais. Continue lendo o livro e absorvendo todo o conhecimento possível. Interaja sempre com seu professor para sanar suas dúvidas e fazer exercícios adicionais.

As unidades 16 e 17 tratam, respectivamente, das arquiteturas atualmente disponíveis para suportar os diversos tipos de bancos de dados e métodos de extração de informações. Siga em frente e aproveite!



# **Unidade de estudo 16**

## **Seções de estudo**

- Seção 1 – Sistemas de computadores pessoais**
- Seção 2 – Plataformas centralizadas**
- Seção 3 – Sistemas cliente-servidor**
- Seção 4 – Sistemas distribuídos**

# Arquiteturas de Bancos de Dados

## SEÇÃO 1

### Sistemas de computadores pessoais

Para os mais variados tipos de necessidades, há uma solução de banco de dados que se difere em sua arquitetura.

Devem ser considerados vários aspectos para garantir a boa *performance* das transações do banco de dados, como localização do público usuário, métodos de trabalho, número de consultas  $\times$  alterações, entre outros, que são detalhados no levantamento de requisitos e regras de negócio na confecção do projeto de banco de dados. Nas próximas seções, você verá uma breve análise de algumas arquiteturas de banco de dados disponíveis.

Sistemas que rodam em computadores pessoais geralmente possuem um pequeno número de usuários, podendo ser até mesmo um simples usuário que deseja controlar suas contas pessoais ou de sua microempresa.

Não exigem grandes investimentos, e um usuário avançado pode buscar um SGBD gratuito, instalá-lo e configurá-lo com auxílio de tutoriais específicos.

## SEÇÃO 2

### Plataformas centralizadas

Na arquitetura centralizada, existe um computador com grande capacidade de processamento onde é hospedado o SGBD.

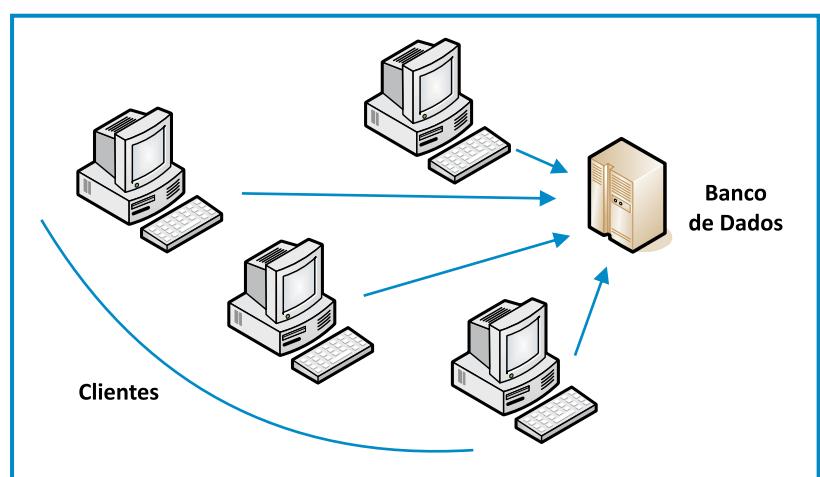


Figura 63: Plataforma centralizada



Essa arquitetura é utilizada geralmente em grandes empresas que conseguem justificar o alto investimento em infraestrutura. Nesse caso, é possível ter milhares de usuários com acessos simultâneos, o que exige alto investimento em *hardware* para possibilitar boa *performance* do banco mesmo com tantos acessos.

## SEÇÃO 3

### Sistemas cliente-servidor

A arquitetura cliente-servidor não deixa de fazer parte de uma arquitetura centralizada, com o processamento das transações centralizados em um servidor único que, dependendo desse *hardware*, pode suportar um extraordinário número de usuários. A diferença é que, nesse caso, o servidor central divide o processamento com o computador do usuário que processa a entrada de requisições, assim como as telas de acesso.

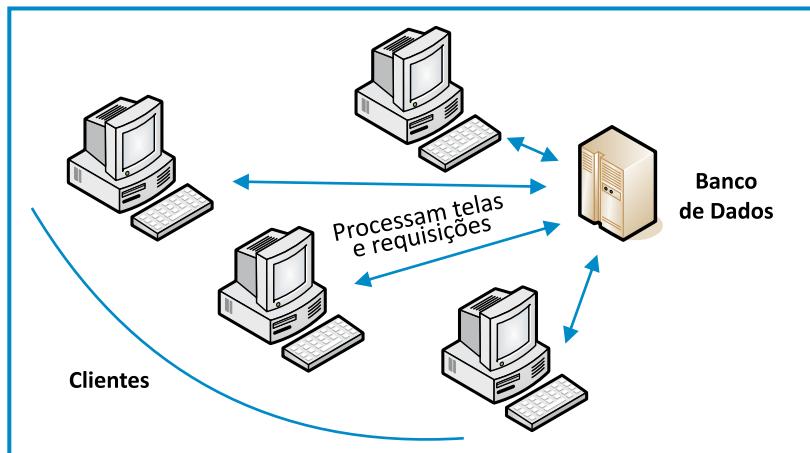


Figura 64: Plataforma cliente-servidor

Essa arquitetura também é usada, majoritariamente, nas grandes empresas, que conseguem justificar facilmente o custo-benefício, inclusive aproveitando o processamento das máquinas de seus usuários.

## SEÇÃO 4

### Sistemas distribuídos

Nessa arquitetura, a informação está distribuída em diversos servidores e cada um deles atua como cliente-servidor, porém as requisições dos aplicativos e usuários são feitas para quaisquer servidores sem diferenças nos resultados.

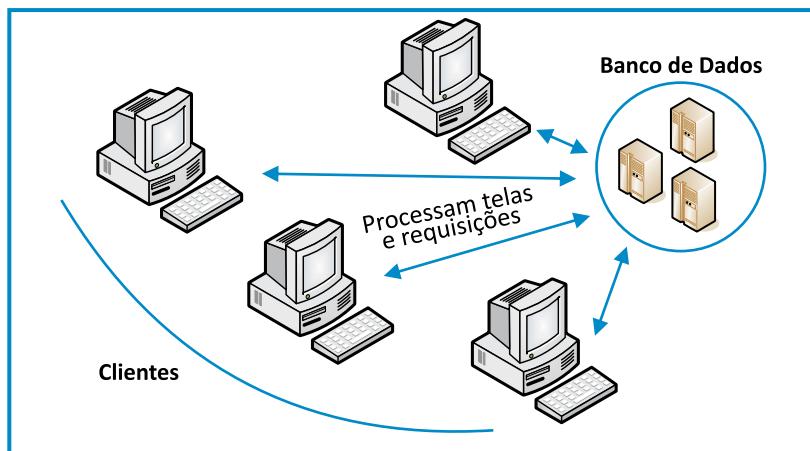


Figura 65: Arquitetura distribuída

Mas, nesse caso, como e em quais servidores são armazenados os dados? Há várias formas de se fazer isso, e uma das premissas para que um sistema distribuído realmente atenda as necessidades é a definição da topologia de rede e a utilização de infraestrutura de comunicação adequada para possibilitar o funcionamento nessa arquitetura.

## ➤ Armazenagem de dados centralizada

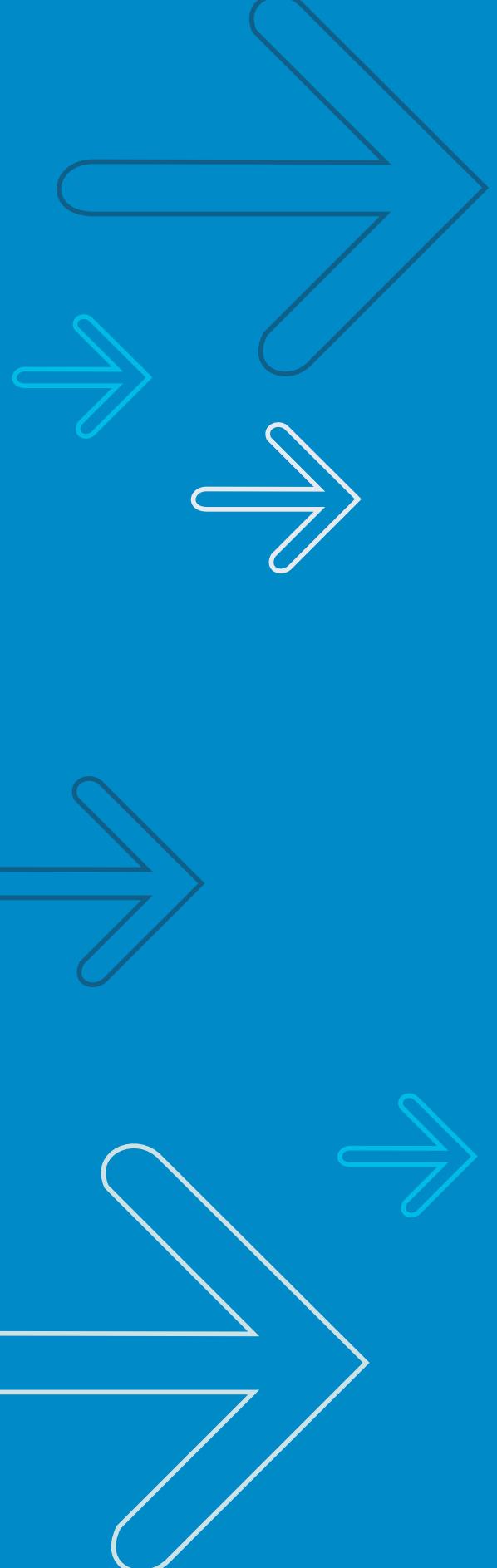
Bom, não parece estranho falar em armazenagem de dados centralizada quando se está explanando um sistema de bancos de dados distribuído? Pode parecer, mas é possível que isso aconteça, já que o conceito de bancos de dados distribuídos representa apenas o recebimento da requisição dos usuários e a execução das transações distribuidamente, mas os dados podem estar em apenas um lugar.

Vários autores abordam esses conceitos de maneiras diferentes, com suas vantagens e desvantagens, porém pode-se afirmar que um armazenamento centralizado de dados não traz alta disponibilidade ao sistema, sendo que, se o banco de dados que está sendo utilizado para armazenar as informações falhar, os dados não são encontrados em outro lugar, tornando-se um ponto de falha. Por esse ponto, é cabível, em grande parte dos sistemas, um armazenamento de dados distribuído.

## ➤ Armazenagem de dados distribuída

Nesse conceito, os dados são replicados de um servidor para todos os outros de forma simultânea, estrutura que garante maior segurança às bases de dados, mas exige um alto grau de investimento em infraestrutura. Considera-se esse tipo de armazenagem mais próprio às empresas que necessitam de alta disponibilidade para executar seus trabalhos, como instituições financeiras, de logística, de manufatura, entre outras.

Mas não basta definir qual será o tipo de armazenagem dos dados. É preciso conhecer algumas ferramentas que dão suporte à tomada de decisão. Quer saber mais? Na próxima unidade você verá!



# Unidade de estudo 17

## Seções de estudo

Seção 1 – *Data warehouse*  
Seção 2 – *Business Intelligence*  
Seção 3 – *Data Mining*  
Seção 4 – Utilização

# Tópicos Especiais – Ferramentas de Suporte à Decisão

## SEÇÃO 1

### *Data warehouse*

Antes de explanar algumas ferramentas de suporte à decisão, é ideal que se tenha uma visão do que é *data warehouse*.

Traduzindo para o português, *data warehouse* significa armazém de dados e prevê o depósito de dados consolidados em um local único. Esses dados contêm o histórico da empresa, como, por exemplo, os clientes que compraram determinado produto em determinado mês, valores, quantidades, formas de pagamento, entre outros dados. Uma *data warehouse* pode armazenar dados constantes em vários bancos de dados espalhados pela corporação, facilitando a geração de consultas e possibilitando suporte à decisão.

Existem várias ferramentas de suporte à decisão que se aproveitam da estrutura de um *data warehouse* para poder gerar as informações necessárias para apoiar os executivos de uma empresa, e duas delas serão abordadas a seguir: o *Business Intelligence* e o *Data Mining*.

## SEÇÃO 2

### *Business Intelligence*

Absorção de conhecimento representa exatamente o que é uma ferramenta de *Business Intelligence* (BI). Você sabe, a partir da leitura do primeiro capítulo deste livro, que, de um banco de dados, é possível extrair dados, e que com eles é possível visualizar as informações necessárias para a tomada de decisão de certa empresa.

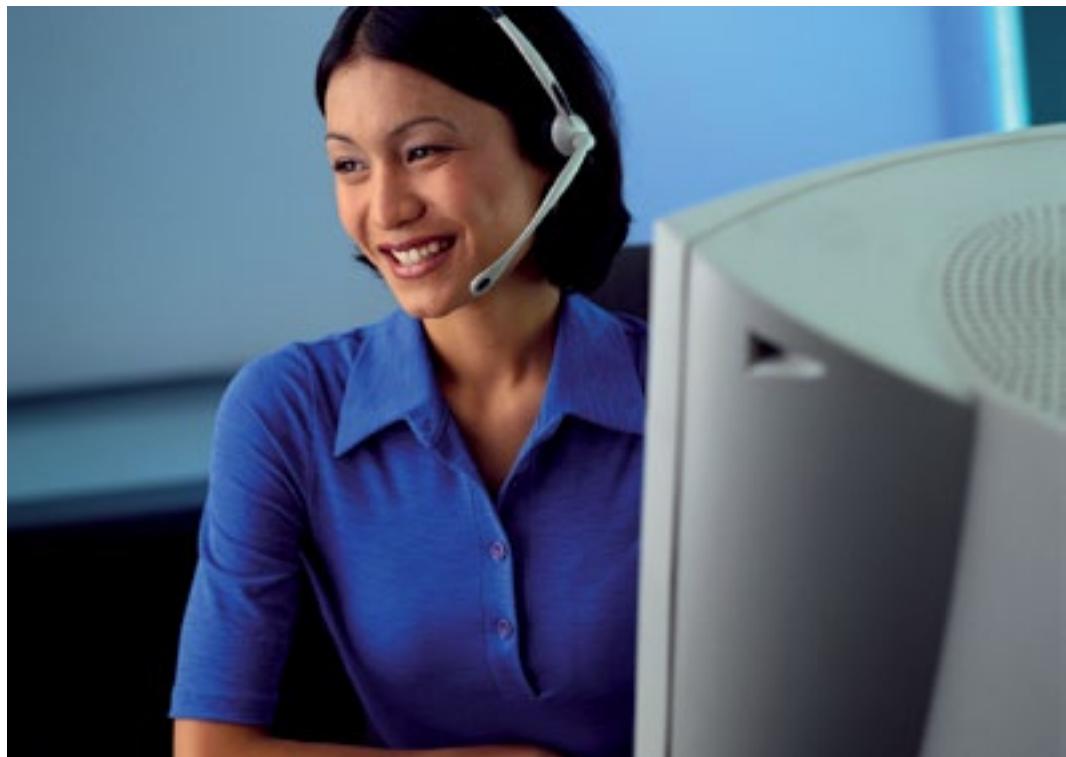
O BI une essas informações em formato dinâmico, gerencial e reproduz a realidade de negócios da corporação. Com uma ferramenta de BI implantada, é possível, por exemplo, observar a evolução de seu faturamento ao longo dos meses, a quantidade de devoluções de mercadorias, seu lucro mensal ou a posição de estoque durante todos os dias do mês corrente.

## SEÇÃO 3

### *Data mining*

*Data mining* significa mineração ou garimpo de dados que se referem à busca de informações relevantes ou “descoberta de conhecimento” a partir de um grande volume de dados armazenados, possivelmente também em um *data warehouse*. Esse tipo de extração de dados tenta encontrar padrões estatísticos, regras e modelos a partir dos dados armazenados e talvez seja esse um dos maiores motivos para a busca cada vez maior da normalização em banco de dados atualmente.

Você já percebeu que, em diversas lojas, televendas, supermercados, entre outros estabelecimentos, ao realizar o seu cadastro, lhe perguntam sua data de nascimento, seu grau de instrução, estado civil e diversas outras informações pessoais, não é mesmo? Com essas informações, ferramentas de *data mining* permitem traçar o perfil do cliente daquele estabelecimento para se determinarem modelos ou regras entre os mais diversos consumidores para, a partir daí, poder criar produtos que melhor atendam determinados perfis de clientes.



## SEÇÃO 4

### Utilização

O qualitativo dessas informações reunidas em formato *on-line* e dinâmico traz um grande conhecimento de negócios ao usuário, que pode, por sua vez, utilizar sua sabedoria e competência para tomar a decisão correta quanto ao rumo da empresa.



Figura 66: Extração da informação

A utilização de ferramentas de BI e *data mining* está conquistando cada vez mais os grandes executivos e, para que seja possível manter esse crescimento, é necessário que um dos pré-requisitos básicos para o bom funcionamento – o banco de dados – seja cada vez mais robusto e performático, utilizando suas mais diversas arquiteturas para suportar as particularidades de cada corporação.

# Finalizando

Com o estudo desta unidade curricular, buscou-se ampliar seus conhecimentos a respeito dos principais conceitos e práticas de bancos de dados, procurando um entendimento com base em fundamentação teórica e a construção de um banco de dados com base nas modelagens e instruções práticas.

Você pôde contemplar toda a teoria em torno dos SGBDs e exercitar a modelagem de dados confeccionando uma base de conhecimentos para poder projetar bancos de dados consistentes de acordo com as melhores práticas utilizadas atualmente. Tudo começou com a compreensão dos conceitos de bancos de dados, SGBDs e modelos de dados, a fim de proporcionar aprendizado básico para, em seguida, praticar a modelagem, criar o banco e utilizar os comandos básicos de SQL.

Os estudos também contemplaram a utilização de ferramentas gratuitas, que atualmente estão em ascendência no mercado, e você as utilizou para modelar e editar um banco de dados de acordo com os requisitos e regras de negócio capturados de uma empresa fictícia. Conheceu e utilizou a linguagem SQL e entendeu os conceitos de armazenagem de dados, infraestrutura de implantação de bancos de dados e formas de extração de informações gerenciais de suporte à decisão.

Caso algo tenha ficado para trás, volte aos estudos e aperfeiçoe seus conhecimentos buscando outras fontes de informação. Lembre-se sempre de que as mudanças nessa área são constantes, por isso é importante que você continue seus estudos sobre o tema, prezando sempre pela busca de melhorias das práticas aplicadas às empresas.

Sucesso para você, e até uma próxima!



# Referências

- ABREU, Mauricio P.; MACHADO, Felipe N. R. **Projeto de banco de dados:** uma visão prática. 11. ed. São Paulo, SP: Ética, 2004.
- ARAÚJO, Márcio S. V.; MATOS, Katherine H. O. **Gestão de projetos.** Florianópolis: 3 maio, 2010.
- CARVALHO, Carlos E. **Testes de software.** Florianópolis: no prelo.
- BARBOSA, Eudes; INGRID, Anne; LABORDE, Gregory. **História dos bancos de dados:** a necessidade de armazenamento. Rio de Janeiro, [2010]. Disponível em: <[www.scribd.com/doc/17324554/HISTORIA-DOS-Bancos-de-dados](http://www.scribd.com/doc/17324554/HISTORIA-DOS-Bancos-de-dados)>. Acesso em: 26 out. 2010.
- CANTO, Cleunisse R. L.; MEDEIROS, Marco A.; VALDO, Clayton A. **Banco de dados.** [Joinville]: [s.n.], fev. 1997. 43 f. (Apostila da disciplina de banco de dados, curso técnico em informática, Escola Técnica Tupy).
- EASYPHP. **EasyPHP.** [S.I], 2010. Disponível em:<[www.easyphp.org](http://www.easyphp.org)>. Acesso em: 7 nov. 2010.
- FERREIRA, João E; ITALIANO, Isabel C.; TAKAI, Osvaldo K. **Introdução a banco de dados.** São Paulo, 2005. 124 f. (Apostila de Introdução a banco de dados). Disponível em: <[www.apostilando.com/download.php?cod=3131&categoria=Banco%20de%20Dados](http://www.apostilando.com/download.php?cod=3131&categoria=Banco%20de%20Dados)>. Acesso em: 25 out. 2010.
- HEUSER, Carlos A. **Projeto de banco de dados.** Porto Alegre: Sagra Luzzatto, 2004.
- MYSQL. **MySQL:** The world's most popular open source database. [S.I.], 2010a. Disponível em: <[www.mysql.com/products](http://www.mysql.com/products)>. Acesso em: 2 nov. 2010.
- \_\_\_\_\_. 2010b. Disponível em: <[www.mysql.com/products/workbench](http://www.mysql.com/products/workbench)>. Acesso em: 14 nov. 2010.
- OLIVEIRA, José P.M.; SANTOS, Clesio S.S. **A informática nos anos 90:** alguns avanços e desafios. Porto Alegre, [1998?]. (Universidade Federal do Rio Grande do Sul, instituto de informática). Disponível em: <[www.buscalegis.ufsc.br/revistas/index.php/buscalegis/article/viewFile/5681/5250](http://www.buscalegis.ufsc.br/revistas/index.php/buscalegis/article/viewFile/5681/5250)>. Acesso em: 28 out. 2010.
- PROJECT MANAGEMENT INSTITUTE. **Project Management body of knowledge:** um em gerenciamento de projetos. 4. ed. Newtown Square, PA, 2008.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados.** 3. ed. São Paulo, SP: Makron Books, c1999. xxii.
- SCHAEFER, Roberto. **Inmersión.** [S.I.], maio. 2010. Disponível em: <[www.inmersion.com.br/Archive/TI/Artigos%20e%20Palestras/Palestra\\_Ferramenta\\_CASE.pdf](http://www.inmersion.com.br/Archive/TI/Artigos%20e%20Palestras/Palestra_Ferramenta_CASE.pdf)>. Acesso em: 13 nov. 2010.
- SILVA, Luciano C. **Banco de dados para web:** do planejamento à implementação. São Paulo: Érica, 2001.

- UNIVERSIDADE DE MARÍLIA. **Banco de dados**. Marília, [s.d.]. 49 f. Disponível em: <[www.apostilando.com/download.php?cod=2837&categoria=Banco%20de%20Dados](http://www.apostilando.com/download.php?cod=2837&categoria=Banco%20de%20Dados)>. Acesso em: 27 out. 2010.
- VIDAL, Vânia M.P. **Banco de dados relacional-objeto**. [Fortaleza], [200-?]. (Universidade Federal do Ceará). Disponível em: <<http://disciplinas.lia.ufc.br/bdnc061/arquivos/partel.ppt>>. Acesso em: 30 out. 2010.

## **Equipe de Desenvolvimento de Recursos Didáticos**

**Coordenação de Educação a Distância**  
Beth Schirmer

**Coordenação Projetos EaD**  
Maristela de Lourdes Alves

**Coordenação de Desenvolvimento de Recursos Didáticos**  
Gisele Umbelino

**Projeto Educacional**  
Angela Maria Mendes  
Israel Braglia

**Projeto Gráfico**  
Daniela de Oliveira Costa  
Jordana Paula Schulka  
Juliana Vieira de Lima

.....

**Design Educacional**  
Evelin Lediani Bao

**Capa, Ilustrações, Tratamento de Imagens**  
D'imitre Camargo Martins  
Diego Fernandes  
Luiz Eduardo Meneghel

**Diagramação**  
Flavia Akemi Ito

**Revisão e Fechamento de Arquivos**  
Daniela de Oliveira Costa  
Juliana Vieira de Lima

**Revisão Ortográfica e Normatização**  
FabriCO