BANCO DE DADOS NOSQL

AULA 1

Prof. Alex Mateus Porn

CONVERSA INICIAL

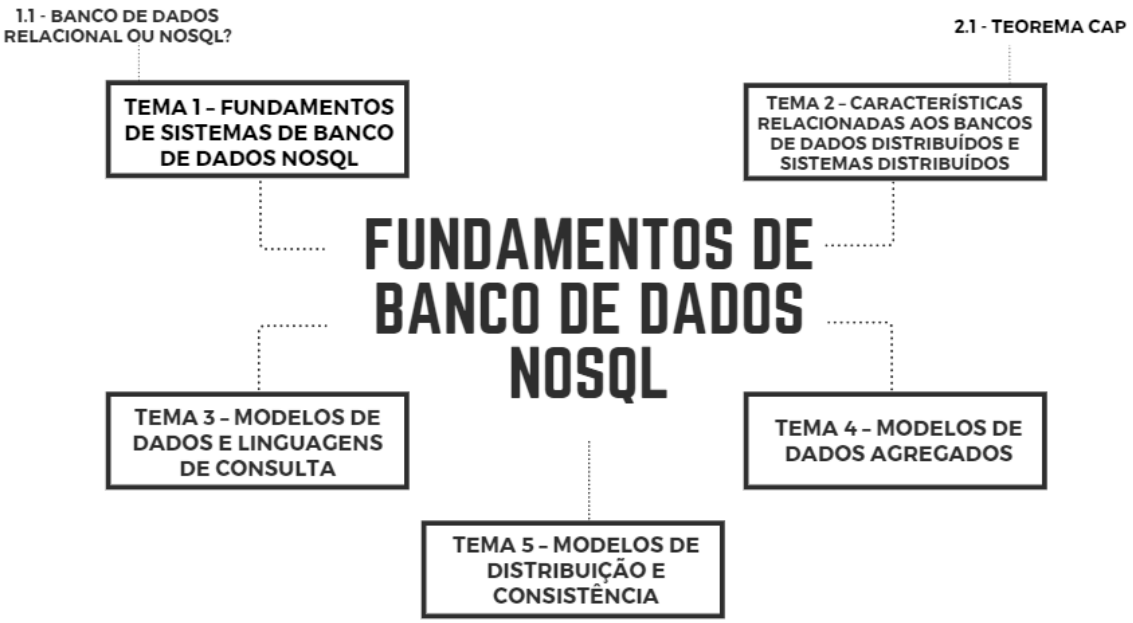
Olá! Nesta disciplina você vai estudar os conceitos relacionados ao universo de bancos de dados NoSQL.

Para começar, nesta aula serão apresentados os principais conceitos sobre sistemas de bancos de dados NoSQL e as suas diferentes categorias. Também será falado sobre os principais fatores que motivam o uso de um banco de dados NoSQL. Entre eles, destaca-se a obtenção de produtividade no desenvolvimento de aplicativos e a capacidade de processamento de grandes quantidades de dados, ou seja, alcançar escalabilidade.

Você aprenderá também que nos bancos de dados NoSQL os dados estão armazenados de forma isolada e em diferentes modelos, que não se tratam de tabelas e colunas como é o caso do modelo relacional, mas são novos tipos de estruturas, como chave-valor, documentos, colunas e grafos. Outra característica interessante é que os bancos de dados NoSQL apresentam esquemas de dados flexíveis, por meio da definição da agregação de dados. Essa agregação possibilita a adição de novos campos aos registros. Por fim, entenderá como é o comportamento de distribuição e consistência de dados.

Ao longo desta aula, serão trabalhados os seguintes conteúdos:

Figura 1 – Conteúdos



A computação, em linhas gerais, é caracterizada por apresentar constantes mudanças e progresso nos recursos disponíveis, sejam esses relacionados a software ou a hardware. No entanto, Sadalage e Fowler (2019, p. 13) relatam que ao longo de todos esses avanços, os bancos de dados relacionais conceitualmente se mostram estáveis, armazenando dados de forma estruturada e consolidada.

Entretanto, Elmasri (2018, p. 795) expõe que nos dias de hoje temos novas necessidades que requerem manipulação e gerenciamento de grandes volumes de dados, como ocorre em aplicações de mídias sociais, links da web, postagens em geral, entre outros.

Figura 2 – Mídias sociais

Créditos: solomon7/Shutterstock.

Como consequência dessa demanda, em 2009 foram manifestadas iniciativas provindas de vários projetos que buscavam armazenamento alternativo de dados, sendo esses apresentados em uma reunião envolvendo importantes desenvolvedores globais. A esta reunião, deu-se o nome de NoSQL, sendo desde então utilizado esse termo para denominar esse novo movimento de bancos de dados não relacionais.

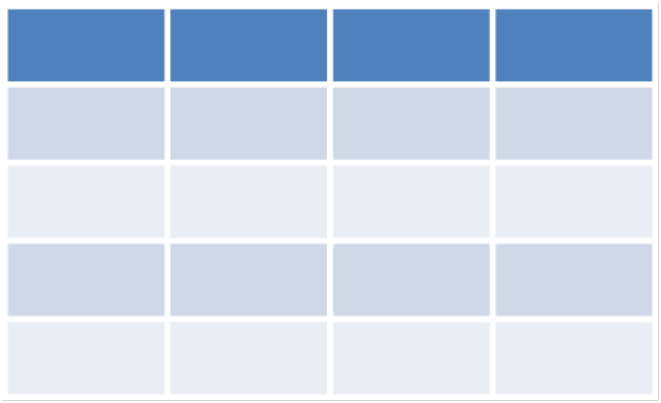
Para Elmasri (2018, p. 795):

> A maioria dos sistemas NoSQL são bancos de dados distribuídos ou sistemas de armazenamento distribuído, com foco no armazenamento de dados semiestruturados, alto desempenho, disponibilidade, replicação de dados e escalabilidade, ao contrário da ênfase em consistência imediata de dados, linguagens de consulta poderosas e armazenamento de dados estruturados.

Antes de apresentarmos alguns modelos de bancos de dados NoSQL, vale ressaltar uma grande diferença entre esses e os modelos relacionais, deixando mais clara a compreensão e a diferença entre ambos.

Quando falamos em bancos de dados relacionais, logo nos vêm à mente alguns termos fundamentais, como esquema, relacionamentos, coleção de dados estruturados e inter-relacionados, etc.

Figura 3 – Bancos de dados relacionais



Já nos bancos de dados NoSQL, Sadalage e Fowler (2019, p. 36) destacam que não é necessário um esquema, pois os dados podem ser armazenados sem a necessidade de definição de uma estrutura, além de estes possuírem uma série de possibilidades de armazenamento semiestruturados.

Foi diante da necessidade de trabalhar com grandes volumes de dados que Elmasri (2018, p. 796) menciona a iniciativa de algumas empresas que desenvolveram seus próprios sistemas de gerenciamento de dados. A Google desenvolveu o Bigtable para usar em suas aplicações com necessidade de armazenamento de grandes quantidades de dados, como o Gmail, o Google Maps, entre outros. Além desse, desenvolveu outro sistema NoSQL para armazenamento de família de colunas, o Hbase, que estudaremos em outra aula. Já a Amazon desenvolveu o sistema NoSQL DynamoDB criando a categoria de armazenamento de dados usando o conceito de chave-valor. Ainda, o Facebook desenvolveu o sistema NoSQL chamado *Cassandra*, que usa conceitos de armazenamento de chave-valor e sistemas baseados em colunas. Além desses, outras empresas também desenvolveram soluções como os sistemas baseados em documentos, entre eles se destaca o MongoDB e CouchDB, e os sistemas baseados em grafos, incluindo soluções como o Neo4J e GraphBase.

Entretanto, antes de explorar mais detalhes sobre os bancos de dados NoSQL, precisamos de uma compreensão sobre computação distribuída no contexto de bancos de dados, que resultou em bancos de dados distribuídos (BDD). Um sistema de computação distribuída trabalha subdividindo a relação de um problema em relações menores – nodos ou nós; ao longo do texto será usado o termo *nodo*(s), mas algumas citações de Elmasri (2018) usarão o termo *nó*(s) –, que podem ser gerenciadas independentemente, mas quando interconectadas, trabalham de forma coordenada. Desse modo, os sistemas distribuídos possuem uma capacidade maior de processamento, apresentando bom desempenho, confiabilidade e suportando um número maior de usuários.

Inicialmente, como relatado por Elmasri (2018, p. 757), os bancos de dados distribuídos buscavam:

> Resolver as questões de distribuição de dados, replicação de dados, consulta distribuída e processamento de transação, gerenciamento de metadados de banco de dados distribuído e outros temas. Mais recentemente, surgiram muitas tecnologias novas, que combinam tecnologias distribuídas e de bancos de dados. Essas tecnologias e esses sistemas estão sendo desenvolvidos para lidar com o armazenamento, a análise e a mineração de grandes quantidades de dados que estão sendo produzidos e coletados, e geralmente são conhecidas como tecnologias big data.

Atualmente, surgiram novas tecnologias, como os sistemas de bancos de dados NoSQL, “voltadas para oferecer soluções distribuídas no gerenciamento de grandes quantidades de dados necessárias em aplicações como mídia social, saúde, segurança, para citar apenas algumas” (Elmasri, 2018, p. 758). Por meio desses estudos, vamos compreender que cada um deles possui diferentes modelos de estrutura para armazenamento, sendo baseados em: documentos, chave-valor, colunas e em grafos. Na sequência, é apresentada uma breve explicação sobre cada uma dessas categorias de bancos de dados NoSQL, conforme exposto por Elmasri (2018, p. 799):

> 1. Sistemas NoSQL baseados em documentos: esses sistemas armazenam dados na forma de documentos usando formatos conhecidos, como JSON (*JavaScript Object Notation*). Os documentos são acessíveis por meio de seu ID de documento, mas também podem ser acessados rapidamente usando outros índices.
> 2. Armazenamentos de chave-valor do NoSQL: esses sistemas possuem um modelo de dados simples, com base no acesso rápido pela chave ao valor associado a esta chave; o valor pode ser um registro, um objeto, um documento ou até mesmo ter uma estrutura de dados mais complexa.
> 3. Sistemas NoSQL baseados em colunas ou em largura de colunas: esses sistemas particionam uma tabela por coluna em famílias de colunas [...] em que cada família de colunas é armazenada em seus

próprios arquivos. Eles também permitem o versionamento dos valores de dados.
4. Sistemas NoSQL baseados em grafos: os dados são representados como gratos e os nós relacionados podem ser encontrados percorrendo suas arestas por meio de expressões de caminho.

Para um melhor entendimento, nos próximos parágrafos serão exploradas cada uma dessas categorias de bancos de dados NoSQL e serão apresentados exemplos, iniciando pelo modelo de armazenamento chave-valor (Key/value).

A **categoria de armazenamento chave-valor** utiliza um modelo de depósito de dados no qual são armazenados os pares (chave e valor) que correspondem a objetos indexados por chaves.

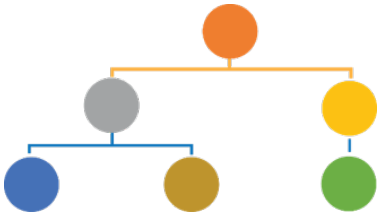
Figura 4 – Armazenamento chave-valor

No modelo chave-valor, é possível inserir, consultar e apagar um valor por meio de uma determinada chave (Strauch, 2011). O valor é o termo usado para se referir ao objeto armazenado, o qual corresponde a uma coleção de dados. “Já que depósitos de chave-valor sempre fazem o acesso pela chave primária, eles têm, geralmente um ótimo desempenho e podem ser escaláveis facilmente” (Sadalage; Fowler, 2019, p. 123).

Sistemas de bancos de dados NoSQL que usam esse modelo chave-valor são: DynamoDb, Couchbase, Riak, Azure Table Storage, Redis, entre outros.

A categoria de armazenamento orientado a documentos possibilita o armazenamento de dados semiestruturados. Dessa forma, o banco possibilita armazenar documentos sem que haja uma estrutura comum, um esquema, embora ainda façam parte de uma mesma coleção.

Figura 5 – Armazenamento orientado a documentos



Fazendo um comparativo com o sistema gerenciador de banco de dados relacional (SGBDR), o banco de dados NoSQL orientado a documentos utiliza uma “diferente representação de dados; não é a mesma que se utiliza em um SGBDR, em que todas as colunas devem ser definidas e, se não contiverem dados, são marcadas como vazias ou nulas” (Sadalage; Fowler, 2019, p. 135). Nessa categoria, os “documentos são estruturas de dados na forma de árvores hierárquicas e autodescritivas, constituídas de mapas, coleções e valores escalares” (Sadalage; Fowler, 2019, p. 133). Possibilita armazenar e recuperar documentos em formatos como: XML, JSON, BSON, entre outros.

Sistemas de bancos de dados NoSQL que usam esse modelo orientado a documentos são: MongoDb, CouchDB, RavenDb, entre outros.

A categoria de **armazenamentos de famílias de colunas** “permite que você armazene dados com chaves mapeadas para valores, e os valores são agrupados em múltiplas famílias de colunas, cada uma dessas famílias de colunas funcionando como um mapa de dados” (Sadalage; Fowler, 2019, p. 147).

Figura 6 –Armazenamento de famílias de colunas

Fazendo um comparativo com um banco de dados relacional, pode-se destacar que esse é um modelo eficiente no processo de escrita dos dados de um registro, já o banco de dados de famílias de colunas é mais eficiente em aplicações em que é preciso otimizar a leitura de dados.

Sistemas de bancos de dados NoSQL que usam esse modelo de famílias de colunas são: Cassanda, Hypertable, Amazon SimpleDB, Hbase, Bigtable, Hadoop, entre outros.

A **categoria de armazenamento de grafos** “permite que você armazene entidades e também relacionamentos entre essas entidades” (Sadalage; Fowler, 2019, p. 161). As entidades são os nodos do grafo e os relacionamentos as arestas, sendo que ambos podem ter propriedades.

Figura 7 – Armazenamento de grafos

“A organização do grafo permite que os dados sejam armazenados uma vez e depois interpretados de formas diferentes baseadas em relacionamentos” (Sadalage; Fowler, 2019, p. 161).

Exemplos de sistemas de bancos de dados NoSQL que usam esse modelo de grafos são: Neo4J, Infinite Graph, OrientDB, entre outros.

Na Tabela 1, é apresentada uma classificação comparativa de classes de bancos de dados NoSQL e banco de dados relacional versus propriedades não funcionais.

Tabela 1 – Comparativo de classes de modelos de bancos de dados versus propriedades não funcionais

Modelo	Performance	Escalabilidade	Flexibilidade	Complexidade	Funcionalidade
Chave-valor	Alta	Alta	Alta	Nenhuma	Variável
Colunas	Alta	Alta	Moderada	Baixa	Mínima
Documentos	Alta	Variável	Alta	Baixa	Variável
Grafos	Variável	Variável	Alta	Alta	Teoria de Grafos
Relacional	Variável	Variável	Baixa	Moderada	Álgebra Relacional

Fonte: Elaborado com base em Strauch, 2011, p. 26.

Nos parágrafos anteriores, conhecemos as diferentes categorias de bancos de dados NoSQL. De agora em diante, veremos algumas características dos sistemas NoSQL x Relacional.

1.1 E agora? Banco de dados relacional ou NoSQL?

Neste ponto, Sadalage e Fowler (2019, p. 37) enfatizam que é preciso primeiro compreender quais dados queremos armazenar e como queremos manipular. Como resultado dessa análise, observa-se que há uma grande probabilidade de optar por uma combinação de diferentes tecnologias de armazenamento de dados. “Esse ponto de vista é, muitas vezes, chamado de

persistência poliglota” (Sadalage; Fowler, 2019, p. 37). Isso se deve ao fato de existirem diferentes características de cada banco de dados em diferentes circunstâncias.

Enquanto os bancos de dados relacionais se destacam por possibilitar consistência dos dados, os bancos NoSQL são recomendados quando se espera por desempenho de processamento, em outras palavras, rápida recuperação, mostrando-se uma solução para o problema de escalabilidade existente nos bancos de dados relacionais. Isso ocorre porque no NoSQL o armazenamento de um conjunto de dados está em um mesmo registro e não depende do processamento de dados armazenados em outras tabelas, como se dá no modelo relacional.

Por exemplo, imagine um fórum em que cada postagem pode receber diversas interações. Ao usarmos um banco de dados relacional para a modelagem deste domínio, teremos a definição das tabelas: postagem e comentários, sendo necessário percorrer todos os comentários para identificar os que se referem a uma determinada postagem. Esse é um exemplo clássico a ser mencionado, no qual os comentários não fazem sentido sem a postagem e é preciso pesquisar milhares de comentários para identificar os que pertencem a uma postagem, podendo, assim, ocasionar um verdadeiro colapso para o servidor (óbvio que aqui colocamos uma pitada de dramatização). Com esse exemplo, podemos compreender que, neste caso, faz muito mais sentido ter os dados de postagens juntamente com os seus respectivos comentários. Assim, quando forem requisitados, a busca será completa e facilitada. “De modo geral, é muito útil poder colocar uma estrutura rica de informações em uma única solicitação ou resposta para reduzir o número de idas e vindas envolvidas nas comunicações remotas” (Sadalage; Fowler, 2019, p. 31).

Para ampliar o entendimento acerca da aplicabilidade e uso de um adequado modelo de banco de dados, no Quadro 1 são apresentados exemplos e tipos de aplicações que melhor condizem com um banco de dados relacional versus tipos de aplicações em que um banco de dados NoSQL melhor se adéqua.

Quadro 1 – Banco de Dados Relacional ou NoSQL – Exemplo de Aplicações

Banco de Dados Relacional	Banco de Dados NoSQL
• Aplicações centralizadas, ex: ERP, CRM. • Requerem alta disponibilidade, quando necessário. • Dados são gerados em velocidades moderadas. • Dados gerados a partir de poucas fontes. • Dados estruturados. • Transações complexas. • Moderado volume de dados.	• Aplicações descentralizadas, ex: Web, Mobile, Big Data, Iot. • A disponibilidade precisa ser contínua, sem interrupção. • Dados gerados em alta velocidade, ex: sensores. • Dados gerados a partir de múltiplas fontes, semi ou não estruturados. • Transações simples. • Alto volume de dados.

Fonte: Elaborado com base em Pereira, 2020.

Em suma, tanto o banco de dados relacional quanto o NoSQL são recomendados. Assim sendo, é comum ter aplicações com arquitetura híbrida, a qual faz uso de ambos os modelos, aproveitando o que se pode obter de melhor em cada qual, cabendo a escolha da solução ao que for mais adequado ao cenário.

Para concluir essa fundamentação teórica de bancos de dados NoSQL, segundo Sadalage e Fowler (2019, p. 18), “precisamos enfatizar que essa é uma área da computação que está em constante mudança. Aspectos importantes acerca destes tipos de armazenamento modificam-se todos os anos, e surgem novos recursos, novos bancos de dados”.

TEMA 2 – CARACTERÍSTICAS RELACIONADAS AOS BANCOS DE DADOS DISTRIBUÍDOS E SISTEMAS DISTRIBUÍDOS

Como relatado por Elmasri (2018, p. 797), essas características estão relacionadas à alta disponibilidade necessária para viabilizar o compartilhamento dos dados e também a escalabilidade, em virtude do contínuo aumento do volume dessas bases e alto desempenho. Para um melhor entendimento, nos próximos parágrafos serão explorados os detalhes a respeito dessas características.

A escalabilidade de bancos de dados distribuídos pode ser horizontal e vertical. Já nos bancos de dados NoSQL, em linhas gerais, se usa escalabilidade horizontal, “na qual o sistema distribuído é expandido, adicionando mais nodos para armazenamento e processamento de dados à medida que o volume de dados aumenta” (Elmasri, 2018, p. 797).

Os bancos de dados NoSQL têm a necessidade de permanecerem disponíveis, ainda que ocorram falhas em nodos ou até mesmo se algum hardware ou software ficar indisponível. Para isso, “os dados são replicados em dois ou mais nodos de maneira transparente, de modo que, se um nodo falhar, os dados ainda estarão disponíveis em outros nodos” (Elmasri, 2018, p. 797).

Em relação a modelos de replicação, Elmasri (2018, p. 797) afirma que os bancos de dados NoSQL usam dois modelos, chamados de *replicação mestre-escravo* e *ponto a ponto*. “A replicação mestre-escravo exige que uma cópia seja a principal; todas as operações de gravação devem ser aplicadas à cópia principal e, em seguida, propagadas para as cópias escravas” (Elmasri, 2018, p. 797). Já a replicação ponto a ponto consiste em balancear a carga de acesso simultâneo aos registros de um arquivo, não armazenando o arquivo inteiro em apenas um nodo, mas distribuindo para vários nodos, e com isso aumentando a disponibilidade dos dados (Elmasri, 2018, p. 798).

No que diz respeito ao acesso a dados de alto desempenho, Elmasri (2018, p. 798) relata a problemática de encontrar registros ou itens de dados em um emaranhado de registros e itens de dados. Podemos associar isso à “busca por uma agulha no palheiro”. Para atenuar isso, os bancos de dados NoSQL geralmente usam técnicas de *hashing*, ou particionamento.

2.1 Teorema CAP

As transações de um banco de dados relacional mantêm a integridade dos dados por possuírem propriedades denominadas ACID (atomicidade, consistência, isolamento e durabilidade) (Pereira, 2020). Já os bancos de Dados NoSQL, visto que realizam armazenamento de dados distribuído, possuem propriedades de ambiente distribuído, sendo elas: consistência, disponibilidade e tolerância a partições, denominadas CAP, uma abreviação dos termos em inglês (Browne, 2020). Na sequência, será apresentada uma explicação sobre cada uma destas propriedades:

- **Consistência:** obter um estado consistente após a execução de uma operação. Ao ocorrer uma operação entre uma das cópias compartilhadas do sistema distribuído, sendo cada cópia um nodo, cada modificação de um nodo deve ter seus dados replicados entre as demais cópias, ou seja,

a atualização do nodo deve simultaneamente atualizar suas cópias, possibilitando ter dados atualizados em todos os nodos.

- **Disponibilidade:** manter o sistema em operação, ainda que ocorram falhas entre os nodos ou até mesmo indisponibilidade de hardware e software.
- **Tolerância a partições:** manter o sistema em funcionamento mesmo que ocorram partições entre nodos devido a falhas ou a adição/remoção de nodos. O fato é que se falamos de bancos de dados com sistemas distribuídos, falamos de partição de cluster.

O teorema CAP foi introduzido por Brewer (2000) e, embora ele compreenda três propriedades que influenciam na arquitetura e funcionamento do armazenamento dos dados, somente se pode ter a garantia de apenas duas delas. Assim, é preciso optar por quais propriedades se deseja garantir, e isso pode influenciar diretamente na escolha do banco de dados NoSQL a ser utilizado em determinadas aplicações. Dessa forma, considerando a escolha de duas propriedades, se obtém a combinação de três alternativas. Essas combinações se referem às interseções entre consistência e disponibilidade (CA), consistência e tolerância a partições (CP) e disponibilidade e tolerância a partições (AP), destacadas em amarelo na representação do Teorema CAP, como mostra a Figura 8:

Figura 8 – Diagrama do Teorema CAP para distribuição

Fonte: Elaborado com base em Brown, 2020.

A interseção CA se refere à junção das propriedades: consistência e disponibilidade. As aplicações que optam por essa combinação de propriedades necessitam de forte consistência de leitura e escrita de dados e altíssima disponibilidade das aplicações, bastante clássico de aplicações com bancos de dados relacionais (Steppat, 2020; Araujo, 2020). “Nesse modelo, qualquer falha que ocorra em um dos nós, o sistema todo fica indisponível até que o nó que falhou volte ao normal” (Sirqueira, 2018).

A interseção CP se refere à junção das propriedades: consistência e tolerância a partições. Essa combinação de propriedades permite que as aplicações tenham consistência forte dos dados e tolerância a particionamento, permitindo resposta rápida às falhas. “Nesse modelo pode ocorrer que uma operação de escrita gere conflito entre os nós do particionamento de rede, sendo a disponibilidade comprometida até que ocorra um consenso entre os nós” (Sirqueira, 2018), ou seja, o SGBD tem autonomia para aceitar uma escrita ou não. Exemplos de bancos de dados com essas propriedades são: BigTable, Hbase ou MongoDB, Hadoop, entre outros (Steppat, 2020; Araujo, 2020).

A interseção AP se refere à junção das propriedades: disponibilidade e tolerância a partições. Essas propriedades são essenciais para aplicações que precisam estar disponíveis todos os dias e em todos os horários, ou seja, jamais podem ficar offline. Assim, o sistema sempre aceita operações de escrita e a consistência de dados é tratada posteriormente, por meio da sincronização. Tendo em vista esses dois aspectos, o período em que ocorre a persistência de um dado em um determinado nodo e o sincronismo desse nodo com os demais nodos do cluster, salienta-se que “existe uma janela de inconsistência, onde uma operação de leitura em um nodo que ainda não foi atualizado pode retornar dados desatualizados” (Sirqueira, 2018). Exemplos de bancos de dados com essas propriedades são: Amazon DynamoDB, MongoDB, Cassandra, Riak, Voldemort, entre outros (Steppat, 2020; Araujo, 2020).

“O conhecimento do teorema CAP é fundamental para problemas do mundo real, em que a solução do problema atacado envolve o uso de sistemas distribuídos” (Sirqueira, 2018).

TEMA 3 – MODELOS DE DADOS E LINGUAGENS DE CONSULTA

“Sistemas NoSQL enfatizam desempenho e flexibilidade em relação a poder de modelagem e consulta complexa” (Elmasri, 2018, p. 798). Para melhor

compreendermos, nos próximos parágrafos serão detalhadas essas características.

Quando criamos um banco de dados relacional, o primeiro passo consiste em criar um esquema. Entretanto, Sadalage e Fowler (2019, p. 61) expõem que é difícil saber antecipadamente o que armazenar. Assim, definir o tipo de armazenamento conforme se conhece mais sobre o projeto permite suportar dados não uniformes, ou seja, quando cada registro pode ser composto por diferentes campos.

Os bancos de dados NoSQL não utilizam esquema. Segundo Elmasri (2018, p. 798), a não exigência de um esquema é uma característica de vários sistemas de bancos de dados NoSQL, o que possibilita armazenar dados semiestruturados, como JSON, XML e autodescritivos. Mas isso não impede que seja especificado um esquema parcial, visando melhorar a eficiência do armazenamento.

De acordo com Sadalage e Fowler (2019, p. 61):

> Um armazenamento de chave-valor permite o armazenamento de quaisquer dados sob uma chave. Um banco de dados de documentos faz, efetivamente, o mesmo, uma vez que não tem restrições à estrutura dos documentos armazenados. Bancos de dados de famílias de colunas permitem que sejam armazenados quaisquer dados sob qualquer coluna escolhida. Bancos de dados de grafos permitem que sejam adicionadas, livremente, novas arestas e propriedades aos nodos e às arestas.

Assim, ter um banco de dados sem esquema significa ter um banco de dados que suporta que seus registros sejam compostos por novos ou diferentes dados sem que haja a necessidade de realizar alterações na estrutura do banco. Isso permite que os registros contenham apenas os dados necessários e facilita a manipulação de dados não uniformes. São esses fatores que possibilitam alta escalabilidade e disponibilidade nos bancos de dados NoSQL.

Apesar disso, “não utilizar esquemas é algo atrativo e, certamente, evita muitos problemas [...] novos problemas podem surgir” (Sadalage; Fowler, 2019, p. 62), visto que não ter um esquema também quer dizer que não há integridade de dados. Assim, para que a estrutura do banco de dados possa ser interpretada, é necessário ter um esquema implícito, ou seja, a interpretação do banco de dados passa para o software aplicativo.

Vogels (2020) utiliza como exemplo o carrinho de compras de uma loja virtual. O aplicativo que gerencia o carrinho de compras de um cliente pode unificar versões conflitantes e gerar um carrinho de compras unificado, ou então,

optar por exibir apenas o último carrinho registrado. Perceba que esse gerenciamento não foi atribuído ao banco de dados, como ocorre nos bancos relacionais com a integridade dos dados, mas se trata de uma especificação funcional do aplicativo.

Figura 9 – Carrinho de compras

Crédito: Mariyani Sugianto/Shutterstock.

Outro ponto destacado por Elmasri (2018, p. 799) é que, diferentemente das aplicações com bancos de dados relacionais que usam o SQL, as aplicações com sistemas de bancos de dados NoSQL não podem realizar consultas com tantas condições e restrições. Assim, utiliza-se o termo de consulta menos poderosa, pois nesse modelo, a leitura identifica itens de dados em um único arquivo.

Outra característica importante a ser destacada e que é indispensável em qualquer aplicação é o versionamento. “Alguns sistemas NoSQL fornecem armazenamento de múltiplas versões dos itens de dados” (Elmasri, 2018, p. 799).

TEMA 4 – MODELOS DE DADOS AGREGADOS

Usamos o termo *modelo de dados* para nos referirmos ao “modelo pelo qual o gerenciador do banco de dados organiza seus dados” (Sadalage; Fowler, 2019, p. 49). “Bancos de dados relacionais não possuem o conceito de agregado em seu modelo de dados, de modo que os chamamos de ‘não agregados’” (Sadalage; Fowler, 2019, p. 49).

Os bancos de dados NoSQL nas categorias chave-valor, documento e família de colunas possuem o conceito de orientação agregada em seu modelo de dados (Sadalage; Fowler, 2019, p. 42). “Bancos de dados de grafos não são agregados” (Sadalage; Fowler, 2019, p. 49). Eles diferem dos agregados em diversos aspectos, exceto por também diferirem dos bancos de dados

relacionais e por coincidentemente terem sua ascensão no mesmo período (Sadalage; Fowler, 2019, p. 61).

Para exemplificar a diferença entre o modelo relacional e o modelo agregado, no Quadro 2, é mostrado um modelo relacional elaborado para suportar transações de um website de comércio eletrônico.

Quadro 2 – Modelo relacional para um website de comércio eletrônico

Cliente	
id	Nome
1	Martin

Pedido		
id	ClienteId	EnderecoEntregaId
99	1	77

Produto	
Id	Nome
27	NoSQL Distilled

EnderecoCobranca		
Id	ClienteId	EnderecoId
55	1	77

ItemPedido			
Id	PedidoId	ProdutoId	Preco
100	99	27	32.45

Endereco	
Id	Cidade
77	Chicago

PagamentoPedido				
Id	PedidoId	NumeroCartao	EnderecoCobrancaId	txnId
33	99	1000-1000	55	abelif879rft

Fonte: Elaborado com base em Sadalage e Fowler, 2019, p. 43.

Ao apresentar esse exemplo, o autor explica o modelo supondo que neste comércio eletrônico “venderemos itens diretamente aos clientes pela web e teremos de armazenar informações sobre os usuários, o nosso catálogo de produtos, os pedidos, as remessas, os endereços de envio, os endereços de cobrança e os dados sobre o pagamento” (Sadalage; Fowler, 2019, p. 61).

Considerando esse cenário, mas adaptando essa modelagem para orientação de dados agregados, na figura a seguir temos a representação dessa modelagem no formato JSON, que é comumente usada para representação de dados NoSQL. Analisando o código apresentado, os comentários “// em clientes” e “// em pedidos” representam o trecho do código que divide essa agregação em duas partes: cliente e pedido. “O cliente contém uma lista de endereços de cobrança; o pedido contém uma lista de itens solicitados, um endereço de envio e pagamentos. O próprio pagamento contém um endereço de cobrança” (Sadalage; Fowler, 2019, p. 46).

Figura 10 – Agregação de dados para um website de comércio eletrônico

```
//em Clientes
{
  "id": 1,
  "nome": "Martin",
  "enderecoCobranca": [{"cidade": "Chicago"}]
}

//em Pedidos
{
  "id": 99,
  "clienteId": 1,
  "itemPedido": [
    {
      "produtoId": 27,
      "preco": 32.45,
      "produtoNome": "NoSQL Distilled",
    }],
    "enderecoEntrega": [{"cidade": "Chicago"}],
    "pagamentoPedido": [
      {
        "numeroCartao": "1000-1000",
        "txnd": "abelif879rft",
        "enderecoCobranca": {"cidade": "Chicago"}
      }]
}
```

Fonte: Sadalage e Fowler, 2019, p. 45.

TEMA 5 – MODELOS DE DISTRIBUIÇÃO E CONSISTÊNCIA

Nos temas anteriores desta aula, falamos do armazenamento distribuído de dados dos bancos de dados NoSQL, que propicia escalabilidade à medida que aumenta o volume de dados das aplicações. No Tema 4, falamos sobre o modelo de dados agregado, que é apropriado para um modelo de distribuição de dados. A seguir, vamos conhecer um pouco do comportamento da distribuição de dados.

5.1 Distribuição

A distribuição de dados pode ser realizada de duas formas: replicação e fragmentação. “A replicação obtém os mesmos dados e os copia em múltiplos nodos. A fragmentação coloca dados diferentes em nodos diferentes” (Sadalage; Fowler, 2019, p. 73). Ambas as técnicas podem ser usadas de forma isolada ou em conjunto.

Cabe ressaltar que o mais recomendado é não utilizar distribuição, mantendo o banco de dados em um único servidor. Embora se tenha uma forte

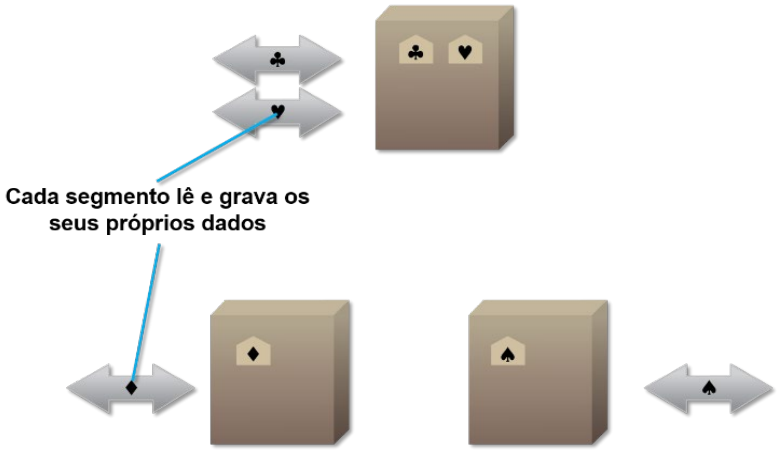
associação entre bancos de dados NoSQL com execução em cluster, a opção em um único servidor é mais facilmente gerenciada e menos complexa para os desenvolvedores, por não terem que se preocupar com a distribuição dos dados no desenvolvimento das aplicações (Sadalage; Fowler, 2019, p. 74).

No entanto, “a medida em que o volume de dados cresce, aumenta a necessidade de escalabilidade e melhoria de desempenho” (Monge, 2020). Podemos considerar também outros fatores como, por exemplo, a necessidade de aumentar o processamento e armazenamento das máquinas, assim como o acesso a diferentes partes de dados. Para essa conjuntura, é necessária a utilização de recursos de distribuição e consequentemente um aumento de máquinas que possam realizar o armazenamento e processamento de dados.

5.1.1 Fragmentação

Uma solução é utilizar a técnica de fragmentação, também conhecida por *sharding*, que possibilita escalabilidade horizontal, direcionando o acesso a diferentes partes de dados em diferentes servidores. Cada servidor desempenha o papel de gerenciar um subconjunto de dados, que representa um fragmento, o qual lê e grava seu próprio dado. Na Figura 11, é ilustrada a movimentação dos dados com a fragmentação, que “coloca dados diferentes em nodos separados, cada um desses executando suas próprias leituras e gravações” (Sadalage; Fowler, 2019, p. 75).

Figura 11 – Fragmentação



Fonte: Elaborado com base em Sadalage e Fowler, 2019, p. 75.

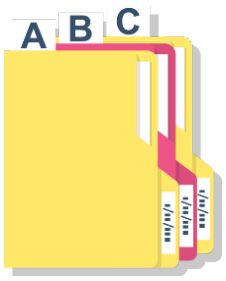
Crédito: EgudinKa/Shutterstock.

A complexidade de uso da fragmentação está no modelo de dados, o qual deve ser projetado com a agregação mais adequada possível, de modo que o acesso a um servidor (nodo) possibilite ao usuário encontrar a maior parte dos dados. Consequentemente, isso possibilita que esse conjunto de agregados sejam distribuídos uniformemente pelos nodos (Sadalage; Fowler, 2019, p. 75).

Exemplificando a fragmentação, imagine que a organização dos registros de clientes seja separada em nodos seguindo um modelo de arquivamento de fichários que usa, por exemplo, organização em ordem alfabética.

Nesse caso, cada nodo receberia apenas registros de clientes com a letra alfabética de armazenamento estabelecida, ou seja, um nodo para clientes em que o nome inicia com a letra A, outro com a letra B, assim por diante (Sadalage; Fowler, 2019, p. 76).

Figura 12 – Registros de clientes pelas iniciais do nome



Crédito: hvostik/Shutterstock.

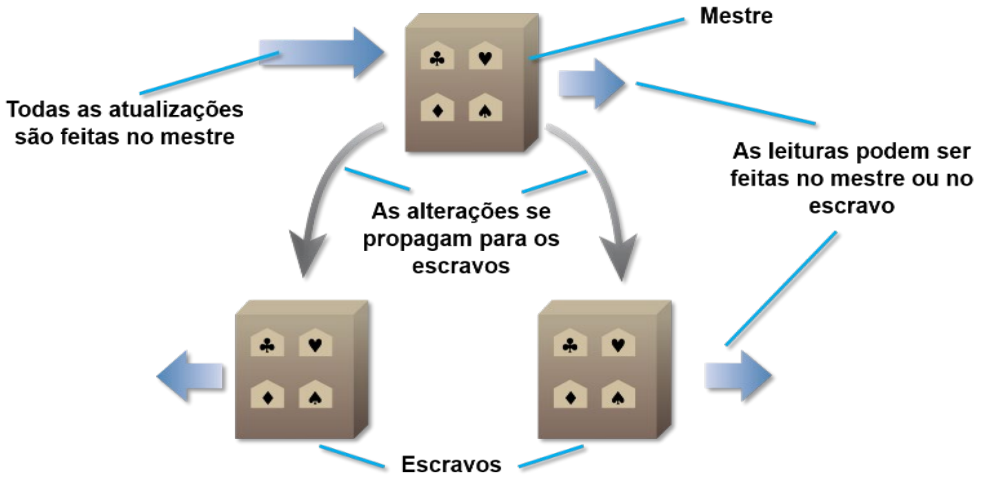
No cenário descrito, a programação do aplicativo é quem deve gerenciar esta organização, de modo que as consultas sejam realizadas no fragmento correspondente. Felizmente, muitos bancos de dados NoSQL já fazem o gerenciamento de alocação e a consulta de dados nos fragmentos automaticamente.

5.1.2 Replicação

"A replicação copia os dados para múltiplos servidores, de modo que cada parte dos dados pode ser encontrada em múltiplos lugares" (Sadalage; Fowler, 2019, p. 82). Sendo assim, esta diminui o tempo de recuperação de informações e propicia escalabilidade. Há dois tipos de replicação: mestre-escravo e ponto a ponto (p2p).

Na replicação mestre-escravo, os dados são copiados para múltiplos nodos, sendo um nodo designado como mestre e os demais escravos. O nodo mestre é “a fonte oficial dos dados e, geralmente, fica responsável por processar quaisquer atualizações nesses dados” (Sadalage; Fowler, 2019, p. 77). Os nodos escravos recebem os dados por um processo de sincronização com o mestre e são fontes apenas para leitura, o que proporciona escalabilidade horizontal quando há muitas solicitações de leitura (Sadalage; Fowler, 2019, p. 77). Na Figura 13, é ilustrada a movimentação dos dados nesse tipo de replicação.

Figura 13 – Replicação mestre e escravo

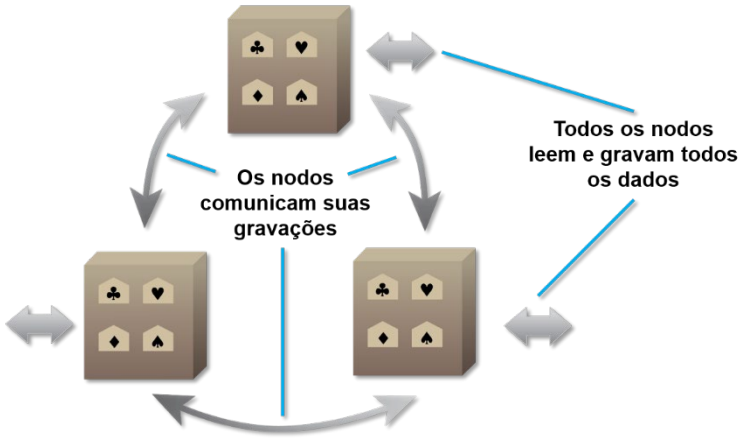


Fonte: Elaborado com base em Sadalage e Fowler, 2019, p. 78.
Crédito: EgudinKa/Shutterstock.

“A replicação mestre-escravo ajuda com a escalabilidade de leitura, mas não com a escalabilidade de gravação” (Sadalage; Fowler, 2019, p. 79). Já na replicação ponto a ponto, não há diferenciação entre os nodos, e todos permitem leitura e gravação de dados, sendo possível “contornar falhas nos nodos sem perder o acesso aos dados. Além disso, você pode “adicionar facilmente nodos para melhorar o seu desempenho” (Sadalage; Fowler, 2019, p. 80).

Na Figura 14, é ilustrada a movimentação dos dados no tipo de replicação ponto a ponto.

Figura 14 – Replicação ponto a ponto



Fonte: Elaborado com base em Sadalage e Fowler, 2019, p. 80.
Crédito: EgudinKa/Shutterstock.

5.2 Consistência

A seção 2.1 abordou os conceitos do Teorema CAP e vimos que os bancos de dados NoSQL possuem propriedades de ambiente distribuído, sendo a tolerância a partições mandatória para atender aos conceitos de distribuição que vimos na seção 5.1. Outro paradigma que difere os bancos de dados NoSQL dos bancos de dados relacionais é a forma como se trata a consistência dos dados, denominada *consistência eventual*, pois é preciso criar recursos programáveis para estabelecer a consistência dos dados.

Com isso, podemos ter conflitos de gravação e de leitura-gravação. O primeiro ocorre, por exemplo, quando temos dois usuários tentando gravar os mesmos dados simultaneamente. Já o segundo ocorre quando um usuário lê um dado inconsistente durante uma gravação que está sendo realizada por outro usuário.

Perante esses conflitos, é possível atacar duas abordagens de controle de concorrência: pessimista e otimista. “Abordagens pessimistas bloqueiam os registros de dados para evitar conflitos. Abordagens otimistas detectam conflitos e os resolvem” (Sadalage; Fowler, 2019, p. 100).

Embora a consistência seja algo importante para a integridade dos dados, às vezes precisamos correr o risco de perdê-la a fim de não sacrificar outras características. Esse balanceamento é contornado estabelecendo recursos programáveis que podem ser definidos no desenvolvimento da aplicação.

FINALIZANDO

Por meio desse estudo inicial, adquirimos uma visão geral sobre os elementos que compõem os bancos de dados NoSQL e suas principais características.

Compreendemos também que os bancos de dados NoSQL não vieram para substituir os bancos de dados relacionais, e que tanto o seu comportamento quanto sua finalidade de uso diferem bastante daqueles, de modo que temos aplicações que combinam os dois tipos, aproveitando o que há de melhor em cada um.

Aprendemos alguns conceitos essenciais para iniciarmos nossos projetos usando bancos de dados NoSQL, como o teorema CAP, modelos de dados agregados, distribuição e consistência.

Nas próximas aulas, conheceremos alguns sistemas gerenciadores que aplicam essas propriedades e diferentes modelos, brevemente citados ao longo desta aula.

REFERÊNCIAS

ARAUJO, J. R. **Medium**. Disponível em: <https://medium.com/@jrobertoaraujo/teorema-cap-3094645d7249>. Acesso em: 28 jan. 2021.

BREWER, E. A. **Towards Robust Distributed Systems**. Keynote at the ACM Symposium on Principles of Distributed Computing (PODC), 2000. Disponível em: <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>. Acesso em: 28 jan. 2021

BROWNE, J. **Brewer's CAP Theorem**. Disponível em: <http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>. Acesso em: 28 jan. 2021

BROWN; C. **NoSql Tips and Tricks**. Disponível em: <http://blog.nosqltips.com/search?q=CAP>. Acesso em: 28 jan. 2021

ELMASRI, N. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson, 2018.

MONGE; W. **Técnicas de Modelagem de Dados (NOSQL)**. Disponível em: <https://medium.com/t%C3%A9cnicas-de-modelagem-de-dados-nosql/banco-de-dados-nosql-58726ce6886c>. Acesso em: 28 jan. 2021

PEREIRA, N. **Quando Utilizar RDBMS ou NOSQL?**. Disponível em: <http://datascienceacademy.com.br/blog/quando-utilizar-rdbms-ou-nosql/>. Acesso em: 28 jan. 2021

SADALAGE, P. J.; FOWLER, M. **NoSQL Essencial:** Um guia conciso para o Mudo emergente da persistência poliglota. 1. ed. São Paulo: Novatec, 2019.

SIRQUEIRA, T.; DALPRA, H. **NoSQL e a Importância da Engenharia de Software e da Engenharia de Dados para o Big Data:** Jornadas de Atualização em Informática. Sociedade Brasileira de Computação, 2018.

STEPPAT, N. **NoSQL – Do teorema CAP para P?(A|C):(C|L)**. Disponível em: <https://blog.caelum.com.br/nosql-do-teorema-cap-para-paccl/>. Acesso em: 28 jan. 2021

STRAUCH, C. **NoSQL databases:** Lecture Notes J. Stuttgart Media University, 2011.

VICTORINO, M. **NOSQL**. Disponível em: <http://www.ms.senai.br/transparencia_senai/uploads/integridade/17-04-2019-01-No-SQL.pdf>. Acesso em: 28 jan. 2021

VOGELS, W. **All Things Distributed**. Disponível em: <http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html>. Acesso em: 28 jan. 2021