UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

HENRIQUE MENDES DE MOURA

Análise de desempenho do banco de dados MongoDB quanto ao armazenamento de informações diárias

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof[a]. Dr[a]. Renata Galante

Porto Alegre
2022

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos André Bulhões Mendes
Vice-Reitora: Profª. Patricia Helena Lucas Pranke
Pró-Reitor de Graduação: Profª. Cíntia Inês Boll
Diretora do Instituto de Informática: Profª. Carla Maria Dal Sasso Freitas
Coordenador do Curso de Engenharia de Computação: Prof. Walter Fetter Lages
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Agradeço primeiramente à minha família, especialmente ao meu pai, Dalton Eliseu Lima de Moura, e minha mãe, Nilda Mendes de Moura, por todo o apoio e confiança desde o inicio desse longo processo.

Agradeço também à minha esposa Rayana Caroline Picolotto de Moura, que esteve comigo em todos os momentos, sempre me apoiando e me motivando, sendo com certeza, fundamental na caminhada até aqui.

Agradeço também a todos os professores que tive durante a graduação, por toda a bagagem de conhecimento compartilhada, em especial à professora Renata de Matos Galante, orientadora deste trabalho, por toda a atenção e dedicação prestada a mim e meus colegas.

Muito obrigado.

RESUMO

Banco de dados não-relacionais estão sendo cada vez mais utilizados devido a sua flexibilidade em lidar com um grande volume de dados de vários tipos diferentes. Porém, essa flexibilidade pode fazer com que o desempenho seja perdido dependendo da maneira como os dados estão organizados. O objetivo deste trabalho é propor três formas diferentes de modelar uma base de dados financeiros que é atualizada diariamente, mapear e implementar em um banco de dados de documentos e, então, analisar o desempenho das implementações propostas. As propostas foram analisadas quanto ao armazenamento em disco, o tempo médio de execução de consultas, inserções, alterações e exclusões de informações diárias. A proposta 1, que tem como característica principal alocar as informações diárias em um campo do tipo lista, apresentou melhor desempenho quanto ao armazenamento em disco, tempo médio de execução de consultas (sem índice `_id`) e exclusões de uma informação diária. A proposta 2, que tem como característica principal um campo do tipo chave-valor, apresentou melhor desempenho ao realizar consultas (com índice `_id`) e alterações em uma informação diária. A proposta 3, que tem como característica principal apresentar um documento para cada informação diária, apresentou melhor desempenho quanto ao tempo médio de resposta ao realizar a inserção de uma nova informação diária. Esses resultados indicam que, tendo em vista o desempenho, a forma como os dados são organizados é tão importante quanto a escolha de um banco de dados.

Palavras-chave: MongoDB. banco de dados não-relacional. análise de desempenho. dados com informações diárias.

Daily information performance analysis using MongoDB database

ABSTRACT

Non-relational databases are increasingly being used with a great deal of flexibility on data of many different types. However, this flexibility can make performance secure the way the data is organized. The objective of this work is to propose three different ways to model a financial database that is updated daily, map and implement in a document database, then analyze the proposal implementations. As the proposals were related to disk storage, the average execution time of daily queries, inserts, changes and deletions. The proposal 1, whose main feature is to allocate daily information in an array-type field, presented better performance in terms of disk storage, average query execution time (without `_id` index) and exclusions of daily information. Proposal 2, whose main feature is a key-value field, presented better performance when performing queries (with`_id` index) and changes in daily information. Proposal 3, whose main characteristic is to present a document for each daily time, presented better performance in terms of the average response time when performing the insertion of new daily information. These results indicate that, in terms of performance, the way the data is organized is as important as the choice of a database.

Keywords: MongoDB. non-relational database. performance analysis. data with daily information.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	*Atomicity, Consistency, Isolation, Durability* - Atomicidade, Consistência, Isolamento e Durabilidade
BSON	*Binary JSON* - JSON binário
CNPJ	Cadastro Nacional de Pessoas Jurídicas
CSV	*Comma Separated Values* - Valores separados por vírgulas
CVM	Comissão de Valores Mobiliários
DBMS	*Data Base Management System* - Sistema de gerenciamento de banco de dados
CRUD	*Create, Read, Update and Delete* - Criação, consulta, atualização e destruição de dados
GB	*Gigabyte*
JSON	*JavaScript Object Notation* - Notação de Objeto JavaScript
KB	*Kilobyte*
MB	*Megabyte*
NoSQL	*No Structured Query Language ou Not Only Structured Query Language* - Sem linguagem de consulta estruturada ou não apenas linguagem de consulta estruturada
SQL	*Structured Query Language* - Linguagem de consulta estruturada
XML	*eXtensive Markup Language* - Linguagem Extensível de Marcação Genérica
K	Mil
M	Milhão

SUMÁRIO

1 INTRODUÇÃO

Com o rápido desenvolvimento da Internet e dos dispositivos móveis nos últimos anos, tornou-se cada vez mais necessário a criação de bancos de dados que fossem capazes de armazenar de forma eficiente a enorme quantidade de dados gerados (GUNAWAN; RAHMATULLOH; DARMAWAN, 2019). Por muito tempo, os bancos de dados relacionais juntamente com a linguagem SQL padrão eram a principal escolha lógica para as empresas. Porém, à medida que o volume de dados cresceu, os bancos de dados relacionais começaram a apresentar limitações como o tamanho de armazenamento, escalabilidade e eficiência de consulta (MARTINS; ABBASI; SÁ, 2019). Para resolver esse problema, engenheiros de computação começaram a buscar alternativas aos bancos de dados relacionais, e, como resultado, os bancos de dados NoSQL apareceram. Eles oferecem melhores recursos para desempenho e escalabilidade e um modelo de dados muito mais flexível do que bancos de dados relacionais (FRACZEK; PLECHAWSKA-WOJCIK, 2017).

Embora os bancos de dados NoSQL tenham surgido como uma alternativa ao armazenamento e gerenciamento de grandes bases de dados, o seu desempenho é fortemente influenciado pela forma como a modelagem de dados e os recursos de consulta se ajustam aos casos de uso da aplicação (KLEIN et al., 2015). Por esse motivo, são necessários testes e caracterização específicos do sistema. Por mais que seja difícil determinar uma única resposta “certa” ao selecionar um componente complexo para uma aplicação, a seleção de componentes inadequados pode custar muito mais caro e ser muito mais trabalhoso. Essas inadequações podem levar ao cancelamento de projetos, principalmente em sistemas de big data de grande escala devido à sua complexidade e a magnitude do investimento (KLEIN et al., 2015).

Bancos de dados que recebem aporte de informações diárias são um exemplo de banco de dados muito volumoso e de difícil gerenciamento. Estes bancos de dados são muito comuns nos mais variados âmbitos da sociedade. Os bancos de dados com informações diárias estão presentes, por exemplo, na área da saúde, compondo os prontuários de pacientes de hospitais e clínicas das mais variadas especialidades, cujo *status* é atualizado todos os dias. Dados espaciais e meteorológicos também são armazenados em bancos de dados que são atualizados diariamente, sendo comumente utilizados em pesquisas da área ambiental, por exemplo. No mercado financeiro, as informações são atualizadas constantemente, como por exemplo o preço de abertura, fechamento e as variações que ocorreram ao longo do dia em um ativo financeiro, gerando um histórico com grande volume de in-

formações diárias.

O presente trabalho tem como objetivo analisar o desempenho de uma base de dados com informações diárias do mercado financeiro, implementada a partir de três propostas distintas de modelagem de dados. A proposta 1 tem como principal característica alocar as informações diárias em um campo do tipo lista. A proposta 2 tem como principal característica alocar as informações diárias em um campo do tipo chave-valor. A proposta 3, por sua vez, tem como característica principal apresentar um documento para cada informação diária. O intuito é identificar se há diferença no desempenho ao realizar inserções, alterações, consultas e exclusões das informações diárias sobre as três propostas implementadas. Essa avaliação é realizada a partir da comparação entre o tempo de resposta e o armazenamento utilizado para cada uma das propostas, utilizando o banco de dados não-relacional MongoDB.

Os resultados mostraram que a proposta 1 apresentou melhor desempenho quanto ao armazenamento em disco e obteve o menor tempo médio ao executar consultas (sem o índice "`_id`") e exclusões de informações diárias. A proposta 2 apresentou melhor desempenho ao realizar consultas (com o índice "`_id`") e alterações nas informações diárias. A proposta 3, por sua vez, apresentou melhor desempenho quanto ao tempo médio de resposta ao realizar a inserção de uma nova informação diária.

O restante do trabalho está organizado da seguinte forma: O capítulo 2 apresenta os conceitos e tecnologias que nortearam o desenvolvimento deste trabalho e alguns trabalhos relacionados. O capítulo 3 descreve a visão geral do trabalho e o banco de dados que foi utilizado, detalhando de que forma os dados foram processados e como foram desenvolvidas as três diferentes propostas de modelagem de dados. O capítulo 4 apresenta a avaliação experimental, onde são descritos todos os experimentos realizados e seus resultados, que são discutidos na última seção do capítulo. Por fim, o item 5 apresenta as conclusões do trabalho obtidas a partir dos experimentos realizados.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Neste capítulo, são apresentados os conceitos que nortearam o desenvolvimento deste trabalho. Também são apresentadas as tecnologias que foram utilizadas desde a criação da base de dados até o desenvolvimento dos testes de desempenho das diferentes propostas de modelagem de banco de dados. Por fim, são apresentados alguns trabalhos relacionados.

2.1 Banco de dados

Os sistemas de banco de dados são projetados para gerenciar grandes volumes de informações. Essas informações que compõem os bancos de dados são gerenciadas por um sistema de gerenciamento de banco de dados (SGBD), que é definido como um conjunto de dados inter-relacionados bem como um conjunto de programas para acessar esses dados. O gerenciamento de dados envolve tanto a definição de estruturas para o armazenamento de informações quanto para a manipulação dessas informações. O principal objetivo de um SGBD é fornecer uma maneira conveniente e eficiente de armazenar e recuperar as informações contidas no banco de dados (SILBERSCHATZ et al., 2002).

Nos últimos anos, o conceito de SGBD tornou-se, informalmente, sinônimo de banco de dados. Os bancos de dados tornaram-se um recurso utilizado todos os dias por milhões de pessoas, o que impulsionou a necessidade de ter todos esses dados estruturados e organizados da melhor maneira para que possam ser utilizados de forma rápida e fácil (ABRAMOVA; BERNARDINO, 2013).

Há dez anos atrás, os “Big data” já estavam em listas do Gartner como “Top 10 Critical Tech Trends For the Next Five Years"(SAVITZ, 2012). O conceito de Big data se refere a uma coleção de conjuntos de dados muito grandes com uma grande diversidade de tipos, de modo que se torna difícil processar usando abordagens de processamento de dados de última geração ou plataformas tradicionais de processamento de dados (CHEN; ZHANG, 2014). Em 2012, Gartner trouxe uma definição mais detalhada: “Big Data são ativos de informação de alto volume, alta velocidade e/ou alta variedade que requerem novas formas de processamento para permitir uma melhor tomada de decisão, descoberta de *insights* e processos otimização”. Uma década depois, os Big Data seguem sendo um desafio para a coleta, armazenamento, análise e visualização de dados, uma vez que os bancos de dados seguem aumentando em uma taxa exponencial.

2.2 Abordagens relacional e não-relacional

De modo geral, os bancos de dados são divididos em dois grandes grupos: relacionais (dados estruturados armazenados em tabelas) e não-relacionais (dados não-estruturados armazenados em arquivos, grafos, chave-valor ou colunas). Dependendo das características dos dados e como eles estão sendo disponibilizados, um modelo de banco de dados torna-se mais aplicável, ou, ainda, é possível que se faça uso de modelos híbridos. O modelo relacional é considerado mais simples do que outros modelos e por isso, é o principal modelo de dados para aplicativos comerciais de processamento de dados utilizado atualmente. A estrutura fundamental do banco de dados relacional é a relação, também conhecida como tabela (ELMASRI et al., 2005). As tabelas podem se relacionar por meio de chaves, evitando duplicidade de dados, e podem fazer consultas por *joins* (junções).

No entanto, o crescimento dinâmico da Internet e dos dispositivos móveis gerou um aumento significativo na quantidade de dados. Isso fez com que os bancos de dados relacionais não fossem mais tão eficazes, uma vez que não foram projetados para lidar com tamanha quantidade de dados. Para sustentar essa nova demanda, surgiram os bancos de dados não-relacionais, também conhecidos como banco de dados NoSQL. Os bancos de dados NoSQL oferecem melhores recursos para desempenho e escalabilidade, sendo um modelo de dados muito mais flexível do que bancos de dados relacionais (KOLONKO, 2018). A principal diferença entre os modelos relacionais e não-relacionais é que os bancos de dados não-relacionais não utilizam relações (tabelas) como estrutura de armazenamento. Além disso, não utilizam SQL como linguagem de consulta, não executam operações do tipo *join*, não garantem propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) e podem ser dimensionados horizontalmente (JATANA et al., 2012).

2.3 Classificação dos bancos de dados não-relacionais

Os bancos de dados não-relacionais são classificados de acordo com a forma em que os dados são armazenados (TIWARI, 2011):

1) *Orientado a colunas* - diferente dos dados relacionais que são armazenados em linhas, esse tipo de dado é armazenado em colunas. Isso faz com que o armazenamento seja mais eficiente, pois uma vez que o valor de alguma coluna não existe, este não es-

tará escrito no banco de dados, economizando espaço (CHOVANEC; CHOVANCOVÁ; DUFALA, 2014);

2) *Orientado a documentos* - armazenam dados estruturados em documentos e os formatos mais usados são JSON ((JavaScript Object Notation), BSON (Binary JSON) ou XML. Os documentos são armazenados em coleções e cada documento individual pode possuir colunas diferentes. Isso faz com que o banco de dados seja extremamente flexível e dinâmico, podendo armazenar vários tipos de dados, sendo ideal para o desenvolvimento de aplicativos da Web;

3) *Banco de dados chave-valor* - constituem a forma mais simples de banco de dados. Esses bancos de dados armazenam pares de chave-valor semelhantes a tabelas hash ou campos associativos. O par de chave-valor deve ser identificado por uma chave e a chave deve ser conhecida ao recuperar o registro. Esses bancos de dados são eficazes para trabalhar com grandes quantidades de dados (ÁDÁM et al., 2014);

4) *Orientado a grafos* - útil para representar relações complexas, os dados podem ser armazenados na forma de grafos. As propriedades desses bancos de dados são baseadas na teoria dos grafos. Grandes sites como Amazon, Facebook e Yahoo fazem uso desse tipo de dado, pois eles são usados principalmente para a representação visual de informações. Esse tipo de armazenamento não permite escala horizontal.

2.4 MongoDB

Esta seção descreve o banco de dados MongoDB, que foi escolhido para a implementação deste trabalho. Por utilizar documentos como base de armazenamento de dados, o MongoDB permite que praticamente qualquer estrutura de dados possa ser modelada e manipulada facilmente. O MongoDB é um tipo de sistema gerenciador de banco de dados orientado a documentos. Portanto, ele não possui o conceito de esquemas, tabelas e linhas. Neste banco de dados, não há conformidade com as restrições ACID (Atomicidade, Consistência, Isolamento e Durabilidade), não há transações, chaves estrangeiras e nem *joins* (junções). O MongoDB foi criado com o intuito de trabalhar com documentos ao invés de linhas, com o objetivo de ser extremamente rápido, amplamente escalável e fácil de usar. Uma grande vantagem que o MongoDB apresenta é a de não precisar fazer com que os seus dados encaixem-se em uma tabela, o que de fato nem sempre é possível. Isto significa que o MongoDB é indicado para armazenar dados complexos (HOWS; MEMBREY; PLUGGE, 2019).

A simultaneidade e durabilidade são umas das principais características do MongoDB. A simultaneidade ocorre porque são permitidos vários usuários executando operações de leitura e gravação nos mesmos dados. A durabilidade é garantida por meio da criação de réplicas para evitar a perda de dados (ANUSHA et al., 2021). O MongoDB também oferece um bom desempenho, acessibilidade e escalabilidade. Todos os documentos são agrupados como grupos com base em sua estrutura e os dados são armazenados no formato BSON (CHAUHAN, 2019). Além disso, o MongoDB também possui flexibilidade para trabalhar com os dados, pois os dados não estão associados a nenhuma coluna ou tipo de dados predefinidos como ocorre em bancos relacionais. Porém, essa flexibilidade pode afetar o desempenho do banco de dados.

É importante destacar que o MongoDB não inclui semântica para transações. Isso significa que ele não oferece garantias sobre a consistência e o armazenamento dos dados. Isso é fundamental para que ele se mantenha simples, rápido e escalável, pois uma vez que recursos pesados são deixados de lado, escalar horizontalmente torna-se mais fácil. Nesse caso, é melhor usar vários computadores menores e menos potentes do que um único computador muito potente, pois o MongoDB oferece durabilidade quando usado em conjunto com pelo menos três servidores, que é o mínimo recomendado para instalações em ambiente de produção (HOWS; MEMBREY; PLUGGE, 2019).

Um dos recursos disponíveis no MongoDB que contribui para que ele seja amplamente utilizado em diferentes aplicações é a indexação. A indexação é uma das formas especiais da estrutura de dados. A indexação desempenha um papel importante na melhoria de desempenho, economizando tempo de execução na busca de documentos (CHOPADE; PACHGHARE, 2020). Os índices suportam a execução eficiente de consultas no MongoDB. Sem índices, o MongoDB deve realizar uma varredura em todos os documentos de uma coleção e assim selecionar os que correspondem à instrução de consulta. Utilizando um índice apropriado, é possível limitar o número de documentos que devem ser inspecionados (MONGODB, 2022).

Índices são estruturas de dados especiais que armazenam uma pequena porção do conjunto de dados da coleção de forma fácil de percorrer. O índice armazena o valor de um campo específico ou conjunto de campos, ordenados pelo valor do campo. A ordenação das entradas de índice suporta correspondências de igualdade eficientes e operações de consulta baseadas em intervalo. Além disso, o MongoDB pode retornar resultados classificados usando a ordenação no índice (MONGODB, 2022). O índice default do MongoDB é o índice `_id`. O MongoDB cria um índice exclusivo no campo `_id` durante

a criação de uma coleção, impedindo que sejam inseridos dois documentos com o mesmo valor para o campo `_id` (MONGODB, 2022).

2.5 Trabalhos relacionados

A literatura disponível sobre avaliação do desempenho de banco de dados está dividida em dois grandes grupos. A maioria dos trabalhos compara o desempenho de banco de dados relacionais e não-relacionais e a outra parte compara diferentes tipos de bancos de dados não-relacionais entre si. Trabalhos que avaliam a performance de banco de dados não-relacionais baseados na modelagem, projeto e organização dos dados são muito escassos, o que dificulta a busca por informações sobre o tema mas ao mesmo justifica a proposta do presente trabalho.

Em 2019, Gunawan e colaboradores compararam o tempo de resposta de cada consulta para o processo de criação, leitura, atualização e exclusão (CRUD) em um documento armazenado entre os bancos de dados NoSQL, MongoDB, ArangoDB e CouchDB. Os resultados experimentais do estudo revelaram que o MongoDB teve a menor média de tempo de resposta para o processo de leitura (0,017 segundos), atualização (25,358 segundos) e exclusão (0,055 segundos) em comparação com ArangoDB e CouchDB. No entanto, para o processo de inserção, o tempo de resposta do ArangoDB foi de 28,493 segundos, sendo menor em comparação com o MongoDB e CouchDB (GUNAWAN; RAHMATULLOH; DARMAWAN, 2019). Vokorokos e colaboradores (2016) compararam os bancos de dados não-relacionais MongoDB e ElasticSearch com o intuito de verificar a velocidade de leitura enquanto preservam a integridade dos dados. Eles constataram que ambos fazem um bom trabalho em preservar a integridade dos dados mantendo alta velocidade de leitura e destacam pontos onde cada um dos bancos de dados apresenta melhor performance (VOKOROKOS; UCHNÁR; LEŠČIŠIN, 2016).

Em 2016, Fraczek e Plechawska-Wojcik compararam o desempenho de banco de dados relacionais com banco de dados não-relacionais. Eles observaram que para conjuntos de dados suficientemente grandes, o número de operações realizadas por um banco de dados relacional é várias vezes menor do que de bancos não-relacionais, onde MongoDB apresentou, por exemplo, a leitura mais rápida entre os bancos testados. Os autores ainda reforçam a tendência de os bancos de dados não-relacionais se tornarem cada vez mais usuais em decorrência do desenvolvimento da Internet e do uso de dispositivos móveis, que forçarão os desenvolvedores de *software* a aumentar o uso de banco de dados NoSQL

(FRACZEK; PLECHAWSKA-WOJCIK, 2017).

Convergindo com a ideia do presente trabalho, Makris e colaboradores (2021) utilizaram banco de dados com informações diárias. Nesse caso, foram utilizados dados espaciais, que aumentam de forma exponencial todos os dias. Os autores compararam o desempenho de banco de dados NoSQL e relacional (MongoDB e PostgreSQL) quanto ao tempo de resposta. Os autores observaram que o MongoDB respondeu mais rápido que o PostgreSQL em quase todas as consultas. Isso porque a indexação (realizada pelo MongoDB) afeta significativamente o tempo de resposta (MAKRIS et al., 2019).

A Tabela 2.1 apresenta uma visão geral dos trabalhos relacionados de acordo com algumas características do presente trabalho.

Tabela 2.1 – Tabela comparativa entre as características dos trabalhos relacionados encontrados na literatura. SQL vs. NoSQL: Comparações entre bancos relacionais e não-relacionais; NoSQL vs. NoSQL: Comparações entre bancos não relacionais; Storage: Avaliação de armazenamento; CRUD: Realiza testes e comparações para inserção, consulta, atualização e exclusão de dados, onde 4/4 indica que todos os parâmetros são contemplados; Propostas modelagem: Desenvolveu diferentes propostas de modelagem dos dados.

Autores, ano	**SQL vs. NoSQL**	**NoSQL vs. NoSQL**	**Storage**	**CRUD**	**Propostas modelagem**
(GUNAWAN; RAHMATULLOH; DARMAWAN, 2019)	Não	Sim	Não	4/4	Não
(VOKOROKOS; UCHNÁR; LEŠČIŠIN, 2016)	Não	Sim	Não	3/4	Não
(FRACZEK; PLECHAWSKA-WÓJCIK, 2016)	Não	Sim	Não	2/4	Não

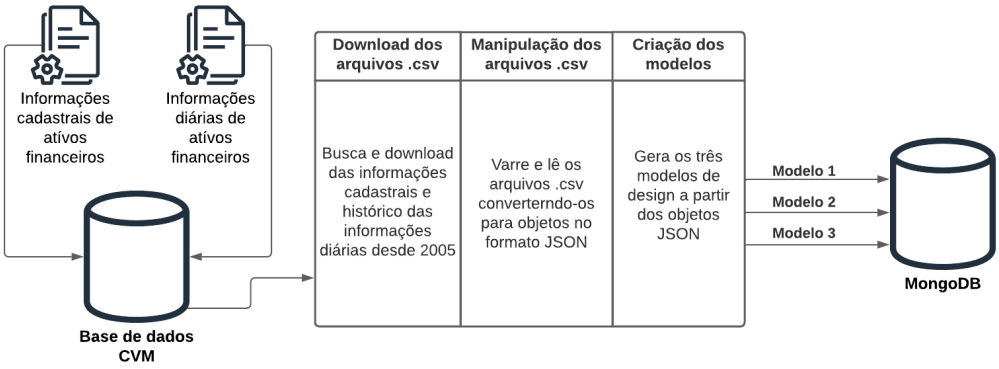
3 METODOLOGIA

3.1 Visão Geral

Bases de dados muito robustas podem apresentar melhor ou pior desempenho dependendo da maneira como os dados estão organizados. Encontrar a melhor forma de organizar um grande volume de dados, principalmente aqueles que recebem um aporte de informações diariamente, pode trazer benefícios para uma ampla gama de serviços e pessoas. Portanto, esse trabalho tem como principal intuito mostrar qual o impacto no desempenho que uma aplicação pode trazer quando modelada, projetada e implementada de diferentes formas e populadas com os mesmos dados.

Para isso, foram desenvolvidas três diferentes propostas de modelagem de dados para uma mesma base de dados. Em seguida, cada modelagem foi mapeada e implementada no MongoDB e submetida a uma bateria de testes para a avaliar o espaço de armazenamento ocupado, número de documentos gerados e tamanho (KB) de cada documento, o tempo de execução para realizar consultas, inserções, alterações e exclusões de informações diárias de um ativo financeiro no banco de dados. Os experimentos que foram realizados estão descritos a partir do Capítulo 4. O esquemático do funcionamento da aplicação que realizou o tratamento dos dados e o carregamento das modelagens está sintetizado e expresso na figura 3.1.

Figura 3.1 – Esquemático do funcionamento da aplicação.



3.2 Base de dados

Para o desenvolvimento deste trabalho, foi utilizada a base de dados abertos da Comissão de Valores Mobiliários (CVM). Esses dados contém informações cadastrais e diárias de ativos financeiros desde 2005, constituindo um conjunto volumoso de dados que atendem aos requisitos necessários para a implementação do trabalho. Os dados são disponibilizados em arquivos no formato CSV e estão disponíveis em http://dados.cvm.gov.br/dados/ (acesso em 02/10/2022).

O campo CNPJ está presente no arquivo de informações cadastrais e também no arquivo de informações diárias. As informações cadastrais são, por exemplo: nome, razão social, endereço, CNPJ. Neste trabalho, são utilizadas as informações cadastrais referentes ao nome e o CNPJ. As informações diárias, por sua vez, se referem às informações que são atualizadas diariamente, como por exemplo o preço de um ativo financeiro, patrimônio líquido da instituição, data de referência, valor de mercado entre outras. Neste trabalho, foi utilizado o preço de um ativo financeiro, patrimônio líquido da instituição e a data de referência.

Através do campo CNPJ foi possível juntar os dados do arquivo de informações cadastrais com os dados dos arquivos de informações diárias. Para a geração das propostas de modelagem, foi implementada uma aplicação utilizando a linguagem de programação NodeJS. Essa aplicação tem como objetivo realizar o *download* do histórico dos dados, converter os arquivos CSV *Comma Separated Values* em objetos JSON, manipular e implementar os modelos no banco de dados e popular o banco de dados MongoDB com as três propostas de modelagem de dados.

Para a carga do histórico de informações cadastrais dos ativos financeiros foi feito o *download* de um arquivo CSV com aproximadamente 70 mil linhas. Devido a alterações em informações de ativos ao longo do tempo, alguns CNPJs se repetiram, portanto, há aproximadamente 61 mil ativos financeiros diferentes. Para a carga de informações diárias dos ativos financeiros desde 2005 até os dias atuais, foram baixados 213 arquivos CSV, sendo um arquivo para cada mês, contendo entre 170 mil linhas e 500 mil linhas por arquivo. Devido ao grande volume de dados, foram necessários alguns procedimentos para não ocasionar estouro de memória ao fazer a carga dos dados no banco de dados MongoDB.

3.3 Implementação da Aplicação

O objetivo desta seção é apresentar a aplicação que foi desenvolvida para o tratamento dos dados e o carregamento das modelagens no banco de dados MongoDB. As etapas de desenvolvimento da aplicação serão descritas a seguir.

Para duas das três propostas de modelagem de dados criadas, foi possível alimentar o banco de dados separadamente - primeiro com as informações cadastrais e depois com as informações diárias. O processo de carga das informações cadastrais ocorreu da seguinte maneira:

- Leitura do arquivo `cad_fi.CSV` ;
- Conversão dos dados lidos no arquivo CSV em um lista de objetos JSON contendo somente as informações utilizadas na carga do banco de dados;
- Leitura da lista de objetos JSON, disparando as inserções no banco de dados MongoDB. Foram executadas mil chamadas ao banco de dados MongoDB por vez para evitar problemas de estouro de memória caso todas as chamadas ao banco de dados MongoDB fossem executadas em paralelo e também para não tornar o processo tão lento inserindo um de cada vez.

A modelagem de dados em que não foi possível popular o banco de dados com informações cadastrais e informações diárias separadamente, foram seguidos os seguintes passos:

- Ler e converter para JSON o arquivo de informações cadastrais, guardando em memória a lista com as informações a serem utilizadas;
- Ler e converter para JSON um arquivo de informações diárias por vez;
- Mesclar cada informação diária com suas determinadas informações cadastrais que estão guardadas em memória, gerando uma lista de objetos JSON a partir desta mescla de informações;
- Leitura da lista de objetos JSON, disparando as inserções no banco de dados MongoDB. Como neste caso o tamanho de cada objeto JSON é fixo e com poucas chaves, foi possível executar dez mil chamadas ao banco de dados por vez sem correr o risco de qualquer estouro de memória.

3.4 Modelagens de Dados

Nesta seção, serão apresentados as três modelagens de dados propostas para serem implementadas e avaliadas no MongoDB. De agora em diante, a modelagem 1 será chamada de P1-Lista, a modelagem 2 será chamada de P2-Chave-valor e a modelagem 3 será chamada de P3-Documento diário, onde P se refere a proposta de modelagem (1, 2 e 3) e a palavra em sequência indica uma característica atribuída a cada proposta de modelagem. Essas denominações foram feitas para auxiliar na compreensão dos experimentos e nas comparações realizadas entre as propostas de modelagem que serão discutidas no decorrer do trabalho.

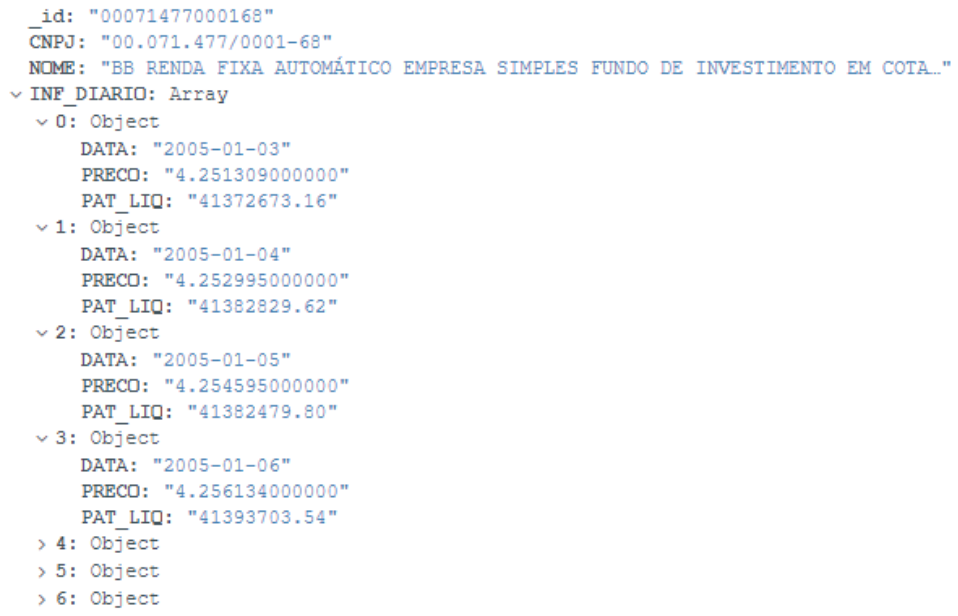
3.4.1 Modelagem 1: P1-Lista

O objetivo da P1-Lista foi verificar como o MongoDB se comporta ao realizar buscas, inserções e exclusões em *arrays*. Nesta primeira proposta de modelagem, o documento foi definido da seguinte forma:

- O campo único "`_id`"sendo composto pelo CNPJ do ativo;
- Informações cadastrais "`NOME`"e "`CNPJ`"como campos do tipo texto;
- Informações diárias no campo do tipo *array* denominado "`INF_DIARIO`", que contém um objeto para cada data com os campos “`DATA`”, “`PRECO`” e “ `PAT_LIQ`”.

Ao acrescentar um novo item diariamente no *array* de informações diárias, esta modelagem pode, mesmo que em um futuro distante, gerar problemas devido ao crescimento vertical do documento. Isso pode fazer com que o documento ultrapasse o tamanho máximo de 16MB. A figura 3.2 ilustra o exemplo de um documento que foi estruturado a partir da P1-Lista, onde as informações cadastrais estão representadas pelos campos “`NOME`” e “`CNPJ`” e as informações diárias estão representadas pelo campo “`INF_DIARIO`”.

Figura 3.2 – Exemplo da estruturação da P1-Lista.



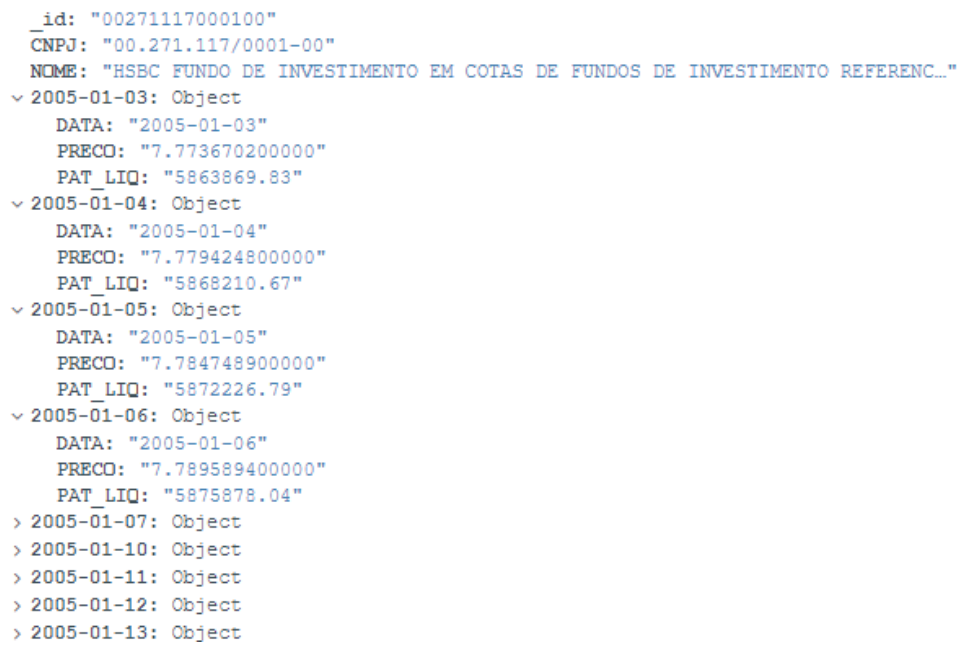
3.4.2 Modelagem 2: P2-Chave-valor

O objetivo da P2-Chave-valor foi verificar o comportamento do MongoDB diante de documentos com crescimento vertical, onde a cada dia um novo campo é adicionado ao documento. Na P2-Chave-valor o documento foi definido da seguinte forma:

- O campo único "`_id`" sendo composto pelo CNPJ do ativo;
- Informações cadastrais como "`NOME`"e "`CNPJ`"como campos do tipo texto;
- Informações diárias inseridas como campos do tipo objeto, utilizando como chave a data no formato "yyyy-MM-dd" e como valor um objeto com os campos "`DATA`", "`PREÇO`" e "`PAT_LIQ`".

Assim como na P1-Lista, a P2-Chave-valor também pode gerar um problema devido ao crescimento vertical do documento, uma vez que é gerado diariamente um novo campo com a informação diária do ativo financeiro. A figura 3.3 ilustra o exemplo de um documento que foi estruturado a partir da P2-Chave-valor, onde as informações cadastrais estão representadas pelos campos "NOME" e "CNPJ" e as informações diárias estão representadas pelos campos de datas no formato "yyyy-MM-dd".

Figura 3.3 – Exemplo da estruturação da P2-Chave-valor.



3.4.3 Modelagem 3: P3-Documento diário

O objetivo da P3-Documento diário é verificar o comportamento do MongoDB diante de um grande volume de documentos, porém cada documento apresentando um tamanho fixo. O documento foi definido da seguinte forma:

- Informações cadastrais ("NOME"e "CNPJ") do ativo financeiro foram repetidas em cada uma das informações diárias, criando um documento para cada data;
- Campo único "_id" sendo definido pelo CNPJ do ativo financeiro concatenado com a data da informação diária separados por um "_";
- Informações cadastrais ("NOME"e "CNPJ") como campos do tipo texto;
- Informações diárias ("DTPOSICAO", "PRECO","PAT_LIQ") também como campos do tipo texto.

Neste caso, não há o risco de ocorrer um problema de crescimento vertical do documento, pois este apresenta um tamanho fixo. Porém, as informações cadastrais serão repetidas para cada data. Por se tratar de um banco de dados não-relacional, o grande volume de dados gerados por essa proposta de modelagem pode não ser um problema, uma vez que este tipo de banco de dados foi desenvolvido para lidar com bases de dados robustas. A figura 3.4 ilustra o exemplo de um documento que foi estruturado a partir da P3-Documento diário, onde as informações cadastrais estão representadas pelos

campos "NOME" e "CNPJ" e as informações diárias estão representadas pelos campos "DTPOSICAO", "PRECO" e "PAT_LIQ".

Figura 3.4 – Exemplo da estruturação da P3-Documento diário


```
_id: "00000432000100_2005-01-06"
CNPJ: "00.000.432/0001-00"
NOME: "COPESUL  - FUNDO DE INVESTIMENTO MULTIMERCADO  -  CREDITO PRIVADO"
DTPOSICAO: "2005-01-06"
PRECO: "18.062034000000"
PAT_LIQ: "87216017.10"
```

4 AVALIAÇÃO EXPERIMENTAL

4.1 Descrição do Problema

O objetivo desta seção é apresentar quais experimentos foram realizados e quais métricas foram utilizadas para comparar as diferentes propostas de modelagem. Além disso, também são apresentados os equipamentos e *softwares* utilizados no desenvolvimento do trabalho.

4.1.1 Metodologia do Experimentos

Para avaliar o desempenho das três diferentes modelagens propostas, foram elaborados os seguintes experimentos com o objetivo de verificar:

- Experimento I: o espaço de armazenamento utilizado em cada uma das modelagens com a mesma carga de dados;
- Experimento II: o número de documentos gerados em cada uma das modelagens e tamanho médio em KB de cada documento;
- Experimento III: o tempo de execução da consulta de uma informação diária de um ativo financeiro;
- Experimento III: o tempo de execução da consulta de uma informação diária de um ativo financeiro;
- Experimento IV: o tempo de execução ao inserir uma nova informação diária de um ativo financeiro;
- Experimento V: o tempo de execução de uma alteração em uma informação diária de um ativo financeiro;
- Experimento VI: o tempo de execução da exclusão de uma informação diária de um ativo financeiro;

4.1.2 Execução e métricas

As métricas utilizadas para comparar o desempenho das três propostas de modelagem foram o espaço de armazenamento ocupado, número de documentos gerados e

tamanho (KB) de cada documento, o tempo de execução para realizar consultas, inserções, alterações e exclusões de informações diárias de um ativo financeiro no banco de dados. Para as métricas que envolvem tempo de execução (consultas, inserções, alterações e exclusões), os testes foram rodados 10 vezes utilizando ativos financeiros e datas diferentes para cada uma das três propostas de modelagem. A partir desses 10 testes, foi calculado o tempo médio de resposta para cada métrica.

4.1.3 Equipamento e Software

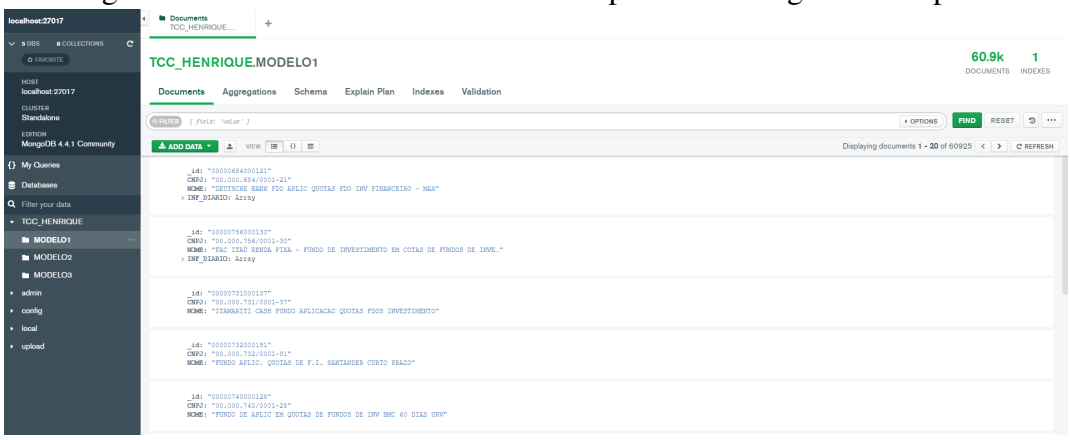
Os experimentos foram realizados em um equipamento com as seguintes especificações:

- Processador: Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz 3.70 GHz
- Memória RAM: 16,0 GB
- Sistema Operacional: Windows 10 Home 64bits

Os experimentos foram executados utilizando os seguintes aplicativos:

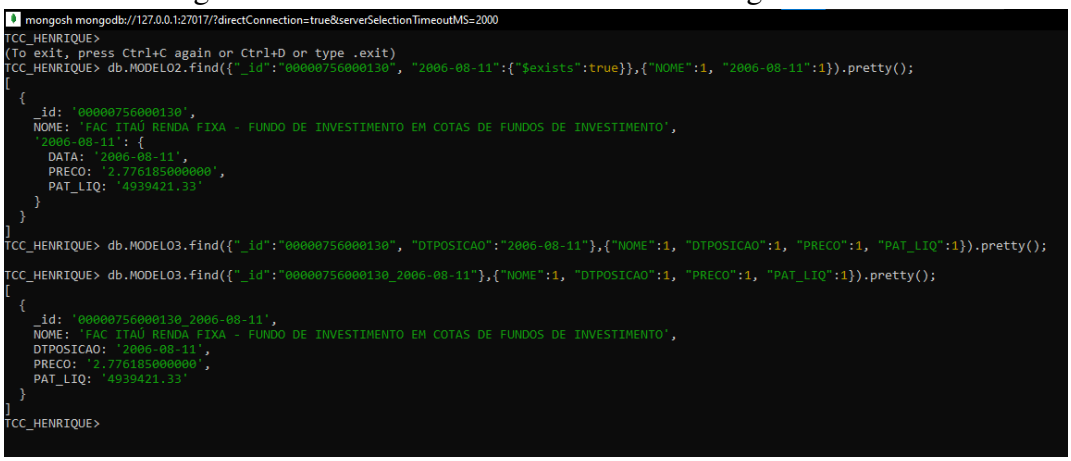
- MongoDB Compass - É uma ferramenta utilizada para consultar, analisar e visualizar seus dados de forma mais interativa no banco de dados MongoDB (Figura 4.1).

Figura 4.1 – *PrintScreen* da interface do aplicativo MongoDB Compass.



- MongoDB Shell - é um terminal de linha de comando utilizado para configurar e interagir com seus dados no banco de dados MongoDB (Figura 4.2).

Figura 4.2 – *PrintScreen* da interface do MongoDB Shell.



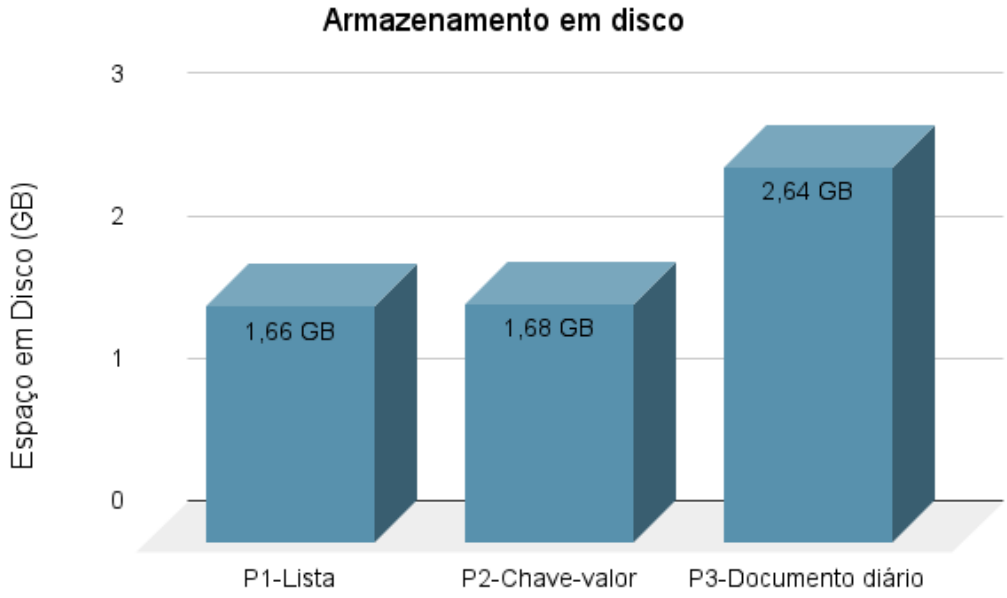
4.2 Experimentos e Resultados

O objetivo desta seção é descrever como cada um dos seis experimentos foi estruturado e executado, bem como os resultados obtidos em cada experimento. Por fim, ao final desta seção, é apresentada uma análise geral de todos os resultados obtidos em todos os experimentos.

4.2.1 Experimento I: Armazenamento

Este experimento tem como objetivo verificar o espaço de armazenamento utilizado em cada uma das modelagens, utilizando a mesma base de dados. A Figura 4.3 ilustra o espaço de armazenamento em disco utilizado com cada uma das modelagens propostas. A P1-Lista e a P2-Chave-valor apresentaram resultados semelhantes (1.66 GB e 1.68 GB respectivamente) e ambos diferiram da P3-Documento diário (2.64 GB) quanto ao armazenamento em disco. A diferença entre a P1-Lista e a P2-Chave-valor se deve ao fato de na P2-Chave-valor a cada informação diária inserida, uma nova chave é adicionada ao documento, enquanto na P1-Lista esta informação diária é inserida diretamente na lista do campo "`INF_DIARIO`". A grande diferença de armazenamento ocupado em disco da P3-Documento diário em comparação com as outras duas modelagens era esperada, pois na P3-Documento diário as informações cadastrais são repetidas a cada nova informação diária.

Figura 4.3 – Armazenamento em disco utilizado por cada uma das propostas de modelagem de dados.



4.2.2 Experimento II: Número de documentos

Este experimento tem por objetivo comparar o número de documentos gerados em cada uma das modelagens, considerando a mesma base de dados, e também o tamanho médio em KB de cada um dos documentos.

Na P1-Lista e na P2-Chave-valor, o número de documentos é igual ao número de ativos financeiros existentes. Nesse caso, o número de documentos aumenta conforme surgem novos ativos financeiros. Porém, o tamanho médio dos documentos aumenta com o passar dos dias, pois a cada dia uma nova informação diária é inserida no documento de cada ativo financeiro que ainda esteja em funcionamento. Na P3-Documento diário, o número de documentos aumenta de forma significativa a cada dia, pois a cada informação diária de um ativo financeiro, é criado um novo documento no banco de dados. No entanto, o tamanho médio de cada documento permanece sempre o mesmo, pois cada documento tem sempre o mesmo número de informações. A Tabela 4.1 ilustra os resultados anteriormente discutidos, apresentando o número de documentos e o tamanho médio de um documento para cada proposta de modelagem.

Tabela 4.1 – Número de documentos gerados em cada uma das propostas de modelagem de dados e tamanho médio de cada documento.

Modelagem	Número de documentos	Tamanho médio de um documento
P1-Lista	60.9 K	73.95 KB
P2-Chave-valor	60.9 K	79.83 KB
P3-Documento diário	54.2 M	0.262 KB

4.2.3 Experimento III: Consultas em informações diárias

Este experimento tem como objetivo verificar o tempo de execução da consulta de uma informação diária de um ativo. Para cada modelagem, foram realizadas 10 consultas utilizando o índice *default* do MongoDB (campo "`_id`") e 10 consultas sem o uso do campo "`_id`". No experimento em que não foi utilizado o campo "`_id`", utilizou-se o campo "`CNPJ`" na busca do ativo financeiro. Neste caso, o MongoDB precisa percorrer todos os documentos da coleção para encontrar o ativo.

As Figuras 4.4, 4.5 e 4.6 mostram um exemplo de *query* de consulta que foi executada para cada uma das três propostas de modelagem utilizando o índice "`_id`" como condição de busca.

A Figura 4.4 corresponde a *query* de consulta da P1-Lista utilizando o índice "`_id`", onde a linha 3 representa as condições de busca, a linha 4 representa a projeção de quais campos serão retornados na resposta e entre as linhas 7 e 17 encontra-se a resposta da *query* executada.

Figura 4.4 – Exemplo de *query* de consulta da P1-Lista utilizando índice.

```
1  //QUERY
2  db.MODELO1.find(
3      {"_id":"00000756000130","INF_DIARIO.DATA": "2006-08-11"},
4      {"NOME": 1,"INF_DIARIO.$": 1});
5  //RESPOSTA
6  [
7      {
8          _id: '00000756000130',
9          NOME: 'FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO',
10         INF_DIARIO: [
11             {
12                 DATA: '2006-08-11',
13                 PRECO: '2.776185000000',
14                 PAT_LIQ: '4939421.33'
15             }
16         ]
17     }
18 ]
```

A Figura 4.5 corresponde a *query* de consulta da P2-Chave-valor utilizando o

índice "_id", onde a linha 3 representa as condições de busca, a linha 4 representa a projeção de quais campos serão retornados na resposta e entre as linhas 7 e 15 encontra-se a resposta da *query* executada.

Figura 4.5 – Exemplo de *query* de consulta da P2-Chave-valor utilizando índice.

```
1  //QUERY
2  db.MODELO2.find(
3      {"_id":"00000756000130", "2006-08-11":{"$exists":true}},
4      {"NOME":1, "2006-08-11":1})
5  //RESPOSTA
6  [
7    {
8      _id: '00000756000130',
9      NOME: 'FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO',
10     '2006-08-11': {
11       DATA: '2006-08-11',
12       PRECO: '2.776185000000',
13       PAT_LIQ: '4939421.33'
14     }
15   }
16 ]
```

A Figura 4.6 corresponde a *query* de consulta da P3-Documento diário utilizando o índice "_id", onde a linha 3 representa as condições de busca, a linha 4 representa a projeção de quais campos serão retornados na resposta e entre as linhas 7 e 13 encontra-se a resposta da*query* executada.

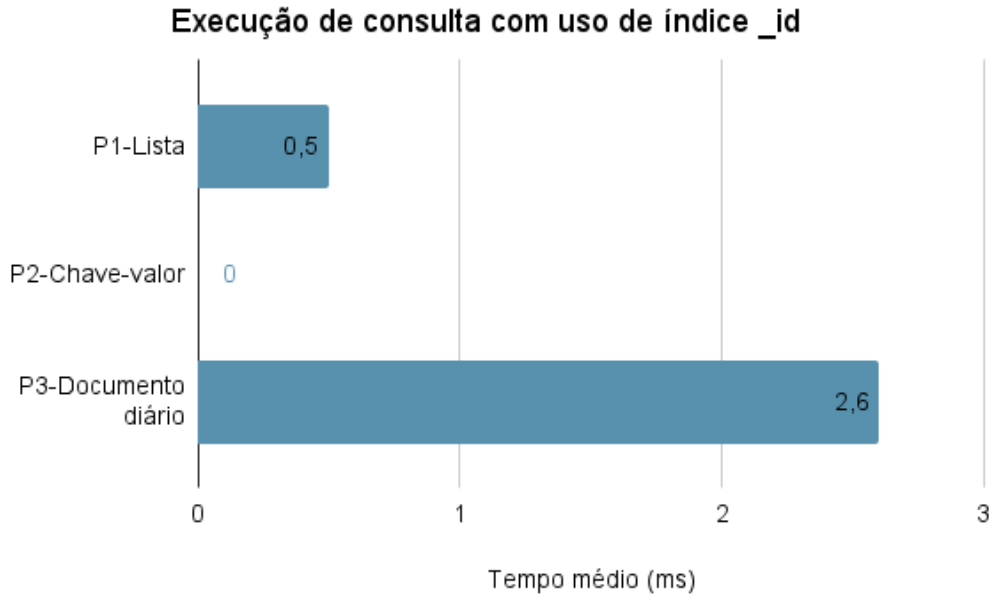
Figura 4.6 – Exemplo de *query* de consulta da P3-Documento diário utilizando índice.

```
1  //QUERY
2  db.MODELO3.find(
3    {"_id":"00000756000130_2006-08-11"},
4    {"NOME":1, "DTPOSICAO":1, "PRECO":1, "PAT_LIQ":1});
5  //RESPOSTA
6  [
7    {
8      _id: '00000756000130_2006-08-11',
9      NOME: 'FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO',
10     DTPOSICAO: '2006-08-11',
11     PRECO: '2.776185000000',
12     PAT_LIQ: '4939421.33'
13   }
14 ]
```

As consultas foram feitas para diferentes ativos financeiros e datas. Sendo assim, foram selecionados os tempos médios para cada modelagem. Com a utilização do campo "_id", a execução da consulta sobre P1-Lista levou em média 0,5 ms para executar a

consulta (Figura 4.7) e a execução da P2-Chave-valor realizou todas as consultas instantaneamente (Figura 4.7). A consulta sobre a P3-Documento diário levou em média 2,3 ms para executar (Figura 4.7).

Figura 4.7 – Tempo médio de execução da consulta de uma informação diária de um ativo financeiro em cada uma das três modelagens de dados.



É possível perceber que com o uso do índice o MongoDB faz buscas quase instantaneamente. No entanto, há uma grande diferença no tempo médio de resposta na execução das consultas sobre a P1-Lista e a P2-Chave-valor comparadas a P3-Documento diário. A base de dados da P1-Lista e da P2-Chave-valor possui aproximadamente 61 mil documentos em sua coleção, enquanto que a base de dados da P3-Documento diário possui 54.1 milhões de documentos, levando aproximadamente três vezes mais que a P1-Lista e a P2-Chave-valor para realizar a busca.

As Figuras 4.8, 4.9 e 4.10 apresentam um exemplo de *query* de consulta que foi executada sem a utilização do índice "`_id`" para cada uma das três propostas de modelagem. Neste caso, o campo "`CNPJ`" foi utilizado como condição de busca. A Figura 4.8 corresponde a *query* de consulta da P1-Lista sem o uso do índice "`_id`", onde a linha 3 representa as condições de busca, a linha 4 representa a projeção de quais campos serão retornados na resposta e entre as linhas 7 e 17 encontra-se a resposta da *query* executada.

Figura 4.8 – Exemplo de *query* de consulta da P1-Lista sem a utilização de índice.

```
1  //QUERY
2  db.MODELO1.find(
3      {"CNPJ":"00.000.756/0001-30","INF_DIARIO.DATA": "2006-08-11"},
4      {"NOME": 1,"INF_DIARIO.$": 1});
5  //RESPOSTA
6  [
7      {
8          _id: '00000756000130',
9          NOME: 'FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO',
10         INF_DIARIO: [
11             {
12                 DATA: '2006-08-11',
13                 PRECO: '2.776185000000',
14                 PAT_LIQ: '4939421.33'
15             }
16         ]
17     }
18 ]
```

A Figura 4.9 corresponde a *query* de consulta da P2-Chave-valor sem o uso do índice "`_id`", onde a linha 3 representa as condições de busca, a linha 4 representa a projeção de quais campos serão retornados na resposta e entre as linhas 7 e 15 encontra-se a resposta da *query* executada.

Figura 4.9 – Exemplo de *query* de consulta da P2-Chave-valor sem a utilização de índice.

```
1  //QUERY
2  db.MODELO2.find(
3      {"CNPJ":"00.000.756/0001-30", "2006-08-11":{"$exists":true}},
4      {"NOME":1, "2006-08-11":1})
5  //RESPOSTA
6  [
7    {
8      _id: '00000756000130',
9      NOME: 'FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO',
10     '2006-08-11': {
11       DATA: '2006-08-11',
12       PRECO: '2.776185000000',
13       PAT_LIQ: '4939421.33'
14     }
15   }
16 ]
```

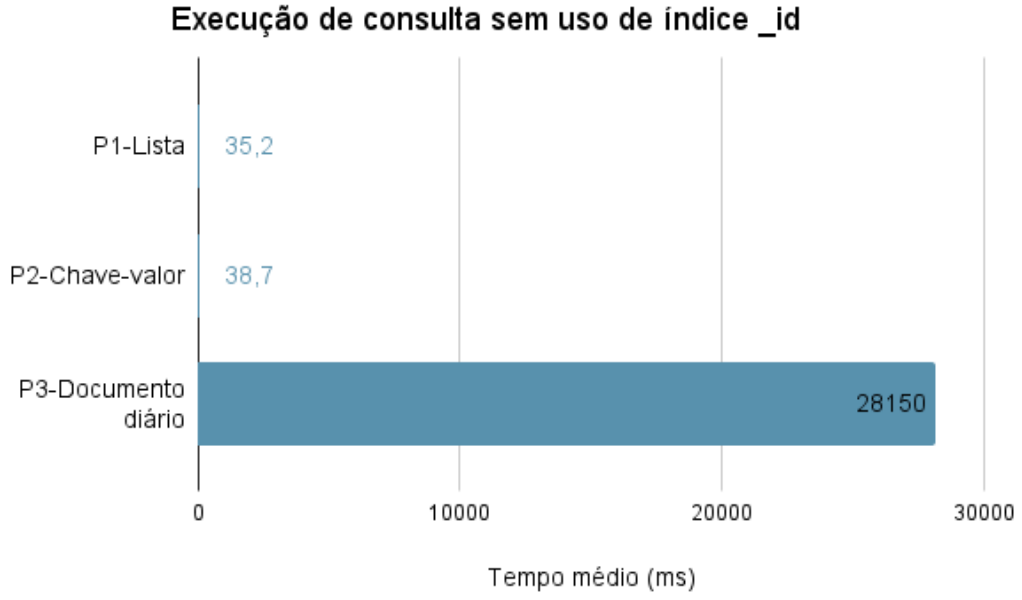
A Figura 4.10 corresponde a *query* de consulta da P3-Documento diário sem o uso do índice "`_id`", onde a linha 3 representa as condições de busca, a linha 4 representa a projeção de quais campos serão retornados na resposta e entre as linhas 7 e 13 encontra-se a resposta da *query* executada.

Figura 4.10 – Exemplo de *query* de consulta da P3-Documento diário sem a utilização de índice.

```
//QUERY
db.MODELO3.find(
  {"CNPJ":"00.000.756/0001-30", "DTPOSICAO":"2006-08-11"},
  {"NOME":1, "DTPOSICAO":1, "PRECO":1, "PAT_LIQ":1});
//RESPOSTA
[
  {
    _id: '00000756000130_2006-08-11',
    NOME: 'FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO',
    DTPOSICAO: '2006-08-11',
    PRECO: '2.776185000000',
    PAT_LIQ: '4939421.33'
  }
]
```

Utilizando o campo "CNPJ" nas buscas, os resultados foram muito diferentes (Figura 4.11). As consultas sobre a P1-Lista e a P2-Chave-valor levaram em média, respectivamente, 35,2 e 38,7 ms para serem executadas. A execução da consulta sobre a P3-Documento diário, por sua vez, levou centenas de vezes mais tempo que as outras modelagens, levando em média 28.150 ms para ser executada.

Figura 4.11 – Tempo médio de execução da consulta de uma informação diária de um ativo financeiro em cada uma das três propostas de modelagem de dados sem a utilização de índice.



O tempo médio semelhante entre as consultas executadas entre a P1-Lista e P2-Chave-valor deve-se ao fato de que ambas apresentam o mesmo número de documentos.

O resultado observado na execução da consulta sobre a P3-Documento diário ocorreu pelo fato de a função de busca ter que percorrer todos os 54.1 milhões de documentos existentes em sua coleção para executar a consulta.

4.2.4 Experimento IV: Inserção de nova informação diária

Este experimento tem como objetivo simular como uma nova informação diária de um ativo financeiro é inserida no banco de dados com o passar dos dias e analisar o tempo de execução das inserções para cada uma das implementações das três modelagens. Para isso foram executadas 10 inserções para cada uma das modelagens, selecionando a média dos tempos de cada um. As inserções para cada uma das modelagens foram feitas da seguinte maneira:

- P1-Lista - foram executadas *queries* de busca e atualização de um documento. A busca foi feita utilizando o campo "`CNPJ`", e inserindo ao campo do tipo *array* "`INF_DIARIO`" um novo item com as novas informações diárias.
- P2-Chave-valor - foram executadas *queries* de busca e atualização de um documento. A busca foi feita utilizando o campo "`CNPJ`", e inserindo um novo campo no documento, este tendo como chave a data da informação diária no formato yyyy-MM-dd e como valor o objeto com as novas informações diárias.
- P3-Documento diário - foram executadas *queries* de inserção, criando novos documentos com as novas informações cadastrais e diárias de cada um dos 10 ativos financeiros.

As Figuras 4.12, 4.13 e 4.14 mostram um exemplo de *query* de inserção de uma nova informação diária que foi executada para cada uma das três propostas de modelagem.

A Figura 4.12 corresponde a *query* da P1-Lista para a inserção de uma nova informação diária, onde a linha 3 representa as condições de busca, e entre as linhas 4 e 6 é definida a operação de inserção da nova informação diária na lista representada pelo campo "`INF_DIARIO`".

A Figura 4.13 corresponde a *query* da P2-Chave-valor para a inserção de uma nova informação diária, onde a linha 3 representa as condições de busca e entre as linhas 4 e 6 é definida a operação de criação de uma nova chave com nova informação diária.

Figura 4.12 – Exemplo de *query* da P1-Lista para a inserção de uma nova informação diária.

```
1  //QUERY
2  db.MODELO1.findOneAndUpdate(
3   {"CNPJ":"00.000.756/0001-30"},
4   {"$push":{"INF_DIARIO":
5     {"DATA":"2022-10-05", "PRECO":"1.00","PAT_LIQ":"100.00"}}
6   })
```

Figura 4.13 – Exemplo de *query* da P2-Chave-valor para a inserção de uma nova informação diária.

```
1  //QUERY
2  db.MODELO2.findOneAndUpdate(
3    {"CNPJ":"00.000.756/0001-30"},
4    {"$set":{"2022-10-05":
5      {"DATA":"2022-10-05", "PRECO":"1.00","PAT_LIQ":"100.00"}}
6    })
```

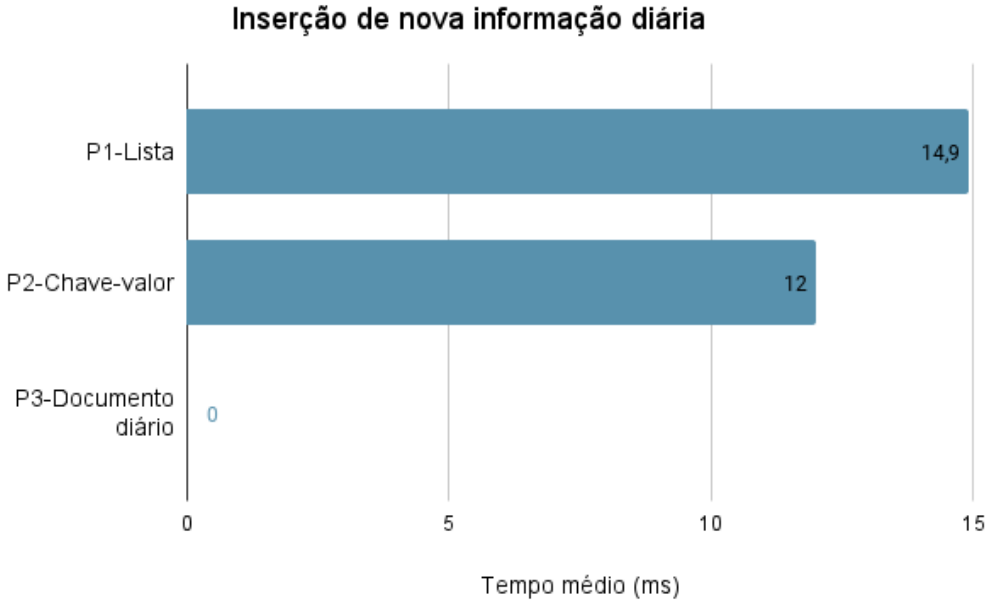
A Figura 4.14 corresponde a *query* da P3-Documento diário para a inserção de uma nova informação diária. Entre as linhas 3 e 8 é definido o novo documento com a informação cadastral e a nova informação diária a ser inserido no banco de dados.

Figura 4.14 – Exemplo de *query* da P3-Documento diário para a inserção de uma nova informação diária.

```
1  //QUERY
2  db.MODELO3.insertOne({
3    "_id":"00000756000130_2022-10-01",
4    "NOME": "FAC ITAÚ RENDA FIXA - FUNDO DE INVESTIMENTO EM COTAS DE FUNDOS DE INVESTIMENTO",
5    "CNPJ":"00.000.756/0001-30",
6    "DTPOSICAO":"2022-10-01",
7    "PRECO":"1.00",
8    "PAT_LIQ":"100.00"
9  })
```

O tempo médio para a inserção de uma nova informação diária para a P1-Lista e a P2-Chave-valor foi de 14,9 ms e 12 ms, respectivamente. Para a P3-Documento diário, as inserções foram feitas instantaneamente (Figura 4.15).

Figura 4.15 – Tempo médio de execução da inserção de uma nova informação diária de um ativo financeiro em cada uma das três modelagens de dados.



O desempenho do MongoDB para a atualização de um campo do tipo *array* (P1-Lista) é bastante semelhante ao de uma atualização de um campo do tipo chave-valor (P2-Chave-valor), apresentando uma diferença de apenas 2,9 ms. Para a P3-Documento diário, a inserção é instantânea, pois é realizada apenas a inserção de um novo documento, não sendo necessário fazer buscas no banco de dados.

4.2.5 Experimento V: Alteração em uma informação diária

Este experimento tem como objetivo simular uma alteração em uma informação diária de um ativo financeiro e analisar o tempo de execução das alterações para cada uma das três modelagens. Para isso, foram executadas 10 alterações para cada uma das três modelagens, selecionando a média dos tempos de cada um. As alterações para cada modelagem foram feitas da seguinte maneira:

- P1-Lista - foram executadas duas *queries* de busca e atualização de um documento para cada uma das alterações. As buscas foram feitas utilizando o campo "`CNPJ`". Na primeira *query* foi preciso percorrer o campo do tipo *array* "`INF_DIARIO`" e excluir o item que possuía o valor do campo "`DATA`" igual a data que seria alterada. A segunda *query* foi responsável por inserir o novo item no campo "`INF_DIARIO`"

com a informação diária atualizada.

- P2-Chave-valor - foram executadas *queries* de busca e atualização de um documento. A busca foi feita utilizando o campo "CNPJ" e foi inserido um campo com chave igual a data que seria alterada no documento, tendo como valor o objeto com a informação diária atualizada. Para esta modelagem, foi utilizada a mesma estrutura de *query* tanto na inserção de uma nova informação diária (Experimento V) quanto na alteração em uma informação diária existente.
- P3-Documento diário - foram executadas *queries* de busca e atualização de um documento. A busca foi feita utilizando o campo "CNPJ" e o campo "DATA", inserindo as informações diárias atualizadas que sobrescrevem as informações diárias existentes no documento.

As Figuras 4.16, 4.17 e 4.18 mostram um exemplo de *query* de alteração de uma informação diária que foi executada para cada uma das três propostas de modelagem.

A Figura 4.16 corresponde às duas *queries* necessárias para a P1-Lista conseguir executar a alteração de uma informação diária, onde a linha 3 representa a condição de busca da primeira *query* e as linhas 4 e 5 correspondem a operação de remoção da informação diária na qual deseja-se fazer a alteração. A linha 9 representa a condição de busca da segunda *query*, e as linhas 10 e 11 correspondem a inserção da informação diária com o valor atualizado na lista representada pelo campo "INF_DIARIO".

Figura 4.16 – Exemplo de *query* da P1-Lista para a alteração de uma informação diária.

```
1   //QUERY 1
2   db.MODELO1.findOneAndUpdate(
3     {"CNPJ":"00.000.756/0001-30"},
4     {"$pull":{"INF_DIARIO":
5     {"DATA":"2022-10-05"}}
6   })
7   //QUERY 2
8   db.MODELO1.findOneAndUpdate(
9     {"CNPJ":"00.000.756/0001-30"},
10    {"$push":{"INF_DIARIO":
11    {"DATA":"2022-10-05", "PRECO":"1.00", "PAT_LIQ":"100.00"}}
12  })
```

A Figura 4.17 corresponde a *query* da P2-Chave-valor para a atualização de uma informação diária. A linha 3 representa a condição de busca, e nas linhas 4 e 5 é definida

a operação de atualização de uma chave já existente com a informação diária atualizada.

Figura 4.17 – Exemplo de *query* da P2-Chave-valor para a alteração de uma informação diária.

```
1  //QUERY
2  db.MODELO2.findOneAndUpdate(
3    {"CNPJ":"00.000.756/0001-30"},
4    {"$set":{"2022-10-05":
5      {"DATA":"2005-01-10", "PRECO":"1.00","PAT_LIQ":"100.00"}}
6  })
```

A Figura 4.18 corresponde a *query* da P3-Documento diário para a atualização de uma informação diária. A linha 3 representa as condições de busca, e na linhas 4 é definida a informação diária atualizada.

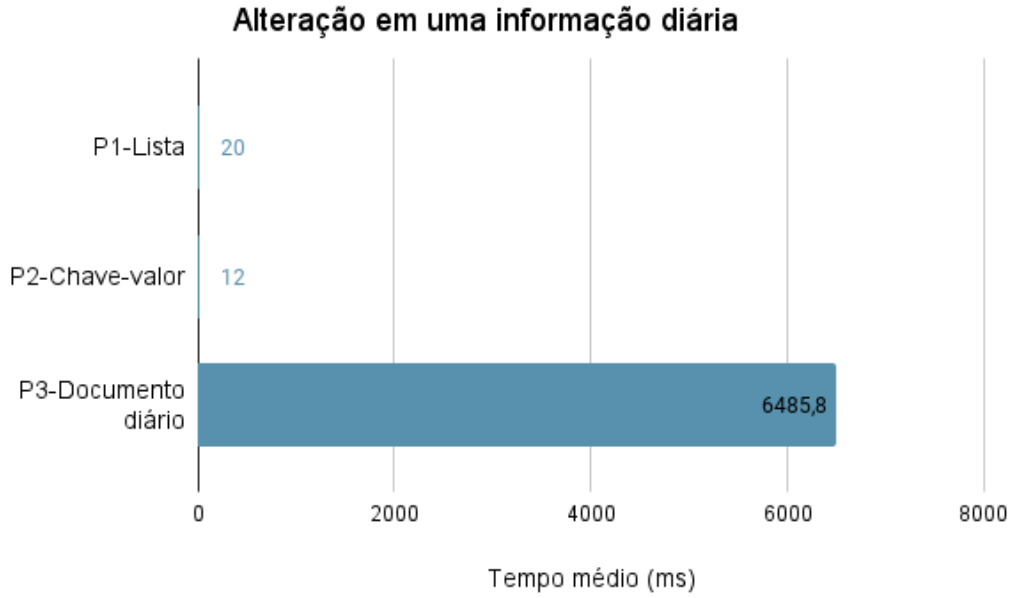
Figura 4.18 – Exemplo de *query* da P3-Documento para a alteração de uma informação diária.

```
1  //QUERY
2  db.MODELO3.findOneAndUpdate(
3    {"CNPJ":"00.000.756/0001-30", "DTPOSICAO":"2022-10-05"},
4    {"$set":{"PRECO":"1.00","PAT_LIQ":"100.00"}
5  })
```

O tempo médio para a alteração de uma informação diária na implementação das modelagens P1-Lista, P2-Chave-valor e P3-Documento diário foi de 20 ms, 12 ms e 6.485,8 ms, respectivamente (Figura 4.19).

Foi possível verificar uma diferença de quase duas vezes o tempo médio para a alteração de uma informação diária entre as implementações da P1-Lista e da P2-Chave-valor. Isso ocorreu porque para a P1-Lista foi preciso fazer duas *queries* para executar a alteração de uma informação diária, enquanto para a implementação da P2-Chave-valor apenas uma *query* foi suficiente para realizar a alteração. A diferença exorbitante na média de tempo para a implementação da P3-Documento diário deve-se ao número de

Figura 4.19 – Tempo médio de execução da alteração de uma informação diária de um ativo financeiro em cada uma das três propostas de modelagem de dados.



documentos existentes em sua coleção (54 milhões). Assim, o tempo de execução da *query* depende da posição em que o documento se encontra no banco de dados, podendo levar mais ou menos tempo. A escolha dos documentos a serem alterados tentou abranger situações em que a *query* levasse mais e menos tempo para realizar as alterações.

4.2.6 Experimento VI - Exclusão de uma informação diária

Este experimento tem como objetivo simular uma exclusão de uma informação diária de um ativo financeiro e analisar o tempo de execução das exclusões para cada uma das modelagens de dados. Para isso, foram executadas 10 exclusões para cada uma das implementações das modelagens propostas, selecionando a média dos tempos de cada uma. As exclusões para cada proposta de modelagem foram feitas da seguinte maneira:

- P1-Lista - foi executada uma *query* de busca e atualização de um documento. A busca foi feita utilizando o campo "`CNPJ`". Para a exclusão, foi preciso percorrer o campo do tipo *array* "`INF_DIARIO`" e excluir o item que possuía o valor do campo "`DATA`" igual a data da informação que seria excluída.
- P2-Chave-valor - foi executada uma *query* de busca e atualização de um documento. A busca foi feita utilizando o campo "`CNPJ`". Para a exclusão, foi preciso informar

a data a ser excluída no formato "yyyy-MM-dd", junto ao comando "`$unset`" do MongoDB, que é responsável por fazer a exclusão do campo no documento.

- P3-Documento diário - Foi executada uma *query* de remoção de um documento, onde é feita uma busca utilizando os campos "`CNPJ`" e "`DATA`", e a remoção do documento assim que este foi encontrado.

As Figuras 4.20, 4.21 e 4.22 mostram um exemplo de *query* de exclusão de uma informação diária que foi executada para cada uma das três propostas de modelagem.

A Figura 4.20 corresponde a *query* da P1-Lista para a exclusão de uma informação diária, onde a linha 3 representa a condição de busca, e nas linhas 4 e 5 é definida a operação responsável pela exclusão da informação diária na lista representada pelo campo "`INF_DIARIO`".

Figura 4.20 – Exemplo de *query* da P1-Lista para a exclusão de uma informação diária.

```
1  //QUERY 1
2  db.MODELO1.findOneAndUpdate(
3    {"CNPJ":"00.000.756/0001-30"},
4    {"$pull":{"INF_DIARIO":
5    {"DATA":"2005-01-10"}}
6  })
```

A Figura 4.21 corresponde a *query* da P2-Chave-valor para a exclusão de uma informação diária, onde a linha 3 representa a condição de busca, e na linha 4 é definida a operação responsável pela exclusão da chave que contém a informação diária que deseja-se excluir.

A Figura 4.22 corresponde a *query* da P3-Documento diário para a exclusão de uma informação diária, onde a linha 3 representa a condição de busca do documento que deseja-se excluir.

Figura 4.21 – Exemplo de *query* da P2-Chave-valor para a exclusão de uma informação diária.

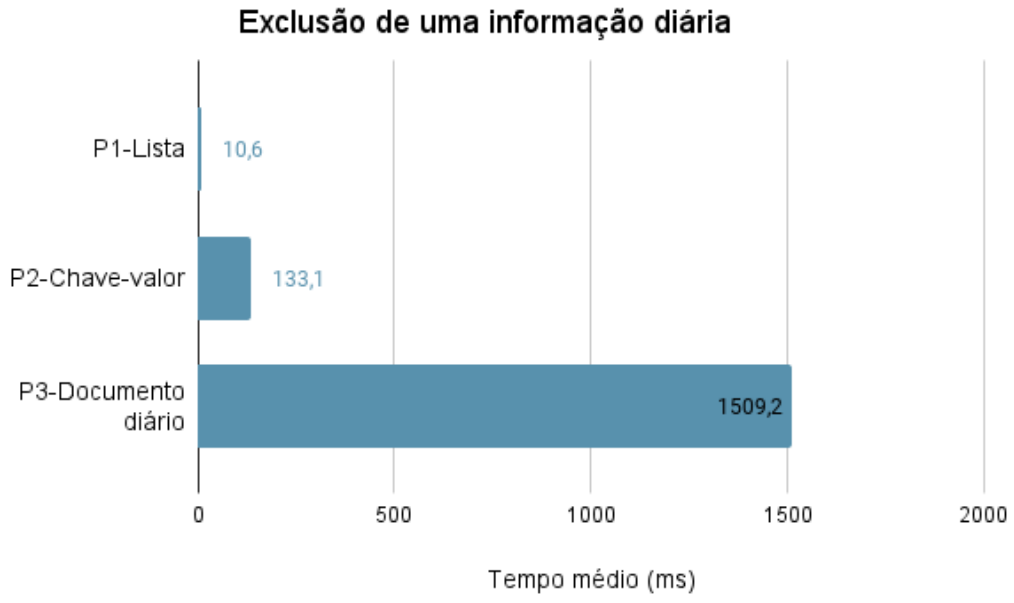
```
1  //QUERY
2  db.MODELO2.findOneAndUpdate(
3    {"CNPJ":"00.000.756/0001-30"},
4    {"$unset":{"2022-10-05":""}
5  })
```

Figura 4.22 – Exemplo de *query* da P3-Documento diário para a exclusão de uma informação diária.

```
1  //QUERY
2  db.MODELO3.deleteOne({
3    "CNPJ":"00.000.756/0001-30", "DTPOSICAO":"2022-10-05"
4  })
```

O tempo médio para a exclusão de uma informação diária na implementação das modelagens P1-Lista, P2-Chave-valor e P3-Documento diário foi de 10,6 ms, 133,1 ms e 1.509,2 ms, respectivamente (Figura 4.23).

Figura 4.23 – Tempo médio de execução da exclusão de uma informação diária de um ativo financeiro em cada uma das três propostas de modelagem de dados.



Houve uma grande diferença entre o tempo de execução da implementação da P1-Lista e P2-Chave-valor, onde foi possível observar que o desempenho do MongoDB é melhor ao excluir um item de uma lista do que ao excluir um campo com chave-valor. A execução da implementação da P3-Documento diário foi mais uma vez a mais lenta das três modelagens propostas devido ao grande volume de documentos presentes em sua coleção.

4.3 ANÁLISE GERAL DOS RESULTADOS

Um ponto importante que deve ser levado em consideração ao definir qual modelagem apresenta o melhor desempenho é que a escolha sempre vai depender dos objetivos e recursos disponíveis por parte de quem estiver desenvolvendo a aplicação. Neste trabalho, o desempenho das três diferentes propostas de modelagem foi avaliado quanto ao armazenamento em disco, o tempo médio de execução de consultas, inserções, alterações e exclusões de informações diárias. De maneira geral, a P1-Lista e a P2-Chave-valor apresentaram resultados mais semelhantes entre si quando comparados a P3-Documento diário em todas as métricas analisadas.

Quanto ao armazenamento em disco, a P1-Lista e a P2-Chave-valor apresentaram melhor desempenho em relação a P3-Documento diário, ocupando menos espaço. Essa diferença ocorreu devido a repetição de informações cadastrais a cada nova informação diária que ocorre na P3-Documento diário. Sendo assim, se o armazenamento é um fator determinante para quem estiver desenvolvendo a aplicação, a P1-Lista e a P2-Chave-valor apresentaram melhor desempenho. Com relação ao número de documentos e tamanho médio de um documento em cada coleção, a P1-Lista e a P2-Chave-valor apresentaram o mesmo número de documentos, que foi inferior a P3-Documento diário. O tamanho médio de um documento foi menor na P3-Documento diário, pois esta modelagem possui documentos de tamanho fixo. Na P1-Lista e P2-Chave-valor, o tamanho médio de um documento aumenta ao passar dos dias, pois uma informação diária é inserida no documento diariamente. O aumento no tamanho médio do documento pode não ser considerado um problema uma vez que o tamanho máximo de um documento é de 16 MB. Sendo assim, seriam necessários muitos anos para que a P1-Lista e a P2-Chave-valor alcançassem esse limite.

Nas consultas de uma informação diária realizadas no experimento III, foi utilizado o índice "`_id`" do MongoDB em um dos testes. Esse índice foi utilizado porque

era necessário mostrar que essa funcionalidade existe e que faz muita diferença no tempo de resposta das buscas. Todas as propostas de modelagem executaram as consultas quase que instantaneamente quando o índice foi utilizado. Ainda assim, a P3-Documento diário demorou mais do que a P1-Lista e a P2-Chave-valor. Sem a utilização do índice, as propostas de modelagem apresentaram o mesmo padrão de resposta, porém com tempos médios de execução muito maiores.

Nas inserções de uma nova informação diária em cada uma das propostas de modelagem realizadas no experimento IV, a P3-Documento diário apresentou, pela primeira vez, o melhor desempenho comparada com P1-Lista e a P2-Chave-valor, realizando as inserções instantaneamente. Isso ocorreu porque, nesse caso, é realizada a inserção de um novo documento sem realizar buscas no banco de dados, enquanto que na P1-Lista e na P2-Chave-valor, é necessária a busca do documento do ativo financeiro e posteriormente a inserção da nova informação diária.

Nas alterações de uma informação diária realizadas no experimento V, a P3-Documento diário apresentou o pior desempenho novamente devido a necessidade de percorrer o grande volume de documentos para realizar a alteração. A P2-Chave-valor apresentou o melhor tempo médio entre as propostas de modelagem porque precisou de apenas uma *query* para realizar a alteração, enquanto para a P1-Lista foram necessárias duas *queries*.

No último experimento onde foram realizadas exclusões, foi possível observar uma grande diferença entre a P1-Lista e a P2-Chave-valor. Esse resultado indica que o MongoDB apresenta melhor desempenho ao excluir um item de uma lista do que ao excluir um campo com chave-valor. A P3-Documento diário novamente apresentou o pior tempo médio entre as três propostas de modelagem.

5 CONCLUSÃO

Este trabalho fez análise de desempenho entre três diferentes propostas de modelagem utilizando o banco de dados não-relacional MongoDB. Para isso, foi utilizada uma base de dados com informações diárias de ativos financeiros. As propostas foram analisadas quanto ao armazenamento em disco, o tempo médio de execução de consultas, inserções, alterações e exclusões de informações diárias. Para tanto, foi necessário desenvolver uma aplicação que foi responsável pelo *download* dos arquivos com as informações, implementar no MongoDB as três modelagens propostas e popular o banco de dados MongoDB.

A P1-Lista apresentou melhor desempenho quanto ao armazenamento em disco, tempo médio de execução de consultas de uma informação diária sem utilizar o índice "`_id`" e tempo de médio de execução de exclusão de uma informação diária. A P2-Chave-valor apresentou melhor desempenho ao realizar consultas de uma informação diária utilizando o índice "`_id`" e ao realizar alterações em uma informação diária. A P3-Documento diário, por sua vez, apresentou melhor desempenho quanto ao tempo médio de resposta ao realizar a inserção de uma nova informação diária. Esses resultados indicam que, tendo em vista o desempenho, a forma como os dados são organizados é tão importante quanto a escolha de um banco de dados.

Bancos de dados com informações diárias são utilizados em muitos âmbitos da sociedade e nem sempre são fáceis de manipular devido ao grande volume de informações que possuem. Os resultados deste trabalho podem incentivar as empresas ou instituições que fazem uso deste tipo de dado a levar em consideração o planejamento da organização dos dados, que muitas vezes é uma etapa negligenciada nos projetos e que pode ocasionar problemas futuros. Esse planejamento pode ser feito com o intuito de priorizar os recursos disponíveis em cada caso.

Sabendo-se que dados com informações diárias são muito utilizados em análises na forma de séries históricas, uma possível continuidade do trabalho seria analisar o desempenho das propostas de modelagem em consultas por intervalos de tempo. Dependendo do objetivo da aplicação, essa comparação temporal pode ser útil. Além disso, este trabalho também pode ser desenvolvido utilizando outras propostas de modelagem e avaliando outras métricas que não foram contempladas neste trabalho.

REFERÊNCIAS

ABRAMOVA, V.; BERNARDINO, J. Nosql databases: Mongodb vs cassandra. In: **Proceedings of the international C* conference on computer science and software engineering**. [S.l.: s.n.], 2013. p. 14–22.

ÁDÁM, N. et al. Methods of the data mining and machine learning in computer security. **Acta Electrotechnica et Informatica**, v. 2, p. 46–50, 2014.

ANUSHA, K. et al. Comparative study of mongodb vs cassandra in big data analytics. In: IEEE. **2021 5th International Conference on Computing Methodologies and Communication (ICCMC)**. [S.l.], 2021. p. 1831–1835.

CHAUHAN, A. A review on various aspects of mongodb databases. **International Journal of Engineering Research & Technology (IJERT)**, v. 8, n. 5, 2019.

CHEN, C. P.; ZHANG, C.-Y. Data-intensive applications, challenges, techniques and technologies: A survey on big data. **Information sciences**, Elsevier, v. 275, p. 314–347, 2014.

CHOPADE, R.; PACHGHARE, V. Mongodb indexing for performance improvement. In: **ICT Systems and Sustainability**. [S.l.]: Springer, 2020. p. 529–539.

CHOVANEC, M.; CHOVANCOVÁ, E.; DUFALA, M. Dids based on hybrid detection. In: IEEE. **2014 IEEE 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)**. [S.l.], 2014. p. 79–83.

ELMASRI, R. et al. Sistemas de banco de dados. Pearson Addison Wesley São Paulo, 2005.

FRACZEK, K.; PLECHAWSKA-WOJCIK, M. Comparative analysis of relational and non-relational databases in the context of performance in web applications. In: SPRINGER. **International Conference: Beyond Databases, Architectures and Structures**. [S.l.], 2017. p. 153–164.

GUNAWAN, R.; RAHMATULLOH, A.; DARMAWAN, I. Performance evaluation of query response time in the document stored nosql database. In: IEEE. **2019 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering**. [S.l.], 2019. p. 1–6.

HOWS, D.; MEMBREY, P.; PLUGGE, E. **Introdução ao MongoDB**. [S.l.]: Novatec Editora, 2019.

JATANA, N. et al. A survey and comparison of relational and non-relational database. **International Journal of Engineering Research & Technology**, v. 1, n. 6, p. 1–5, 2012.

KLEIN, J. et al. Performance evaluation of nosql databases: a case study. In: **Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems**. [S.l.: s.n.], 2015. p. 5–10.

KOLONKO, K. **Performance comparison of the most popular relational and non-relational database management systems**. 2018.

MAKRIS, A. et al. Performance evaluation of mongodb and postgresql for spatio-temporal data. In: **EDBT/ICDT Workshops**. [S.l.: s.n.], 2019.

MARTINS, P.; ABBASI, M.; SÁ, F. A study over nosql performance. In: SPRINGER. **World Conference on Information Systems and Technologies**. [S.l.], 2019. p. 603–611.

MONGODB. **MongoDB manual**. 2022. Url http:/https://www.mongodb.com/docs/manual/indexes/.

SAVITZ, E. Gartner: 10 critical tech trends for the next five years. In: **Reported on David Cappuccio's keynote at the Gartner Symposium ITxpo**. [S.l.: s.n.], 2012.

SILBERSCHATZ, A. et al. **Database system concepts**. [S.l.]: McGraw-Hill New York, 2002. v. 5.

TIWARI, S. **Professional nosql**. [S.l.]: John Wiley & Sons, 2011.

VOKOROKOS, L.; UCHNÁR, M.; LEŠČIŠIN, L. Performance optimization of applications based on non-relational databases. In: IEEE. **2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)**. [S.l.], 2016. p. 371–376.