ESTUDO DE PERFORMANCE ENTRE SGBDS RELACIONAIS E NÃO RELACIONAIS

Felipe Ioavasso Vieira dos Santos[1], Geraldo Henrique Neto[1], Débora Pelicano Diniz[1]

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

felipeioavasso@gmail.com,
geraldo.henrique01@fatec.sp.gov.br,
debora.diniz2@fatec.sp.gov.br

Resumo. *Este artigo compara o desempenho de Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDRs) e bancos de dados NoSQL. A crescente demanda por armazenamento de dados diversos exige uma análise criteriosa entre essas duas abordagens. O estudo examina operações de leitura e escrita em MySQL (relacional) e MongoDB (NoSQL), considerando a complexidade das consultas e otimizações disponíveis. A análise utiliza métricas de desempenho em um ambiente controlado, evidenciando as vantagens e limitações de cada sistema.*

Abstract. *This paper compares the performance of Relational Database Management Systems (RDBMS) and NoSQL databases. The increasing demand for diverse data storage requires a thorough analysis between these two approaches. The study examines read and write operations in MySQL (relational) and MongoDB (NoSQL), considering the complexity of queries and available optimizations. The analysis uses performance metrics in a controlled environment, highlighting the advantages and limitations of each system.*

1. Introdução

A crescente diversidade de aplicações de *software* requer soluções de armazenamento de dados adequadas às necessidades específicas de cada domínio. Enquanto Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDRs) têm sido a base de muitas aplicações tradicionais, bancos de dados NoSQL oferecem alternativas que se destacam em cenários de dados convencionais, como documentos semi-estruturados, grafos e dados distribuídos.

As problemáticas entre SGBDR e Sistemas de Banco de Dados NoSQL surgem de suas diferenças fundamentais em termos de modelo de dados, escalabilidade, consistência e flexibilidade. Essas diferenças têm implicações significativas em uma ampla gama de aplicações e setores, desde sistemas de *e-commerce* e redes sociais até aplicações científicas e de Internet das Coisas (IoT). Este uso intenso de aplicações de *softwares* gera um crescimento exponencial da quantidade de dados gerados e consumidos. E é neste contexto que a escalabilidade do modelo e do sistema utilizado se tornou uma consideração crítica.

Este estudo se justifica ao explorar como SGBDRs e bancos de dados NoSQL lidam com a escalabilidade e o desempenho em cenários de grande volume de dados

convencionais. O conhecimento sobre as vantagens e limitações dos SGBDRs e bancos de dados NoSQL é fundamental para que profissionais e organizações tomem decisões ao escolher o sistema de banco de dados mais apropriado para suas necessidades.

O objetivo deste trabalho é o estudo de performance entre sistemas de gerenciamento de banco de dados (SGBDs) relacionais e não relacionais (NoSQL), para comparação entre os sistemas será feita as operações de leitura e escrita nos SGBDS, levando em consideração a complexidade das consultas e as otimizações disponíveis nos SGBDs.

Este trabalho está organizado em oito seções. A primeira, Introdução, fornece uma visão geral do problema, contexto e objetivos. A segunda aborda conceitos de SGBDs, tipos, dados convencionais e desafios, com destaque para SGBDs populares. A terceira seção detalha a metodologia, seleção de SGBDs, métricas de desempenho, ambiente de teste e análise dos resultados. Na quarta, examinamos as estruturas de dados, focando em modelos e desempenho de armazenamento. A quinta aborda operações de leitura e escrita, consultas e otimizações. Na sexta, apresentamos resultados comparativos e *insights*. A sétima resume os principais resultados, e a oitava lista as referências.

2. Referencial Teórico

Segundo Korth, Silberschatz e Sudarshan (1999, página 1),

> [...]"os Sistemas de Gerenciamento de Bancos de Dados (SGBDs) são uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados. O gerenciamento de dados envolve definir estruturas para armazenamento de informação e fornecer mecanismos para a manipulação de informações. Esses sistemas são projetados para lidar com grandes quantidades de informação."

SGBDs são *softwares* projetados para gerenciar o armazenamento, a recuperação, a organização e a segurança dos dados em um banco de dados. O sistema de banco de dados é um sistema computadorizado que tem o propósito de armazenar informações e permitir que os usuários a recuperem e a atualizem conforme a necessidade (DATE, 2003)[1].

Temos que entender que o banco de dados é uma parte do SGBD no qual armazena dados ou informações de uma empresa. Assim como afirma Date (2003), um banco de dados é uma coleção de dados persistente usada pelos sistemas de aplicação de uma determinada empresa.

Os bancos de dados podem ser classificados como relacionais ou não relacionais (NoSQL). Os bancos de dados que seguem o modelo relacional têm dados estruturados com uso de tabelas, que consistem em linhas e colunas, usam um esquema bem definido e fazem uso de *Structured Query Language* (SQL). Além dessas características, Edgar Frank Codd (1985) enumera treze regras para que um banco de dados seja considerado relacional:

1. Regra fundamental: um SGBD relacional deve gerir os seus dados usando apenas suas capacidades relacionais.

[1] C. J. Date trata os termos dados e informações como sinônimos.

2. Regra da informação: toda informação deve ser representada de uma única forma, como dados em uma tabela.
3. Regra de acesso garantido: todo o dado (valor atómico) pode ser acedido logicamente (e unicamente) usando o nome da tabela, o valor da chave primária da linha e o nome da coluna.
4. Tratamento sistemático de valores nulos: os valores nulos (diferente do zero, da *string* vazia, da *string* de caracteres em brancos e outros valores não nulos) existem para representar dados não existentes de forma sistemática e independente do tipo de dado.
5. Catálogo *on-line* baseado no modelo relacional: a descrição do banco de dados é representada no nível lógico como dados ordinários (isto é, em tabelas), permitindo que usuários autorizados apliquem as mesmas formas de manipular dados aplicados aos dados comuns ao consultá-las.
6. Sub linguagem abrangente: um sistema relacional pode suportar várias linguagens e formas de uso, porém deve possuir ao menos uma linguagem com sintaxe bem definida e expressa por cadeia de caracteres e com habilidade de apoiar a definição de dados, a definição de visões, a manipulação de dados, as restrições de integridade, a autorização e a fronteira de transações.
7. Regra da atualização de visões: toda visão que for teoricamente atualizável será também atualizável pelo sistema.
8. Inserção, atualização, e exclusão de alto nível: qualquer conjunto de dados que pode ser manipulado com um único comando para retornar informações, também deve ser manipulado com um único comando para operações de inserção, atualização e exclusão. Simplificando, significa dizer que as operações de manipulação de dados devem poder ser aplicadas a várias linhas de uma vez, ao invés de apenas uma por vez.
9. Independência dos dados físicos: programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as modificações na representação de armazenagem ou métodos de acesso internos.
10. Independência lógica de dados: programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as mudanças de informação que permitam, teoricamente, a não alteração das tabelas base.
11. Independência de integridade: as relações de integridade específicas de um banco de dados relacional devem ser definidas em uma sub linguagem de dados e armazenadas no catálogo (e não em programas).
12. Independência de distribuição: a linguagem de manipulação de dados deve possibilitar que as aplicações permaneçam inalteradas estejam os dados centralizados ou distribuídos fisicamente.
13. A não transposição das regras: se o sistema relacional possui uma linguagem de baixo nível (um registro por vez), não deve ser possível subverter ou ignorar as regras de integridade e restrições definidas no alto nível (muitos registros por vez).

Os bancos de dados NoSQL armazenam dados de uma maneira não tabular, são altamente escaláveis, utilizam dados semiestruturados ou não estruturados, utilizam esquemas flexíveis ou nenhum esquema, a linguagem padrão não é o SQL e oferece funcionalidades avançadas que um banco de dados SQL tradicional não possui. Normalmente os bancos de dados NoSQL têm quatro categorias de modelos de armazenamento, a saber, Chave-Valor, Documento, Coluna e Grafo. Pode-se descrever cada categoria de modelos de armazenamento da seguinte maneira (SADALAGE, 2014):

a) **Chave-Valor**: é o tipo mais simples de banco de dados NoSQL onde cada chave única aponta para um valor.
b) **Documento**: este tipo de banco de dados se assemelha ao banco de dados de modelo Chave-Valor. os dados neste tipo de banco de dados são armazenados em documentos semelhantes a objetos JSON, XML BSON. Esses documentos são estruturas de dados hierárquicas em forma de árvore que podem conter mapas, coleções e valores escalares.
c) **Coluna**: o banco de dados de coluna armazena os dados em tabelas, linhas e colunas dinâmicas. Dentro de cada família de colunas, os dados são organizados de forma que várias colunas estejam associadas a uma chave de linha específica. Essas famílias de colunas são projetadas para agrupar dados relacionados que costumam ser acessados em conjunto. Nesse contexto, a chave de linha funciona como um identificador exclusivo para cada linha, e as linhas são compostas por várias colunas. No entanto, a principal diferença é que, em bancos de dados de colunas, as linhas não precisam necessariamente ter as mesmas colunas, e novas colunas podem ser adicionadas a qualquer linha a qualquer momento, sem a necessidade de que essas mesmas colunas sejam adicionadas a todas as outras linhas. Essa flexibilidade permite uma modelagem de dados mais dinâmica e adaptável.
d) **Grafo**: o banco de dados em Grafo armazena dados em nós e arestas. Os nós também são conhecidos como Entidades (*Entities*) e as arestas são as relações entre as entidades.

Armazenar e manipular dados pode ser um desafio devido à sua natureza diversificada e complexa. Alguns dos desafios incluem a integração de dados em diferentes formatos, a enorme velocidade e volume dos dados não estruturados, a falta de um esquema predefinido para dados semiestruturados e não estruturados, além das preocupações em termos de governança, privacidade e segurança dos dados armazenados. Para lidar com esses desafios existem sistemas de banco de dados não relacionais, como o MongoDB, Cassandra, Firebase e DynamoDB.

3. Metodologia

O estudo comparativo de desempenho seleciona o MySQL, como o modelo relacional, e o MongoDB, como o modelo NoSQL. Ambos são amplamente utilizados no mercado de gerenciamento de banco de dados. A escolha da Amazon Web Services (AWS) como ambiente de teste se deve à sua infraestrutura altamente escalável e confiável. Serão utilizadas máquinas virtuais Ubuntu Server 20.04 LTS com SSD de arquitetura de 64 bits dentro de uma instância EC2, garantindo um ambiente consistente e de alto desempenho para ambas as soluções.

MySQL é um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR) de código aberto, desenvolvido David Axmark, Allan Larsson e Michael "Monty" Wideniuse e que foi comprado pela Oracle Corporation em 2008 (MYSQL, 1995). O MongoDB, é baseado em software livre e utiliza documentos flexíveis para armazenar dados (MONGODB, 2024). É importante lembrar que ambos os sistemas possuem versões pagas e abertas.

Para os testes de inserção de dados, foram criadas duas massas de dados, totalizando 1GB e 2GB. Os dados foram organizados em arquivos no formato .txt. Conforme mostrado na Tabela 1, os testes foram realizados com volumes totais estimados

de 1GB e 2GB, envolvendo a inserção de 148 e 279 arquivos, respectivamente. Cada arquivo foi limitado a 16,9MB devido à restrição de tamanho de arquivos no MongoDB.

Tabela 1 – Constituintes da massa de dados

Inserção Total (Gb)	Quantidade de arquivos	Tamanho menor arquivo	Tamanho maior arquivo
1	148	1 Kb	16,9 Mb
2	279	1 Kb	16,9 Mb

Fonte: Elaborado pelo próprio autor, 2024

O acesso aos SGBDs, a inserção e a reconstituição dos arquivos, bem como o registro dos tempos de execução, foram realizados via linha de comando em uma instância EC2 da AWS.

Todo o trabalho foi desenvolvido utilizando a linguagem de programação Python, devido à sua presença nativa no Ubuntu. O repositório foi organizado em um *workspace* chamado *"Repositório",* como observado na Figura 1. Um ambiente virtual, denominado *"venvrepositorio"*, foi criado para isolar o ambiente de desenvolvimento e instalar as bibliotecas necessárias para a conexão com os bancos de dados.

Dentro do *workspace "Repositório"*, foi criado o diretório *"repos"*, contendo quatro subdiretórios:

a) massa_Dados: Contém os *scripts* para gerar as massas de dados utilizadas no trabalho, organizados nos subdiretórios: gerador_massa_dados e massa_dados.
b) reconstituicao_dados: Armazena os arquivos recuperados dos bancos de dados.
c) resultado_dos_testes: Inclui todos os arquivos com os registros de aferição do tempo.
d) testes: Contém todos os *scripts* para inserção e reconstituição dos dados.

Com essa estrutura organizacional, a execução e análise dos testes são otimizadas, garantindo uma gestão eficiente do código e dos arquivos. O procedimento de análise de desempenho entre os Sistemas Gerenciadores de Banco de Dados (SGBDs) MySQL e MongoDB segue uma abordagem sistemática dividida em três etapas distintas, cada uma composta por operações específicas. O processo é repetido três vezes para inserção, recuperação e eliminação de dados de 1GB e 2GB em ambos os SGBDs, garantindo a consistência dos resultados.

A primeira etapa consiste na inserção de arquivos em que os arquivos totalizando 1 Gb e 2 Gb são inseridos em uma única transação. Após a inserção, registra-se o par ordenado tamanho (Kb) por tempo (ms) em um arquivo CSV e prossegue com a reinicialização do serviço na EC2 para evitar qualquer cache que possa influenciar nos resultados dos testes. A segunda etapa é a recuperação dos arquivos e o registro do par ordenado tamanho (KB) por tempo (ms) em um arquivo CSV. Na terceira etapa acontece a eliminação de todos os registros da tabela. Na Figura 2 está demonstrado o processo.

No MySQL, utiliza-se a instrução *INSERT* para inserir os arquivos, a instrução SELECT para recuperar os arquivos e a instrução *TRUNCATE* para limpar a tabela (Figura 3). No MongoDB, a instrução *collection.insert_one({"coluna": dados})* é empregada para inserir um documento na coleção *collection* do banco de dados 'tcc', onde dados representa o conteúdo a ser inserido. Para recuperar todos os documentos da coleção, utiliza-se a instrução *collection.find()*. Por fim, para excluir todos os documentos

da coleção, é utilizada a instrução *collection.delete_many({})*. Na Figura 4 está demonstrado o processo das instruções no MongoDB.

4. Armazenamento de Dados no MongoDB

O MongoDB organiza os dados em *databases*, *collections* e documentos. Estes últimos são estruturados no formato JSON (*JavaScript Object Notation*), mas armazenados em BSON (*Binary JSON*). Um documento MongoDB é, portanto, um conjunto de pares chave-valor, onde os valores podem ser de tipos variados, incluindo JSON, BSON, *arrays*, objetos compostos de pares chave-valor e nulos.

Este banco de dados não relacional é reconhecido por seu desempenho em termos de eficiência de espaço e tempo de resposta. Isso se deve à utilização do mecanismo de armazenamento *WiredTiger*, que inclui um processo de compactação automática para reduzir o espaço em disco. Além disso, o uso de índices eficientes contribui significativamente para a economia de espaço e agilidade nas consultas.

Os índices são fundamentais para otimizar a busca por documentos, permitindo consultas rápidas aos dados. Esses recursos, aliados às operações atômicas empregadas pelo MongoDB, não apenas tornam o sistema eficiente em termos de resposta, mas também auxiliam na economia de espaço. Adicionalmente, o Sharding distribui os dados em partes menores, espalhando-os por vários servidores, o que também contribui para a otimização do armazenamento. Outro recurso utilizado pelo banco de dados é o cache em memória, que armazena partes dos dados frequentemente acessados.

5. Manipulação de Dados

Este capítulo aborda a manipulação de dados em dois tipos distintos de sistemas de gerenciamento de bancos de dados: relacionais (MySQL) e não relacionais (NoSQL, exemplificado pelo MongoDB). No MySQL, as operações de leitura e escrita são executadas utilizando a linguagem SQL (*Structured Query Language*), com cada operação de inserção, leitura ou deleção de dados requerendo um comando SQL correspondente. Já o MongoDB adota uma abordagem diferente para operações de leitura e escrita, utilizando sua própria linguagem de consulta.

As diferenças entre MySQL e MongoDB têm um impacto significativo nas etapas de inserção de dados. No *script* para inserção de dados com MySQL, foram necessárias cinco etapas distintas. A primeira etapa é a conexão com o banco de dados MySQL, estabelecendo a comunicação entre a aplicação e o banco de dados. A segunda etapa consiste na criação da tabela "tabela1gb", caso ela não exista. Esta etapa é crucial para garantir que a estrutura necessária para armazenar os dados esteja disponível. A terceira etapa envolve o início de uma transação. As transações são importantes para agrupar operações relacionadas e garantir a consistência dos dados. A quarta etapa está relacionada a cada arquivo a ser inserido. Para cada arquivo, é necessário abrir o arquivo, ler os dados e, em seguida, realizar a inserção desses dados na tabela correspondente. Por fim, a quinta etapa é o *commit* da transação. O *commit* confirma as alterações feitas durante a transação, garantindo que todas as operações sejam aplicadas de forma consistente no banco de dados. Na Figura 5 estão as etapas de inserção para o MySQL.

Foram necessárias apenas duas etapas para realizar o mesmo processo de inserção de dados no MongoDB. A primeira etapa é a conexão com o banco de dados e a segunda etapa é relacionada a cada arquivo a ser inserido, processo este semelhante ao do MySQL,

que é a abertura do arquivo e leitura dos dados e a consequente inserção dos dados em um documento na coleção. Na Figura 6 estão as etapas de inserção para o MongoDb.

Para recuperar os dados dos bancos de dados os *scripts* contaram com cinco etapas, a primeira sendo a conexão com o banco de dados. A segunda é a definição do diretório de destino onde os arquivos recuperados serão salvos. A terceira etapa, caso do MongoDB, consulta para recuperar todos os documentos da coleção especificada e no caso do MySQL executar uma consulta SQL para selecionar todos os registros da tabela especificada. A quarta etapa é a gravação dos documentos em arquivos que itera sobre os documentos recuperados e os grava em arquivos individuais no diretório de destino. A quinta etapa é o fechamento da conexão. Na Figura 7 está o *script* para o MongoDb. Na Figura 8 podemos observar que o processo para recuperar os arquivos do MySQL são bem semelhantes ao recuperar os arquivos do MongoDB.

Na seção 6 de resultados pode-se ver que apesar das etapas serem parecidas o tempo utilizado para recuperar os dados são bem diferentes.

6. Resultados e Discussão

6.1 Ambiente

Os testes de desempenho dos SGBD's MySQL e MongoDB foram realizados na instância da EC2 na AWS.

6.1.1 Inserção de 1Gb arquivos

A inserção de 1Gb de arquivos realizada pelas respectivas instruções de *collection.insert_one* no MongoDB e *insert* no MySQL teve os seguintes resultados, conforme a Figura 9.

Na Figura 9 pode-se observar o comportamento dos dois sistemas fazendo a inserção de dados com tamanho de até 1000 Kb. O tempo necessário para inserir arquivos de até 23 KB mantém-se estável para ambos os sistemas. Entretanto, a partir desse tamanho, ambos apresentam variações de tempo. O MySQL começa a demonstrar uma tendência em que quanto maior o arquivo, mais tempo é necessário para inseri-lo no banco de dados. Já o MongoDB exibe um comportamento oscilante, com picos de tempo maiores que os do MySQL, mas demandando menos tempo na maioria dos casos.

Os arquivos no intervalo de 1000 Kb e 16,90 Mb mostra a tendência de aumento de tempo do MySQL conforme cresce o tamanho do arquivo é confirmada, assim como mostrado na Figura 10. Por exemplo, para um arquivo de 16,967 MB, o tempo necessário foi de 1523 ms. De acordo com a Figura 10, o MongoDB apresenta o melhor desempenho. O maior tempo registrado para o MongoDB foi de 1357 ms ao inserir um arquivo de 15,626 MB, com uma média geral de 340 ms para inserir os arquivos. Em comparação, a média para o MySQL foi de 799 ms.

No processo de inserção, o tempo total médio foi de 50389,78 ms para o MongoDB e 118296,97 ms para o MySQL. Na inserção de 1 GB, o sistema não relacional demonstrou ser 57,41% mais rápido que o sistema relacional.

6.1.2 Inserção de 2b arquivos

Na inserção de 2 GB de dados, o MongoDB também apresentou um tempo de inserção

inferior ao do MySQL. O banco de dados não relacional mostrou tempos menores e mais estáveis na maioria dos casos, embora com alguns picos significativos. Em contraste, o MySQL apresentou um aumento mais consistente nos tempos de inserção conforme o tamanho dos arquivos aumentava, com alguns *outliers* significativos.

O tempo total médio para a inserção de 2 GB foi de 114539,24 ms para o MongoDB e 148107,87 ms para o MySQL. A média dos tempos de inserção foi de 410 ms para o MongoDB e 530 ms para o MySQL. Portanto, o banco de dados não relacional foi 22,66% mais rápido do que o banco de dados relacional. Na Figura 11 está apresentado o resultado para a inserção da massa de dados de 2Gb.

Na Tabela 2 está mostrado o tempo total final para as inserções de 1Gb e 2Gb, o banco de dados MongoDB manteve a superioridade em ambos os cenários de inserção de dados no ambiente da EC2 na AWS.

Tabela 2 – Tempos de inserção (ms) usando a instância da EC2

	MongoDB	**MySQL**
Inserção de 1Gb de dados	50389,78	118296,97
Inserção de 2Gb de dados	114539,24	148107,87

Fonte: Elaborado pelo autor, 2024.

6.1.3 Recuperação de 1Gb arquivos

A recuperação de 1Gb de arquivos realizada pelas respectivas instruções de *collection.find* no MongoDB e *select* no MySQL teve os seguintes resultados, conforme a Figura 12.

Para entender melhor a performance na recuperação de arquivos entre MongoDB e MySQL os dados serão divididos em faixas de tamanho de arquivo e comparar os tempos de resposta médios em cada faixa, como demonstrado na Tabela 3.

Tabela 3 – Tempo de resposta na recuperação de dados 1Gb

Faixa de tamanho (Kb)	**Média MongoDB (ms)**	**Média MySQL (ms)**
0 – 1	3.00	1.99
1 – 100	4.67	3.20
100 – 1000	9.15	13.14
1000 – 10000	50.73	106.87
10000 – 17000	83.29	183.97

Fonte: Elaborado pelo autor, 2024

Na primeira faixa de tamanho o MySQL é ligeiramente mais rápido e para os arquivos maiores de 1Kb o MongoDB mostra tempos de resposta significativamente menores que o MySQL. O tempo de resposta médio do MongoDB é de 40,26 ms e de 96,14 ms para o MySQL. Todo o processo de recuperação de 1Gb de arquivos teve um tempo total médio de 5959,61 ms usando o banco de dados não relacional e um tempo total médio de 14343,38 ms para o Banco de dados relacional. Assim sendo em média o MongoDB é 58,12% mais rápido que o MySQL na recuperação de arquivos.

6.1.4 Recuperação de 2Gb arquivos

A recuperação de 2Gb segue o mesmo processo e usa as mesmas instruções do teste anterior com 1Gb de arquivo. A Figura 13 mostra o resultado dos testes.

Na Tabela 4 estão reunidos os dados em faixas de tamanho de arquivo e compara os tempos de resposta médio entre os dois sistemas para 2Gb de arquivos.

Tabela 4 – Faixa de tamanho de arquivo de 2Gb

Faixa de tamanho (Kb)	Média MongoDB (ms)	Média MySQL (ms)
0 – 100	**4.25**	**6.66**
100 – 1000	**10.65**	**79.01**
1000 – 10000	**25.94**	**207.57**
10000 - 17000	**68.82**	**523.73**

Fonte: Elaborado pelo autor, 2024

Os tempos de resposta na faixa de tamanho de 0 – 100Kb são relativamente baixos com o MongoDB sendo ligeiramente mais rápido. O tempo de resposta médio nesta faixa é de 5.95 ms no banco de dados não relacional e de 7.21ms com o MySQL. Para os arquivos maiores de 100 Kb os tempos de resposta são significativamente menores com o MongoDB do que com o MySQL. Esse comportamento reflete no tempo total médio para recuperar os arquivos, com o banco de dados não relacional o tempo total médio é de 13065.61 ms com tempo médio de 46.83 ms e o tempo total para o banco de dados relacional é de 60234.12 ms com tempo médio de 215.89 ms.

7. Conclusão

Este estudo comparativo analisou o desempenho de sistemas de gerenciamento de banco de dados relacionais (MySQL) e não relacionais (MongoDB) em cenários de grandes volumes de dados. Através de testes de inserção e recuperação de dados em um ambiente controlado, utilizando instâncias EC2 da AWS, foi possível observar diferenças significativas no comportamento e desempenho dos dois SGBDs.

Os resultados demonstraram que o MongoDB superou o MySQL em termos de tempo de inserção tanto para 1 GB quanto para 2 GB de dados. Especificamente, o MongoDB foi 57,41% mais rápido na inserção de 1 GB de dados e 22,66% mais rápido na inserção de 2 GB de dados. Este desempenho superior pode ser atribuído à flexibilidade do modelo de dados do MongoDB e à eficiência de seu mecanismo de armazenamento, que utiliza compactação automática e índices eficientes.

Na recuperação de dados, o MongoDB também apresentou tempos de resposta significativamente menores em comparação com o MySQL. Para arquivos de 1 GB, o MongoDB foi em média 58,12% mais rápido, enquanto para arquivos de 2 GB, a diferença de desempenho também foi notável, com o MongoDB mantendo tempos de resposta menores na maioria das faixas de tamanho de arquivo.

Os resultados indicam que o MongoDB é uma escolha mais adequada para aplicações que exigem alta performance em operações de inserção e recuperação de grandes volumes de dados não estruturados ou semiestruturados. Por outro lado, o MySQL pode ainda ser preferível em cenários onde a integridade transacional e a

consistência forte são cruciais, devido às suas características inerentes como um banco de dados relacional.

Este estudo focou em um ambiente específico e em conjuntos de dados simulados, o que pode limitar a generalização dos resultados para todos os tipos de aplicações e ambientes. Futuros trabalhos podem explorar outras dimensões de comparação, como a performance em operações complexas de *join*, análise de desempenho em outros tipos de cargas de trabalho (e.g., leitura intensiva), e a avaliação de outros SGBDs relacionais e não relacionais. Além disso, seria interessante investigar o impacto de diferentes configurações de *hardware* e otimizações de *software* na performance dos SGBDs.

O estudo contribui para a compreensão das vantagens e limitações dos SGBDs relacionais e não relacionais, oferecendo *insights* valiosos para profissionais e organizações na tomada de decisões sobre a escolha do sistema de banco de dados mais apropriado para suas necessidades específicas.

8. Referências

COOD, E. F. (1985). Is Your DBMS Really Relational?. Computer World

DATE C. J. (2003). Introdução A Sistemas de Banco de Dados. 8. Ed. Rio de Janeiro: Elsevier

KORTH, H. F.; SILBERSCHATZ, A.; SUDARSHAN, S. (1999). Sistemas de Banco de Dados. 3. Ed. - São Paulo: Makron Books

MONGODB (2023). What is NoSQL? NoSQL Databases Explained [on-line]. Disponível em “https://www.mongodb.com/nosql-explained”.

SADALAGE P. (2014). NoSQL Databases: An Overview [on-line]. Disponível em “https://www.thoughtworks.com/insights/blog/nosql-databases-overview”.

UNIVERSIDADE DO ESTADO DE SANTA CATARINA. O que é o MySQL. Disponível em: https://www.udesc.br/arquivos/ceavi/id_cpmenu/291/o_que_e_o_mysql_15380696706509_291.pdf. Acesso em: 22 maio 2024.

Apêndice

Para visualização das figuras, acesse o link do github:

https://github.com/felipeioavasso/Figuras_TCC_FATEC_RP_2024