

**EAD FURB**

# **BANCO DE DADOS**

**ALEXANDER ROBERTO VALDAMERI**

**LIVRO DA DISCIPLINA**

**CICLO 1**



# **Banco de Dados**



**Reitor**

profa. Ma. Marcia Cristina Sardá Espíndola

**Vice-Reitor**

prof. Dr. João Luiz Gurgel Calvet da Silveira

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

Profª Drª Romeu Hausmann

**Pró-Reitor de Administração**

Prof. Jamis Antônio Piazza

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

Prof. Dr. Oklinger Mantovaneli Junior

**Professor Autor**

Alexander Roberto Valdameri

**Ambiente Virtual de Aprendizagem**

Guilherme Legal de Oliveira

**Design Instrucional**

Clarissa Josgrilberg Pereira

Marcia Luci da Costa

**Revisão Textual**

Odair José Albino

**Monitoria**

Cibele Bohn

**Produção de Mídia**

Gerson Souza

**Diagramação**

Bárbara Marciniak

**Design Gráfico**

Amanda Ventura de Oliveira

Bianca Klegin Borges


João Pedro Roncalio

Vinícius Bretzke Amaral

Vitor Hugo Miranda

# Ícones

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

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

## Apresentação do Autor

---

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

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

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

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



# Sumário

---

Apresentação do Autor.....	5
Apresentação da Disciplina .....	7
1. Introdução ao Banco de Dados .....	8
1.1 Conceitos e evolução dos bancos de dados .....	9
1.1.1 A origem dos bancos de dados.....	9
1.1.2 Evolução dos bancos de dados .....	11
1.1.3 Arquiteturas de bancos de dados .....	13
1.2 Abordagens de bancos de dados .....	18
1.2.1 Modelo de banco de dados hierárquico .....	18
1.2.2 Modelo de banco de dados em rede .....	20
1.2.3 Modelo de banco de dados relacional .....	22
1.2.4 Modelo de banco de dados orientado a objetos .....	25
1.2.5 Modelo de banco de dados pós-relacional .....	27
1.3 Arquitetura de um sistema gerenciador de banco de dados relacional (SGBDR) ...	30
1.3.1 Processamento de arquivos versus SGBDR .....	30
1.3.2 Funções de um SGBDR .....	33
1.3.3 Arquitetura de um SGBDR .....	38
1.4 Interfaces de SGBD relacionais existentes .....	43
Conclusão .....	45
Referências .....	47

# **Apresentação da Disciplina**

---

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

## **Ciclo 1**

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

## **Ciclo 2**

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

## **Ciclo 3**

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

Bons estudos!

# 1. Introdução ao Banco de Dados

Estamos iniciando a primeira etapa do nosso percurso. Neste primeiro Ciclo você terá a oportunidade de aprender o que é um banco de dados, a diferença entre dado e informação, por que os sistemas computacionais utilizam bancos de dados, um breve relato histórico sobre a evolução, as abordagens de bancos de dados existentes e as arquiteturas onde encontramos os bancos de dados.

A resposta é complexa, mas vamos lá! Pretendemos ajudá-lo(a) a entender as razões trazendo à tona nossa experiência de formação em relação a bancos de dados. Na década de 90, quando tivemos contato pela primeira vez com o assunto, o momento era outro, ou seja, o que se discute hoje era improvável, sequer se cogitava existir. Mesmo assim, tivemos a oportunidade de conhecer um pouco da evolução e, conseqüentemente, pudemos avaliar os fatores que promoveram as melhorias e, muitas vezes, a influência dos paradigmas vigentes à época.

Hoje, felizmente por conta desta bagagem conceitual que tivemos durante a formação acadêmica, conseguimos compreender as diferentes realidades e demandas de cada geração por que os bancos de dados passaram. Assim, sentimo-nos em condições de avaliar de forma mais eficiente as novas realidades emergentes. E é exatamente isto que esperamos oferecer a você: informações e relatos de vivências que possam ajudá-lo(a) na sua trajetória profissional.

O objetivo ao término deste Ciclo é que você conheça os conceitos relacionados a bancos de dados, e compreenda a evolução e os cenários nos quais eles são aplicados. Esta etapa é fundamental para um melhor aproveitamento dos próximos ciclos. Preparado(a)? Então vamos em frente.



você deve estar se perguntando: por que preciso conhecer os fundamentos teóricos e históricos? Não podemos ir direto para a parte prática dos bancos de dados?



## 1.1 Conceitos e evolução dos bancos de dados

### 1.1.1 A origem dos bancos de dados

Antes de trabalharmos o conceito, é preciso entender como surgiram os “bancos de dados”. Há 30 anos o termo utilizado era “processamento de dados”, isto é, o uso de computadores num contexto de negócios. Isso significa que o foco era o processamento de grandes quantidades de dados. Hoje o termo “processamento de dados” deu lugar a “tecnologia da informação”, pois, além da preocupação convencional com o armazenamento dos dados, outras perspectivas foram criadas, como a mineração de dados por meio dos sistemas de apoio à decisão.

Nosso objetivo agora não é discutir a evolução dos sistemas de informação propriamente ditos. Nosso foco está no armazenamento dos dados. Imagine o cenário informatizado de uma instituição de ensino superior, como a nossa, sem o uso de bancos de dados. Para isso, considere o seguinte cenário apresentado na Figura 1:



É possível afirmar que os termos significam a mesma coisa. Mas é necessário atentar para um deslocamento sutil de foco sobre os dados.

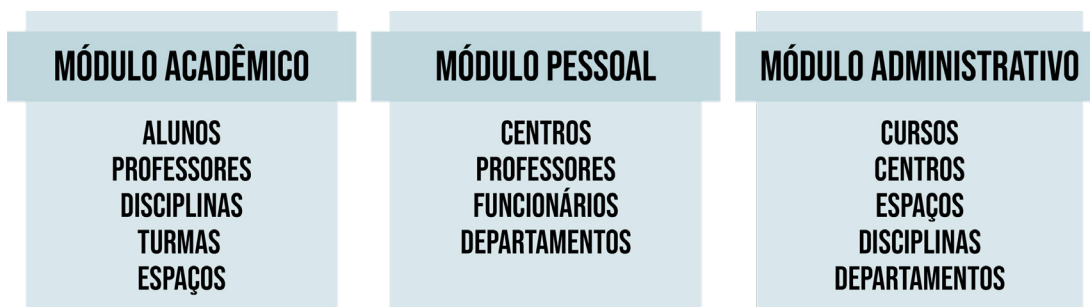
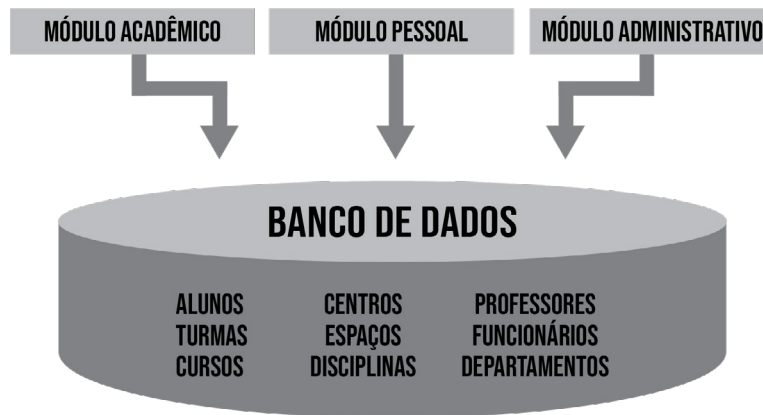


Figura 1: Cenário de módulos informatizados. Fonte: DME/FURB (2019) baseado no autor.

Nesse exemplo, podemos identificar que os módulos manipulam dados que poderiam ser compartilhados, como, por exemplo, professores, departamentos, disciplinas, espaços e centros. Porém foram projetados para atender a uma aplicação específica. Como consequência, temos problemas de redundância nos dados, de integridade e de segurança, entre muitos outros.

Considerando o mesmo cenário, observe na Figura 2 como atualmente são construídos os sistemas, tendo como recurso para armazenamento um banco de dados:



Você consegue imaginar vantagens e/ou desvantagens da utilização dos bancos de dados?

Figura 2: Sistemas construídos com um banco de dados. Fonte: DME/FURB (2019) baseado no autor.

Aqui podemos perceber que os diferentes módulos compartilham dados através de um repositório único. Desta forma não há redundância dos dados e os dados inseridos ou alterados por quaisquer um dos módulos são percebidos pelos demais. Nesse momento já é possível imaginar as vantagens na utilização de um banco de dados.

Mas por que utilizar a expressão “banco de dados”, e não “banco de informações”? Por vezes podemos entender um dado como sendo uma informação, e uma informação como sendo um dado. Por exemplo, ao responder à pergunta “que dia é hoje?”, sua resposta seria um dado ou uma informação? Ou seja, em alguns cenários e/ou contextos, “dado” e “informação” dão o mesmo significado ao fato em si. Mas no contexto da tecnologia da informação e da comunicação eles se apresentam como termos distintos.

E afinal, o que é dado e o que é informação? Segundo Date (2004), dado é o elemento de partida que serve de base para o tratamento sobre o qual o computador efetua as operações necessárias à tarefa a que foi programado. Podemos dizer que o dado é a representação primária de um fato, conceito ou instrução. Portanto, não representa uma informação ao indivíduo. Já a informação, conforme Date (2004), é um conjunto de dados processados e organizados de tal forma que representam um significado de um fato ou conceito. Pode-se dizer que é uma abstração informal, isto é, que não pode ser formalizada através de uma teoria lógica ou matemática.

Em síntese, podemos afirmar que o dado é a representação bruta da informação. Em geral, o dado é isolado. Quando um conjunto de dados é organizado conforme padrões pré-estabelecidos, obtemos a informação sobre os quais justifica a existência dos dados. Há também o termo conhecimento, mas este discutiremos em outra oportunidade.

De acordo com Elmasri e Navathe (2012), organizar e manipular dados em planilhas eletrônicas, arquivos criados por aplicativos específicos ou até mesmo editores de texto são exemplos que nos remetem ao conceito de banco de dados. Para os autores, a afirmação é justificada, visto que os dados estão estruturados e seguem padrões

de organização que permitem o seu armazenamento, recuperação e manipulação. Porém, a evolução das tecnologias aliada à necessidade das empresas fez com que outras características e funcionalidades dos bancos de dados também evoluíssem nos dias atuais. Quando se pesquisa e se discute sobre bancos de dados, logo temos uma avalanche de informações que nos remetem a estruturas bem elaboradas e em geral associadas a um conjunto de recursos avançados que compõem a estrutura da grande maioria dos sistemas informatizados.

Até o momento já abordamos os conceitos de dado e de informação... Mas afinal, o que é um banco de dados? O banco de dados pode ser definido como um conjunto de dados organizados que representam informações sobre um domínio específico (DATE, 2004).

A seguir vamos conhecer um pouco sobre a evolução dos bancos de dados. Esta etapa é importante para que possamos compreender as razões pelas quais algumas características estão presentes nas atuais soluções mercadológicas e, principalmente, traçar um paralelo com as demandas emergentes dos sistemas altamente conectados e distribuídos.

### 1.1.2 Evolução dos bancos de dados

A síntese histórica da evolução dos bancos de dados está baseada na obra de Ramakrishnan e Gehrke (2008). Para os autores, é possível classificar a evolução das estruturas de bancos de dados e o gerenciamento sobre os dados em **três gerações**. Cada uma delas tem suas particularidades, mas todas têm em comum a necessidade cada vez maior de manter os dados e controlar as operações sobre eles.

A **1.ª geração** das estruturas de bancos de dados é marcada pelo armazenamento dos dados totalmente dependente do programa ou aplicação que os controla. Vejamos uma representação gráfica (Figura 3) desse cenário:

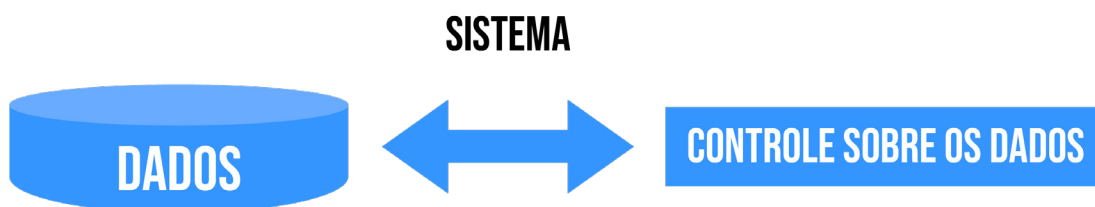


Figura 3: Sistemas e dados sob a mesma estrutura. Fonte: DME/FURB (2019) baseado no autor.

Pode-se observar que o sistema é responsável pelo armazenamento e controle sobre os dados, ou seja, os dados são acessíveis somente por esse sistema ou aplicação.

Imagine que você está construindo uma aplicação utilizando a linguagem de programação C para controlar a agenda de contatos de seus amigos. Será necessário armazenar esses dados em um arquivo. Então você vai criar uma estrutura de registro

(*record* ou *struct*) e associá-la a um arquivo. Este arquivo binário que será criado vai armazenar os registros definidos pela sua estrutura.

Agora imagine um aplicativo que você está construindo em Java. Ele vai controlar suas despesas ao longo de um determinado período. Você poderá, por exemplo, definir uma classe “despesa” com os respectivos atributos que julgar necessários para o seu propósito. Utilizando uma interface (`java.io.Serializable`), você poderá salvar uma instância dessa classe, ou seja, um objeto em um arquivo.

Já na 2.<sup>a</sup> geração das estruturas de bancos de dados, temos os dados armazenados de forma independente das estruturas que os controlam. Observe a representação gráfica desse cenário na Figura 4:

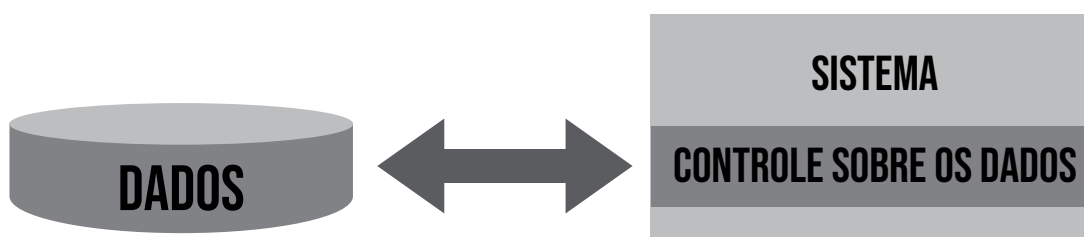


Figura 4: Sistema controlando dados externamente. Fonte: DME/FURB (2019) baseado no autor.

Observa-se que o sistema é responsável pelo controle sobre os dados. Já a estrutura de armazenamento é independente do sistema, ou seja, os dados podem ser acessados por outros sistemas ou aplicações.

Você já deve ter ouvido falar nas estruturas padrão DBF (Data Base File) ou simplesmente DBase. Essas estruturas marcaram esta geração em termos de armazenamento. A organização dos dados é simples e pode ser acessada por diferentes ambientes de desenvolvimento, assim como aplicações de controle e manutenção. Para se ter uma ideia, aplicativos que mantêm planilhas eletrônicas permitem o armazenamento e a recuperação de dados nessas estruturas.

Outro exemplo usual nos dias atuais se refere à persistência de dados em estrutura de arquivo padrão XML (Extensible Markup Language). O padrão XML é um formato para a criação de documentos com dados organizados de forma hierárquica, o que geralmente contempla as necessidades das aplicações. Os arquivos XML podem ser acessados por inúmeras linguagens de programação, bem como por aplicativos de leitura e manipulação.

A 3.<sup>a</sup> geração das estruturas de bancos de dados é atualmente a mais utilizada. Nela temos a inserção do Sistema Gerenciador de Bancos de Dados (SGBD). O SGBD é o responsável pelo acesso e manutenção dos dados armazenados, ou seja, o sistema se comunica com o SGBD e este interage com os dados. Vejamos a representação gráfica desse cenário através da Figura 5:



Quer saber mais sobre serialização de dados na linguagem Java? Consulte o manual da linguagem clicando no QRcode abaixo:



Saiba mais sobre a linguagem XML e seus padrões neste QRcode:





Figura 5: Gestão dos dados independente do sistema. Fonte: DME/FURB (2019) baseado no autor.

São alguns exemplos de SGBDs: ORACLE, Microsoft SQL Server, PostgreSQL e MySQL. Todos com muitas características em comum, estrutura complexa, mas com acesso simplificado por meio das interfaces de comunicação que proporcionam sua utilização por diferentes ambientes de desenvolvimento, assim como aplicações de controle e manutenção. Para se ter uma ideia, é raro você encontrar um ambiente de desenvolvimento ou linguagem de programação de sistemas que não ofereça recursos de acesso e manutenção de estruturas e dados mantidos nos principais SGBDs do mercado.

Os SGBDs compreendem um componente fundamental na arquitetura de desenvolvimento de aplicações em camadas.

Nos dias atuais os sistemas estão cada vez mais flexíveis, conectando tudo a todos (SILVA, 2001). Isso tem exigido que os sistemas e aplicativos façam uso de estruturas de armazenamento híbridas. Abaixo vemos na Figura 6 uma representação gráfica desse cenário.



Figura 6: Sistema com gestão de dados flexível. Fonte: DME/FURB (2019) baseado no autor.

Chegamos ao final da seção que trouxe, de forma sucinta e objetiva, aspectos que envolveram conceitos e a evolução histórica dos bancos de dados. A seguir convidamos você a conhecer um pouco sobre as arquiteturas de bancos de dados existentes.

### 1.1.3 Arquiteturas de bancos de dados

A expressão “arquitetura de bancos de dados” refere-se à forma como o banco de dados está sendo colocado em um contexto amplo, associado a outras tecnologias de informação e comunicação (GARCIA-MOLINA, 2001). Nesse caso, vamos considerar a forma como o banco de dados está sendo usado pelos sistemas. As seguintes arquiteturas serão apresentadas e discutidas:

- centralizada
- cliente-servidor



Neste momento não vamos nos aprofundar no "desenvolvimento de aplicações em camadas". Mas se você quiser conhecer um pouco mais sobre o assunto acesse este QRcode:



Não podemos afirmar que a utilização de formas variadas de armazenamento representa uma nova geração para estruturas de bancos de dados. Afinal, o que vemos claramente é uma junção de estruturas já consolidadas, não é mesmo?

- distribuída
- paralela

A arquitetura centralizada apresenta as seguintes características:

- A utilização de *mainframes* para executar o processamento principal e de todas as funções do sistema.
- Terminais remotos, ligados por redes de comunicação, que não necessitam de poder de processamento, apenas da capacidade de entrada de dados e visualização.
- Custo elevado em razão do valor do equipamento de grande porte (*mainframe*).

Na arquitetura de banco de dados centralizada temos um computador com grande capacidade de processamento. Este computador hospeda o sistema operacional, o banco de dados e a aplicação. A principal vantagem desta arquitetura é a possibilidade de haver muitos usuários manipulando grandes volumes de dados, visto que seu poder de processamento é elevado. Sua principal desvantagem está no seu alto custo, pois exige um ambiente especial para *mainframe*; e soluções centralizadas, em geral, são de elevado custo.

A Figura 7 ilustra esta representação.

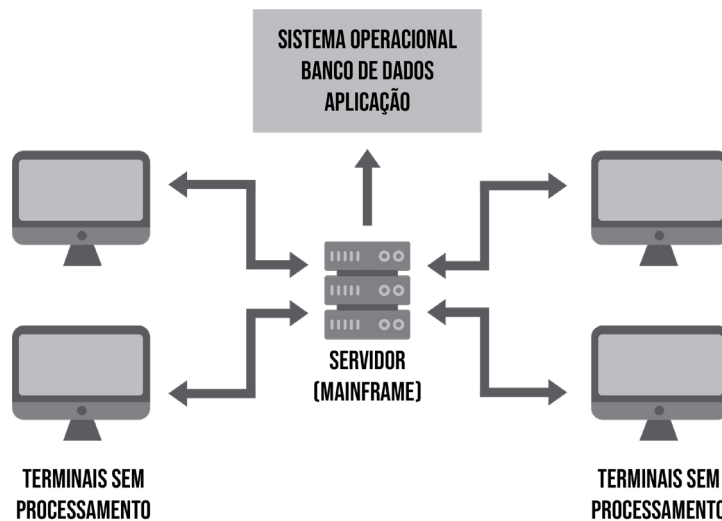


Figura 7: Arquitetura de banco de dados centralizada. Fonte: DME/FURB (2019) baseado no autor.

A arquitetura cliente-servidor apresenta as seguintes características:

- O cliente executa as tarefas do aplicativo (tela e processamento de entrada e saída); o servidor executa as consultas no banco de dados e retorna os resultados ao cliente – nesta arquitetura é comum encontrarmos os SGBDs.
- O processamento é dividido entre o cliente e o servidor, reduzindo-se o tráfego de rede.

A arquitetura cliente-servidor é a mais comum nos dias atuais. Nela encontramos uma interface do usuário que executa as tarefas por meio de um aplicativo ou sistema específico e o servidor, que tem como função executar as consultas no sistema gerenciador de banco de dados e retornar os resultados ao cliente. Apesar de ser uma arquitetura bastante popular, são necessárias soluções sofisticadas e só quem garante isto é o sistema gerenciador de banco de dados, o SGBD. A principal vantagem desta arquitetura é a divisão do processamento entre dois sistemas.

A Figura 8 ilustra a representação gráfica desta arquitetura.

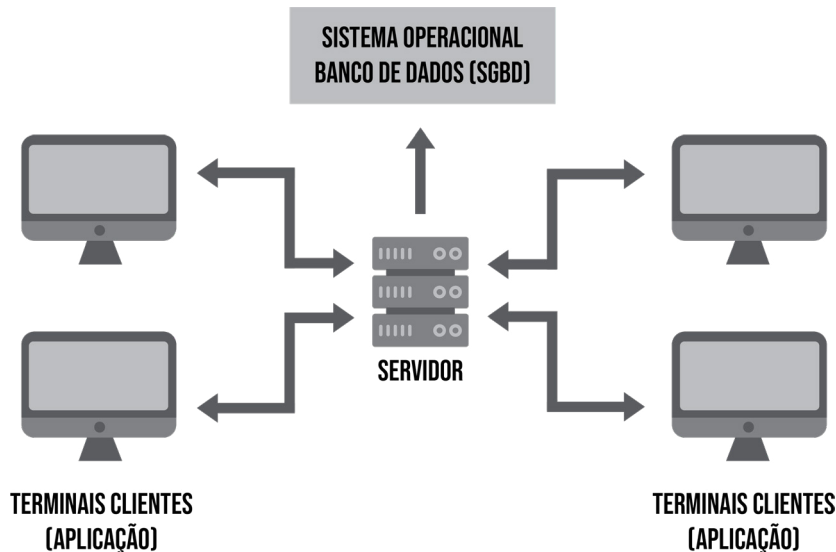


Figura 8: Arquitetura de banco de dados cliente-servidor. Fonte: DME/FURB (2019) baseado no autor.

A arquitetura distribuída, menos comum nos sistemas existentes, apresenta as seguintes características:

- A informação está distribuída em diversos servidores, o que exige um completo sistema de controle das operações.
- Cada servidor atua como no sistema cliente-servidor, porém as consultas dos aplicativos são feitas para qualquer servidor indistintamente.
- O sistema encarrega-se de obter a informação de maneira transparente para o aplicativo.

A proposta da arquitetura de banco de dados distribuídos propõe que os dados estejam segmentados em dois ou mais servidores. Neste caso, cada servidor exerce o papel de um sistema cliente-servidor, onde as consultas oriundas dos clientes podem ser atendidas por qualquer um dos servidores, indistintamente. É importante destacar que o sistema se encarrega de obter a informação necessária de maneira transparente para o cliente que, naturalmente, desconhecer a existência da estrutura de múltiplos

servidores. São exemplos as bases de dados corporativas em que o volume de informação é muito grande e por isso deve ser distribuído em diversos servidores.

A Figura 9 expõe esta arquitetura.

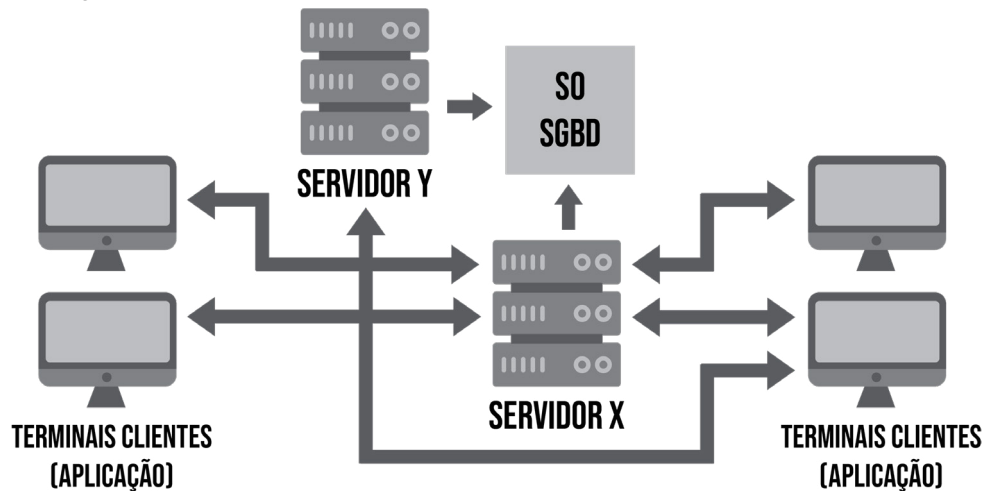


Figura 9: Arquitetura de banco de dados distribuída. Fonte: DME/FURB (2019) baseado no autor.

Por fim, temos a arquitetura paralela, para a qual destacamos as seguintes características:

- A informação está distribuída em diversos servidores fortemente acoplados, constituindo um único sistema de banco de dados.
- O objetivo é aumentar os recursos de processamento e armazenamento.
- É a solução utilizada para o aumento de escala, quando maiores cargas de trabalho são necessárias, sem aumento no tempo de resposta.

A arquitetura de banco de dados paralela combina técnicas de gerência de dados e processamento paralelo para aumentar o desempenho e a confiabilidade. Há menos de dez anos isto parecia algo impossível, pois havia muitas restrições dos mecanismos de comunicação. Hoje soluções envolvendo um banco de dados altamente paralelos estão começando a substituir os tradicionais *mainframes* para processamento de base de dados e transações. Em síntese, podemos dizer que dois ou mais equipamentos trabalham em conjunto para atender às necessidades dos usuários de bancos de dados. E como isso é feito? Não é algo trivial, mas imagine que os dados são particionados entre os servidores e o controle é realizado pelo SGBD. É importante destacar que um sistema que utiliza dados de bancos paralelos é uma resposta para a necessidade de implementação de grandes bases de dados com grande número de acessos sem a necessidade de máquinas com um grande poder computacional ou de memória.

Observe a Figura 10 para conhecer a representação gráfica desta arquitetura.



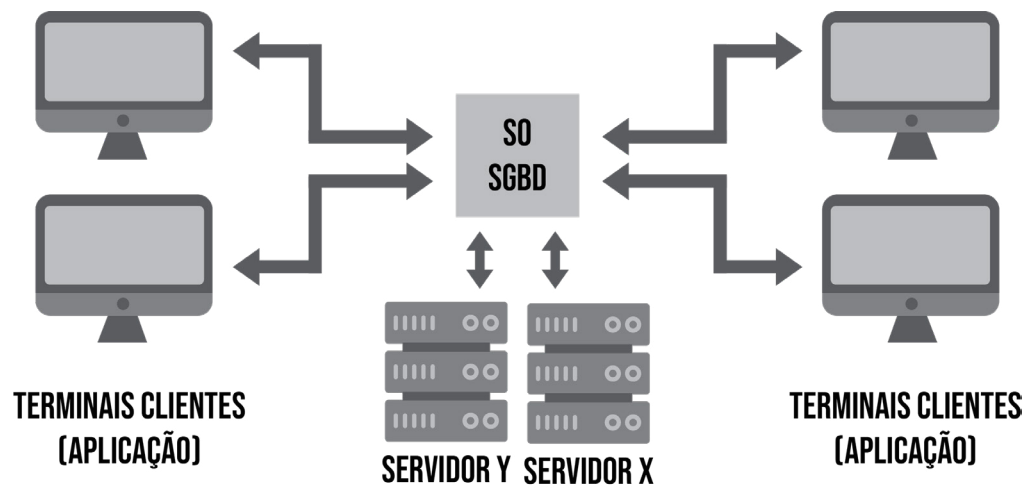


Figura 10: Arquitetura de banco de dados paralela. Fonte: DME/FURB (2019) baseado no autor.

A seguir vamos apresentar e discutir um pouco as abordagens de bancos de dados existentes.

## 1.2 Abordagens de bancos de dados

Antes de detalhar os SGBDs, é importante conhecer as abordagens de bancos de dados existentes. Algumas literaturas falam também em modelos de bancos de dados. As duas formas estão corretas, pois são sinônimos, mas nós utilizaremos o termo “modelo” ao nos referirmos às diferentes formas de estruturação e gerenciamento dos dados mantidos. Como dito, os modelos nos mostram como os dados são organizados e representados, além de mostrar características específicas de armazenamento e recuperação. A escolha do modelo é determinante para o processo de armazenamento dos dados de um sistema de informação.

Mas antes de conhecermos os modelos existentes, precisamos tratar de alguns conceitos preliminares. Vamos começar pelo termo registro, muito comum no contexto dos bancos de dados. Um registro é um conjunto de atributos organizados que representam dados. Um atributo é a menor porção do registro, isto é, constitui uma parte do conjunto de dados armazenado.

Imagine a seguinte estrutura que mantém dados de clientes (Figura 11):



Figura 11: Exemplo de registro. Fonte: DME/FURB (2019) baseado no autor.

No decorrer do conteúdo será possível perceber que boa parte dos modelos de bancos de dados faz uso dessas estruturas para organizar os dados. Porém, esta é apenas uma das características comuns entre as que iremos conhecer em seguida. Existem outros pontos em comum, assim como significativas diferenças entre eles.

### 1.2.1 Modelo de banco de dados hierárquico

O modelo hierárquico foi o primeiro a ser reconhecido como abordagem de banco de dados, tendo sido o principal modelo até a década de 80 (DATE, 2004). Porém, isso não significa que atualmente ele não é mais utilizado. Há grandes sistemas, alguns considerados legados, cujos dados estão mantidos no modelo hierárquico. Neste modelo de dados, os dados são organizados na forma de registro e estruturados em hierarquias ou árvores. Os nós das hierarquias contêm ocorrências de registros, onde cada registro é uma coleção de atributos. O registro da hierarquia que precede os



Você poderá encontrar na literatura o termo "abordagens" de bancos de dados. Saiba que são sinônimos, isto é, abordagem e modelo correspondem à mesma coisa: a representação básica do armazenamento dos dados.

outros é o registro pai. Os outros são, naturalmente, chamados de registros filho. Uma ligação é uma associação entre dois registros. Um registro pode estar associado a vários registros diferentes, desde que seja replicado.

A representação gráfica do modelo de banco de dados hierárquico pode ser observada na Figura 12:

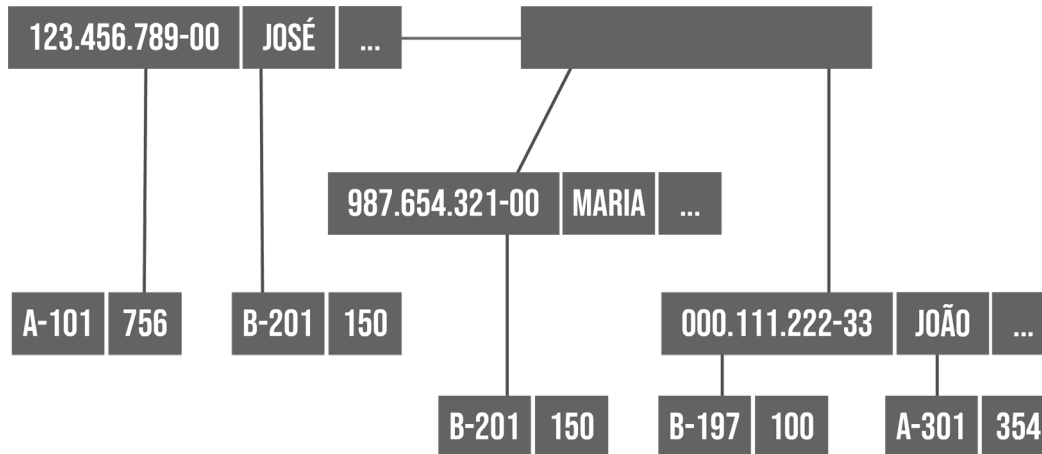


Figura 12: Representação gráfica dos dados no modelo hierárquico.  
Fonte: DME/FURB (2019) baseado no autor.

Ao considerarmos as características apresentadas, podemos concluir que o modelo de banco de dados hierárquico apresenta redundância e inconsistência nos dados, assim como gera desperdício de espaço. Ele ainda torna as buscas complexas, pois seu ponto de partida é o registro principal (nodo raiz da árvore). A Figura 13 ilustra algumas destas afirmações:

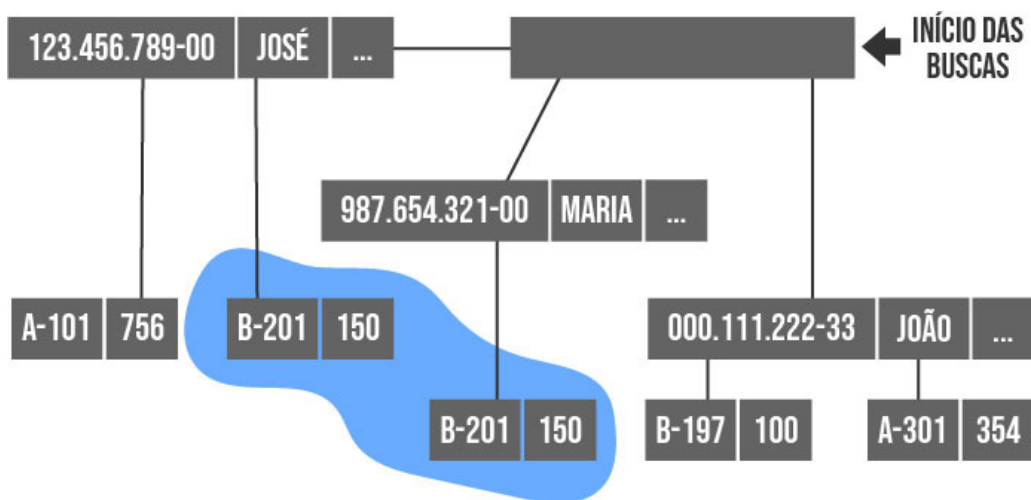


Figura 13: Redundância de dados. Fonte: DME/FURB (2019) baseado no autor.

Conforme você pode observar na representação gráfica, a replicação possui duas grandes desvantagens: pode causar inconsistência de dados quando houver atualização

e o desperdício de espaço é inevitável. Os dados organizados segundo esta abordagem podem ser acessados segundo uma sequência hierárquica com uma navegação do topo para as folhas e da esquerda para a direita, o que a torna um tanto restritiva.

Grande parte das restrições e consistências de dados estão contidas dentro dos programas escritos para as aplicações, o que faz com que estas aplicações tenham que controlar boa parte das operações sobre os dados. A seguir temos um trecho de código na linguagem DLI/IMS, que define as estruturas representadas:

```

schema name = BANCO
Hierarchies = HIERARQUIA1
record name = CLIENTE
    type = RAIZ DE HIERARQUIA1
    data items =
        CPF      integer
        NOME     character(30)
        EMAIL    character(50)

    key = CPF

record name = CONTA
parent = CLIENTE
child NUMBER = 1
data items =
    NRO      integer
    SALDO    float(8,2)

key = NRO

```

Fonte: DME/FURB (2019) baseado no autor.

Acima podemos ter uma ideia de como ficaria a codificação para definir as estruturas representadas. Facilmente você pode perceber que a linguagem, baseada no DLI/IMS do Information Management System, estrutura os dados na forma de dependência dos registros através de níveis hierárquicos. Há uma definição do registro pai e em seguida os registros filho.

Você deve estar se perguntando: e como se dá as ligações entre os registros? As ligações entre os registros se dão através das linguagens de manipulação dos dados, ou seja, no momento em que os dados são inseridos, é informada sua dependência. Aqui devemos registrar também a ausência de uma linguagem padrão para a definição e manipulação dos dados. Mas isto não é o foco neste momento.

Este modelo de banco de dados não contemplou um número significativo de produtos acadêmicos e tampouco comerciais. Alguns dos produtos conhecidos são:

- Information Management System – Empresa IBM
- ADABAS – Empresa Software AG

### 1.2.2 Modelo de banco de dados em rede

Ao estudar e conhecer o modelo de banco de dados hierárquico você teve a oportunidade de identificar algumas fragilidades relacionadas principalmente à estrutura e à manipulação dos dados. É nessa perspectiva que surgiu o modelo de banco de dados em rede.



O ADABAS surgiu na década de 70 e é considerado o primeiro banco de dados comercial. Passou por atualizações ao longo do tempo, por isso está no mercado até hoje.

O modelo em rede surgiu como uma evolução da abordagem hierárquica, visto que propôs a eliminação da necessidade de hierarquia dos dados, possibilitando que o mesmo registro estivesse envolvido em várias associações sem a necessidade da sua replicação, evitando, assim, a redundância. Da mesma forma, como no modelo de banco de dados hierárquico, na abordagem em rede os dados são representados na forma de registros. Também da mesma forma, as associações entre os registros são representadas através de ligações, e a estruturação dos dados se dá na forma de grafo, em vez de árvore, como era no modelo hierárquico.

A representação gráfica do modelo de banco de dados em rede pode ser observada na Figura 14.

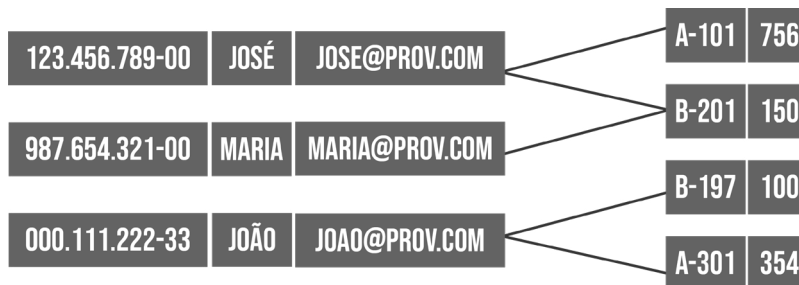


Figura 14: Representação gráfica dos dados no modelo em rede.  
Fonte: DME/FURB (2019) baseado no autor.

O gestor de trabalho na área de banco de dados conhecido como Data Base Task Group (DBTG) da CODASYL (Committee on Data Systems Language), ou seja, comitê dos sistemas de dados e linguagens, estabeleceu uma norma para este modelo, constituindo-o com linguagem própria para a definição e manipulação de dados baseados no Cobol. Ao contrário da abordagem hierárquica, onde qualquer acesso aos dados necessariamente passaria pela raiz, a abordagem em rede possibilitaria o acesso a registros da rede a partir de qualquer posição inicial (nó) de referência e a navegação através das ligações – os *links*.

Apresenta-se abaixo como exemplo o código na linguagem baseada no Cobol que define as estruturas representadas:

```

schema name is BANCO
record name is CLIENTE
    duplicates are not allowed for CPF
    CPF    type is numeric integer
    NOME   type is character(30)
    EMAIL  type is character(50)

record name is CONTA
    duplicates are not allowed for NRO
    NRO    type is numeric integer
    SALDO  type is numeric float
  
```

Fonte: DME/FURB (2019) baseado no autor.

Acima podemos ter uma ideia de como ficaria a codificação para definir as estruturas representadas. Facilmente você pode perceber que a linguagem, baseada no Cobol, controla a redundância nos dados, não permitindo a duplicidade de determinados atributos. As ligações entre os registros se dão através das linguagens de manipulação dos dados, ou seja, no momento em que os dados são inseridos é informada sua ligação através de conexões.

Com a homologação por parte do grupo gestor (CODASYL), o modelo obteve significativa representação em produtos comerciais. Alguns dos produtos conhecidos são:

- CA-IDMS – Empresa Computer Associates
- VAX DBMS – Empresa Digital
- IDS – Empresa Honeywell Inc.
- DMS1100 – Empresa UNIVAC



O CA-IDMS é um dos bancos de dados em rede comerciais mais conhecidos e utilizados atualmente.

### 1.2.3 Modelo de banco de dados relacional

Até o momento foram abordados o modelo de banco de dados hierárquico e o modelo de banco de dados em rede, cada qual com suas particularidades e funcionalidades. Mesmo assim, ambos têm características comuns, como a representação dos dados em forma de registros. Isso os torna modelos orientados a registros, pois qualquer acesso à base de dados para manipulação (consulta, inclusão, alteração ou exclusão) é feito em um registro de cada vez. A partir de agora conheceremos as características do modelo mais utilizado atualmente: o modelo relacional.

Inicialmente vamos compreender a utilização do termo “relacional”. A origem dessa denominação está associada à forma como os dados, também representados em forma de registros, estão ligados uns aos outros (ELMASRI; NAVATHE, 2012). Neste modelo não há ponteiros de ligação (*links*), pois os dados “se relacionam” uns aos outros através de valores próprios. Daí a nomenclatura “modelo relacional”.

No início da década de 70, após longos estudos tendo como base a teoria dos conjuntos e a álgebra relacional, o matemático Edgar Frank Codd apresenta uma proposta teórica para representação de dados e suas relações (DATE, 2004). Segundo o autor, este foi o marco referencial para o que conhecemos hoje como abordagem ou modelo de banco de dados relacional. Um dos objetivos da abordagem relacional foi aumentar a independência dos dados em relação às aplicações, mas para isso era necessária a definição de um conjunto de funções para armazenamento e recuperação dos dados. A estrutura fundamental do modelo relacional é a relação, ou seja, a tabela. Uma relação é constituída por um ou mais atributos, também conhecidos como campos, que traduzem o tipo de dado a armazenar. Cada instância do esquema linha é chamado de “tupla” ou também “registro”.

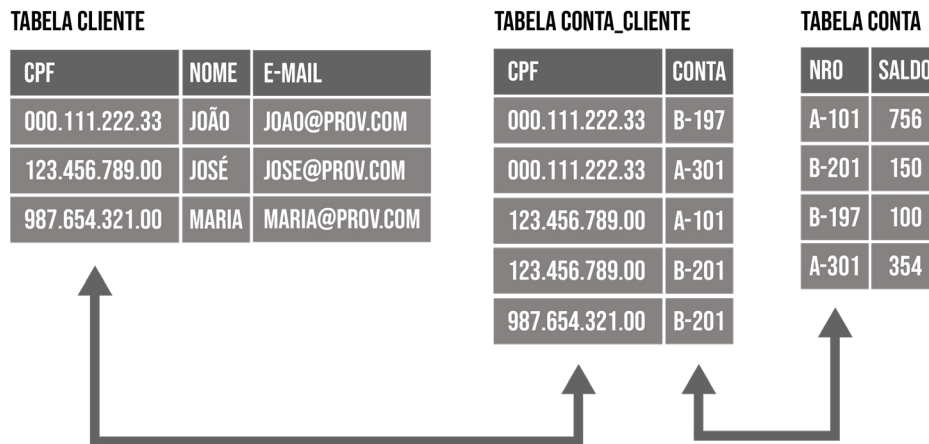


Figura 15: Representação dos dados no modelo relacional.  
Fonte: DME/FURB (2019) baseado no autor.

Quanto à representação gráfica, podemos observar na Figura 15 a tabela cliente e a tabela conta. Observe que CPF é a chave para a relação com a tabela conta, onde NRO é o atributo chave. Considerando que um cliente pode assumir uma ou mais contas, e por sua vez uma conta pode estar ligada a um ou mais clientes, tornou-se necessária a criação de uma estrutura de ligação: outra tabela, para estabelecer a relação entre os dados.

A seguir apresenta-se o código na linguagem SQL que define as estruturas representadas na Figura 15:

```
create database BANCO;
create table CLIENTE (
    CPF integer not null,
    NOME varchar(30),
    EMAIL varchar(50),
    primary key (CPF));

Create table CONTA (
    NRO integer not null,
    SALDO float,
    primary key(NRO));

Create table CLIENTE_CONTA (
    CPF integer not null,
    NRO integer not null,
    primary key (CPF, NRO),
    foreign key (CPF) references CLIENTE(CPF),
    foreign key (NRO) references CONTA(NRO));
```

Fonte: DME/FURB (2019) baseado no autor.

Para Elmasri e Navathe (2012), o modelo relacional se apresenta como a abordagem mais flexível e adequada para o armazenamento de dados nas soluções da grande maioria dos problemas que se colocam no nível da concepção e implementação da base de dados. Além disso, as estruturas do modelo para representação dos dados são consideradas simples e com isso apresentam alta *performance* na manipulação.

O modelo relacional não tem caminhos pré-definidos para se fazer acesso aos dados como nas demais abordagens já apresentadas – hierárquico e rede –,

ou seja, qualquer atributo de qualquer tabela pode ser objeto de recuperação, independentemente de suas relações. A busca e a recuperação, assim como qualquer operação de manipulação, inserção, alteração e exclusão, são realizadas através do conjunto de comandos definidos na linguagem SQL, portanto a SQL é a linguagem padrão para a abordagem relacional.

Como já citado, a abordagem implementa estruturas de dados organizados em tabelas, porém para trabalhar com essas tabelas algumas restrições precisam ser impostas para evitar aspectos indesejáveis, como repetição e perda de informação, por exemplo. Algumas dessas restrições estão relacionadas à integridade referencial e à consistência entre as tabelas, representadas no modelo através das chaves.

Observe a definição das chaves:

```
create database BANCO;
create table CLIENTE (
    CPF        integer not null,
    NOME        varchar(30),
    EMAIL       varchar(50),
    primary key (CPF));

Create table CONTA (
    NRO        integer not null,
    SALDO       float,
    primary key (NRO));

Create table CLIENTE_CONTA (
    CPF        integer not null,
    NRO        integer not null,
    primary key (CPF, NRO),
    foreign key (CPF) references CLIENTE(CPF),
    foreign key (NRO) references CONTA(NRO));
```

Fonte: DME/FURB (2019) baseado no autor.



Mas para que servem estas chaves (keys)?

Aqui podemos ter uma ideia de como ficaria a codificação para definir as estruturas representadas. Facilmente você pode perceber que o *script* de definição baseada na SQL controla a redundância nos dados, não permitindo a duplicidade de determinados atributos, o que garante a unidade ou identidade de cada registro. Você deve estar se perguntando: E como se dá o relacionamento entre os registros? O relacionamento entre os registros se dá através das chaves definidas no momento da definição das estruturas. Mas o que são chaves? E para que servem? É exatamente isso que você vai descobrir em seguida.

É possível observar que há definição de chaves primárias (*primary key*) e chaves estrangeiras (*foreign key*). Entenda em que consiste cada uma:

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



O modelo relacional apresenta inúmeras soluções comerciais e de código aberto. Alguns produtos que na década passada eram estritamente licenciados e com valor agregado significativo atualmente disponibilizam versões livres com nenhuma ou poucas restrições de funcionalidades, porém com limitação de espaço de gerenciamento de dados. Alguns dos produtos mais conhecidos são:

- MYSQL – Empresa Sun Microsystems
- ORACLE – Empresa Oracle
- MSSQL – Empresa Microsoft
- POSTGRESQL – Projeto Open Source do PostgreSQL Global Development Group



Você vai conhecer mais sobre cada uma das soluções existentes no mercado na Unidade que trata dos Sistemas Gerenciadores de Bancos de Dados (SGBDs).

### 1.2.4 Modelo de banco de dados orientado a objetos

Com o estudo das abordagens de banco de dados hierárquico, em rede e relacional é possível identificar o que esses modelos têm em comum: todos são orientados a registros. Mas essa é apenas uma das suas semelhanças. Essa informação é importante, pois a partir de agora será observada uma mudança significativa no paradigma: o modelo que será abordado a seguir é orientado a objetos, ou seja, sua estrutura apresenta um conjunto de variáveis (similar aos atributos dos modelos que o precederam), porém com mecanismos de controle totalmente diferenciados e específicos.

A orientação a objetos é um paradigma de análise, projeto e programação de sistemas de *software* baseado na composição e interação entre diversas unidades de *software* denominadas “objetos” (MULLER, 2002).

Porém, estas unidades de *software* (objetos) em geral são transientes. Objetos transientes são objetos que existem apenas enquanto a aplicação que os criou continuar executando-os. Após o término da aplicação eles deixam de existir.

Para resolver esse problema é necessário tornar esses objetos persistentes, ou seja, fazer com que os dados referentes a eles continuem existindo após o término do programa que os criou. Daí a criação do modelo de banco de dados orientado a objetos.

Segundo Muller (2002), os bancos de dados orientados a objetos surgiram em ambiente acadêmico e começaram a se tornar comercialmente viáveis em meados da década de 80. Ainda segundo o autor, a motivação para o seu surgimento está em função dos limites de armazenamento e a representação semântica impostas no modelo relacional. Os sistemas de informações geográficas dos sistemas CAD/CAM são mais facilmente construídos com tipos complexos de dados e, portanto, podem ser considerados exemplos desta abordagem. A habilidade para criar os tipos de dados necessários é uma característica das linguagens de programação orientadas a objeto e, portanto, estes sistemas necessitam guardar representações das estruturas de dados

que utilizam o armazenamento permanente.

A estrutura padrão para os bancos de dados orientados a objeto foi feita pelo Object Database Management Group (ODMG). Formado por representantes dos principais fabricantes de bancos de dados orientado a objetos disponíveis comercialmente, este grupo tem o compromisso de incorporar o padrão em seus produtos.

A Figura 16 ilustra a representação gráfica da abordagem orientada a objetos.

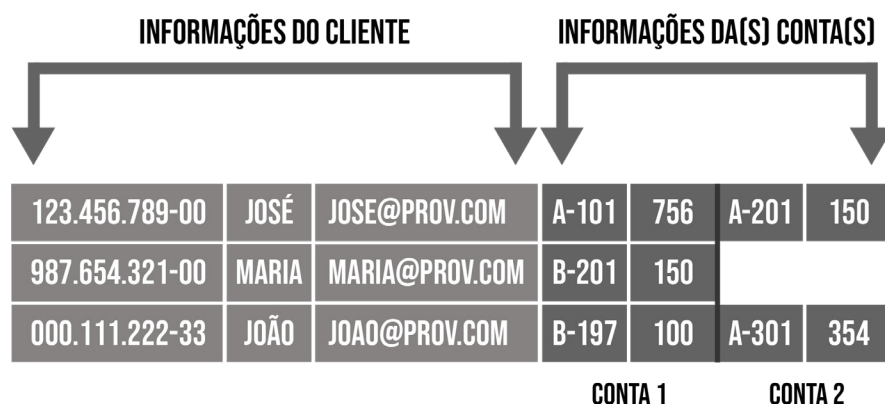


Figura 16: Representação dos dados da abordagem orientada a objetos.  
Fonte: DME/FURB (2019) baseado no autor.

Na representação gráfica acima é possível observar que não existe nenhuma ligação entre os objetos, pois cada objeto possui identidade própria e incorpora a definição de outro objeto. No exemplo, o objeto “cliente” apresenta suas variáveis, os atributos, e uma lista de variáveis do objeto “conta”. No momento da persistência, todos os valores mantidos nas variáveis serão armazenados permanentemente.

Pode-se dizer que o surgimento das estruturas de bancos de dados orientados a objetos supriu, em parte, problemas de difícil solução em estruturas relacionais. A partir disso, acreditava-se que tais bancos de dados ganhariam grande parcela do mercado. Hoje, porém, tem-se a convicção de que os bancos de dados orientados a objetos são usados em aplicações especializadas, enquanto os sistemas relacionais continuam a sustentar os negócios tradicionais, onde as estruturas de dados baseadas em relações são suficientes.

O diagrama de classes da Open Modeling Language (OML) serve, geralmente, como esquema para o modelo de dados orientado a objeto, onde os objetos que representam o mesmo tipo de valores (variáveis) e os mesmos métodos são agrupados em classes (MULLER, 2002). A classe, por sua vez, pode ser vista como uma definição do tipo para objetos, e esta combinação compacta de dados e métodos abrangendo uma definição do tipo é similar ao tipo abstrato em uma linguagem de programação. Para que o objeto acesse os dados de outro objeto, é necessária a utilização dos métodos deste outro objeto. Assim é possível garantir a abstração de dados.

Para tentar ganhar mercado e adeptos, a ODMG propôs uma linguagem dos moldes do que é SQL da abordagem relacional. Pelo que se pode observar nos produtos comerciais encontrados, não obtiveram sucesso, pois na Object Query Launch (ODL) ela ainda não é largamente utilizada.

O código a seguir define as estruturas representadas para a linguagem C++ com as adequações da ODMG:

```
Class Cliente: Public Objeto_Persistente {
Private:
    String nome;
    String email;
Public:
    Int cpf;
    Set <Ref <Conta>> contas inverse Conta :: proprietarios;
};

Class Conta: Public Objeto_Persistente {
Private:
    double saldo;
Public:
    Int numero;
    Set <Ref <Cliente>> proprietarios inverse Cliente :: contas;
    Int entrar_saldo();
    Int atualizar_saldo(int valor);
};
```

Fonte: DME/FURB (2019) baseado no autor.

Observe no exemplo a definição da classe cliente com atributos visíveis somente pela classe (cláusula *private*) e atributos visíveis por todas as classes (cláusula *public*). A definição set que contempla a palavra preservada *inverse* é utilizada para especificar restrições da integridade referencial. Já na classe “conta” podemos visualizar também a definição dos métodos públicos. Lembrando que não estão sendo definidos os métodos “getters” e “setters” para a manutenção das variáveis de cada classe.

A abordagem de banco de dados orientado a objetos teve uma nova chance de crescimento em meados de 2004, quando foram lançados alguns produtos de código aberto escritos em linguagens orientadas a objetos, como o Java e o C++. Com isso, acreditava-se que o modelo poderia se consolidar no mercado. Mas não foi o que ocorreu. Alguns dos produtos mais conhecidos são:

- ORION – Empresa MCC
- GemStone – Empresa GemStone Systems
- ObjectStore – Empresa Progress Software
- POET – Empresa Versant Corp.

### 1.2.5 Modelo de banco de dados pós-relacional

O paradigma da orientação a objetos existe há no mínimo três décadas e vem se consolidando cada vez mais em termos de análise, projeto e programação de sistemas.



Para conhecer mais sobre o modelo de banco de dados orientado a objetos, recomendamos a leitura da obra de Khoshafian (1994).

Por sua vez, o modelo de banco de dados relacional também está consolidado quanto às estruturas de armazenamento e gerenciamento de dados.

Dessa forma, como seria possível unir o que há de melhor da orientação a objetos com o potencial do modelo relacional? Para atender essa necessidade surgiu a abordagem de banco de dados pós-relacional, que será estudado a seguir.

O modelo pós-relacional, também conhecido como objeto-relacional, incorpora ao modelo relacional já consolidado os recursos cada vez mais emergentes da especificação e programação orientada a objetos. Há boas perspectivas para o modelo que está sendo apresentando, pois muitos dos bancos de dados relacionais utilizados atualmente estão adicionando características da orientação a objetos. A Figura 17 ilustra a organização dos dados no modelo pós-relacional:

**TABELA CLIENTE**

CPF	NOME	E-MAIL	CONTAS			
123.456.789.00	JOSÉ	JOSE@PROV.COM	A-101	756	B-201	150
987.654.321.00	MARIA	MARIA@PROV.COM	B-201	150		
000.111.222.33	JOÃO	JOAO@PROV.COM	B-197	100	A-301	354

Figura 17: Representação dos dados no modelo orientado a objetos.  
Fonte: DME/FURB (2019) baseado no autor.

O modelo de banco de dados pós-relacional surgiu na tentativa de suprir as dificuldades dos sistemas relacionais convencionais no que tange à representação e manipulação de dados complexos. Como alternativa, então, se propõe a adição de recursos da orientação a objeto ao modelo relacional, permitindo-se assim o manuseio de tais dados com a utilização das facilidades da SQL.

Para tanto foi necessário adicionar extensões de tipos de dados básicos e objetos complexos no contexto da SQL. Para manipular os tipos complexos, o modelo adota um padrão SQL3 que é uma extensão da linguagem SQL para suportar o modelo de objetos. Com isso, a SQL passa a permitir a definição e armazenamento de procedimentos, assim como efetuar chamadas de rotinas externas de dentro de sentenças SQL.

Foi necessário praticamente uma década para que surgisse uma nova versão da SQL, a SQL3 (BEIGHLEY, 2008). Algumas diretrizes apresentadas foram adotadas pelos principais bancos de dados relacionais, como os tipos complexos para armazenamento de dados multimídia e geográficos, assim como herança de tipos de dados estruturados. Porém, boa parte deles seguiu uma carreira solo e passou a implementar funcionalidades para a chamada externa de rotinas em comandos SQL. Isso fez com que as soluções que

adotassem tais recursos ficassem restritas ao uso do banco de dados em questão. Mesmo assim, boa parte permite ao seu modo a definição de objetos com tipos primitivos ou definidos pela necessidade da aplicação, e para acessar tais objetos específicos muitas vezes é necessária a implementação de métodos que podem ser invocados diretamente nos comandos de recuperação de dados.

A seguir é possível observar um exemplo de definição da estrutura apresentada na figura acima (17), onde o código é baseado na linguagem SQL-3, que define as estruturas:

```
create type ObjCONTA (  
    NRO        integer not null,  
    SALDO      float);  
  
create type ObjCLIENTE (  
    CPF        integer not null,  
    NOME       varchar(30),  
    EMAIL      varchar(50),  
    CONTAS     setof (ObjCONTA));  
  
create table CLIENTE of type ObjCliente;
```

Fonte: DME/FURB (2019) baseado no autor.

No código podemos observar a definição de dois tipos específicos ao cenário da aplicação: objeto conta e objeto cliente. No objeto cliente temos a criação de um tipo na forma de lista ou a rede de valores, que por sua vez é do tipo objeto conta. Por fim temos a criação da tabela cliente, que é do tipo cliente. O exemplo corresponde à estrutura ilustrada desde o primeiro modelo apresentado. Devemos reconhecer que não é a estrutura ideal para demonstrar as potencialidades do modelo. Quanto à manipulação dos dados, são utilizados os mesmos comandos reconhecidos pelo modelo relacional.

O modelo pós-relacional apresenta soluções comerciais, inclusive de código aberto. Alguns destes produtos mais conhecidos são:

- ORACLE (a partir da versão 9i) – Empresa Oracle
- DB2 (a partir da versão 7.1) – Empresa IBM Corp.
- POSTGRESQL (a partir da versão 6.0) – Projeto Open Source do PostgreSQL Global Development Group

Chegamos no final da seção que discutiu os modelos de bancos de dados. Mas antes de avançar para a arquitetura de um SGBDR, precisamos tranquilizar você acerca de um tema que não iremos explorar nesta disciplina. É possível que você já tenha ouvido falar no modelo NoSQL (Not Only SQL). Sobre este modelo falaremos em outra disciplina, dada sua relevância no contexto atual do desenvolvimento de sistemas não convencionais. Além disso, primeiramente é preciso que você domine os aspectos que envolvem o modelo relacional, foco principal desta disciplina.



Se você ficou interessado em saber mais sobre os recursos de cada um dos bancos de dados do modelo pós-relacional apresentados, recomendamos acessar o site do fabricante, que é a fonte mais confiável de informação.

## 1.3 Arquitetura de um sistema gerenciador de banco de dados relacional (SGBDR)

Na seção anterior foi possível conhecer os modelos de bancos de dados, ou seja, quais as principais possibilidades de organização de estruturas para o armazenamento dos dados. Atualmente é possível encontrar aplicações nos mais variados segmentos, cada qual com suas especificidades, e com os mais variados modelos de bancos de dados apresentados.

No entanto, o modelo relacional é atualmente o mais utilizado, principalmente em sistemas de informação que se propõem a manter dados do cotidiano das empresas. Agora vamos abordar com maior ênfase o modelo relacional, principalmente sua evolução para os sistemas gerenciadores de bancos de dados relacionais (SGBDRs).

Mas, antes, faremos uma analogia entre o armazenamento baseado em arquivos e os dados mantidos em SGBDRs. Esta etapa é fundamental e vai ajudá-lo a compreender os diferenciais de um SGBDR e sua importância no contexto do desenvolvimento de sistemas, em especial os corporativos.

### 1.3.1 Processamento de arquivos versus SGBDR

Segundo Elmasri e Navathe (2012), quando o modelo relacional foi apresentado, na década de 70, tinha-se clareza quanto a como os dados seriam organizados e armazenados: através de tabelas e relações entre as tabelas com valores próprios (chaves). Porém, ainda segundo o autor, não havia uma solução para controlar essas relações e muitos outros aspectos que iriam surgir ao longo do tempo. Eis que durante alguns anos o modelo foi utilizado através de arquivos gerenciados pelo sistema operacional, caracterizando o que se denominou processamento de arquivos.

Algumas características marcaram o processamento tradicional de arquivos (ELMASRI; NAVATHE, 2012):

- Definição dos dados como parte do código da aplicação:

As estruturas utilizadas pelo sistema eram definidas e mantidas pela aplicação, ou seja, as propriedades sobre as estruturas, como nomes de arquivos, atributos, tipos de dados e mecanismos de acesso, entre outras, faziam parte da codificação da aplicação, estabelecendo certa dependência.

- Ausência de múltiplas visões sobre os dados:

Os dados armazenados não dispunham de funcionalidades para que os usuários pudessem visualizá-los sob diferentes visões. Imagine a apuração de um relatório

de vendas agrupado por vendedores, por exemplo. Para tanto, seria necessário o processamento significativo dos arquivos envolvidos com total controle da aplicação.

- Controle do compartilhamento dos dados deficitário:

Mesmo naquela época era comum uma aplicação ser acessada por vários usuários simultaneamente. Por isso era necessário controlar o acesso aos dados para evitar inconsistências. Este controle era feito pela aplicação em conjunto com as funcionalidades disponíveis no sistema operacional – era, portanto, suscetível a erros de codificação.

- Dados armazenados sem mecanismos de controle de segurança:

Os arquivos eram gerenciados pela aplicação, assim como o acesso aos registros e as operações sobre eles. Dessa forma, o controle da segurança sobre os dados era estabelecido na codificação da aplicação, o que também está propenso a erros. Além disso, não havia mecanismos de recuperação (em caso de falhas) e de criptografia dos dados armazenados. Boa parte dessa responsabilidade também era atribuída ao sistema operacional.

Durante muitos anos, empresas de desenvolvimento de *softwares* utilizaram as soluções baseadas no processamento tradicional de arquivos. Atualmente há muitos sistemas que manipulam grandes volumes de dados que ainda utilizam essa solução. Esses sistemas são denominados sistemas legados, e dificilmente deverão migrar para novas formas de estrutura de armazenamento, pois o custo da migração é significativo.

Com o passar dos anos, empresas que lançaram soluções para o armazenamento de dados utilizando o modelo relacional identificaram a necessidade de incorporar funcionalidades à proposta inicial de Edgar Frank Codd. Assim surgiu o sistema gerenciador de banco de dados relacional (SGBDR).

Segundo Date (2004), Edgar Frank Codd, precursor do modelo relacional, definiu um conjunto de regras básicas que caracterizam a abordagem, denominadas “as 13 regras de Codd”. Antes de abordar o conceito de SGBDR, é importante conhecer estas regras. Ainda segundo Date (2004), na década de 80, Codd, como era conhecido, publicou um artigo onde definia as 13 regras para que um SGBD fosse considerado relacional. Elas estão descritas a seguir:

**1.ª – Regra fundamental** - Um SGBD relacional deve gerir os seus dados usando apenas suas capacidades relacionais, ou seja, tabelas e relações entre tabelas.

**2.ª – Regra da informação** - Toda informação deve ser representada de uma única forma: como dados em uma tabela, ou seja, a tabela é a unidade básica de armazenamento.



Utilizamos o termo “sistema legado” para caracterizar um sistema antigo que permanece em operação. Em geral, são sistemas que utilizam tecnologias obsoletas, ou seja, antigos não mais usuais.



**3.<sup>a</sup> – Regra da garantia de acesso** - Todo dado (valor atômico) pode ser acessado logicamente (e unicamente) através do nome da tabela, do valor da chave primária da linha, e do nome da coluna.

**4.<sup>a</sup> – Tratamento sistemático de valores nulos** - Os valores nulos (diferentes do zero, da *string* vazia, da *string* de caracteres em branco e de outros valores não nulos) existem para representar dados não existentes de forma sistemática, independentemente do tipo de dado.

**5.<sup>a</sup> – Catálogo on-line baseado no modelo relacional** - A descrição do banco de dados é representada no nível lógico como dados ordinários, isto é, em tabelas. Isso permite que usuários autorizados manipulem o metadados da mesma forma como é manipulado o conjunto de dados, em termos de consulta.

**6.<sup>a</sup> – Regra da sublinguagem compreensiva** - Um sistema relacional pode suportar várias linguagens e formas de uso, mas deve possuir ao menos uma linguagem com sintaxe bem definida e expressa por cadeia de caracteres, com habilidade de apoiar a definição de dados, a definição de visões, a manipulação de dados, as restrições de integridade, a autorização e a fronteira de transações.

**7.<sup>a</sup> – Regra da atualização de visões** - Toda visão que for teoricamente atualizável será também atualizável pelo sistema.

**8.<sup>a</sup> – Inserção, atualização e eliminação de alto nível** - A capacidade de manipular a relação base ou as relações derivadas como um operador único não se aplica apenas à recuperação de dados, mas também à inserção, alteração e eliminação de dados.

**9.<sup>a</sup> – Independência física dos dados** - Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas, quaisquer que sejam as modificações na representação de armazenagem ou nos métodos de acesso internos.

**10.<sup>a</sup> – Independência lógica dos dados** - Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as mudanças de informação que permitam, teoricamente, a não alteração das tabelas base.

**11.<sup>a</sup> – Independência de integridade** - As relações de integridade específicas de um banco de dados relacional devem ser definidas em uma sublinguagem de dados e armazenadas no catálogo, e não em programas.

**12.<sup>a</sup> – Independência de distribuição** - A linguagem de manipulação de dados deve possibilitar que as aplicações permaneçam inalteradas, estejam os dados centralizados ou distribuídos fisicamente.

**13.<sup>a</sup> – Regra da não subversão** - Se o sistema relacional possui uma linguagem de baixo nível (um registro por vez), não deve ser possível subverter ou ignorar as regras de integridade e restrições definidas no alto nível (muitos registros por vez).



Para atender a todas as regras de Codd, é necessário um sistema de relativa complexidade. É exatamente o caso do SGBD. Para Silberschatz (2012), um sistema gerenciador de banco de dados relacional (SGBDR) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados. A coleção de dados, normalmente conhecida como banco de dados, contém informações relevantes para um empresa.

Portanto, segundo Date (2004), o Sistema Gerenciador de Banco de Dados é o *software* que manipula todos os acessos ao banco de dados relacional.

O SGBDR – “R” de relacional – é um *software* complexo que envolve um conjunto de processos que garante, além do acesso, a correta manutenção sobre os dados. Todas as deficiências identificadas no modelo de processamento tradicional de arquivos são solucionadas pelo SGBDR. Além disso, inúmeras outras funcionalidades são implementadas e disponibilizadas, como veremos a seguir.

### 1.3.2 Funções de um SGBDR

No tópico anterior foram apresentadas algumas limitações do processamento de arquivos tradicional e a proposta de Codd (as conhecidas 13 regras do modelo relacional). Foi apresentada também a definição de SGBDR.

Agora serão abordadas as funções que podemos encontrar num SGBDR. Com isso será possível compreender melhor a complexidade envolvida nesse *software* fundamental para a gerência dos dados em sistemas corporativos ou até mesmo de pequeno porte.

Mas antes de conhecer as funções do SGBDR, é necessário entender por que elas são denominadas dessa forma. O termo é utilizado porque se trata de recursos implementados que garantem determinadas funcionalidades aos usuários que utilizam o SGBDR.

A maioria das funções são abstratas ao usuário, ou seja, transparentes, e visam principalmente garantir a integridade e a consistência dos dados no banco de dados. Na sequência estão descritas as **principais funções**.

A 1.<sup>a</sup> função se refere ao **gerenciamento do dicionário de dados**. Trata-se da definição ou descrição das estruturas do banco de dados. As definições são mantidas através de um catálogo de dados conhecido como metadados. Podemos afirmar que se trata do gerenciamento das estruturas onde os dados serão mantidos.

É necessário entender primeiro o que é o dicionário de dados, também conhecido como metadados ou catálogo de dados. O dicionário de dados de um SGBDR armazena toda a definição dos elementos estruturais dos dados, seus relacionamentos, características, parâmetros, configurações, enfim tudo o que é necessário para o acesso



Ainda nesta Unidade você vai conhecer mais sobre o perfil dos usuários envolvidos com os SGBDRs.

aos dados. Se o dicionário de dados de um SGBDR é corrompido ou perdido, o acesso aos dados do banco de dados fica comprometido. Vale lembrar que graças à existência do dicionário de dados do SGBDR não se faz necessária a inclusão de códigos complexos dos programas de aplicações para definir e acessar os dados.

A 2.<sup>a</sup> função consiste na **independência dos dados armazenados**. Toda e qualquer definição sobre as estruturas de dados deve ser realizada e mantida no SGBDR, e não na aplicação. O usuário tem uma visão abstrata dos dados, ou seja, conceitual, não se atendo a detalhes das estruturas físicas de como e onde os dados estão armazenados.

O SGBDR cria estruturas complexas necessárias ao armazenamento dos dados, consequentemente livrando os programas desta tarefa de tratar as características físicas dos dados. A independência dos dados deve garantir a abstração de como os dados estão armazenados fisicamente. Em geral, quando utilizamos um SGBDR o acesso aos dados é feito através de mecanismos, isto é, protocolo, porta e *drive* de comunicação, tornando desconhecida a localização dos diretórios e arquivos sob a gerência do SGBDR.

A 3.<sup>a</sup> função trata do **gerenciamento da segurança**. Este aspecto representa o controle sobre “quem” pode acessar os dados e “quais” operações são permitidas aos usuários que os acessam. Acrescentam-se também aspectos de criptografia sobre as estruturas físicas mantidas pelo SGBDR.

O SGBDR cria um sistema de segurança que obriga que tudo passe por este mecanismo, e desta forma mantém a privacidade dos dados. As regras de segurança determinam quais usuários podem acessar o banco de dados, quais itens de dados os usuários podem acessar e que operações de manipulação sobre os dados o usuário pode executar: consulta, inserção, exclusão e alteração, por exemplo. Isto é especialmente importante em sistemas de informações multiusuários, onde vários usuários podem acessar o banco de dados simultaneamente.

A 4.<sup>a</sup> função estabelece aspectos relacionados ao **gerenciamento de acesso a múltiplos usuários**. Todo o controle do acesso aos dados deve ser feito através de algoritmos e tabelas de bloqueio com o objetivo de garantir a integridade e a consistência dos dados. Isso é feito através dos mecanismos de controle transacional encontrados nos SGBDRs quando o usuário utiliza a linguagem de manipulação de dados.

O SGBDR mantém estruturas complexas que permitem a muitos usuários acessar os dados, com o objetivo de prover integridade e consistência do banco de dados. O SGBDR utiliza-se de algoritmos sofisticados e tabelas de bloqueio para assegurar que os usuários possam acessar o banco de dados corretamente e garantir a integridade do banco de dados. Este controle de concorrência envolve o conceito de controle transacional que você irá conhecer quando fizer uso da linguagem de manipulação dos dados mantidos em um SGBDR.



Quando utilizamos um SGBDR passamos a nos preocupar com “quais” dados vamos armazenar e recuperar, e não com “como” faremos para armazená-los e recuperá-los. O papel do “como” fica com o SGBDR.



Quem controla o acesso aos dados é o SGBDR, não a aplicação. Da mesma forma, o acesso concorrente aos dados (arquivos físicos) também é realizado pelo SGBDR, e não pelo sistema operacional.

A 5.<sup>a</sup> função é uma das mais relevantes, pois trata do **gerenciamento de integridade de dados**: é também conhecido como “integridade referencial”. Através de regras de relacionamento entre os dados e o uso de atributos de identificação (chaves), o SGBDR garante a consistência e os estados possíveis que determinados dados podem assumir.

O SGBDR aplica e força as regras de integridade de dados para eliminar problemas de inconsistência, consequentemente minimizando a redundância. Os relacionamentos dos dados armazenados do dicionário de dados são usados para forçar a integridade de dados, garantindo os estados possíveis e a sua consistência. Isto é garantido pelo uso de atributos de identificação, também conhecido como chaves. Para muitos esta é uma das principais funções do SGBDR, pois o controle da consistência dos dados exigiria muito critério e processamento pela aplicação.

A 6.<sup>a</sup> função é caracterizada pelo **gerenciamento de backup e recovery**. Através de mecanismos de controle de falhas, o SGBDR deve manter o estado dos dados de acordo com as regras de integridade e segurança definidos em seu dicionário de dados. Da mesma forma, deve permitir a realização de cópia de segurança (*backup*) e a recuperação (*recovery*) do banco de dados.

O SGBDR prevê procedimentos para a realização de cópia de segurança e recuperação dos dados de forma segura e íntegra. Os atuais SGBDRs proveem utilitários que permitem ao administrador de banco de dados (DBA) executar procedimentos de *backup* e *recovery*. A recuperação do banco de dados é necessária após uma falha, e às vezes é preciso fazer uso da cópia de segurança para restaurar o banco de dados a um estado consistente. Além disso, esta função deve garantir a recuperação de falhas associadas à consistência e à integridade dos dados, mantendo-os correlatos. A Figura 18 ilustra um conjunto de dados para exemplificar a integridade de dados.

DEPARTAMENTO			FUNCIONÁRIO		
CD_DEPTO#	NM_DEPTO	VL_ORCTO	CD_FUNC#	CD_DEPTO	NM_FUNC
100	MARKETING	10.000,00	1001	100	MARIA
110	MANUTENÇÃO	12.000,00	1002	100	JOSÉ
120	PESQUISA	7.000,00	1003	110	PAULO



Chave primária



Chave estrangeira

Figura 18: Representação de dados relacionais.  
Fonte: DME/FURB (2019) baseado no autor.



Tente imaginar a complexidade na implementação do mecanismo de gerenciamento de integridade de dados. Toda e qualquer operação sobre os dados requer avaliação prévia para permitir ou não a sua total execução. Quem realiza este papel é o SGBDR.

Observe que nas estruturas acima temos uma relação entre funcionário e departamento, ou seja, uma instância (registro) de funcionário possui uma referência (chave) indicando o local onde ele está atuando. Portanto, sob hipótese alguma poderá ser excluída uma instância de “departamento” que esteja sendo referenciada por “funcionário”.

A 7.<sup>a</sup> função é caracterizada pela **linguagem de acesso aos dados**. Para que estruturas sejam criadas e dados manipulados, precisamos fazer uso de uma linguagem. Trata-se de linguagem não procedural para a definição e manipulação das estruturas e dados mantidos no banco de dados. Através dessa linguagem é possível acessar os dados independentemente da linguagem utilizada para a construção da interface final que será disponibilizada ao usuário.

O SGBDR permite o acesso aos dados do banco de dados através de uma linguagem de consulta. Uma linguagem de consulta como, por exemplo, a SQL para SGBD relacionais é uma linguagem não procedural, isto é, ela permite que o usuário especifique o que deve ser feito sem ter que especificar como será feito. A linguagem de consulta do SGBDR contém, no mínimo, dois componentes: uma linguagem de definição, conhecida como *Data Definition Language* (DDL), e outra linguagem de manipulação de dados, conhecida como *Data Manipulation Language* (DML). A DDL define as estruturas na qual os dados serão criados, e a DML permite ao usuário extrair dados do banco de dados.

Os SGBDRs atuais dispõem de uma extensão da linguagem de consulta SQL para uma linguagem procedural interna, além de permitirem também acesso aos dados através de linguagens procedurais externas, tais como JAVA, C e outras. Os SGBDRs também apresentam utilitários administrativos que são usados pelos DBAs, e projetistas de bancos de dados para criar, implementar, monitorar e manter o banco de dados.

A 8.<sup>a</sup> função trata das **múltiplas visões dos dados**. Esta função estabelece que deve haver a possibilidade de diferentes usuários do banco de dados visualizarem os mesmos dados sob visões distintas, ou específicas ao contexto pretendido pelo usuário. Esta função está associada também à independência dos dados.

É comum em um banco de dados a presença de multiusuários, onde cada qual pode solicitar diferentes visões do banco de dados. Uma visão pode ser um subconjunto de dados ou conter uma visão virtual dos dados, ou seja, pode-se estar visualizando dados que se referem a dados derivados e que não necessariamente estão armazenados. Exemplo: cálculos matemáticos de apuração de valores. A seguir veja um exemplo de múltiplas visões dos dados.

O esquema apresentado na Figura 19 ilustra um exemplo de múltiplas visões dos dados:

NM_DEPTO	NM_DEPTO	NM_DEPTO	QT_FUNC		
MARKETING	JOSÉ	MARKETING	2		
MARKETING	MARIA	MANUTENÇÃO	1	QT_DEPTO	QT_FUNC
MANUTENÇÃO	PAULO	PESQUISA	0	3	29.000,00

CD_DPTO#	NM_DEPTO	VL_ORCTO	CD_FUNC#	CD_DEPTO	NM_FUNC
100	MARKETING	10.000,00	1001	100	JOSÉ
110	MANUTENÇÃO	12.000,00	1002	100	MARIA
120	PESQUISA	7.000,00	1003	110	PAULO

DEPARTAMENTO

FUNCIONÁRIO

Figura 19 Representação de dados relacionais com múltiplas visões.  
 Fonte: DME/FURB (2019) baseado no autor.



É provável que você já tenha ouvido falar sobre a Linguagem SQL. Caso não, fique tranquilo. Falaremos sobre ela mais adiante em nossa disciplina.

Percebe-se que as funções dos SGBDRs nos dão uma ideia do que esperar deste *software*. Agora faremos uma reflexão sobre quão complexo é controlar os vários aspectos apresentados. Como exemplo inicial temos o controle de integridade referencial. É necessário visualizar a definição e a codificação das regras, e principalmente a necessidade de adaptação do mecanismo de controle nos diversos cenários que serão submetidos quando da utilização para atender às especificidades das aplicações.

O controle da integridade referencial se faz necessário em toda e qualquer operação que envolva inserção, alteração ou exclusão de dados, o que é bastante complexo. Imagine então o mecanismo responsável pelo controle de compartilhamento dos dados... deve-se saber que por conta das definições do sistema operacional dois processos não podem alterar o mesmo arquivo. Não estamos falando de leitura, e sim manutenção dos dados armazenados. Mas como isso é possível em um SGBDR? Ao contrário do processamento de arquivos tradicional, onde quem gerencia o acesso aos arquivos é a aplicação no contexto que envolve um SGBD, é ele, e somente ele, quem acessa as estruturas e os dados dos arquivos. Portanto o SGBD, através de algoritmos e tabelas de bloqueios, é quem estabelece regras para a liberação de acesso aos dados, ou seja, ocorre uma abstração do processo isentando a aplicação deste controle. É necessário reforçar que a evolução dos SGBDs está principalmente associada ao modelo relacional, ou seja, outros modelos também apresentam um *software* de gerenciamento, mas os mais complexos e utilizados são os seus bebês relacionais.

Tamanha complexidade e recursos fizeram com que ao longo do tempo fosse necessário definir papéis e responsabilidades. Ao envolver os usuários do sistema, uma arquitetura é constituída para representar os níveis de envolvimento e interação com os SGBDRs e é desse assunto que o tópico a seguir irá tratar.

### 1.3.3 Arquitetura de um SGBDR

A forma como o termo “arquitetura” está sendo utilizado aqui difere da apresentada no início do material. Aqui “arquitetura” é utilizado por sua capacidade de representar um padrão definido pela ANSI/SPARC para o desenvolvimento de tecnologias que envolvem bancos de dados, em especial os SGBDs. Trata-se de um conjunto de requisitos básicos que devem nortear quaisquer modelos de bancos de dados relacionais.

Imagine o seguinte cenário: um funcionário do departamento de produção de uma indústria utiliza-se de uma aplicação para atualizar o estoque. Mas qual seria a necessidade de um funcionário do departamento de produção conhecer um banco de dados? Antes de respondermos à pergunta, vamos conhecer a representação gráfica da arquitetura de um SGBDR, apresentada na Figura 20.

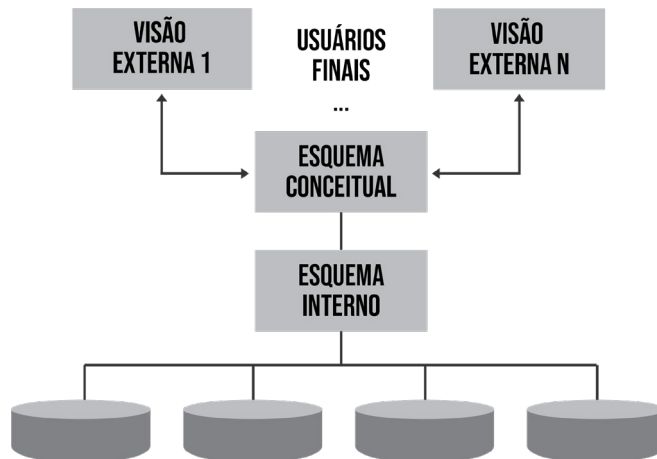


Figura 20: Representação da arquitetura de um SGBDR.  
Fonte: DME/FURB (2019) baseado no autor.

A resposta provavelmente seria não, ou seja, o funcionário do departamento de produção não precisa conhecer um banco de dados para executar as atividades. Para ilustrar melhor a arquitetura de um SGBDR, observe a figura acima. Nela podemos identificar níveis, os quais você irá conhecer em breve. Podemos afirmar que cada nível – ou, se preferir, visão – constitui um perfil de usuário. Este perfil está associado principalmente às necessidades do usuário ou atores em cada cenário. Além disso, as capacidades e competências também se alteram em cada nível da arquitetura. Mas antes de avançar mais sobre cada um dos níveis, convidamos você a conhecer os requisitos para a arquitetura de um SGBDR.

Vamos conhecer os requisitos impostos pela arquitetura do SGBD antes de conhecermos seus níveis, papéis e atores envolvidos:

- Todos os usuários devem ser capazes de acessar os mesmos dados.
- Uma visualização dos dados não deve interferir na visualização por outros

usuários.

- Os usuários não devem necessitar conhecer os detalhes do armazenamento físico do banco de dados.
- O administrador do banco de dados deve ser capaz de fazer alterações na estrutura sem afetar as visualizações dos usuários.
- A estrutura interna do banco de dados não deve ser afetada por mudanças no aspecto físico do armazenamento.

Abordados os requisitos, é chegado o momento de conhecer também os níveis.

**São basicamente três níveis**, conforme ilustrado na Figura 21.

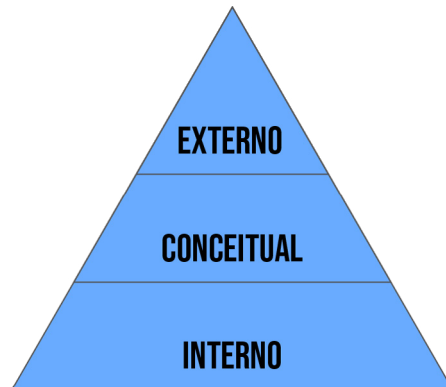


Figura 21: Níveis da arquitetura de um SGBDR.  
Fonte: DME/FURB (2019) baseado no autor.

Em seguida vamos caracterizar cada um desses níveis e os atores envolvidos neles. A ilustração, em forma de pirâmide, não foi feita assim por acaso. Adiante será apresentada a razão da escolha dessa figura geométrica.

O primeiro deles é o **nível externo**, também conhecido como nível de visão do usuário. Ele possibilita o acesso a conjuntos de dados específicos da necessidade ou interesse dos usuários. A Figura 22 apresenta o nível externo.

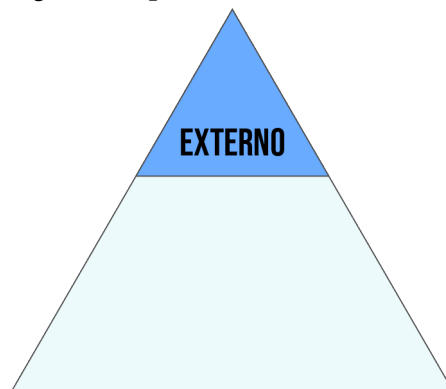


Figura 22: Nível externo da arquitetura de um SGBDR.  
Fonte: DME/FURB (2019) baseado no autor.

No nível externo o usuário não precisa saber como os dados estão armazenados nem de que forma eles são mantidos e recuperados. Este nível apresenta um alto nível



de abstração em relação aos conceitos que estamos discutindo em nossos estudos, visto que estamos falando de usuários finais dos sistemas de informação.

Em seguida há o **nível conceitual**. Também conhecido como nível intermediário, o nível conceitual possui o esquema conceitual que descreve a estrutura de todo o banco de dados para a comunidade de usuários. A Figura 23 apresenta o nível conceitual.

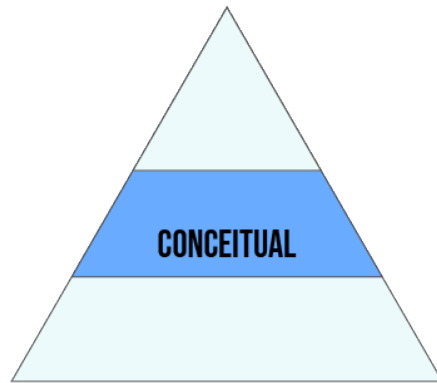


Figura 23: Nível conceitual da arquitetura de um SGBDR.  
Fonte: DME/FURB (2019) baseado no autor.

O esquema conceitual abstrai os detalhes das estruturas de armazenamento físico e se concentra na descrição de entidades, tipos de dados, relações entre os dados, operações dos usuários sobre os dados e restrições de dados. A abstração neste nível é mediana, pois os atores envolvidos precisam necessariamente conhecer um pouco de banco de dados – mais especificamente, a forma como os dados estão organizados: o modelo de dados.

Por fim, o **nível físico** possui o esquema interno. O esquema interno descreve a estrutura de armazenamento físico do banco de dados através de um modelo interno de armazenamento de dados onde estão descritos todos os seus detalhes. A Figura 24 ilustra esse nível.

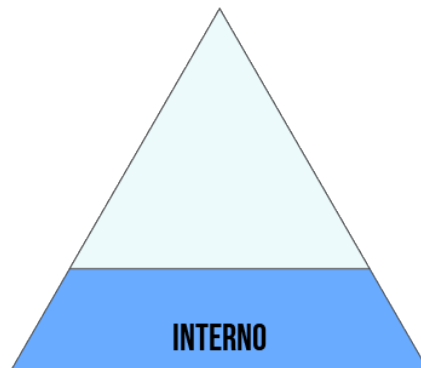


Figura 24: Nível físico da arquitetura de um SGBDR.  
Fonte: DME/FURB (2019) baseado no autor.

O esquema interno apresenta os caminhos e todas as operações de baixo nível para o armazenamento e a recuperação dos dados. É o nível mais próximo que um usuário



O próximo ciclo é dedicado à modelagem de dados em um SGBDR. Fique tranquilo, pois iremos discutir e exercitar bastante sobre o assunto.



pode chegar dos dados. Neste nível a abstração é mínima, limitada às operações de restrições estabelecidas pelos desenvolvedores dos SGBDRs. O principal ator inserido neste nível é o Administrador do Banco de Dados (DBA).

Após abordar os níveis, vamos conhecer os atores envolvidos em cada um dos níveis de SGBDR. No **nível externo** temos os usuários finais, que são pessoas cujas tarefas exigem o acesso ao banco de dados para consultas, atualização e geração de relatórios. Na essência, o banco de dados existe principalmente para ser utilizado por parte destes usuários. Em razão disso existem diversas categorias de usuários finais. Há um conjunto de usuários, denominados casuais, que ocasionalmente acessam o banco de dados, porém podem necessitar de informações diferentes a cada vez. Para isso, eles utilizam uma linguagem sofisticada de consulta de banco de dados para especificar suas solicitações. Geralmente são gerentes de nível hierárquico médio ou elevado.

“Paramétricos” é a denominação atribuída aos usuários que representam a maior porção dentre os usuários finais do SGBDR. As principais tarefas giram em torno de realizar buscas constantemente e atualizar o banco de dados utilizando tipos padronizados de consultas e atualizações chamadas de transações padronizadas, que foram cuidadosamente programadas e testadas. Um exemplo disso são os operadores de caixa.

Os usuários sofisticados incluem ainda engenheiros, cientistas, analistas de negócios e outros que se familiarizaram profundamente com as facilidades do SGBDR, no sentido de implementar suas aplicações de forma que estas atendam às suas exigências complexas.

No **nível conceitual** temos os atores responsáveis por identificar as demandas dos usuários finais, e projetar e implementar os sistemas de informação. É de responsabilidade do analista de sistemas identificar e determinar em conjunto com os usuários finais as funcionalidades que o sistema deverá apresentar para atender às necessidades desses usuários.

O projetista, também conhecido como arquiteto de banco de dados, é responsável por identificar os dados a serem armazenados no banco de dados e por escolher as estruturas apropriadas para representar e armazenar esses dados. Essas tarefas são realizadas antes de o banco de dados ser efetivamente implementado. Já o programador implementa as especificações através de ambiente e linguagens de programação de computadores. O resultado do trabalho dos atores envolvidos neste nível é o *software* para o usuário final. A Figura 25 expõe os usuários mais conhecidos do nível conceitual.

O **nível físico** é o mais complexo na arquitetura de um SGBDR. É neste nível que está inserido o administrador de banco de dados mais conhecido como DBA (*Database Administrator*). O DBA é responsável por autorizar o acesso ao banco de dados, por coordenar e monitorar sua utilização, assim como gerenciar as demandas sobre os recursos de *hardware* e *software* quando necessários. O DBA deve conhecer

em detalhes o funcionamento e os recursos do SGBDR.

A Figura 25 apresenta a distribuição dos diferentes tipos de usuários na arquitetura de um SGBDR.

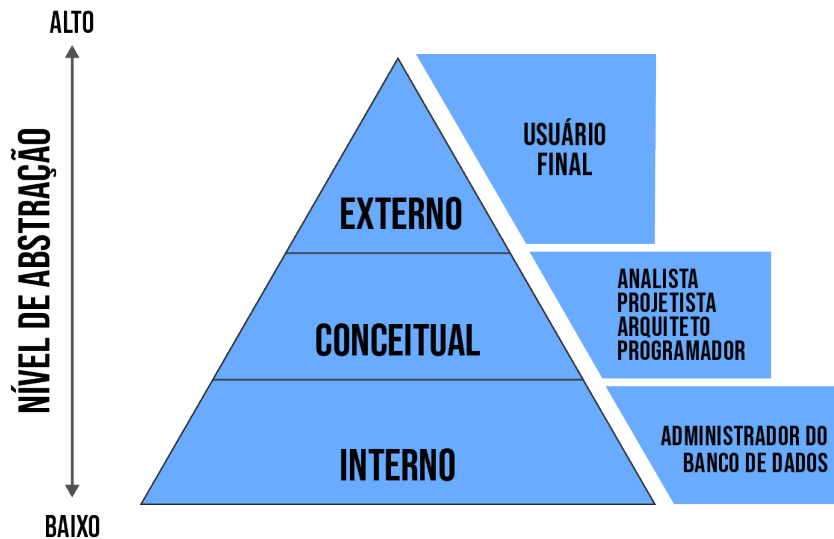


Figura 25: Distribuição dos usuários nos níveis da arquitetura de um SGBDR.  
Fonte: DME/FURB (2019) baseado no autor.

Avaliando de forma geral, quando falamos em arquitetura de um SGBD, nos referimos à forma de organização e mapeamento das funcionalidades e os papéis dos usuários envolvidos com o SGBDR. A arquitetura estabelece níveis de abstração distintos: quanto mais próximo da base da pirâmide, menor é o nível de abstração, ou seja, maior o nível de envolvimento dos usuários. Neste caso, estamos falando do administrador do banco de dados.

A arquitetura pressupõe que todos os usuários devem ser capazes de acessar os mesmos dados; não deverá haver necessidade de os usuários conhecerem detalhes de armazenamento físico do banco de dados, assim como a estrutura interna do banco de dados não deverá ser afetada por mudanças de aspectos físicos de armazenamento, como, por exemplo, a extensão ou ajustes dos discos rígidos.

O administrador do banco de dados, o DBA, deve ser capaz de mudar a estrutura conceitual de um banco de dados sem afetar todos os usuários. Mas o DBA, inserido no nível interno, é apenas um dos vários atores em cena quando falamos de SGBDR. Temos também os analistas de sistemas, projetistas, arquitetos e programadores, assim como os usuários finais que, através de inúmeros sistemas de informações desenvolvidos, utilizam-se de SGBD sem ao menos saber de sua existência. Isto é uma forma de visualizar na prática a existência dos níveis de arquitetura de um SGBDR.

Agora que você conhece as diferenças entre um sistema convencional de armazenamento de dados, as principais funções e a arquitetura de um SGBDR, convido-o(a) a explorar, de forma sintetizada, algumas das características dos principais SGBDRs existentes no mercado.



Fique tranquilo, que iremos conhecer na prática alguns destes SGBDRs em nossas aulas laboratoriais.

## 1.4 Interfaces de SGBD relacionais existentes

Serão abordadas agora algumas das características e funcionalidades dos principais SGBDs relacionais do mercado. Há muito material disponível na internet sobre os bancos de dados. Compilamos alguns produtos para auxiliá-lo(a) nos estudos para os quais apresentamos informações breves. Isso lhe dá a liberdade para pesquisar e definir seus próprios caminhos de aprendizado. Seleccionamos os seguintes SGBDRs: MySQL, Oracle, PostGreSQL e Firebird.

Segundo pesquisas e números disponíveis em sites na internet, o MySQL é considerado o banco de dados mais utilizado atualmente. Mais de 10 milhões de usuários o utilizam. Sua popularização se deve principalmente à facilidade de integração com a linguagem PHP, que é considerada a mais utilizada no desenvolvimento de páginas dinâmicas da internet.

Atualmente o MySQL é de propriedade da Oracle e está disponível com base na licença General Public License (GPL).

E o que esperar do MySQL? Até a versão 3.x, em razão de não possuir funcionalidades consideradas essenciais em muitas áreas, como controle de integridade referencial, subconsultas, controle transacional e rotinas armazenadas, o MySQL era considerado um sistema mais “leve” e, portanto, recomendado para sistemas menos exigentes.

Mas isso mudou. Atualmente ele está integrado a sistemas de grande porte e sua utilização cresce a cada dia.

Com a expansão dos bancos de dados livres, principalmente do MySQL e do PostGreSQL, empresas com a Microsoft e a Oracle se viram obrigadas a lançar versões “livres” de seus produtos para o armazenamento de dados.

A primeira a fazê-lo foi a Oracle, que em 2005 lançou o Oracle Database Express Edition 10g. A versão gratuita dessa ferramenta apresenta o mesmo núcleo da versão paga – com algumas limitações, é claro.

O que podemos esperar do Oracle? A versão livre disponibilizada oferece os principais recursos necessários à utilização, sem deixar nada a desejar em relação aos concorrentes livres.

As principais limitações estão no tamanho máximo de memória para processamento (RAM) e para armazenamento secundário (hardisk). Além disso, há outras limitações em termos de recursos/funcionalidades de múltiplas instâncias, replicação de dados, etc., mas estas limitações são direcionadas para aplicações de grande porte, forçando a aquisição do produto.

O PostGreSQL ganhou muito espaço e uma reputação positiva nos últimos anos



Curioso em saber mais sobre a licença GPL? Em nossa webaula explicamos, mas se preferir faça uma pesquisa em um sistema de busca e aprimore seus conhecimentos agora mesmo.



Para saber mais sobre o MySQL acesse.



Para saber mais sobre o Oracle, acesse.



em relação ao seu principal concorrente, o MySQL. Isso fez com que ele se tornasse, segundo os críticos, o melhor SGBDR gratuito existente. Desde sua concepção (final da década de 80) o PostgreSQL trilhou um caminho marcado pela evolução constante. Para muitos, o “pulo do gato” foi em 2005, quando foi lançada a primeira versão nativa para Windows.

O PostgreSQL é um banco de dados corporativo com funcionalidades sofisticadas e com suporte para muitos recursos emergentes, como, por exemplo, o gerenciamento de dados geoespaciais e multimídia. Até o final da última década, um dos pontos fracos deste SGBDR era a limitação nas plataformas de utilização e a documentação disponibilizada aos desenvolvedores. Mas o PostgreSQL evoluiu e agora é uma referência no mercado.

Se você deseja utilizar conceitos de orientação a objetos, o PostgreSQL apresenta características de um banco objeto-relacional, suportando várias funcionalidades desse modelo.

Em 2006, a empresa Borland, para concorrer com os SGBDRs livres, em especial o MySQL e o PostgreSQL, resolve abrir o código fonte do seu banco de dados relacional Interbase. Alguns programadores tiveram a iniciativa de assumir o código disponibilizado, surgindo assim o Firebird, que é amplamente utilizado no Brasil e em vários países da Europa.

O Firebird já nasceu com características avançadas, o que o torna seguro e estável. Este SGBDR é livre e não apresenta limitações de uso. Seu suporte é facilitado por um amplo debate em listas de discussão para apoio técnico. Ele apresenta versões para diversos sistemas operacionais, sendo considerado um SGBD “leve” em relação às funcionalidades que disponibiliza. É uma boa opção de SGBD livre, porém dá sinais de que está em processo de descontinuidade, o que não estimula muito a adesão de novos usuários.

O que acabamos de apresentar é apenas de uma breve síntese para auxiliá-lo na escolha do SGBDR para a continuidade dos nossos estudos.



Para saber mais sobre o PostgreSQL, acesse.



Para saber mais sobre o Firebird, acesse.



Em nossas aulas práticas laboratoriais você terá a oportunidade de conhecer estes SGBDRs.

## Conclusão

Bem, chegamos ao fim deste primeiro Ciclo de estudos em nossa disciplina. Você teve a oportunidade de conhecer um pouco os conceitos fundamentais, um breve histórico e os modelos de bancos de dados existentes. No modelo hierárquico os dados são organizados em estruturas baseadas em árvores. Aqui algumas fragilidades são evidentes, como a necessidade de redundância nos dados e a forte dependência entre as estruturas e a aplicação. Já no modelo em rede os dados são organizados em estruturas representadas através de gráficos arbitrários. Este modelo é marcado pela utilização da linguagem Cobol como linguagem de definição e manipulação dos dados. Pode-se dizer que ele é uma evolução do modelo hierárquico.

Vimos que o modelo relacional incorpora o que até então havia de melhor nos antecessores (modelos hierárquico e em rede). É atualmente o modelo mais utilizado, por isso damos destaque para apresentar com maior ênfase as características e funcionalidades dos principais produtos no mercado. O modelo orientado a objetos surgiu em função dos limites de armazenamento e representação semântica impostos pelo modelo relacional, ou seja, alguns sistemas necessitam de tipos complexos para representar os dados. Como exemplo temos os sistemas geográficos ou especiais. Este modelo não é largamente utilizado no mercado ainda, mas há expectativas de que isso mude no futuro.

Na sequência constatamos que, com as expectativas vislumbradas pelo modelo orientado a objeto no final da década de 90, o modelo relacional incorporou as principais funcionalidades do conceito que envolve orientação a objeto. Com isto surge o modelo objeto relacional ou pós-relacional, que nada mais é do que o modelo relacional com funcionalidades da orientação a objetos. Atualmente boa parte dos principais bancos de dados relacionais apresentam funcionalidades do modelo orientado a objeto, mas isso não significa que os sistemas de informação estão sendo desenvolvidos utilizando-se deste novo paradigma de armazenamento e gerenciamento de dados.

Na continuidade, exploramos o SGBDR, onde você teve a oportunidade de entender as diferenças entre o processamento em arquivos convencionais e a gerência realizada por um sistema gerenciador. Conhecemos as 13 regras de Codd e as principais funções de um SGBDR.

Na sequência discutimos acerca da arquitetura de um SGBDR, destacando características dos níveis externo, conceitual e interno. Falamos também dos tipos de usuários e seus respectivos enquadramentos. Finalizamos os estudos deste Ciclo falando um pouco de quatro SGBDRs existentes no mercado: MySQL, ORACLE, PostgreSQL e Firebird.

E é isso... Esperamos que tenha gostado! O próximo Ciclo dos estudos estará focado na temática projeto de banco de dados, cujo objetivo principal será desenvolver em você habilidades e competências para a criação de modelos de bases de dados relacionais. Até lá!

## Referências

BEIGHLEY, Lynn. **Use a cabeça SQL**. Rio de Janeiro: Alta Books, 2008.

DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Elsevier, Campus, 2004.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Education do Brasil; Addison Wesley, 2012.

GARCIA-MOLINA, H. **Implementação de sistemas de bancos de dados**. Rio de Janeiro: Campus, 2001.

KHOSHAFIAN, Setrag. **Banco de dados orientado a objeto**. Rio de Janeiro: Infobook, 1994.

LISBOA, F. **Zend Framework**: desenvolvendo em PHP 5 orientado a objetos com MVC. São Paulo: Novatec, 2008.

MULLER, Robert J. **Projeto de banco de dados**: usando UML para modelagem de dados. São Paulo: Berkeley, 2002.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados**. São Paulo: McGraw-Hill, 2008.

SILBERSCHATZ, A. **Sistema de banco de dados**. Rio de Janeiro: Elsevier, 2012.

SILVA, L. **Banco de dados para web**: do planejamento à implementação. São Paulo: Érica, 2001.