TRABALHO DE CONCLUSÃO DE CURSO (TCC)

Sistemas de Informação / 2024

# ANÁLISE DE DESEMPENHO ENTRE BANCOS DE DADOS NÃO RELACIONAIS

NASCIMENTO, Anderson Prado[1]
OLIVEIRA, José Gabriel Alves[2]
JÚNIOR, Flávio Rubens Massaro[3]

Centro Universitário Hermínio Ometto – FHO, Araras – SP, Brasil

## Resumo

*Sistemas de gerenciamento de banco de dados não relacionais, conhecidos como Not Only SQL (NoSQL) surgiram como uma resposta à crescente demanda por soluções que pudessem lidar com o armazenamento de grandes volumes de dados, especialmente dados não estruturados e semiestruturados, que não se adequam corretamente ao modelo tradicional dos bancos de dados relacionais. Os sistemas NoSQL oferecem maior flexibilidade e escalabilidade em comparação aos modelos relacionais. Entretanto, os bancos NoSQL também apresentam desafios, como a falta de padronização, maior complexidade em transações, e dificuldades de integração com tecnologias existentes. Este projeto teve como objetivo realizar uma análise comparativa do desempenho entre diferentes bancos de dados NoSQL, com foco no tempo de resposta das operações de leitura e escrita. Para garantir melhores condições na execução de testes, foram utilizadas duas máquinas virtuais idênticas no VirtualBox, configuradas para minimizar interferências externas. O projeto em questão desenvolveu códigos para a simulação dos testes, que foram desenvolvidos na linguagem Ruby, com o suporte do Visual Studio Code. O desempenho de cada banco foi avaliado por meio de testes de execuções nos modelos de tabelas simples e complexas, alterando a quantidade de registros manipulados e a quantidade de threads. Os resultados foram apresentados através de tabelas e separados por cores, que são alteradas com base no desempenho em relação ao outro banco de dados, permitindo uma análise visual e facilitando a comparação. Os resultados também foram organizados com base no tipo de linguagem Structured Query Language (SQL). Através dos testes realizados, foi evidenciado que o banco de dados Apache Cassandra possui um desempenho superior em relação ao tempo de resposta nas operações de manipulação com grandes quantidades de dados, entretanto, o banco de dados MongoDB se sobressaiu nos testes nas operações de manipulação dos dados até 10.000 registros.*

*Palavras chave: NoSQL, tempo de resposta, análise comparativa, SQL, análise de desempenho*

## 1 Introdução

### 1.1 Contextualização

[1] FHO|UNIARARAS. Aluno do Curso de Sistemas de Informação, NASCIMENTO, andersonnascimento@alunos.uniararas.br
[2] FHO|UNIARARAS. Aluno do Curso de Sistemas de Informação, OLIVEIRA, jgararas@alunos.uniararas.br
[3] FHO|UNIARARAS. Professor do Curso de Sistemas de Informação, JÚNIOR, frmassaro@uniararas.br

Com o aumento da geração de dados no cenário de mercado atual, o armazenamento eficiente e rápida disponibilidade de informações tornou-se uma das principais preocupações para empresas e organizações, em função do alto volume de dados que são armazenados e processados diariamente. Os bancos de dados relacionais, mesmo que amplamente utilizados, apresentam limitações em termos de escalabilidade e flexibilidade para lidar com dados não estruturados e semiestruturados. O surgimento dos bancos de dados *NoSQL* trouxe uma nova abordagem projetada para gerenciar grandes volumes de dados. Esses bancos foram desenvolvidos com o objetivo de suportar uma variedade de tipos de dados e oferecer operações mais rápidas em comparação aos bancos de dados relacionais, principalmente em cenários de leitura intensiva e escalabilidade horizontal. Dependendo do tipo de operação *SQL* realizada, os bancos *NoSQL* podem fornecer um desempenho significativamente superior, tornando-os essenciais para aplicações modernas que requerem alta disponibilidade e baixa latência. Para garantir a consistência e adequação das informações, é necessário contar com um Sistema de Gerenciamento de Banco de Dados (SGBD) capaz de integrar e gerenciar esses dados de maneira eficiente e consistente.

### 1.2 Tema de Pesquisa

Efetuar uma análise comparativa do desempenho, com foco no tempo de resposta das operações *SQL* entre os bancos de dados *NoSQL* escolhidos.

### 1.3 Motivações e Justificativas

O projeto foi motivado pela alta demanda por armazenamento e processamento de grandes volumes de dados. Com o surgimento dos bancos de dados *NoSQL*, é válido comparar seu desempenho, especialmente no que diz respeito ao tempo de resposta.

### 1.4 Objetivos

#### 1.4.1 Objetivo Geral

Realizar uma análise comparativa do desempenho do tempo de resposta utilizando os bancos de dados *NoSQL* e as operações *SQL* selecionadas.

#### 1.4.2 Objetivos Específicos

- Analisar e selecionar os bancos de dados *NoSQL;*
- Analisar e selecionar as operações *SQL;*
- Desenvolver uma ferramenta para a análise do tempo de resposta das operações;
- Realizar os testes utilizando a ferramenta desenvolvida;
- Avaliar o tempo de resposta de cada operação;
- Apresentar os resultados de forma clara e objetiva.

## 2 Revisão Bibliográfica

### 2.1 Sistemas de Gerenciamento de Banco de Dados

Um SGBD é um conjunto de *softwares* responsáveis por definir, construir e manipular a estrutura de um banco de dados e gerenciar o acesso aos dados armazenados de forma eficiente e segura (CÔRTES, 2001).
Uma importante função de um SGBD, são as transações, as quais podemos defini-las como uma coleção de operações lógicas que são realizadas no SGBD. Essas operações são de extrema importância pois têm como objetivo garantir a consistência e integridade dos dados. (GARCIA; SOTTO, 2019; MATSUMOTO, 2006).

## 2.2 Dados

Segundo ELMASRI e NAVATHE (2005), dados são: “fatos que podem ser gravados e que possuem um significado implícito”. Podemos citar uma agenda telefônica, onde possui números, endereços e nomes, na qual essas informações são consideradas dados, podendo ser armazenados em um registro. Neste cenário, a agenda telefônica representa um banco de dados e as informações são os dados armazenados.

## 2.3 Tipos de dados

Dados estruturados são organizados em tabelas, com tipos de dados definidos, como números, textos curtos e datas. Os dados não estruturados ou semi-estruturados não se encaixam facilmente em uma tabela devido à sua complexidade ou tamanho, como por exemplo arquivos de áudio, vídeo e documentos de texto extensos, como o *JavaScript Object Notation*(*JSON*). Dados numéricos ou textuais podem ser considerados não estruturados ou semi-estruturados, dependendo de sua modelagem (AMAZON, 2024).

## 2.4 Índices

Os índices em uma base de dados funcionam como uma lista de ingredientes de uma receita. Em vez de ter que ler a receita inteira para achar o ingrediente que você quer, você olha diretamente na lista para encontrá-lo, facilitando a busca. Quando a base de dados faz uma busca sem usar o índice, é como se você tivesse que ler toda a receita, linha por linha, até achar o ingrediente desejado, causando um processamento mais lento em comparação a busca por índices, principalmente se a "receita" for muito longa (CHODOROW, 2013; NEERAJ, 2013).

## 2.5 Identificadores de registros

Os identificadores de registros (*IDs*) são informações únicas em cada tabela, podendo ser sequenciais, como 1, 2, 3, 4, ou baseados em informações específicas. Por exemplo, em uma tabela de pessoas, o Cadastro de Pessoas Físicas(CPF) pode ser utilizado como um identificador único, atendendo à condição de ser considerado um *ID*. Dependendo do banco de dados que é utilizado, existe a possibilidade de não existir os *IDs*, sendo substituídos por identificadores únicos universais(*UUIDs*), que agem da mesma forma que os *IDs* porém os *UUIDs* são significativamente maiores em questão de caracteres (ELMASRI; NAVATHE, 2005).

## 2.6 *Cache*

O *Cache* é um tipo de armazenamento temporário de dados que são acessados com frequência. O objetivo é guardar essas informações em um lugar rápido e de fácil acesso, para que, da próxima vez que os dados forem acessados, eles sejam retornados com maior agilidade e sem precisar buscar na origem dos dados (NEERAJ, 2013).

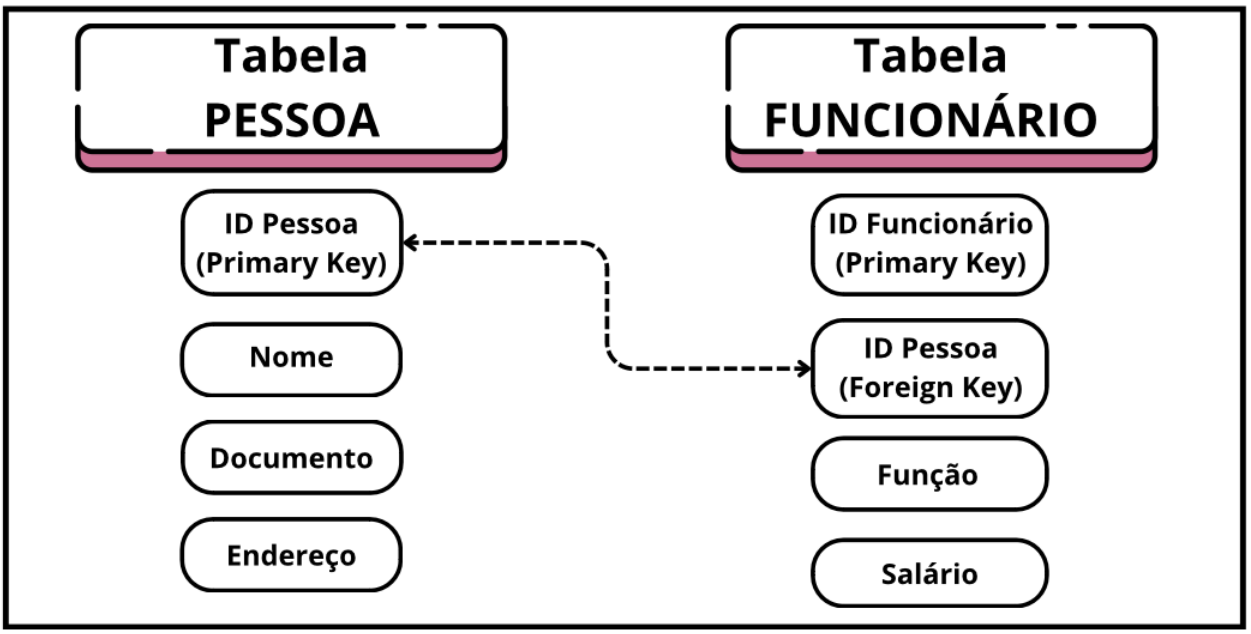
## 2.7 Linguagem *SQL*

Um SGBD, possui diversas transações, onde podemos classificá-las em tipos de linguagens, cada uma com seu objetivo e finalidade. Dentre essas linguagens, as mais utilizadas são as linguagens de manipulação de dados (*DML*) e a de consulta de dados (*DQL*). A linguagem *DML* é utilizada para realizar operações como inserção, alteração e remoção de dados, já a linguagem *DQL* é utilizada para realizar a consulta dos dados. Além disso, um SGBD possui outras linguagens importantes, como a linguagem de definição dos dados (*DDL*), utilizada na construção da estrutura do banco de dados, e a linguagem de controle de dados (*DCL*), realizando o controle de acessos e permissões dos usuários (ELMASRI; NAVATHE, 2005; ABRANTES, 2023).

## 2.8 Modelo Relacional

Os modelos relacionais em bancos de dados são fundamentados pela sua organização dos dados em tabelas, estruturadas em linhas e colunas. Cada tabela em um banco de dados relacional possui seus devidos atributos, sendo os mais importantes para estabelecer os relacionamentos entre as chaves primárias e as chaves estrangeiras. A chave primária é o identificador de cada tabela, não permitindo repetição na mesma tabela, geralmente referenciado como *ID*. A chave estrangeira, é o identificador que tem como objetivo manter o relacionamento entre as tabelas. A utilização dessas chaves facilita a organização e a integridade dos dados, permitindo consultas eficientes e relacionamentos consistentes entre diferentes tabelas. Neste exemplo, visualiza-se que a tabela funcionário, possui um relacionamento com a tabela pessoa por possuir a chave estrangeira (*Foreign Key*) “*ID* Pessoa” (ELMASRI; NAVATHE, 2005; BEHS, 2020; LAUDON; LAUDON, 1999).

Figura 1 - Modelo Relacional




Fonte: Elaborado pelos autores

Atualmente, existem vários SGBD Relacionais disponíveis no mercado. Na Figura 2, é apresentado um *ranking* dos mais utilizados, conforme evidenciado pelo *site DB-Engines* em março de 2024.

Figura 2 - Tabela *Ranking* banco de dados

418 systems in ranking, March 2024

| Rank Mar 2024 | Rank Feb 2024 | Rank Mar 2023 | DBMS | Database Model | Score Mar 2024 | Score Feb 2024 | Score Mar 2023 |
|---|---|---|---|---|---|---|---|
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1221.06 | -20.39 | -40.23 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1101.50 | -5.17 | -81.29 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 845.81 | -7.76 | -76.20 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 634.91 | +5.50 | +21.08 |
| 5. | 5. | 5. | MongoDB | Document, Multi-model | 424.53 | +4.18 | -34.25 |
| 6. | 6. | 6. | Redis | Key-value, Multi-model | 157.00 | -3.71 | -15.45 |
| 7. | 7. | ↑ 8. | Elasticsearch | Search engine, Multi-model | 134.79 | -0.95 | -4.28 |
| 8. | 8. | ↓ 7. | IBM Db2 | Relational, Multi-model | 127.75 | -4.47 | -15.17 |
| 9. | 9. | ↑ 11. | Snowflake | Relational | 125.38 | -2.07 | +10.98 |
| 10. | 10. | ↓ 9. | SQLite | Relational | 118.16 | +0.88 | -15.66 |

Fonte:*DB-Engines*, 2024

A partir deste *ranking*, observa-se que entre os dez bancos de dados abordados, sete são relacionais, destacando sua importância e predominância no mercado.

## 2.9 Modelo Não Relacional

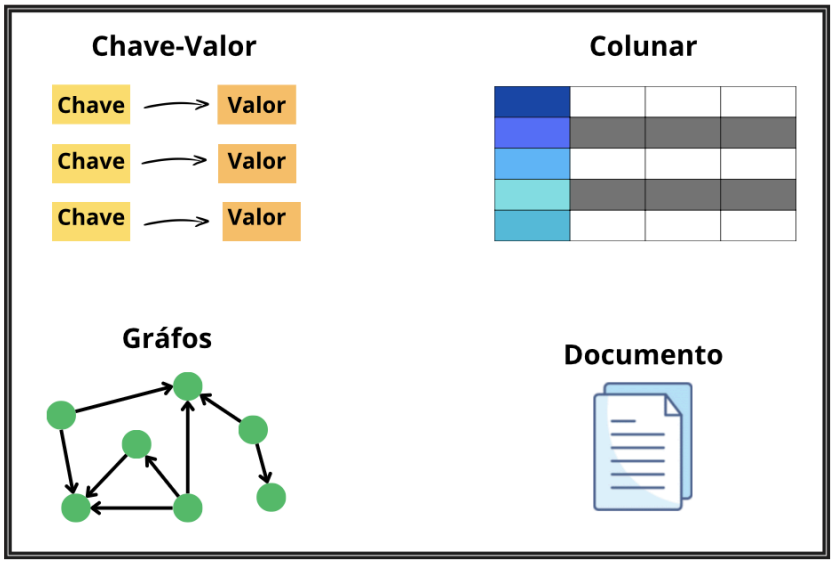
Os modelos não relacionais surgiram para suprir a necessidade de lidar com um grande volume de dados não estruturados e semiestruturados, destacando-se por uma alta

escalabilidade e pela sua capacidade de lidar com diferentes tipos de dados, possuindo uma fácil distribuição. O termo *NoSQL* foi introduzido primeiramente em 1998 pelo seu autor Carlo Strozz. Após um período de onze anos, o nome *NoSQL* foi reintroduzido em 2009 por Eric Evans em uma palestra sobre banco de dados *open source* e desde então o termo *NoSQL* predominou no mercado. Os modelos de banco de dados não relacionais apresentam características distintas dos modelos relacionais. Os bancos de dados não relacionais oferecem escalabilidade horizontal, apresentando uma vantagem para lidar com dados que crescem de forma exponencial. Outra característica é o esquema flexível, que permite uma estrutura de dados mais adaptável. Em contrapartida, essa flexibilidade pode comprometer a integridade e consistência dos dados, ao contrário do que ocorre nos bancos de dados relacionais (GARCIA; SOTTO, 2019; NASCIMENTO, 2014).

## 2.10 Tipos de bancos de dados não relacionais

Os bancos de dados não relacionais possuem subdivisões e classificações, cada um com uma forma específica de armazenamento e utilização. Através da Figura 3, é apresentado os modelos de bancos não relacionais abordados neste projeto (CARDOSO, 2012; IBM, 2024; MICROSOFT, 2024; OLIVEIRA, 2014).

Figura 3 - Bancos de dados *NoSQL*




Fonte: Elaborado pelos autores

**Chave-Valor** - Na Figura 4, é apresentado o conceito do banco de dados orientado a Chave-Valor, cujo o mesmo é considerado um modelo básico perante os demais, com objetivo em definir um valor para uma chave, possuindo uma escalabilidade predominante (CARDOSO, 2012; IBM, 2024; MICROSOFT, 2024; OLIVEIRA, 2014).

Figura 4 - Modelo *NoSQL* Chave-Valor

| Chave | Valor |
|---|---|
| Nome | Luis Oliveira |
| Idade | 21 |
| CPF | 99999999999 |
| Telefone | (00)000000000 |

Fonte: Elaborado pelos autores

**Colunar** - Mediante a Figura 5, observa-se o modelo colunar, consistindo na armazenagem de dados em colunas e linhas, permitindo uma pesquisa específica de um conjunto de dados. O destaque desse modelo é a pesquisa pela “família”, que consiste agrupar os dados em uma única coluna (CARDOSO, 2012; IBM, 2024; MICROSOFT, 2024; OLIVEIRA, 2014).

Figura 5 - Modelo *NoSQL* Colunar

| ID | Coluna_Familia: Identidade |
|---|---|
| 001 | Primeiro_Nome: Luis<br>Ultimo_Nome: Oliveira |
| 002 | Primeiro_Nome: Pedro<br>Ultimo_Nome: Santana<br>Suffixo: Jr. |
| 003 | Primeiro_Nome: Wilson<br>Ultimo_Nome: Santos<br>Titulo: Dr. |

| ID | Coluna_Familia: Info_Contato |
|---|---|
| 001 | Telefone: (00)000000000 |
| 002 | Telefone: (99)999999999<br>Email: pedrosantana@gmail.com |
| 003 | Email: wilsonsantos@gmail.com |

Fonte: Elaborado pelos autores

**Documentos** - Com base na Figura 6, o modelo orientado a documentos tem como objetivo armazenar arquivos em formato *JSON* e *XML*, localizando-os por um *ID* ou por outro tipo de registro. Também é possível armazenar textos sem nenhum tipo de formatação, permitindo realizar um filtro por valores armazenados (CARDOSO, 2012; IBM, 2024; MICROSOFT, 2024; OLIVEIRA, 2014).

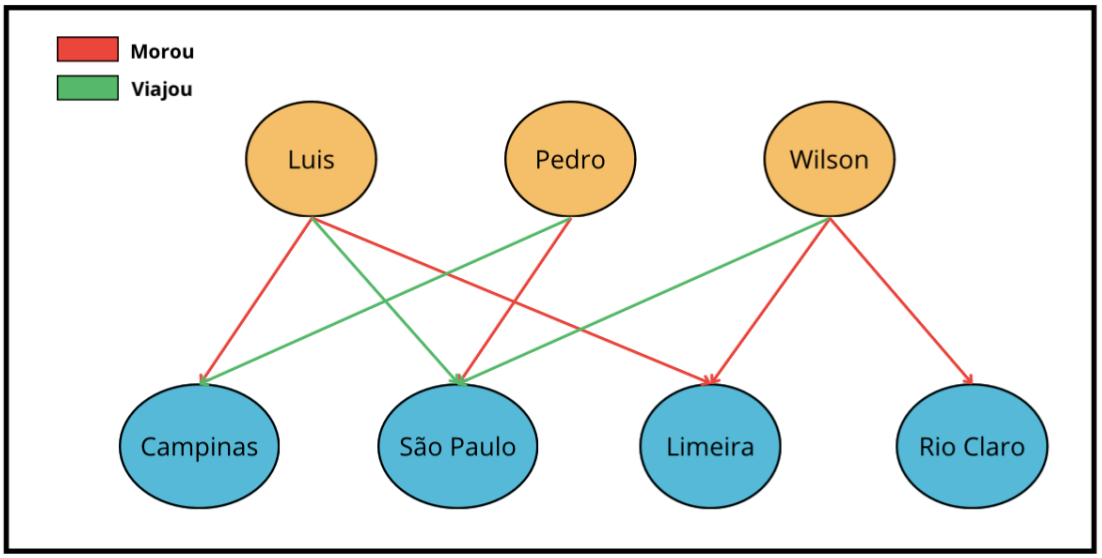
Figura 6 - Modelo *NoSQL* Documento

| Chave | Documento |
|---|---|
| 001 | { "id": 001, "Info_Contato": [ { "Telefone": 00000000000 } ] |
| 002 | { "id": 002, "Info_Contato": [ { "Telefone": 99999999999, "Email": pedrosantana@gmail.com } ] |
| 003 | { "id":003, "Info_Contato": [ { "Email": wilsonsantos@gmail.com } ] |

Fonte: Elaborado pelos autores

**Grafos** - O modelo orientado a grafos, apresentado na Figura 7, consiste em nós e arestas, projetado para armazenar e representar os dados via grafos, destacando-se pela sua eficiência em pesquisas devido às relações de entidades (CARDOSO, 2012; IBM, 2024; MICROSOFT, 2024; OLIVEIRA, 2014).

Figura 7. Modelo *NoSQL* Grafo




Fonte: Elaborado pelos autores

## 2.11 *Threads*

*Threads* são pequenas unidades de execução de um programa que podem ser processadas simultaneamente. Elas permitem que diferentes tarefas ocorram de forma simultânea dentro de um mesmo aplicativo ou serviço, distribuindo o trabalho de maneira eficiente. Esse recurso é especialmente utilizado em programas complexos, que precisam realizar múltiplas operações ao mesmo tempo, otimizando o desempenho e aproveitando melhor os recursos de *hardware* (MICROSOFT, 2024).

## 2.12 Virtualização

A virtualização é um processo que possibilita um computador compartilhar seus recursos de *hardware* com múltiplos ambientes digitais isolados e alocados. Cada ambiente virtualizado possui recursos previamente configurados, como memória, processamento e armazenamento (AMAZON, 2024).

## 2.13 *Benchmarking*

B*enchmarking* é utilizado de forma estratégica para melhorar o planejamento em busca de melhores resultados em processos e produtos. Dessa forma, empresas buscam a melhoria contínua e a redução de custos em seus produtos e processos para competir com seus concorrentes no mesmo nível de mercado. O *benchmarking* nada mais é do que estabelecer um padrão de referência para realizar análises de comparações. Quando aplicada de maneira correta e de acordo com a necessidade, essa ferramenta permite que a organização identifique, adapte e melhore as práticas em seus processos e produtos, tornando-se assim mais competitiva no mercado, independentemente de seu tamanho ou área de atuação, promovendo melhorias por meio do aprendizado organizacional mútuo. Essa técnica é valiosa para identificar lacunas de desempenho, definir metas de melhoria e promover mudanças necessárias para atingir um nível maior de maturidade nos processos organizacionais que nesta teoria é dividido em quatro etapas: planejamento do projeto, coleta de dados, análise de dados e adaptação/melhoria. No planejamento do projeto, é essencial identificar e determinar o objeto da análise comparativa a ser realizada. Na coleta de dados, é válido assegurar a consistência do ambiente e dos dados que serão inseridos na ferramenta para análise de desempenho. A análise de dados envolve a avaliação das diferenças nos resultados, mostrando o desempenho e identificando os pontos fortes ao gerar os dados manipulados. A fase final é a adaptação e implementação da melhoria contínua, onde as conclusões das etapas anteriores são usadas para realizar adaptações nas operações, visando melhorar o desempenho a longo prazo. Esses processos devem ser cuidadosamente trabalhados para garantir resultados satisfatórios para todas as partes envolvidas (JUNIOR, 2005; SILVA, 2021).

# 3 Trabalhos Relacionados

O projeto de FARAON, Renan, traz a importância da utilização do *benchmarking* na aplicação de comparar e avaliar o desempenho de sistemas de bancos de dados. Ele auxilia na escolha da tecnologia mais adequada para diferentes necessidades, podendo ser para bancos de dados relacionais ou não relacionais. O *TPC* (*Transaction Processing Performance Council*) é amplamente utilizado para simular cenários de transações e medir a eficiência do sistema. No caso dos bancos de dados *NoSQL*, o *YCSB (Yahoo!*

*Cloud Serving Benchmark* é a ferramenta mais comum, oferecendo uma análise detalhada de operações de leitura, escrita, atualização e exclusão, gerando estatísticas como tempo de execução e taxa de operações por segundo. Estudos comparativos demonstram que, embora bancos de dados relacionais, como o *MySQL*, tenham bom desempenho com grandes volumes de dados, eles podem ser menos eficientes em termos de espaço, enquanto bancos *NoSQL*, como o *Apache Cassandra*, podem apresentar quedas de desempenho com o aumento da carga de dados.

O estudo de GODOY, Felipe Pereira de, e PINHEIRO, Gabriel Benedito, teve como objetivo comparar o desempenho entre o banco de dados relacional *MySQL* e o banco de dados não relacional *MongoDB* (orientado a documentos). Os autores realizaram testes com diferentes volumes de transações e dados simultâneos, utilizando os métodos *DML* e *DQL* para realizarem os testes da pesquisa. A análise revelou que o *MongoDB* se destacou nas operações de *INSERT* e *DELETE*, enquanto o *MySQL* apresentou desempenho superior na operação de *UPDATE*. Em relação à operação *SELECT*, ambos os bancos apresentaram resultados semelhantes.

## 4 Metodologia

Dividida em 6 subseções, cada uma com seu objetivo específico de apresentar e detalhar a análise de desempenho entre os bancos de dados *NoSQL*.

### 4.1 - Identificação dos SGBD *NoSQL*

Para a identificação dos SGBD *NoSQL*, é demonstrado na Figura 8 alguns SGBD *NoSQL*, baseados no *ranking* do *site DB-Engines*.

Figura 8 - Tabelas de Modelos Não Relacionais

| Modelo Chave - Valor | Modelo Grafo |
|---|---|
| Redis | Neo4j |
| Memcached | Memgraph |
| RocksDB | Nebulagraph |
| Erospike | Janusgraph |
| **Modelo Documento** | **Modelo Colunar** |
| Mongo DB | Apache Cassandra |
| Couchbase | Hbase |
| CouchDB | Microsoft Azure |
| Google Cloud Firestone | Accumulo |

Fonte: Elaborado pelos autores

### 4.2 - Seleção dos SGBD para comparação e análises

Os critérios de seleção dos bancos de dados, demonstrados através da Figura 9, incluem alguns dados técnicos, como a capacidade de escalabilidade e a flexibilidade dos dados, levando em consideração sua popularidade no mercado atual através do *site DB-Engines*

em março de 2024, optando por opções gratuitas pela disponibilidade de acesso aos usuários. Os bancos de dados orientados a chave-valor e grafos, serão desconsiderados desta análise pois o modelo grafo tem como objetivo lidar com dados interconectados e consultas complexas de relacionamento. Já o modelo chave-valor é um modelo mais simples e otimizado para armazenamento rápido, sem foco em relacionamentos. O foco deste trabalho é apenas o armazenamento de dados, sem o intuito de relacionamentos complexos ou armazenamento temporário.

Figura 9 - Critérios de Seleção

| Tipo de Banco | Nome | Escalabilidade | Disponibilidade | Flexibilidade | Confiabilidade | Popularidade |
|---|---|---|---|---|---|---|
| Colunar | Apache Cassandra | ✓ | ✓ | | | ✓ |
| Colunar | Apache HBase | ✓ | | | | |
| Colunar | Microsoft Azure | ✓ | ✓ | | | |
| Colunar | Apache Accumulo | ✓ | | ✓ | | |
| Documentos | MongoDB | | ✓ | ✓ | ✓ | ✓ |
| Documentos | Couchbase | ✓ | ✓ | | ✓ | |
| Documentos | Firebase | | | ✓ | | |
| Documentos | CouchDB | | | ✓ | ✓ | |

Fonte: Elaborado pelos autores

Os bancos de dados que se encaixaram nos critérios de seleção citados na Figura 9 foram os bancos *MongoDB*, orientado a documento e o *Apache Cassandra*, orientado a colunas, onde o principal critério de escolha foi sua popularidade no mercado atual, retirados através da Figura 2 do *site DB-Engines* em março de 2024.

### 4.3 - Preparação do Ambiente

Optou-se por configurar dois ambientes virtuais idênticos, para que não haja nenhuma influência de *softwares* de cada banco e aplicativos externos que possam prejudicar os testes. Utilizou-se o *Oracle VM Virtualbox,* um *software* gratuito que oferece suporte para as ferramentas necessárias para a execução dos testes e para a virtualização de cada ambiente.

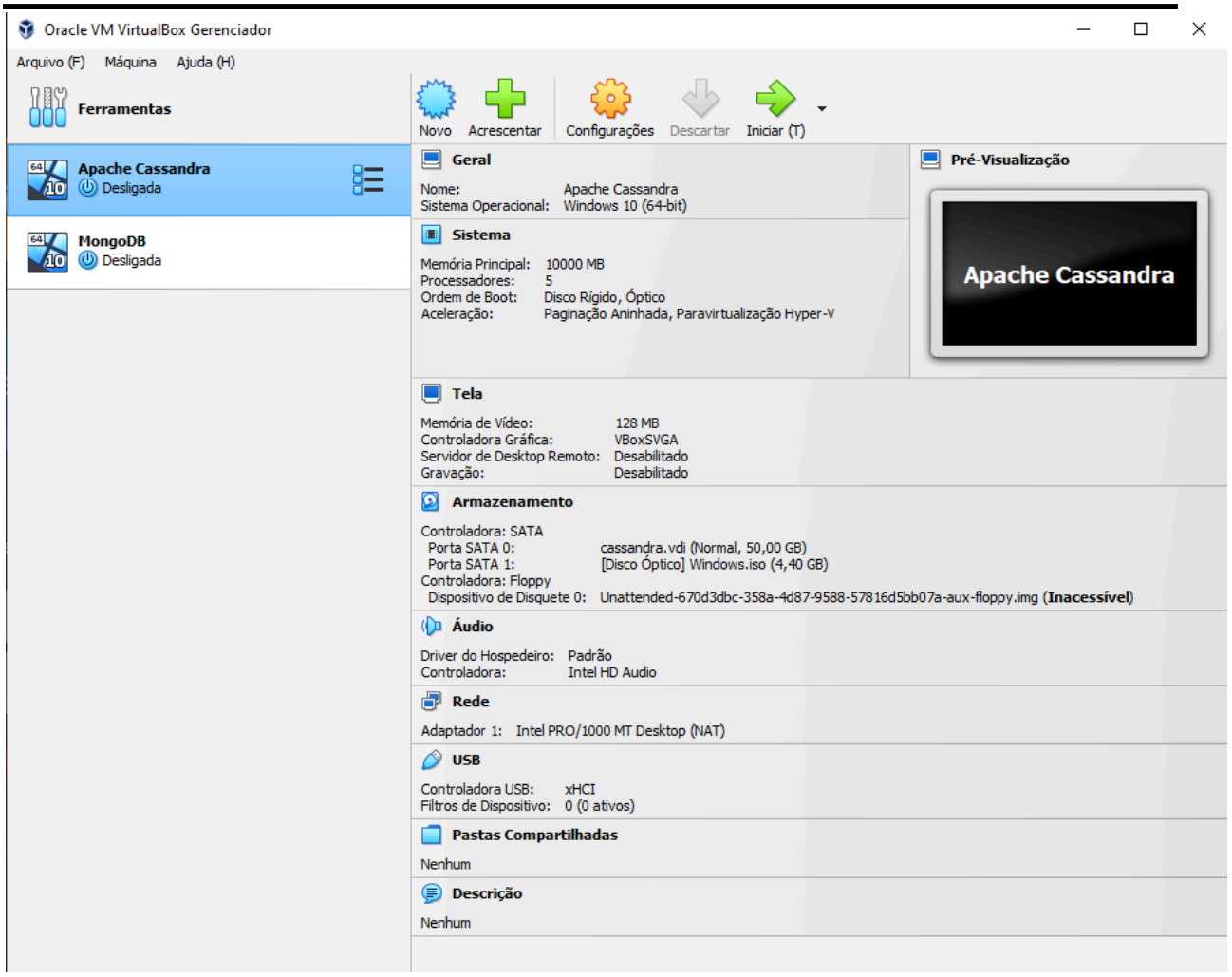
#### 4.3.1 - Máquina utilizada

Um *Desktop* utilizando o sistema operacional *Windows* 10 *Home*, um processador de 10ª Geração *Intel ® Core TM* i5-10400F, placa de vídeo dedicada *Nvidia RTX* 4060 com 8*GB*, 16*GB* de memória *RAM* e *SSD* de 512*GB*.

#### 4.3.2 - Ambiente Virtual de Teste

O ambiente virtual demonstrado através da Figura 10, conta com 10*GB* de *RAM*, 5 núcleos do processador e um *SSD* de 50*GB*. O sistema operacional utilizado foi o *Windows* 10 *home* devido sua utilização diária e estabilidade no mercado.

Figura 10 - Ambiente Virtual




Fonte: Elaborado pelos autores

### 4.3.3 - Modelagem da base de dados

A modelagem da base de dados é composta por dois modelos distintos: o modelo apresentado na Figura 11, é o modelo simples, que utiliza uma única tabela com seis campos, sendo eles de três tipos diferentes de dados(*string*, *int* e *boolean*).

Figura 11 - Modelo Simples

```
Tabela_01
{
  _id: <ObjectId>,
  campo01_Tabela01: string,
  campo02_Tabela01: string,
  campo03_Tabela01: int,
  campo04_Tabela01: int,
  campo05_Tabela01: boolean,
  campo06_Tabela01: boolean
}
```

Fonte: Elaborado pelos autores

O segundo modelo, representado através da Figura 12, é considerado complexo pois envolve cinco tabelas inter-relacionadas, com a adição gradual de campos em cada tabela, com o equilíbrio dos tipos de dados(*string, int* e *boolean*).

Figura 12 - Modelo Complexo

```
Tabela_01
{
  _id: <ObjectId>,
  campo01_Tabela01: string,
  campo02_Tabela01: int
},

Tabela_02
{
  _id: <ObjectId>,
  campo01_Tabela02: string,
  campo02_Tabela02: int,
  campo03_Tabela02: boolean
},

Tabela_03
{
  _id: <ObjectId>,
  campo01_Tabela03: string,
  campo02_Tabela03: string,
  campo03_Tabela03: int,
  campo04_Tabela03: int
},

Tabela_04
{
  _id: <ObjectId>,
  campo01_Tabela04: string,
  campo02_Tabela04: string,
  campo04_Tabela04: int,
  campo05_Tabela04: int,
  campo06_Tabela04: boolean
},

Tabela_05
{
  _id: <ObjectId>,
  campo01_Tabela05: string,
  campo02_Tabela05: string,
  campo03_Tabela05: int,
  campo04_Tabela05: int,
  campo05_Tabela05: boolean,
  campo06_Tabela05: boolean
}
```

Fonte: Elaborado pelos autores

Através desses modelos, foram realizados os testes de desempenho do tempo de resposta de cada operação *SQL*, utilizando as linguagens *DML* e *DQL*, onde será importado em cada banco de dados, seus respectivos modelos com uma base fictícia de dados.

### 4.4 - Desenvolvimento

Para o desenvolvimento, foi utilizado o ambiente *Visual Studio Code* e a linguagem de programação *Ruby*.

#### 4.4.1 - *Visual Studio Code*

O *Visual Studio Code* (*VSCode*) é um editor de código-fonte altamente reconhecido por sua interface intuitiva e customizável, que suporta muitas extensões e múltiplas linguagens de programação, facilitando a integração com sistemas de controle de versão, ferramentas de depuração eficientes e uma comunidade ativa com ampla documentação. Essas características tornam o *VSCode* uma poderosa ferramenta para desenvolvimento, facilitando a produtividade e a colaboração durante o projeto.

#### 4.4.2 - *Ruby*

A linguagem *Ruby* foi escolhida pela sua simplicidade e legibilidade, o que facilita o desenvolvimento e manutenção do código. Além disso, a linguagem *Ruby* possui uma vasta biblioteca de recursos e uma comunidade ativa, tornando-se a linguagem ideal para garantir a eficiência, flexibilidade e escalabilidade do projeto, garantindo a qualidade e facilidade de manutenção do projeto.

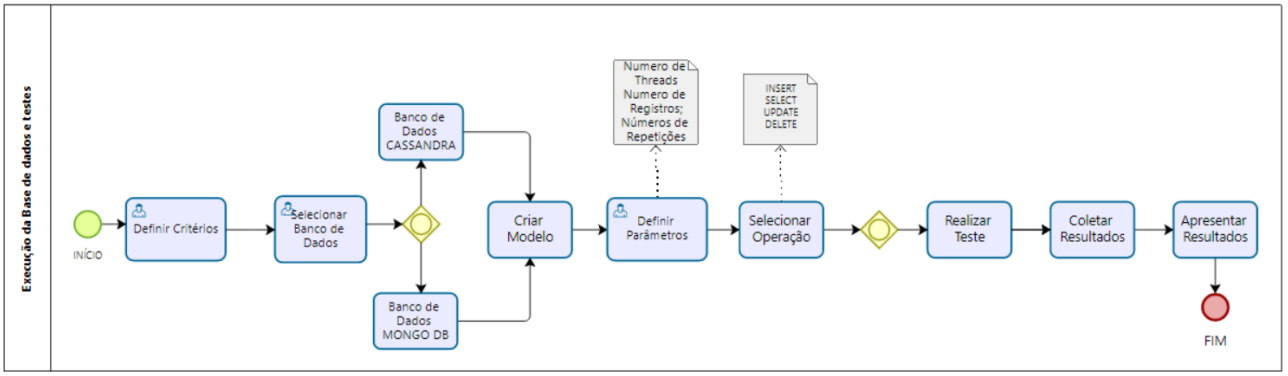
### 4.5 - Plano de validação

O plano de validação do projeto foi evidenciado através dos resultados obtidos, avaliando o tempo de resposta dos bancos de dados, foram criados programas que simulam operações *SQL*. Essas operações foram escolhidas por serem utilizadas no dia a dia dos usuários. Para as simulações, definimos alguns parâmetros, sendo eles: a quantidade de *threads* (como se fossem “braços” que processam dados ao mesmo tempo), configuradas em 1, 10, 50 e 100, e o número de registros de dados processados, variando entre 100, 1.000, 10.000 e 100.000. Cada *thread*, ou “braço”, manipula a quantidade de registros definida, repetindo cada operação 10 vezes para garantir resultados mais precisos. Foram utilizados dois modelos de tabelas para os testes, o modelo simples, com apenas uma tabela e tipos de dados definidos, e o modelo complexo, com várias tabelas e diferentes tipos de dados.

### 4.6 - Execução da base de dados e testes

Através da Figura 13, podemos visualizar o processo realizado durante a execução deste projeto.

Figura 13 - Processo




Fonte: Elaborado pelos autores

## 5 Resultados

Como resultados deste projeto, foram selecionados dois bancos de dados *NoSQL*(*MongoDB e Apache Cassandra*), para realizar os testes de desempenho em relação ao tempo de resposta, utilizando as linguagens *DML* e *DQL*. Os resultados são evidenciado através das cores com os resultados de cada operação, sendo elas: verde, caso tenha um tempo de resposta superior do que o outro banco de dados; vermelho, caso tenha um tempo de resposta inferior ao outro banco de dados e amarelo, caso o tempo de resposta sejam iguais em ambos os bancos de dados.

### 5.1 - *SELECT*

Ao utilizar a operação *SELECT* no modelo simples, observa-se, conforme a Figura 14, que o banco de dados *MongoDB* apresentou um melhor desempenho quando os testes são realizados com menores números de registros. Esses resultados indicam um desempenho superior do banco de dados *MongoDB* em relação ao banco de dados *Apache Cassandra*, em transações com baixo volume de dados.

Figura 14 - *SELECT* Simples

| ***SELECT - MongoDB*** | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,03 | 0,01 | 0,05 | 0,10 |
| 1K | 0,05 | 0,06 | 0,24 | 0,46 |
| 10K | 0,09 | 0,37 | 1,79 | 2,01 |
| 100K | 0,90 | 6,03 | 9,31 | 7,03 |

| ***SELECT - Apache Cassandra*** | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,03 | 0,03 | 0,08 | 0,10 |
| 1K | 0,05 | 0,08 | 0,29 | 0,65 |
| 10K | 0,20 | 0,57 | 1,65 | 1,98 |
| 100K | 0,18 | 0,60 | 2,65 | 5,33 |

Fonte: Elaborado pelos autores

Realizando os testes da operação *SELECT* no modelo complexo, observa-se, conforme a Figura 15, que o banco de dados *Apache Cassandra* apresentou um desempenho superior de forma geral, enquanto em algumas operações específicas, o banco de dados *MongoDB* obteve um melhor desempenho. Esses resultados indicam um desempenho

superior do banco de dados *Apache Cassandra* em relação ao banco de dados *MongoDB*, na maioria das transações realizadas utilizando a operação *SELECT* no modelo complexo.

Figura 15 - *SELECT* Complexo

| *SELECT - MongoDB* | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 0,08 | 0,67 | 0,32 | 0,55 |
| 1K | 0,22 | 0,29 | 1,29 | 2,42 |
| 10K | 0,51 | 2,24 | 10,30 | 14,75 |
| 100K | 3,67 | 33,92 | 39,45 | 22,54 |

| *SELECT - Apache Cassandra* | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 0,13 | 0,45 | 0,60 | 0,73 |
| 1K | 0,20 | 0,38 | 1,13 | 2,08 |
| 10K | 1,02 | 1,75 | 5,94 | 10,87 |
| 100K | 2,98 | 13,76 | 31,24 | 48,67 |

Fonte: Elaborado pelos autores

## 5.2 - *INSERT*

Na operação *INSERT* no modelo simples, conforme mostrado na Figura 16, o banco de dados *MongoDB* se destacou em operações com poucos registros. No entanto, a partir de 10.000 registros, o banco de dados *Apache Cassandra* apresentou um desempenho superior.

Figura 16 - *INSERT* Simples

| *INSERT- MongoDB* | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,27 | 0,78 | 3,12 | 5,65 |
| 1K | 1,10 | 7,66 | 20,30 | 27,94 |
| 10K | 12,59 | 78,59 | 87,70 | 104,74 |
| 100K | 125,56 | 352,07 | 663,65 | 769,33 |

| *INSERT- Apache Cassandra* | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,30 | 0,90 | 4,12 | 6,15 |
| 1K | 1,42 | 6,76 | 24,95 | 30,12 |
| 10K | 16,22 | 60,47 | 85,79 | 113,20 |
| 100K | 120,41 | 342,12 | 578,98 | 750,65 |

Fonte: Elaborado pelos autores

Na operação *INSERT* do modelo complexo, conforme mostrado na Figura 17, o banco de dados *Apache Cassandra* obteve um melhor desempenho na maioria dos testes, ficando com resultados inferiores ao banco de dados *MongoDB* quando a manipulação dos dados é realizada com 100 registros.

Figura 17 - *INSERT* Complexo

| *INSERT - MongoDB* | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 0,86 | 3,90 | 12,89 | 16,37 |
| 1K | 10,54 | 39,08 | 107,13 | 41,27 |
| 10K | 71,86 | 390,89 | 574,87 | 798,24 |
| 100K | 127,62 | 537,64 | 740,21 | 985,73 |

| *INSERT - Apache Cassandra* | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 1,03 | 4,78 | 17,32 | 23,86 |
| 1K | 14,73 | 37,29 | 73,65 | 123,81 |
| 10K | 67,18 | 276,49 | 538,43 | 734,94 |
| 100K | 146,5 | 424,78 | 689,25 | 887,31 |

Fonte: Elaborado pelos autores

## 5.3 - *UPDATE*

A operação *UPDATE* no modelo simples, observa-se através da Figura 18, que o banco de dados *MongoDB* se destacou na maior parte dos testes, obtendo resultados inferiores ao banco de dados *Apache Cassandra* na manipulação específica de 100.000 registros.

Figura 18 - *UPDATE* Simples

| ***UPDATE - MongoDB*** | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,32 | 0,84 | 3,26 | 5,86 |
| 1K | 2,92 | 8,16 | 18,11 | 29,97 |
| 10K | 103,38 | 79,43 | 116,99 | 262,90 |
| 100K | 283,93 | 334,04 | 665,02 | 815,78 |

| ***UPDATE - Apache Cassandra*** | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,54 | 1,89 | 5,47 | 8,12 |
| 1K | 3,21 | 6,23 | 19,15 | 26,98 |
| 10K | 112,2 | 130,87 | 164,10 | 252,23 |
| 100K | 273,32 | 315,63 | 593,47 | 781,20 |

Fonte: Elaborado pelos autores

Diante os testes da operação *UPDATE* no modelo complexo, conforme mostrado na Figura 19, os bancos de dados se comportam de forma flexível, uma vez que para algumas operações onde demanda exige pouco acesso simultâneo e poucos registros, o banco de dados *MongoDB* se destaca, entretanto, perde eficiência quando o número registros aumenta em comparação ao banco de dados *Apache Cassandra*.

Figura 19 - *UPDATE* Complexo

| ***UPDATE - MongoDB*** | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 3,10 | 4,05 | 17,35 | 14,34 |
| 1K | 33,77 | 40,77 | 126,20 | 74,06 |
| 10K | 141,29 | 400,97 | 729,46 | 1147,21 |
| 100K | 1265,20 | 4197,96 | 6354,84 | 8960,27 |

| ***UPDATE - Apache Cassandra*** | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 4,16 | 6,26 | 16,86 | 26,35 |
| 1K | 21,68 | 46,14 | 90,74 | 133,56 |
| 10K | 156,75 | 378,12 | 779,72 | 1113,68 |
| 100K | 724,89 | 3516,78 | 6542,54 | 8235,15 |

Fonte: Elaborado pelos autores

## 5.4 - *DELETE*

Na operação *DELETE* no modelo simples, conforme a Figura 20, é comprovado uma maior agilidade do banco de dados *MongoDB* na maioria dos testes realizados, entretanto, conforme o aumento da quantidade de registros, em certas ocasiões o banco de dados *Apache Cassandra* obteve um resultado superior.

Figura 20 - *DELETE* Simples

| ***DELETE - MongoDB*** | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,51 | 0,87 | 3,24 | 6,05 |
| 1K | 2,20 | 7,88 | 15,30 | 27,19 |
| 10K | 29,74 | 77,77 | 119,99 | 224,55 |
| 100K | 304,99 | 328,94 | 645,46 | 807,52 |

| ***DELETE - Apache Cassandra*** | | | | |
|---|---|---|---|---|
| Simples | 1 | 10 | 50 | 100 |
| 100 | 0,49 | 1,32 | 4,65 | 7,02 |
| 1K | 3,32 | 6,98 | 25,18 | 43,30 |
| 10K | 32,48 | 86,15 | 127,78 | 180,21 |
| 100K | 289,65 | 345,90 | 626,51 | 768,43 |

Fonte: Elaborado pelos autores

A operação *DELETE* no modelo complexo, observa-se através da Figura 21, que o banco de dados *Apache Cassandra* se destacou na maior parte dos testes quando a quantidade de registros manipulados é maior e em alguns casos isolados, o banco de dados *MongoDB* obteve resultados superiores.

Figura 21 - *DELETE* Complexo

| ***DELETE - MongoDB*** | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 1,33 | 4,25 | 18,94 | 13,91 |
| 1K | 10,78 | 41,19 | 127,43 | 60,20 |
| 10K | 292,24 | 523,65 | 838,76 | 1466,48 |
| 100K | 1538,46 | 3291,87 | 5392,84 | 4382,01 |

| ***DELETE - Apache Cassandra*** | | | | |
|---|---|---|---|---|
| Complexo | 1 | 10 | 50 | 100 |
| 100 | 1,64 | 4,11 | 23,87 | 38,26 |
| 1K | 13,08 | 46,89 | 105,36 | 193,24 |
| 10K | 257,65 | 503,26 | 798,54 | 1346,78 |
| 100K | 1325,14 | 2890,10 | 4938,98 | 6059,48 |

Fonte: Elaborado pelos autores

A validação dos resultados evidenciou que o banco de dados *Apache Cassandra* apresentou um desempenho superior em operações com grandes volumes de registros, quando o volume é superior a 10.000 registros. Em contrapartida, o *MongoDB* destacou-se em operações com menor quantidade de registros. Em certas operações, foi evidenciado uma diferença considerável do tempo de resposta entre bancos de dados comparados. Conforme os autores, os índices dos bancos de dados afetam fatores como tempo de resposta, consumo de memória e o uso do processador, influenciando diretamente o desempenho em diferentes cenários. O *cache* é outro fator que causou uma diferença nos testes, conforme evidenciado nas tabelas acima. O banco de dados *MongoDB*, ao manipular uma quantidade significativa de registros, como 10.000 ou mais, observa-se um decaimento no tempo de resposta devido ao limite de transações que o banco pode processar. Em contrapartida, o banco de dados *Apache Cassandra* não possui esse limite, permitindo a execução de um número maior de transações sem comprometer o desempenho. Outro aspecto importante é a geração automática de *IDs* sequenciais, no banco de dados *Apache Cassandra*, essa configuração não é suportada, sendo a alternativa a geração de *UUIDs*. Já no banco de dados *MongoDB*, a criação de *IDs* é feita de forma nativa e automática, facilitando o gerenciamento dos identificadores e consultas.

## 6 - Considerações Finais

Com o avanço dos *softwares* e o crescente volume de dados a serem armazenados diariamente, é ideal adotar um SGBD adequado para garantir que os dados sejam armazenados com consistência, integridade e acessibilidade. O trabalho desenvolvido realizou uma análise de desempenho, comparando o tempo de resposta das operações *SQL* em bancos de dados *NoSQL*. Foram utilizados dois modelos diferentes de SGBD *NoSQL*: o modelo orientado a documentos (*MongoDB*) e o modelo orientado a colunas (*Apache Cassandra*). A análise envolveu a aplicação de dois modelos de tabelas, simples e complexas, diferentes tipos e quantidade de dados e *threads.* Os resultados foram apresentados de forma clara em relação ao tempo de resposta para cada operação *SQL* deste projeto. Independentemente da operação *SQL*, o banco de dados *Apache Cassandra* se sobressai nos testes quando a quantidade de registros manipulados é

superior a 10.000 e na modelagem da tabela complexa, entretanto, o banco de dados *MongoDB* apresentou resultados superiores na manipulação de menores quantidades de dados na modelagem da tabela simples. Este projeto abre caminho para futuras vertentes, como a comparação com outros modelos não relacionais (quando disponíveis em um ambiente adequado) e a inclusão de modelos relacionais. Além disso, é possível realizar uma integração com *softwares* de monitoramento, como *Grafana* e *Zabbix*, para avaliar outros aspectos de desempenho, como por exemplo o uso de memória em cada operação *SQL*.

## Referências Bibliográficas

ABRANTES, Andressa. **Desvendando os Segredos do SQL e Dominando o SQL Server: Do Básico ao Avançado**. 2023. Disponível em: https://www.dio.me/articles/desvendando-os-segredos-do-sql-e-dominando-o-sql-server-do-basico-ao-avancado. Acesso em: 17 mar. 2024.

AMAZON. **O que é virtualização?**. 2024. Disponível em: https://aws.amazon.com/pt/what-is/virtualization/. Acesso em: 22 out. 2024.

AMAZON. **Qual é a diferença entre dados estruturados e dados não estruturados?**. 2024. Disponível em: https://aws.amazon.com/pt/compare/the-difference-between-structured-data-and-unstructured-data/. Acesso em: 22 out. 2024.

BEHS, Guilherme Nunes. **Análise de desempenho de banco de dados NoSQL**. 2020. p.14-16. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Feevale, Novo Hamburgo. 2020. Disponível em: https://tconline.feevale.br/tc/files/0001_5192.pdf. Acesso em: 06 abr. 2024.

CARDOSO, Ricardo Manuel Fonseca. **Bases de Dados NoSQL**. 2012. Dissertação (Mestrado em Engenharia de Informática e Especialização em Arquiteturas, Sistemas e Redes) - Instituto Superior de Engenharia do Porto, 2012. p. 18-24. Disponível em: https://core.ac.uk/download/pdf/47141116.pdf. Acesso em: 24 fev. 2024.

CHODOROW, Kristina. **MongoDB: The Definitive Guide**. 2. ed. CA, Sebastopol. O'Reilly, 2013. p. 341-353. Disponível em: https://pepa.holla.cz/wp-content/uploads/2016/07/MongoDB-The-Definitive-Guide-2nd-Edition.pdf. Acesso em: 27 nov. 2024.

CÔRTES, Sérgio da Costa; LUCENA, Carlos José Pereira. **Um Framework para construção de Sistemas de Banco de Dados Móvel com Regras Ativas**. 2001. p. 1–6, out. Disponível em: http://bib-di.inf.puc-rio.br/ftp/pub/docs/techreports/01_35_cortes.pdf. Acesso em: 27 jan. 2024.

DB-ENGINES. **Ranking dos bancos de dados**. 2024. Disponível em: https://db-engines.com/en/ranking. Acesso em: 22 mar. 2024.

FARAON, Renan. **Banco de dados NewSQL: Conceitos, ferramentas e comparativo para grandes quantidades de dados**. 2018. Trabalho de Conclusão de Curso (Ciência da Computação) - Universidade de Caxias do Sul, Caxias do Sul, 2018. Disponível em: https://repositorio.ucs.br/xmlui/bitstream/handle/11338/4769/TCC%20Renan%20Faraon.pdf?sequence=1&isAllowed=y. Acesso em: 30 set. 2024.

GARCIA, Vinícius Salles; SOTTO, Eder Carlos Salazar. **Comparativo entre os modelos de banco de dados relacional e não relacional**. *Revista Interface Tecnológica*, [S. l.], v. 16, n. 2, p. 12-24, 2019. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/673/408. Acesso em: 22 mar. 2024.

GODOY, Felipe Pereira de; PINHEIRO, Gabriel Benedito. **Análise de desempenho de banco de dados relacional (MySQL) e não relacional (MongoDB)**. 2021. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - FHO-Uniararas, Araras, 2021. https://courseware.fho.edu.br/repositorio-publico/eyJpdiI6IkdrZ2QyMU96MnZEdkM5YTFETXFZRkE9PSIsInZhbHVlIjoiTjlSWDA4ZWY5NE9KbDgyaEJhWEpYZz09IiwibWFjIjoiNmQ4NzAyNDNlMTZiYjRkNmU0MmI0YWJmNmMxNDIzMTM5ZTc1ZDczNWMyYzVkOTQ4YzI1YWRmYTVhNzkxMGNmOCIsInRhZyI6IiJ9?search=An%C3%A1lise%20de%20desempenho%20de%20banco%20de%20dados%20relacional. Acesso em: 30 set. 2024.

IBM. **O que é um banco de dados NoSQL**. 2024. Disponível em: https://www.ibm.com/br-pt/topics/nosql-databases#:~:text=IBM,nos%20bancos%20de%20dados%20relacionais. Acesso em: 15 out. 2024.

JUNIOR, Jair Ferreira Filho. **Proposta de um repositório de dados de benchmarking**. 2005. Dissertação (Mestrado em Engenharia de Produção) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2005. p. 14-33. Disponível em: https://www.teses.usp.br/teses/disponiveis/18/18140/tde-01082017-114551/publico/Dissert_LimaJr_JairF.pdf . Acesso em: 31 ago. 2024.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de informação gerenciais**. 7. ed. São Paulo. Pearson Education do Brasil, 1999. p. 136–178. Disponível em: https://edisciplinas.usp.br/pluginfile.php/7552318/mod_resource/content/1/Laudon%20e%20Laudon.pdf. Acesso em: 18 fev. 2024.

MATSUMOTO, Cristina Yoshie. **A importância do banco de dados em uma organização**. *Maringá Management: Revista de Ciências Empresariais*. 2006. p. 45-55. Disponível em: https://core.ac.uk/download/pdf/199473173.pdf. Acesso em: 12 mar. 2024.

MICROSOFT. **Dados não relacionais e NoSQL - Azure Architecture Center**. 2024. Disponível em: https://learn.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data. Acesso em: 06 abr. 2024.

MICROSOFT. **Guia de arquitetura de tópicos e tarefas**. 2024. Disponível em: https://learn.microsoft.com/pt-br/sql/relational-databases/thread-and-task-architecture-guide?view=sql-server-ver16. Acesso em: 21 out. 2024.

NASCIMENTO, Matheus Bellio. **MongoDB: Um Estudo Teórico-Prático do Conceito de Banco de Dados NoSQL**. 2014. p. 13. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Campo Limpo Paulista, São Paulo. 2014. Disponível em: https://www.researchgate.net/profile/Computacao-Unifaccamp/publication/272887058_MongoDB_Um_Estudo_Teorico-Pratico_do_Conceito_de_Banco_de_Dados_NoSQL_-_Trabalho_de_Diplomacao/links/54f24f3b0cf2f9e34f0357e6/MongoDB-Um-Estudo-Teorico-Pratico-do-Conceito-de-Banco-de-Dados-NoSQL-Trabalho-de-Diplomacao.pdf. Acesso em: 03 fev. 2024.

NAVATHE, Ramez; ELMASRI, Shamkant B. **Sistemas de bancos de dados**. 1. ed. São Paulo. Pearson Education do Brasil, 2005. p. 3-94. Disponível em: http://www.tonysoftwares.com.br/attachments/article/5297/Sistema_de_banco_de_dados_Navathe.pdf. Acesso em: 03 fev. 2024.

NEERAJ, Nishant. **Mastering Apache Cassandra**. 2. ed. UK, Birmingham. Packt Publishing, 2013. p. 30-166. Disponível em: https://www.k0d.cc/storage/books/Databases/Cassandra/Mastering%20Apache%20Cassandra.pdf. Acesso em: 27 nov. 2024.

OLIVEIRA, Samuel Silva. **Banco de dados não-relacionais: um novo paradigma para armazenamento de dados em sistemas de ensino colaborativo**. *ISSN 2175-6147*, Macapá. 2014. p. 184–194. Disponível em: https://www2.unifap.br/oliveira/files/2016/02/35-124-1-PB.pdf. Acesso em: 22 mar. 2024.

SILVA, Phillipe Idivaldo Mendonça. **Benchmarking em classificação de dados: Redes neurais e árvores de decisão**. 2021. p. 15. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Maranhão, São Luís. 2021. Disponível em: https://monografias.ufma.br/jspui/bitstream/123456789/6914/1/PhilipeSilva.pdf. Acesso em: 27 abr. 2024.