BANCO DE DADOS NOSQL

AULA 6

Prof. Alex Mateus Porn

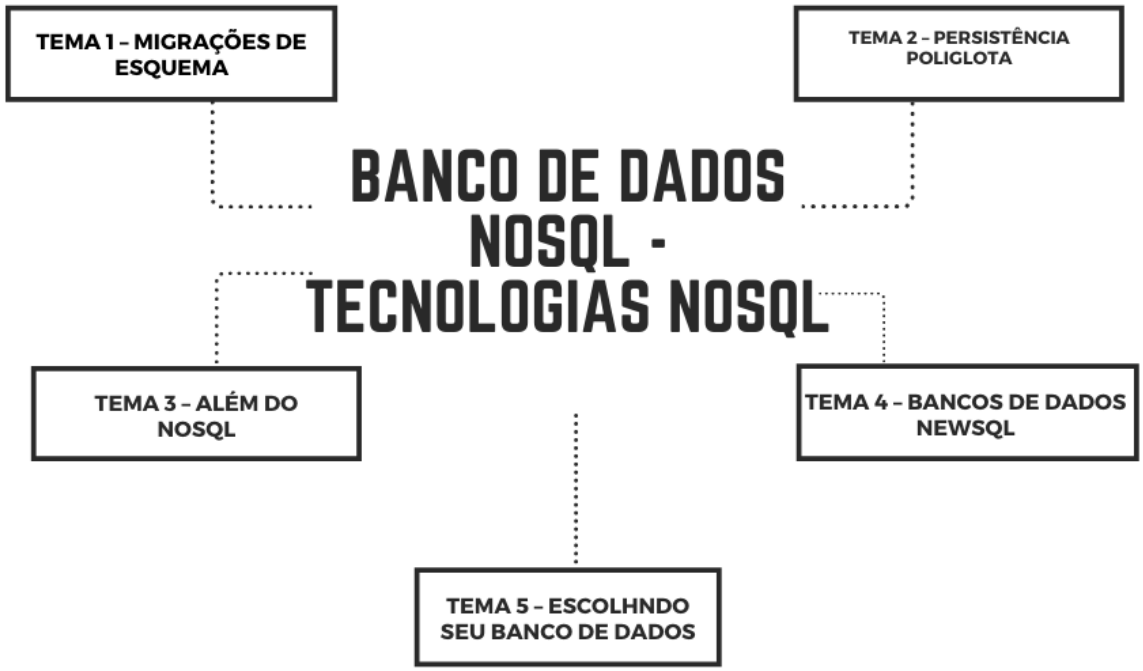
CONVERSA INICIAL

Olá! Nesta aula, vamos abordar de um modo abrangente as tecnologias NoSQL e o que está além delas. Nosso objetivo principal é compreender a abordagem de persistência NoSQL que estudamos anteriormente, identificando possibilidades do uso de mais de uma estrutura NoSQL em um mesmo projeto, ou em conjunto, com modelos de dados relacionais. Estudaremos algumas técnicas e metodologias de migrações de esquema, tanto entre modelos NoSQL, como de modelo relacional para NoSQL.

Iniciaremos abordando os conceitos de migração de esquemas, para então compreender os conceitos de persistência poliglota. Conhecendo essa possiblidade do uso de mais de uma estrutura em conjunto, analisaremos outras tecnologias de persistência de dados, além do NoSQL, com foco na nova tecnologia NewSQL.

Finalizaremos esta aula analisando alguns métodos para avaliação e seleção de uma estrutura ou de um sistema gerenciador de banco de dados, dentre todos os que estudamos, que se adeque melhor aos futuros projetos que vamos desenvolver.

Figura 1 – Roteiro da aula



TEMA 1 – MIGRAÇÕES DE ESQUEMA

Uma questão-chave a destacar, com base em tudo o que estudamos anteriormente sobre NoSQL, é a natureza livre dos esquemas de bancos de dados. Conforme destacam Sadalage e Fowler (2013, p. 177):

> Essa natureza livre de esquemas é um recurso popular que permite aos desenvolvedores concentrarem-se no projeto do domínio sem a preocupação com alterações no esquema. Isso torna-se particularmente verdadeiro com a ascensão dos métodos ágeis, em que é importante atender as mudanças nos requisitos.

Diante dessa abordagem, percebemos que, ao contrário dos bancos de dados relacionais, que precisam ser alterados antes do aplicativo, a abordagem sem esquema dos bancos de dados NoSQL visa garantir flexibilidade nas alterações, com o objetivo de atender às frequentes alterações no mercado e às inovações de produtos de software.

Mesmo diante dessa análise, de que em alguns casos o esquema não precisa ser planejado antecipadamente em bancos de dados NoSQL, e diante da flexibilidade da natureza livre de esquemas, ainda é preciso fazer o planejamento e o projeto de alguns aspectos (Sadalage; Fowler, 2013, p. 183):

- Tipos de relacionamentos para bancos de dados orientados a grafos;
- Nomes das famílias de colunas, linhas, ou a ordem das colunas para os bancos de dados orientados a colunas;
- Como as chaves estão atribuídas e qual é a estrutura dos dados dentro do objeto de valor para os bancos de dados orientados à chave-valor.

Considerando essa análise, Sadalage e Fowler (2013, p. 183) apontam que os bancos de dados NoSQL não são inteiramente desprovidos de esquema, mesmo que armazenem os dados sem considerar o esquema a que estão associados:

> esse esquema tem de ser definido pelo aplicativo, pois o fluxo de dados tem de ser analisado por ele ao fazer a leitura dos dados do banco de dados. Além disso, o aplicativo tem de criar os dados que seriam gravados no banco de dados. Se ele não puder analisar os dados do banco de dados, temos uma incompatibilidade de esquema mesmo que, ao invés de o banco de dados relacional gerar um erro, esse erro é, agora, encontrado pelo aplicativo.

Diante dessa abordagem, podemos citar o exemplo das migrações em bancos de dados NoSQL orientados a grafos (Sadalage; Fowler, 2013, p. 187). Nesse tipo de banco de dados, se alteramos o tipo de uma aresta no aplicativo,

não podemos mais percorrer o banco de dados, o que o torna inútil. Uma solução para esse caso seria percorrer todas as arestas do banco e alterá-las conforme necessário. Porém, dependendo do tamanho do banco de dados, essa operação seria extremamente custosa, exigindo a criação de códigos para a migração de todas as arestas necessárias.

Como vimos anteriormente, a associação entre dois nós pode ocorrer por mais de uma aresta, inclusive por arestas iguais. Nesse caso, uma solução alternativa ao problema seria criar novas arestas entre os nós, mantendo as arestas existentes. Posteriormente, quando tivermos certeza quanto à alteração, as arestas antigas podem ser eliminadas.

Outra possibilidade no contexto da migração de esquemas é a migração dos já conhecidos e conceituados bancos de dados relacionais para um modelo NoSQL. Souza, Paula e Barros (2020) apresentam metodologias de migração de bancos de dados relacionais para bancos orientados a documentos. Os autores apresentam três metodologias que também se aplicam a outras estruturas NoSQL:

1. **Metodologia baseada em grafos:** metodologia proposta por Zhao et al. (2014), desenvolvida para a realização da conversão do modelo de dados relacional para um modelo de dados NoSQL qualquer. O início do processo de migração se dá com a construção de um grafo G = <V, E>, em que V é o conjunto de vértices, representando as tabelas do banco de dados relacional, e E é o conjunto de arestas direcionadas no grafo, representando todas as dependências entre as tabelas do banco de dados relacional. *Dependência* significa que uma tabela faz referência a outra por chave estrangeira.
2. **Metodologia baseada em consultas:** metodologia proposta por Li, Ma e Chen (2014), que parte da migração, sendo necessário considerar quais consultas serão realizadas no banco de dados, a fim de aumentar o desempenho da busca, uma vez que operações de junção não são aconselháveis em bancos de dados NoSQL.
3. **Metodologia baseada na definição dos níveis físico e lógicos dos dados:** metodologia proposta por Karnitis e Arnicans (2015), a migração dos dados passa por três passos: nível físico dos dados, primeiro nível lógico dos dados, e segundo nível lógico dos dados. Na etapa do nível físico, as tabelas são identificadas e caracterizadas. Na etapa de primeiro

nível lógico dos dados, os metadados do nível físico são acrescidos de informações lógicas, em linguagem natural. No segundo nível lógico dos dados, as tabelas são utilizadas para gerar o esquema inicial dos documentos e, consequentemente, a semântica do negócio.

Essas três metodologias são apenas algumas das abordagens propostas na literatura para mapeamento do modelo relacional para uma estrutura de banco de dados NoSQL. Sadalage e Fowler (2013, p. 189) frisam alguns pontos importantes a serem considerados para a migração de um esquema, seja de um modelo NoSQL para outro modelo NoSQL ou do modelo relacional para alguma estrutura NoSQL:

- Bancos de dados com esquemas fortes, como os bancos de dados relacionais, podem ser migrados gravando cada alteração do esquema, mais a sua migração de dados, em uma sequência controlada por versões;
- Bancos de dados sem esquema ainda precisam de uma migração cuidadosa, por conta do esquema implícito nos códigos que acessam os dados.
- Bancos de dados sem esquema podem utilizar as mesmas técnicas de migração dos bancos de dados com esquemas fortes.
- Bancos de dados sem esquema também podem ler dados de forma tolerante às alterações no esquema implícito de dados; além disso, podem utilizar migração incremental para atualizá-los.

TEMA 2 – PERSISTÊNCIA POLIGLOTA

Diferentes bancos de dados são projetados para resolver diferentes problemas. Conforme destacam Sadalage e Fowler (2013, p. 191),

> utilizar um único mecanismo de banco de dados para todas as necessidades resulta em soluções de baixo desempenho; armazenar dados transacionais, guardar em cache as informações de sessão, percorrer grafos de clientes e os produtos que seus amigos compraram são problemas essencialmente diferentes. Mecanismos de bancos de dados são projetados para executar muito bem certas operações em determinadas estruturas de dados e em determinadas quantidades de dados, tais como operar em conjuntos ou armazenamentos de dados e recuperar chaves e seus valores de modo muito rápido ou, ainda, armazenar documentos ricos ou grafos com informações complexas.

Ainda segundo Sadalage e Fowler (2013, p. 191-192), muitas instituições tendem a utilizar o mesmo mecanismo de banco de dados para armazenar transações de negócio, dados de gerenciamento de sessão e outras necessidades de armazenamento, como relatórios, BI, Data Warehouse ou informações de registros. Os autores ainda destacam que “os dados da sessão, do carrinho de compras ou do pedido não precisam das mesmas propriedades de disponibilidade, consistência ou necessidades de cópias de segurança”.

Atualmente, cada vez mais os projetos de software que necessitam de persistência de dados têm exigido diferentes tecnologias de armazenamento, ou seja, diferentes linguagens e mecanismos para lidar com os dados das aplicações. Conforme destacam Sadalage e Fowler (2013, p. 192), em 2006, Neal Ford propôs o termo *programação poliglota* para expressar a ideia de que os aplicativos devem ser escritos em uma mistura de linguagens, de modo a aproveitar o fato de que diferentes linguagens são apropriadas para lidar com diferentes problemas. Portanto, no cenário atual, de necessidade de diferentes tecnologias para armazenamento de dados em um mesmo sistema, o termo *persistência poliglota* passou a definir essa abordagem híbrida para a persistência de dados.

Existem diversos cenários em que devemos usar os tradicionais bancos de dados relacionais, ou seja, os bancos com SQL, além de cenários em que precisamos usar bancos não relacionais, ou seja, NoSQL. Como vimos anteriormente, esses bancos NoSQL classificam-se em várias categorias. Destacamos as quatro principais (chave-valor, documentos, família de colunas e grafos), estudadas detalhadamente em aulas diferentes. Aqui, a questão mais pertinente é identificar a melhor estrutura, seja ela SQL ou NoSQL, que se adequa ao projeto em desenvolvimento.

Nesse contexto, podemos estar diante de situações nas quais, em um único projeto, podemos implementar bancos de dados relacionais, orientados a chave-valor, orientados a documentos, orientados a colunas e orientados a grafos. Diante dessa situação, vale destacar o exemplo apresentado por Sadalage e Fowler (2013, p. 193):

> em um sistema de comércio eletrônico, o armazenamento de dados orientado a chave-valor pode ser utilizado para armazenar os dados do carrinho de compras antes de o pedido ser confirmado pelo cliente e, também para armazenar os dados da sessão, de modo que o banco de dados relacional não seja utilizado para esses dados transientes. Os bancos de dados orientados a chave-valor são aplicáveis para armazenamento e acesso ao carrinho de compras do usuário, de modo

> que uma vez confirmado e pago pelo cliente, os dados são gravados no banco de dados relacional. Por outro lado, havendo a necessidade de recomendar produtos para clientes quando eles estiverem colocando-os em seus carrinhos de compras, o uso de um banco de dados orientado a grafos seria relevante.

Como podemos perceber, em cada situação um modelo de dados específico pode ser aplicado, não havendo necessariamente a melhor ferramenta adequada para cada situação. Cabe aqui considerar uma analogia com uma caixa de ferramentas, com alicate, martelo, chave de fenda, serra etc. Não podemos, por exemplo, dizer qual é a melhor ferramenta na caixa, tudo dependerá do serviço a ser realizado, se apertar um parafuso, pregar um prego ou serrar uma madeira. Portanto, para cada atividade implementada, cabe analisar a melhor ferramenta.

Voltando ao exemplo anterior, considerando um sistema de comércio eletrônico que precisa recomendar ao cliente produtos similares àqueles que ele esteja comprando, poderíamos perfeitamente fazer essa implementação com um banco de dados relacional. Porém, para alterar a travessia, será necessário refatorar o banco de dados, migrar os dados e começar a persistir os novos dados. Se utilizarmos um banco de dados orientado a grafos, seria necessário simplesmente programar as novas relações e continuar utilizando o mesmo banco de dados com poucas alterações.

Sadalage e Fowler (2013, p. 193) ainda destacam que, à medida que múltiplos bancos de dados são implementados em um aplicativo, outros aplicativos na empresa poderão se beneficiar do uso desses bancos ou dos dados que armazenam. Com base no exemplo do sistema de comércio eletrônico, os autores propõem que um banco de dados de grafo “pode fornecer dados para outros aplicativos que precisam entender, por exemplo, quais produtos estão sendo comprados por um determinado segmento da base de clientes”.

Sadalage e Fowler (2013, p. 199) destacam dois pontos importantes a serem considerados sobre a persistência poliglota:

- Está relacionada com o uso de diferentes tecnologias de armazenamento de dados, de forma que possa lidar com necessidades variáveis de armazenamento de dados;
- Pode ser aplicada em uma empresa ou dentro de um único aplicativo.

TEMA 3 – ALÉM DO NOSQL

Conforme abordam Sadalage e Fowler (2013, p. 208), o NoSQL é apenas um conjunto de tecnologias de armazenamento de dados. Como a persistência poliglota aumenta o leque de facilidades, devemos analisar outras tecnologias de armazenamento de dados, com ou sem o rótulo NoSQL.

- **Bancos de dados XML:** Os bancos de dados XML podem ser vistos como bancos de dados orientados a documentos, no qual os documentos são armazenados em um modelo de dados compatível com XML, com diversas tecnologias XML sendo utilizadas para manipular o documento. Sadalage e Fowler (2013, p. 207) ainda destacam que os bancos de dados relacionais misturaram as capacidades XML com as de bancos de dados relacionais, inserindo documentos XML como um tipo de coluna e possibilitando alguma forma de misturar as linguagens de consulta SQL e XML.
- **Bancos de dados de objetos:** Os bancos de dados de objetos começaram a surgir com a popularização da programação orientada a objetos. Parafraseando Sadalage e Fowler (2013, p.208), o foco era a complexidade do mapeamento de estruturas de dados na memória para tabelas relacionais. O objetivo desse tipo de banco de dados é evitar essa complexidade, de modo que o banco gerencie automaticamente o armazenamento das estruturas da memória para o disco.
- **Elasticsearch:** De acordo com Paniz (2016, p. 171), o Elasticsearch não é necessariamente um banco de dados, mas sim uma ferramenta para processamento de *queries* envolvendo textos. Ao contrário de uma consulta direta em um banco de dados, como por exemplo, pesquisar o nome de um cliente, em que necessitaríamos utilizar o *LIKE* para retornar um valor *booleano* informando se o valor existe ou não, o Elasticsearch utiliza lógica difusa, em que cada palavra recebe uma nota com base no termo buscado, para assim retornar palavras semelhantes.
- **openCypher:** Já estudamos a linguagem Cypher em contraste à linguagem SQL para manipulação e gerenciamento dos bancos de dados orientados a grafos, na ocasião em que estudamos o SGBD Neo4j, precursor da linguagem Cypher. Parafraseando Paniz (2016, p. 172), essa linguagem despertou a atenção de outras empresas que desenvolvem

bancos orientados a grafos, que se uniram e criaram uma versão pública de código aberto desta linguagem, denominada openCypher.

- **Datomic:** Ao contrário dos demais bancos de dados, tanto relacionais quando NoSQL, o Datomic armazena todo o histórico de atualizações dos registros. De acordo com Paniz (2016, p. 172), quando alteramos um registro, estamos na verdade adicionando um novo fato a ele. Por isso, podemos facilmente carregar apenas um conjunto de dados atual de um registro, ou podemos carregar todas as transações que modificaram o registro e verificar cada valor antes e depois de cada transação.
- **Spark:** O Spark, assim como o Elasticsearch, não é um banco de dados, mas sim um motor para processamento de fluxo. Conforme Paniz (2016, p. 173), com o Spark é possível montar e gerenciar um cluster de máquinas, para executar processamentos de grandes volumes de dados. Ele possibilita a conexão de vários tipos de origens de dados, incluindo bancos de dados relacionais e NoSQL.
- **PostgreSQL Document Store:** O PostgreSQL é um sistema gerenciador de banco de dados relacional muito conhecido. Conforme destaca Paniz (2016, p. 173), em suas últimas versões, passou a oferecer suporte para armazenar documentos JSON, tanto no formato JSON puro quanto no formato binário (jsonb), semelhante ao formato BSON, usado pelo MongoDB. Desse modo, o PostgreSQL Document Store permite a realização de consultas por atributos, elementos aninhados e em arrays, à semelhança de um banco orientado a documentos. Nesse cenário, Paniz (2016, p. 174) ainda destaca:

> Diferente do MongoDB, o PostgreSQL continua garantindo consistência e durabilidade, podendo ser classificado como CP, se baseado no teorema CAP. Além do MongoDB ser duramente criticado pela sua arquitetura e os vários bugs reportados sobre perda de dados e vazamento de memória, na maioria dos testes de performance, o PostgreSQL se mostrou mais rápido e resiliente que ele.

- **Redis e Memcached:** Os bancos de dados Redis e Memcached são bancos NoSQL orientados a chave-valor, assim como o DynamoDB que já estudamos. Esses três bancos de dados podem ser considerados os bancos orientados a chave-valor mais utilizados. Conforme Paniz (2016, p. 174), o Redis usa uma forma de replicação de dados baseada em *master*/*slave*, e também suporta tipos de dados especiais, como conjuntos (*Set*) e listas (*List*), de forma similar ao que estudamos anteriormente,

quando analisamos o DynamoDB. Já o Memcached segue uma filosofia mais simplista. De acordo com Paniz (2016, p. 174), parte da lógica deve ficar no cliente, pois o servidor não suporta nenhum tipo de replicação ou dados especiais. Como o próprio nome sugere, sua melhor aplicação é para cache de dados.

- **NewSQL:** Considerando a intensa utilização dos bancos de dados NoSQL e algumas de suas limitações, como a falta de transações, a falta de consultas SQL e a complexidade pela fraca modelagem, os bancos de dados conhecidos como NewSQL surgiram como uma nova proposta. Conforme destaca Pereira (2015), os bancos de dados NewSQL buscam promover a mesma melhoria de desempenho e escalabilidade dos sistemas NoSQL, porém mantendo os benefícios dos bancos de dados relacionais, da linguagem SQL e das propriedades ACID. Stonebraker (2010) apresenta cinco características de um SGBD NewSQL:

    - Linguagem SQL como meio de interação entre o SGBD e a aplicação;
    - Suporte para transações ACID;
    - Controle de concorrência não bloqueante, para que as leituras e escritas não causem conflitos entre si;
    - Arquitetura que forneça um maior desempenho por nó de processamento;
    - Arquitetura escalável, com memória distribuída e com capacidade de funcionar em um aglomerado com um grande número de nós.

Conforme destaca Pereira (2015):

> Diferente dos SGBD tradicionais, que eram considerados soluções para qualquer tipo de aplicação, os NewSQL utilizam uma estratégia diferente, onde cada novo sistema desenvolvido visa atender a uma necessidade específica do mercado e busca alcançá-lo de forma separada, terminando com o antigo conceito de ter um único sistema que sirva para qualquer tipo de aplicação, fazendo com que os bancos de dados sejam especialistas para um propósito, não gerando mais um número absurdo de funções e comportamentos desnecessários para uma determinada aplicação.

Vamos agora destacar os SGBD NewSQL mais populares (Pereira, 2015):

- **MemSQL:** é operado em memória, sendo um sistema de banco de dados de alta escala, por sua combinação de desempenho e pela compatibilidade com o SQL transacional e ACID na memória.

- **VoltDB:** esse banco oferece a velocidade e a alta escalabilidade dos bancos de dados NoSQL, mas com garantias ACID e latência em milissegundos.
- **SQLFire:** Servidor de banco de dados NewSQL da VMware, desenvolvido para escalar em plataformas nas nuvens e tomar as vantagens de infraestrutura virtualizadas.
- **MariaDB:** foi desenvolvido pelo criador do MySQL, sendo totalmente compatível com o MySQL. Também pode interagir com os bancos de dados NoSQL.

TEMA 4 – BANCOS DE DADOS NEWSQL

Com base em tudo o que estudamos até este momento, sobre bancos de dados NoSQL, e principalmente sobre as vantagens e desvantagens que visualizamos através de várias comparações realizadas entre bancos de dados relacionais e NoSQL, podemos observar duas importantes situações:

- Bancos de dados NoSQL oferecem alto desempenho em escalabilidade, mas não oferecem suporte as propriedades ACID;
- Bancos de dados relacionais dão suporte as propriedades ACID, porém não oferecem desempenho escalável.

Diante dessa situação, os bancos de dados NewSQL surgem a partir da necessidade de se ter consistência dos dados, e de poder escalonar o sistema mais facilmente. Esses bancos de dados referem-se a uma classe de sistemas de gerenciamento de bancos de dados relacionais, que procura oferecer o mesmo desempenho escalável do modelo NoSQL, para cargas de trabalho de leitura e gravação no processamento de transações on-line, mantendo as garantias ACID do modelo relacional.

Como podemos observar, o modelo NewSQL é baseado no modelo de dados relacional. Nesse contexto, Grolinger et al. (2013) definem os bancos de dados NewSQL como “sendo baseados no modelo relacional, oferecendo uma visão puramente relacional dos dados. Embora os dados possam ser armazenados em forma de tabelas e relações, é interessante observar que as soluções NewSQL podem usar diferentes representações de dados”.

Segundo Ryan (2018), os bancos de dados NewSQL fornecem um sistema com a finalidade de processar milhões de transações por segundo, em

uma plataforma de hardware possível de se escalonar horizontalmente, similar aos bancos de dados NoSQL. Aao contrário dos bancos de dados relacionais, possibilita apenas escalonamento vertical.

Nesse sentido, Stonebraker et al. (2007) destacam que uma das principais características dos armazenamentos de dados NoSQL e NewSQL é sua capacidade de escalar horizontalmente e efetivamente, adicionando mais servidores. Mesmo que tenha havido tentativas de escalar bancos de dados relacionais horizontalmente, o contrário, esses bancos são projetados para escalar verticalmente, por meio da adição de mais potência a um único servidor existente.

Ryan (2018) destaca que os bancos de dados NewSQL possibilitam a sua execução em memória, com algumas vantagens, entre elas:

- Banco de dados totalmente relacional
- Conformidade com as propriedades ACID
- Latência de milissegundos
- Tolerância a falhas
- Execução local ou na nuvem

Conforme destacam Grolinger et al. (2013), mesmo que os usuários interajam com os armazenamentos de dados NewSQL em termos de tabelas e relações, é interessante observar que as soluções NewSQL podem usar diferentes representações de dados internamente. Por exemplo, o NuoDB pode armazenar seus dados em qualquer armazenamento orientado a chave-valor.

Em se tratando da aplicação dos bancos de dados NewSQL, Grolinger et al. (2007) destacam que, de um modo geral, esses bancos são apropriados em cenários nos quais o tradicional banco de dados relacional é usado, mas quando há requisitos adicionais de escalabilidade e desempenho. Grolinger et al. (2007) ainda destacam que:

> o armazenamento de dados NewSQL é apropriado para aplicativos que requerem o uso de transações que manipulam mais de um objeto, ou têm requisitos de consistência forte. Os exemplos clássicos são os aplicativos no mercado financeiro, onde operações como transferências de dinheiro precisam atualizar duas contas automaticamente e todos os aplicativos precisam ter a mesma visão do banco de dados.

A maioria dos bancos de dados NoSQL não oferece suporte a transações de vários objetos. Muitos deles são eventualmente soluções consistentes, o que os torna inadequados para esses casos de uso.

TEMA 5 – ESCOLHENDO SEU BANCO DE DADOS

Durante nosso estudo de bancos de dados NoSQL, abordamos várias questões gerais sobre esse modelo de persistência de dados; estudamos em detalhes as quatro principais estruturas NoSQL (orientada a documentos, orientada a chave-valor, orientada a família de colunas e orientada a grafos); e estudamos quatro SGBDs para cada uma dessas estruturas.

Naturalmente, é impossível apresentar uma resposta ou um simples conjunto de regras a seguir para determinar a melhor estrutura de persistência de dados em cada caso, incluindo nesse ponto o modelo de dados relacional, ou qual o melhor SGBD a ser utilizado para um problema específico a ser solucionado.

Conforme apresentam Sadalage e Fowler (2013, p. 209), duas importantes situações podem ser consideradas no momento de escolher um banco de dados NoSQL: a produtividade do programador e o desempenho no acesso aos dados.

5.1 Produtividade do programador

Quanto à produtividade do programador, de acordo com Salage e Fowler (2013, p. 2010), os tipos de sistemas NoSQL são mais apropriados para dados não uniformes. Se há dificuldades para obter um esquema forte capaz de suportar campos ad-hoc, então os bancos de dados NoSQL sem esquema podem oferecer uma ajuda considerável. Nesse sentido, cabe destacar o exemplo apresentado por Sadalage e Fowler (2013, p. 2010): “Bancos de dados relacionais não fazem um bom trabalho com dados que possuem muitos relacionamentos. Um banco de dados de grafos oferece uma API de armazenamento mais natural para esse tipo de dado e uma capacidade de consulta projetada em torno desses tipos de estruturas”.

Esse é o principal motivo pelo qual o modelo de programação de bancos de dados NoSQL pode melhorar a produtividade de uma equipe de desenvolvimento. O primeiro passo na avaliação, conforme destacam Sadalage e Fowler (2013, p. 2010), é:

- Examinar o que o software precisará fazer;
- Observar os recursos atuais;
- Verificar como o uso dos dados é mais apropriado.

A partir dessas três premissas é possível que um modelo de dados apropriado comece a se formar, sugerindo que, com o uso desse modelo, a programação se tornará mais fácil. Dada a possibilidade de diferentes modelos de dados ajustarem-se a diferentes partes dos dados, caso isso ocorra, sugere-se o uso de diferentes bancos de dados para diferentes aspectos.

Parafraseando Sadalage e Fowler (2013, p. 211), não há como medir apropriadamente a produtividade dos projetos, sendo uma solução para isso considerar as seguintes alternativas:

- Escolher alguns recursos iniciais do projeto e desenvolvê-los, ao mesmo tempo em que se presta atenção se é realmente fácil utilizar a tecnologia em questão;
- Considerar a criação dos mesmos recursos com alguns bancos de dados diferentes, para ver qual se adequa melhor ao problema que se quer solucionar.

5.2 Desempenho no acesso aos dados

Com relação ao desempenho no acesso aos dados de um banco de dados, Sadalage e Fowler (2013, p. 212) apontam que o mais importante a ser feito é testar seu desempenho em cenários adequados às necessidades do desenvolvedor. A melhor forma de avaliar o desempenho apropriadamente é criando uma solução para um problema específico, executando-a e medindo-a.

Neste cenário, Sadalage e Fowler (2013, p. 213) destacam:

> Não é possível testar todas as formas como o aplicativo será utilizado, sendo necessário criar um conjunto representativo de testes, selecionando cenários que sejam os mais comuns, os mais dependentes de desempenho e os que não pareçam se adaptar bem ao modelo de banco de dados proposto. Este último pode ser muito útil para alertar para quaisquer riscos fora de seus principais cenários de uso.

Diante de todas essas possibilidades de avaliação dos bancos de dados, Sadalage e Fowler (2013) ainda apontam que há muitas situações, na atualidade a maioria dos casos, em que é melhor permanecer com a opção padrão de um banco de dados relacional ao optar por um banco de dados NoSQL. Os autores esclarecem que, para escolher um banco de dados NoSQL, é necessário mostrar uma vantagem real em relação aos bancos de dados relacionais para cada situação específica. Portando, de acordo com Sadalage e Fowler (2013, p. 215):

- Os dois principais motivos para utilizar a tecnologia NoSQL são:
    - melhorar a produtividade do programador utilizando um banco de dados que se adapte melhor às necessidades de um aplicativo;
    - melhorar o desempenho no acesso aos dados por meio de alguma combinação na manipulação de volumes maiores de dados, reduzindo a latência e melhorando o rendimento.
- É essencial testar as expectativas sobre a produtividade do programador e/ou desempenho antes de decidir utilizar uma tecnologia NoSQL.

FINALIZANDO

Esta foi nossa última aula sobre os bancos de dados NoSQL, em que abordamos de modo geral a tecnologia NoSQL, estudando migrações de esquema, persistência poliglota, tecnologias além do NoSQL, tecnologia NewSQL e modos de avaliar e selecionar uma tecnologia de banco de dados para projetos específicos.

Aprendemos nesta aula que, mesmo que os bancos de dados NoSQL sejam livres de esquemas, é necessário planejar e projetar alguns aspectos do banco de dados, como os tipos de relacionamentos de um banco de grafos, nomes de colunas e linhas em um banco orientado a colunas, ou distribuição de chaves e estrutura dos dados em um banco orientado à chave-valor.

Frisamos, nesta aula, com base no estudo de Souza, Paula e Barros (2020), três importantes metodologias para mapeamento de bancos de dados relacionais para NoSQL: metodologia baseada em grafos**,** baseada em consultas e baseada na definição dos níveis físico e lógico dos dados. Tais metodologias são adequadas principalmente em um momento de adaptação de uma estrutura NoSQL, em conjunto com um banco de dados relacional, com base nos conceitos de persistência poliglota.

Além do NoSQL, conhecemos nesta aula outras tecnologias de bancos de dados, como os bancos de dados orientados a objetos e XML. Pais que não tenham tido expressividade relativa como os bancos de dados que estudamos, propõem a persistência segura com métodos de busca aos dados, tais como os modelos relacional e NoSQL. O modelo NewSQL é uma nova abordagem, que propõe assegurar a Atomicidade, Consistência, Isolamento e Durabilidade, conhecidas como propriedades ACID dos bancos de dados relacionais, que não

são suportadas pelos bancos de dados NoSQL. Assim, possibilita que os bancos de dados relacionais possam ser escalados verticalmente, de forma similar aos bancos de dados NoSQL, de modo que atualmente só podem ser escalados horizontalmente.

Encerramos esta aula analisando a abordagem de Saladage e Fowler (2013) com relação aos métodos de avaliação, para identificar a melhor estrutura de persistência de dados para os projetos em desenvolvimento. Identificamos, por meio dessa abordagem, que não existem métodos padronizados ou um passo a passo para esse tipo de avaliação, de modo que cada situação específica demanda um ou vários modelos de persistência, cabendo ao desenvolvedor avaliar todas as possibilidades e, com base em sua experiência e no desempenho das estruturas testadas, selecionar o melhor método de persistência dos dados e SGBD para o problema em questão.

REFERÊNCIAS

GROLINGER, K. et al. Data management in cloud environments: NoSQL and NewSQL data stores. **Journal Of Cloud Computing: Advances, Systems And Applications**, 2013. Disponível em: <https://link.springer.com/content/pdf/10.1186%2F2192-113X-2-22.pdf>. Acesso em: 1 maio 2021.

KARNITIS, G.; ARNICANS, G. Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation. In: COMPUTATIONAL INTELLIGENCE, COMMUNICATION SYSTEMS AND NETWORKS, 7. **Proceedings**... pg. 113–118, 2015. Disponível em: <https://www.researchgate.net/publication/308734675_Migration_of_Relational_Database_to_Document-Oriented_Database_Structure_Denormalization_and_Data_Transformation>. Acesso em: 1 maio 2021.

LI, X.; MA, Z.; CHEN, H. QODM: A query-oriented data modeling approach for NoSQL databases. In: IEEE WORKSHOP ON ADVANCED RESEARCH AND TECHNOLOGY IN INDUSTRY APPLICATIONS. **Proceedings**... p. 338–345, 2014. Disponível em: <https://www.semanticscholar.org/paper/QODM%3A-A-query-oriented-data-modeling-approach-for-Li-Ma/d4fd7d4acf194c6f8aceb1b868586802f8444103>. Acesso em: 1 maio 2021.

PANIZ, D. **NoSQL**: como armazenar os dados de uma aplicação moderna. São Paulo: Casa do Código, 2016.

PEREIRA, G. A. N. Conheça a geração de banco de dados NoSQL e NewSQL. **Devmedia**, 2015. Disponível em: <https://www.devmedia.com.br/perfil/gutierry-antonio-neto-pereira>. Acesso em: 1 maio 2021.

RYAN, J. Oracle vs. NoSQL vs. NewSQL Comparing Database Technology. **VoltDB**, 2018. Disponível em: <https://www.voltdb.com/wp-content/uploads/2018/01/VoltDB_Oracle_vs_NoSQL_vs_NewSQL_eBook_7March2019.pdf>. Acesso em: 1 maio 2021.

SADALAGE, P. J.; FOWLER, M. **NoSQL Essencial**: um guia conciso para o mundo emergente da persistência poliglota. São Paulo: Novatec, 2013.

SOUZA, V. C. O.; PAULA, M. M. V.; BARROS, T. C. G. M. Comparação de Metodologias de Migração de Bancos de Dados Relacionais para Bancos

Orientados a Documentos. In: COMPUTER ON THE BEACH, 11., 2-4 set. 2020, Balneário Camboriú, SC. **Anais**... p. 261-268, 2020.

STONEBRAKER, M. et al. The end of an architectural era: (it's time for a complete rewrite). In: INTERNATIONAL CONFERENCE ON LARGE DATA BASES, 33., 2007. **Proceedings**... p.1150–1160, 2007. Disponível em: <http://nms.csail.mit.edu/~stavros/pubs/hstore.pdf>. Acesso em: 1 maio 2021.

STONEBRAKER, M. SQL databases v. NoSQL databases. **Communications of the ACM**, v. 53, n. 4, p. 10-11, 2010.

ZHAO, G. et al. Schema conversion model of SQL database to NoSQL. In: INTERNATIONAL CONFERENCE OF P2P, PARALLEL, GRID, CLOUD INTERNET COMPUT, 9., 2014. **Proceedings**.... p. 355–362, nov. 2014. Disponível em: <https://dl.acm.org/doi/10.1109/3PGCIC.2014.137>. Acesso em: 1 maio 2021.