INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS GOIÂNIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANÁLISE COMPARATIVA DE DESEMPENHO ENTRE BANCOS DE DADOS RELACIONAL E NÃO RELACIONAL EM UMA CONSULTA A UMA BASE DE DADOS DE E-COMMERCE

THIAGO GOMES ROSA

Trabalho de Conclusão de Curso – Bacharelado em Sistemas de Informação

Orientador: **Prof. Me. Ariel Cardoso Mendes**

GOIÂNIA-GO

2025

INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor: Thiago Gomes Rosa
Matrícula: 20191011090234
Título do Trabalho: ANÁLISE COMPARATIVA DE DESEMPENHO ENTRE BANCOS DE DADOS RELACIONAL E NÃO RELACIONAL EM UMA CONSULTA A UMA BASE DEDADOS DE E-COMMERCE

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Goiânia ,11/04/2025

gov.br Documento assinado digitalmente
THIAGO GOMES ROSA
Data: 11/04/2025 13:52:26-0300
Verifique em https://validar.iti.gov.br

_____
Assinatura do Autor e/ou Detentor dos Direitos Autorais

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS GOIÂNIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANÁLISE COMPARATIVA DE DESEMPENHO ENTRE BANCOS DE DADOS RELACIONAL E NÃO RELACIONAL EM UMA CONSULTA A UMA BASE DE DADOS DE E-COMMERCE

THIAGO GOMES ROSA

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação, do Instituto Federal de Educação, Ciência e Tecnologia, Câmpus Goiânia.

Orientador: Prof. Me. Ariel Cardoso Mendes

GOIÂNIA-GO

2025

R788a Rosa, Thiago Gomes.
Análise comparativa de desempenho entre bancos de dados relacional e não relacional em uma consulta a uma base de dados de e-commerce / Thiago Gomes Rosa. – Goiânia: Instituto Federal de Educação, Ciência e Tecnologia de Goiás, 2025.
63f.

Orientação: Prof. Me. Ariel Cardoso Mendes.

TCC (Trabalho de Conclusão de Curso) – Curso de Bacharelado em Sistemas de Informação, Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

1. Banco de dados. 2. MySQL. 3. Comércio eletrônico. I. Mendes, Ariel Cardoso (orientação). II. Instituto Federal de Educação, Ciência e Tecnologia de Goiás. III. Título.

CDD 005.13

Ficha catalográfica elaborada pela bibliotecária Lana Cristina Dias Oliveira CRB1/ 2.631
Biblioteca Professor Jorge Félix de Souza,
Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia.

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS GOIÂNIA

Folha de Aprovação

THIAGO GOMES ROSA

ANÁLISE COMPARATIVA DE DESEMPENHO ENTRE BANCOS DE DADOS RELACIONAL E NÃO RELACIONAL EM UMA CONSULTA A UMA BASE DE DADOS DE E-COMMERCE

Trabalho de Conclusão de Curso apresentado no Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Campus Goiânia, como requisito básico para a conclusão do Curso de Bacharelado em Sistemas de Informação.

Orientador(a): Prof. Me. ARIEL CARDOSO MENDES

Aprovada em 18 de março de 2025

BANCA EXAMINADORA

Prof. Me. ARIEL CARDOSO MENDES- Orientador

Coordenação de Informática/DAAIV/IFG - Campus Goiânia

Prof. Me. DORY GONZAGA RODRIGUES

Coordenação de Informática/DAAIV/IFG - Campus Goiânia

Prof. Me. CAROLINA SZKRUC DE CARVALHO

Coordenação de Informática/DAAIV/IFG - Campus Goiânia

GOIÂNIA

2025

Documento assinado eletronicamente por:

- **Carolina Szkruc de Carvalho, PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO** , em 20/03/2025 16:03:08.
- **Dory Gonzaga Rodrigues, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 20/03/2025 14:43:27.
- **Ariel Cardoso Mendes, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 19/03/2025 10:24:00.

Este documento foi emitido pelo SUAP em 19/03/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifg.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 628995
Código de Autenticação: e8c60b1d57

A minha querida esposa e meu filho,
com amor e carinho.

AGRADECIMENTOS

Agradeço, primeiramente a Deus, fonte de força, sabedoria e perseverança, e à intercessão de Nossa Senhora, que me guiou e protegeu em cada passo desta jornada.

À minha esposa, meu grande amor, pelo apoio incondicional, paciência e incentivo nos momentos mais desafiadores, e ao meu amado filho, que é minha maior motivação e inspiração para seguir sempre em frente.

À minha família, que sempre esteve ao meu lado, oferecendo amor.

Ao meu orientador, Prof. Me. Ariel Cardoso Mendes, pela dedicação, paciência e valiosas orientações que tornaram este trabalho possível. Seu conhecimento e apoio foram essenciais para o desenvolvimento deste estudo.

A todos que de alguma forma contribuíram para a realização deste trabalho, meu sincero muito obrigado.

RESUMO

A presente pesquisa pretendeu analisar e comparar o desempenho de bancos de dados relacionais e não relacionais em buscas por produtos em um e-commerce determinando qual abordagem é mais eficiente em um cenário específico. Primeiramente, foi traçado um percurso que perpassa pela compreensão dos conceitos fundamentais de banco de dados, dos bancos de dados relacionais com destaque para o Mysql, dos bancos de dados não relacionais com destaque para o MongoDB, a comparação entre esses dois modelos e por fim, uma breve análise do sistema de e-commerce. Posteriormente foi conduzido um experimento em um ambiente controlado, no qual ambos os sistemas gerenciadores de banco de dados (SGBD) foram testados em relação a métricas como tempo médio de resposta, consumo de CPU e memória, taxa de leitura e escrita em disco e volume de consultas realizadas. Os resultados obtidos permitiram avaliar que o MongoDB se destacou em áreas fundamentais como a quantidade de operações realizadas e o tempo médio de resposta, desta forma, ele aparece como a escolha mais indicada para resolver o problema apresentado.

PALAVRAS-CHAVE: Banco de Dados. SQL. NoSQL. *E-commerce.*

ABSTRACT

This research aimed to analyze and compare the performance of relational and non-relational databases in product searches in an e-commerce, determining which approach is more efficient in a specific scenario. First, a path was traced that goes through the understanding of the fundamental concepts of databases, relational databases with emphasis on MySQL, non-relational databases with emphasis on MongoDB, the comparison between these two models and finally, a brief analysis of the e-commerce system. Subsequently, an experiment was conducted in a controlled environment, in which both database management systems (DBMS) were tested in relation to analyses such as average response time, CPU and memory consumption, read and write rates on disk and volume of queries performed. The results obtained allowed us to evaluate that MongoDB stood out in fundamental areas such as the number of operations performed and the average response time, thus, it appears as the most suitable choice to solve the presented problem.

Keywords: Database. SQL. NoSQL. E-commerce.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	Application Programming Interface
BSON	Binary JSON
CAP	Consistência, Disponibilidade e Tolerância à Partição
CPU	Central Processing Unit
CSV	Comma-Separated Values
ERP	Enterprise Resource Planning
GTID	Global Transaction Identifier
I/O	Input/Output
JSON	JavaScript Object Notation
LGPD	Lei Geral de Proteção de Dados
MB	Megabyte
NoSQL	Not Only SQL
RBAC	Role-Based Access Control
RAM	Random Access Memory
SBR	Statement-Based Replication
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
SSL/TLS	Secure Sockets Layer / Transport Layer Security
VM	Virtual Machine

SUMÁRIO

1 INTRODUÇÃO

O avanço da tecnologia mudou drasticamente a maneira como as pessoas fazem compras, tornando o e-commerce uma das principais formas de consumo atualmente. O comércio continuou a crescer e se adaptar às novas tecnologias, reforçando sua posição central no mercado de consumo. Atualmente os consumidores encontram uma variedade enorme de produtos e serviços com apenas alguns cliques. Podendo comparar preços facilmente e escolher entre diferentes fornecedores sem precisar sair de casa para finalizar suas compras. No entanto, para garantir que os usuários tenham uma experiência satisfatória, é crucial que essas plataformas apresentem desempenho, principalmente quando se trata de eficiência nas buscas por produtos. A agilidade no tempo de resposta pode influenciar diretamente na decisão de compra, caso a busca leve mais tempo do que o previsto, ou até mesmo, ser causa de desistência de uma possível compra.

Diante desse cenário, se propõe a seguinte questão norteadora: qual modelo de banco de dados é mais adequado para aplicações de e-commerce? Alguns estudos apontam que os bancos de dados relacionais como a melhor escolha, pois oferecem maior integridade e consistência aos dados transacionais. Segundo Erickson (2024), o MySQL é amplamente utilizado no gerenciamento de dados de clientes e produtos para sites de e-commerce, sendo uma tecnologia essencial para consultas complexas e grandes conjuntos de dados.

Por outro lado, existem pesquisas que indicam que bancos de dados NoSQL são mais eficientes para lidar com grandes volumes de informações e proporcionam maior escalabilidade em ambientes de alta demanda. De acordo com a Oracle (s.d.), bancos de dados NoSQL oferecem esquemas flexíveis e suportam diferentes modelos de dados, tornando-se ideais para aplicativos que exigem baixa latência e respostas rápidas, como plataformas de e-commerce e jogos online.

Diante dessas perspectivas, torna-se fundamental investigar qual modelo de banco de dados oferece maior eficiência para buscas em e-commerce, proporcionando uma base sólida para a escolha estratégica do SGBD mais adequado a esse cenário.

Desta forma, a presente pesquisa tem como objetivo analisar e comparar os resultados de cada bancos de dados SQL e NoSQL, por meio de testes em uma situação específica de busca por produtos em um e-commerce, determinando qual abordagem é mais eficiente. Como objetivos específicos se tenciona: compreender os conceitos, arquiteturas e características fundamentais dos bancos de dados MySQL e MongoDB, aplicar uma metodologia de avaliação, fazer uma análise e a partir dela comparar métricas de desempenho dos objetos dentro de um

cenário controlado e por fim, discutir os resultados obtidos mediante os resultados da análise comparativa de desempenho.

Para o alcance dos objetivos supracitados, se propõe uma metodologia cuja abordagem é qualitativa. Wainer (2007, p. 28) apresenta que a pesquisa qualitativa se configura por ser um "estudo aprofundado de um sistema no ambiente onde ele está sendo usado, ou, em alguns casos, onde se espera que o sistema seja usado". Quanto ao seu objetivo, a pesquisa será exploratória, pois em um primeiro momento serão levantadas as informações dos objetos de estudo como fundamentação para a posterior análise deles. Severino (2017, p.94) reverbera que "A pesquisa exploratória busca apenas levantar informações sobre um determinado objeto, delimitando assim um campo de trabalho, mapeando as condições de manifestação desse objeto".

No que tange a natureza das fontes utilizadas para a abordagem e tratamento do objeto, a pesquisa será bibliográfica e experimental. A revisão bibliográfica se faz necessária pois constitui a base de conhecimento indispensável para a compreensão e análise dos cenários apresentados. Segundo Severino (2017, p. 93) a pesquisa bibliográfica "Utiliza-se de dados ou de categorias teóricas já trabalhados por outros pesquisadores e devidamente registrados".

Com relação à natureza experimental, para a realização do comparativo entre os bancos de dados não relacionais e relacionais, serão implementados alguns testes, que colocarão ambas as bases à prova e avaliará conforme os parâmetros estabelecidos, possibilitando o alcance de resultados esperados ou inesperados. Para isso, será realizado um experimento onde ambos os SGBDs serão submetidos a uma consulta de pesquisa por um produto, permitindo avaliar métricas essenciais como tempo de resposta, consumo de CPU e memória, taxa de leitura e escrita em disco e número total de consultas realizadas em um período determinado.

A partir dessa análise comparativa, este estudo busca oferecer uma resposta fundamentada sobre qual modelo de banco de dados se mostra mais eficiente para esse tipo de operação. Embora existam diversos fatores que influenciam o desempenho de um e-commerce, a escolha do SGBD é um dos elementos centrais, impactando diretamente na escalabilidade, confiabilidade e velocidade das consultas. Ao final, espera-se contribuir para a tomada de decisão de desenvolvedores e arquitetos de sistemas, fornecendo uma visão mais clara sobre as vantagens e desafios de cada abordagem em um contexto de busca de produtos em plataformas digitais.

O trabalho está estruturado em cinco capítulos. O segundo capítulo aborda o referencial teórico, reunindo conceitos fundamentais sobre bancos de dados SQL e NoSQL, suas arquiteturas, vantagens e desafios. No terceiro capítulo, detalha-se a metodologia empregada,

descrevendo o ambiente experimental, os critérios de comparação e as métricas avaliadas. O quarto capítulo expõe e analisa os resultados obtidos no experimento, comparando o desempenho dos bancos de dados testados. Por fim, o quinto capítulo apresenta as conclusões, destacando as principais observações e recomendações obtidas com este estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção traz os conceitos essenciais para a compreensão dos principais temas da pesquisa. Se propõe explorar os fundamentos e características de Banco de Dados, com ênfase nas abordagens Relacionais e Não Relacionais, destacando suas arquiteturas, modelos de armazenamento, benefícios e desafios.

2.1 Conceitos Fundamentais de Banco de Dados

Um Sistema de gerenciamento de bancos de dados (SGBD), conforme afirma Ramakrishnan e Gehrke (2015, p. 3) “é um software projetado para auxiliar a manutenção e utilização de vastos conjuntos de dados.” Por isso, são de grande importância nas organizações atualmente, responsáveis por persistir e gerir as informações, além de desempenhar papel essencial no funcionamento de aplicações e sistemas. Segundo Elmasri e Navathe (2011, p.3) “um banco de dados pode ser definido como uma coleção estruturada de dados que representam uma fração do mundo real, também chamada de minimundo.” Essa estruturação permite que as informações sejam armazenadas e acessadas de maneira eficiente, garantindo a integridade e a consistência dos dados.

Atualmente, os principais modelos utilizados se dividem entre relacionais e não relacionais. Mesmo possuindo a mesma função, de persistir e manipular os dados, seguem arquiteturas diferentes. Os modelos relacionais se caracterizam por ser representados pelo uso de tabelas estruturadas, além de seguir propriedades ACID, que se referem a Atomicidade, Consistência, Isolamento e Durabilidade. Já os modelos não relacionais, se destacam por sua capacidade de gerenciar grande volume de dados, além da sua flexibilidade e escalabilidade. (LÓSCIO et al., 2011)

2.2. Banco de Dados Relacionais

Em meados da década de 1970 surge o modelo relacional como conhecemos hoje, proposto por Edgar F. Codd, se consolidando no mercado a partir de 1980. Essa inovação trouxe benefícios que se estendem até os dias atuais. Segundo Elmasri & Navathe (2011, p.2), "É correto afirmar que os bancos de dados desempenham um papel crítico em quase todas as áreas em que os computadores são usados, incluindo negócios, comércio eletrônico, engenharia, medicina, genética, direito, educação e biblioteconomia."

O modelo relacional é descrito em tabelas, onde cada tabela contém linhas com atributos e estabelece relações por meio das suas chaves primárias e estrangeiras, que também garante a integridade dos dados. Elmasri & Navathe (2011, p.39) explicam que "O modelo relacional representa o banco de dados como uma coleção de relações. Informalmente, cada relação é semelhante a uma tabela de valores ou, até certo ponto, a um arquivo plano de registros."

Devido às suas características o processo de normalização é imprescindível para mitigar redundâncias e anomalias, para Heuser (2009, p.190) "Este processo baseia-se no conceito de forma normal. Uma forma normal é uma regra que deve ser obedecida por uma tabela para que esta seja considerada 'bem projetada'".

Outra importante característica presente no modelo relacional são as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo o processamento e consistência das informações. Conforme exposto por Ramakrishnan (2015, p. 435 e 436),

> 1. Os usuários devem ser capazes de enxergar a execução de cada transação como atômica: ou todas as ações são executadas ou nenhuma delas é executada. Os usuários não devem se preocupar com o efeito de transações incompletas (digamos, quando ocorre uma falha de sistema).
> 2. Cada transação, executada sozinha, sem nenhuma execução concorrente de outras transações, deve preservar a consistência do banco de dados. O SGBD presume que a consistência é preservada por transação. Garantir essa propriedade de uma transação é responsabilidade do usuário.
> 3. Os usuários devem ser capazes de entender uma transação sem considerar o efeito de outras transações em execução concorrente, mesmo que o SGBD intercale as ações de várias transações por motivos de desempenho. Essa propriedade é algumas vezes chamada de isolamento: as transações são isoladas (ou protegidas) dos efeitos do plano de execução concorrente de outras transações.
> 4. Uma vez que o SGBD informe ao usuário que uma transação foi concluída com êxito, seus efeitos devem persistir, mesmo que o sistema falhe antes que todas as suas alterações sejam refletidas no disco. Essa propriedade é chamada de durabilidade.

Seguindo os princípios mencionados, o MySQL é uma solução amplamente utilizada no mercado, sendo um dos SGBDs a serem utilizados neste trabalho. Sua popularidade se justifica por ser uma tecnologia open source, amplamente utilizada por empresas e desenvolvedores, além de proporcionar eficiência, flexibilidade e suporte à modelagem relacional. Apesar de suas vantagens, os modelos relacionais encontram desafios relacionados à escalabilidade, pois o aumento dos dados pode exigir investimento em hardware, para manter o desempenho, tendo em vista que em aplicações de grande escala a escalabilidade horizontal é a mais indicada (SOARES, 2023).

2.2.1 MySQL

O MySQL é um dos sistemas de gerenciamento de bancos de dados relacionais (SGBDs) mais utilizados no mundo, ocupando a segunda colocação dentre os bancos de dados mais populares, (Segundo o Raking da db-engines.com). Foi criado inicialmente pela empresa sueca MySQL AB em 1995, por Michael Widenius (Monty), David Axmark e Allan Larsson (RIEUF, 2016).

Sua popularidade deve-se não apenas ao fato de ser um software open source, mas também por fazer uso da linguagem SQL (Structured Query Language). Segundo Elmasri & Navathe (2011, p. 57) "A linguagem SQL pode ser considerada um dos principais motivos para o sucesso dos bancos de dados relacionais comerciais." Além disso, ele possui recursos importantes como: robustez de transações, que incluem o suporte a commit, rollback e recuperação de falhas, garantindo a integridade dos dados em situações adversas. (Magalhães & Portugal, 2024). O MySQL também conta com outros fatores que reforçam a popularidade, conforme exposto por Erickson (2024):

- Código aberto com suporte da comunidade: A comunidade global do MySQL fornece tutoriais, fóruns e recursos úteis. Além disso, contribui para a identificação e correção de bugs, tornando o banco confiável.
- Alto desempenho e confiabilidade: O MySQL se adapta a diversos cenários, suportando grandes volumes de dados e conexões simultâneas. Seus mecanismos de replicação e *failover*[1] garantem estabilidade e minimizam perdas.
- Facilidade de uso e compatibilidade: O MySQL é intuitivo e compatível com diversas linguagens, como Java, Python e PHP. Suporta replicação entre versões, facilitando a migração e manutenção de aplicações.

O MySQL utiliza um modelo cliente/servidor que permite várias conexões simultâneas ao servidor, possibilitando operações de leitura ou escritura nos bancos de dados de maneira eficiente. Ele conta também com diversos recursos que otimizam sua performance, como por exemplo, o recurso de replicação que beneficia escalabilidade e distribuição de carga entre servidores diferentes, além de diferentes métodos de sincronização garantindo maior segurança, consistência dos dados, proporcionando uma gestão mais eficiente dos recursos.

- Escalabilidade e Desempenho: A replicação no MySQL permite distribuir a carga entre múltiplas réplicas, melhorando o desempenho. Enquanto a origem se concentra em

[1] Processo em que um banco secundário assume a função primária quando este falhar.

gravações e atualizações, as leituras podem ser feitas nas réplicas, acelerando as consultas. Esse modelo reduz a sobrecarga no servidor principal e melhora a experiência do usuário. Além disso, aumenta a disponibilidade e a capacidade de resposta do sistema. O MySQL também conta com ferramentas de otimização de consultas, como o comando EXPLAIN, que permite visualizar o plano de execução de uma consulta podendo contribuir para o ajuste de índices e junções a fim de melhorar o desempenho (MYSQL, 2025).

- Métodos de Replicação: O MySQL suporta replicação baseada em log binário e em GTIDs (Global Transaction Identifiers). A replicação por log binário requer rastreamento manual de arquivos e posições, enquanto a GTID é transacional e mais simples de gerenciar. Também existem diferentes formatos: SBR (replica comandos SQL), RBR (replica apenas os dados alterados) e MBR (mistura os dois). A escolha do método influencia a performance e a consistência (MYSQL, 2025).
- Tipos de Sincronização e Segurança: A replicação pode ser assíncrona, semissíncrona, dependendo da necessidade de consistência. A semissíncrona garante que pelo menos uma réplica receba os dados antes da confirmação. A replicação atrasada evita a propagação imediata de erros, funcionando como um backup temporário. Essas abordagens melhoram a segurança e a resiliência do banco de dados (MYSQL, 2025).

A segurança é um fator primordial para qualquer SGBD, o MySQL implementa diversos mecanismos para garantir a proteção dos dados. Segundo a documentação oficial do MySQL (MySQL, 2025), entre as principais funcionalidades de segurança do MySQL estão:

- Gerenciamento de Acesso e Privilégios no MySQL: O SGBD emprega um sistema robusto de gerenciamento de privilégios para controle de acesso ao banco, permitindo que os administradores definam com quais operações cada usuário pode executar. Isso inclui a capacidade de conceder ou revogar privilégios específicos, como CREATE, DROP, INSERT, UPDATE, DELETE e SELECT, em diferentes níveis de acesso, garantindo um maior controle, reforçando a segurança do ambiente (MySQL, 2025).
- Criptografía de dados: O MySQL oferece criptografía para proteger dados, utilizando SSL/TLS para conexões seguras e funções de criptografia para proteger informações sensíveis (MySQL, 2025).
- Backup e recuperação: Ferramentas como mysqldump e MySQL Enterprise Backup permitem a realização de cópias de segurança e recuperação eficiente de dados em caso de falha, garantindo a continuidade das operações mesmo diante de falhas inesperadas (MySQL, 2025).

Embora os bancos de dados relacionais, como o MySQL, sejam amplamente utilizados devido à sua estrutura bem definida e ao suporte a transações ACID, certos cenários exigem maior flexibilidade, escalabilidade e desempenho em grandes volumes de dados. Com o crescimento exponencial das informações e a necessidade de processamento distribuído, surgiram os bancos de dados não relacionais (NoSQL), que oferecem abordagens alternativas para armazenar e gerenciar dados de forma mais eficiente em aplicações modernas.

2.3 Bancos de Dados Não Relacionais (NoSQL)

Em resposta ao grande volume de dados semiestruturados e não estruturados, aliado a necessidade de maior escalabilidade e desempenho, surge um novo modelo no contexto dos dados, conhecido por NoSQL (Not Only SQL – Não Apenas SQL). Com características distintas comparado ao tradicional relacional, trouxe a flexibilidade como um dos atributos principais, sem a necessidade de um esquema predefinido de dados, aliado a uma maior eficiência em escalabilidade horizontal. Segundo Lóscio et al. (2011, p. 2) o paradigma "[...] foi proposto com o objetivo de atender aos requisitos de gerenciamento de grandes volumes de dados, semi-estruturados ou não estruturados[, que necessitam de alta disponibilidade e escalabilidade.]"

O termo NoSQL foi utilizado por Carlo Strozzi em 1998 para descrever um banco de dados relacional de código aberto que não utilizava a linguagem SQL. No entanto, ele argumentava que o movimento moderno deveria ser chamado de "NoREL". Com o crescimento da internet e o aumento dos dados, surgiram desafios de escalabilidade e modelagem flexível. Em 2006, o artigo "BigTable: A Distributed Storage System for Structured Data", do Google, reacendeu a discussão sobre bancos distribuídos. Em 2009, Eric Evans reintroduziu o termo NoSQL. A partir daí, ele passou a ser associado a tecnologias como MongoDB, Cassandra e sistemas inspirados no Dynamo da Amazon, focados em processamento distribuído e alta disponibilidade (IANNI,2012).

Uma característica fundamental dos sistemas NoSQL é sua adoção do Teorema CAP[2], apresentado por Eric Brewer em 2000 e formalizado posteriormente por Gilbert e Lynch (2002). Esse teorema estabelece que, em um sistema distribuído, é impossível garantir simultaneamente

[2] O Teorema CAP estabelece que um sistema distribuído não pode garantir simultaneamente consistência, disponibilidade e tolerância à partição. Para uma explicação detalhada, ver: GILBERT, Seth; LYNCH, Nancy. *Towards robust distributed systems*. Disponível em: https://www.researchgate.net/publication/221343719_Towards_robust_distributed_systems. Acesso em: 21 mar. 2025.

Consistência (C), Disponibilidade (A) e Tolerância à Partição (P). Assim, os bancos NoSQL geralmente priorizam dois desses três aspectos, dependendo do tipo de aplicação para o qual são projetados. (LÓSCIO et.al, 2011).

Os sistemas NoSQL são classificados em quatro categorias principais, cada uma com características e aplicações específicas, conforme ponderam Garcia e Sotto (2019):

- Orientado a Documentos: Armazena dados na forma de documentos (geralmente em JSON, XML ou BSON). Cada documento é uma coleção autônoma de campos e valores, permitindo flexibilidade na modelagem, já que documentos de uma mesma coleção podem ter estruturas diferentes. Exemplo: MongoDB, CouchDB.
- Chave-Valor: Baseia-se em uma estrutura simples de pares chave-valor, semelhante a uma tabela hash, permitindo operações extremamente rápidas de leitura e escrita. Utilizado em aplicações que precisam armazenar e recuperar dados rapidamente, mas não exigem consultas complexas. Exemplo: Redis, DynamoDB, Riak. Essa abordagem é ideal para cache de dados, sessões de usuário e processamento em tempo real.
- Orientado a Colunas: Estrutura os dados em famílias de colunas, ao invés de linhas e tabelas tradicionais. Permite consultas altamente eficientes para grandes volumes de dados, sendo amplamente utilizado em análises de Big Data e data warehouses. Exemplo: Apache Cassandra, HBase.
- Orientado a Grafos: Modela os dados como nós e arestas, permitindo a representação explícita de relações entre entidades. Ideal para aplicações que envolvem redes sociais, recomendação de conteúdos e análise de conexões complexas. Exemplo: Neo4j, ArangoDB.

Os modelos NoSQL se destacam pela escalabilidade horizontal, permitindo que os dados sejam distribuídos em múltiplos nós, e pela flexibilidade, pois não necessitam de um esquema fixo. Isso facilita adaptações dinâmicas, conforme o volume e o formato dos dados evoluem. Porém, apesar das vantagens, existem desafios associados ao uso de bancos NoSQL, como a falta de suporte nativo a transações complexas em alguns modelos, a necessidade de consistência eventual e as dificuldades na migração de dados de sistemas relacionais (LÓSCIO, et.al, 2011).

Entre os diversos SGBDs NoSQL, o MongoDB se destaca como um dos mais populares e amplamente utilizados, sendo um dos SGBDs a serem utilizados neste trabalho. Sua estrutura orientada a documentos permite uma modelagem flexível, facilitando a manipulação de dados sem a necessidade de esquemas rígidos. Além disso, suas funcionalidades avançadas, como replicação, particionamento horizontal (*sharding*) e suporte a consultas complexas, tornam-no

uma escolha robusta para aplicações que exigem alta escalabilidade e desempenho. A seguir, será realizada uma análise sobre suas principais características e aplicabilidades (LÓSCIO, et.al, 2011).

2.3.1 MongoDB

O MongoDB é um dos principais sistemas de gerenciamento de banco de dados NoSQL, (Segundo o Ranking da db-engines.com) é primeira colocada entre os SGBD NoSQL, é amplamente adotado por empresas que necessitam de flexibilidade e escalabilidade no armazenamento de grandes volumes de dados. Criado em 2007 pela empresa 10gen, posteriormente renomeada como MongoDB Inc., o sistema foi projetado para atender demandas de aplicações modernas que exigem alta disponibilidade, desempenho otimizado e processamento eficiente de dados semiestruturados (MongoDB, s.d.).

Por não seguir um padrão presentes nos paradigmas relacionais, que utilizam tabelas para organizar informações, o MongoDB adota um modelo orientado a documentos, onde os dados são armazenados no formato BSON (Binary JSON). Essa abordagem permite que os documentos tenham estruturas flexíveis, eliminando a necessidade de esquemas rígidos e facilitando adaptações conforme os dados evoluem (MongoDB, s.d.).

Além disso, o MongoDB segue os princípios do Teorema CAP. Embora possa ser configurado para priorizar diferentes combinações desses fatores, por padrão, o MongoDB enfatiza Consistência e Tolerância à Partição (CP), podendo ser ajustado conforme as necessidades específicas de cada aplicação (IBM, s.d.).

No MongoDB, os dados são organizados em coleções, que funcionam como agrupamentos lógicos de documentos. Cada documento pode conter uma estrutura flexível e sem esquema rígido, permitindo persistir diferentes dados com diferentes atributos, dentro da mesma coleção. Essa abordagem torna o MongoDB uma solução ideal para aplicações que precisam lidar com dados dinâmicos e em constante mudanças. Diferente dos bancos relacionais, não existe necessidade de normalização e junções complexas, o que melhora consideravelmente o desempenho em aplicações distribuídas (MongoDB, s.d.).

Um dos principais diferenciais do MongoDB é sua escalabilidade horizontal, possibilitada pelo mecanismo de fragmentação, que distribui os dados automaticamente entre múltiplos servidores. Esse método é essencial para aplicações de grande porte, pois melhora a capacidade de processamento, apenas incluindo mais servidores no intuito de dividir a carga de trabalho de um único servidor.

O MongoDB implementa um sistema robusto de replicação através de replica sets, que são grupos de instâncias mongod que mantêm o mesmo conjunto de dados. Cada réplica set contém um nó primário que recebe todas as operações de escrita e um ou mais nós secundários, que replicam o log de operações (oplog) do primário e aplicam essas operações aos seus próprios conjuntos de dados, mantendo uma cópia idêntica dos dados do primário. Se o nó primário se tornar indisponível, os membros do replica set realizam uma eleição para selecionar um novo primário, garantindo a continuidade das operações de escrita. Esse modelo de replicação assegura alta disponibilidade e tolerância a falhas, garantindo que os dados permaneçam acessíveis mesmo em caso de falha de um servidor (MongoDB, s.d.).

Na parte de segurança o MongoDB se destaca com os seguintes recursos:

- Autenticação baseada em usuários e permissões: O MongoDB permite a criação de usuários com diferentes níveis de acesso, garantindo que apenas indivíduos autorizados possam realizar operações específicas no banco de dados (MongoDB, 2025).
- Controle de acesso baseado em funções (RBAC - Role-Based Access Control): Esse mecanismo possibilita a atribuição de permissões específicas a grupos de usuários, simplificando a gestão de acesso e aumentando a segurança do sistema (MongoDB, 2025).
- Criptografia de dados em repouso e em trânsito: O sistema suporta a criptografia de dados tanto quando armazenados quanto durante a transmissão, assegurando a confidencialidade e integridade das informações (MongoDB, 2025).

De acordo com a documentação oficial do MongoDB, a implementação de mecanismos de segurança é fundamental para proteger informações sensíveis e garantir conformidade com as regulamentações de proteção de dados. No Brasil, é importante considerar a LGPD (Lei Geral de Proteção de Dados) ao implementar esses mecanismos. Além disso, é crucial considerar que os bancos de dados NoSQL como o MongoDB, enfrentam desafios específicos relacionados à segurança devido ao seu modelo flexível e distribuído, tornando essencial a adoção de estratégias robustas de autenticação, controle de acesso e criptografia para garantir a proteção dos dados.

2.4 Comparação entre Modelos Relacional e NoSQL

Embora ambos os modelos de bancos de dados realizem a função de persistência de dados, cada abordagem possui características que as tornam mais adequadas para diferentes cenários.

Os bancos de dados relacionais oferecem alta integridade, suporte a transações complexas e forte consistência, fundamentados nas propriedades ACID. Porém, podem enfrentar dificuldades de escalabilidade e desempenho quando submetidos a grandes volumes de dados, devido a necessidade de múltiplas junções e complexidade de manipular os dados.

Em contrapartida, os sistemas NoSQL foram concebidos para oferecer alta escalabilidade e flexibilidade, permitindo a distribuição dos dados em ambientes de alta carga. Esse modelo facilita a manipulação de dados não estruturados e amplamente adotado em cenários que exigem desempenho elevado e disponibilidade.

Em ambientes de e-commerce, onde o tempo de resposta e a capacidade de processar um grande número de consultas são determinantes para a conversão, a escolha do Sistema de Gerenciamento de Banco de Dados (SGBD) é fundamental:

- O MySQL, representando o modelo relacional, garante robustez, integridade e confiabilidade dos dados, sendo ideal para transações financeiras e aplicações empresariais (Erickson, 2024).
- O MongoDB, representante dos sistemas NoSQL orientados a documentos, se destaca pela flexibilidade e escalabilidade horizontal, permitindo respostas mais rápidas em consultas simples e alto desempenho em cargas distribuídas (MONGODB, 2025).

Além do desempenho e da adaptabilidade do SGBD a ser escolhido, deve-se também levar em conta aspectos como os custos operacionais e a dificuldade na manutenção. Os bancos relacionais requerem um cuidadoso planejamento prévio e uma administração dos índices e das junções eficientes, juntamente com um esquema rígido que pode aumentar a complexidade da manutenção necessária. Os bancos NoSQL oferecem maior flexibilidade e podem exigir técnicas avançadas para particionamento de dados e garantir consistência a longo prazo.

Os bancos de dados relacionais organizam as informações em tabelas usando chaves primárias e estrangeiras para garantir consistência nos dados, no entanto precisam realizar operações como junções e agregações para acessar as informações desejadas. Embora confiáveis e seguros por natureza, esse modelo pode demandar um alto custos computacional em consultas mais complexas, especialmente ao lidar com pesquisas textuais que exigem combinação entre várias tabelas, impactando no tempo necessário concluir uma busca por produto, além de consumir demasiadamente recursos como memória RAM ou CPU.

Por outro lado, os sistemas de banco de dados NoSQL que seguem o modelo orientado a documentos armazenam os dados em BSON, eliminando assim a necessidade de operações complexas de junção e possibilitando uma recuperação rápida das informações. Essas práticas

resultam em tempos de resposta mais ágil e maior capacidade de escalonamento horizontal, embora possam exigir um maior uso da CPU, devido a execução dinâmica de agregações. Estas características são fundamentais para análises comparativas neste contexto, uma vez que permitem avaliar qual modelo proporcionará o melhor desempenho na busca por um produto específico.

O estudo comparativo realizado neste trabalho considera aspectos essenciais de acesso ao banco de dados que influenciam diretamente o desempenho nas operações de busca em um ambiente de comércio eletrônico. Considerando as diferenças entre os bancos de dados relacionais e o NoSQL juntamente com suas vantagens e desvantagens quando aplicados ao modelo de *e-commerce* é crucial estabelecer uma maneira sistemática de avaliar como eles se saem em termos de desempenho e adequação nesse cenário específico. Assim sendo, no próximo capítulo, será apresentada a metodologia utilizada para a execução deste trabalho, detalhando os critérios de análise, as ferramentas empregadas e o ambiente de testes configurado para comparar as soluções MySQL e MongoDB.

2.5 Sistema de e-commerce

Comércio eletrônico, também conhecido como *e-commerce*, refere-se a um modelo de negócio em que produtos e serviços são comprados e vendidos online em vez de em lojas físicas tradicionais (SAMPAIO, 2024). Este setor tem visto um aumento significativo em popularidade, graças a sua conveniência para os consumidores que podem encontrar uma grande variedade de produtos sem precisarem sair de casa para realizarem compras. Diversas empresas de diferentes áreas têm adotado o comércio eletrônico como forma de alcançarem mais clientes, reduzirem custos operacionais e oferecerem uma experiência personalizada e eficiente aos consumidores durante o processo de compra.

Principais características do e-commerce:

- Variedade de Produtos e Segmentação – Diferente do varejo físico, o e-commerce permite disponibilizar uma grande variedade de produtos em um único ambiente virtual, abrangendo diferentes nichos e facilitando a busca por itens específicos.
- Acessibilidade e Disponibilidade – As lojas online funcionam 24 horas por dia, permitindo compras a qualquer momento e em qualquer lugar, o que amplia o alcance do negócio e facilita a experiência do consumidor.

- Personalização e Experiência do Usuário – Utilizando algoritmos e inteligência artificial, o e-commerce pode recomendar produtos com base no comportamento do usuário, criando uma experiência de compra mais eficiente e direcionada.
- Segurança e Proteção de Dados – Como envolve transações financeiras e informações sensíveis, o e-commerce adota protocolos de segurança avançados, como criptografia de dados e até mesmo reconhecimento facial.
- Logística e Eficiência na Entrega – Para garantir a satisfação do cliente, o e-commerce investe em estratégias logísticas otimizadas, como múltiplas opções de envio, rastreamento de pedidos e estoques.

3 DESENVOLVIMENTO

Esta seção aborda a análise comparativa entre os dois modelos de banco de dados, com o objetivo de determinar qual é mais indicado para um cenário hipotético de implementação de um sistema de e-commerce.

3.1 Considerações iniciais

Serão avaliadas questões de desempenho para cada SBGD, incluindo a quantidade de operações realizadas em um período determinado, o tempo médio de resposta, o uso de CPU, o uso de memória e taxa de I/O de disco. Essa análise permitirá compreender o desempenho e a eficiência de cada modelo em um ambiente de alto volume de requisições, através do uso de múltiplas *threads* concorrentes durante os testes, gerando diversas consultas simultâneas ao banco de dados, como é comum em plataformas de comércio eletrônico, contribuindo para a escolha da solução mais adequada conforme as necessidades do sistema.

No parágrafo seguinte, informa-se que a análise de desempenho focou na operação de apresentação de dados a partir de uma consulta por produto, representando uma situação prática comum em plataformas de *e-commerce*. A justificativa baseia-se no volume de dados retornados em uma pesquisa por um item em determinados sites de e-commerce testados e a necessidade de um curto tempo de resposta, garantindo uma experiência mais fluida ao usuário.

Foram realizados testes em dois sites de e-commerce distintos para medir o tempo necessário para a exibição dos resultados de uma pesquisa pelo termo "tv". As medições foram feitas utilizando a opção inspecionar. A opção inspecionar refere-se à ferramenta de inspeção de elementos dos navegadores, utilizada para monitorar o tempo de carregamento das requisições. As métricas de tempo de resposta foram obtidas observando a aba *Network* da ferramenta. Analisando os tempos das principais requisições responsáveis pela montagem da página.

- Carrefour:

  Principais tempos registrados:

  -busca.data?term=tv: 259 ms

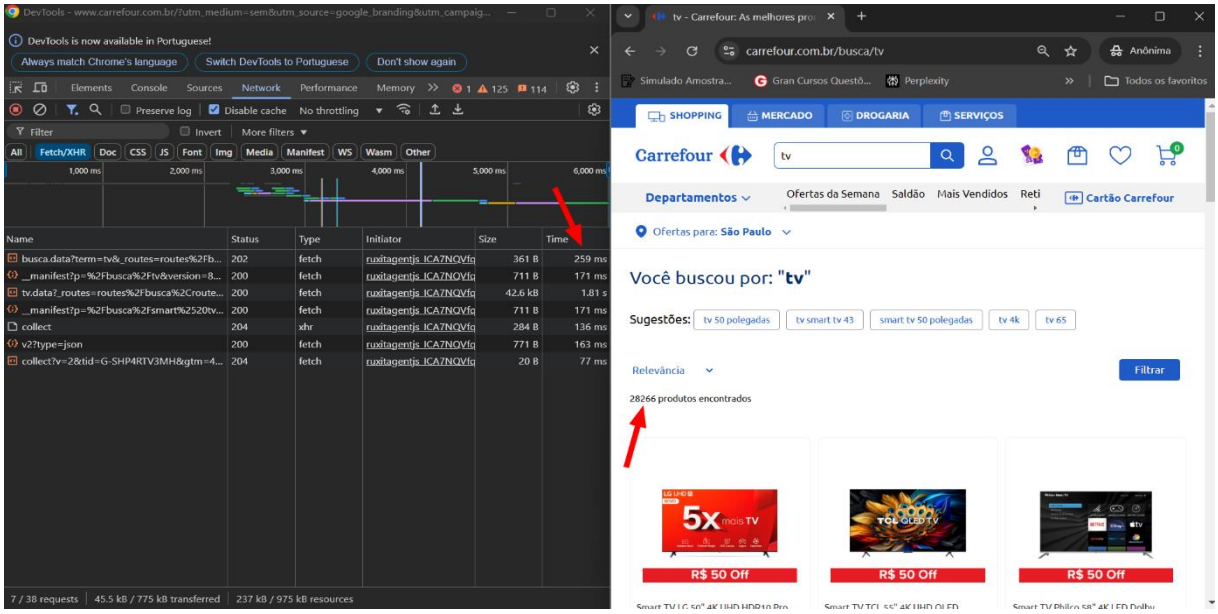
  -_manifest: 171 ms

  -tv.data: 1.81 s

  -Média de outras requisições (~143 ms)

  -Tempo total estimado: 2.7 segundos para 28.266 itens.

Figura 1 - Tempo de resposta da busca por produto 01



Fonte: Elaborado pelo autor (2025).

- Magazine Luiza

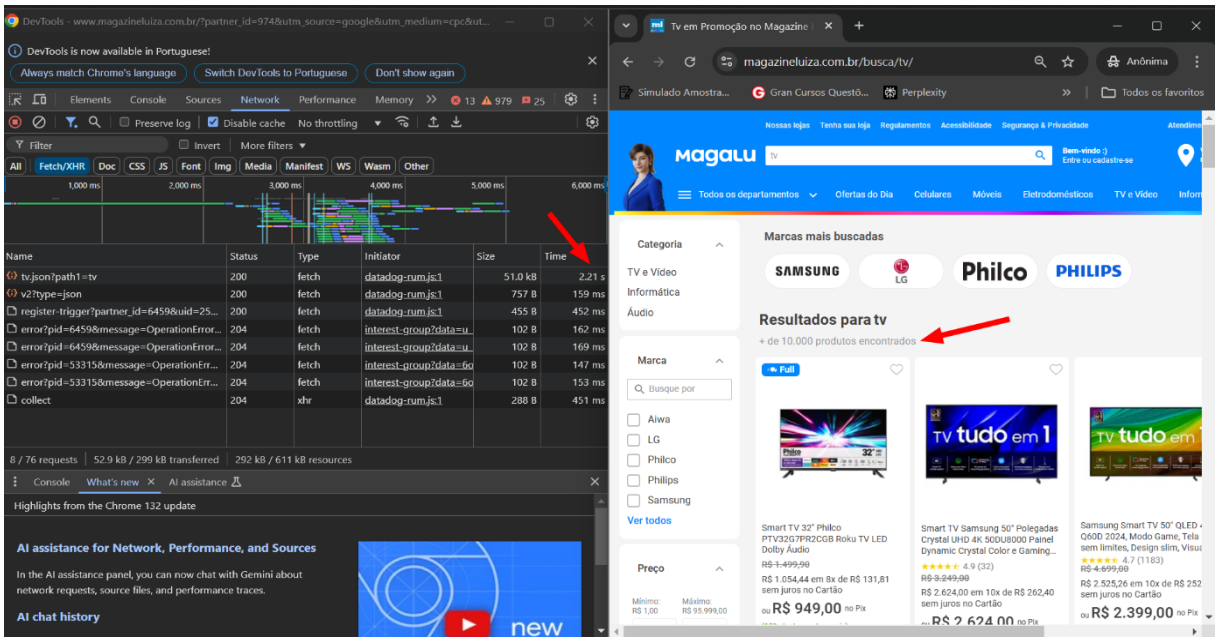
  Principais tempos registrados:

  -tv.json?path1=tv: 2.21 s

  -Média de outras requisições (~241 ms)

  -Tempo total estimado: 3.9 segundos para mais de 10.000 produtos.

Figura 2 - Tempo de resposta da busca por produto 02



Fonte: Elaborado pelo autor (2025).

3.2 Descrição do Ambiente Experimental

A máquina utilizada nos testes foi um notebook Dell Latitude 3540, equipado com processador Intel Core i7-1355U de 10 núcleos e 12 threads, 16 GB de memória RAM, SSD NVMe SK Hynix BC711 de 512 GB e com sistema operacional Windows 11 Pro (64 bits).

Para assegurar um ambiente em idênticas condições de teste para ambos os SGBDs, inicialmente foi criada uma máquina virtual (VM) base utilizando o sistema operacional Ubuntu 24.04.01 LTS, escolha que se deve à sua compatibilidade com ambos SGBDs, conforme descrito em sua documentação (MySQL, 2025) e (MongoDB, 2025).

Além disso, optou-se pelo uso do hipervisor VMware Workstation 17 PRO para a criação e gerenciamento das VMs, o motivo dessa escolha será detalhado na etapa de instalação do MongoDB, onde se evidenciaram incompatibilidades com outros hipervisores testados.

A VM base foi configurada com as seguintes especificações:

- Memória RAM: 4 GB – Esta configuração foi definida considerando os requisitos mínimos recomendados, para executar ambos SGBDs, conforme documentação.
- Processador: 4 processadores, cada um com 3 cores – Essa configuração multi-core possibilita a simulação de cargas de trabalho distribuídas, permitindo que os testes de desempenho possam avaliar a resposta dos SGBDs sob condições de execução concorrente e uso intensivo de CPU, sendo superior aos requisitos mínimos recomendados.

Após finalizar o processo de instalação e configuração do sistema operacional na máquina virtual base (VM), realizamos duas clonagens dela para garantir que cada cópia apresentasse um ambiente exatamente igual para instalar os Sistemas de Gerenciamento de Bancos de Dados (SGBDs). Esta prática garantiu que só um SGBD por vez estivesse instalado no sistema, eliminando qualquer possibilidade de interferência entre eles. Cada clone recebeu a instalação isolada de um SGBD, conforme descrito:

- Clone 1 – VM para MySQL:

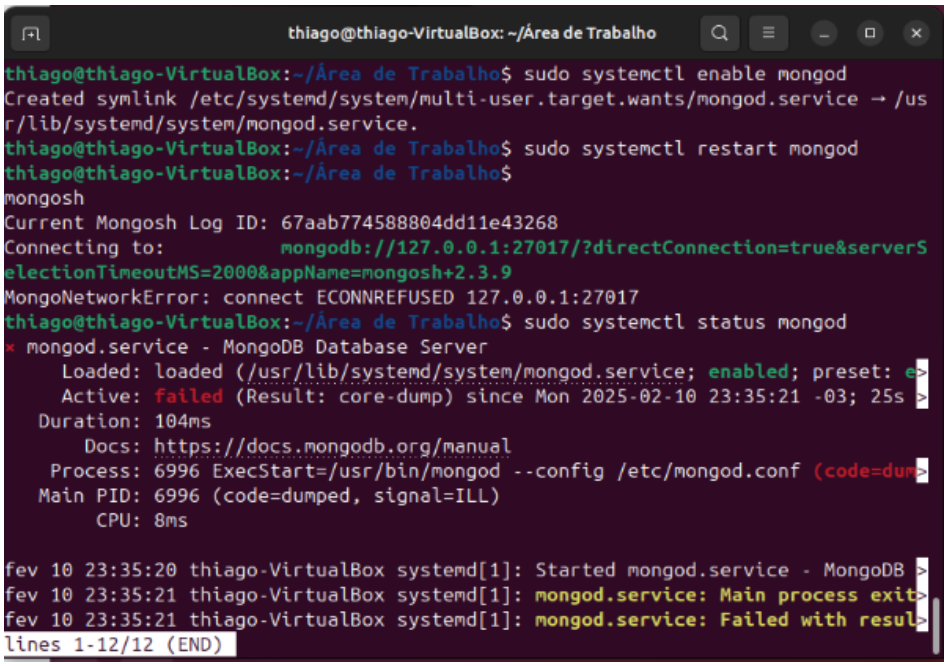
  Nesta VM, o MySQL foi instalado utilizando as configurações padrão. A escolha do MySQL se deve à sua ampla adoção, robustez e facilidade de uso, características que o tornam uma referência em sistemas relacionais.

- Clone 2 – VM para MongoDB:

  Na máquina virtual em questão foi feita a instalação do MongoDB com as configurações padrão, durante o processo de instalação inicial do serviço MongoDB encontramos algumas dificuldades específicas em relação à compatibilidade, ao tentar iniciar o

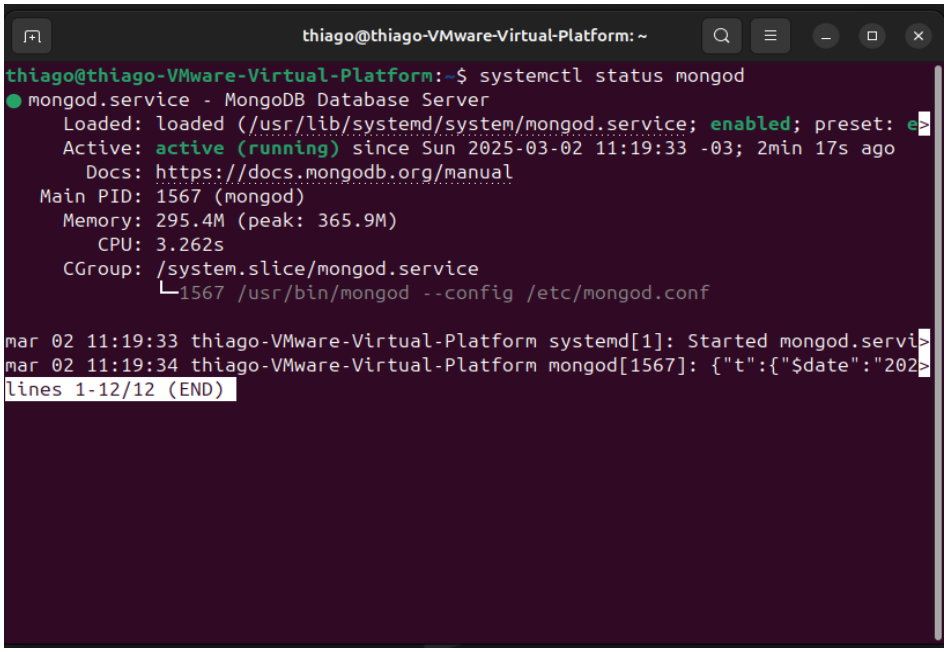
serviço nos hipervisores com o VirtualBox e Hyper-V, nessas plataformas o serviço não inicializava adequadamente resultando no erro “Illegal instruction (core dumped)”. Apenas com o uso do VMware Workstation 17 PRO que se mostrou adequado às diretrizes necessárias pelo MongoDB e MySQL, foi viável iniciar o serviço sem enfrentar nenhum contratempo específico. Esta constatação ressalta a relevância da seleção cuidadosamente do hipervisor correto para assegurar a estabilidade e confiança do ambiente experimental.

Figura 3 - Erro ao Iniciar Serviço do MongoDB no VirtualBOX



Fonte: Elaborado pelo autor (2025).

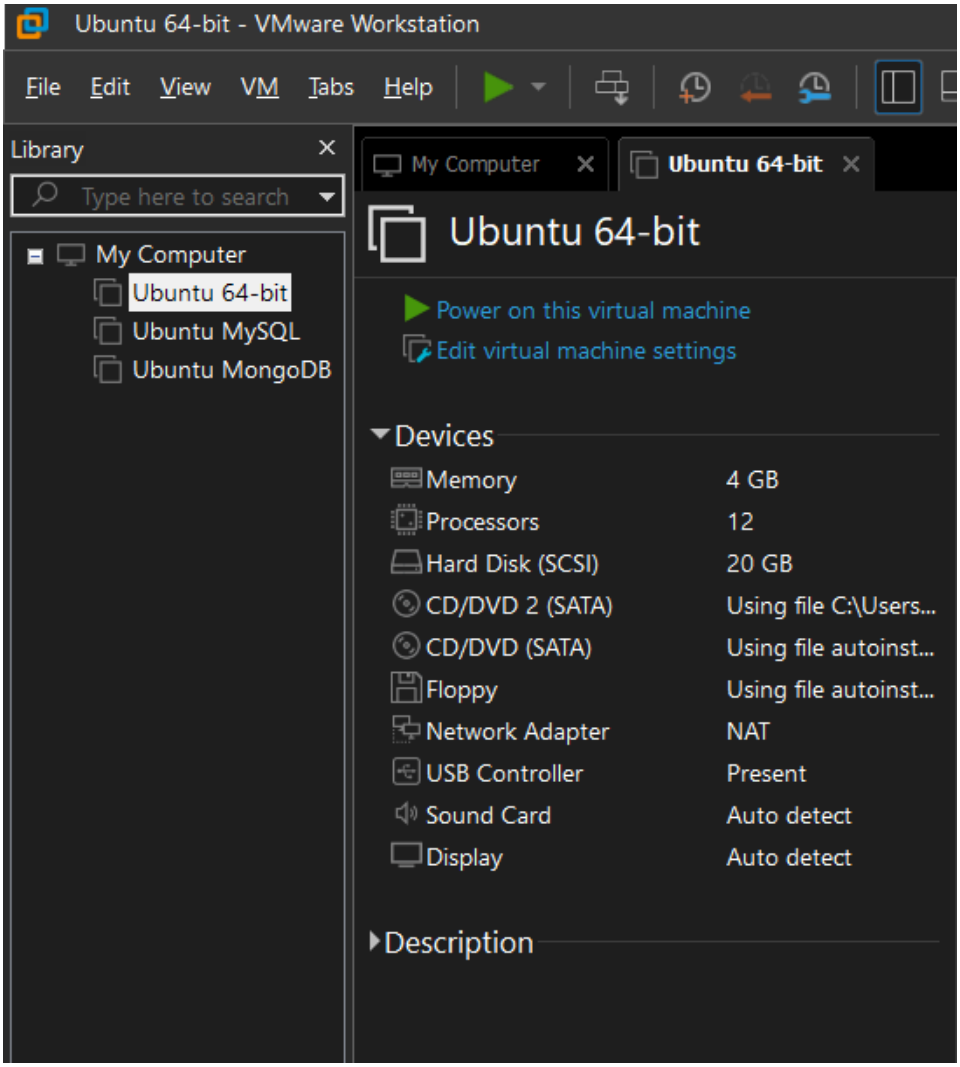
Figura 4 - Serviço do MongoDB no Hipervisor VMware



Fonte: Elaborado pelo autor (2025).

Em resumo, a criação de um ambiente experimental padronizado e reprodutível, com a clonagem da imagem base e a escolha criteriosa do hipervisor, possibilitou que os testes de desempenho fossem realizados de maneira imparcial e precisa. Essa abordagem meticulosa assegura que as comparações entre o MySQL e o MongoDB se baseiem em condições idênticas, contribuindo para a confiabilidade dos resultados obtidos.

Figura 5 - Serviço do MongoDB no Hipervisor VMware



Fonte: Elaborado pelo autor (2025).

Diante das incompatibilidades encontradas durante a instalação do MongoDB em hipervisores como o VirtualBox e o Hyper-V, foi necessário refazer o ambiente experimental. Optou-se, então, pela utilização do VMware Workstation 17 PRO, por apresentar compatibilidade com ambos os SGBDs. Para garantir igualdade nas condições de teste, os procedimentos de clonagem foram novamente realizados a partir da VM base, assegurando que tanto o MySQL quanto o MongoDB fossem executados em ambientes idênticos e devidamente padronizados.

3.3 Instalação e Configuração dos SGBDs

Em cada uma das VMs clonadas, foi realizada a instalação dos Sistemas de Gerenciamento de Bancos de Dados (SGBDs) utilizando as configurações padrão. A opção pela configuração padrão de cada um dos SGBDs se deve a necessidade de manter a isonomia na comparação, não exercendo nenhuma ação adicional no ambiente que pudesse favorecer um deles nos testes a serem executados. A seguir, descrevemos detalhadamente os procedimentos adotados para cada SGBD.

3.3.1 Instalação do MySQL

Procedimento de Instalação:

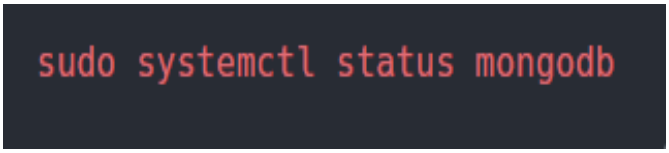
- Download e Instalação:
- O MySQL foi baixado a partir do repositório oficial, com a versão 8.0.1.41
- Configuração Inicial: Após a instalação, o processo de configuração inicial foi executado. Foram definidas as configurações iniciais, como a senha do usuário root e as opções de segurança recomendadas, removidas contas anônimas, além de usuários remotos e ajustadas as configurações de acesso, garantindo que o ambiente operasse de forma segura e confiável.
- Configuração Default: O MySQL foi mantido com suas configurações padrão (default), na configuração padrão o mecanismo de armazenamento utilizado é o InnoDB, que já vem ativado automaticamente. O servidor aceita múltiplas conexões ao mesmo tempo e permite que mais de um núcleo da CPU seja usado para melhorar o desempenho. A quantidade padrão de conexões simultâneas é de 151. A criação de clusters e o particionamento de tabelas são recursos suportados, mas não são ativados por padrão. O consumo de memória e o uso de cache são ajustados de forma automática, de acordo com a capacidade da máquina. Além disso, o MySQL registra logs de erros e fornece opção de ativar outros tipos de logs.
- Verificação do Funcionamento: Após a configuração, foram executados comandos básicos para validar o funcionamento do MySQL, como a criação de um banco de base de dados teste, com registro aleatórios, para a execução de consultas simples. Esses testes iniciais confirmaram que o SGBD estava operando de forma adequada para a população do modelo fictício de e-commerce.

3.3.2 Instalação do MongoDB

Procedimento de Instalação:

- Download e Instalação: O MongoDB foi obtido a partir do repositório oficial na versão 8.0.4 e instalado na segunda VM clonada. Devido a necessidade de compatibilidade específica, a instalação foi realizada no ambiente VMware Workstation 17 PRO, uma vez que tentativas anteriores em hipervisores como VirtualBox e Hyper-V resultaram em erros ao tentar iniciar o serviço dos bancos, mencionando anteriormente.
- Configuração padrão e Ajustes Específicos: O MongoDB foi instalado utilizando as configurações padrão, sem modificações manuais nos arquivos de configuração. Esse procedimento garantiu que o comportamento do sistema refletisse a configuração default, permitindo uma comparação direta com o MySQL. Foi verificado que o serviço iniciava corretamente, sem a ocorrência do erro anteriormente identificado em outros hipervisores.
- Validação do Serviço: Após a instalação, o serviço do MongoDB foi iniciado e sua plena operação validada por meio do comando:

Figura 6 - Comando para verificar status do serviço do MongoDB



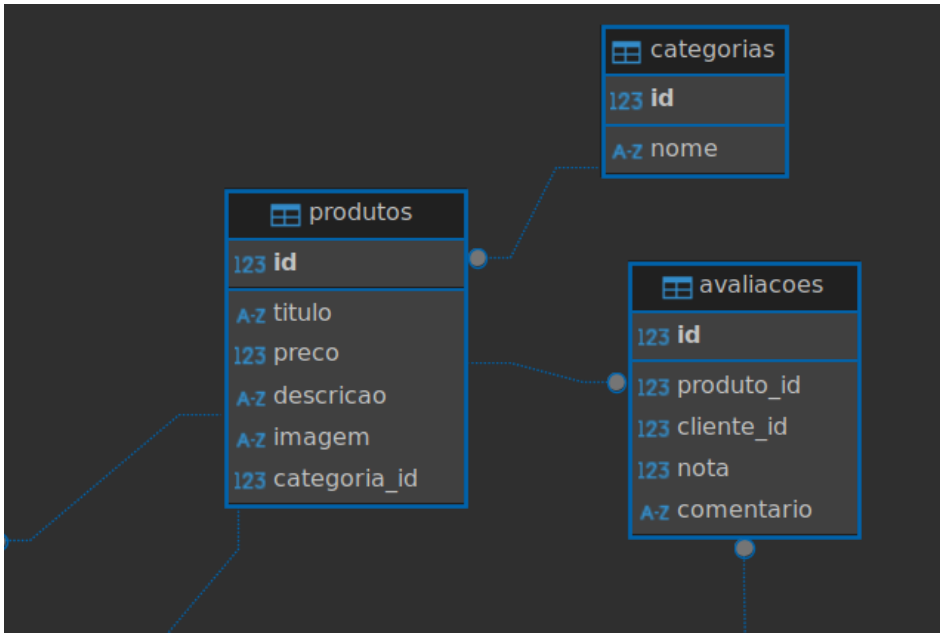
Fonte: Elaborado pelo autor (2025).

Essas verificações confirmaram que o MongoDB estava operando conforme esperado no ambiente VMware Workstation 17 PRO.

3.4 População da Base de Dados

Após a instalação e configuração dos SGBDs nas VMs, o próximo passo consistiu em popular as bases de dados com um modelo fictício de e-commerce.

Figura 7 - Tabelas Utilizadas na Consulta Teste



Fonte: Elaborado pelo autor (2025).

Este modelo foi criado para representar um cenário comum de e-commerce envolvendo diferentes elementos (como clientes, pedidos e avaliações), mostrando conexões autênticas entre eles. No entanto, a principal consulta está relacionada à tabela de produtos junto com suas avaliações, retornando os itens com “TV”, no título juntamente com a média de suas classificações. Assim sendo, as outras tabelas são usadas apenas para ilustrar a estrutura de um ambiente completo, embora não seja o foco da análise.

3.4.1 População no MySQL

Para o MySQL foram criados dois scripts em Python responsáveis por coletar dados de forma organizada para inserção no banco de dados. Esses scripts fizeram uso de diferentes bibliotecas importantes para desempenhar as seguintes tarefas:

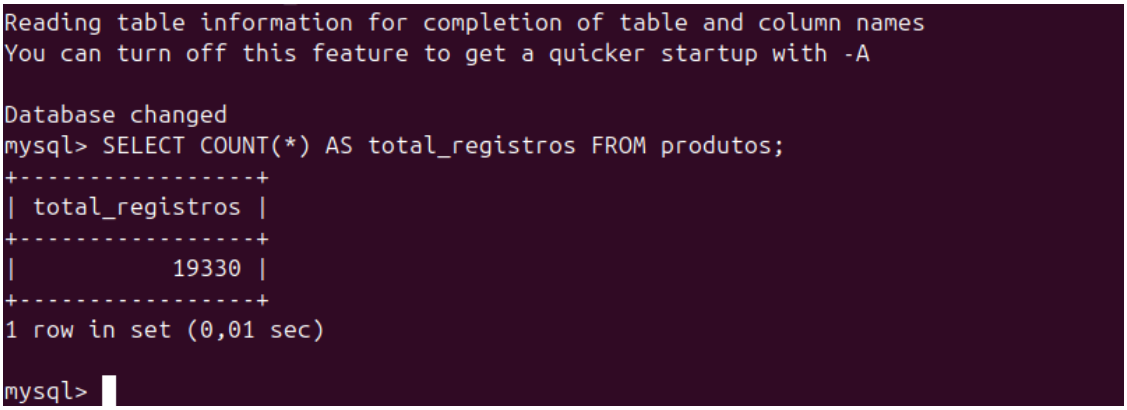
- mysql.connector: Para conectar ao servidor MySQL e executar comandos SQL é utilizada uma biblioteca específica que possibilitará a interação direta com o banco de dados para efetuar operações como inserção de dados novos ou atualização de registros existentes.
- requests: Realiza requisições HTTP para acessar APIs públicas e obter informações atualizadas sobre produtos, essenciais para alimentar o catálogo da loja virtual com dados relevantes.

- random e time: A biblioteca random é utilizada para gerar valores aleatórios, como a quantidade de produtos disponíveis, simulando variações reais do mercado. Já a biblioteca time controla o intervalo entre as requisições à API, evitando bloqueios ou sobrecarga no servidor da API, garantindo assim uma coleta contínua e estável de dados.
- faker: Utilizada para gerar dados fictícios de clientes, endereços e outros registros necessários para completar o cenário de e-commerce. Essa biblioteca cria informações plausíveis, como nomes, endereços e e-mails, proporcionando uma população realista da base de dados.

Procedimento de População:

- Coleta de Dados via API Públicas: O primeiro script consulta a API públicas para coletar informações de produtos em diversas categorias (por exemplo: geladeira, TV, notebook, etc.). Para cada categoria, o script realiza requisições em intervalos controlados, buscando até 1000 produtos (em lotes de 50) e armazenando os seguintes dados: título, preço, descrição, imagem (thumbnail), e uma quantidade aleatória para simular o estoque.
- Inserção dos Dados nas Tabelas Relacionais: Os dados coletados são então processados e inseridos na base de dados MySQL. O script realiza a inserção nas tabelas de produtos e, simultaneamente, insere informações relacionadas, como os registros de estoque. O uso do comando INSERT IGNORE nas tabelas de categorias garante que não ocorram duplicidades, e as chaves estrangeiras associam os produtos às suas respectivas categorias, preservando a integridade referencial.
- Validação da População: Após a inserção, foram realizadas consultas de verificação para confirmar que os dados foram corretamente inseridos. Esses testes preliminares asseguraram que a base do e-commerce estava devidamente populada com mais de 19 mil registros de produtos, simulando um ambiente robusto para os testes de desempenho subsequentes.

Figura 8 - Quantidade de Registro de Produtos MySQL



Fonte: Elaborado pelo autor (2025).

3.4.2 Exportação e Importação para o MongoDB

- Para permitir uma comparação direta entre os dois sistemas de gerenciamento de banco de dados SQL e NoSQL era necessário assegurar que ambos os bancos de dados contivessem os mesmos registros. Foi então realizado o processo de popular o MySQL com os dados, antes de serem exportados para arquivos CSV, para facilitar a padronização e transferência precisa dos registros para o ambiente NoSQL.
- Procedimento de Exportação e Importação:
  -Exportação dos Dados do MySQL
  -Por meio das ferramentas padrão do MySQL foi feita a extração dos dados das tabelas e sua gravação em arquivos CSV, como parte do procedimento utilizado o comando:

Figura 9 - Comando para Exportar Dados do MySQL

```
SELECT * FROM produtos
INTO OUTFILE '/caminho/para/saida.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Fonte: Elaborado pelo autor (2025).

Este comando possibilita exportar o conteúdo de uma tabela diretamente para um arquivo CSV, com a definição dos delimitadores e do caminho de destino, mantendo assim todos os registros com a estrutura e formatação idênticas às existentes na base relacional.

- Importação dos Dados no MongoDB: Com os dados exportados, o próximo passo foi importá-los para o MongoDB, assegurando que ambos os SGBDs contivessem informações idênticas. Para essa finalidade, utilizou-se o comando mongoimport conforme exemplo descrito abaixo:

Figura 10 - Comando para Importar Dados MongoDB

```
mongoimport --uri "mongodb://localhost:27017" --db ecommerce --
collection produtos --type csv --file produtos.csv --headerline
```

Fonte: Elaborado pelo autor (2025).

Esse comando realiza a importação do arquivo CSV para a coleção "produtos" no banco de dados "ecommerce", utilizando o parâmetro --headerline para mapear corretamente os nomes dos campos.

Figura 11 - Quantidade de Registros de Produtos MongoDB

```
> show collections
< avaliacoes
  categorias
  clientes
  itens_pedido
  pedidos
  produtos
> db.produtos.countDocuments();
< 19330
```

Fonte: Elaborado pelo autor (2025).

Ao término desse procedimento específico de povoamento de ambas as bases de dados MySQL e MongoDB com registros idênticos, foi viabilizada a comparação igualitária entre os dois sistemas de gerenciamento de banco de dados (SGBDs), utilizando os mesmos conjuntos de dados originais como referência direta.

Importante destacar que para manter uma base de comparação justa entre os bancos de dados, a modelagem utilizada no MongoDB manteve a estrutura relacional, mantendo as coleções separadas como "produtos" e "avaliações", em vez de incorporar os dados de

avaliações dentro dos documentos de produtos, conhecida como estrutura aninhada, como seria comum em um modelo orientado a documentos. Essa decisão foi tomada com o objetivo de garantir equidade na comparação, mesmo que isso represente uma modelagem menos otimizada para o MongoDB.

Apesar disso, reconhece-se que essa escolha não representa o cenário ideal de uso do MongoDB, podendo impactar negativamente em seu desempenho, já que não aproveita uma das suas principais vantagens do SGBD. Esse aspecto é considerado uma limitação metodológica da pesquisa, mas foi necessário para garantir um ponto de partida igualitário entre as tecnologias.

3.5 Execução dos Testes de Desempenho

Para avaliar o desempenho de cada sistema de gerenciamento de banco de dados em um cenário hipotético de e-commerce foram criados conjuntos de scripts em Python, que realizam consultas simuladas e coletam diversas métricas de desempenho. Essa etapa é essencial para comparar de forma objetiva o tempo de resposta esperado pela capacidade computacional e a eficiência ao lidar com um grande volume de consultas, aspectos cruciais em ambientes onde a rapidez no acesso aos dados pode impactar diretamente na experiência do usuário e consequentemente nas conversões das vendas.

3.5.1. Objetivos dos Testes

Os testes de desempenho foram definidos para avaliar as seguintes métricas:

- Tempo de Resposta: Medido em milésimos de segundo (ms), esse valor indica o tempo decorrido desde o envio da consulta até a obtenção dos resultados correspondentes. Medir essa métrica envolve a utilização de funções de medição temporal como time.perf_counter(), as quais proporcionam uma precisão elevada na avaliação do tempo necessário para a execução das pesquisas.
- Uso de CPU e Memória: Durante os testes, o uso de CPU e memória foi monitorado por meio da biblioteca psutil em Python. Os scripts acompanharam o consumo dos processos correspondentes a cada SGBD (mysqld para o MySQL e mongod para o MongoDB), coletando dados de uso de CPU (%), memória (MB) e operações de leitura/escrita em disco (convertidas de bytes para megabytes). Foram utilizadas 10 threads simultâneas para simular múltiplas conexões concorrentes, o que pode impactar

o desempenho de forma diferente entre os bancos. No MySQL, cada conexão cria uma nova thread, o que pode aumentar o consumo de recursos em ambientes de alta concorrência. Já o MongoDB utiliza um único processo gerenciando múltiplas conexões de forma mais centralizada, o que pode resultar em comportamento mais eficiente nesse cenário.

- Número Máximo de Consultas Concluídas: Esses dados mostram quantas solicitações o sistema pode lidar em um determinado período de tempo (por exemplo, 60 segundos), demonstrando o desempenho do banco de dados em situações de trabalho intenso.

3.5.2 Bibliotecas e Ferramentas Utilizadas

Os scripts de teste foram desenvolvidos utilizando diversas bibliotecas Python, cada uma com uma função específica:

- mysql.connector: Encarregado de estabelecer a ligação com o MySQL, para executar as consultas SQL. Isso possibilita o envio das consultas para obter os resultados desejados.
- pymongo: Usada para se conectar e integrar com o MongoDB de forma a facilitar a execução de pipelines de agregação, que replicam a consulta correspondente à procura por “TV".
- csv: Permite a escrita e leitura dos resultados obtidos em arquivos CSV. "Arquivos CSV (Comma-Separated Values, valores separados por vírgula), são utilizados devido a sua estrutura simples, onde os dados são organizados em um formato tabular e separados por vírgulas." (Shaibu, 2024). Essa funcionalidade é utilizada para registrar as métricas de cada execução de consulta, garantindo que os dados possam ser analisados posteriormente.
- *threading*: Foi utilizado o módulo de *threading* para replicar uma situação de carga simultânea em que várias threads executam as consultas ao mesmo tempo, para simular como uma aplicação com diversos usuários acessando ao mesmo tempo o sistema se comportaria na prática.
- time e psutil: A biblioteca time é usada para medir com precisão o tempo de execução das consultas. Já a psutil fornece informações detalhadas sobre o consumo de CPU, memória e I/O de disco, métricas essenciais para a avaliação do desempenho.
- Pandas: Em outro script subsequente foi empregada a biblioteca pandas para analisar os arquivos CSV produzidos e obter as médias das métricas obtidas para viabilizar uma avaliação minuciosa do desempenho dos SGBDs. Os dados foram apresentados em

forma tabular e os gráficos comparativos demonstrando as disparidades de desempenho foram então gerados no Excel.

3.5.3. Desenvolvimento dos Scripts de Teste

Foram elaborados dois conjuntos principais de scripts para a execução dos testes:

- Script de Teste para MySQL:

  -Objetivo: Executar uma consulta SQL pré-definida que simula a busca por produtos com a palavra “TV”, medindo quantidade máxima de consulta realizada, tempo médio de resposta, consumo de CPU, uso de memória RAM e taxa de I/O de disco.

  -Descrição: O script estabelece a conexão com o MySQL através da biblioteca mysql.connector e utiliza o módulo *threading* para lançar várias threads simultâneas, simulando um ambiente de alta carga. Em cada iteração, o script registra o tempo de execução da consulta (usando time.perf_counter()), monitora o uso de CPU, memória e taxa I/O de disco com a psutil e armazena os dados em arquivos CSV para posterior análise.

Figura 12 - Script de Realização dos Testes MySQL

```python
import mysql.connector
import csv
import threading
import time
import psutil

config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'admin',
    'database': 'ecommerce_mercadoLivre'
}

consulta = ("""
    SELECT
        p.titulo,
        p.imagem,
        p.preco,
        COALESCE(AVG(a.nota), 0) AS media_avaliacao
    FROM produtos p
    LEFT JOIN avaliacoes a ON p.id = a.produto_id
    WHERE p.titulo LIKE '% TV %'
    GROUP BY p.id, p.titulo, p.imagem, p.preco
    ORDER BY media_avaliacao DESC;
""")

DURACAO_TESTE = 60
NUM_THREADS = 10

metricas_consultas = []  # (thread_id, tempo_ms)
metricas_sistema = []    # (timestamp, cpu_percent, mem_percent, read_bytes, write_bytes)
lock = threading.Lock()

def monitorar_sistema():
    fim_teste = time.time() + DURACAO_TESTE
    while time.time() < fim_teste:
        timestamp = time.strftime('%Y-%m-%d %H:%M:%S')
        cpu = psutil.cpu_percent(interval=1)
        mem = psutil.virtual_memory().percent
        io = psutil.disk_io_counters()
        with lock:
            metricas_sistema.append((timestamp, cpu, mem, io.read_bytes, io.write_bytes))

def executar_consulta(thread_id):
    inicio_teste = time.time()
    while time.time() - inicio_teste < DURACAO_TESTE:
        conn = mysql.connector.connect(**config, autocommit=True)
        cursor = conn.cursor()
        inicio = time.perf_counter()
        cursor.execute(consulta)
        cursor.fetchall()
        fim = time.perf_counter()
        with lock:
            metricas_consultas.append((thread_id, (fim - inicio) * 1000))
        cursor.close()
        conn.close()

def main():
    thread_monitoramento = threading.Thread(target=monitorar_sistema)
    thread_monitoramento.start()

    threads = []
    for i in range(NUM_THREADS):
        t = threading.Thread(target=executar_consulta, args=(i,))
        threads.append(t)
        t.start()

    for t in threads:
        t.join()
    thread_monitoramento.join()

    with open('metricas_consultas.csv', 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['ThreadID', 'Tempo (ms)'])
        writer.writerows(metricas_consultas)
        total = len(metricas_consultas)
        media = sum(m[1] for m in metricas_consultas) / total if total else 0
        writer.writerow([])
        writer.writerow(['Total de consultas executadas', total])
        writer.writerow(['Tempo médio de resposta (ms)', f'{media:.2f}'])

    with open('metricas_sistema.csv', 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['Timestamp', 'CPU (%)', 'Memória (%)', 'Leitura (bytes)', 'Escrita
(bytes)'])writer.writerows(metricas_sistema)

    print("\nTeste concluído.")
    print(f"Total de consultas executadas: {total}")
    print(f"Tempo médio de resposta: {media:.2f} ms")
    print("Métricas salvas em 'metricas_consultas.csv' e 'metricas_sistema.csv'.")

if __name__ == "__main__":
    main()
```

Fonte: Elaborado pelo autor (2025).

- Script de Teste para MongoDB:

  -Objetivo: Executar um *pipeline* de agregação que simula a mesma consulta de busca por “TV” no MongoDB, utilizando o operador $regex e o $*lookup* para integrar informações de avaliações, medindo as mesmas métricas.

  -Descrição: O script se conecta ao MongoDB usando pymongo e emprega *threading* para realizar consultas simultâneas. Cada thread executa um *pipeline* de agregação que filtra os documentos cujo campo "título" contenha "TV" (usando $regex), integra os dados de avaliações através do $*lookup*, agrupa os resultados e ordena pela média das avaliações. Paralelamente, outra thread monitora o uso de CPU, memória e I/O de disco com psutil. Os resultados, incluindo o tempo de resposta de cada consulta e as métricas do sistema, são salvos em arquivos CSV.

Figura 13 - Script de Realização dos Testes MongoDB

```python
from pymongo import MongoClient
import csv
import threading
import time
import psutil

client = MongoClient("mongodb://localhost:27017")
db = client["ecommerce_mercadoLivre"]
produtos_coll = db["produtos"]
avaliacoes_coll = db["avaliacoes"]

pipeline = [
    {"$match": {"titulo": {"$regex": "TV", "$options": "i"}}},
    {"$lookup": {"from": "avaliacoes", "localField": "id", "foreignField": "produto_id", "as":
"avaliacoes"}},
    {"$unwind": {"path": "$avaliacoes", "preserveNullAndEmptyArrays": True}},
    {"$group": {
        "_id": "$id",
        "titulo": {"$first": "$titulo"},
        "imagem": {"$first": "$imagem"},
        "preco": {"$first": "$preco"},
        "media_avaliacao": {"$avg": {"$ifNull": ["$avaliacoes.nota", 0]}}
    }},
    {"$sort": {"media_avaliacao": -1}}
]

DURACAO_TESTE = 60
NUM_THREADS = 10

metricas_consultas = []
metricas_sistema = []
lock = threading.Lock()

def monitorar_sistema():
    fim_teste = time.time() + DURACAO_TESTE
    while time.time() < fim_teste:
        timestamp = time.strftime('%Y-%m-%d %H:%M:%S')
        cpu = psutil.cpu_percent(interval=1)
        mem = psutil.virtual_memory().percent
        io = psutil.disk_io_counters()
        with lock:
            metricas_sistema.append((timestamp, cpu, mem, io.read_bytes, io.write_bytes))

def executar_consulta(thread_id):
    inicio_teste = time.time()
    while time.time() - inicio_teste < DURACAO_TESTE:
        inicio = time.perf_counter()
        list(produtos_coll.aggregate(pipeline))
        fim = time.perf_counter()
        with lock:
            metricas_consultas.append((thread_id, (fim - inicio) * 1000))

def main():
    monitor_thread = threading.Thread(target=monitorar_sistema)
    monitor_thread.start()

    threads = []
    for i in range(NUM_THREADS):
        t = threading.Thread(target=executar_consulta, args=(i,))
        threads.append(t)
        t.start()

    for t in threads:
        t.join()
    monitor_thread.join()

    with open('metricas_consultas_mongo.csv', 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['ThreadID', 'Tempo (ms)'])
        writer.writerows(metricas_consultas)
        total = len(metricas_consultas)
        media = sum(m[1] for m in metricas_consultas) / total if total else 0
        writer.writerow([])
        writer.writerow(['Total de consultas executadas', total])
        writer.writerow(['Tempo médio de resposta (ms)', f'{media:.2f}'])

    with open('metricas_sistema_mongo.csv', 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['Timestamp', 'CPU (%)', 'Memória (%)', 'Leitura (bytes)', 'Escrita (bytes)'])
        writer.writerows(metricas_sistema)

    print(f"Total de consultas executadas: {total}")
    print(f"Tempo médio de resposta: {media:.1f} ms")
    print("Teste concluído. Métricas salvas em 'metricas_consultas_mongo.csv' e
'metricas_sistema_mongo.csv'.")

if __name__ == "__main__":
    main()
```

Fonte: Elaborado pelo autor (2025).

- Script de Processamento dos Resultados:

  -Objetivo: Processar os arquivos CSV gerados pelos testes demonstrados anteriormente, calcular as médias das métricas.

  -Descrição: Esse script utiliza a biblioteca pandas para manipulação dos dados, definindo as médias dos resultados gerando um arquivo em formato .xlsx com os resultados. Os gráficos comparativos de desempenho entre MySQL e MongoDB foram elaborados no Excel, através do arquivo mencionado anteriormente, proporcionando uma visualização clara das diferenças entre os dois sistemas.

Figura 14 - Script de Tratamento de Dados dos Testes

```python
import os
import pandas as pd

def listar_arquivos_csv(padrao):
    return [f for f in os.listdir() if f.endswith(".csv") and padrao in f]

def processar_dados(file_paths, colunas_metricas, num_cores):
    dfs = []
    for file in file_paths:
        if os.path.getsize(file) > 0:
            try:
                df = pd.read_csv(file, sep=",")
                if all(col in df.columns for col in colunas_metricas):
                    dfs.append(df)
            except Exception:
                pass
    if not dfs:
        return {col: 0 for col in colunas_metricas}, 0
    dados = pd.concat(dfs, ignore_index=True)
    if "CPU (%)" in dados.columns:
        dados["CPU (%)"] = (dados["CPU (%)"] / num_cores).clip(lower=0, upper=100)
    if "Memória (%)" in dados.columns:
        dados["Memória (%)"] = dados["Memória (%)"].clip(lower=0, upper=100)
    if "Leitura (bytes)" in dados.columns:
        dados["Leitura (MB)"] = dados["Leitura (bytes)"] / (1024 * 1024)
    if "Escrita (bytes)" in dados.columns:
        dados["Escrita (MB)"] = dados["Escrita (bytes)"] / (1024 * 1024)
    medias = {}
    for col in colunas_metricas:
        medias[col] = round(dados[col].mean(), 2) if col in dados.columns else 0
    if "Leitura (MB)" in dados.columns:
        medias["Leitura (MB)"] = round(dados["Leitura (MB)"].mean(), 2)
    if "Escrita (MB)" in dados.columns:
        medias["Escrita (MB)"] = round(dados["Escrita (MB)"].mean(), 2)
    return medias, len(dados)

def main():
    num_cores = os.cpu_count()

    arquivos_consultas_mongo = listar_arquivos_csv("mongo_consulta")
    arquivos_recursos_mongo = listar_arquivos_csv("mongo_recurso")
    arquivos_consultas_mysql = listar_arquivos_csv("mysql_consulta")
    arquivos_recursos_mysql = listar_arquivos_csv("mysql_recurso")

    colunas_consultas = ["Tempo (ms)"]
    medias_consultas_mongo, total_consultas_mongo = processar_dados(arquivos_consultas_mongo,
colunas_consultas, num_cores)
    medias_consultas_mysql, total_consultas_mysql = processar_dados(arquivos_consultas_mysql,
colunas_consultas, num_cores)

    colunas_sistema = ["CPU (%)", "Memória (%)", "Leitura (bytes)", "Escrita (bytes)"]
    medias_sistema_mongo, _ = processar_dados(arquivos_recursos_mongo, colunas_sistema, num_cores)
    medias_sistema_mysql, _ = processar_dados(arquivos_recursos_mysql, colunas_sistema, num_cores)

    def formatar_percentual(valor):
        return f"{valor}%"

    mongo_dict = {
        "Total de Consultas Executadas": total_consultas_mongo,
        "Tempo Médio de Resposta (ms)": medias_consultas_mongo.get("Tempo (ms)", 0),
        "CPU Média (%)": formatar_percentual(medias_sistema_mongo.get("CPU (%)", 0)),
        "Memória Média (%)": formatar_percentual(medias_sistema_mongo.get("Memória (%)", 0)),
        "Leitura Média (MB)": medias_sistema_mongo.get("Leitura (MB)", 0),
        "Escrita Média (MB)": medias_sistema_mongo.get("Escrita (MB)", 0)
    }

    mysql_dict = {
        "Total de Consultas Executadas": total_consultas_mysql,
        "Tempo Médio de Resposta (ms)": medias_consultas_mysql.get("Tempo (ms)", 0),
        "CPU Média (%)": formatar_percentual(medias_sistema_mysql.get("CPU (%)", 0)),
        "Memória Média (%)": formatar_percentual(medias_sistema_mysql.get("Memória (%)", 0)),
        "Leitura Média (MB)": medias_sistema_mysql.get("Leitura (MB)", 0),
        "Escrita Média (MB)": medias_sistema_mysql.get("Escrita (MB)", 0)
    }

    df_comparacao = pd.DataFrame({
        "Métrica": [
            "Total de Consultas Executadas",
            "Tempo Médio de Resposta (ms)",
            "CPU Média (%)",
            "Memória Média (%)",
            "Leitura Média (MB)",
            "Escrita Média (MB)"
        ],
        "MongoDB": [
            mongo_dict["Total de Consultas Executadas"],
            mongo_dict["Tempo Médio de Resposta (ms)"],
            mongo_dict["CPU Média (%)"],
            mongo_dict["Memória Média (%)"],
            mongo_dict["Leitura Média (MB)"],
            mongo_dict["Escrita Média (MB)"]
        ],
        "MySQL": [
            mysql_dict["Total de Consultas Executadas"],
            mysql_dict["Tempo Médio de Resposta (ms)"],
            mysql_dict["CPU Média (%)"],
            mysql_dict["Memória Média (%)"],
            mysql_dict["Leitura Média (MB)"],
            mysql_dict["Escrita Média (MB)"]
        ]
    })

    print("\nComparação de Desempenho:")
    print(df_comparacao)

    try:
        df_comparacao.to_excel("analise_metrica.xlsx", index=False, engine="openpyxl")
        print("\nAnálise concluída. Resultados salvos em 'analise_metrica.xlsx'.")
    except ModuleNotFoundError:
        print("Pacote 'openpyxl' não encontrado. Instale com: pip install openpyxl")

if __name__ == "__main__":
    main()
```

Fonte: Elaborado pelo autor (2025).

3.5.4 Documentação das Consultas

As consultas utilizadas estão descritas nas Figuras 15 e 16, que apresentam os scripts equivalentes utilizados para o MySQL e o MongoDB, respectivamente.

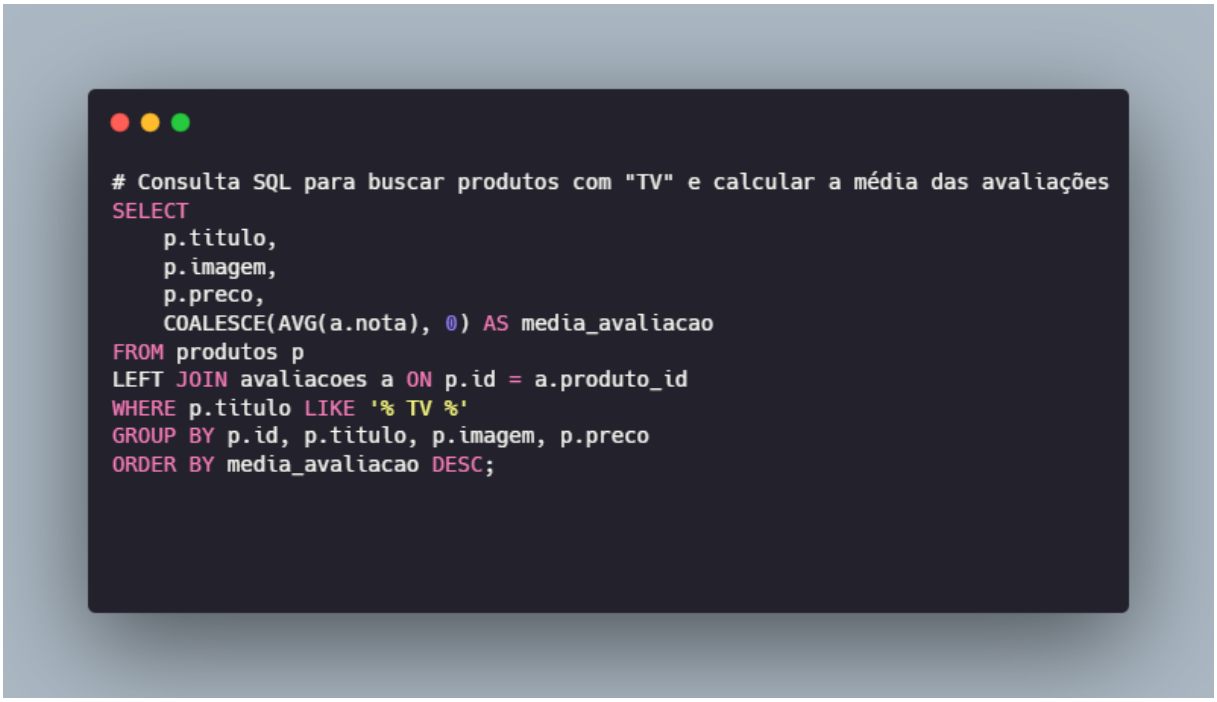
- Imagem 15 – Consulta SQL:

  Contém a query completa utilizada no MySQL para simular a busca por “TV”, incluindo as junções e a ordenação dos resultados por média de avaliações.

- Imagem 16 – *Pipeline* de Agregação no MongoDB:

Contém o *pipeline* utilizado para simular a mesma consulta no MongoDB, detalhando cada estágio do processamento (match, lookup, unwind, group e sort).

Figura 15 - Consulta MySQL



```sql
# Consulta SQL para buscar produtos com "TV" e calcular a média das avaliações
SELECT
    p.titulo,
    p.imagem,
    p.preco,
    COALESCE(AVG(a.nota), 0) AS media_avaliacao
FROM produtos p
LEFT JOIN avaliacoes a ON p.id = a.produto_id
WHERE p.titulo LIKE '% TV %'
GROUP BY p.id, p.titulo, p.imagem, p.preco
ORDER BY media_avaliacao DESC;
```

Fonte: Elaborado pelo autor (2025).

Figura 16 - Consulta MongoDB

```
#Pipeline de agregação: busca produtos com "TV" e calcula a média das
avaliações
pipeline = [
    {"$match": {"titulo": {"$regex": "TV", "$options": "i"}}},
    {"$lookup": {
        "from": "avaliacoes",
        "localField": "id",
        "foreignField": "produto_id",
        "as": "avaliacoes"
    }},
    {"$unwind": {
        "path": "$avaliacoes",
        "preserveNullAndEmptyArrays": True
    }},
    {"$group": {
        "_id": "$id",
        "titulo": {"$first": "$titulo"},
        "imagem": {"$first": "$imagem"},
        "preco": {"$first": "$preco"},
        "media_avaliacao": {"$avg": {"$ifNull": ["$avaliacoes.nota", 0]}}
    }},
    {"$sort": {"media_avaliacao": -1}}
]
```

Fonte: Elaborado pelo autor (2025).

3.6 Procedimentos de Análise dos Resultados

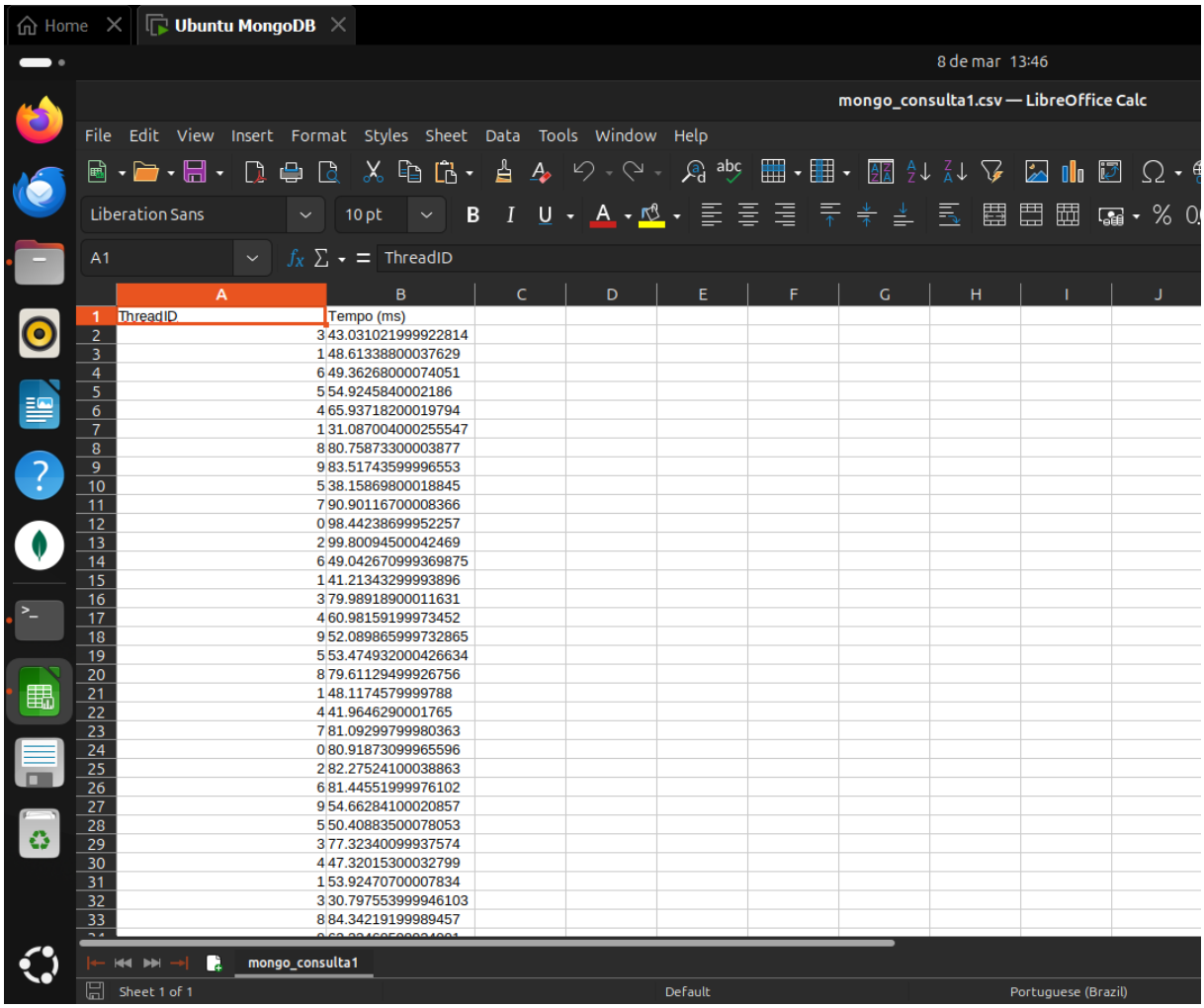
Após realizar os testes de performance e coletar os resultados obtidos para ambos os sistemas de gerenciamento de banco de dados (SGBDs), as métricas de cada consulta (temporalidade da resposta do sistema utilizado em relação ao tempo real gasto na execução da tarefa demandada pelo usuário), o uso da unidade central de processamento (CPU), o consumo de memória, taxa de escrita e leitura de disco e o número total de consultas finalizadas foram armazenados em arquivos no formato CSV. Essencial para converter os dados brutos em informações comparáveis e possibilitar uma avaliação sólida e quantitativa das capacidades dos sistemas.

3.6.1 Coleta e Consolidação dos Dados

Os resultados coletados de cada teste (realizados de forma concorrente por múltiplas threads) foram armazenados em arquivos CSV separados para MySQL e MongoDB. O primeiro arquivo bancoRespectivo_consultaX.csv contendo:

- Identificador da Thread: Para identificar a origem dos dados de execução.
- Tempo de Resposta (ms): Tempo medido em milissegundos para cada execução de consulta.

Figura 17 - Resultados dos Testes CSV



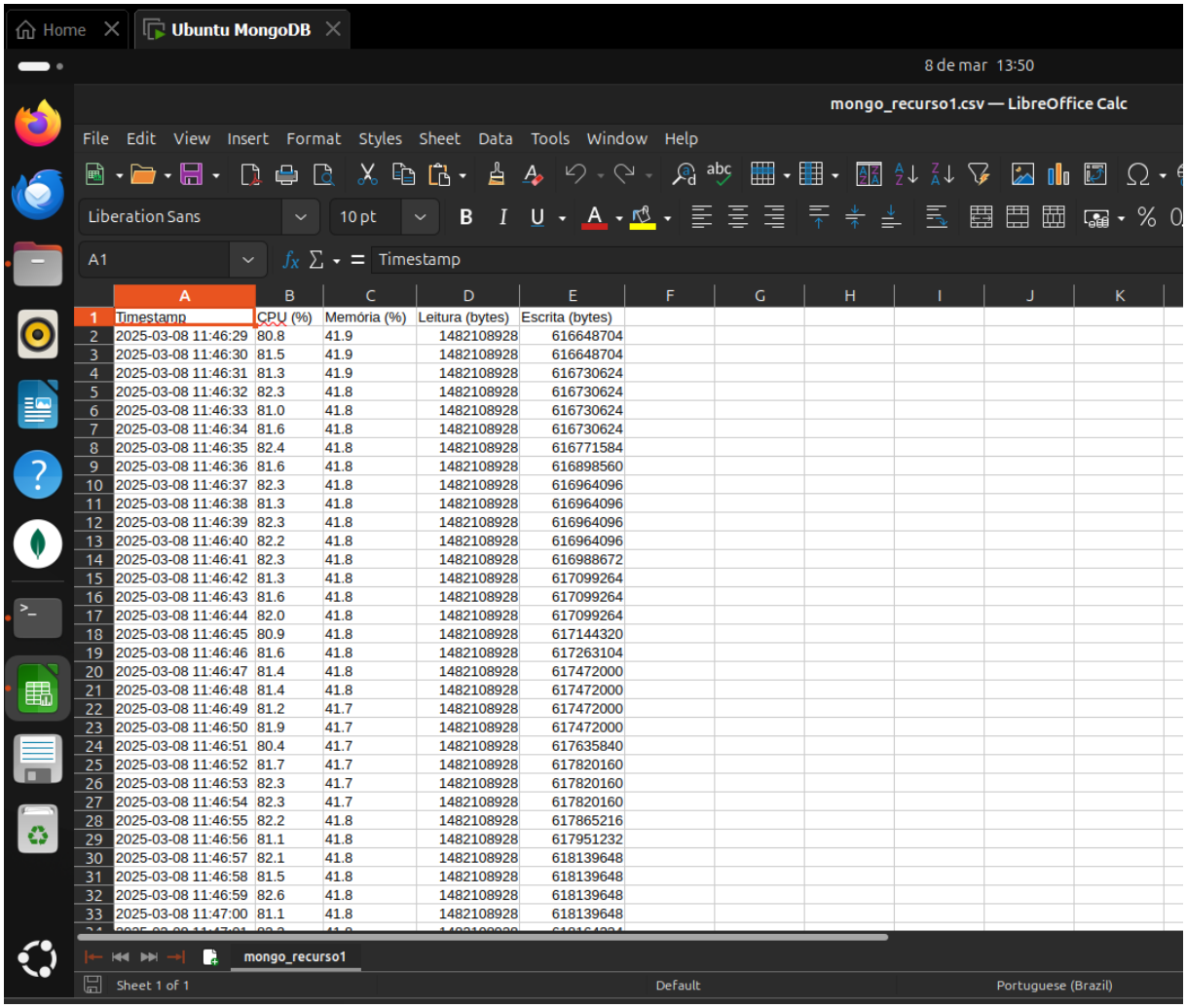
Fonte: Elaborado pelo autor (2025).

O segundo arquivo gerado pelo teste nomeado de bancoRespectivo_recursoX.csv é composto por:

- Timestamp: Indica a data e hora em que as métricas foram coletadas.

- CPU (%): Percentual de uso da CPU no momento da coleta, obtido via psutil.
- Memória (%): Porcentagem de memória RAM utilizada, também capturada pelo psutil.
- Leitura (bytes): Quantidade de bytes lidos do disco, conforme informado pelo psutil.disk_io_counters().
- Escrita (bytes): Quantidade de bytes escritos no disco, também monitorada pelo psutil.disk_io_counters().

Figura 18 - Uso de Recursos Durante os Testes CSV



	A	B	C	D	E
1	Timestamp	CPU (%)	Memória (%)	Leitura (bytes)	Escrita (bytes)
2	2025-03-08 11:46:29	80.8	41.9	1482108928	616648704
3	2025-03-08 11:46:30	81.5	41.9	1482108928	616648704
4	2025-03-08 11:46:31	81.3	41.9	1482108928	616730624
5	2025-03-08 11:46:32	82.3	41.8	1482108928	616730624
6	2025-03-08 11:46:33	81.0	41.8	1482108928	616730624
7	2025-03-08 11:46:34	81.6	41.8	1482108928	616730624
8	2025-03-08 11:46:35	82.4	41.8	1482108928	616771584
9	2025-03-08 11:46:36	81.6	41.8	1482108928	616898560
10	2025-03-08 11:46:37	82.3	41.8	1482108928	616964096
11	2025-03-08 11:46:38	81.3	41.8	1482108928	616964096
12	2025-03-08 11:46:39	82.3	41.8	1482108928	616964096
13	2025-03-08 11:46:40	82.2	41.8	1482108928	616964096
14	2025-03-08 11:46:41	82.3	41.8	1482108928	616988672
15	2025-03-08 11:46:42	81.3	41.8	1482108928	617099264
16	2025-03-08 11:46:43	81.6	41.8	1482108928	617099264
17	2025-03-08 11:46:44	82.0	41.8	1482108928	617099264
18	2025-03-08 11:46:45	80.9	41.8	1482108928	617144320
19	2025-03-08 11:46:46	81.6	41.8	1482108928	617263104
20	2025-03-08 11:46:47	81.4	41.8	1482108928	617472000
21	2025-03-08 11:46:48	81.4	41.8	1482108928	617472000
22	2025-03-08 11:46:49	81.2	41.7	1482108928	617472000
23	2025-03-08 11:46:50	81.9	41.7	1482108928	617472000
24	2025-03-08 11:46:51	80.4	41.7	1482108928	617635840
25	2025-03-08 11:46:52	81.7	41.7	1482108928	617820160
26	2025-03-08 11:46:53	82.3	41.7	1482108928	617820160
27	2025-03-08 11:46:54	82.3	41.7	1482108928	617820160
28	2025-03-08 11:46:55	82.2	41.8	1482108928	617865216
29	2025-03-08 11:46:56	81.1	41.8	1482108928	617951232
30	2025-03-08 11:46:57	82.1	41.8	1482108928	618139648
31	2025-03-08 11:46:58	81.5	41.8	1482108928	618139648
32	2025-03-08 11:46:59	82.6	41.8	1482108928	618139648
33	2025-03-08 11:47:00	81.1	41.8	1482108928	618139648

Fonte: Elaborado pelo autor (2025).

3.6.2 Processamento Estatístico dos Dados

Para a análise dos dados, foi desenvolvido um script em Python utilizando as bibliotecas pandas. Este script realizou as seguintes operações:

- Leitura dos Arquivos CSV: O script utiliza a função listar_arquivos_csv para identificar e importar os arquivos gerados pelos testes, filtrando-os por padrões específicos e armazenando os dados em DataFrames.
- Cálculo das Médias: A função processar_dados consolida os DataFrames, converte valores (por exemplo, de bytes para megabytes) e calcula as médias das métricas, como o tempo de resposta, uso de CPU (normalizado pelo número de núcleos) e uso de memória, utilizando métodos como mean().
- Total de Consultas: É feita a contagem total de consultas ou operações executadas durante os testes, utilizando a função len() aplicada aos DataFrames consolidados.
- Geração dos Resultados: Os resultados são organizados em um DataFrame comparativo e exportados para um arquivo em formato Excel (.xlsx), facilitando a análise visual e a comparação do desempenho entre MySQL e MongoDB através de gráficos.

3.6.3 Geração de Gráficos Comparativos

Após a consolidação dos dados coletados nos testes, foram geradas visualizações comparativas para facilitar a análise do desempenho entre os bancos MySQL e MongoDB. As médias de tempo de resposta, uso de CPU, memória e taxa de I/O de disco foram calculadas e representadas por meio de gráficos de barras, produzidos no Microsoft Excel. Esses gráficos evidenciam as principais diferenças de comportamento entre os dois SGBDs em relação aos recursos consumidos durante as requisições.

3.6.4 Conclusão da Análise

Com base nos passos delineados anteriormente e na integração das medidas obtidas, foi possível ter uma visão completa do desempenho dos Sistemas Gerenciadores de Bancos de Dados (SGBDs), no contexto de teste em uma rotina de consulta de produtos em um sistema de e-commerce conforme proposto. A análise dos resultados será respaldada por cálculos estatísticos e gráficos comparativos para ajudar na identificação do sistema com melhor desempenho em termos de tempo de resposta, consumo de recursos e capacidade para lidar com

um grande volume de consultas. Essas avaliações visam embasar a hipotética escolha do SGBD mais adequado para o problema proposto.

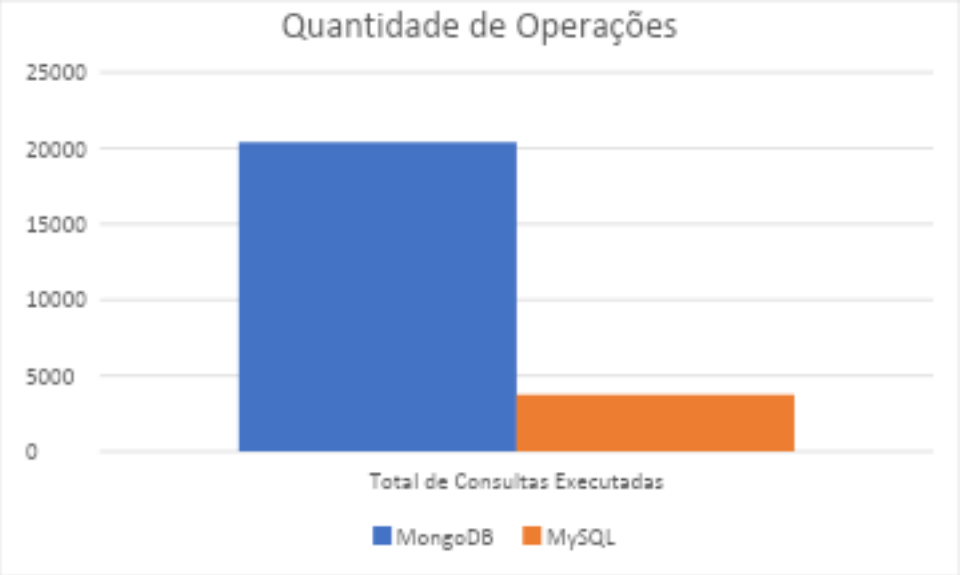
4 RESULTADOS E DISCUSSÃO

4.1 Análise Comparativa

A partir dos dados coletados e consolidados durante os testes de desempenho, foi possível comparar MySQL e MongoDB em quatro métricas principais: quantidade de operações concluídas, tempo médio de resposta, uso médio de CPU e uso médio de memória.

4.1.1 Quantidade de Operações Concluídas

Figura 19 - Resultado Quantidade de Operações

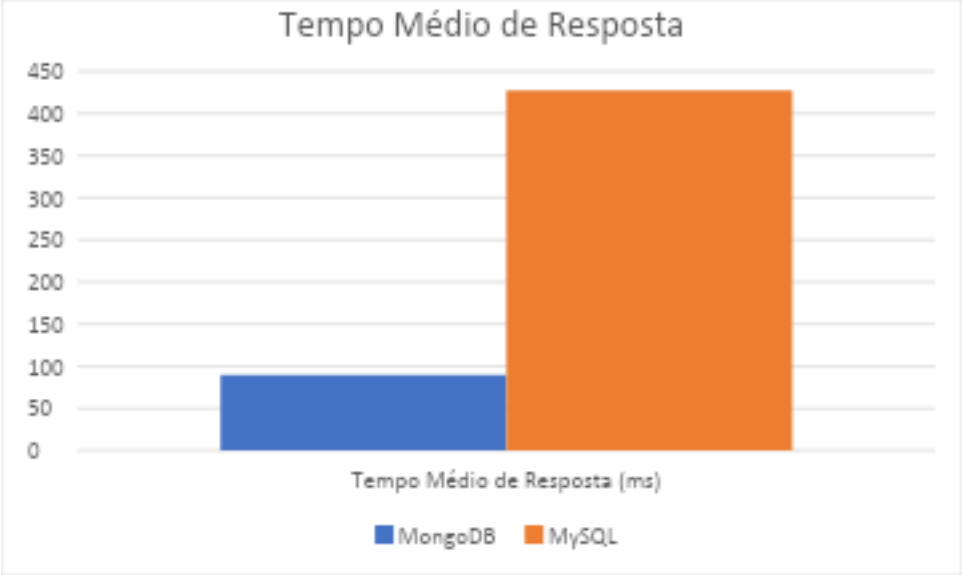


Fonte: Elaborado pelo autor (2025).

Durante os testes realizados, observou-se que o MongoDB executou 20.400 operações, enquanto o MySQL completou 3.752 operações no mesmo intervalo de tempo. Essa diferença significativa pode ser atribuída às arquiteturas distintas de cada sistema de gerenciamento de banco de dados. O MongoDB, sendo um banco de dados NoSQL orientado a documentos, armazena dados em estruturas flexíveis em BSON. Conhecido por sua alta performance, conforme mencionado no artigo "MongoDB: o que é, quais suas características e benefícios" da Alura, o MongoDB é "otimizado para oferecer alta performance em operações de leitura e gravação" (ALURA, s.d.).

4.1.2 Tempo Médio de Resposta (ms)

Figura 20 - Resultado Tempo Médio de Resposta

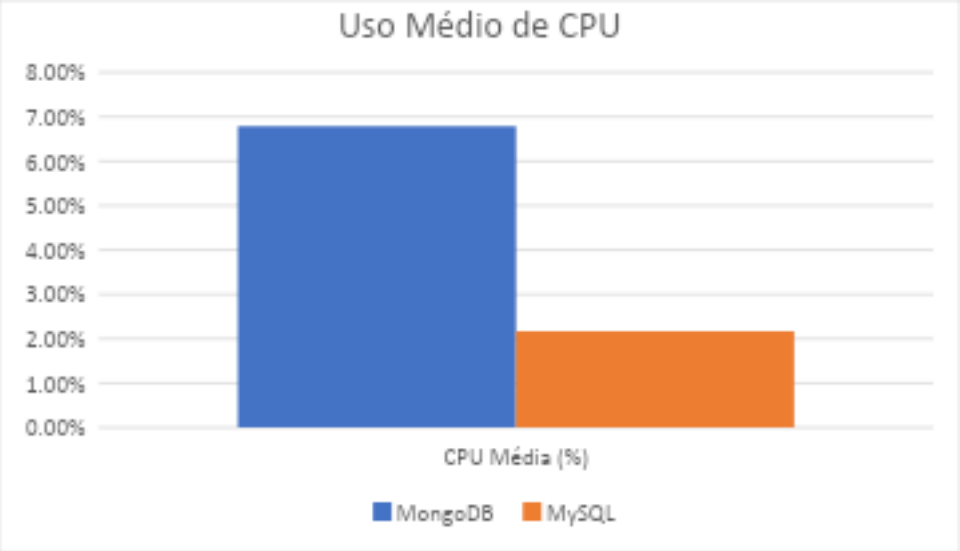


Fonte: Elaborado pelo autor (2025).

O tempo médio de resposta reflete o período necessário para que um SGBD retorne o resultado de uma consulta específica a base de dados. Com base nos resultados obtidos, observa-se que o MongoDB apresentou um tempo médio de 89,28 ms, enquanto o MySQL registrou um tempo médio de 427,09 ms. Em contextos de e-commerce a agilidade nas consultas pode influenciar diretamente a decisão de compra do usuário. Embora seja importante considerar que o tempo de resposta do banco de dados é apenas um dos fatores que compõem o desempenho final percebido pelo usuário.

4.1.3 Uso Médio de CPU (%)

Figura 21 - Resultado Uso Médio de CPU

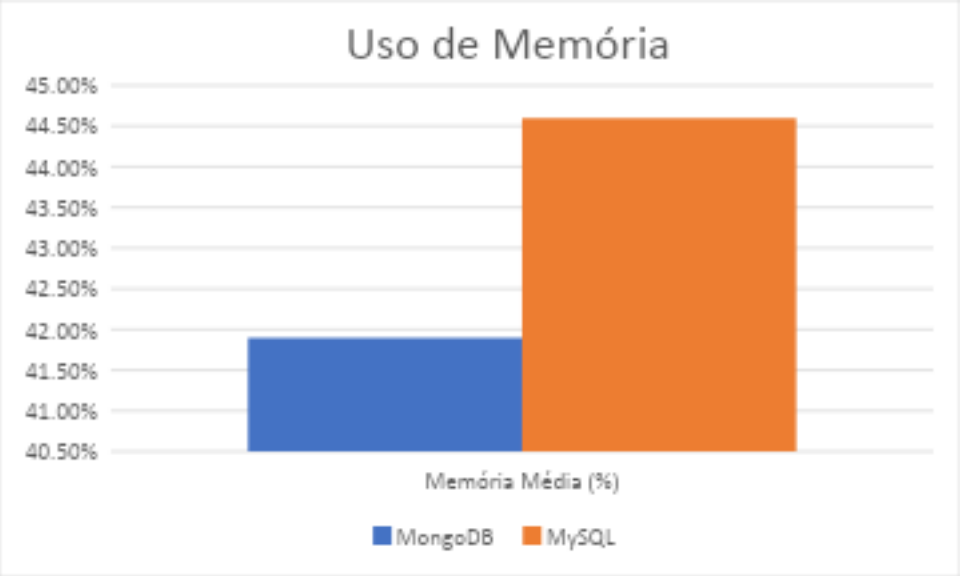


Fonte: Elaborado pelo autor (2025).

Em relação a utilização média da CPU o MongoDB apresentou um uso médio de 6,79%, enquanto o MySQL registrou 2,17% no mesmo período. Essa diferença está atribuída às características específicas de cada sistema, o MongoDB realiza operações de agregação sobre documentos flexíveis, exigindo um maior processamento, enquanto o MySQL executa consultas baseadas em tabelas relacionais estruturadas, resultando em menor consumo médio de CPU, conforme demonstrado com os resultados dos testes.

4.1.4 Uso Médio de Memória (MB)

Figura 22 - Resultado Uso de Memória

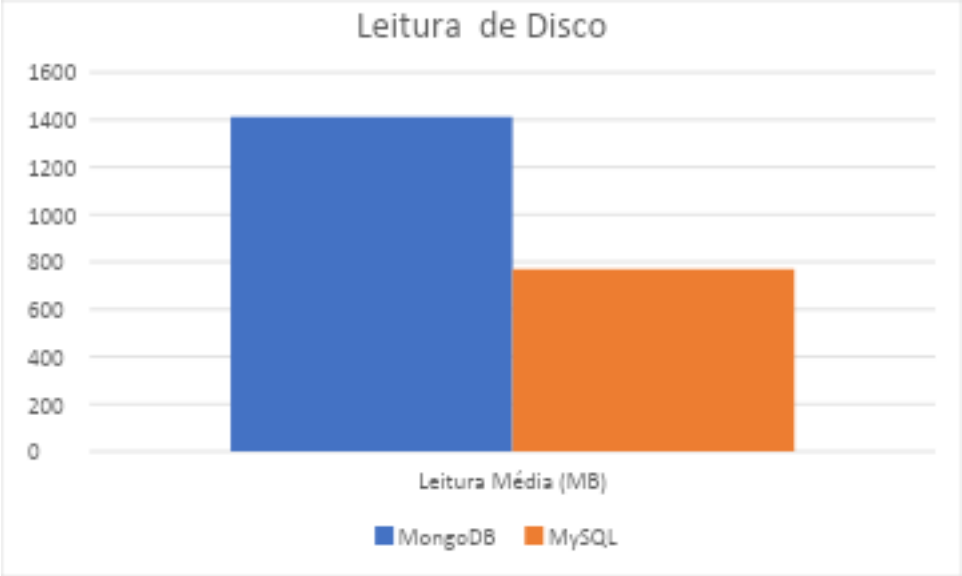


Fonte: Elaborado pelo autor (2025).

Outro ponto importante a considerar é a utilização média de memória: o MySQL apresentou um consumo médio de 44,60%, enquanto o MongoDB consumiu em média 41,90%. Esses valores demonstram que durante os testes realizados o MySQL apresentou um uso ligeiramente superior de memória em comparação ao MongoDB. Essa diferença foi obtida diretamente por meio dos dados monitorados durante o período das consultas executadas pelos scripts, sem considerar fatores externos aos processos dos próprios sistemas.

4.1.5 Leitura de Disco (MB)

Figura 23 - Resultado Leitura de Disco



Fonte: Elaborado pelo autor (2025).

Durante o período de testes o MongoDB apresentou uma leitura média de disco de 1413,45 MB, enquanto o MySQL registrou 768,99 MB. Esta média justifica-se devido o MongoDB ter realizado 5x mais consultas em relação ao MySQL.

4.1.6 Escrita de Disco (MB)

Figura 24 - Resultado Escrita de Disco

Fonte: Elaborado pelo autor (2025).

Em relação a escrita média de disco, o MongoDB registrou 597,02 MB, enquanto o MySQL apresentou 55,1 MB durante o mesmo intervalo de testes. Isso demonstra que o MongoDB executou significativamente mais operações de escrita em comparação ao MySQL.

Tabela 1 - Resultados Gerais

Métrica	**MongoDB**	**MySQL**
Quantidade de Operações	20.400	3.752
Tempo Médio (ms)	89,28	427,09
Uso Médio de CPU (%)	6,79	2,17
Uso Médio de Memória (%)	41,90	44,60
Leitura Média de Disco (MB)	1413,45	768,99
Escrita Média de Disco (MB)	597,02	55,1

Fonte: Elaborado pelo autor (2025).

4.1.7 Considerações Finais

Esses resultados reforçam as características centrais de cada SGBD:

- MongoDB:
  - Pontos Fortes:
    - Tempo médio de resposta baixo (89,28 ms).
    - Alta quantidade de operações concluídas (20.400 consultas).
    - Maior eficiência na leitura e escrita de disco.
  - Ponto de Atenção:
    - Maior consumo de CPU (6,79%).
- MySQL:
  - Pontos Fortes:
    - Menor uso médio de CPU (2,17%).
    - Modelo relacional consolidado e suporte robusto a transações ACID.
- Pontos de Atenção:
    - Tempo médio de resposta elevado (427,09 ms).
    - Menor quantidade de operações concluídas (3.752 consultas).
    - Maior consumo médio de memória (44,60%).

Os resultados demonstram que o MongoDB é mais eficiente no processamento de consultas sob alta carga, apresentando menor tempo de resposta e maior volume de operações concluídas. No entanto, esse desempenho vem acompanhado de um consumo mais elevado de CPU. Por outro lado, o MySQL se destaca pelo menor uso de CPU e pelo suporte consolidado a transações ACID, embora apresente desempenho inferior em termos de velocidade e volume de consultas concluídas.

5 CONCLUSÃO

Os testes realizados mostraram uma diferença considerável no desempenho entre o MongoDB e o MySQL, com o MongoDB se destacando em áreas fundamentais como a quantidade de operações realizadas e o tempo médio de resposta. Essas características evidenciam sua habilidade em lidar com um grande volume de consultas com maior eficiência, tornando-o uma opção viável para situações que requerem respostas rápidas e processamento ágil.

Embora o MongoDB consuma mais CPU comparado ao MySQL, este fato pode ser considerado, mas uma possível solução seria a sua capacidade de escalabilidade horizontalmente, o que permite a distribuição equitativa das operações conforme necessário. Além disso, a sua estrutura baseada em documentos simplifica processos de junção, resultando em maior eficiência ao lidar com grandes quantidades de dados. Portanto, no âmbito deste estudo específico, o MongoDB aparece como a escolha mais indicada para resolver o problema apresentado.

6 REFERÊNCIAS BIBLIOGRÁFICAS

DATE, C. J. Introdução a Sistemas de Banco de Dados. 8. ed. Rio de Janeiro: LTC, 2004.

DB-ENGINES. Ranking dos sistemas de gerenciamento de banco de dados. Disponível em: https://db-engines.com/en/ranking. Acesso em: 21 mar. 2025.

DEL FABRO, Marcos Didonet. Bancos de Dados NoSQL: Modelos e Arquiteturas. Porto Alegre: Bookman, 2016.

ELMASRI, R.; NAVATHE, S. B. Fundamentos de Sistemas de Banco de Dados. 6. ed. São Paulo: Pearson, 2011.

ELMASRI, R., & NAVATHE, S. B. (2019). Sistemas de Banco de Dados (6ª ed.). Pearson Universidades.

ERICKSON, Jeffrey. MySQL: Entendendo o que é e como é usado. Oracle, 29 ago. 2024. Disponível em: https://www.oracle.com/br/mysql/what-is-mysql/. Acesso em: 11 mar. 2025.

GARCIA, V. S.; SOTTO, E. C. S. Comparativo entre os modelos de banco de dados relacional e não-relacional. Interface Tecnológica, v. 16, n. 2, 2019. Disponível em: https://www.researchgate.net/publication/338190693_COMPARATIVO_ENTRE_OS_MODELOS_DE_BANCO_DE_DADOS_RELACIONAL_E_NAO-RELACIONAL. Acesso em: 3 mar. 2025.

HEUSER, C. A. (2009). Projeto de Banco de Dados (6ª ed.). Bookman.
MACHADO, Felipe Nery Rodrigues. Banco de Dados: Projeto e Implementação. 4ª ed. São Paulo: Érica, 2018.

IANNI, V. Introdução aos bancos de dados NoSQL. Disponível em: https://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044. Acesso em: 02 mar. 2025

LÓSCIO B.F.; OLIVEIRA, H.R.; PONTES, J.C.S.; NoSql no desenvolvimento de aplicações

Web colaborativas. Disponível em: <https://www.researchgate.net/profile/Bernadette_Loscio/publication/268201466_NoSQL_no_desenvolvimento_de_aplicacoes_Web_colaborativas/links/576aa72008aef2a864d1ef8c/NoSQL-nodesenvolvimento-de-aplicacoes-Web-colaborativas.pdf>. Acesso em: 07 mar 2025.

MONGODB. MongoDB Manual – Security Features. Disponível em: https://www.mongodb.com/docs/manual/. Acesso em: 01 de março de 2025.

MySQL. Documentação Oficial do MySQL 8.0 (em português). Disponível em: https://dev.mysql.com/doc/refman/8.0/en/. Acesso em: 02 mar. 2025

OLIVEIRA, J. R. Bancos de Dados: Modelagem, Projeto e Implementação. 2. ed. São Paulo: Erica, 2016.

PORTO, Fábio; VALDURIEZ, Patrick. Bancos de Dados: Conceitos, Modelos e Sistemas. Rio de Janeiro: Elsevier, 2019.

SAMPAIO, Daniel. E-commerce: guia completo para começar sua loja virtual. RockContent, 05 jun. 2024. Disponível em: https://rockcontent.com/br/blog/e-commerce-guia/. Acesso em: 08 mar. 2025

SHAIBU, Samuel. CSV versus Excel: Fazendo a escolha certa para seus projetos de dados. Datacamp, 29 jul. 2024. Disponível em: https://www.datacamp.com/pt/blog/csv-vs-excel. Acesso em: 08 mar. 2025

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Sistemas de Banco de Dados. 6. ed. Rio de Janeiro: Elsevier, 2006.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. (2020). Sistemas de Banco de Dados (7ª ed.). LTC.

SOARES, Einstein. Banco de Dados Relacional: melhor ou pior?. Disponível em: https://www.dio.me/articles/banco-de-dados-relacional-melhor-ou-pior. Acesso em: 8 mar. 2025.

RIEUF, Emmuelle. History of MySQL. Disponível em:https://www.datasciencecentral.com/history-of-mysql/. Acesso em: 8 mar. 2025.