UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOSÉ ROBERTO FARIAS LOPES JÚNIOR

POSTGRESQL VERSUS MONGODB: UM ESTUDO COMPARATIVO BASEADO EM BENCHMARK DE CONSULTAS

QUIXADÁ
2016

JOSÉ ROBERTO FARIAS LOPES JÚNIOR

POSTGRESQL VERSUS MONGODB: UM ESTUDO COMPARATIVO BASEADO EM BENCHMARK DE CONSULTAS

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do Título de Bacharel em Sistemas de Informação.

Orientadora Profª. Ma. Ticiana Linhares Coelho da Silva

QUIXADÁ
2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá
Gerada automaticamente pelo módulo Catalog, mediante aos dados fornecidos pelo(a) autor(a)

L853p Lopes Júnior, José Roberto Farias.
PostgreSQL versus MongoDB: Um estudo comparativo baseado em benchmark de consultas / José Roberto Farias Lopes Júnior. – 2016.
63 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2016.
Orientação: Profa. Ma. Ticiana Linhares Coelho da Silva

1. Banco de dados. 2.Banco de dados relacional. 3. Banco de dados não relacionais. 4. NoSQL banco de dados. I. Título.

CDD 005

JOSÉ ROBERTO FARIAS LOPES JÚNIOR

POSTGRESQL VERSUS MONGODB: UM ESTUDO COMPARATIVO BASEADO EM BENCHMARK DE CONSULTAS

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.
Área de concentração: computação

Aprovado em: _____ / julho / 2016.

BANCA EXAMINADORA

_____
Prof[a]. Ma. Ticiana Linhares Coelho da Silva (Orientadora)
Universidade Federal do Ceará-UFC

_____
Prof. MSc. Régis Pires Magalhães.
Universidade Federal do Ceará-UFC

_____
Prof[a]. Ma. Lívia Almada Cruz
Universidade Federal do Ceará-UFC

A minha avó,

Aos meus pais,

A minha família e amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus.

Agradeço a minha avó, Maria Farias Lima, que sempre está comigo.

A minha família. Meus pais José Roberto Farias Lopes e Maria Elizângela Oliveira Bezerra, a minha irmã Maria das Graças.

Aos meus amigos que ao longo desta jornada sempre me apoiaram e me deram forças para continuar.

A minha orientadora, Ticiana, por acompanhar e contribuir com o desenvolvimento do meu trabalho, mesmo com as dificuldades que tivémos ao longo deste.

A banca avaliadora, por fazerem considerações importantes trazendo melhorias para o trabalho.

Em suma, agradeço a todos que contribuíram de alguma forma

"Todas as grandes coisas são simples. E muitas podem ser expressas em algumas palavras: liberdade; justiça; honra; dever; piedade; esperança."
(Winston Churchill)

RESUMO

A geração de dados devido ao advento da Internet tem aumentado ao longo dos últimos anos. Porém, a manipulação e análise desses dados se torna cada vez mais complexa, necessitando de novas tecnologias que realizem estas tarefas. Dentro desse âmbito, surgiram as soluções NoSQL que auxiliam na manipulação dessa extensa quantidade de dados, por serem tecnologias criadas para gerenciar grande volume de dados que muitas vezes são semi-estruturados ou até mesmo sem estrutura e que trabalham com modelos diferentes do paradigma relacional. Com isso, surge a necessidade de um estudo comparativo para determinar em quais situações uma tecnologia pode apresentar melhor desempenho que a outra, ou seja, em quais situações ou operações em banco de dados, o NoSQL é melhor que o relacional, ou o contrário. Dessa forma, buscou-se realizar este estudo comparativo utilizando um sistema gerenciador de banco de dados que utiliza o paradigma relacional e outro que utiliza o paradigma não-relacional. Nesse caso com o modelo orientado a documentos. O presente trabalho apresenta todo o processo de determinação de métricas para mensurar resultados e a criação de um *microbenchmark* de consultas utilizado na experimentação. Foi realizada uma análise comparativa de desempenho com esses dois tipos de sistemas de bancos de dados (PostgreSQL e MongoDB).

Palavras-chave: Banco de dados. Bancos de dados relacional. Bancos de dados não relacionais. NoSQL banco de dados.

ABSTRACT

The data generation due to the advent of the Internet has increased over the past few years. However, the manipulation and analysis of this data become increasingly complex, requiring new technologies to carry out these issues. Within this context, the NoSQL solution help to deal with this extensive amount of data, since they are technologies designed to manage large volumes of data that are often semi-structured or even unstructured, and to work on different models than relational. In this work, our goal is to compare which technology can perform better than the other, i.e., in which situations or operations in the database, NoSQL is better than relational or otherwise. Thus, we sought to carry out this comparative study using a database management system that uses the relational paradigm and another that uses the non-relational paradigm, in this case with the oriented model documents. This paper presents the the metrics to measure the results and the proposal of a microbenchmark used in experimentation phase. A comparative performance analysis was performed with two types of database systems (PostgreSQL and MongoDB).

Keywords: Databases. Relational databases. Non relational databases. NoSQL databases.

LISTA DE ILUSTRAÇÕES

Figura 1 - Exemplo de um documento no MongoDB 19
Figura 2 - Exemplo da tabela Trajetória 20
Figura 3- Full scan sem índice na coluna id da tabela trajetorias. 29
Figura 4 - Full scan com índice na coluna id da tabela trajetorias. 29
Figura 5 - Exact match sem índice na coluna id da tabela trajetorias. 31
Figura 6 - Exact match com índice na coluna id da tabela trajetorias. 31
Figura 7 - Insert sem índice na coluna id da tabela trajetorias. 33
Figura 8 - Insert com índice na coluna id da tabela trajetorias. 34
Figura 9 - Update sem índice na coluna id da tabela trajetorias. 35
Figura 10 - Update com índice na coluna id da tabela trajetorias. 36
Figura 11 - Delete se índice na coluna id da tabela trajetorias. 38
Figura 12 - Delete com índice na coluna id da tabela trajetorias. 38
Figura 13 - Aggregation sem índice na coluna id da tabela trajetorias. 40
Figura 14 - Aggregation com índice na coluna id da tabela trajetorias. 40

SUMÁRIO

1 INTRODUÇÃO 13

2 FUNDAMENTAÇÃO TEÓRICA 16

2.1 NoSQL 16
2.1.1 Modelo Orientado a Documentos 18
2.2 Modelo Relacional 19

3 TRABALHOS RELACIONADOS 21

3.1 SQL x NoSQL: Análise de desempenho do uso do MongoDB em relação ao uso do PostgreSQL 21
3.2 A performance comparison of SQL and NoSQL 21
3.3 Comparing NoSQL MongoDB to an SQL DB 22

4 PROCEDIMENTOS METODOLÓGICOS 24

4.1 Set up dos experimentos 24
4.2 Escolha dos SGBDs 24
4.3 Métricas de comparação 25
4.4 Microbenchmark 25
4.5 Coleta dos dados 27

5 RESULTADOS 28

5.1 Full scan 29
5.2 Exact match 31
5.3 Insert 33
5.4 Update 35
5.5 Delete 38
5.6 Aggregation 40

6 DISCUSSÃO 42

7 CONSIDERAÇÕES FINAIS 43

REFERÊNCIAS 44

APÊNDICES 46

APÊNDICE A – Plano de Consulta MongoDB Full Scan sem Índice 46
APÊNDICE B – Plano de Consulta PostgreSQL Full Scan sem Índice 47
APÊNDICE C – Plano de Consulta MongoDB Full Scan com Índice 47
APÊNDICE D – Plano de Consulta PostgreSQL Full Scan com Índice 48
APÊNDICE E – Plano de Consulta MongoDB Exact Match sem Índice 48
APÊNDICE F – Plano de Consulta PostgreSQL Exact Match sem Índice 49
APÊNDICE G – Plano de Consulta MongoDB Exact Match com Índice 50
APÊNDICE H – Plano de Consulta PostgreSQL Exact Match com Índice 52
APÊNDICE I – Plano de Consulta PostgreSQL Insert sem Índice 52
APÊNDICE J – Plano de Consulta PostgreSQL Insert com Índice 52
APÊNDICE K – Plano de Consulta MongoDB Update sem Índice 53

APÊNDICE L – Plano de Consulta PostgreSQL Update sem Índice 54
APÊNDICE M – Plano de Consulta MongoDB Update com Índice 54
APÊNDICE N – Plano de Consulta PostgreSQL Update com Índice 57
APÊNDICE O – Plano de Consulta MongoDB Delete sem Índice 57
APÊNDICE P – Plano de Consulta PosgreSQL Delete sem Índice 58
APÊNDICE Q – Plano de Consulta MongoDB Delete com Índice 59
APÊNDICE R – Plano de Consulta PostgreSQL Delete com Índice 61
APÊNDICE S – Plano de Consulta MongoDB Aggregate sem Índice 61
APÊNDICE T – Plano de Consulta PostgreSQL Aggregate sem Índice 62
APÊNDICE U – Plano de Consulta MongoDB Aggregate com Índice 62
APÊNDICE V – Plano de Consulta PostgreSQL Aggregate com Índice 63

1 INTRODUÇÃO

Grandes quantidades de dados vêem sendo geradas constantemente com a expansão do uso das redes sociais, aliado ao crescimento do número de dispositivos de coleta de dados. Nas grandes aplicações web é comum ver uma gigantesca quantidade de informações sendo criada e a tendência é que haverá mais dados a se armazenar. Toda essa grande quantidade de dados, geralmente semi-estruturados ou até mesmo não-estruturados, necessita de modelos ou tecnologias flexíveis de manipulação. Dentro desse contexto, surgiram as bases de dados NoSQL (STRAUCH et al. 2011), Not Only SQL, que quer dizer "Não apenas SQL" (DE DIANA e GEROSA, 2010). O modelo de dados NoSQL é uma abordagem semi ou não estruturada de gerenciamento de dados que preza desempenho, escalabilidade e disponibilidade (MARTINS FILHO, 2015).

Apesar do movimento não-relacional estar bastante forte e com um futuro promissor, as bases de dados relacionais não perderam sua importância. Apresentando vantagens que são de valia para as organizações, tais como: redução da redundância de dados, evitar a inconsistência, compartilhar dados, reforçar padrões, aplicar restrições de segurança, manter a integridade, equilibrar as necessidades conflitantes (MARTINS FILHO, 2015). Os sistemas de gerenciamento de bancos de dados relacionais são a tecnologia para o armazenamento de dados estruturados em aplicações web e de negócios predominante (STRAUCH et al. 2011).

O modelo relacional, que se baseia na ideia de que todos os dados são armazenados em tabelas, é um modelo consagrado no mercado adotado por muitas organizações. Muitas delas não vêem a necessidade de migrar seus dados para o modelo não-relacional, certificando ainda mais o respaldo que esses sistemas de banco de dados ainda detêm na atualidade. Ao contrário do que muito se especula, a tecnologia NoSQL não veio com o intuito de abolir as tecnologias relacionais, mas como uma alternativa para suprir suas limitações (KOKAY, 2012). Uma das principais limitações é a manipulação de dados sem uma estrutura padrão, pois nos bancos de dados relacionais os dados são armazenados em tabelas e essas tabelas possuem restrições de integridade que invibializam armazenar dados que violam essas restrições.

Visto as diferenças que as duas abordagens apresentam, surgiu a motivação para realizar este estudo, que comparou o desempenho do modelo de dados relacional e um modelo não-relacional orientado a documentos. A análise comparativa foi realizada, utilizando um

conjunto de dados de trajetórias da Microsoft, o T-Drive[1] (YUAN, ZHENG, XIE et at. 2011), um *microbenchmarck,* e as seguintes métricas de comparação: quantidade de operações de entrada e saída, tempo de consulta e o uso da CPU. A comparação foi entre um SGBD[2] relacional e um não-relacional. Os SGBDs escolhidos para a realização deste trabalho foram respectivamente: PostgreSQL[3] para o relacional, por ser um banco de dados tradicional e o MongoDB[4] para o não-relacional, por apresentar uma abordagem de modelo de dados bastante conhecida e que têm sido utilizada em diversas aplicações, o modelo orientado a documentos.

Existem muitos trabalhos similares ao que foi feito com relação ao estudo do desempenho de SGBDs relacionais e não-relacionais, porém divergem com relação ao que foi analisado. As métricas de comparação e os modelos de dados utilizados são diferentes. Por meio do resultado deste trabalho, espera-se poder sugerir a um administrador de banco de dados ou desenvolvedor de *software*, por exemplo, qual sistema gerenciador de banco de dados apresenta melhor desempenho de acordo com as atividades a serem desempenhadas: consultas, inserções, atualizações e remoções de dados.

Como objetivo principal deste trabalho tem-se a realização de uma análise comparativa entre um SGBD não-relacional e um SGBD relacional, para compreender em que situações um SGBD apresenta melhor desempenho que outro. Tendo como objetivos específicos:

- Contrução de um *microbenchmark* de consultas.
- Comparar o desempenho dos SGBDs com relação ao tempo de consulta.
- Comparar o desempenho dos SGBDs com relação a quantidade de operações de I/O.
- Comparar o desempenho dos SGBDs com relação ao uso de CPU.

O trabalho segue a seguinte divisão: inicialmente, no Capítulo 2 são apresentados os conceitos chave necessários para melhorar a compreensão deste trabalho. O paradigma NoSQL, as definições do modelo orientado a documentos e do modelo relacional. Os trabalhos relacionados que foram utilizados como base para a elaboração da proposta aqui apresentada, são abordados no Capítulo 3. No Capítulo 4, são apresentados os Procedimentos Metodológicos desse trabalho. Os resultados desse trabalho são mostrados no Capítulo 5, e no Capítulo 6 é

[1] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/
[2] Sistema de Gerenciamento de Banco de Dados
[3] http://www.postgresql.org
[4] http://www.mongodb.com

apresentada a discussão sobre as informações obtidas e os resultados. Por fim, as considerações finais e trabalhos futuros compõem o Capítulo 7.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é feita uma apresentação sobre os conceitos do paradigma NoSQL e o modelos de dados (orientado a documento) utilizados neste trabalho. No entando, o foco desta seção é maior no modelo não-relacional, em razão de que a tecnologia relacional é amplamente conhecida na academia e no mercado.

2.1 NoSQL

A abordagem NoSQL atualmente recebe grande motivação principalmente dos dispositivos de coleta de dados e das aplicações decorrentes da web 2.0. Algumas destas aplicações exigem uma grande capacidade de armazenamento, que transpõe a capacidade de bancos de dados relacionais (HECHT e JABLONSKI, 2011). As tecnologias NoSQL vêm com uma proposta cujo intuito é corresponder aos requisitos de gerência do grande volume de dados, semi-estruturados e até mesmo sem estrutura nenhuma (MARTINS FILHO, 2015).

O NoSQL envolve uma grande quantidade de diferentes tecnologias de bancos de dados: modelo chave-valor, orientado a grafos, orientado a colunas e orientado a documentos. A tecnologia não-relacional escolhida para este trabalho foi a orientada a documentos, por ser *open-source*, por ser uma das mais famosas e utilizadas, e ainda por armazenar os dados de maneira de fácil compreensão e utilização. As tecnologias NoSQL foram desenvolvidas para se adaptarem ao crescente volume de dados, gerados constantemente sobre usuários e produtos, que são armazenados na web, a frequência que esses dados são acessados e as necessidades de processamento e disponibilidade. Os sistemas NoSQL em sua grande maioria possuem as seguintes características: esquema dinâmico ou sem esquema, escalabilidade horizontal, suporte nativo à replicação, API[5] simples para acesso aos dados e consistência eventual. Algumas dessas características são o que mais os diferenciam dos sistemas de bancos de dados relacionais tradicionais, tornando-os adequados para armazenar uma grande quantidade de dados estruturados, semi-estruturados ou sem estrutura (LÓSCIO et al. 2011). A seguir, são definidas algumas dessas características:

- **Esquema dinâmico ou sem esquema:** a total ausência ou ausência parcial do esquema que define a estrutura dos dados no modelo é uma forte característica dos bancos de dados NoSQL. Não ficar preso a um esquema definido auxilia na escalabilidade. Por outro lado, a ausência de um

[5] Application Programming Interface, que significa “Interface de Programação de Aplicativos”

esquema, prejudica a integridade dos dados, algo que é extremamente prezado nos bancos de dados relacionais, pois possuem uma estrutura rigorosa.

- **Escalabilidade horizontal:** a necessidade de escalabilidade e aumento no desempenho crescem linearmente com o aumento no volume de dados. Em resposta a essa necessidade, têm-se a escalabilidade horizontal, que consiste em aumentar o número de máquinas para armazenamento e processamento de dados. A escalabilidade horizontal se torna uma solução mais viável para solucionar o problema do rápido aumento no volume de dados, porém essa estratégia requer que vários processos de uma determinada tarefa sejam criados e distribuídos. Em um sistema relacional essa estratégia se torna inviável, tendo em vista que vários processos se conectando ao mesmo tempo em um mesmo aglomerado de dados acarretaria uma alta concorrência. Como consequência disso, ocorre o aumento no tempo de acesso aos dados envolvidos. Bancos de dados NoSQL têm como uma de suas principais peculiaridades a ausência de bloqueio, ou seja, ele não é afetado pela alta concorrência, tornando esta tecnologia apropriada para resolver problemas de gerenciamento de grandes volumes de dados.
- **Suporte nativo à replicação:** outra forma de melhorar a escalabilidade é a replicação. Os bancos de dados NoSQL em sua grande maioria, fornecem essa capacidade de replicação nativamente, diminuindo o tempo necessário para recuperar informações. As duas principais estratégias de replicação são, a *master-slave* ou mestre-escravo e *multi-master*. Na estratégia mestre-escravo, a escrita é feita no nó mestre e refeita em cada nó escravo pelo nó mestre. Essa estratégia torna a leitura mais rápida, porém a escrita torna-se um gargalo nesta abordagem. Na estratégia *multi-master*, temos vários nós mestres para diminuir o gargalo gerado pela escrita, que acontece na abordagem anterior, porém essa estratégia pode causar conflito de dados.
- **API simples para acesso aos dados:** um sistema NoSQL tem como objetivo fornecer eficiência no acesso aos dados, propiciando uma alta disponibilidade e escalabilidade. O foco não é no armazenamento dos dados e sim na velocidade em como eles são recuperados. Em vista disso, é conveniente que as APIs sejam desenvolvidas para facilitar o acesso aos dados.

- **Consistência eventual:** essa característica de algumas tecnologias não-relacionais está relacionada ao fato de que a consistência nem sempre é realizável entre vários pontos de distribuição de dados. É uma característica que tem como base o teorema CAP, que basicamente diz que, em um determinado momento só é possível assegurar duas dentre essas três propriedades: consistência, disponibilidade e tolerância à participação (LÓSCIO et al. 2011).

Existem técnicas que são importantes para a implementação das funcionalidades de um SGBD NoSQL, tais como: *map/reduce* que dá suporte ao gerenciamento de altos volumes de dados distribuídos ao longo dos nós de uma rede, o *consistent hashing* que suporta mecanismos de armazenamento e recuperação em banco de dados distribuídos, o *multiversion concurrency control* um mecanismo que dá suporte a transações paralelas em um banco de dados e os *vector clocks* que são usados para gerar uma ordenação dos eventos acontecidos em um sistema (LÓSCIO et al. 2011). Os sistemas NoSQL aparecem com uma grande diversificação de formas e funcionalidades, a única característica comum entre todos eles, é que todas as abordagens são não-relacionais. Mesmo existindo diversos SGBDs NoSQL com diferentes formas, a indústria de *software* classificou esses bancos de dados em pequenos conjuntos (MARTINS FILHO, 2015). O tópico logo a seguir trata de explicar a classificação de um banco de dados não-relacional que é abordado neste trabalho.

2.1.1 Modelo Orientado a Documentos

Nas bases de dados que se baseiam no modelo de dados orientado a documentos, como o própio nome já deixa claro, os dados são armazenados em coleções de documentos, geralmente no formato JSON[6] ou em documentos com formatos semelhantes ao JSON. Todo documento contém uma chave única de identificação especial "_id", que por sua vez é único também dentro da coleção de documentos, que é um agrupamento de vários documentos, ou seja, identifica o documento globalmente dentro da coleção (HECHT e JABLONSKI, 2011). Outra peculiaridade desse modelo é que ele não se prende a um esquema rígido, logo não requer uma estrutura fixa como acontece nos bancos que se baseiam no modelo relacional. Dessa forma, é possível atualizar a estrutura do documento, adicionando-se novos campos, por exemplo, que não irá causar nenhuma inconsistência ao banco de dados (LÓSCIO et al. 2011).

[6] http://www.json.org

Os SGBDs mais utilizados que implementam o modelo orientado a documentos são: CouchDB[7] e o MongoDB.

Neste trabalho o sistema NoSQL escolhido para esse modelo de dados, como já mencionado anteriormente, foi o MongoDB. Na Figura 1, é apresentado um exemplo de um documento no MongoDB, similar aos dados que o autor deste trabalho utilizou nos experimentos representando uma trajetória de um táxi, onde os campos foram especificados como: _id, id, date_time, longitude e latitude.

Figura 1 - Exemplo de um documento no MongoDB

```
{
    trajetoria : {
        "_id" : ObjectId("571d96a7204602b1e798b30c"),
        "id" : 8343,
        "date_time" : "2016-06-18",
        "longitude" : 5.1919864,
        "latitude" : 39.293399
    }
}
```

Fonte: elaborada pelo autor.

O esquema flexível, como uma das principais vantagens do modelo orientado a documentos, permite modificar a estrutura do documento acima, da forma como se achar necessário na aplicação.

2.2 Modelo Relacional

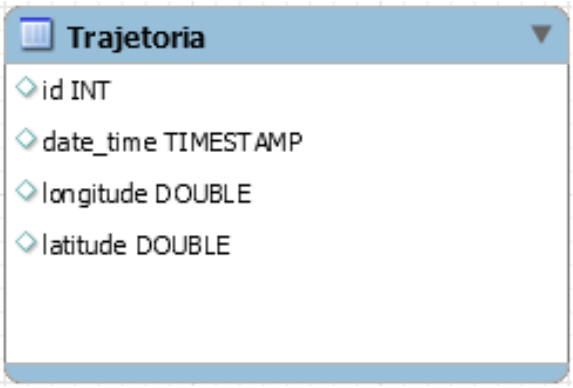
Os bancos de dados relacionais surgiram como sucessores dos modelos hierárquicos e de rede. O modelo relacional modela os dados de forma que eles fiquem organizados no formato de tabelas, de maneira mais convencional, em relações (MARTINS FILHO, 2015). Tal modelo tornou-se padrão para a maior parte dos sistemas gerenciadores de bancos de dados. Este modelo possui uma característica muito importante, que são as restrições de integridade que viabilizam a consistência de um SGBD relacional. O modelo relacional

[7] http://couchdb.apache.org

adotou uma linguagem padrão para o manuseio dos dados contidos no banco de dados, o SQL (KOKAY, 2012).

Uma outra característica que vale ressalva no modelo relacional é o processo proposto por Edgar Frank Codd, o criador do modelo relacional, a normalização. Com a normalização aplica-se uma série de passos e através disso é possível remodelar as relações a fim de que o mesmo fique mais coerente e se torne capaz de suportar alterações nos dados. Qualquer operação que pode ocasionar problemas como redundância e perda de informação caso o banco de dados não esteja normalizado (MACHADO e DE ABREU, 1996). O processo de normalização, decompõe uma relação em novas relações, o que aumenta o custo de consultas do tipo junção, pois em uma consulta desse tipo o banco de dados porderia ter que realizar múltiplas junções em várias tabelas até recuperar o(s) registro(s). Já nos bancos NoSQL, em geral, esse processo não é realizado, justamente pelo fato dessas tecnologias serem caracterizadas por serem desnormalizadas. A Figura 2 representa o modelo relacional do conjunto de dados utilizado nos experimentos deste trabalho.

Figura 2 - Exemplo da tabela Trajetória



Fonte: elaborada pelo autor.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os principais trabalhos relacionados a este de acordo com os conceitos chaves. Os trabalhos, a seguir, assim como este são estudos comparativos e fazem uma análise comparando determinados aspectos entre as tecnologias estudadas. Ao final deste capítulo, o Quadro 1 desmonstra um comparativo dos principais aspectos de cada trabalho relacionado comparado com os aspectos deste trabalho.

3.1 SQL x NoSQL: Análise de desempenho do uso do MongoDB em relação ao uso do PostgreSQL

O trabalho de Martins Filho (2015) analisa o desempenho entre duas abordagens de SGBDs, a abordagem relacional e não-relacional, tendo como escolha o PostgreSQL para a primeira abordagem e o MongoDB para a segunda abordagem. Dois importantes SGBDs no mercado e amplamente utilizados, por trabalharem com modelos de dados bastantes distintos, um relacional e o outro orientado a documentos respectivamente. O plano de experimento definido por Martins Filho (2015) é baseado em requisições feitas por vários usuários executando uma consulta de projeção de valores, e dessa forma permitiu-o analisar o desempenho dos bancos de acordo com o aumento do número de usuários realizando essa determinada consulta. A análise dos resultados obtidos por Martins Filho (2015) foi utilizando a ferramenta JMeter[8], que fornece um controle de *threads*, assim o autor pôde simular os usuários realizando as várias requisições.

De forma semelhante a Martins Filho (2015), neste trabalho também foi utilizado o PostgreSQL e o MongoDB na comparação de desempenho. O presente trabalho propôs uma abordagem diferente do trabalho de Martins Filho (2015) para analisar o desempenho dos SGBDs, baseando-se em métricas definidas para a análise: o tempo de execução das consultas, que também foi uma métrica analisada por Martins Filho (2015), não se restringindo somente a esta métrica, também analisou o uso de CPU e a quantidade de operações de IO.

3.2 A performance comparison of SQL and NoSQL

Li e Manoharan (2013) tiveram como foco comparar vários SGBDs NoSQL e um SGBD relacional utilizando armazenamento chave-valor. Li e Manoharan (2013) realizaram operações de leitura, escrita, deleção e instanciação, usando armazenamento chave-valor. Esse

[8] http://jmeter.apache.org

trabalho analisou o desempenho dos bancos de dados de acordo com cada operação. Isso também é um objetivo deste trabalho. Porém cada operação no trabalho de Li e Manoharan (2013) foi realizada somente uma vez, o que não gera resultados tão precisos, no presente trabalho todas as operações foram realizadas 5 vezes, a partir do resultado de cada operação foi extraída uma média, obtendo assim um resultado final que será mostrado sua análise na Seção 5. O uso da média é que, os experimentos foram realizados em um ambiente de computação em nuvem, em que a máquina pode apresentar diferentes desempenhos ao longo do tempo (COELHO DA SILVA, 2012). Porém, mesmo não estando em um abiemte de computação em nuvem, é importante se observar a média de um conjunto de testes.

Li e Manoharan (2013) analisam os bancos somente com relação ao tempo de resposta. No presente trabalho além do tempo de resposta, os SGBDs foram analisados com relação as métricas definidas pelo autor. As métricas que são: quantidade de operações de IO, uso da CPU e também o tempo de reposta de uma consulta. Elas exploraram outros fatores de desempenho a fim de obter uma análise mais concisa e mostrar aspectos que poderam ser avaliados no momento de escolher o modelo de dados a se utilizar.

3.3 Comparing NoSQL MongoDB to an SQL DB

Zachary et al. (2013) realizou seu estudo de desempenho de um banco NoSQL sobre uma base de dados estruturada, característica que é favorável para um SGBD SQL. Já no presente trabalho foi utilizada uma base de dados semi-estruturados, característica que é mais favorável para o uso de um SGBD NoSQL. Assim pode-se explorar o potencial dessa tecnologia. Zachary et al. (2013) utilizou o SQL Server para o modelo relacional e comparou o desempenho com o MongoDB. Neste trabalho optou-se por usar o PostgreSQL como SGBD relacional, por ser um poderoso sistema de banco de dados objeto-relacional e open-source.

Para determinar o desempenho dos bancos de dados Zachary et al. (2013) fez a comparação de três aspectos principais de performance: velocidade de inserção, atualização e tempo de reposta de operações de projeção. Neste presente trabalho, além desses aspectos, também foram levados em conta a operação de remoção, além das métricas de desempenho já mencionadas.

Quadro 1 – Comparativo das principais características dos trabalhos relacionados

	Uso de base de dados não estruturado	Observação da variação da máquina	Uso de benchmark de consultas	Métricas de comparação	Modelo de dados NoSQL usado
[Martins Filho, 2015]	Sim	Não	Realiza operações: um full scan.	Cobre as métricas: tempo de execução de consulta.	Orientado a documentos e modelo relacional
[Li e Manoharam, 2013]	Sim	Não	Realiza operações: instanciação, full scan, exact match e delete.	Cobre as métricas: tempo de execução de consulta	Orientado a documentos, orientado a colunas e modelo relacional
[Zachary et al. 2013]	Não	Não	Realiza operações: insert, update e exact match;	Cobre as métricas: tempo de execução de consulta	Orientado a documentos e modelo relacional
Este trabalho	Sim	Sim	Realiza operações: full scan, insert, select, update, delete e agreggation.	Cobre as métricas: tempo de execução de consulta, uso de CPU e operações de I/O	Orientado a documentos e modelo relacional

Fonte: elaborado pelo autor.

4 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo serão apresentadas as etapas que envolveram o desenvolvimento da solução deste trabalho.

4.1 Set up dos experimentos

Para inicio dos experimentos, foi preparado um ambiente de trabalho, onde foram definidos o conjunto de dados que seria utilizado na pesquisa e a máquina onde foram executados todos os experimentos. A base de dados escolhida para este trabalho foi a base de trajetórias T-Drive da Microsoft, que contém uma semana de trajetórias de 10.357 táxis. O número total de pontos neste conjunto de dados é de cerca de 15 milhões e a distância total das trajetórias atinge 9 milhões de quilômetros. Essa base foi escolhida por ser aberta e simples de trabalhar.

A máquina em que foram realizados os experimentos, foi concedida pela Universidade Federal do Ceará. Uma máquina virtual da infraestrutura de nuvem mantida pela própria instituição, com as seguintes configurações: sistema operacional ubuntu 14.04 server de 64 bits, memória de 4Gb de ram e disco de 40Gbs.

4.2 Escolha dos SGBDs

Para este trabalho foram escolhidos dois SGBDs com os respectivos modelos de dados estudados: orientado a documentos e modelo relacional. O mesmo conjunto de dados foi persistido nos dois bancos de dados, onde foram aplicadas as operações definidas no *microbenchmark*. Este será demonstrado logo a seguir, e teve como finalidade estudar o desempenho de cada SGBD em determinadas situações. A escolha de cada SGBD tanto não-relacional quanto relacional foi baseada na relevância que os mesmos tinham para o mercado e para academia, e em seus modelos de dados.

Para o modelo relacional foi escolhido o PostgreSQL, um poderoso sistema de gerenciamento de banco de dados, *open-source* e baseado no modelo objeto-relacional, possui mais de 15 anos de desenvolvimento, com uma comunidade ativa e uma arquitetura que ganhou uma forte reputação de confiabilidade e integridade dos dados. Para o modelo não-relacional, orientado a documentos, foi escolhido o MongoDB, um dos mais conhecidos pela comunidade nessa abordagem, de alta performance, *open-source*, sem esquema, o que o torna um SGDB interessante de se trabalhar, pois se difere bastante do modelo relacional convencional, onde ao invés de tabelas, temos documentos que podem ser sem estrutura definida.

4.3 Métricas de comparação

Para a realização do estudo comparativo foram definidas métricas. São elas: quantidade de operações de entrada e saída, uso de CPU e tempo de resposta de consulta.

Essas métricas serviram de base para comparar em quais operações um SGBD pode ter melhor desempenho que o outro. A partir dessas métricas, foram mensurados os resultados coletados com a aplicação do *microbenchmark*, o mesmo será apresentado na próxima subseção.

4.4 Microbenchmark

Foi definido um *microbenchmark* de consultas para mensurar com o auxílio das métricas de comparação o desempenho dos SGBDs. Esses testes são definidos sobre valores de chave primária, compostos de 6 tipos de operações: *full scan, exact match, insert, update, delete* e *aggregation*. O Quadro 2 demonstra as operações no MongoDB e no PostgreSQL.

- **Full scan:** utilizando somente uma tabela, projetando todos os atributos que pertencem a mesma tabela
- **Exact match:** consulta realizada sobre uma tabela da base de dados, com a finalidade de projetar um ou mais atributos da mesma tabela e sendo a condição de seleção sobre apenas um atributo.
- **Insert:** utilizando somente uma tabela para inserir vários registros na mesma tabela.
- **Update:** utilizando somente uma tabela para atualizar todos os registros que atendem a condição de atualização.
- **Delete:** utilizando somente uma tabela para remover todos os registros, onde esses registros atendem a condição de deleção.
- **Aggregation:** utilizando uma tabela e projetando da mesma tabela um atributo ou mais que pertence a função de agregação.

Quadro 2 – *Mictonechmark* de consultas

MongoDB	PostgreSQL	Descrição
db.trajetorias.find({ })	**SELECT** (*) **FROM** trajetorias	*Full Scan*
db.trajetorias.find({ **id**: 8343 }, { **longitude**: 1, **latitude**: 1, **_id**: 0 })	**SELECT** t.longitude, t.latitude **FROM** trajetorias t **WHERE** t.id = 8343	*Exact Match*
db.trajetorias.insert({ **id**: 17662985, **date_time**: '2016-06-07', **longitude**: 5.1919864, **latitude**: 39.293399 })	**INSERT INTO** trajetorias (id, date_time, longitude, latitude) **VALUES** (17662985, '2016-06-17', 5.1919864, 39.293399)	*Insert*
db.trajetorias.update({**id**:8343}, {**$set**: { **date_time**: '2016-06-18', **longitude**: 5.1919864, **latitude**: 39.293399 }}, {**multi**: true})	**UPDATE** trajetorias **SET** date_time='2016-06-07', longitude=5.1919864, latitude=39.293399 **WHERE** id=8343	*Update*
db.trajetorias.remove({ **id**: 8343 })	**DELETE FROM** trajetorias **WHERE** id=8343	*Delete*
db.trajetorias.aggregate([{ **$group**: { **_id** : "$latitude", **count** : { **$sum** : 1 } } }])	**SELECT** t.latitude, count(t.latitude) **FROM** trajetorias t **GROUP BY** t.latitude;	*Aggregate*

Fonte: elaborada pelo autor

4.5 Coleta dos dados

Os dados foram coletados após a aplicação do *microbenchmark* nos dois SGBDs, com o auxílio de algumas ferramentas. Para a coleta dos dados de CPU foi utilizado o IOStat[9] que informa a porcentagem média do uso de CPU para uma determinada operação. Para apresentar a quantidade de operações de entrada e saída que estão sendo realizadas foi utilizado o VMStat[10], que expressa as operações de I/O em blocos lidos e gravados no disco por segundo, que são respectivamente as métricas: bi e bo. O autor poderia ter utilizado apenas uma das ferramentas para todas as métricas. Porém, achou melhor utilizar as duas por comodidade, o IOStat apresenta uma média de uso de CPU em porcentagem, enquanto o VMStat apresenta os resultados de CPU em valores separados, então o autor teria que juntar esses valores e calcular o uso da CPU, algo que o IOStat já realiza automaticamente. No caso doVMStat para mostrar as operações de I/O, foi porque ele mostra os resultados em blocos de memória por segundo, já o IOStat mostra as operações em Kilobytes (KB), ou seja, novamente o autor teria que realizar a soma dos KBs para um tamanho de um bloco e depois calcular quantos blocos são lidos e gravados no disco por segundo, algo que o VMStat já faz automaticmente.

Essas ferramentas foram acionadas para cada operação do *microbenchmark*. Para conseguir resultados mais precisos as operações do *benchmark* foram executadas nos SGBDs com e sem índice, 5 vezes para cada e todos os valores extraídos, foram somados e depois realizados uma média para cada métrica. Essas médias são os valores comparativos finais de desempenho dos sistemas gerenciadores de banco de dados escolhidos para este trabalho.

---

[9] http://linuxcommand.org/man_pages/iostat1.html
[10] http://linuxcommand.org/man_pages/vmstat8.html

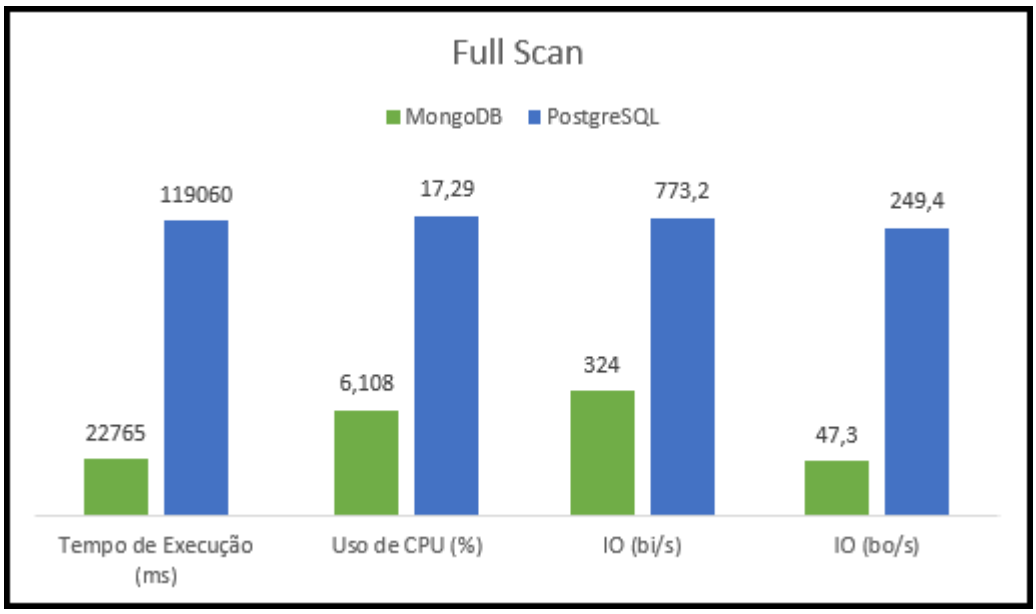
5 RESULTADOS

Após todos os experimentos terem sido realizados e os dados coletados, os resultados foram dispostos em gráficos. Como mencionado no capítulo anterior, os experimentos foram realizados sobre os bancos de dados com e sem índice nas tabelas. Nas consultas que utilizam índice, a coluna que recebe o índice é a coluna que representa o id dos pontos da trajetória. Índices são estruturas especiais que armazenam uma pequena parte dos dados da coleção no caso do MongoDB e da tabela no caso do PostgreSQL. O índice armazena o valor de um campo específico ou um conjunto de campos, que permite o sistema de banco de dados encontrar e recuperar linhas específicas mais rapidamente. O índice no MongoDB utiliza o atributo que identifica o documento globalmente dentro da coleção, o _id, mencionado no Capítulo 2 Subseção 2.1.1. No PostgreSQL os possíveis tipos de indíce, são: B-Tree, Hash, GiST, SP-GiST e GIN. Cada tipo de índice usa um algoritmo diferente que é mais adequado para diferentes tipos de consulta. O uso do índice até mesmo nas consultas que percorrem todos os blocos de memória da tabela e consequentemente não utilizam índice, foi para analisar o comportamento da base de dados antes da criação do índice e depois da criação do índice.

Nas seções a seguir, os gráficos demonstram os resultados desta pesquisa e ao final de cada seção, quadro(s) demonstrando os mínimo(s) e máximo(s) de cada execução. Para as consultas que o índice não é considerado pelo SGBD é demonstrado apenas o quadro das opereções executadas sem índice. O tempo de consulta, informa o tempo em millisegundos para execução daquela determinada consulta, o uso de CPU está em unidade de porcentagem, as operações de IO estão mensuradas pela quantidade de blocos lidos do disco por segundo (bi) e a quantidade de blocos enviados para o disco por segundo (bo). Todas as operações foram executadas 5 vezes para cada banco e foi calculado-se a média do resultados ao final da execuções. Os planos de consulta encontram-se no apêndice ao final deste documento.

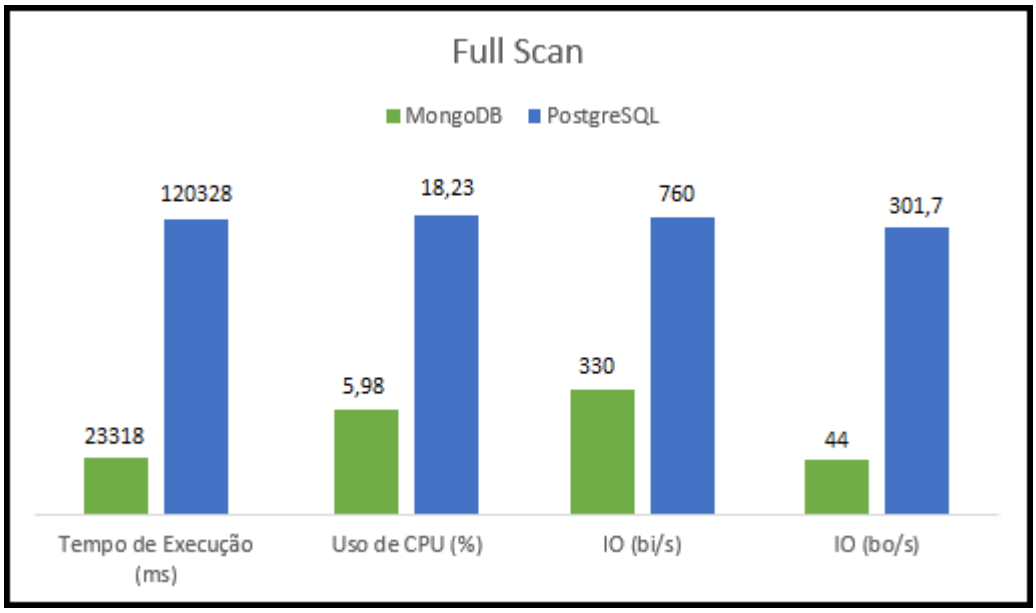
5.1 Full scan

Figura 3- Full scan sem índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor

Figura 4 - Full scan com índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

A primeira consulta executada do benchmark foi o *full scan* e após todas as rodadas de experimentos foi extraída a média dos valores de todas as métricas. É notável a diferença de performance que o MongoDB tem sobre o PostgreSQL, nesse experimento, onde o mesmo teve melhores resultados em todas as métricas, percorrendo todo o dataset em menor tempo, teve

um menor custo no uso de CPU e realizou menos operações de I/O. A explicação para essa discrepância, se dá pelo número de operações de I/O que são realizadas, o que afeta diretamente todas as outras métricas, pois quando uma consulta é executada, todos os acessos aos blocos de disco são operações de entrada e saída. Caso os dados se encontrem em memória, a consulta basicamente só consumiria CPU.

No caso desta consulta, inclui a leitura de cada bloco da tabela, pois é realizado um *full scan*. No PostgreSQL além da alta taxa de blocos lidos, temos uma alta taxa de blocos gravados no disco, isso acontece pois o PostgreSQL tenta trazer muitas páginas simultâneas do disco de uma vez, fazendo com que o buffer cache fique lotado e quando isso ocorre é necessária uma operação lógica de I/O, ou seja, é realizado um acesso ao disco o que é muito custoso para o SGBD. No gráfico, com as consultas utilizadas com índice, não há muita diferença, pois como o *full scan* percorre toda a tabela o índice não exerce nenhuma influência e isso já era o resultado esperado. A pequena variação nos valores das métricas são somente a variação de máquina, o que foi explicado no Capítulo 3 Seção 3.2.

Todos os custos desta consulta podem ser lidos nos respectivos planos de execução, no Apêndice A e B para as consultas sem índice e no Apêndice C e D para as consultas com utilização de índice.

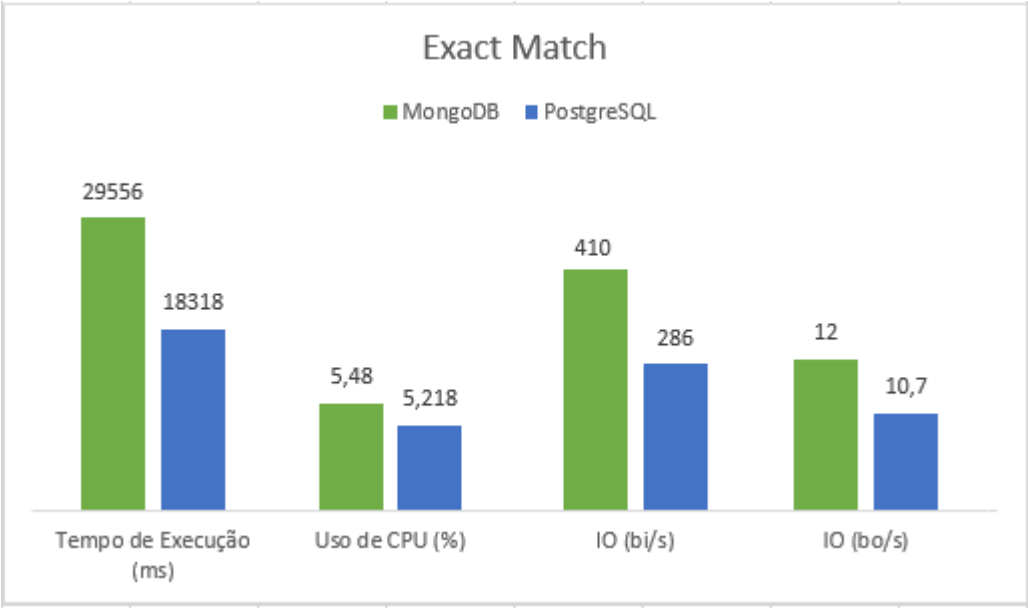
Quardo 3 – Mínimos e máximos do full scan sem índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	20183	25633	118230	122034
Uso de CPU (%)	5,76	7,62	17,2	19
I/O (bi/s)	305	440	730	853
I/O (bo/s)	28	58	230	343

Fonte: elaborada pelo autor

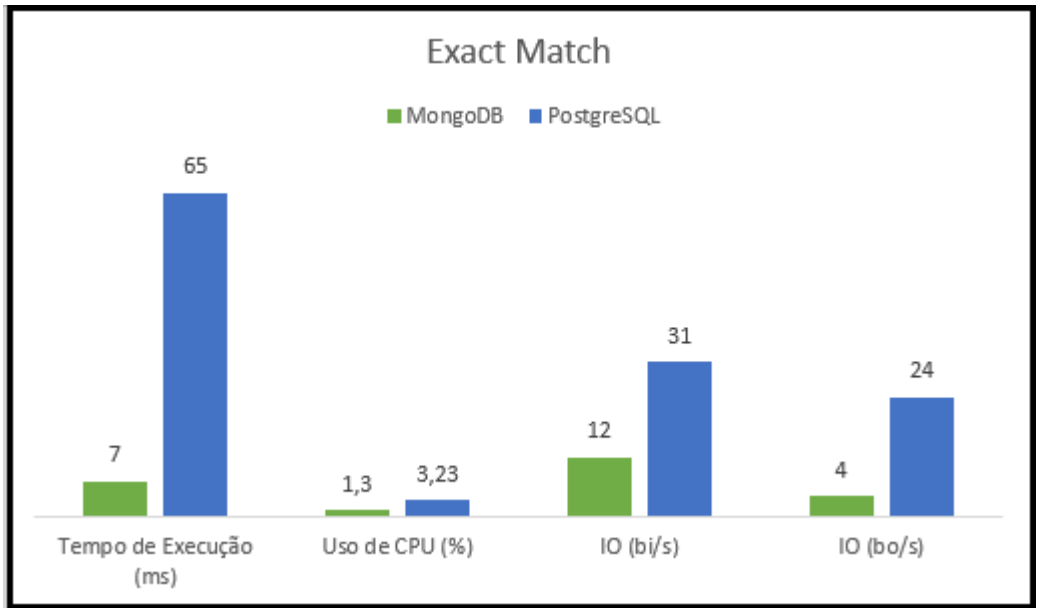
5.2 Exact match

Figura 5 - Exact match sem índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

Figura 6 - Exact match com índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

No *exact match*, temos um comportamento a ser observado com mais atenção, onde quando não utilizado o índice, o MongoDB tem um desempenho inferior ao PostgreSQL quanto a métrica tempo de execução. Isso acontece, pois a consulta no MongoDB mesmo tendo um filtro especificado como condição, no caso o id da trajetória, ainda assim percorre todos os

dados do dataset de uma ponta a outra, para só depois então retornar a projeção da consulta. Ou seja, o SGDB realiza um *collscan*, percorre toda a coleção de documentos, depois uma *projection* do documento que atende as condições de projeção. O mesmo acontece para o PostgreSQL que também realiza uma varredura em toda a tabela.

Quando aplicado o índice, os resultados se invertem e o MongoDB se sobresai nos testes. Isso acontece, pois como dito anteriormente todo documento da coleção de documentos no MongoDB, possui uma chave única de identificação que identifica o documento globalmente dentro da coleção, permitindo acesso direto aquele documento e quando utilizamos o índice no MongoDB esse acesso direto é acionado. Por isso obtívemos um tempo de execução, operações de I/O e uso de CPU com valores tão baixos. O índice no PostgreSQL trabalha de uma forma diferente, pois o índice arbitrário escolhido pelo próprio sistema gerenciador de banco de dados, foi o índice Bitmap, que monta um mapa de bits para todas as linhas da tabela, contendo os valores possíveis da coluna indexada. O PostgreSQL tinha várias opções de índice para escolher, como mencionado no início do capítulo. O Bitmap foi a escolha, pois o índice foi criado em cima de uma coluna, o id, que possuia uma baixa cardinalidade, ou seja, em que os valores se repetiam muitas vezes. O SGBD grava um bit 1 onde o valor existe em uma tupla e 0 para os valores que não existem nesta tupla. Portanto, assim otimizando-a obtendo melhores resultados em todas as métricas. O leitor deve assumir essa explicação sobre índices para todos os resultados que utilizam índice neste trabalho.

Os custos desta consulta podem ser lidos em seus respectivos planos de execução, Apêndice E e F para as consultas que não utilizaram índice e Apêndice G e H para as que utilizaram índice.

Quardo 4 – Mínimos e máximos do exact match sem índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	28465	32022	17202	20425
Uso de CPU (%)	5,17	5,60	5,08	5,57
I/O (bi/s)	409	411	246	320
I/O (bo/s)	11	13	9	12

Fonte: elaborada pelo autor

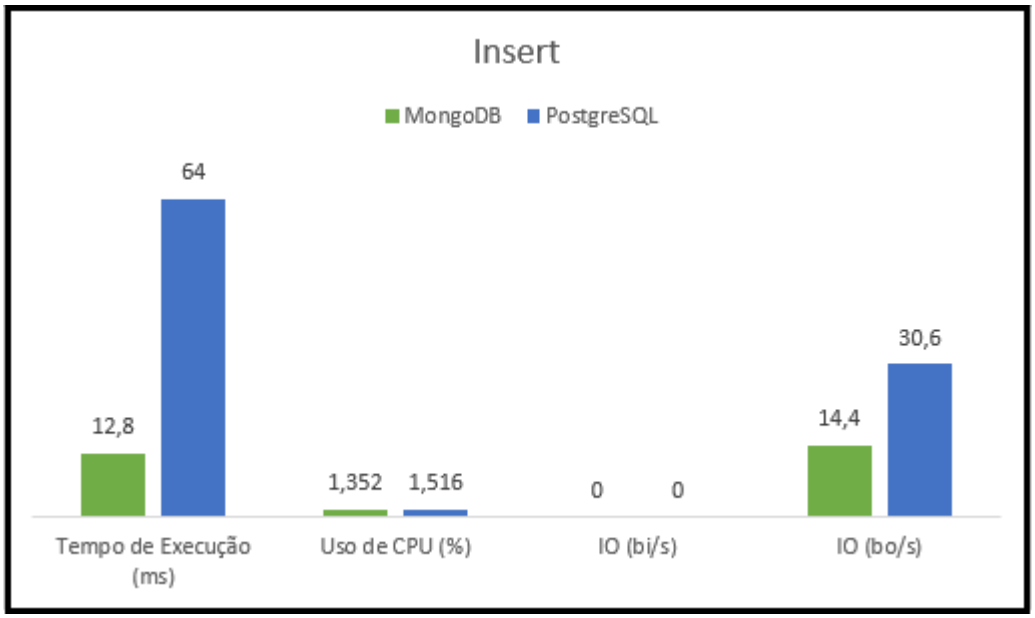
Quadro 5 – Mínimos e máximos do exact match com índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	4	14	45	95
Uso de CPU (%)	1	1,7	2,48	4,3
I/O (bi/s)	10	14	16	65
I/O (bo/s)	2	8	10	32

Fonte: elaborada pelo autor

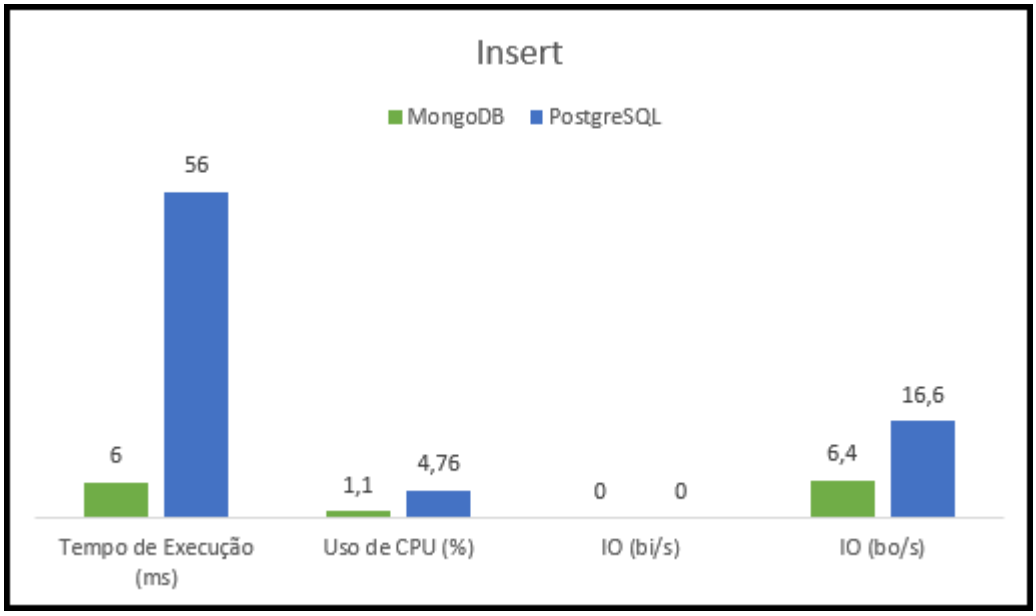
5.3 Insert

Figura 7 - Insert sem índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

Figura 8 - Insert com índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

A operação de *insert* foi realizada para inserir um registro a cada rodada de experimentos e partir daí mensurar suas métricas. Nestes gráficos tanto para o que utiliza índice e o que não utiliza índice, o MongoDB teve um melhor desempenho em todas as métricas, mas nessa operação o que mais contou para o desempenho, foi a forma como os SGBDs implementam os modelos de dados. Como o MongoDB não preza pelo esquema definido, um novo documento de qualquer estrutura pode ser adicionado em uma coleção a qualquer momento com um custo quase mínimo para o sistema gerenciador de banco de dados, tendo mais interferência da variação de máquina do que do próprio sistema em si. Nota-se também que nenhuma operação de leitura de disco foi realizada nos dois sistemas, pelo fato de ser uma inserção. Os planos de execução para a operação no PostgreSQL, com e sem índice, podem encontrados respectivamente no Apêndice I e J. Já o MongoDB não fornece um plano de execução para operação de inserção. O fato de verificar a inserção considerando que o índice é presente no banco de dados foi para investigar as mudanças de desempenho considerando também, a escrita no índice.

Quardo 6 – Mínimos e máximos do insert sem índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	10	20	43	84
Uso de CPU (%)	1,11	1,16	1,01	2,77
I/O (bi/s)	0	0	0	0
I/O (bo/s)	12	18	21	48

Fonte: elaborada pelo autor

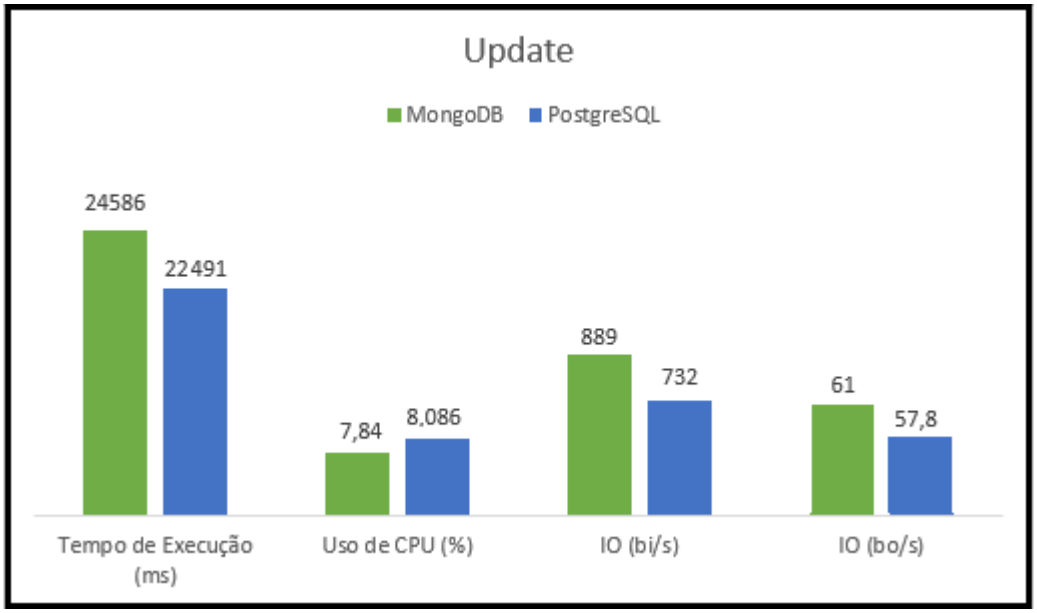
Quadro 7 – Mínimos e máximos do insert com índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	3	12	32	76
Uso de CPU (%)	0,9	1,2	2,23	5,44
I/O (bi/s)	0	0	0	0
I/O (bo/s)	2	11	11	26

Fonte: elaborada pelo autor

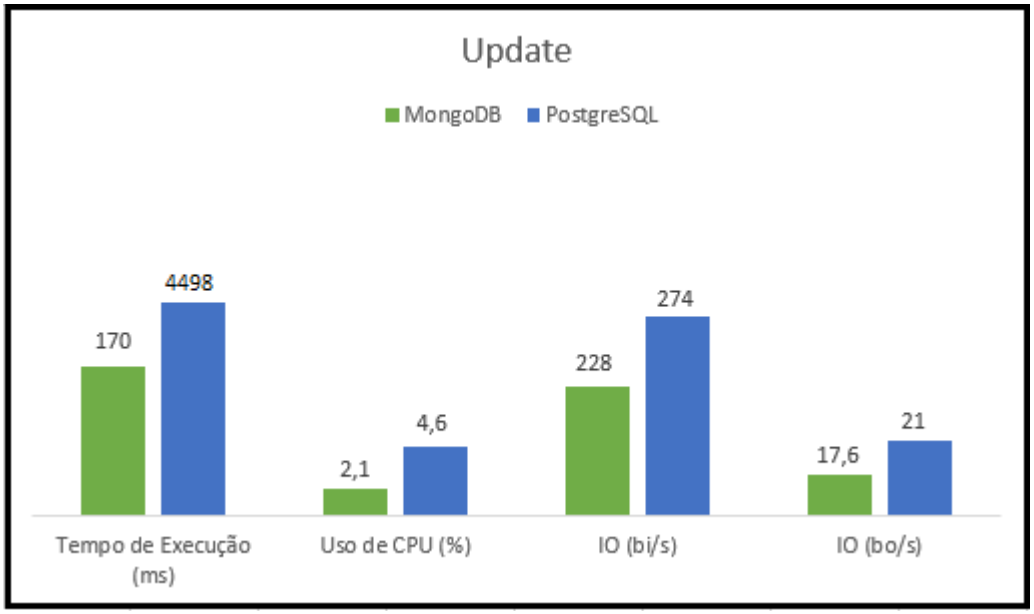
5.4 Update

Figura 9 - Update sem índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

Figura 10 - Update com índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

A operação de *update* realiza a atualização de todos os registros que atendem a condição de atualização. Como na consulta de projeção, *exact match*, aqui temos duas situações onde em uma, sem índice, o PostgreSQL tem melhor desempenho e na outra, com índice, que o MongoDB tem melhor desempenho. A causa desse comportamento é o mesmo que ocorre no *exact match*, onde o MongoDB quando não utiliza índice realiza primeiramente um collscan para depois, neste caso, atualizar os documentos e quando utiliza índice realiza o acesso direto. No caso do PostgreSQL quando não utiliza índice, realiza um seqscan utilizando um filtro interno que quando encontra a condição necessária para atualizar as tuplas, elimina as outras que restaram do plano e finaliza a consulta. Quando PostgreSQL utiliza o índice tem o aumento de desempenho, porém como o MongoDB acessa seus documentos diretamente tem um desempenho melhor ainda. Os planos das consultas desta operação se encontram respectivamente no Apêndice K e L para as consultas sem índice e no Apêndice M e N para as consultas com índice.

Quardo 8 – Mínimos e máximos do update sem índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	23970	25610	20549	23770
Uso de CPU (%)	7,03	8,76	7,18	8,81
I/O (bi/s)	877	913	710	753
I/O (bo/s)	58	65	54	58

Fonte: elaborada pelo autor

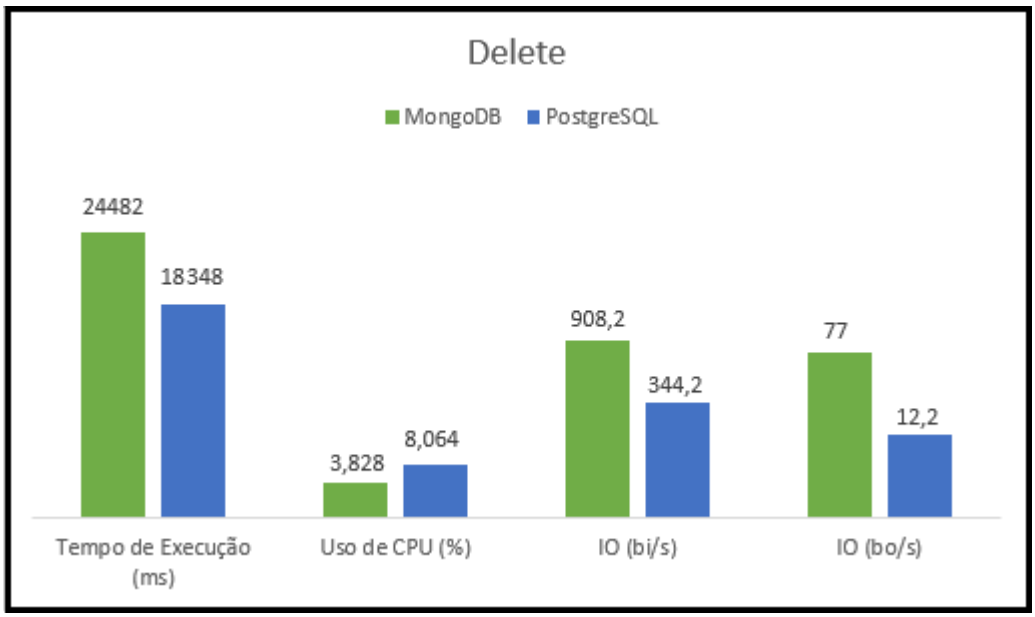
Quadro 9 – Mínimos e máximos do update com índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	85	230	2680	5260
Uso de CPU (%)	1,1	2,2	3,6	5,6
I/O (bi/s)	148	286	192	324
I/O (bo/s)	15	21	17	27

Fonte: elaborada pelo autor

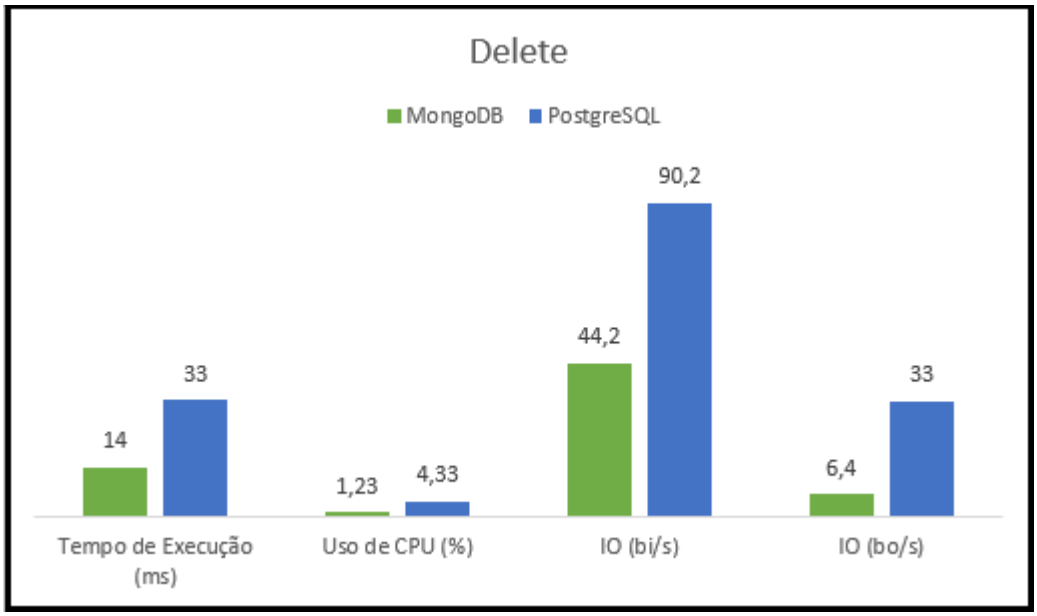
5.5 Delete

Figura 11 - Delete sem índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

Figura 12 - Delete com índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

No *delete* são excluídos todos os registros e documentos que atendem a condição (mais de 1000 tuplas foram removidas), nesses gráficos ocorrem o mesmo comportamento das operações de *exact match* e *update*.

Neste ponto o leitor deve estar se perguntando porque nestas consultas ocorre esse comportamento repetitivo de desempenho. Esse comportamento padrão de desempenho nas operações com e sem indice, ocorre porque essas operações são executadas com uma determinada condição e independente da operação, seja ela: projeção, atualização ou deleção. Os sistemas gerenciadores de bancos de dados vão trabalhar a consulta usando esta condição e apresentando assim resultados com um padrão. Logo, nesta operação o MongoDB apresenta melhor desempenho quando se utilizado índice e o PostgreSQL apresenta melhor desempenho sobre o MongoDB quando não utilizado índice, porém melhora sua performance também utilizando o índice. Nos Apêndices O e P estão os planos de consulta sem índice e nos Apêndices Q e R estão os planos de consulta com índice.

Quardo 10 – Mínimos e máximos do delete sem índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	23770	26840	17632	19340
Uso de CPU (%)	3,66	4,09	7,20	8,95
I/O (bi/s)	884	930	301	392
I/O (bo/s)	12	13	77	77

Fonte: elaborada pelo autor

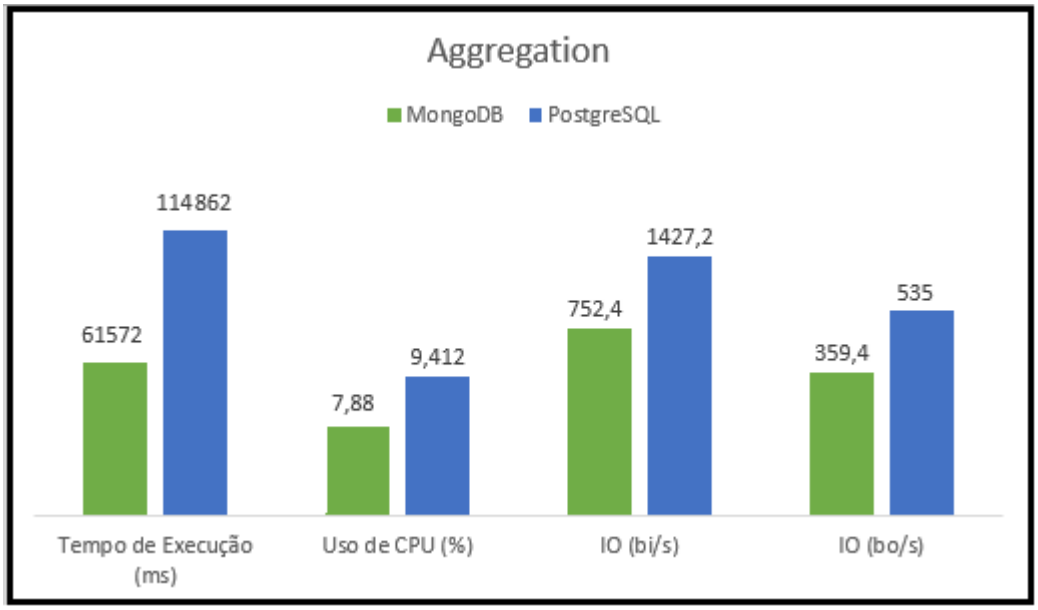
Quadro 11 – Mínimos e máximos do delete com índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	7	21	15	66
Uso de CPU (%)	0,9	1,33	2,76	5,43
I/O (bi/s)	44	45	90	91
I/O (bo/s)	4	7	17	69

Fonte: elaborada pelo autor

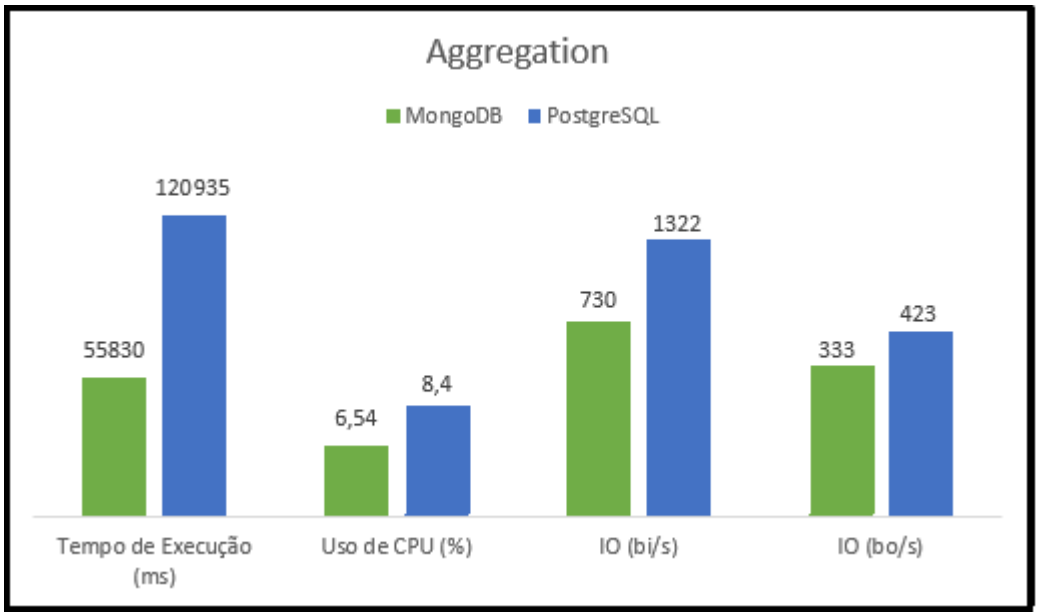
5.6 Aggregation

Figura 13 - Aggregation sem índice na coluna id da tabela trajetorias.



Fonte: elaborada pelo autor.

Figura 14 - Aggregation com índice na coluna id da tabela trajetorias.



Fonte:elaborada pelo autor.

A operação de *aggregation*, como a varredura de toda a tabela, mostra uma alta taxa de operações de I/O sendo realizadas, pois todos os blocos do conjunto de dados são percorridos. Perceba nos planos de consultas do apendice, que para a agregação ambos PostgreSQL e MongoDB utilizaram a varredura em todos os dados,ou seja, em nenhum caso

houve uso do índice. Algo já esperado já que a operação de agregação do *benchmark* não envolve indice na sua descrição.

Os dois SGDBs no momento de realizar a agregação, em seus planos, utilizam a chave de agrupamento para agrupar os valores necessários, depois disto um *collscan* é realizado para fazerem a relação chave valor, ou seja, chave de agrupamento e valor agrupado. Porém quando o MongoDB em seu plano de consulta define a chave de agrupamento, executa o agrupamento acessando diretamente cada valor e o agrupando, o PostgreSQL realiza uma ordenação interna de acordo com a chave de agrupamento, feito isso da-se início ao agrupamento, comprometendo o seu desempenho. O autor acredita que por isso o MongoDB tem melhor desempenho. A chave de agrupamento se refere ao valor passado na clálsula GROUP BY no PostgreSQL, no MongoDB o valor passado para a chave "_id". Os planos desta consulta estão nos Apêndices S e T para as consultas sem índice e nos Apêndices U e V para as consultas com índices.

Quardo 12 – Mínimos e máximos do aggregation sem índice.

	MongoDB		PostgreSQL	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	56330	66120	112600	116233
Uso de CPU (%)	7,59	8,33	7,54	11,41
I/O (bi/s)	712	844	1144	1614
I/O (bo/s)	174	790	526	546

Fonte: elaborada pelo autor

6 DISCUSSÃO

Os resultados apresentados na seção anterior comprovam que o objetivo principal foi atendio: (i) determinar em quais situações o SGBD não-relacional MongoDB apresenta melhor desempenho que o SGBD relacional PostgreSQL e em quais situações o PostgreSQL é melhor que o MongoDB, e (ii) todos os resultados de acordo com as métricas e o *microbenchmark* estabelecido pelo autor deste trabalho.

Com os gráficos que foram projetados dos experimentos, além de se analisar os resultados explanados na seção anterior, pode se notar que em todos os casos, exceto por um, o *exact match* sem índice, o MongoDB teve o menor uso de CPU em todos os experimentos e se levarmos em conta a variação de máquina, pode-se dizer que até mesmo no *exact match*, o MongoDB se saiu melhor. Dessa forma, têm-se uma confirmação de que o MongoDB faz um menor uso da CPU. Analisando mais a fundo também as operações de I/O mensuradas pelo *benchmark* criado pelo autor, é perceptível que o MongoDB também realiza menos operações de entrada e saída no disco.

Com relação as consultas executadas com índices, sem levar em conta as consultas que todos os blocos da tabela são percorridos, é mostrado que as duas ferramentas ganham uma quantidade significativa de desempenho.

No contexto de quando não se utilizado índice sobre a coluna id da tabela no banco de dados, o MongoDB se sobresai nas seguintes situações: *full scan*, *insert* e *aggregation*. O PostgreSQL tem melhor desempenho no: *exact-match*, *update* e *delete*. Quando se aplicado o índice, o MongoDB tem melhor desempenho em todas as situações, porém o PostgreSQL também ganha aumento de desempenho nas mesmas situações, mas como foi explicado anteriormente, o índice no MongoDB da a ele um maior potencial. O PostgreSQL é bom de se utilizar quando possuir muitas tabelas no banco de dados, em que é necessário correlacioná-las. O MongoDB é bom de se utilizar quando há uma necessidade de muitas operações de leitura e um conjunto de dados com uma grande quantidade de registros.

7 CONSIDERAÇÕES FINAIS

Saber quando escolher uma tecnologia para armazenamento dos seus dados em uma aplicação é fundamental e principalmente se a aplicação escolhida cobre suas necessidades. Nestre trabalho foram analisados duas tecnologias uma NoSQL e uma relacional, respectivamente MongoDB e PostgreSQL.

Ao final deste, podemos dizer quais as situações e como utilizarmos um dos SGBDs propostos. Tendo definido também métricas e um *benchmark* de operações que poderão ser utilizadas por outros trabalhos comparativos que levem em conta outros aspectos.

As principais dificuldades deste trabalho foram analisar as métricas selecionadas para a análise comaparativa, captar a o uso de CPU, as operações de I/O e analisar os planos de consulta dos SGBDs para obter os custos de cada consulta.

Para trabalhos futuros com os mesmos SGBDs, MongoDB e PostgreSQL, alguns aspectos devem ser considerados: executar o *microbenchmark* em um dataset com muitas tabelas correlacionadas; com o mesmo dataset utilizar índices espaciais nas colunas longitude e latitude, e explorar os resultados; realizar o trabalho utilizando os outros índices disponíveis para o PostgreSQL; re-executar os experimentos adicionando a métrica de uso de memória; realizar a análise comparativa com operações do tipo junção.

REFERÊNCIAS

ABRAMOVA, Veronika; BERNARDINO, Jorge. **NoSQL databases: MongoDB vs cassandra**. In: Proceedings of the International C* Conference on Computer Science and Software Engineering. ACM, 2013. p. 14-22.

DA SILVA, Ticiana LC et al. **Towards non-intrusive elastic query processing in the cloud**. In: Proceedings of the fourth international workshop on Cloud data management. ACM, 2012. p. 9-16.

DE DIANA, Mauricio; GEROSA, Marco Aurélio. **Nosql na web 2.0: Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0**. 2010.

HECHT, Robin; JABLONSKI, Stefan. Nosql evaluation. In: International Conference on Cloud and Service Computing. 2011. p. 336-41.

KOKAY, Marília Cavalcante. **Banco de dados NoSQL: Um novo paradigma**. Disponível em <http://www.devmedia.com.br/websys.5/webreader.asp?cat=2&artigo=4773&revista=sqlmagazine_102#a-4773>. Em: 14 abr. 2015.

LI, Yishan; MANOHARAN, Sathiamoorthy. **A performance comparison of SQL and NoSQL databases**. In: Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on. IEEE, 2013. p. 15-19.

LÓSCIO, Bernadette Farias; OLIVEIRA, Hélio Rodrigues de; PONTES, Jonas César de Sousa. **NoSQL no desenvolvimento de aplicações Web colaborativas**. VIII SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, Paraty, RJ: SBC, 2011.

MACHADO, Felipe Nery Rodrigues; DE ABREU, Maurício Pereira. **Projeto de banco de dados: uma visão prática**. Ed. Érica, 1996.

MARTINS FILHO, Marcos André Pereira; DE GRADUAÇÃO, Trabalho. **SQL X NoOSQL: Análise de desempenho do uso do MongoDB em relação ao uso do PostgreSQL**. 2015.

MCMURTRY, Douglas et al. **Data Access for Highly- Scalable Solutions: Using SQL, NoSQL, and Polyglot Persistence**. Disponível em: <http://www.microsoft.com/en-us/download/details.aspx?id=40327>. Em: 8 maio 2015.

NoSQL Databases. Disponível em <http://nosql-database.org/>. Em: 3 maio 2015.

PARKER, Zachary; POE, Scott; VRBSKY, Susan V. **Comparing nosql mongodb to an sql db**. In: Proceedings of the 51st ACM Southeast Conference. ACM, 2013. p. 5.

STONEBRAKER, Mike et al. **C-store: a column-oriented DBMS**. In: Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, 2005. p. 553-564.

STRAUCH, Christof; SITES, Ultra-Large Scale; KRIHA, Walter. **NoSQL databases**. Lecture Notes, Stuttgart Media University, 2011.

YUAN, Jing; ZHENG, Yu; XIE, Xing; SUN, Guangzhong. **Driving with knowledge from the physical world**. In The 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining, KDD'11, New York, NY, USA, 2011. ACM.

YUAN, Jing; ZHENG, Yu; ZHANG, Chengyang; XIE, Wenlei; XIE, Xing; SUN, Guangzhong; HUANG, Yan. **T-drive: driving directions based on taxi trajectories**. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '10, pages 99-108, New York, NY, USA,2010. ACM.

APÊNDICES

APÊNDICE A – Plano de Consulta MongoDB Full Scan sem Índice


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "test.rotas",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [ ]
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 17661819,
                "executionTimeMillis" : 25318,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 17661819,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "nReturned" : 17661819,
                        "executionTimeMillisEstimate" : 23318,
                        "works" : 17661821,
                        "advanced" : 17661819,
                        "needTime" : 1,
                        "needYield" : 0,
                        "saveState" : 139284,
                        "restoreState" : 139284,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 17661819
                }
        }
}
```

APÊNDICE B – Plano de Consulta PostgreSQL Full Scan sem Índice

```
Seq Scan on trajetoria
        (cost=0.00..119546 rows=17664961 width=28)
        (actual time=17.145..15544.728 rows=17657538 loops=1)
```

APÊNDICE C – Plano de Consulta MongoDB Full Scan com Índice


```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "test.rotas",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [ ]
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 17661821,
                "executionTimeMillis" : 25633,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 17661821,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [ ]
                        },
                        "nReturned" : 17661821,
                        "executionTimeMillisEstimate" : 22540,
                        "works" : 17661823,
                        "advanced" : 17661821,
                        "needTime" : 1,
                        "needYield" : 0,
                        "saveState" : 138205,
                        "restoreState" : 138205,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 17661821
                }
        }
}
```

APÊNDICE D – Plano de Consulta PostgreSQL Full Scan com Índice

```
Seq Scan on trajetoria
        (cost=0.00..126567 rows=17659116 width=28)
        (actual time=158.536..16247.038 rows=17659116 loops=1)
```

APÊNDICE E – Plano de Consulta MongoDB Exact Match sem Índice


```
{
"queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.rotas",
        "indexFilterSet" : false,
        "parsedQuery" : {
                "id" : {
                        "$eq" : 8343
                }
        },
        "winningPlan" : {
                "stage" : "PROJECTION",
                "transformBy" : {
                        "longitude" : 1,
                        "latitude" : 1,
                        "_id" : 0
                },
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "id" : {
                                        "$eq" : 8343
                                }
                        },
                        "direction" : "forward"
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1843,
        "executionTimeMillis" : 54191,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 17661819,
        "executionStages" : {
                "stage" : "PROJECTION",
                "nReturned" : 1843,
                "executionTimeMillisEstimate" : 29139,
                "works" : 17661821,
                "advanced" : 1843,
                "needTime" : 17660107,
                "needYield" : 0,
```

```
                "saveState" : 138438,
                "restoreState" : 138438,
                "isEOF" : 1,
                "invalidates" : 0,
                "transformBy" : {
                        "longitude" : 1,
                        "latitude" : 1,
                        "_id" : 0
                },
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "id" : {
                                        "$eq" : 8343
                                }
                        },
                        "nReturned" : 1843,
                        "executionTimeMillisEstimate" : 28465,
                        "works" : 17661821,
                        "advanced" : 1843,
                        "needTime" : 17660107,
                        "needYield" : 0,
                        "saveState" : 138438,
                        "restoreState" : 138438,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 17661819
                }
        }
    }
}
```


APÊNDICE F – Plano de Consulta PostgreSQL Exact Match sem Índice

```
Seq Scan on trajetoria t
        (cost=0.00..19245.01 rows=1843 width=16)
         (actual time=7432.932..17292.230 rows=5144 loops=1)
        Filter: (id = 8343)
                Rows Removed by Filter: 17652394
```

APÊNDICE G – Plano de Consulta MongoDB Exact Match com Índice


```
{
"queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.rotas",
        "indexFilterSet" : false,
        "parsedQuery" : {
                "id" : {
                        "$eq" : 8343
                }
        },
        "winningPlan" : {
                "stage" : "PROJECTION",
                "transformBy" : {
                        "longitude" : 1,
                        "latitude" : 1,
                        "_id" : 0
                },
                "inputStage" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "id" : 1
                                },
                                "indexName" : "id_1",
                                "isMultiKey" : false,
                                "isUnique" : true,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 1,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "id" : [
                                                "[8343.0, 8343.0]"
                                        ]
                                }
                        }
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1710,
        "executionTimeMillis" : 7,
        "totalKeysExamined" : 1710,
        "totalDocsExamined" : 1710,
```

```
"executionStages" : {
        "stage" : "PROJECTION",
        "nReturned" : 1710,
        "executionTimeMillisEstimate" : 10,
        "works" : 1711,
        "advanced" : 1710,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 13,
        "restoreState" : 13,
        "isEOF" : 1,
        "invalidates" : 0,
        "transformBy" : {
                "longitude" : 1,
                "latitude" : 1,
                "_id" : 0
        },
        "inputStage" : {
                "stage" : "FETCH",
                "nReturned" : 1710,
                "executionTimeMillisEstimate" : 10,
                "works" : 1711,
                "advanced" : 1710,
                "needTime" : 0,
                "needYield" : 0,
                "saveState" : 13,
                "restoreState" : 13,
                "isEOF" : 1,
                "invalidates" : 0,
                "docsExamined" : 1710,
                "alreadyHasObj" : 0,
                "inputStage" : {
                        "stage" : "IXSCAN",
                        "nReturned" : 1710,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 1711,
                        "advanced" : 1710,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 13,
                        "restoreState" : 13,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "keyPattern" : {
                                "id" : 1
                        },
                        "indexName" : "id_1",
                        "isMultiKey" : false,
                        "isUnique" : true,
                        "isSparse" : false,
```

```
                                        "isPartial" : false,
                                        "indexVersion" : 1,
                                        "direction" : "forward",
                                        "indexBounds" : {
                                                "id" : [
                                                        "[8343.0, 8343.0]"
                                                ]
                                        },
                                        "keysExamined" : 1710,
                                        "dupsTested" : 0,
                                        "dupsDropped" : 0,
                                        "seenInvalidated" : 0
                                }
                        }
                }
        }
}
```


APÊNDICE H – Plano de Consulta PostgreSQL Exact Match com Índice

```
Bitmap Heap Scan on trajetoria t
(cost=38.72..6733.66 rows=1843 width=16)
(actual time=0.365..0.913 rows=1694 loops=1)
 Recheck Cond: (id = 8343)
  ->  Bitmap Index Scan on id_index
        (cost=0.00..38.26 rows=1843 width=0)
        (actual time=0.345..0.345 rows=1694 loops=1)
              Index Cond: (id = 8343)
```

APÊNDICE I – Plano de Consulta PostgreSQL Insert sem Índice

```
Insert on trajetoria  (cost=0.00..0.01 rows=1 width=0)
                      (actual time=0.190..0.190 rows=0 loops=1)
  ->  Result
        (cost=0.00..0.01 rows=1 width=0) (actual time=0.001..0.002 rows=1 loops=1)
```

APÊNDICE J – Plano de Consulta PostgreSQL Insert com Índice

```
Insert on trajetoria
        (cost=0.00..0.01 rows=1 width=0) (actual time=0.056..0.056 rows=0 loops=1)
->  Result
        (cost=0.00..0.01 rows=1 width=0) (actual time=0.001..0.001 rows=1 loops=1)
```

APÊNDICE K – Plano de Consulta MongoDB Update sem Índice


```
{
"queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.rotas",
        "indexFilterSet" : false,
        "parsedQuery" : {
                "id" : {
                        "$eq" : 8343
                }
        },
        "winningPlan" : {
                "stage" : "UPDATE",
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "id" : {
                                        "$eq" : 8343
                                }
                        },
                        "direction" : "forward"
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 0,
        "executionTimeMillis" : 23770,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 17661821,
        "executionStages" : {
                "stage" : "UPDATE",
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 20549,
                "works" : 17661824,
                "advanced" : 0,
                "needTime" : 17661823,
                "needYield" : 0,
                "saveState" : 138102,
                "restoreState" : 138102,
                "isEOF" : 1,
                "invalidates" : 0,
                "nMatched" : 1843,
                "nWouldModify" : 1843,
                "nInvalidateSkips" : 0,
                "wouldInsert" : false,
                "fastmod" : false,
                "fastmodinsert" : false,
```

```
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "id" : {
                                        "$eq" : 8343
                                }
                        },
                        "nReturned" : 1843,
                        "executionTimeMillisEstimate" : 33990,
                        "works" : 17661823,
                        "advanced" : 1843,
                        "needTime" : 17660109,
                        "needYield" : 0,
                        "saveState" : 139815,
                        "restoreState" : 139815,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 17661821
                }
        }
    }
}
```


APÊNDICE L – Plano de Consulta PostgreSQL Update sem Índice

```
Update on trajetoria  (cost=0.00..350808.40 rows=0 width=10)
                      (actual time=17235.209..17235.209 rows=1713  loops=1)
 ->  Seq Scan on trajetoria
        (cost=0.00..350808.40 rows=1713 width=10)
        (actual time=17212.379..17214.565 rows=1843 loops=1)
Filter: (id = 8343)
        Rows Removed by Filter: 1765582
```

APÊNDICE M – Plano de Consulta MongoDB Update com Índice


```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.rotas",
        "indexFilterSet" : false,
        "parsedQuery" : {
            "id" : {
                "$eq" : 8343
            }
        },
        "winningPlan" : {
            "stage" : "UPDATE",
            "inputStage" : {
```

```
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "id" : 1
                                },
                                "indexName" : "id_1",
                                "isMultiKey" : false,
                                "isUnique" : true,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 1,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "id" : [
                                                "[8343.0, 8343.0]"
                                        ]
                                }
                        }
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 0,
        "executionTimeMillis" : 170,
        "totalKeysExamined" : 1710,
        "totalDocsExamined" : 1710,
        "executionStages" : {
                "stage" : "UPDATE",
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 70,
                "works" : 1712,
                "advanced" : 0,
                "needTime" : 1711,
                "needYield" : 0,
                "saveState" : 14,
                "restoreState" : 14,
                "isEOF" : 1,
                "invalidates" : 0,
                "nMatched" : 1710,
                "nWouldModify" : 1710,
                "nInvalidateSkips" : 0,
                "wouldInsert" : false,
                "fastmod" : false,
                "fastmodinsert" : false,
                "inputStage" : {
                        "stage" : "FETCH",
                        "nReturned" : 1710,
```

```
                    "executionTimeMillisEstimate" : 20,
                    "works" : 1711,
                    "advanced" : 1710,
                    "needTime" : 0,
                    "needYield" : 0,
                    "saveState" : 1724,
                    "restoreState" : 1724,
                    "isEOF" : 1,
                    "invalidates" : 0,
                    "docsExamined" : 1710,
                    "alreadyHasObj" : 0,
                    "inputStage" : {
                        "stage" : "IXSCAN",
                        "nReturned" : 1710,
                        "executionTimeMillisEstimate" : 10,
                        "works" : 1711,
                        "advanced" : 1710,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 1724,
                        "restoreState" : 1724,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "keyPattern" : {
                            "id" : 1
                        },
                        "indexName" : "id_1",
                        "isMultiKey" : false,
                        "isUnique" : true,
                        "isSparse" : false,
                        "isPartial" : false,
                        "indexVersion" : 1,
                        "direction" : "forward",
                        "indexBounds" : {
                            "id" : [
                                "[8343.0, 8343.0]"
                            ]
                        },
                        "keysExamined" : 1710,
                        "dupsTested" : 0,
                        "dupsDropped" : 0,
                        "seenInvalidated" : 0
                    }
                }
            }
        }
    }
```

APÊNDICE N – Plano de Consulta PostgreSQL Update com Índice

```
Update on trajetoria
        (cost=38.72..6733.74 rows=0 width=10)
        (actual time=345.725..345.725 rows=1713  loops=1)"
 ->  Bitmap Heap Scan on trajetoria
        (cost=38.72..6733.74 rows=1713 width=10)
         (actual time=1.278..2.814 rows=1843 loops=1)
        Recheck Cond: (id = 8343)"
                ->  Bitmap Index Scan on id_index
                        (cost=0.00..38.26 rows=1843 width=0)
                        (actual time=1.149..1.149 rows=6114 loops=1)
                                Index Cond: (id = 8343)
```

APÊNDICE O – Plano de Consulta MongoDB Delete sem Índice


```
{
"queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.rotas",
        "indexFilterSet" : false,
        "parsedQuery" : {
                "id" : {
                        "$eq" : 8343
                }
        },
        "winningPlan" : {
                "stage" : "DELETE",
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "id" : {
                                        "$eq" : 8343
                                }
                        },
                        "direction" : "forward"
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 0,
        "executionTimeMillis" : 35234,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 17661821,
        "executionStages" : {
                "stage" : "DELETE",
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 26840,
```

```
                "works" : 17661823,
                "advanced" : 0,
                "needTime" : 17661822,
                "needYield" : 0,
                "saveState" : 138120,
                "restoreState" : 138120,
                "isEOF" : 1,
                "invalidates" : 0,
                "nWouldDelete" : 1843,
                "nInvalidateSkips" : 0,
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "id" : {
                                        "$eq" : 8343
                                }
                        },
                        "nReturned" : 1843,
                        "executionTimeMillisEstimate" : 23770,
                        "works" : 17661823,
                        "advanced" : 1843,
                        "needTime" : 17660109,
                        "needYield" : 0,
                        "saveState" : 139833,
                        "restoreState" : 139833,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 17661821
                }
        }
    }
}
```


APÊNDICE P – Plano de Consulta PosgreSQL Delete sem Índice

```
Delete on trajetoria
        (cost=0.00..350843.49 rows=0 width=6)
        (actual time=18902.243..18902.243 rows=1713  loops=1)
->  Seq Scan on trajetoria
        (cost=0.00..350843.49 rows=1713 width=6)
         (actual time=18876.334..18877.209 rows=1843 loops=1)
                Filter: (id = 8343)"
                        Rows Removed by Filter: 17655826
```

APÊNDICE Q – Plano de Consulta MongoDB Delete com Índice


```
{
"queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.rotas",
        "indexFilterSet" : false,
        "parsedQuery" : {
                "id" : {
                        "$eq" : 8343
                }
        },
        "winningPlan" : {
                "stage" : "DELETE",
                "inputStage" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "id" : 1
                                },
                                "indexName" : "id_1",
                                "isMultiKey" : false,
                                "isUnique" : true,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 1,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "id" : [
                                                "[8343.0, 8343.0]"
                                        ]
                                }
                        }
                }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 0,
        "executionTimeMillis" : 14,
        "totalKeysExamined" : 1710,
        "totalDocsExamined" : 1710,
        "executionStages" : {
                "stage" : "DELETE",
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 20,
                "works" : 1711,
                "advanced" : 0,
```

```
"needTime" : 1710,
"needYield" : 0,
"saveState" : 13,
"restoreState" : 13,
"isEOF" : 1,
"invalidates" : 0,
"nWouldDelete" : 1710,
"nInvalidateSkips" : 0,
"inputStage" : {
        "stage" : "FETCH",
        "nReturned" : 1710,
        "executionTimeMillisEstimate" : 20,
        "works" : 1711,
        "advanced" : 1710,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 1723,
        "restoreState" : 1723,
        "isEOF" : 1,
        "invalidates" : 0,
        "docsExamined" : 1710,
        "alreadyHasObj" : 0,
        "inputStage" : {
                "stage" : "IXSCAN",
                "nReturned" : 1710,
                "executionTimeMillisEstimate" : 0,
                "works" : 1711,
                "advanced" : 1710,
                "needTime" : 0,
                "needYield" : 0,
                "saveState" : 1723,
                "restoreState" : 1723,
                "isEOF" : 1,
                "invalidates" : 0,
                "keyPattern" : {
                        "id" : 1
                },
                "indexName" : "id_1",
                "isMultiKey" : false,
                "isUnique" : true,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 1,
                "direction" : "forward",
                "indexBounds" : {
                        "id" : [
                                "[8343.0, 8343.0]"
                        ]
                },
                "keysExamined" : 1710,
```

```
                                "dupsTested" : 0,
                                "dupsDropped" : 0,
                                "seenInvalidated" : 0
                        }
                }
        }
    }
}
```


APÊNDICE R – Plano de Consulta PostgreSQL Delete com Índice

```
Delete on trajetoria
        (cost=38.72..6733.77 rows=0 width=6)
        (actual time=33.488..33.488 rows=1585  loops=1)
->  Bitmap Heap Scan on trajetoria
        (cost=38.72..6733.77 rows=1585 width=6)
        (actual time=2.002..27.880 rows=1843 loops=1)"
        Recheck Cond: (id = 8343)
                ->  Bitmap Index Scan on id_index
                        (cost=0.00..38.26 rows=1843 width=0)
                        (actual time=1.700..1.700 rows=7545 loops=1)"
                                Index Cond: (id = 8343)
```

APÊNDICE S – Plano de Consulta MongoDB Aggregate sem Índice


```
{
"waitedMS" : 66330,
"stages" : [
        {
                "$cursor" : {
                        "query" : {

                        },
                        "fields" : {
                                "latitude" : 1,
                                "_id" : 0
                        },
                        "queryPlanner" : {
                                "plannerVersion" : 1,
                                "namespace" : "test.rotas",
                                "indexFilterSet" : false,
                                "parsedQuery" : {
                                        "$and" : [ ]
                                },
                                "winningPlan" : {
                                        "stage" : "COLLSCAN",
                                        "filter" : {
                                                "$and" : [ ]
                                        },
```

```
                                        "direction" : "forward"
                                },
                                "rejectedPlans" : [ ]
                        }
                }
        },
        {
                "$group" : {
                        "_id" : "$latitude",
                        "count" : {
                                "$sum" : {
                                        "$const" : 1
                                }
                        }
                }
        }
],
"ok" : 1
}
```


APÊNDICE T – Plano de Consulta PostgreSQL Aggregate sem Índice

```
GroupAggregate
        (cost=3399723.69..3532533.33 rows=29695 width=8)
         (actual time=114185.580..129683.126 rows=148400 loops=1)
        ->  Sort
                (cost=3399723.69..3443894.58 rows=17668359 width=8)
                (actual time=114153.657..124062.937 rows=17655826 loops=1)
                Sort Key: latitude"
                Sort Method: external merge  Disk: 310696kB"
                ->  Seq Scan on trajetoria t
                        (cost=0.00..306672.59 rows=17668359 width=8)
                        (actual time=12.182..15927.917 rows=17655826 loops=1)
```

APÊNDICE U – Plano de Consulta MongoDB Aggregate com Índice


```
{
"waitedMS" : 55830,
"stages" : [
        {
                "$cursor" : {
                        "query" : {

                        },
                        "fields" : {
                                "latitude" : 1,
                                "_id" : 0
                        },
                        "queryPlanner" : {
```

```
                                "plannerVersion" : 1,
                                "namespace" : "test.rotas",
                                "indexFilterSet" : false,
                                "parsedQuery" : {
                                        "$and" : [ ]
                                },
                                "winningPlan" : {
                                        "stage" : "COLLSCAN",
                                        "filter" : {
                                                "$and" : [ ]
                                        },
                                        "direction" : "forward"
                                },
                                "rejectedPlans" : [ ]
                        }
                }
        },
        {
                "$group" : {
                        "_id" : "$latitude",
                        "count" : {
                                "$sum" : {
                                        "$const" : 1
                                }
                        }
                }
        }
    ],
    "ok" : 1
}
```


APÊNDICE V – Plano de Consulta PostgreSQL Aggregate com Índice

```
GroupAggregate
        (cost=3399035.17..3531819.32 rows=29695 width=8)
        (actual time=105182.362..120934.805 rows=148400 loops=1)
        ->  Sort
                (cost=3399035.17..3443197.57 rows=17664961 width=8)
                (actual time=104387.995..114494.103 rows=17657538 loops=1)
                 Sort Key: latitude
                 Sort Method: external merge  Disk: 310712kB
                ->  Seq Scan on trajetoria t
                        (cost=0.00..306613.61 rows=17664961 width=8)
                        (actual time=18.848..12353.478 rows=17657538 loops=1)
```