Associação Propagadora Esdeva

Centro Universitário Academia – UniAcademia

Curso de Engenharia de Software

Trabalho de Conclusão de Curso – Artigo

Desempenho e performance: Estudo comparativo entre MongoDB e PostgreSQL

Gustavo Andrade Medeiros[1]

Centro Universitário Academia, Juiz de Fora, MG

Daves Marcio Silva Martins[2]

Centro Universitário Academia, Juiz de Fora, MG

Linha de pesquisa: Engenharia de Software

RESUMO

O aumento exponencial no volume de dados gerados tem impactado diretamente a eficiência de sistemas de banco de dados, tornando essencial a escolha entre paradigmas relacionais e não-relacionais. Este trabalho realiza uma análise estatística comparativa do desempenho do PostgreSQL, um banco relacional, e do MongoDB, um banco não-relacional, focando na busca de dados e explorando as características arquiteturais de cada tecnologia. A análise considera métricas como tempo de resposta, uso de CPU e consumo de memória, avaliadas em cenários que refletem as identidades dos bancos. Com base nos testes realizados, busca-se compreender as diferenças significativas entre as tecnologias e oferecer subsídios para a escolha da solução mais adequada em diferentes contextos de aplicação.

Palavras-chave: Arquitetura de dados. Performance de sistemas. Sistemas Gerenciamento de Banco de Dados. PostgreSQL. MongoDB.

1 Discente do Curso de Engenharia de Software do Centro Universitário Academia – UniAcademia Email: gustavomedeiros15@outlook.com.br

2 Docente do Curso de Sistemas de Informação do Centro Universitário Academia - UniAcademia. Orientador.

ABSTRACT

The exponential increase in the volume of generated data has directly impacted the efficiency of database systems, making the choice between relational and non-relational paradigms essential. This study conducts a statistical analysis of the performance of PostgreSQL, a relational database, and MongoDB, a non-relational database, focusing on data retrieval and exploring the architectural characteristics of each technology. The analysis considers metrics such as response time, CPU usage, and memory consumption, evaluated in scenarios that highlight the unique features of each database. Based on the conducted tests, the study aims to identify significant differences between the technologies and provide insights to guide the selection of the most suitable solution for various application contexts.

Keywords: *Data architecture. system performance. Database Management Systems. PostgreSQL. MongoDB.*

1. INTRODUÇÃO

O crescente volume de dados gerados e coletados pelas empresas tem moldado o desenvolvimento de software, tornando a eficiência no processamento, análise e disponibilização desses dados um dos principais desafios da tecnologia da informação (Manyika et al., 2011). À medida que esse volume aumenta, o acesso rápido a informações para análise e tomada de decisão se torna essencial para a competitividade das empresas (Stonebraker e Hellerstein, 2005).

Um exemplo relevante é o telescópio James Webb, que gera mais de 57 GB de dados diariamente, exigindo sistemas robustos para gerenciar grandes volumes e consultas intensivas de dados (NASA, 2023). Esse tipo de cenário demanda bancos de dados que ofereçam alto desempenho e suportem consultas complexas, tornando a escolha entre paradigmas relacionais e não-relacionais essencial para garantir performance e escalabilidade.

Este estudo analisa a escolha entre PostgreSQL e MongoDB, dois bancos de dados amplamente adotados e complementares em suas características. O PostgreSQL, um banco

relacional, é destacado por sua consistência e robustez transacional (Krebs, 2018), enquanto o MongoDB, um banco não-relacional, se sobressai em cenários de alta escalabilidade e grandes volumes de dados não estruturados (Banker, 2011). A pesquisa avalia o desempenho desses sistemas por meio de métricas como tempo de resposta, uso de CPU e consumo de memória, em testes que focam na característica de cada banco.

O artigo está dividido em seis seções: introdução, conceitos teóricos sobre NoSQL e SQL, uma revisão sistemática sobre o tema, metodologia aplicada, descrição do estudo de caso, análise dos resultados, e considerações finais com sugestões para pesquisas futuras.

2. REFERENCIAL TEÓRICO

2.1. Banco de Dados

Segundo C. J. Date (2004), um banco de dados é um sistema computadorizado cuja finalidade geral é armazenar informações e permitir que usuário busquem e atualize essas informações. É um conjunto de dados necessários para um grupo de pessoas ou determinada organização, possuindo uma interpretação do mundo real, com vários níveis de interação.

Existem três principais entidades a ser destacadas, o próprio banco de dados, usuário e aplicações que terão acesso aos dados, e o Sistema Gerenciador de Banco de Dados, que gerência as operações no banco de dados, garantindo que os usuários possam inserir, atualizar e buscar informações de maneira eficiente e segura.

2.2 Modelo Relacional

O banco de dados relacional, proposto por E.F. Codd em 1970, organiza dados em tabelas compostas por colunas (atributos) e linhas (tuplas), representando relações entre entidades do mundo real. Essas tabelas são interligadas por chaves primárias e estrangeiras, garantindo a integridade referencial dos dados (Codd, 1970). A principal característica do modelo relacional é a consistência dos dados, obtida por meio da normalização, que evita redundâncias e permite consultas robustas utilizando SQL (Date, 2004).

Os bancos relacionais, como o PostgreSQL, utilizam estruturas de índices para otimizar consultas. Os índices B-Tree, padrão no PostgreSQL, são ideais para consultas com intervalos ou

ordenações, oferecendo acessos rápidos mesmo em grandes volumes de dados (Comer, 1979). Os índices Hash são eficientes em consultas por igualdade, como *WHERE id = 10*, mas apresentam limitações em ordenações e intervalos (Knuth, 1998).

2.3 Modelo Não-Relacional

Os bancos de dados não-relacionais surgiram no final dos anos 2000, impulsionados por grandes empresas como Google e Amazon, para gerenciar volumes massivos de dados com maior eficiência do que os sistemas relacionais (Cattell, 2011). Embora o termo "NoSQL" tenha sido cunhado por Carlo Strozzi em 1998 (Suissa, 2010), o movimento ganhou popularidade apenas em 2009, quando sistemas não-relacionais projetados para lidar com Big Data e alta escalabilidade se tornaram amplamente adotados (Strauch, 2011).

O modelo chave-valor é eficiente para operações rápidas de leitura e escrita. O modelo colunar, por sua vez, organiza os dados em colunas em vez de linhas, o que o torna especialmente otimizado para consultas de agregação em cenários de grandes volumes de dados distribuídos horizontalmente. Já o modelo orientado a documentos utiliza documentos como unidade de armazenamento, geralmente no formato JSON ou BSON (uma versão binária do JSON). Bancos de dados como o MongoDB oferecem flexibilidade significativa, permitindo que documentos distintos tenham conjuntos de campos variados sem a necessidade de seguir esquemas rígidos, tornando-o ideal para aplicações dinâmicas e heterogêneas (Strauch, 2011).

Um dos formatos mais amplamente utilizados nesses sistemas é o JSON (JavaScript Object Notation), criado por Douglas Crockford no início dos anos 2000. Sua simplicidade e capacidade de adaptação a ambientes dinâmicos, como aplicações web e móveis, tornam-no uma escolha natural para bancos de dados NoSQL (Crockford, 2006). A integração do JSON com tecnologias como o MongoDB permite lidar eficientemente com cenários reais, como catálogos de produtos ou perfis de usuários, onde a estrutura dos dados pode variar conforme as necessidades da aplicação.

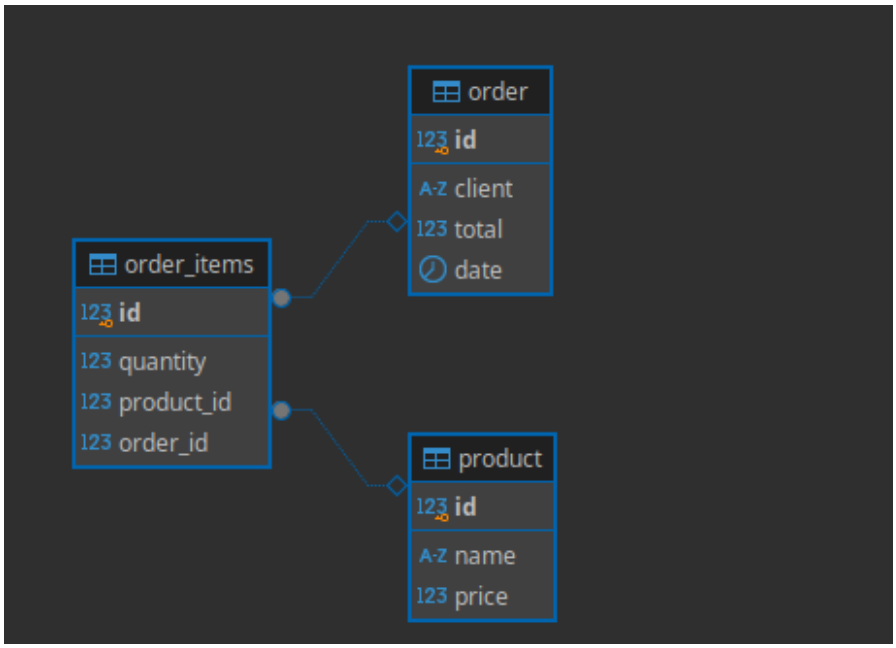
2.4 PostgreSQL

O PostgreSQL é um dos bancos de dados relacionais mais avançados e amplamente utilizado no mundo. Criado em 1986 no Departamento de Ciência da Computação da

Universidade da Califórnia, Berkeley, sob a liderança de Michael Stonebraker, o PostgreSQL foi projetado inicialmente como um sistema de pesquisa acadêmica, mas evoluiu para um sistema de gerenciamento de banco de dados robusto e de código aberto (Stonebraker, 1996).

Conhecido por sua eficiência, o PostgreSQL utiliza índices como B-tree, Hash e GIN, otimizando pesquisas e garantindo respostas rápidas, mesmo em grandes volumes de dados (PostgreSQL Global Development Group, 2023). Essa característica é ilustrada na **Figura 1**, que apresenta um exemplo de diagrama SQL.

Figura 1: Exemplo de diagrama SQL.



Fonte: Elaborada pelo autor

No mercado, o PostgreSQL se destaca por sua confiabilidade. Segundo o Engines (2023), é o quarto banco mais popular globalmente, com cerca de 10% de participação de mercado. Sua confiabilidade e capacidade de lidar com grandes volumes de dados o tornam uma escolha popular para setores que exigem conformidade com padrões SQL e alta extensibilidade.

2.5 MongoDB

O MongoDB, inicialmente desenvolvido pela 10gen (MongoDB Inc.) em 2007 e lançado ao público em fevereiro de 2009, é um banco de dados NoSQL orientado a documentos que utiliza documentos semelhantes ao JSON com esquemas opcionais. Um exemplo desses documentos é apresentado na **Figura 2**.

Figura 2: Exemplo de documento JSON utilizado pelo mongo.

Fonte: Elaborada pelo autor

O MongoDB é eficiente em leituras e consultas rápidas, destacando-se em grandes volumes de dados distribuídos, graças ao sharding e suporte a índices avançados, como compostos e geoespaciais (Stonebraker, 2010). Ele é o banco NoSQL mais utilizado, ocupando a quinta posição no ranking DB-Engines (2023), com 8% do mercado. Empresas como Uber, Lyft e eBay o adotam por sua capacidade de gerenciar grandes volumes com alta performance (MongoDB, 2023).

2.6 Teste de Performance e Desempenho em Banco de Dados

O desempenho de bancos de dados é essencial para aplicações que exigem rapidez e precisão na manipulação de dados. Segundo Menasce et al. (2004), testes de performance simulam cenários reais para avaliar como o sistema responde a diferentes níveis de demanda, permitindo identificar gargalos e mensurar o impacto de queries específicas em recursos como CPU e memória.

2.6.1 Importância dos Testes de Desempenho

Realizar testes de desempenho garante que o banco de dados consiga gerenciar operações intensivas de leitura e escrita sem comprometer a integridade dos dados e a velocidade de resposta, especialmente em ambientes de alta demanda.

2.6.2 Métodos de Teste de Performance

O teste de capacidade avalia o limite máximo de dados que um banco de dados pode

gerenciar eficientemente, identificando pontos de saturação e garantindo escalabilidade.

2.6.3 Critérios e Variáveis Utilizadas na Avaliação

Na avaliação do desempenho, as métricas analisadas foram diversas. O **tempo de resposta** mede o tempo necessário para executar consultas em cenários variados, como buscas completas, junções e agregações O **uso de recursos** foi analisado observando o consumo de CPU durante as operações, identificando gargalos e a eficiência dos sistemas em utilizar recursos computacionais (Elmasri & Navathe, 2010). Além disso, foram consideradas as **operações de entrada e saída (I/O)**, examinando o impacto da leitura e gravação em disco, representando a interação entre a CPU e o armazenamento de dados. Para o PostgreSQL, foi avaliado o impacto da **normalização** em consultas relacionais, como a eficiência no uso de índices avançados (ex.: B-Tree, GIN). Já no MongoDB, o foco foi uma busca utilizando documentos embutidos, analisando o desempenho em consultas que realizam combinações entre coleções utilizando identificadores (ObjectIDs).

3. REVISÃO SISTEMÁTICA

Para conduzir uma revisão sistemática que compare bancos de dados relacionais, como o PostgreSQL, e bancos de dados NoSQL, como o MongoDB, foi realizada uma busca por trabalhos que abordassem de forma qualitativa e quantitativa.

Uma das principais características de uma pesquisa de qualidade baseada em evidências é a realização de questionamentos práticos bem construídos, como cita o Center for Evidence Based Management (Management 2019).

A aplicação do método PICOC (População, Intervenção, Comparação, Resultado e Contexto) auxilia na formulação de perguntas de pesquisa claras e objetivas.

Tabela 1: PICOC

PICOC	Palavra-chave
População	Bancos de dados, Sistemas de Gerenciamento de Banco de Dados (SGBDs), PostgreSQL, MongoDB, NoSQL, Bancos de Dados Relacionais.
Intervenção	PostgreSQL, MongoDB
Comparação	PostgreSQL X MongoDB
Resultado	Avaliação de desempenho
Contexto	Aplicações em desenvolvimento de software

A partir do método PICOC, foi possível definir com clareza os objetivos que o trabalho busca atingir, bem como a pergunta que deve ser respondida. Dessa forma, a pesquisa foi pautada em efetuar um estudo comparativo de bancos de dados a fim de esclarecer uma diferença estatística entre eles.

3.1 Protocolo de Seleção de Trabalhos Relevantes

Para escolha dos trabalhos e estudos mais relevantes, foram realizadas buscas em plataformas gratuitas utilizando uma estratégia de filtro que seguia os seguintes critérios:

• Avaliação de qualidade de estudos primários: A análise foi realizada de forma qualitativa baseada na fonte de extração do material e na aplicação dos critérios de exclusão e inclusão de artigos.

• Estratégia de extração de informação: Foram extraídos os seguintes dados dos artigos encontrados: título, resumo, contexto e resultados;

• Busca: O foco foram artigos que tivessem ligação ao tema da pesquisa que estivessem, exclusivamente, em português ou inglês. A pesquisa foi restringida a partir da utilização de uma string de busca. Uma string de busca é utilizada para filtrar e pesquisar em uma biblioteca digital, que no nosso caso foi o google acadêmico, utilizando palavras-chaves relacionadas ao tema buscado. A string de busca utilizada para responder à pergunta desta pesquisa foi:

- “((PostgresSQL OR MongoDB) AND (‘desempenho’ or ‘performance) AND (‘leitura’ OR 'consulta’) AND (‘banco de dados’ OR ‘database’))”.

Tabela 2: Critérios de inclusão e exclusão

Critério	Descrição
Seleção de Fontes	Artigos científicos, periódicos e livros.
Palavras-Chaves	PostgreSQL, MongoDB, Banco de Dados, Desempenho, Leitura, Consulta, SQL, NoSQL
Idioma dos Estudos	Português e Inglês
Listagem das Fontes	Portal de Periódicos da CAPES, Google Acadêmico.
Critérios de Inclusão e Exclusão de Artigos	Materiais disponíveis na web, utilizando PostgreSQL e MongoDB, com conteúdo integral. Materiais duplicados; estudos que não abordem comparações práticas entre os bancos de dados.

3.2 Resultados da Busca

A revisão sistemática identificou inicialmente 80 estudos potenciais por meio das plataformas de busca utilizadas. Após a aplicação dos critérios de inclusão e exclusão, 30 estudos foram selecionados para análise mais detalhada. Destes, 15 artigos foram considerados altamente relevantes para o tema proposto, apresentando contribuições diretas para a comparação entre PostgreSQL e MongoDB em termos de desempenho e consumo de recursos.

Esses 15 artigos abordaram uma variedade de cenários, incluindo operações de leitura, consultas complexas, consumo de CPU e memória, e aplicabilidade em diferentes contextos, como sistemas de gestão de dados e aplicações de grande escala. A análise desses trabalhos fornece a base para discutir os resultados e identificar padrões de desempenho entre as tecnologias comparadas.

3.3 Protocolo de Seleção de Trabalhos Relevantes

[Sharma et al., 2019]: No estudo "Performance Evaluation of NoSQL and SQL Databases for Big Data Applications", os autores avaliam o desempenho de bancos de dados relacionais e não relacionais, destacando PostgreSQL e MongoDB. O trabalho apresenta uma abordagem abrangente, considerando diferentes tamanhos de datasets e analisando métricas fundamentais como tempo de resposta e consumo de CPU em operações de leitura e escrita

[Lee & Kim, 2021]: No trabalho "Performance Comparison of MongoDB and PostgreSQL for Sensor Data Management", os autores trazem uma abordagem prática e focada em cenários específicos, analisando o desempenho de PostgreSQL e MongoDB em operações comuns de leitura e escrita, com atenção especial ao uso de CPU e memória. A inclusão deste estudo foi motivada pela replicabilidade dos experimentos e pela relevância de suas conclusões para ambientes de aplicações modernas.

[Makris et al., 2020]: O artigo "A Comparative Study of SQL and NoSQL Databases" apresenta uma análise detalhada entre PostgreSQL e MongoDB com foco em tempo de resposta e consumo de recursos. A escolha deste estudo se justifica pela profundidade da análise e pela robustez dos experimentos realizados, que destacam características importantes para aplicações de grande escala

Diversos estudos trazem abordagens distintas para comparar PostgreSQL e MongoDB, utilizando métricas como tempo de resposta, consumo de CPU e uso de memória. Estas características são frequentemente analisadas como indicadores de eficiência em diferentes cenários. Neste trabalho, o foco será realizar uma análise estatística detalhada dessas métricas, em ambientes isolados, garantindo que as medições sejam precisas e sem interferências de outros programas. O objetivo é identificar diferenças significativas entre as tecnologias e compreender como suas características arquiteturais influenciam no desempenho.

4. METODOLOGIA

Este capítulo descreve as etapas e processos utilizados para configurar o ambiente de testes e conduzir as avaliações de performance entre diferentes bancos de dados. A metodologia foi estruturada para garantir resultados precisos e comparáveis, com foco na configuração do ambiente, modelagem dos dados, geração de dados de teste e monitoramento de métricas de desempenho durante os testes.

4.1 Isolamento do Ambiente

Para garantir precisão nos testes, foram utilizadas duas máquinas virtuais (VMs) isoladas no plano Free Tier da Amazon Web Services (AWS), uma dedicada ao PostgreSQL e outra ao MongoDB. Ambas rodavam Ubuntu 22.04 LTS, com 1 GB de RAM, 30 GB de SSD e uma vCPU.

Esta abordagem é alinhada à recomendação de Lee & Kim (2021), que destacaram a necessidade de ambientes controlados para análises comparativas confiáveis.

4.2 Configuração das Máquinas com Ansible

A configuração das máquinas foi automatizada utilizando o Ansible, uma ferramenta de automação que simplifica o gerenciamento de configurações e infraestrutura. Para isso, foram criados playbooks, que são scripts contendo as instruções para configurar os ambientes de forma padronizada. Foram desenvolvidos dois playbooks: um para instalar e configurar o PostgreSQL e outro para o MongoDB, incluindo todas as dependências necessárias para o funcionamento adequado de cada banco. O uso do Ansible garantiu consistência nas configurações, redução de erros manuais e agilidade para ajustes ou reconfigurações (Geerling, 2023).

4.3 Modelagem dos Dados

Para a modelagem dos dados, foi feito um caso mais próximo ao mundo real e seguindo as características de cada banco de dados.

4.3.1 PostgreSQL

Seguiu-se o modelo relacional com estrutura normalizada, garantindo consistência e integridade referencial, conforme descrito por Sharma et al. (2019). Foram criadas tabelas para ordens de compra, produtos e itens de ordem, onde as relações são estabelecidas por chaves estrangeiras. Os testes foram realizados em três escalas de dados: produtos com 1.000 registros fixos; ordens de compra com 1.000, 20.000 e 30.000 registros; e itens de ordem com quantidade variando entre 1 e 10 produtos por ordem.

4.3.2 MongoDB

No MongoDB, a modelagem dos dados foi feita utilizando arrays de IDs para associar ordens de compra aos produtos, uma prática recomendada por Makris et al. (2020). Em vez de armazenar diretamente os dados dos produtos dentro de cada documento de ordem, apenas os IDs dos produtos foram incluídos, enquanto os detalhes completos foram mantidos separadamente na coleção product. Os testes envolveram produtos com 1.000 registros e ordens de compra com 1.000, 10.000 e 30.000 registros, contendo arrays de 1 a 10 produtos embutidos em cada documento.

4.3.3 Geração e Inserção de Dados

A geração e inserção de dados foram realizadas de forma automatizada por meio de scripts desenvolvidos em Node.js com TypeScript. Esses scripts simularam cenários reais, gerando grandes volumes de dados aleatórios estruturados para o PostgreSQL e dados semiestruturados para o MongoDB.

No caso do PostgreSQL, os scripts criaram tabelas normalizadas para as entidades products, orders e order_items, seguindo regras de integridade referencial. Para o MongoDB, os dados foram organizados em coleções, com documentos que incorporam informações de ordens e itens relacionados diretamente em um único registro.

4.4 Execução dos Testes de Performance

Após a inserção dos dados, os scripts em Node.js também realizaram operações de leitura nos bancos de dados, simulando consultas comuns, como a recuperação de ordens de compra e seus respectivos produtos. Essas operações permitiram comparar:

- **Tempo de Resposta:** O tempo necessário para inserir e consultar grandes volumes de dados foi monitorado em tempo real. A medição foi feita diretamente nos scripts, registrando o tempo inicial e final de cada operação para calcular a duração total.
- **Uso de CPU:** O consumo de CPU foi monitorado nas VMs dedicadas por meio da biblioteca os.js, fornecida pelo Node.js. Os valores de utilização foram capturados em intervalos regulares durante cada operação.
- **Memória Utilizada:** A memória consumida pelas máquinas durante as operações foi registrada, verificando o impacto das consultas.

Essas métricas foram extraídas com a biblioteca *os.js* do Node.js, verificando o consumo de CPU, memória e tempo de resposta no momento de cada consulta.

4.5 Cálculo de CPU, Memória e Tempo de Resposta

Para mensurar o desempenho dos bancos de dados durante os testes, foram implementadas métricas específicas para uso de CPU, consumo de memória e tempo de resposta. A coleta das métricas é iniciada somente após a conexão com o banco de dados ser estabelecida, garantindo que os dados reflitam exclusivamente o impacto das operações realizadas e não o tempo necessário para estabelecer a conexão.

4.5.1 Cálculo do Uso de CPU

O uso de CPU foi monitorado para avaliar a carga de processamento de cada operação. Optou-se por process.cpuUsage() do Node.js, realizando a captura antes e depois de cada consulta e a diferença ajustada pela contagem de núcleos da CPU, foi possível calcular de forma precisa a porcentagem de CPU utilizada. Essa porcentagem foi obtida comparando o tempo de CPU com o tempo total de execução, garantindo uma visão mais clara do impacto.

4.5.3 Cálculo do Consumo de Memória

Para medir o impacto na memória, foi utilizada process.memoryUsage().rss, que cap- tura a memória residente usada pelo processo com alta precisão. Essa escolha permite avaliar o consumo real de memória antes e depois de cada consulta, facilitando a análise do impacto das operações.

4.5.3 Cálculo do Tempo de Resposta

O tempo de resposta foi medido com a função process.hrtime() do Node.js, que oferece alta precisão, especialmente em medições de curta duração.

4.6 Cenários de Testes

A execução dos testes seguiu o modelo de cenários definidos por Sharma et al. (2019), que utilizou consultas padronizadas para comparar o desempenho de bancos de dados em diferentes volumes de dados. O cenário foi composto por 30 observações, distribuídas igualmente entre três tamanhos de datasets: 10 observações com um dataset de 1.000 registros, 10 com um dataset de 10.000 registros e 10 com um dataset de 30.000 registros.

A abordagem adotada consistiu em realizar consultas específicas que refletem a identidade de cada banco de dados. No MongoDB, foi feita uma busca completa pelas ordens de compra, contendo os IDs dos produtos associados e suas respectivas quantidades diretamente embutidas nos documentos. Já no PostgreSQL, foi utilizado um JOIN para recuperar a quantidade de produtos de cada ordem de compra, combinando informações de tabelas normalizadas. Essa configuração permite comparar o desempenho dos bancos de dados em cenários que destacam suas características arquiteturais.

Essa estrutura permitiu avaliar de forma consistente aspectos como tempo de execução,

uso de CPU e consumo de memória, garantindo representatividade estatística e atendendo aos critérios mínimos para a aplicação de testes paramétricos e não paramétricos.

4.7 Metodologia Estatística

A avaliação de desempenho entre os bancos de dados PostgreSQL e MongoDB seguiu uma abordagem estatística rigorosa, utilizando testes estatísticos paramétricos e não paramétricos, escolhidos com base nas características das amostras e no delineamento experimental.

Análise de Normalidade

Foi aplicado o Teste de Shapiro-Wilk (Shapiro & Wilk, 1965) para verificar se os dados coletados (uso de CPU, memória e tempo de resposta) seguem uma distribuição normal. O teste utiliza dois indicadores principais. A **estatística W** mede o quanto os dados se aproximam de uma distribuição normal. Valores próximos de 1 indicam forte aderência à normalidade. Já o **valor p** representa a probabilidade de que os dados observados sigam uma distribuição normal, dado o valor de W. É usado para decidir se aceitamos ou rejeitamos a hipótese nula.

As hipóteses consideradas foram:

- H_0 (Hipótese Nula): Os dados seguem uma distribuição normal.
- H_1 (Hipótese Alternativa): Os dados não seguem uma distribuição normal.

Critério de Decisão

- $p \geq 0{,}05$: Aceita-se H_0 (os dados seguem uma distribuição normal).
- $p < 0{,}05$: Rejeita-se H_0 (os dados não seguem uma distribuição normal).

Testes Estatísticos Aplicados

Com base nos resultados de normalidade e na homogeneidade das variâncias (analisada pelo Teste de Levene), os seguintes métodos estatísticos foram empregados:

Para Dados Normais e Homocedásticos: O Teste T (Student, 1908) foi empregado para comparar as médias entre PostgreSQL e MongoDB em métricas que apresentaram normalidade e variâncias homogêneas. O valor T representa a estatística do teste, calculada com base na diferença entre as médias dos grupos, levando em conta a variabilidade e o tamanho das amostras. Valores absolutos

mais altos de T indicam maior evidência de diferença entre as médias. As hipóteses adotadas foram:

- H_0: Não há diferença significativa entre as médias de desempenho dos bancos.
- H_1: Há diferença significativa entre as médias de desempenho.

Critério de Decisão:

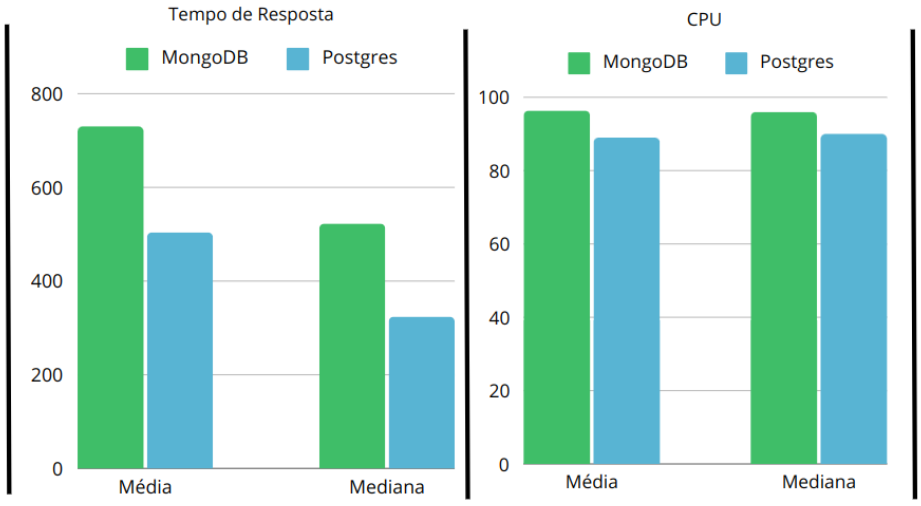
- $p \geq 0,05$: Não rejeita-se H_0 (as médias são estatisticamente iguais).
- $p < 0,05$: Rejeita-se H_0 (as médias são estatisticamente diferentes).

Para Dados Não Normais ou Heterocedásticos: Utilizou-se o Teste de Mann-Whitney, uma alternativa não paramétrica ao Teste t. Em vez de comparar médias, o Mann-Whitney analisa as distribuições das amostras, ordenando os valores e calculando a estatística U. O valor U reflete a soma das posições (ranks) de uma amostra na distribuição combinada. Valores extremos de U indicam maior separação entre as distribuições dos grupos, sugerindo diferenças significativas entre as amostras.

5. RESULTADOS E UTILIZAÇÃO

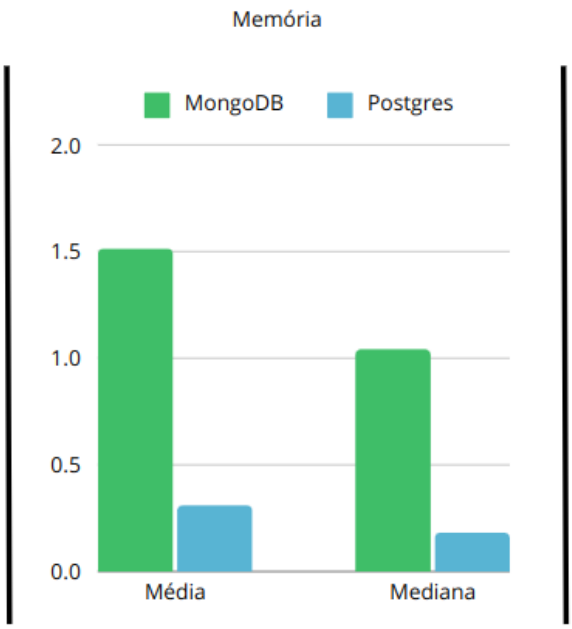
5.1 Resumo completo das métricas

Figura 3 - Média e mediana da CPU e Tempo de Reposta



Fonte: Autor

Figura 4 - Média e mediana da memória



Fonte: Autor

5.1.1 Resultado da Normalidade (Shapiro-wilk)

Tabela 3: Normalidade Shapiro-wilk

Métrica	***Banco de Dados***	***Estatística W***	***Valor p***	***Normalidade***
Tempo de Execução	*MongoDB*	0.8122	0.00015	*Não*
	Postgres	0.8094	0.000098	*Não*
Uso de CPU	*MongoDB*	0.9612	0.7993	*Sim*
	Postgres	0.9833	0.9805	*Sim*
Uso de Memória	*MongoDB*	0.774	0.0000228	*Não*
	Postgres	0.786	0.0000364	*Não*

Os resultados do Teste de Shapiro-Wilk indicaram que a métrica **Uso de CPU** segue uma distribuição normal, com variação estável entre os diferentes tamanhos dos datasets (1.000, 10.000 e 30.000 registros). Por isso, foi aplicado o Teste T para comparar médias.

Já as métricas **Tempo de Execução** e **Uso de Memória** não apresentaram normalidade de acordo com o aumento do dataset, que resultou em maior variabilidade e dispersão nos dados.

Para essas métricas, foi preciso utilizar o Teste de Mann-Whitney, adequado para dados não normais e comparações de distribuições.

5.1.2 Teste T (Para métricas normais)

Tabela 4: Resultado Teste-T

Métricas	***Estatística t***	***Valor p***	***Interpretação***
Uso de CPU	*6,6566*	$3{,}02\times10^{-6}$	Diferença significativa

Ao analiar as métricas de consumo de CPU, a estatística $t = 6{,}6566$ e o valor $p = 3{,}02\times10^{-6}$ indicam uma diferença significativa entre PostgreSQL e MongoDB, com o PostgreSQL consumindo menos CPU. Isso se deve à sua arquitetura projetada para otimizar operações de consulta e execução, reduzindo o uso de recursos por meio de planejamento eficiente e execução focada em desempenho. Por outro lado, o MongoDB, priorizando flexibilidade e escalabilidade, pode apresentar maior uso de CPU em operações com grandes volumes de dados ou agregações complexas

5.1.3 Teste de Mann-Whitney (Para métricas Não normais)

Tabela 5: Resultado Mann-Whitney

Métricas	***Estatística U***	***Valor p***	***Interpretação***
Tempo de Resposta	*283*	0,0138	Diferença Significativa
Uso de Memória	*105*	0,0005	Diferença Significativa

Tempo de Resposta: O PostgreSQL também se destaca em tempo de resposta. O valor $U = 283$ com $p = 0{,}0138$ indica uma diferença significativa entre as distribuições de tempo de resposta entre os bancos. O PostgreSQL apresenta tempos de resposta menores devido à sua arquitetura otimizada para consultas rápidas e operações transacionais eficientes, permitindo um desempenho superior mesmo com grandes volumes de dados. Por outro lado, o MongoDB sofre maiores atrasos a medida que o tamanho dos dados foi crescendo.

Uso de Memória: O valor $U = 105$ com $p = 0{,}0005$ também aponta uma diferença significativa entre as distribuições de uso de memória dos dois bancos de dados com o PostgreSQL utilizando menos memória. Essa eficiência se dá pelo gerenciamento otimizado de buffers e cache

no PostgreSQL, que reduz a necessidade de alocação de memória adicional durante a execução das consultas. Em contraste, o MongoDB, projetado para maior escalabilidade e flexibilidade, pode demandar mais memória em operações com grandes conjuntos de dados e agregações complexas.

6. CONSIDERAÇÕES FINAIS

Este projeto teve como objetivo realizar uma análise estatística comparativa entre dois dos bancos de dados mais amplamente utilizados, PostgreSQL e MongoDB, fornecendo evidências significativas para embasar decisões informadas no uso de cada tecnologia.

Os resultados demonstraram que o PostgreSQL se destacou em termos de eficiência no tempo de resposta, consumo de CPU e uso de memória em cenários específicos de testes. Sua arquitetura é altamente otimizada para consultas rápidas e operações transacionais robustas. Por outro lado, o MongoDB, embora menos eficiente nesses aspectos de desempenho, apresenta maior flexibilidade e escalabilidade, sendo particularmente adequado para aplicações com estruturas de dados dinâmicas ou em constante evolução.

A escolha do banco de dados ideal depende diretamente do contexto e das necessidades específicas do projeto. Aplicações que exigem alta consistência, consultas complexas e operações transacionais robustas tendem a se beneficiar do PostgreSQL. Em contraste, projetos que demandam agilidade na implementação, menor esforço de modelagem inicial e escalabilidade horizontal podem se favorecer com o MongoDB.

Durante o desenvolvimento deste trabalho, constatou-se que o PostgreSQL, embora eficiente, requer maior cuidado na modelagem e manutenção dos dados, devido à sua estrutura mais rigorosa. Já o MongoDB simplifica significativamente esse processo ao permitir estruturas mais flexíveis, o que pode ser vantajoso em projetos que priorizam agilidade e adaptabilidade.

Portanto, a decisão sobre qual banco de dados utilizar deve considerar não apenas os resultados de desempenho, mas também aspectos como a natureza dos dados, a complexidade das operações, o volume de informações e os recursos disponíveis para manutenção e suporte. Este estudo fornece uma base sólida para avaliar esses critérios e contribuir para decisões estratégicas fundamentadas.

REFERÊNCIAS

ALMEIDA, R.; SANTOS, J.; NOGUEIRA, T. Comparative Performance Analysis of Relational and Non-Relational Databases. *Journal of Information Systems*, v. 25, n. 4, p. 49–63, 2019.
BANKER, K. *MongoDB in Action*. 2nd ed. Manning Publications, 2011. Acesso em 03 de dez.
CATTELL, R. (2011) Scalable SQL and NoSQL Data Stores. Acesso em 03 dez 2024.
CHODOROW, K. *MongoDB: The Definitive Guide*. O'Reilly Media, 2013. Acesso em 03 dez.
CODD, E. F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, v. 13, n. 6, p. 377–387, 1970.
COMER, D. The Ubiquitous B-Tree. *ACM Computing Surveys*, v. 11, n. 2, p. 121–137, 1979.
DATE, C. J. *An Introduction to Database Systems*. 8th ed. Addison-Wesley, 2004.
DB-Engines. DB-Engines Ranking. 2023. Disponível em: https://db-engines.com. Acesso em: 03 dez. 2024.
ELMASRI, R.; NAVATHE, S. B. *Fundamentals of Database Systems*. 6th ed. Addison-Wesley, 2010.
GEERLING, J. *Ansible for DevOps*. 2nd ed. 2023.
KNUTH, D. E. *The Art of Computer Programming*. 3rd ed. Addison-Wesley, 1998.
LEE J., & KIM, H. (2021). Performance Comparison of MongoDB and PostgreSQL for Sensor Data Management. Journal of Sensor and Actuator Networks. Acesso em 03 dez.
MAKRIS, C.; TSERPES, K.; PANTZIOU, G. A Comparative Study of SQL and NoSQL Databases. *IEEE Transactions on Big Data*, v. 6, n. 3, p. 556–571, 2020.
MANAGEMENT, C. E. B. (2019). What is a picoc? https://cebma.org/faq/what-is-apicoc/ Acesso em 11 dez 2024.
MANYIKA, J.; et al. *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. McKinsey Global Institute, 2011. Disponível em: https://www.mckinsey.com. Acesso em: 03 dez. 2024.
MARKRIS, N. SMITH, J., & LEE, H. (2020). A Comparative Study of SQL and NoSQL Databases. Journal of Database Management, 31(2). Acesso em 03 dez.
MENASCE, et al. (2004) Performance by Design: Computer Capacity Planning by Example Acesso em 03 dez 2024.
MongoD5B Inc. *MongoDB Documentation*. Disponível em: https://www.mongodb.com/docs. 2023. Acesso em: 03 dez. 2024.
NASA. About James Webb Space Telescope. 2023. Disponível em: https://www.nasa.gov/content/about-jwst. Acesso em: 03 dez. 2024.
POSTGRESQL GLOBAL DEVELOPMENT GROUP. *Indexes*. Disponível em: https://www.postgresql.org/docs/current/indexes.html. 2023. Acesso em: 03 dez. 2024.
POSTGRESQL GLOBAL DEVELOPMENT GROUP. *Query Planning*. Disponível em: https://www.postgresql.org/docs/current/runtime-config-query.html. 2023. Acesso em: 03 dez. 2024.
SHAPIRO, S. S.; WILK, M. B. An Analysis of Variance Test for Normality. *Biometrika*, v. 52, n. 3–4, p. 591–611, 1965.
SHARMA, A., SINGH, R., & KUMAR, N. (2019). Performance Evaluation of NoSQL and SQL Databases for Big Data Applications. International Journal of Computer Applications. Acesso em 03 dez.

STONEBRAKER, M.; HELLERSTEIN, J. What Goes Around Comes Around. *Database Trends and Applications*, 2005.**STRAUCH, C.** *NoSQL Databases*. Stuttgart Media University, 2011. Disponível em: https://www.nosql-database.org. Acesso em: 03 dez. 2024.
STONEBRAKER, M. The PostgreSQL Project. *IEEE Data Engineering Bulletin*, v. 19, n. 2, p. 5–10, 1996.
STUDENT, (1908). The Probable Error of a Mean https://www.jstor.org/stable/2331554. Acesso em 11 dez 2024.
SUISSA, C. NoSQL History and Evolution. *NoSQL Matters Conference*, 2010. Disponível em: https://nosqlmatters.org. Acesso em: 03 dez. 2024.