BANCO DE DADOS NOSQL VS. BANCO DE DADOS RELACIONAL:

Uma análise comparativa.

Marcos Paulo Silva Souza Bastos

Discente em Gestão da Tecnologia da Informação

marcos.silva231@fatec.sp.gov.br

Antonio Rafael Pepece Junior

Docente e Orientador

antonio.pepece@fatec.sp.gov.br

RESUMO

Com o passar dos anos, o volume de dados gerados mundialmente tem aumentado exponencialmente. Sabendo que dados são um dos principais pilares para tomadas de decisões e criação de estratégias empresariais mais assertivas e que a velocidade que estes dados são processados impactam diretamente no poder de ação destas empresas, inclusive para a sua sobrevivência em um mercado cada vez mais voraz, se torna necessário discutirmos como tais dados serão organizados e tratados pelos principais modelos de bancos de dados utilizados atualmente. O presente artigo apresenta uma análise comparativa entre bancos de dados NoSQL (*Not Only SQL*) e bancos de dados relacionais, com o objetivo de avaliar seus desempenhos, flexibilidades de esquema e confiabilidade em cenários do mundo real. A análise e comparação foi realizada com os bancos de dados gratuitos mais utilizados em cada categoria: MongoDB e MySQL. Foi utilizada uma metodologia cuidadosa para avaliar o desempenho, a escalabilidade e a flexibilidade do esquema de cada sistema.

Palavras-chave: Banco de Dados. Comparação. Relacional. NoSQL. Desempenho.

ABSTRACT

Over the years, the volume of data generated worldwide has increased exponentially. Knowing that data is one of the main pillars for decision-making and the creation of more assertive business strategies, and that the speed at which this data is processed directly impacts the power of action of these companies, even for their survival in an increasingly voracious market, it becomes necessary to discuss how such data will be organized and handled by the main database models currently in use. This article presents a comparative analysis between NoSQL (Not Only SQL) databases and relational databases, with the aim of evaluating their performance, schema flexibility, and reliability in real-world scenarios. The analysis and comparison were conducted with the most used free databases in each category: MongoDB and MySQL. A careful methodology was used to evaluate the performance, scalability, and schema flexibility of each system.

Keywords: Database. Comparison. Relational. NoSQL. Performance.

1 INTRODUÇÃO

Nos últimos anos, um assunto que é fortemente abordado e discutido no mercado global é o aumento da geração de dados e informações. Dados, segundo Rob e Coronel (2011), são fatos brutos, que ainda não foram processados, e que com isso, não possuem um significado explicito. Os autores complementam que diferente de dados, as informações são o produto do processamento dos dados brutos, revelando seu significado.

Rob e Coronel (2011) ainda enfatizam sobre a importância dos dados e informações, dizendo que “informações precisas, relevantes e rápidas são a chave para a boa tomada de decisões. A boa tomada de decisão é a chave para a sobrevivência de uma organização no ambiente global”. Isto demonstra que há um vínculo direto entre como os dados e informações são tratados, armazenados e utilizados e a sobrevivência das empresas, em um mundo onde presenciamos este aumento muito considerável de dados, com o passar dos anos.

De acordo com uma matéria do Insper (2021) e dados do *International Data Corporation* (IDC), no relatório *Data Age 2025*, a quantidade de dados consumido, copiado, capturado e criado em 2018 foi de 33 zettabytes. Isto equivale a 33 trilhões de gigabytes. De 2018 a 2020, este número saltou para 59 ZB. Estima-se que em 2025, este número chegue a 175 ZB. Nesta mesma reportagem é citado um estudo da Universidade de Portsmouth, que se estima que em 350 anos, teremos produzido uma quantidade de bits maior que o número de átomos existes em nosso planeta.

Diante deste cenário tão grandioso, um problema deve ser discutido: como trabalhar de forma organizada com tais dados? Qual modelo de banco de dados possui melhores condições para acompanhar este crescimento tão frenético de geração de dados, de forma a garantir um melhor desempenho e escalabilidade, entretanto, prezando pela consistência e integridade dos dados? E quais destes conseguirá melhor apoiar os negócios em suas atividades, garantindo tomadas de decisões mais rápidas, e assim a sua sobrevivência? Estas problemáticas nos fazem refletir sobre os modelos de banco de dados mais utilizados atualmente, o relacional e NoSQL (*Not only SQL)* e traz questionamentos que precisam ser averiguados, sobre o desempenho destes modelos de banco de dados, em testes de performance.

O presente artigo tem como objetivo geral a realização de uma análise comparativa entre os modelos de Banco de Dados Relacional e NoSQL, visando compreender melhor suas diferenças e até mesmo similaridades. Como objetivos específicos, podemos citar a comparação entre os modelos de banco de dados, nas suas características, vantagens, desvantagens, indicações e contraindicações, além dos cenários e áreas com maior sinergia para utilização de cada um destes modelos. Também será realizado a avaliação de desempenho destes modelos em um teste de performance, realizando diversas requisições a este banco de dados.

O presente artigo se justifica e apresenta-se como relevante diante da necessidade de avaliarmos qual dos modelos pode melhor atender as necessidades das empresas a trabalharem com os dados gerados por ela e fora dela, garantindo um bom desempenho, sem descartar a consistência e

integridade dos dados em suas aplicações, possibilitando assim uma escalabilidade das análises e condições de processamento destes dados, gerando, consequentemente, informações que poderão garantir a sobrevivência da empresa e uma melhor tomada de decisão. Diante dos fatos e estimativas apresentadas nesta introdução, não é difícil de considerar que no futuro, não tão distante, outras formas de armazenamento e gerenciamento de dados surgirão, e com isto, este artigo se tornará relevante para uma comparação entre estes possíveis novos modelos de banco de dados.

2 DESENVOLVIMENTO

Neste capítulo apresentaremos alguns dos principais conceitos relacionados a bancos de dados, discorreremos sobre os modelos relacional e NoSQL, os quais foram escolhidos para a presente análise comparativa, apresentando suas características, vantagens, desvantagens, requisitos para utilização e outras informações que sejam relevantes para que a análise seja realizada com sucesso. Apresentaremos também os SGBDs escolhidos para serem utilizados na comparação: o MySQL, representando banco de dados relacionais e o MongoDB representando os bancos de dados NoSQL. Será apresentada também suas características peculiares, requisitos para utilização, cenários mais indicados e outras informações que sejam relevantes para apresentação destes SGBDs.

2.1 Conceitualização - Banco de Dados

Rob e Coronel (2011), em seu livro *"Sistemas de banco de dados"*, nos apresenta a uma definição clara e objetiva sobre o que é banco de dados:

> Um banco de dados (ou base de dados) é uma estrutura computacional compartilhada e integrada que armazena um conjunto de:
> - Dados do usuário final, ou seja, fatos brutos de interesse para esse usuário;
> - Metadados, ou dados sobre dados, por meio dos quais os dados do usuário final são integrados e gerenciados.
>
> Os metadados fornecem uma descrição das características dos dados e do conjunto de relacionamentos que ligam os dados encontrados no banco de dados. Por exemplo, o componente de metadados armazena informações como o nome de cada elemento de dados, o tipo de valor (numérico, datas ou texto) armazenado, a possibilidade ou não de deixar esse elemento vazio, e assim por diante.
> Portanto, os metadados fornecem informações que complementam e expandem o valor e a utilização dos dados. Em resumo, os metadados trazem uma representação mais completa dos dados no banco.
> Dadas as características dos metadados, é possível ouvir a definição de um banco de dados como "um conjunto de dados auto descritivos" (ROB; CORONEL, 2011).

Ainda abordando a conceitualização de banco de dados, Elmasri e Navathe (2005) nos apresentam banco de dados como uma coleção de dados, coerente e lógica, que podem ser gravados e que possuem um significado implícito. Os autores ainda complementam dizendo que os bancos de dados representam aspectos do mundo real, podendo ser chamados as vezes de minimundo ou de universo de discurso (UoD).

Complementando os conceitos apresentados anteriormente, Date (2003) nos apresenta banco de dados como "uma coleção de dados persistentes, usadas pelos sistemas de aplicação de uma

determinada empresa". Este termo "persistente", segundo o autor, consiste na permanência do registro efetuado no banco de dados, onde o mesmo somente poderá ser removido quando ocorrer outra requisição explícita ao sistema gerenciador do banco de dados, solicitando sua remoção.

Com base nos argumentos apresentados pelos autores, podemos então definir Banco de Dados como uma coleção de dados auto descritiva, coerente e lógica, que possuem um significado implícito, utilizando dados persistidos para representar aspectos do mundo real, visando promover apoio aos seus usuários. E atualmente, é possível presenciarmos, em vários cenários e situações, a utilização de sistemas eletrônicos e aplicações, que se tornaram responsáveis pelo gerenciamento destes bancos dados, facilitando a organização, manipulação, consulta e outras necessidades que o usuário possa precisar. Estes sistemas e aplicações são os chamados "Sistemas de Gerenciamento de Banco de Dados".

2.2 Sistema de Gerenciamento de Banco de Dados - SGBD

Quando falamos de bancos de dados e sua utilização prática, é imprescindível falarmos também sobre sistemas de gerenciamento de bancos de dados, os chamados SGBDs. Também conhecidos como *Data Base Management System*, estes sistemas consistem em uma coleção de programas, que possibilitam a criação e a manutenção de um banco de dados pelo usuário, facilitando os processos de **definição**, **manipulação**, **compartilhamento** e **proteção** dos dados (ELMASRI; NAVATHE, 2005).

De acordo com Elmasri e Navathe (2005), quando falamos de **definição** dos dados, isto implica na especificação dos tipos de dados, suas estruturas e restrições de armazenamento. Já quando falamos de **construção**, o foco está no processo de armazenamento destes dados em mídias apropriadas e controladas pelos sistemas de gerenciamento de banco de dados. O **compartilhamento** consiste em permitir múltiplos acessos a este programa, de forma concorrente. E por fim, a **proteção** do sistema e sua manutenção por longos períodos, precavendo contra mau funcionamento ou falhas no *hardware* ou *software*, os chamados *crashes*.

A grande maioria dos SGBDs utilizam como linguagem base para realizar consultas, manipulações e definições nos bancos de dados, a linguagem SQL (*Structured Query Language*), desenvolvida pela IBM em 1970, baseada em álgebra e cálculo relacional, possuindo uma sintaxe bem simples e amigável. Esta linguagem é formatada por cinco principais partes: a DDL; a DML; a DCL; a DTL; e a DQL (CARDOSO; CARDOSO, 2013).

Cardoso e Cardoso (2013) no livro "Linguagem SQL" nos apresenta a definição de cada uma destas partes. Segundo as autoras:

> - A DDL (*Data Definition Language* – linguagem de definição de dados), que permite determinar o esquema do banco de dados, bem como alterá-lo e excluí-lo, e trabalha com os metadados;
> - A DML (*Data Manipulatons Language* – linguagem de manipulação de dados), que permite a manipulação dos dados, ou seja, a inclusão, alteração e exclusão de dados;
> - A DCL (*Data Control Language* – linguagem de controle de dados), que permite controlar a licença e a autorização de acesso dos usuários para com os dados;

> - A DTL (*Data Transaction Language* - linguagem de transação de dados), que oferece comandos para trabalhar com as transações;
> - A DQL (*Data Query Language* – linguagem de consulta de dados), que proporciona a consulta de dados, parte importantíssima dentro de um sistema de banco de dados [...] (CARDOSO; CARDOSO, 2013).

A utilização de um SGBD proporciona diversas vantagens e facilidades, para a gestão dos dados de uma empresa. Como vantagens na utilização de um SGBD, podemos listar: o controle de redundâncias, evitando gerar uma duplicação de esforços para o registro de informações, desperdício de espaço para armazenamento e inconsistência nos dados; a restrição de acesso e ação não autorizada a estas bases de dados, principalmente quando estamos diante de um cenário com vários usuários em uma base que possui dados sensíveis ou críticos para a estratégia da empresa, como por exemplo, dados de vendas e financeiros; armazenamento persistente dos dados, em estruturas que possibilitam a consulta destes dados de forma eficiente; backup e restauração em caso de falhas; múltiplas interfaces para os usuários com níveis diferentes de conhecimento; entre outras vantagens (ELMASRI; NAVATHE, 2005).

Após esta breve apresentação sobre banco de dados e o sistema de gerenciamento de bancos de dados e sua principal linguagem utilizada, apresentaremos os modelos de dados que serão comparados no presente artigo: Relacional e NoSQL.

2.3 Banco de Dados – Modelo Relacional

Bancos de dados do modelo relacional, também conhecidos como RDBMS, segundo a Oracle Brasil (s.d.), tratam-se de um tipo de banco de dados que armazenam e fornecem acesso a dados relacionados entre si, dados estes representados em tabelas, constituídas por atributos e registros (ou linhas), onde cada registro possui um identificador único chamado chave. Estes registros são compostos por valores que são armazenados nos atributos, também conhecidos como colunas. Cada tabela representa uma entidade, que pode se relacionar com outras entidades. Ou seja, as informações de uma tabela podem ter relação (ou ligação) com os dados armazenados em outras tabelas.

Este modelo de banco de dados é altamente utilizado por grandes empresas, principalmente em processos onde a consistência dos dados é essencial, como por exemplo, operações comerciais críticas, sistemas de pagamento eletrônico, faturamento entre outros. Essa confiabilidade gerada na consistência dos dados se dá em razão de algumas propriedades de transação deste modelo, como por exemplo o ACID: atomicidade, consistência, isolamento e durabilidade (ORACLE BRASIL, s.d.).

Reis (2018) nos apresenta uma definição para cada uma destas propriedades de transação. Segundo o autor, a **atomicidade** está relacionada a garantia da transação, onde “todas as operações da transação devem ser executadas com sucesso para que a transação tenha sucesso”. Ou seja, se uma única operação não for bem sucedida, toda a transação será cancelada.

Em seguida, temos a **consistência**, que segundo o autor, “permite assegurar que uma transação [...] leve o banco de dados de um estado válido a outro, mantendo a estabilidade do banco”. Com isto é possível evitar que os dados sejam corrompidos ou que registros inconsistentes sejam gerados, como por exemplo a venda de um produto que não esteja cadastrado (REIS, 2018).

O **isolamento**, outra propriedade do modelo relacional, auxilia nas execuções concorrentes, executadas por vários usuários no banco de dados ao mesmo tempo, fazendo com que o banco de dados fique no mesmo estado em que ele estaria caso as transações tivessem sido realizadas de forma sequencial (REIS, 2018).

Por fim, a **durabilidade**, responsável por garantir que a transação, uma vez executada, seja de fato registrada, mesmo que outros problemas possam ser gerados por meio desta ação, como por exemplo, travamento do sistema, entre outros problemas. Os registros são gravados em "dispositivos de memória permanente (não-volátil), como discos rígidos, de modo que os dados estejam sempre disponíveis, mesmo que a instância do BD seja reiniciada" (REIS, 2018).

2.4 Banco de Dados – Modelo NoSQL

A simplicidade do modelo relacional, segundo Lóscio, Oliveira e Pontes (2011), fez com que ele se tornasse o modelo mais utilizado e aceito por grande parte das empresas e aplicações. Entretanto, com o avanço da tecnologia surgiu a necessidade do desenvolvimento de novos requisitos nos modelos de banco de dados, visando atender aplicações "não-convencionais": a manipulação de grandes volumes de dados não estruturados ou semi-estruturados, o armazenamento de diferentes tipos de dados, como por exemplo, vídeos, som e imagem, entre outros novos requisitos.

Com o desenvolvimento destes requisitos, surgiu o modelo de banco de dados não relacional, também conhecido como NoSQL (*Not Only SQL* – não apenas SQL). De acordo com Lóscio, Oliveira e Pontes (2011), o termo NoSQL faz referência a sistemas de gerenciamento de bancos de dados (SGBDs) "[...] que não adotam o modelo relacional e são mais flexíveis quanto às propriedades ACID. Esta flexibilidade torna-se necessária devido aos requisitos de alta escalabilidade necessários para gerenciar grandes quantidades de dados, bem como para garantir a alta disponibilidade dos mesmos [...]".

Soares e Matos (2017) complementam dizendo que este modelo de banco de dados prioriza a disponibilidade dos dados, e não a consistência absoluta destes. Este tipo de modelo, ainda segundo estes autores, é mais indicado para "aplicações e sistemas que utilizam dados não estruturados e que necessitam de grande desempenho em escalabilidade e no acesso aos dados", como por exemplo, sistemas que funcionam em tempo real.

Um ponto de destaque no modelo NoSQL, está nas suas propriedades de transação, que diferente do modelo relacional, neste é utilizado a sigla BASE (*Basically Available, Soft State and Eventual Consistency* - Basicamente Disponível, Estado Suave e Consistência Eventual). **Basicamente disponível** refere-se ao fato de que bancos de dados NoSQL terá uma resposta para cada solicitação, inclusive quando ocorrer erros, falhas no sistema, ou não sejam retornados os dados esperados. Já a propriedade **"estado suave"** diz respeito a alteração dos dados, que são realizados de forma contínua, visando dar mais consistência a eles. Ou seja, os dados disponíveis hoje, não necessariamente serão iguais aos dados disponíveis de amanhã (SALEM, 2021).

Por fim, temos a última propriedade, a **“consistência eventual”**, ou seja, os dados no modelo NoSQL podem ou não ser consistentes (por isso a eventualidade). Nenhuma segurança é dada sobre quando os dados estarão de fato consistentes. Essa propriedade em especial é justamente o oposto das principais propriedades do modelo relacional, que presa pela consistência dos dados ao ponto de impedir registros inconsistentes (SALEM, 2021).

Ainda sobre o modelo NoSQL, Toth (2011) nos apresenta três principais tipos de bancos de dados não relacionais: o tipo chave-valor, onde os dados são tratados similares a uma tabela *hash*. Estes dados podem ser uma *string*, um número inteiro, uma matriz ou um objeto. Eles são armazenados juntamente com um índice, que é utilizado para mapeá-lo; o tipo documento, ou *document store*, que é similar ao tipo chave-valor, entretanto, os dados necessitam ser de alguns tipos especiais, como por exemplo, xml, JSON, JSON-BLOB e outros; e por fim, o tipo grafos, ou graph, que utiliza vértices e arestas, representando caminhos que os dados poderão seguir.

3 METODOLOGIA

Neste capítulo, será apresentado a metodologia utilizada para conduzir a análise comparativa entre bancos de dados relacionais e NoSQL. Esta seção detalhará as etapas, os procedimentos e as ferramentas empregadas ao longo da pesquisa, permitindo uma compreensão clara de como as análises foram conduzidas, e como os dados foram coletados e processados.

3.1 Seleção dos SGBDs

O primeiro passo destacado na metodologia é a seleção cuidadosa dos bancos de dados que seriam incluídos na análise. Foi optado por considerar dois dos bancos de dados mais amplamente utilizados em ambientes NoSQL e relacionais: MongoDB e MySQL respectivamente. Essa seleção foi baseada em sua popularidade e representatividade, tendo em vista que, dos SGBDs gratuitos no mercado, estes sistemas estão entre as primeiras colocações no *DB-Engines Ranking*, comparando cada um com os seus modelos similares.

3.2 Coleta de Dados

A coleta de dados desempenha um papel fundamental nesta análise comparativa. Para que seja possível obter resultados precisos e significativos, foi implementado um conjunto de critérios para a coleta de dados relevantes de maneira sistemática. Foram coletadas informações sobre os seguintes aspectos:

Desempenho: Foi medido o desempenho de cada banco de dados em termos de velocidade de consulta, tempo de resposta e escalabilidade. Para isso, foram desenvolvidos cenários de teste realistas que refletem situações do mundo real.

Disponibilidade e Confiabilidade: Também foi avaliado a disponibilidade e a confiabilidade dos bancos de dados por meio de testes de falhas e recuperação.

3.3 Ferramentas e Ambiente de Teste

Para a realização dos testes e coletar dados de maneira eficaz, foram empregues ferramentas e ambientes de testes adequados. A ferramenta escolhida foi o *Apache JMeter*^{TM}, responsável por gerar cargas de trabalho de teste, monitorar o desempenho e coleta de análises relevantes.

Segundo Noleto (2020), o *Apache JMeter*^{TM} foi criado em 2007, pela Apache Software Foundation. Trata-se de uma ferramenta escrita em Java, de código aberto, muito utilizada em testes de performance, avaliando o desempenho de uma aplicação, a submetendo a situações extremas, como por exemplo, milhares de acessos simultâneos a um servidor. O autor complementa dizendo que o *Apache JMeter*^{TM} possui uma grande variedade de recursos, como por exemplo, configuração de ambientes de testes, extrator de expressão regular, componentes que auxiliam na avaliação das respostas exibidas, recursos para gerenciamento de scripts de testes e componentes para controle de cenários.

O ambiente de teste foi configurado de acordo com as melhores práticas recomendadas para cada banco de dados, garantindo uma comparação justa entre eles.

3.4 Cenário de Teste

Com o objetivo de realizar uma comparação equilibrada entre os SGBDs, foi definido uma estrutura base de dados, que foi utilizada para os testes. Para o banco de dados MySQL, foi criada uma *Data Base*, intitulada "*dbtggtifatecassis*", ao qual foi criada uma tabela chamada "aluno", composta pelos seguintes atributos: ***idAluno*** *integer(11) NOT NULL AUTO_INCREMENT*, ***nomeAluno*** *varchar(80) DEFAULT NULL,* ***endereco*** *varchar(100) DEFAULT NULL,* ***curso*** *varchar(100) DEFAULT NULL*. A chave primária desta tabela é o atributo ***idAluno***. Já para o banco de dados MongoDB, por ser um banco de dados orientado a documentos, foi criada uma *colletion*, intitulada também como "*dbtggtifatecassis*", que armazenará os documentos criados. Os campos (chaves e valores) que irão compor os documentos serão os mesmos utilizados no MySQL. Ambos os bancos de dados foram configurados localmente (*localhost*), em suas respectivas portas (MySQL: porta 3306; MongoDB: porta 27017).

Na ferramenta *JMeter*^{TM} foram criados scripts que executaram as seguintes requisições aos bancos de dados: inserção, seleção, atualização (update), e deleção. Os dados que foram utilizados e condições de filtros para as requisições foram as mesmas para ambos os bancos de dados.

Para os testes de desempenho, foram executados quatro (4) cenários:

Cenário 01) Um (01) usuário virtual (*thread*), realizando uma (1) requisição sequencialmente;

Cenário 02) Cem (100) usuários virtuais (*threads*), realizando uma (1) requisição sequencialmente;

Cenário 03) Um mil (1.000) usuários virtuais (*threads*), realizando uma (1) requisição sequencialmente;

Cenário 04) Dez mil (10.000) usuários virtuais (*threads*), realizando uma (1) requisição sequencialmente;

Destacamos que cada cenário foi executado em cada banco de dados, em cada um dos comandos básicos (inserção, seleção, atualização e deleção).

4 ANÁLISE DE RESULTADOS

Após execução dos testes de carga e estresse dos bancos de dados, foi possível identificamos comportamentos similares em alguns dos cenários listados acima, em ambos os bancos de dados. Entretanto, conforme o número de usuários virtuais (threads) aumentava, a diferença entre os SGBDs ficou mais evidentes.

Abordando sobre o primeiro cenário, foi possível notarmos um comportamento idêntico em ambos os bancos de dados, tendo em vista que o primeiro teste consistia em apenas um usuário virtual realizando uma única requisição, para cada comando base. O tempo de resposta para estas requisições foram inexpressíveis, sendo o retorno extremamente rápido (até 1 segundo).

Já no segundo cenário, onde 100 usuários realizaram requisições ao banco de dados, ainda tivemos sucesso em 100% das requisições, em ambos os bancos de dados. Entretanto, destaca-se o tempo de resposta em cada um dos bancos, onde o MongoDB teve um tempo de resposta em média 25% menor, comparado com o MySQL.

Tabela 1 – Resultado do Teste de Carga/ Estresse – Cenário 2 – Banco MySQL

Thread	Comando	# Sucesso	# Falha	# Total	% Sucesso	% Falha	Tempo (milissegundos)
100	insert	100	0	100	100,00%	0,00%	1647
100	select	100	0	100	100,00%	0,00%	1432
100	update	100	0	100	100,00%	0,00%	1368
100	delete	100	0	100	100,00%	0,00%	1851

Fonte: Autor.

Tabela 2 – Resultado do Teste de Carga/ Estresse – Cenário 2 – Banco MongoDB

Thread	Comando	# Sucesso	# Falha	# Total	% Sucesso	% Falha	Tempo (milissegundos)
100	create_document	100	0	100	100,00%	0,00%	1235
100	find_document	100	0	100	100,00%	0,00%	1198
100	update_document	100	0	100	100,00%	0,00%	1112
100	delete_document	100	0	100	100,00%	0,00%	1112

Fonte: Autor.

A partir do terceiro cenário os testes começaram a apresentar um resultado diferente, com um grande volume de erros nas requisições, para ambos os bancos, onde a maioria das requisições falharam, tendo em vista o alto acesso demandado para cada um deles. Neste cenário, tivemos o MongoDB com uma média de 32,68% de sucesso nas requisições, contra uma média de 15% de sucesso nas requisições

realizadas ao MySQL. Entretanto, o MongoDB, no quesito tempo de resposta, ficou com uma diferença de 76% a mais, quando comparado com o tempo de resposta das requisições efetivadas ao MySQL.

Tabela 3 – Resultado do Teste de Carga/ Estresse – Cenário 3 – Banco MySQL

Thread	Comando	# Sucesso	# Falha	# Total	% Sucesso	% Falha	Tempo (milissegundos)
1000	insert	148	852	1000	14,80%	85,20%	2586
1000	select	152	848	1000	15,20%	84,80%	5166
1000	update	150	850	1000	15,00%	85,00%	3350
1000	delete	150	850	1000	15,00%	85,00%	2680

Fonte: Autor.

Tabela 4 – Resultado do Teste de Carga/ Estresse – Cenário 3 – Banco MongoDB

Thread	Comando	# Sucesso	# Falha	# Total	% Sucesso	% Falha	Tempo (milissegundos)
100	create_document	373	627	1000	37,30%	62,70%	2121
100	find_document	334	666	1000	33,40%	66,60%	5549
100	update_document	243	757	1000	24,30%	75,70%	6292
100	delete_document	357	643	1000	35,70%	64,30%	10407

Fonte: Autor.

Por fim, o último cenário nos apresenta um resultado diferente dos demais casos apresentados anteriormente. Ambos os bancos de dados apresentaram uma alta taxa de requisições com falhas, entretanto, o SGBD MySQL obteve um percentual de sucesso 96% maior, e um tempo de resposta aproximadamente 53 vezes menor, em comparação com o SGBD MongoDB.

Tabela 5 – Resultado do Teste de Carga/ Estresse – Cenário 4 – Banco MySQL

Thread	Comando	# Sucesso	# Falha	# Total	% Sucesso	% Falha	Tempo (milissegundos)
10000	insert	152	9848	10000	1,52%	98,48%	31230
10000	select	150	9850	10000	1,50%	98,50%	18417
10000	update	152	9848	10000	1,52%	98,48%	21982
10000	delete	152	9848	10000	1,52%	98,48%	18006

Fonte: Autor.

Tabela 6 – Resultado do Teste de Carga/ Estresse – Cenário 4 – Banco MongoDB

Thread	Comando	# Sucesso	# Falha	# Total	% Sucesso	% Falha	Tempo (milissegundos)
10000	create_document	22	9978	10000	0,22%	99,78%	582304
10000	find_document	0	10000	10000	0,00%	100,00%	4084788
10000	update_document	0	10000	10000	0,00%	100,00%	16563
10000	delete_document	2	9998	10000	0,02%	99,98%	14637

Fonte: Autor.

5 CONSIDERAÇÕES FINAIS

Após a execução dos testes de carga e estresse nos bancos de dados MySQL e MongoDB, foi possível obtermos informações relevantes sobre o desempenho, a escalabilidade e a adequação de cada sistema para diferentes cenários de uso.

A análise comparativa mostrou que ambos os bancos de dados apresentaram comportamentos distintos conforme o número de requisições aumentou. Em cenários de menor carga, ambos os sistemas se comportaram de forma satisfatória. Por exemplo, no cenário 2, tanto MySQL quanto MongoDB alcançaram 100% de sucesso nas operações básicas, como inserção, seleção, atualização e exclusão, mas com tempos de resposta ligeiramente diferentes (MongoDB com um tempo 25% menor em comparação ao MySQL).

Entretanto, no terceiro cenário, o MongoDB manteve a vantagem sobre o MySQL, no quesito percentual de requisições com sucesso, possuindo um pouco mais do que o dobro de requisições com sucesso. Porém, neste mesmo cenário, o SGBD relacional apresentou um tempo de resposta as requisições muito superiores ao tempo apresentado pelo SGBD não relacional, com uma diferença de 76% entre eles. Já o cenário 4, por mais que o percentual de falhas para ambos os bancos de dados tenham sido elevados, ainda assim, o MySQL conseguiu contornar a diferença apresentada no cenário 3, ficando à frente do MongoDB tanto em tempo de resposta das requisições, quando no percentual de sucesso dos testes.

Esses resultados indicam que, embora o MongoDB seja uma excelente escolha para aplicações que exigem alta flexibilidade e escalabilidade de esquema, ele pode não ser a melhor opção para cenários de alta carga que requerem uma performance consistente e tempos de resposta rápidos. O MySQL, por outro lado, mostrou um resultado melhor (mesmo que com um cenário com altos percentuais de falhas) em situações de carga intensa, destacando-se pela sua confiabilidade e eficiência em operações de leitura e escrita em massa.

Sendo assim, podemos dizer que a escolha entre MySQL e MongoDB deve ser baseada nas necessidades específicas de cada situação, ou projeto. Para aplicações que demandam alta escalabilidade e flexibilidade, o MongoDB pode ser a opção mais adequada. Já para sistemas que requerem alta performance sob cargas pesadas e consistência de dados, o MySQL se apresenta como a melhor escolha. Ambos os bancos de dados têm seus méritos e são ferramentas poderosas quando usadas nos contextos apropriados.

6 REFERÊNCIAS

CARDOSO, Virgínia. CARDOSO, Giselle. **Linguagem SQL**: fundamentos e práticas. São Paulo: Saraiva, 2013.

DATE, C.J. **Introdução a Sistemas de Bancos de Dados**. Tradução: Daniel Vieira. Rio de Janeiro: Elsevier, 2003.

DB-ENGINES RANKING. DB-Engines, 2023. Disponível em: <https://db-engines.com/en/ranking>. Acesso em: 10 set. 2023.

ELMASRI, Ramez. NAVATHE, Shamkant B. **Sistemas de banco de dados**. 4 ed. Pearson Education, 2005.

LÓSCIO, Bernadette Farias. OLIVEIRA, Hélio Rodrigues de. PONTES, Jonas Césa de Souza. **NoSQL no desenvolvimento de aplicações Web colaborativas**. VIII Simpósio Brasileiro de Sistemas Colaborativos, 2011.

NOLETO, Cairo. **JMeter e os testes de performance!** Trybe. Disponível em: <https://blog.betrybe.com/desenvolvimento-web/jmeter/>. Acesso em: 17 set. 2023.

O MAR DE DADOS VIROU UM OCEANO E NÃO PARA DE CRESCER. MAS NEM TUDO É APROVEITADO. Insper, 2021. Disponível em: <https://www.insper.edu.br/noticias/o-mar-de-dados-virou-um-oceano-e-nao-para-de-crescer-mas-nem-tudo-e-aproveitado/>. Acesso em: 16 set. 2023.

O que é um banco de dados relacional (RDBMS)? Oracle Brasil. s.d. Disponível em: <https://www.oracle.com/br/database/what-is-a-relational-database/>. Acesso em: 25 out. 2023.

REIS, Fábio. **Conceitos de Bancos de Dados – O que significa ACID.** Bóson Treinamentos em Ciência e Tecnologia. Disponível em: <http://www.bosontreinamentos.com.br/bancos-de-dados/conceitos-de-bancos-de-dados-o-que-significa-acid/>. Acesso em: 25 out. 2023.

ROB, Peter. CORONEL, Carlos. **Sistemas de banco de dados:** Projeto, implementação e gerenciamento. 8 ed. Cengate Learning, 2011.

SALEM, Mahmoud. **ACID VS BASE.** LinkedIn. Disponível em: <https://www.linkedin.com/pulse/acid-vs-base-mahmoud-salem/>. Acesso em: 26 out. 2023.

SOARES, Allexandre Sampaio Santos; MATOS, Pablo Freire. **Uma Análise Comparativa entre Sistemas Gerenciadores de Bancos de Dados NoSQL no contexto de Internet das Coisas**. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS (SBBD), 32. , 2017, Uberlândia/MG. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2017 . p. 306-311. ISSN 2763-8979.

TOTH, Renato Molina. **Abordagem NoSQL** – uma real alternativa. Sorocaba, 2011.