SISTEMA GERENCIADOR DE BANCO DE DADOS

AULA 1

Profa. Vívian Ariane Barausse de Moura

CONVERSA INICIAL

Segue a apresentação da aula com a estrutura de conteúdos que serão trabalhados em tópicos:

1. Normalização
2. 1FN e 2FN
   - 2.1 Primeira forma normal
   - 2.2 Segunda forma normal
3. 3FN, 4FN e 5FN
   - 3.1 Terceira forma normal
   - 3.2 Quarta forma normal
   - 3.3 Quinta forma normal
4. Tipos de banco de dados e os principais SGBDs utilizados
5. Arquitetura básica de um SGBD e aspectos operacionais

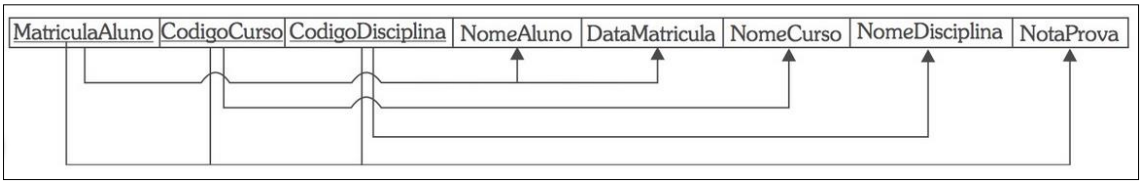
O objetivo da aula é introduzir os principais conceitos sobre normalização, a partir do conceito de dependência funcional e sua importância na modelagem de um banco de dados e detalhar a aplicação das formas normais durante o processo de normalização do banco de dados. Assim como tipos de banco de dados e os principais SGBDs utilizados e a arquitetura básica de um SGBD e aspectos operacionais.

TEMA 1 – NORMALIZAÇÃO

Alves (2014, p. 107) defende que, para entender os conceitos referentes à normalização e às formas normais, é fundamental assimilar o que é a dependência funcional, conceito muito importante em bancos de dados relacionais. A dependência funcional “consiste numa restrição entre dois conjuntos de atributos de uma mesma entidade/relação. Uma dependência funcional é representada pela expressão **X** → **Y**, em que **X** e **Y** são subconjuntos de atributos de uma relação qualquer” (Alves, 2014, p. 107). De acordo com Alves (2014, p. 107) “isso impõe uma restrição na qual um componente Y de um registro é dependente de um valor do componente X (ou é determinado por ele). Do mesmo modo, os valores do componente X determinam de forma unívoca os valores do componente Y. Resumindo, Y é dependente funcionalmente de X”.

Alves (2014) traz como exemplo o esquema representado na Figura 1, em que os três primeiros atributos, cujos nomes se encontram sublinhados, representam a chave primária da relação (nesse caso, uma chave composta).

Figura 1 – Esquema de uma relação



Fonte: Alves, 2014, p. 108.

Nessa representação, é possível estabelecer quatro dependências funcionais.

Figura 2 – Representação das dependências

1) MATRICULAALUNO → {NOMEALUNO,DATAMATRICULA}: o valor do atributo MATRICULAALUNO determina o valor de outros dois atributos, a saber: NOMEALUNO e DATAMATRICULA;

2) CODIGOCURSO → NOMECURSO: o valor do atributo CODIGOCURSO determina o valor do atributo NOMECURSO;

3) CODIGODISCIPLINA → NOMEDISCIPLINA: o valor do atributo CODIGODISCIPLINA determina o valor do atributo NOMEDISCIPLINA;

4) {MATRICULAALUNO,CODIGOCURSO,CODIGODISCIPLINA} → NOTAPROVA: a combinação de valores dos atributos MATRICULAALUNO, CODIGOCURSO e CODIGODISCIPLINA determina o valor do atributo NOTAPROVA.

Fonte: Alves, 2014, p. 108.

Assim, Alves (2014) classifica a dependência funcional em três categorias: total (ou completa), parcial e transitiva. No primeiro tipo, a dependência só existe se a chave primária composta por vários atributos determinar univocamente um atributo ou conjunto de atributos. No exemplo de Alves (2014, p. 108), da Figura 1, “o atributo NOTAPROVA é dependente total da chave primária composta formada pelos atributos MATRICULAALUNO, CODIGOCURSO e CODIGODISCIPLINA”.

Se ocorrer de o atributo ou conjunto de atributos possuir dependência apenas de parte dos valores da chave primária, então tem-se uma dependência parcial (segundo tipo), pois, de acordo com Alves (2014, p. 108), “na dependência transitiva, existe uma situação em que um atributo/conjunto de

atributos depende de outro atributo/ conjunto de atributos que não fazem parte de uma chave primária".

Nesse sentido, Alves (2014, p. 109) define a normalização como "um processo de refinamento do esquema de banco de dados, procurando eliminar possíveis redundâncias (dados repetidos em entidades), sanar problemas de dependências parciais entre atributos e reduzir ao mínimo as anomalias de inclusão, alteração e exclusão".

Conforme defendem Ramakrishnan e Gehrke (2008, p. 514), dado um esquema de relação, é necessário decidir se ele é um bom projeto ou se precisa decompô-lo em relações menores. Tal decisão deve ser conduzida por um entendimento de quais problemas (se houver) surgem a partir do esquema corrente. Para fornecer tal condução, diversas formas normais foram propostas. Se um esquema de relação está em uma dessas formas normais, sabemos que certos tipos de problemas não podem surgir.

Esse processo é dividido em várias etapas, denominadas *formas normais*, nas quais são efetuados diversos testes com o objetivo de se certificar que o esquema satisfaz determinadas condições presentes em cada forma normal. Com base nesses testes, as relações são decompostas em relações menores conforme a necessidade (Alves, 2014).

Por exemplo, "uma relação pode ser dividida em duas ou mais, dependendo da situação. A forma normal de uma relação indica o grau de normalização em que ela se encontra" (Alves, 2014, p. 109). De acordo com o autor, "existem cinco formas normais, embora apenas as três primeiras já sejam suficientes para se ter uma boa definição da estrutura do banco de dados. No fim da normalização, responde à principal pergunta no início de um projeto" (Alves, 2014, p. 108) e ressalta a questão "quantas tabelas são necessárias em nosso banco de dados?".

"As regras de normalização permitem que bancos de dados robustos e eficientes possam ser criados e facilmente alterados. Se essas regras forem seguidas com cuidado, o sistema todo (banco de dados e aplicativo) será bastante flexível e confiável" (Alves 2014, p. 110). O referido autor defende que podem ser utilizados dois tipos de abordagens/metodologias no processo de normalização de um banco de dados, de acordo com o Quadro 1.

Quadro 1 – Metodologias no processo de normalização de um BD

De cima para baixo (*top-down*)	De baixo para cima (*bottom-up*)
Trabalha com agrupamentos de atributos em relações já definidas a partir do projeto conceitual. Uma análise é, então, aplicada a essas relações, decompondo-as em entidades e relacionamentos até serem atingidas as propriedades desejadas para a implementação física.	É um processo inverso ao anterior, que considera os relacionamentos entre os atributos o ponto de partida para o processo, utilizando-os na construção das relações. Também é denominado de projeto por síntese.

Fonte: Alves, 2014, p. 110.

Para representar os exemplos das formas normais, será utilizado como exemplo a ficha de pedido mostrada na Figura 3.

Figura 3 – Exemplo ficha de pedido

Pedido nº.: 00347

Código Cliente: 00341 Nome Cliente: Francisco Albano de Moura

Endereço: Av. Nove de Julho, 193 - Jd. Paulista

Cidade: São Paulo UF: SP

Código	Descrição	Quant.	Preço Unit.	Valor Total
97892567	Caderno Universitário Espiral 5 Mat./200 Fls.	2	8,00	16,00
977987	Caneta Esferográfica Azul	1	1,50	1,50
977990	Caneta Esferográfica Vermelha	1	1,50	1,50
85146879	Papel Sulfite A4	1	12,00	12,00
564779	CD-ROM Virgem Gravável (CDR)	10	1,50	15,00

Código Vendedor: 001 Nome Vendedor: Álvaro Total do Pedido: 46,00

Fonte: Alves, 2014, p. 110.

Em uma primeira análise, é perceptível que o exemplo contém várias informações que se encontram misturadas: nome e endereço do cliente, lista de produtos e identificação do vendedor, e se destacam os atributos relacionados na Tabela 1.

Tabela 1 – Atributos do exemplo ficha de pedidos

• Número do pedido;	• Código do produto;	• Nome do vendedor;
• Nome do cliente;	• Descrição do produto;	• Preço unitário do produto;
• Endereço do cliente (rua, bairro, cidade e estado);	• Quantidade a ser vendida;	• Valor total do item (preço unitário X quantidade);
		• Valor total do pedido.

Fonte: Alves, 2014, p. 111.

O autor destaca que os produtos aparecem repetidos na ficha de pedido, e que se essas informações fossem armazenadas em um banco de dados sem ter sido feita uma prévia organização, o resultado seria similar ao que se pode ver na Figura 3 (por uma questão de visualização a tabela foi subdividida).

Figura 4 - Exemplo de tabela com dados não normalizados

Número Pedido	Cliente	Endereço	Cidade	UF
000347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP
000347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP
000347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP
000347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP
000347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP

CONTINUAÇÃO DA TABELA

Código Produto	Descrição Produto	Quantidade	Preço Unitário	Valor Total	Vendedor
97892567	Caderno Universitário Espiral 5 Mat./200 Fls.	2	8,00	16,00	Álvaro
977987	Caneta Esferográfica Azul	1	1,50	1,50	Álvaro
977990	Caneta Esferográfica Vermelha	1	1,50	1,50	Álvaro
85146879	Papel Sulfite A4	1	12,00	12,00	Álvaro
566779	CD-ROM Virgem Gravável (CDR)	10	1,50	15,00	Álvaro

Fonte: Alves, 2014, p. 111.

A partir da Figura 4, exemplo de tabela com dados não normalizados, verifica-se as anomalias representadas na Tabela 2.

Tabela 2 – Anomalias a partir do exemplo

• O cliente é relacionado junto com a venda.
• Se uma venda inteira for apagada, os dados do cliente também serão.
• No caso de exclusão de produto, todos os registros que contiverem informação sobre ele devem ser apagados.
• Se houver alteração nos dados do cliente, como endereço, é necessário refletir essa alteração em todos os registros em que ele aparecer.
• Da mesma forma, se houver alteração em um produto, como preço. Essa alteração também deve ser repassada a todos os registros desse produto constantes no arquivo.

Fonte: Alves, 2014, p. 111.

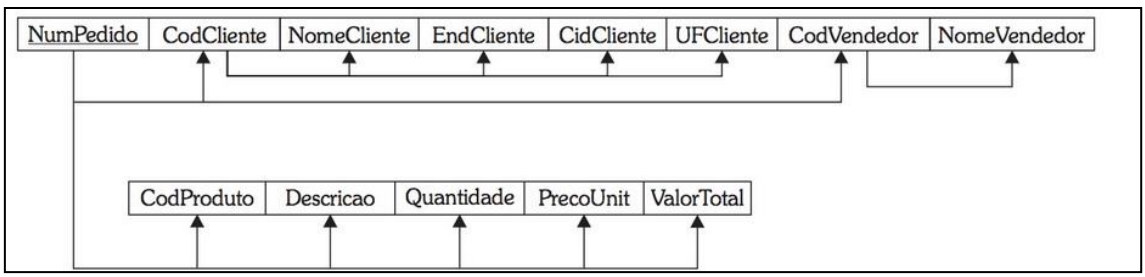
Conforme salienta Alves (2014) “do jeito que está é simplesmente impraticável. Assim, é necessário aplicar o processo de normalização a esse conjunto de dados”. Nos próximos assuntos, vamos abordar essas questões.

TEMA 2 – 1FN e 2FN

2.1 Primeira forma normal (1FN)

Com base na análise da Figura 4, nota-se que o atributo **Número Pedido** se repete a cada linha. De acordo Alves (2014), na regra da 1FN, isso não é permitido, “pois ela define que os valores dos atributos que formam a chave primária da relação não devem se repetir, tampouco a relação deve possuir valores de atributos que sejam multivalorados ou compostos”. Os atributos devem conter apenas valores atômicos (não divisíveis) e únicos (não multivalorados). Assim, para converter uma entidade não normalizada na **1FN**, é necessário decompô-la em tantas entidades quantas forem necessárias para não haver itens repetidos. Inicialmente, cria-se o esquema da entidade **Pedido**, conforme Figura 5, a chave primária da relação é o atributo **NumPedido**, que aparece sublinhado na Figura.

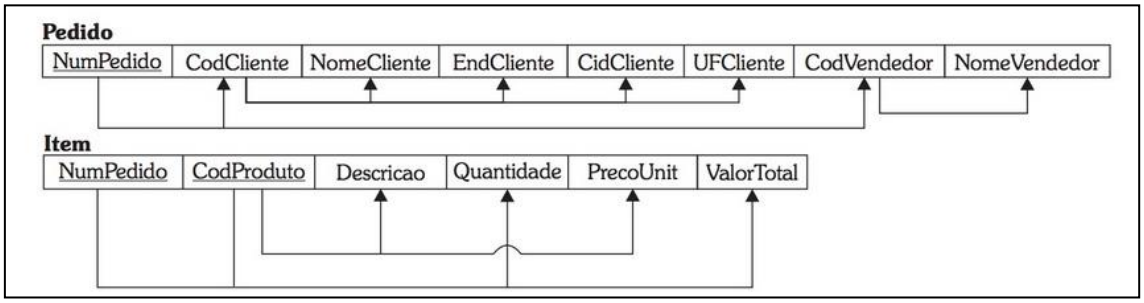
Figura 5 – Esquema da entidade **Pedido**



Fonte: Alves, 2014, p. 112.

Os atributos que se repetem são removidos, que, nesse caso, são **CodProduto, Descricao, Quantidade, PrecoUnit e ValorTotal**. Fazendo essa decomposição, chega-se a duas entidades, uma para a ficha de pedido e outra para os produtos que constam nela, conforme as figuras 6 e 7 (Alves, 2014).

Figura 6 – Aplicação da 1FN



Fonte: Alves, 2014, p. 112.

Figura 7 – Entidades resultantes após aplicação da 1FN

Entidade: Pedido

Número Pedido	Cliente	Endereço	Cidade	UF	Vendedor
00347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP	Álvaro

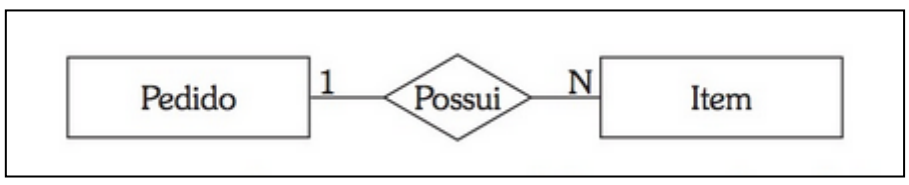
Entidade: Item

Número Pedido	Código Produto	Descrição Produto	Quantidade	Preço Unitário	Valor Total
000347	97892567	Caderno Universitário Espiral 5 Mat./200 Fls.	2	8,00	16,00
000347	977987	Caneta Esferográfica Azul	1	1,50	1,50
000347	977990	Caneta Esferográfica Vermelha	1	1,50	1,50
000347	85146879	Papel Sulfite A4	1	12,00	12,00
000347	566779	CD-ROM Virgem Gravável (CDR)	10	1,50	15,00

Fonte: Alves, 2014, p. 112.

Conforme Alves (2014, p. 112), o relacionamento entre essas duas entidades é mostrado na Figura 8. Uma vez que um pedido pode conter vários produtos, existe, então, uma relação **1:N** (um-para-muitos). A chave primária da entidade **Pedido** é o número do pedido, que é único para cada registro. Já para a entidade **Item** há uma chave primária formada pelo número do pedido e o código do produto.

Figura 8 – Relacionamento Pedido X Item do pedido



Fonte: Alves, 2014, p. 112.

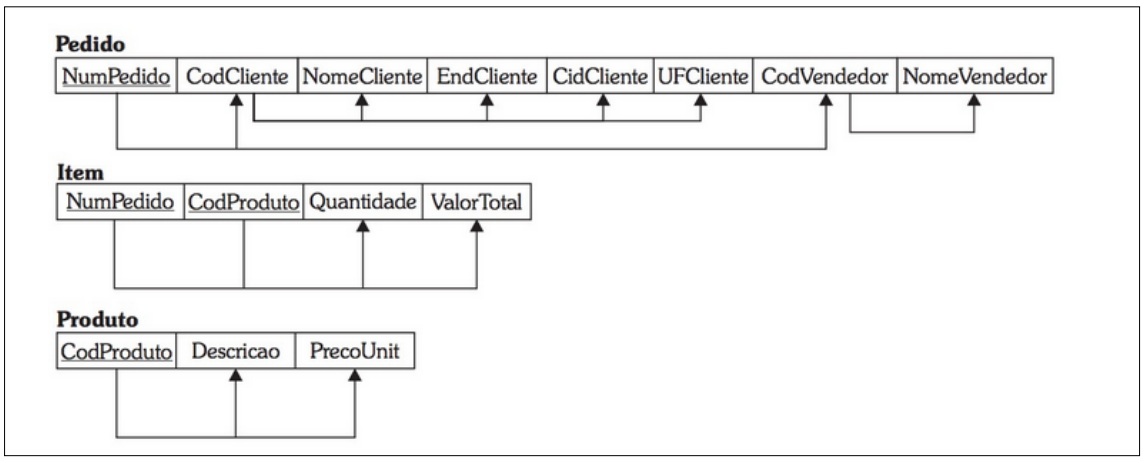
2.2 Segunda forma normal (2FN)

De acordo com Alves (2014, p. 113), colocar “as entidades na segunda forma normal é um pouco mais complicado”, por causa das dependências funcionais, já que ela é baseada na dependência funcional total, implica em **X** → **Y** se um atributo de **X** for removido, então **Y** não pode ser determinado de forma unívoca.

“A entidade se encontra na segunda forma normal se, além de estar na primeira, todos os seus atributos são totalmente dependentes da chave primária composta” (Alves, 2014, p. 113). Significa que atributos que são parcialmente dependentes devem ser removidos.

Tendo como base o exemplo, a entidade **Item** possui chave primária composta, que é constituída pelos atributos **NumPedido** e **CodProduto**. Já os atributos **Descricao** e **PrecoUnit** não dependem totalmente dessa chave, ao contrário de **Quantidade** e **ValorTotal**. Assim, uma terceira entidade denominada **Produto** deve ser definida, conforme representado nas figuras 9 e 10.

Figura 9 – Aplicação da 2FN



Fonte: Alves, 2014, p. 113.

Figura 10 – Entidades resultantes após aplicação da 2FN

Entidade: Pedido

Número Pedido	Cliente	Endereço	Cidade	UF	Vendedor
00347	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP	Álvaro

Entidade: Item

Número Pedido	Código Produto	Quantidade	Valor Total
000347	97892567	2	16,00
000347	977987	1	1,50
000347	977990	1	1,50
000347	85146879	1	12,00
000347	566779	10	15,00

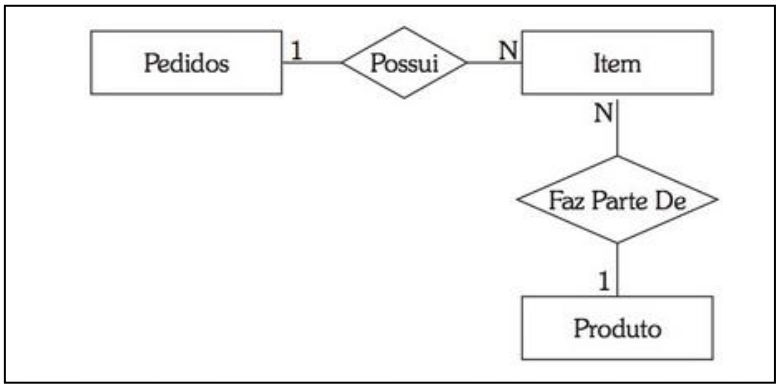
Entidade: Produto

Código Produto	Descrição Produto	Preço Unitário
97892567	Caderno Universitário Espiral 5 Mat./200 Fls.	8,00
977987	Caneta Esferográfica Azul	1,50
977990	Caneta Esferográfica Vermelha	1,50
85146879	Papel Sulfite A4	12,00
566779	CD-ROM Virgem Gravável (CDR)	1,50

Fonte: Alves, 2014, p. 113.

O diagrama ER agora deve se parecer conforme a Figura 11.

Figura 11 – Relacionamentos Pedido X Item e Item X Produto



Fonte: Alves, 2014, p. 114.

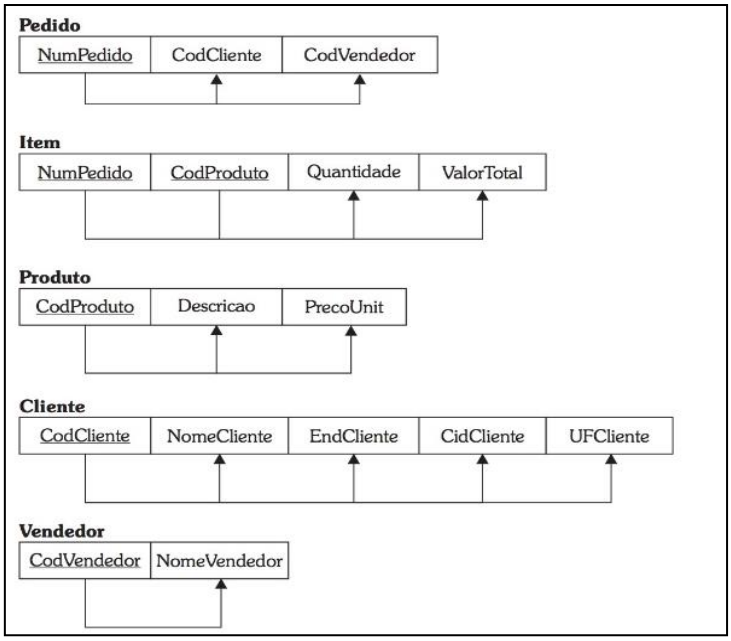
TEMA 3 – 3FN, 4FN e 5FN

3.1 Terceira Forma Normal (3FN)

Para que uma entidade estar na terceira forma normal, de acordo com Alves (2014, p. 114), “é preciso que ela já esteja na segunda e não possua nenhum atributo dependente de outro que não faz parte da chave primária (dependência transitiva)”.

No exemplo de Alves (2014, p. 114) “no pedido existem atributos que identificam um cliente (nome do cliente e endereço completo) e um vendedor (nome do vendedor). Os dados dos atributos **NomeCliente, EndCliente**, **CidCliente** e **UFCliente** são dependentes de **CodCliente**, assim, é possível criar outra entidade, que pode ser denominada de **Cliente**. Da mesma forma, **NomeVendedor** é dependente de **CodVendedor**”. As figuras 12 e 13 apresentam a decomposição dessas relações.

Figura 12 – Aplicação da 3FN



Fonte: Alves, 2014, p. 114.

Figura 13 – Entidades resultantes após aplicação da 3FN

Entidade: Pedido

Número Pedido	Código Cliente	Código Vendedor
00347	00341	001

Entidade: Item

Número Pedido	Código Produto	Quantidade	Valor Total
000347	97892567	2	16,00
000347	977987	1	1,50
000347	977990	1	1,50
000347	85146879	1	12,00
000347	566779	10	15,00

Entidade: Produto

Código Produto	Descrição Produto	Preço Unitário
97892567	Caderno Universitário Espiral 5 Mat./200 Fls.	8,00
977987	Caneta Esferográfica Azul	1,50
977990	Caneta Esferográfica Vermelha	1,50
85146879	Papel Sulfite A4	12,00
566779	CD-ROM Virgem Gravável (CDR)	1,50

Entidade: Cliente

Número Pedido	Cliente	Endereço	Cidade	UF
00341	Francisco Albano de Moura	Av. Nove de Julho, 193 - JD. Paulista	São Paulo	SP

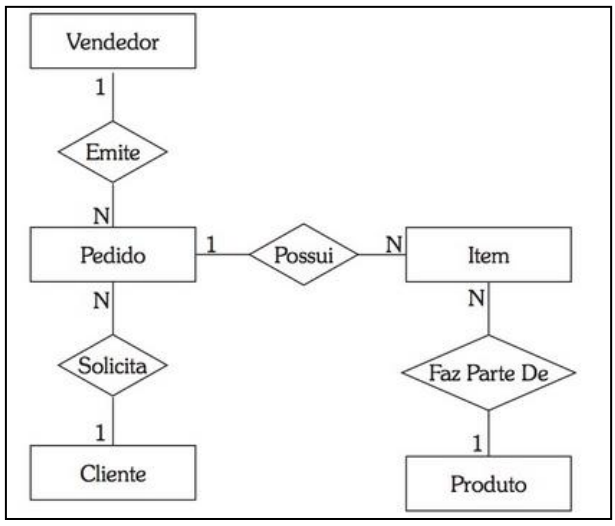
Entidade: Vendedor

Código Vendedor	Vendedor	Nome Completo
001	Álvaro	Álvaro de Moraes

Fonte: Alves, 2014, p. 115.

O diagrama ER corresponde à Figura 15

Figura 15 – Diagrama de entidade-relacionamento final após normalização



Fonte: Alves, 2014, p. 115.

3.2 Quarta forma normal (4FN)

Temos, com base no exemplo da ficha de pedidos, que as entidades se encontram normalizadas na 3FN, de acordo com Alves (2014) e Ramakrishnan e Gehrke (2008), porém pode ocorrer de uma entidade conter um ou mais fatos multivalorados, como na Figura 16.

Figura 16 – Tabela da relação de fornecedores, produtos e clientes

CodigoFornecedor	CodigoProduto	CodigoCliente
1024	12980	0001
1024	01830	0001
1024	10880	0001
1024	12980	0002
1024	01505	0002
2048	08501	0003
8512	15715	0004
8512	01830	0004

Fonte: Alves, 2014, p. 116.

Alves (2014) aponta que nessa relação existem produtos comprados pelos clientes e os fornecedores que abastecem a loja. Ocorrendo duas dependências: uma entre os atributos **CodigoFornecedor** e **CodigoProduto**, e outra entre o **CodigoCliente** e **CodigoProduto**. A atualização dessa entidade, nesse caso, torna-se difícil, apesar de ela estar na 3FN.

Nesse caso entra a quarta forma normal – 4FN –, segundo a qual uma relação deve, além de estar na 3FN, conter apenas um fato multivalorado – dependência multivalorada. Note que isso se refere a um fato multivalorado e não a um atributo multivalorado. Essa dependência multivalorada é uma consequência natural da 1FN, pois ela não permite conjunto de valores para um atributo.

Essa característica pode levar à seguinte situação: necessidade de repetir o valor de um dos atributos com o valor de outro atributo, quando houver dois ou mais atributos multivalorados independentes. A relação representada na Figura 16 deve ser dividida em duas para estar, então, na 4FN, conforme representado no Quadro 2.

Quadro 2 – Relações estabelecidas 4FN

<table>
<tr><th>Relação Fornecedor-Produto</th><th>Relação Cliente-Produto</th></tr>
<tr><td>Relação de fornecedor com produtos fornecidos
<table>
<tr><th>CodigoFornecedor</th><th>CodigoProduto</th></tr>
<tr><td>1024</td><td>12980</td></tr>
<tr><td>1024</td><td>01830</td></tr>
<tr><td>1024</td><td>10880</td></tr>
<tr><td>1024</td><td>12980</td></tr>
<tr><td>1024</td><td>01505</td></tr>
<tr><td>2048</td><td>08501</td></tr>
<tr><td>512</td><td>15715</td></tr>
<tr><td>512</td><td>01830</td></tr>
</table></td><td>Relação de cliente com produto comprado
<table>
<tr><th>CodigoCliente</th><th>CodigoProduto</th></tr>
<tr><td>0001</td><td>12980</td></tr>
<tr><td>0001</td><td>01830</td></tr>
<tr><td>0001</td><td>10880</td></tr>
<tr><td>0002</td><td>12980</td></tr>
<tr><td>0002</td><td>01505</td></tr>
<tr><td>0003</td><td>08501</td></tr>
<tr><td>0004</td><td>15715</td></tr>
<tr><td>0004</td><td>01830</td></tr>
</table></td></tr>
</table>

Fonte: Alves, 2014, p. 116-117.

3.3 Quinta forma normal (5FN)

Alves (2014) e Ramakrishnan e Gehrke (2008) descrevem que a 5FN é a mais difícil de entender, pois ela lida com relacionamentos múltiplos (ternário, quaternário etc.). “Uma entidade está na 5FN se, estando na 4FN, não for possível reconstruir as informações originais a partir do conteúdo dos outros registros menores” (Alves, 2014, p. 117). Embora seja importante e útil o processo de normalização, em certas situações, é preciso fugir às regras de uma ou mais formas normais, visando aprimorar o desempenho do sistema.

Para explicar essa relação, Alves (2014, p. 117), supõe o seguinte exemplo:

> Suponha como exemplo o sistema de uma loja de materiais elétricos, em que são vendidos materiais como fios, fusíveis, lâmpadas etc., e também o que costuma ser chamado de “padrões de entrada do consumidor”. Esses padrões são um tipo de produto, mas formado por um poste de concreto, uma caixa de medidor, “bengala” (conduíte com uma das extremidades curva) etc. Assim, quando de uma venda de padrão, o sistema deve baixar o estoque de cada um dos seus componentes. Cada um desses componentes pode ser comprado pela loja individualmente, o que significa que eles podem constar de vários pedidos de compra.

A partir deste exemplo, o autor representa a entidade na qual se encontram dados para compra de materiais dos fornecedores.

Figura 16 – Relação de produtos comprados e seus respectivos pedidos e fornecedores

Produto	Pedido de Compra	Fornecedor
Padrão B2	07801	00341
Padrão B2	07801	00108
Poste Duplo T 9M	07801	00108
Padrão B2	07802	00108

Fonte: Alves, 2014, p. 117

Se essa relação for desmembrada, o resultado deve ser as seguintes entidades, representadas no Quadro 3.

Quadro 3 – Entidades

Entidade Produto-Pedido de Compra Tabela

Relação de produtos com seus respectivos pedidos

Produto	NumeroPedido
Padrão B2	07801
Padrão B2	07802
Poste Duplot T 9M	07801

Entidade Pedido de Compra-Fornecedor

Relação de pedidos e respectivos fornecedores

NumeroPedido	Fornecedor
07801	00341
07801	00108
07802	00108

Entidade Produto-Fornecedor Fornecedor

Relação de produtos e seus respectivos fornecedores

Produto	Fornecedor
Padrão B2	00341
Padrão B2	00108
Poste Duplo T 9M	00108

Fonte: Alves, 2014, p. 118.

Se for realizada uma junção das três entidades representadas no Quadro 3, utilizando o atributo **NumeroPedido** como o determinante, resultado poderá ser o representado na Figura 17.

Figura 17 – Junção das entidades fornecedor, produto e pedido com base no número do pedido

NumeroPedido	Produto	Fornecedor
07801	Padrão B2	00341
07801	Padrão B2	00108
07801	Poste Duplo T 9M	00108
07801	Poste Duplo T 9M	00341
07802	Padrão B2	00108

Fonte: Alves, 2014, p. 118.

Alves (2014) destaca o penúltimo registro, que não havia na relação original. Se a junção for efetuada tomando o atributo **Fornecedor**, o resultado será o representado na Figura 18.

Figura 18 – Junção das entidades fornecedor, produto e pedido com base no código do fornecedor

Fornecedor	Produto	NumeroPedido
00108	Padrão B2	07801
00108	Poste Duplo T 9M	07801
00108	Padrão B2	07802
00341	Padrão B2	07801
00341	Poste Duplo T 9M	07801

Fonte: Alves, 2014, p. 118.

De acordo com Alves (2014, p. 118) “novamente um registro que não existia na relação original é gerado. Durante o processo de normalização de dados você descobre duas coisas diametralmente opostas: o número de campos das tabelas diminui enquanto o número de tabelas aumenta”. Isso, no entanto, não deve ser algo com que se preocupar, pois um projeto de banco de dados bem elaborado geralmente contém um número grande de tabelas cuja estrutura é bastante simples.

TEMA 4 – TIPOS DE BANCO DE DADOS E PRINCIPAIS SGBDs UTILIZADOS

De acordo com Ramakrishnan e Gehrke (2008, p. 642), os sistemas de banco de dados relacional é representado por coleções de relações, que, no mundo real, assumem a forma de tabelas de registros. Esse modelo procura representar os dados e os relacionamentos existentes entre eles por meio de

uma coleção de tabelas e suportam uma pequena coleção fixa de tipos de dados (por exemplo, inteiros, *datas*, *strings*), os quais têm se mostrado adequados para domínios de aplicação tradicionais como o processamento de dados administrativo. Entretanto, em muitos domínios de aplicação, tipos muito mais complexos precisam ser manipulados.

De acordo com o autor, para suportar as aplicações existentes, um SGBD precisa aceitar tipos de dados complexos. Os conceitos de orientação a objetos influenciaram fortemente os esforços para melhorar o suporte de banco de dados para dados complexos e levaram ao desenvolvimento de sistemas de banco de dados de objetos, que foram desenvolvidos ao longo de dois caminhos distintos: sistemas de banco de dados orientados a objetos e sistemas de banco de dados objeto-relacionais. (Ramakrishnan; Gehrke, 2008).

Ramakrishnan e Gehrke (2008, p. 643)

> defende os sistemas de banco de dados orientados a objetos foram propostos como uma alternativa aos sistemas relacionais e se destinam aos domínios de aplicação onde os objetos complexos desempenham um papel fundamental. A estratégia é fortemente influenciada pelas linguagens de programação orientadas a objetos e pode ser entendida como uma tentativa de acrescentar funcionalidade de SGBD em um ambiente de linguagem de programação.

Para os sistemas de banco de dados objeto-relacionais, Ramakrishnan e Gehrke (2008) defendem que podem ser considerados como uma tentativa de estender os sistemas de banco de dados relacionais com funcionalidade necessária para suportar uma classe mais ampla de aplicações e, de muitas formas, estabelecer uma ligação entre os paradigmas relacional e orientado a objetos.

Ramakrishnan e Gehrke (2008) e Alves (2014) concordam que a literatura usa acrônimos para os sistemas de gerenciamento de banco de dados relacionais, orientados a objetos e objeto-relacionais, respectivamente SGBDR, SGBDOO e SGBDOR.

Além disso, Alves (2014) aborda os fatores que distinguem um sistema de banco de dados distribuído de outros. Para o autor, o primeiro deles “é o grau de homogeneidade, que está diretamente relacionado, com os tipos de software utilizados em todo o sistema” (Alves, 2014, p. 139). Para que um SGBDD seja homogêneo, os servidores e os clientes utilizam *softwares* idênticos, de um mesmo fabricante ou fornecedor.

Se houver uma diversidade de *softwares*, tem-se um SGBDD heterogêneo. Nesse ambiente heterogêneo, deve haver uma convivência pacífica entre os equipamentos de diferentes marcas e modelos, mesmo nos casos em que até os sistemas operacionais são distintos. Essa capacidade de um aplicativo poder acessar informações armazenadas em outro sistema totalmente diferente define o que costuma ser chamado de *interoperabilidade* (Alves, 2014, p. 139).

Um segundo fator, conforme Alves (2014), é o grau de autonomia local, que se refere como a possibilidade de um SGBD que faz parte de o sistema distribuído ser capaz de trabalhar isoladamente (*stand-alone*). Se isso não for possível, então o SGBDD é considerado um sistema que não possui autonomia local; caso contrário, se o acesso direto de transações locais for possível, o sistema tem alguma autonomia local.

Mesmo que se utilize em todo o sistema um SGBD relacional, Alves (2014, p. 140) defende que é possível encontrar alguns problemas, como o fato de cada um utilizar uma versão diferente da linguagem de consulta, por exemplo, o SQL-89, o SQL-92 e o SQL3.

A forma como os dados se encontram modelados também é um fator de complexidade em um SGBDD heterogêneo. Isso ocorre por causa da variedade de bancos de dados existentes em uma organização cada qual seguindo uma modelagem diferente. Mesmo que se tenham bancos de dados rodando em um mesmo ambiente, há chances de atributos similares serem definidos com nomes ou tamanhos diferentes (Alves, 2014; Ramakrishnan; Gehrke, 2008).

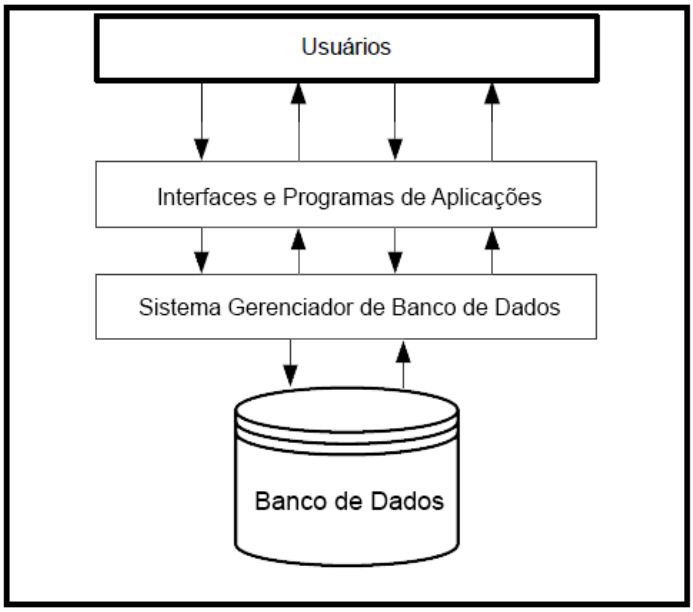
TEMA 5 – ARQUITETURA BÁSICA DE UM SGBD E ASPECTOS OPERACIONAIS

Para entender os aspectos referentes ao funcionamento do SGBD, é necessário assimilar alguns conceitos básicos sobre banco de dados. De acordo com Alves (2014, p. 17), um banco de dados é “um conjunto lógico e ordenado de dados que possuem algum significado, e construído e povoado com dados que têm um determinado objetivo, com usuários e aplicações desenvolvidas para manipulá-los”. O autor defende que o armazenamento pode ser em um ou mais dispositivos e o meio de armazenamento mais utilizado é o disco rígido.

Alves (2014, p. 17) afirma que “se um banco de dados é um conjunto de dados relacionados, um Sistema de Gerenciamento de Banco de Dados (SGBD)

é uma coleção de ferramentas e programas que permitem aos usuários a criação e manutenção do próprio banco de dados", conforme representado na Figura 19.

Figura 19 – Arquitetura simplificada



Fonte: Silberschatz et al., 2006, p. 17.

Dessa forma, de acordo com Alves (2014), o SGBD pode ser considerado como um sofisticado *software* destinado à definição, construção e manipulação, conforme representado no Quadro 4.

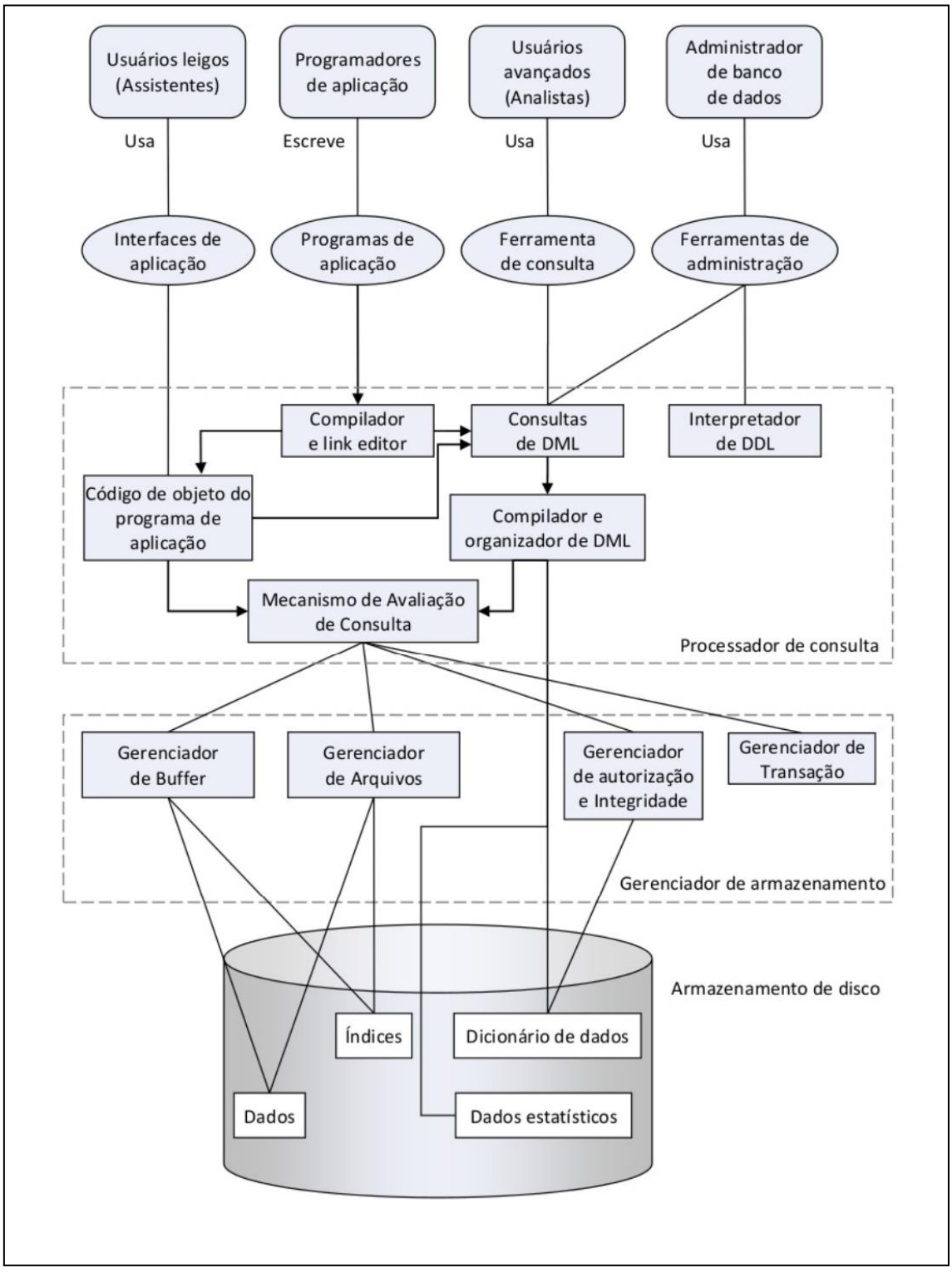
Quadro 4 – Funções de um SGBD

Definição	Especificação dos tipos de dados, das estruturas das tabelas e das restrições que devem ser impostas aos dados que serão armazenados.
Construção	Processo de acumular os dados num meio de armazenamento totalmente controlado pelo SGBD.
Manipulação	Operações como atualização do banco de dados (inclusão, exclusão e alteração de registros) e extração de dados, como consultas e relatórios impressos.

Fonte: Alves, 2014, p. 18.

Para explicar a relação entre os vários componentes, Silberschatz et al. (2006) desenvolveram uma representação da estrutura do sistema de componentes de um banco de dados.

Figura 20 – Estrutura do BD



Fonte: Silberschatz et al., 2006, p. 17.

Os SGBDs são sistemas projetados para gerir grandes volumes de informações e devem proporcionar um ambiente adequado e eficiente para armazenar e recuperar os arquivos. Para realizar esse gerenciamento, são necessárias estruturas de armazenamento e definição dos mecanismos para a

manipulação, além das questões relacionadas à segurança das informações, que vão desde problemas com o sistema como tentativa de acesso não autorizado (Silberschatz et al., 2006). A Figura 20 apresenta a estrutura de um BD, que atende a esses requisitos listados pelo autor, com mecanismos para manipular, armazenar e segurança das informações.

Silberschatz et al. (2006, p. 2) defendem que o "objetivo principal de um sistema gerenciador de banco de dados é proporcionar ao usuário uma visão abstrata dos dados, que podem ser vistas em três níveis: nível físico, nível lógico e os níveis de visão", conforme apresentado na Tabela 3.

Tabela 3 – Níveis

Nível físico	Nível lógico	Nível de visão
Descreve como os dados estão armazenados.	Descreve quais dados estão armazenados e quais os inter-relacionamentos entre eles	Descreve apenas parte do banco de dados através de visões diversas do mesmo banco de dados para proporcionar interações simplificadas ao usuário.

Fonte: Silberschatz et al., 2006, p. 6.

Os autores também descrevem que o esquema de banco de dados "é determinando por um conjunto de definições que são expressas pela linguagem de definição de dados (DDL – *Data Manipulation Language*)", sendo que seus comandos são compilados e armazenados em "tabelas armazenadas em um arquivo especial chamado dicionário de dados" (Silberschatz et al., 2006, p. 6). Para habilitar o acesso e realizar a manipulação desses dados pelo usuário, é utilizada a linguagem de manipulação de dados (DDL – *Data Definition Language*).

Conforme Silberschatz et al. (2006, p. 525) salientam, "a arquitetura de um BD é bastante influenciada pelo sistema de computador subjacente em que o sistema de dados é executado, principalmente por aspectos como operação em: rede, paralelismo e distribuição", apresentados no Quadro 5.

Quadro 5 – Aspectos inerentes à execução dos sistemas

Rede	As redes de computadores permitem que algumas tarefas sejam executadas em um sistema servidor e outras sejam executadas em sistemas cliente. Essa divisão de trabalho tem levado a um sistema de banco de dados cliente-servidor.
Paralelismo	O processamento paralelo dentro de um sistema de computador permite que as atividades do sistema de banco de dados sejam agilizadas, permitindo resposta mais rápida às transações, além de mais de transações por segundo. As consultas podem ser

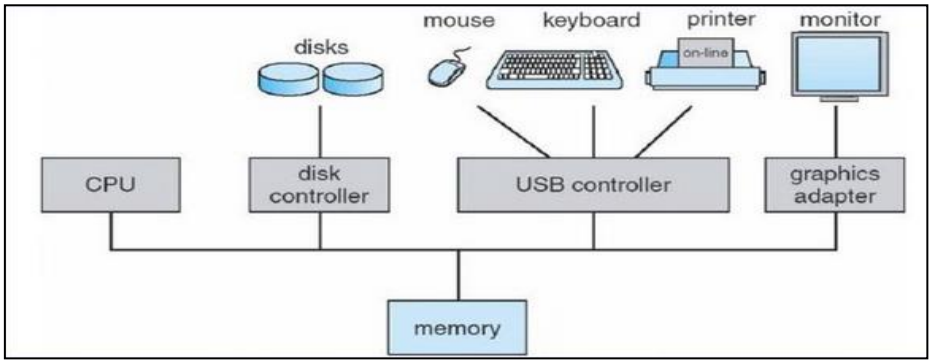
	processadas de um modo que explore o paralelismo oferecido pelo sistema de computador básico. A necessidade de processamento de consulta em paralelo levou a sistemas de banco de dados paralelos.
Distribuição	A distribuição de dados pelos sites em uma organização permite que esses dados residam onde são gerados ou onde são mais necessários, mas ainda precisem ser acessados a partir de outros sites e de outros departamentos. Manter várias cópias do banco de dados em diferentes locais também permite que grandes organizações continuem suas operações de banco de dados mesmo quando um site é afetado por um desastre natural, como inundação, incêndio ou terremoto. Os sistemas de banco de dados distribuídos tratam de dados distribuídos geográfica e administrativamente, espalhados por diversos sistemas de banco de dados.

Fonte: Silberschatz et al., 2006, p. 527.

O autor defende os sistemas de BD podem ser centralizados ou cliente-servidor, com a possibilidade de serem projetados para explorar arquiteturas de computador paralelas (Silberschatz et al., 2006).

O banco de dados centralizado executa em um único sistema de computador e não interagem com outros sistemas do computador. De acordo com Silberschatz et al. (2006, p. 527), “um sistema de computador moderno, de uso geral, consiste em uma ou poucas CPUs e uma série de controladores de dispositivos que estão conectados por meio de um barramento comum, que oferece memória compartilhada”, conforme Figura 21.

Figura 21 – Um sistema de computador centralizado

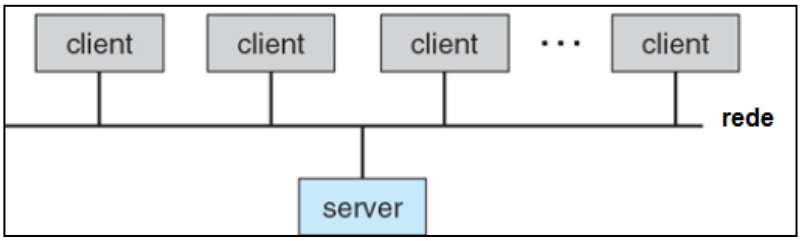


Fonte: Silberschatz et al., 2006, p. 528.

De acordo com Silberschatz et al. (2006), os computadores pessoais foram se tornando mais rápidos com mais capacidade e mais baratos e, com isso, ocorreu um afastamento da arquitetura de sistemas centralizados. “Os computadores pessoais suplantaram os terminais conectados ao sistema centralizado e assumiram a funcionalidade da interface com o usuário, que antes era tratada diretamente pelo sistema centralizado” (Silberschatz et al., 2006, p.

529). Como resultado, os sistemas centralizados de hoje atuam como sistemas servidores que atendem às solicitações geradas pelo sistema cliente, pois têm a funcionalidade dividida entre um sistema servidor e vários sistemas cliente. A Figura 22 apresenta a estrutura geral de um sistema baseado na plataforma cliente-servidor.

Figura 22 – Estrutura geral de um sistema baseado na plataforma cliente-servidor



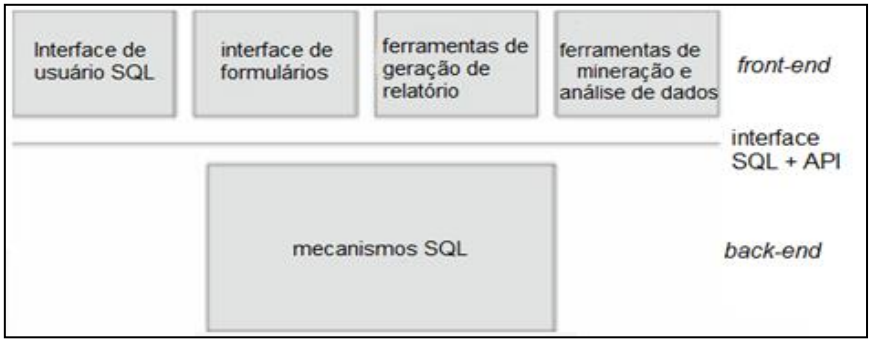
Fonte: Silberschatz et al., 2006, p. 529.

Nos sistemas cliente-servidor, Silberschatz et al. (2006) definem uma máquina servidora executando trabalho em nome de várias máquinas clientes, pois, de acordo com os autores, a maioria dos usuários de um sistema de banco de dados utiliza o serviço a partir da conexão com uma rede, não estando propriamente presente no local do sistema do BD.

A funcionalidade oferecida pelos sistemas de banco de dados pode ser dividida em duas partes: *back-end* e *front-end* (Figura 23).

> O *back-end* controla as estruturas de acesso, avaliação e otimização de consulta controle de concorrência e recuperação. O *front-end* de um sistema de banco de dados consiste em ferramentas como a interface com o usuário da SQL interface de formulário ferramentas e geração de relatório e ferramentas de mineração e análise de dados (Silberschatz et al., 2006, p. 529).

Figura 23 – Funcionalidades de *front-end* e *back-end*



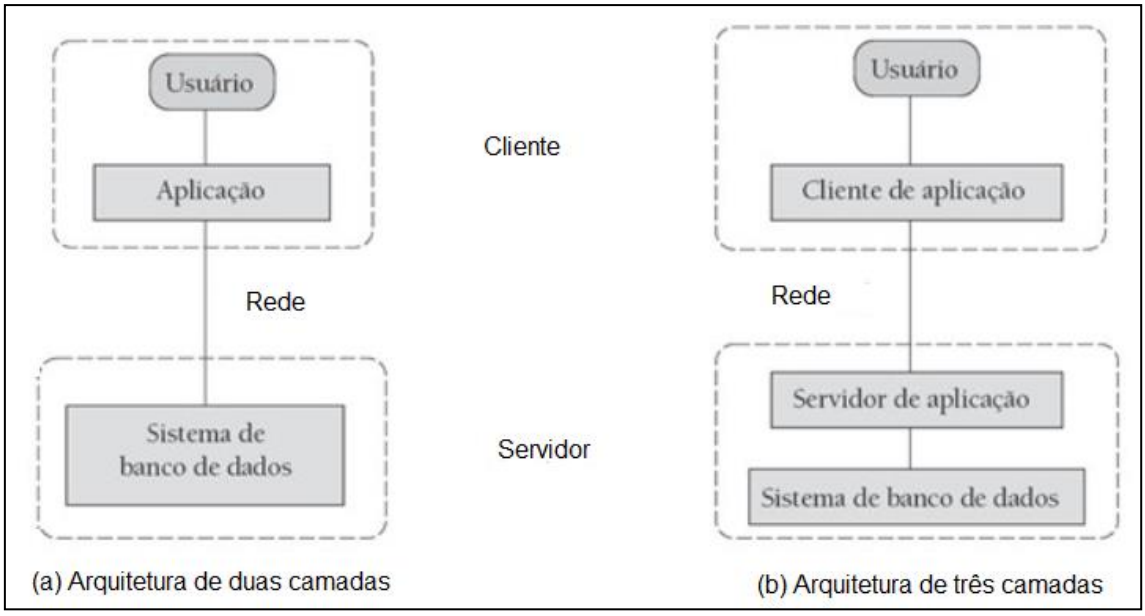
Fonte: Silberschatz et al., 2006, p. 529.

Silberschatz et al. (2006) apresentam a Figura 23 como a interface entre *front-end* e *back-end* por meio da SQL ou por meio de um programa de aplicativo. Padrões como ODBC e JDBC foram desenvolvidos para realizar a interface de clientes com servidores, qualquer cliente que usa interface ODBC e JDBC pode se conectar a qualquer servidor que ofereça a interface.

Alguns programas aplicativos, como planilhas e pacotes de análise estatística, utilizam a interface cliente-servidor diretamente para acessar dados de um servidor de *back-end* com o efeito eles oferecem *back-ends* especializados para tarefas específicas (Silberschatz et al., 2006).

De acordo com Silberschatz et al. (2006), as aplicações de banco de dados normalmente são particionadas em duas ou três partes, como na Figura 24.

Figura 24 – Arquitetura de duas camadas e três camadas



Fonte: Silberschatz et al., 2006, p. 17.

Na arquitetura de duas camadas, de acordo com Silberschatz et al., (2006), a aplicação é particionada em um componente que resiste na máquina cliente, que chama a funcionalidade do sistema de banco de dados na máquina servidora por meio de instruções da linguagem. Os padrões de interface de programa de aplicação, como ODBC e JDBC, são usados para interação entre o cliente e o servidor.

Na arquitetura de três camadas, Silberschatz et al. (2006) defendem que a máquina cliente age meramente como um *front-end* e não contém quaisquer

chamadas de banco de dados diretas. Em vez disso, o cliente finaliza a comunicação com um servidor de aplicação, normalmente por meio de uma interface de formulários. O servidor de aplicação, por sua vez, se comunica com um sistema de banco de dados para acessar os dados. A lógica empresarial de aplicação, que diz quais ações executar sob quais condições, é incorporada ao servidor de aplicações, em vez de ser distribuída por múltiplos clientes. As aplicações de três camadas são mais apropriadas para grandes aplicações e para aquelas executadas na *world wide web*.

Os sistemas que lidam com grande quantidade de usuários adotam uma arquitetura de três camadas, em que o *front-end* ainda é o navegador *web* que se comunica como um servidor de aplicação o servidor de aplicação, com efeito, atua como um cliente para o servidor de banco de dados.

FINALIZANDO

Nesta aula, aprendemos sobre a normalização e como ela pode eliminar a redundância, apresentamos as estratégias para normalizar uma relação com ênfase na sua importância para a modelagem de um banco de dados com a aplicação das formas normais durante o processo de normalização do banco de dados. Foram apresentados os tipos de banco de dados, os principais SGBDs utilizados e os aspectos operacionais e as terminologias quanto à arquitetura básica de um SGBD.

REFERÊNCIAS

ALVES, W. P. **Banco de dados**. São Paulo: Érica, 2014.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de bancos de dados**. 3. ed. Porto Alegre: McGraw Hill, 2008.

SILBERSCHATZ, A. et al. **Sistema de banco de dados**. Tradução de Daniel Vieira. Rio de Janeira: Elsevier, 2006.