UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MAIKE RONY SCHUERZOSKI

ESTUDO E APLICAÇÃO DE *TUNING* EM SISTEMA GERENCIADOR DE BANCO DE DADOS ORIENTADO A DOCUMENTOS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2019

MAIKE RONY SCHUERZOSKI

ESTUDO E APLICAÇÃO DE *TUNING* EM SISTEMA GERENCIADOR DE BANCO DE DADOS ORIENTADO A DOCUMENTOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Tarcizio Alexandre Bini

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

ESTUDO E APLICAÇÃO DE TUNING EM SISTEMA GERENCIADOR DE BANCO DE DADOS ORIENTADO A DOCUMENTOS

por

MAIKE RONY SCHUERZOSKI

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 05 de novembro de 2019 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Tarcizio Alexandre Bini
Orientador(a)

_____
Profa. Dra. Simone de Almeida
Membro titular

_____
Profa. Dra. Simone Bello Kaminski Aires
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. André Pinz Borges
Coordenador do curso

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder a vida e o ensejo de trilhar nesse caminho apinhado de entusiasmos, dificuldades, tristezas, fracassos e sucessos. Sempre me proporcionando perseverança ao longo de minha existência, me concedendo saúde e força para sobrepujar as dificuldades me guiando ao caminho certo.

Aos meus familiares, em especial meus pais Marcilio e Luciana, por todo o esforço investido em minha educação, por sempre estarem ao meu lado me amparando ao longo de toda a minha trajetória acadêmica.

Ao meu professor orientador Tarcizio Alexandre Bini, pela sua paciência e muitas vezes a falta dela, sempre me auxiliando nas dúvidas e dificuldades, por todo o seu tempo dedicado e principalmente pelo incentivo perpétuo no decorrer do trabalho acreditando no meu potencial.

A todos os professores do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná que sempre transmitiram seu saber com sublime destreza, proporcionando um ensino de qualidade a todos os alunos.

A todos que contribuíram de forma direta ou indireta para realização deste trabalho.

“A persistência é o menor caminho do êxito”.

(CHARLES CHAPLIN)

RESUMO

SCHUERZOSKI, Maike. **Estudo e Aplicação de *Tuning* em Sistema Gerenciador de Banco de Dados Orientado a Documentos**. 2019. 2019. 71 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnologia Federal do Paraná. Ponta Grossa 2019.

O processamento de grandes volumes de dados em tempo hábil revelou limitações de processamento e armazenamento dos sistemas gerenciadores de banco de dados relacionais. Para contornar estes problemas, surgiram novos sistemas que não fazem uso do modelo relacional como forma de armazenamento e organização de dados, conhecidos como *NoSQL*. Entre esses sistemas, um dos mais utilizados são os orientados a documentos, que possuem como características marcantes a ausência de relações e esquemas pré-definidos. Mesmo suprindo as limitações de processamento e armazenamento dos sistemas gerenciadores de banco de dados relacionais, os sistemas orientados a documentos necessitam de melhorias de desempenho. Uma técnica com a finalidade de maximizar o desempenho é o *tuning*, atividade que consiste do ajuste de parâmetros que podem afetar direta ou indiretamente o desempenho do sistema de banco de dados. Tal técnica é amplamente utilizada em sistemas gerenciadores de banco de dados relacionais e pode ser aplicada também às bases *NoSQL*. Neste contexto, o presente trabalho busca realizar o estudo e a aplicação de *tuning* sobre um sistema gerenciador de banco de dados orientado a documentos. Criou-se um ambiente experimental e diferentes cenários de testes, para avaliar o impacto do *tuning* no desempenho desse sistema.

Palavras-chave: Sistema Gerenciador de Banco de Dados Orientado a Documentos. Tuning. NoSQL. Banco de Dados Não Relacional.

ABSTRACT

SCHUERZOSKI, Maike. **Study and Application of Tuning in System Manager of Database Oriented to Documents**. 2019. 71 p. Work of Conclusion Course (Graduation in Technology in Systems Analysis and Development) - Federal Technology University - Paraná. Ponta Grossa, 2019.

The processing of large volumes of data in a timely manner revealed process limitations and storage of relational database management system. To get around these problems, appeared new systems that do not use the relational model as a way of storing and data organization, called NoSQL. Between these systems, one of the most used are document oriented, which have outstanding significant characteristics the absence of predefined relationships and schemas. Even supplying the process limitations and storage of relational database management system, the document-oriented systems may require a performance improvement. One technique in order to maximize the performance is the Tuning, activity that consists in the parameters adjustment that can affect directly or indirectly the database system performance. This technique is widely used in relational database management system and it can be applied the NoSQL bases. In this context, this present work seek to execute a research and the tuning application about a document oriented database manager system. An experimental ambient was created and different test scenarios, to evaluate the tuning influence in this system performance.

Keywords: Document Oriented Database Manager System. Tuning. NoSQL. Nonrelational Database.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ACRÔNIMOS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
BSON	*Binary JSON*
CMS	*Centers for Medicare & Medicaid Services*
CSV	*Comma-Separated-Values*
GB	*Gigabytes*
GHz	*GigaHertz*
JSON	*JavaScript Object Notation*
LTS	*Long Term Support*
MB	*Megabytes*
NoSQL	*Not Only SQL*
RAM	*Random Access Memory*
RPM	Rotação por Minuto
SATA	*Serial Advanced Technology Attachment*
SGBD	Sistema Gerenciado de Banco de Dados
SGBDR	Sistema Gerenciado de Banco de Dados Relacional
SO	Sistema Operacional
SQL	*Structured Query Language*
TB	*Terabytes*
THP	*Transparent Huge Pages*
TLB	*Translation Lookaside Buffer*
TSV	*Tab Separated Values*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A necessidade de armazenar dados de diferentes tipos e padrões mantendo a integridade motivou o surgimento dos Sistemas Gerenciadores de Banco de Dados (SGBDs). O objetivo é gerenciar e controlar os dados, garantindo a recuperação e atualização dos mesmos de forma concorrente. Segundo Date (2000) registrar e manter todos os dados consistentes são as características fundamentais dos SGBDs.

O armazenamento de dados baseado em relações é característica dos Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDRs). As relações são compostas por linhas (tuplas), que podem representar diferentes tipos de dados e também por colunas (atributos). Sua característica fundamental é preservar a integridade dos dados, para que os mesmos não se corrompam e gerem inconsistência.

Como forma de garantir a integridade dos dados nas transações, os SGBDRs fazem uso das propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade). A *atomicidade* garante que uma transação seja totalmente executada. A *consistência* assegura que as modificações realizadas nos dados sejam consistentes. Por intermédio da propriedade de *isolamento*, uma transação é executada de forma isolada, não interferindo na execução das demais. Por fim, os dados devem sempre estar disponíveis ao final de uma transação, característica garantida pela propriedade de *durabilidade*.

Os SGBDRs apesar de serem amplamente utilizados possuem limitações. O crescente aumento do volume de dados contribuiu negativamente ao modelo relacional, tornando o sistema limitado em relação ao processamento (SADALAGE; FOWLER, 2013). Esse obstáculo impacta diretamente o desempenho das aplicações. As demandas por acessos concorrentes e o processamento de grande quantidade de dados forçaram os SGBDRs a evoluírem em sua estrutura de armazenamento.

Segundo Brito (2010) os modelos não relacionais surgiram como alternativa às limitações do relacional. O termo utilizado para descrevê-los foi *NoSQL (Not Only SQL)* traduzido como (não apenas SQL), que contempla uma classe de SGBDs que não faz uso do modelo relacional para o armazenamento de dados. Segundo Sadalage e Fowler (2013) e Brito (2010) os bancos de dados *NoSQL* são classificados nas seguintes categorias de acordo com o modelo de armazenamento de dados: *chave-valor, família de colunas, baseados em grafos e orientados a documentos.*

O modelo chave-valor é resumido pela sua simplicidade, armazena seus dados em uma única tabela (SADALAGE; FOWLER, 2013). Dentro dessa tabela há dois campos, uma chave exclusiva e seu respectivo valor. No modelo orientado a grafos os dados são representados por meio da Teoria de Grafos (PENTEADO, 2014). Sua estrutura é composta de 3 componentes: os nós (vértices do grafo), os relacionamentos (arestas) e as propriedades (atributos). LÓSCIO *et al*. (2011) descrevem que o modelo orientado a família de colunas armazena seus dados em linhas dentro de uma tabela. Sendo cada tabela exclusiva, definida pelo campo identificador. Cada linha é um atributo da tabela, podendo haver mais de uma linha.

O modelo orientado a documentos por usa vez tem como principal característica a ausência de um esquema pré-definido, assim como de regras para sua concepção. Segundo Sadalage e Fowler (2013) o documento é o conceito principal do modelo, onde os dados são armazenados no formato chave e valor. Tal modelo tem como característica marcante, o processamento de grande quantidade de dados. Abdicando das propriedades ACID, o modelo orientado a documentos tem o intuito de obter melhorias de desempenho em comparação ao modelo relacional.

Maximizar o desempenho de SGBDs e atenuar a utilização de recursos computacionais (processamento, memória e acesso a disco) são tarefas desafiadoras e assunto de diversos trabalhos, (BINI, 2009); (DOMINICO, 2013); (BINI, 2014); (KUSS, 2017). Identificar operações lentas e gargalos em meio às transações, requer alto conhecimento por parte do DBA (*Database Administrator*). Modificar parâmetros internos ou externos ao SGBD que afetam seu desempenho, demanda tempo e testes exaustivos para avaliação de resultados. Tais modificações são definidas como *tuning* em banco de dados, que segundo Dominico (2013) e Tramontina (2008) consiste no ajuste fino do SGBD, objetivando reduzir o tempo de resposta às solicitações a ele impostas. Esta técnica é amplamente utilizada em SGBDRs e pode também ser aplicada à sistemas não relacionais.

1.1 OBJETIVOS

Aplicar técnicas de *tuning* em um SGBD não relacional orientado a documentos para reduzir o tempo de execução de consultas é o objetivo geral deste trabalho.

Os objetivos específicos são:

- Realizar pesquisas sobre os modelos de armazenamento de dados, em especifico o modelo orientado a documentos;
- Analisar, escolher e instalar um SGBD nativo orientado a documentos;
- Identificar parâmetros sensíveis ao *tuning* do SGBD escolhido;
- Criar cenários de testes, compostos de bases de dados e um conjunto de consultas;
- Analisar os resultados constatando se houve ou não ganhos de desempenho no SGBD.

1.2 JUSTIFICATIVA

A comprovação da eficiência das técnicas de *tuning* em SGBDRs é apontada e justificada em estudos: (BINI, 2009); (DOMINICO, 2013); (BINI, 2014); (KUSS, 2017). Segundo Dominico (2013), à aplicação de tais técnicas contribuiu de forma positiva para melhoria de desempenho dos SGBDRs. Ciente dessa eficiência, também pode-se aplicar técnicas de *tuning* em sistemas *NoSQL*, mais especificamente orientado a documentos.

Observa-se uma escassez de trabalhos discutindo o *tuning* em banco de dados não convencionais. Muito se deve a tais sistemas terem sido concebidos recentemente, quando comparado aos relacionais. Porém em seus manuais é possível encontrar tópicos que sugerem estratégias para obter melhorias de desempenho, pautadas nos parâmetros de configuração do SGBD. Por meio desses materiais busca-se aplicar técnicas de *tuning* e verificar sua eficiência em um sistema de banco de dados orientado a documentos.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho encontra-se divido em 5 capítulos.

No Capítulo 2 encontram-se as definições sobre os diferentes modelos de armazenamento de dados comumente encontrados.

Considerando SGBDs orientado a documentos, o Capítulo 3 apresenta suas principais características e aplicações no mercado, além de descrever elementos internos do sistema de armazenamento de dados.

O Capítulo 4 define o conceito de *tuning,* assim como apresenta os principais parâmetros de configuração empregados na aplicação desta técnica, considerando um SGBD orientado a documentos.

No Capítulo 5 apresentam-se o ambiente experimental, suas características, a base de dados e os casos de testes empregados para a análise da aplicação de técnicas de *tuning* sobre o SGBD. Neste capítulo também é realizada a apresentação e discussão dos resultados obtidos.

O Apêndice A encerra esse documento, apresentando os planos de execução das consultas.

2 MODELO DE ARMAZENAMENTO DE DADOS

Define-se como banco de dados, um sistema computadorizado de manutenção de registros (dados). Sua principal funcionalidade é armazenar dados de forma que esses possam ser consultados e atualizados por usuários (DATE, 2004). Quando processados, esses dados geram informações. Pode-se afirmar que sistemas de banco de dados nada mais são do que um sistema de armazenamento de dados que se baseia em computador, em outras palavras, um sistema em que a finalidade é registrar e armazenar dados.

Os modelos de armazenamentos de dados são ferramentas conceituais, que tem como objetivo descrever como os dados são organizados pelo SGBD. Também regem a forma como os dados são inseridos, atualizados, excluídos e consultados pelos usuários (KORTH *et al*, 2006) (DATE, 2004). Nas próximas seções serão apresentados os modelos de armazenamentos empregados pelos principais SGBDs da atualidade.

2.1 MODELO RELACIONAL

O modelo de dados relacional teve sua origem no ano de 1970 no laboratório de pesquisa da IBM (*International Business Machines*) proposto por Edgar Frank Codd (CODD, 1972), baseado na teoria dos conjuntos e álgebra relacional. Segundo Korth *et al*. (2006) o modelo conquistou destaque devido a sua simplicidade e praticidade no armazenamento dos dados em uma estrutura similar a uma tabela. O modelo foi implementado a partir da década de 1980 em muitos sistemas tais como *Oracle* (ORACLE, 2019)*, MySql* (MYSQL, 2019)*, PostgreSQL* (POSTGRESQL, 2019), entre outros (ELMASRI; NAVATHE, 2011). Após a sua implementação, rapidamente se tornou o paradigma dominante de SGBD (RAMAKRISHNAN; GEHRKE, 2008).

Uma característica forte do modelo relacional são as restrições de integridade (DATE, 2004). Uma das restrições é a chave primaria, que garante a identificação exclusiva das tuplas. Outra restrição é a chave estrangeira, que referencia as relações por meio de um atributo, gerando um relacionamento. O modelo relacional permite aplicar um conjunto de regras visando estruturar o projeto de banco de dados. Essas

regras são chamadas de normalização, um procedimento cuja finalidade é reduzir a redundância de dados.

Além das restrições de integridade o modelo relacional garante a confiabilidade das transações. O SGBD permite a execução de transações de forma concorrentes, além de gerenciá-las. Uma transação é uma visão abstrata que o SGBD tem de um programa de usuário que é uma sequência de leituras (*reads*) e escritas (*writes*) (RAMAKRISHNAN, 2008). Para garantir a integridade do banco de dados nas transações foram criadas as propriedades **ACID** (**A**tomicidade, **C**onsistência, **I**solamento e **D**urabilidade). Segundo Ramakrishnan (2008) as propriedades são definidas:

- **Atomicidade:** No momento em que uma transação é enviada para o servidor de banco de dados, suas operações são completadas com sucesso ou revertidas totalmente. Com isso os dados sempre serão mantidos de maneira consistente.
- **Consistência:** Todas as modificações no banco serão feitas de maneira consistente. Em caso de falha de integridade dos dados, a transação será desfeita.
- **Isolamento:** Garante que cada transação feita no banco de dados seja executada de maneira isolada, ou seja, caso um usuário insira um registro em uma relação no banco de dados, outro usuário não poderá acessar a mesma.
- **Durabilidade:** Ao final de uma transação os dados estarão disponíveis no banco de dados, mesmo em caso de alguma falha catastrófica, como uma queda de energia, ocasionando uma interrupção na conexão com o banco de dados.

Os SGBDs relacionais oferecem aos usuários processos de validação, verificação e garantias de integridade dos dados, controle de concorrência, recuperação de falhas, segurança, controle de transações, otimização de consultas, dentre outros (BRITO, 2010). Todos os recursos oferecidos pelo SGBDs contribuíram para sua popularização. O acesso aos dados de forma simultânea e consistente com um sistema de recuperação de falhas também são características fortes dos SGBDs.

A constante utilização de sistemas computacionais em diversas aplicações contribuiu para o crescente aumento do volume de dados. Essa evolução não contribuiu com o modelo relacional, revelando suas limitações. A principal limitação do modelo é o processamento de grande volume de dados. Conforme a demanda de processamento aumenta, o sistema se torna ineficaz.

Uma maneira de contornar as limitações do modelo relacional é investir em equipamentos robustos, aumentando assim o número de servidores. Porém essa solução não será eficiente, devida a forma como os dados estão crescendo, irá gerar somente um alto custo computacional (BRITO, 2010).

Outra solução é a distribuição do banco de dados em várias máquinas, escalonando o banco de dados. A escalabilidade pode ocorrer de duas maneiras: horizontal e vertical. Define-se por escalabilidade horizontal a utilização de mais equipamentos para dividir o armazenamento e o processamento dos dados. Já na vertical ocorre o aumento dos recursos da máquina em que o sistema gerenciador do banco de dados está instalado.

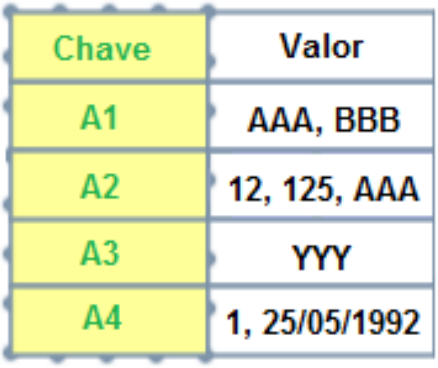
Segundo Leavitt (2010) percebeu-se obstáculos principalmente na organização dos dados no sistema distribuído, dificuldades principalmente em manipular dados particionados, devida a sua forma estruturada. Novas estratégias de armazenamento foram criadas como alternativa às limitações do modelo relacional. Surgiram assim os bancos de dados *NoSQL*, termo atual utilizado para representar uma ampla classe de SGBDs que não utilizam o modelo relacional como forma de armazenamento e manipulação de dados.

Os bancos de dados não relacionais permitem esquemas flexíveis e escabilidade com menor custo. Segundo Leavitt (2010) a sua principal vantagem é que, diferentemente dos bancos de dados relacionais, eles manipulam dados não estruturados. De acordo com (SADALAGE; FOWLER, 2013) e (BRITO, 2010), os bancos de dados *NoSQL* são classificados nas seguintes categorias de acordo com o modelo de armazenamento de dados: *chave-valor, família de colunas, baseados em grafos e orientados a documentos*.

2.2 MODELO ORIENTADO A CHAVE-VALOR

O modelo chave-valor é considerado o mais simples em relação aos demais modelos de banco de dados não relacionais, pois armazena seus dados em uma única tabela. Sua estrutura é uma tabela composta de dois campos, como uma tabela *hash* (SADALAGE; FOWLER, 2013). Um campo contendo uma chave exclusiva de identificação. Outro campo que armazena o valor dessa chave, podendo ser de diferentes tipos de entrada (números, textos, etc), conforme ilustrado na Figura 1.

Figura 1 – Representação do Modelo Chave-Valor



Chave	Valor
A1	AAA, BBB
A2	12, 125, AAA
A3	YYY
A4	1, 25/05/1992

Fonte: Autoria própria

Esse modelo se assemelha às definições de chave primaria nos bancos de dados relacionais, onde o campo de identificação exclusiva do modelo é considerado uma chave primária. O acesso aos dados é feito através desse campo exclusivo definido como “chave”, conforme ilustra a Figura 1. Geralmente a leitura de dados acessando a chave possui bom desempenho.

Uma das aplicações do modelo chave-valor é o gerenciamento de sessões *Web.* Quando o usuário realiza um *login* em um *site* o sistema cria uma sessão, registrando o momento que o usuário iniciou e finalizou o acesso ao *site.* Com a chave exclusiva do modelo chave-valor é possível armazenar o identificador único referente a sessão do usuário, podendo ser um dado do tipo numérico. Sessões W*eb* podem armazenar informações referentes ao tempo de acesso do usuário, mensagens, dados do perfil entre outros dados que serão armazenados no campo valor da chave referente à sessão.

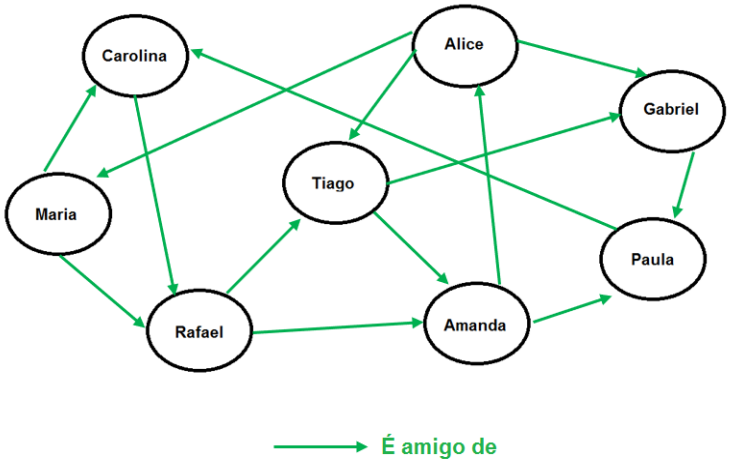
O modelo chave-valor é utilizado geralmente para leitura frequente de informações, com grande desempenho em consultas buscando a chave exclusiva. Uma desvantagem segundo Sadalage e Fowler (2013) é que os dados do campo valor não podem ser indexados, ficando a aplicação responsável por identificar os dados armazenados no campo valor. Logo a indexação pela chave é útil para buscas mais complexas entre os valores. São exemplos de banco de dados orientados a chave-valor o Riak (RIAK, 2019) e o Redis (REDIS, 2019).

2.3 MODELO ORIENTADO A GRAFOS

O modelo orientado a grafos é representado por meio da Teoria de Grafos (PENTEADO, 2014). Segundo Lóscio *et al*. (2011) sua estrutura é composta de 3 componentes: os nós (vértices do grafo), os relacionamentos (arestas) e as propriedades (atributos) dos nós. Seu principal objetivo é a criação de um modelo de dados mais heterogêneo comparado ao modelo relacional, com dados desestruturados.

Na Figura 2 pode-se observar como são estruturados os dados no modelo. Os círculos são os nós (entidades), e as setas representam a direção dos relacionamentos. A forma como os dados são apresentados são completamente diferentes de outros modelos, que armazenam os dados no formato de tabelas ou documentos. A representação dos dados na Figura 2 é chamada de simples-relacional, a forma mais simples de representação dos dados no modelo de grafos, onde todos os vértices são do mesmo tipo e as arestas são definidas com o mesmo tipo de relacionamento (PRIGOL, 2016).

Figura 2 – Representação do Modelo Grafos



Fonte: Autoria própria

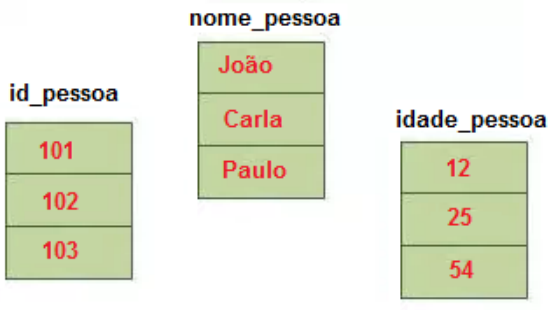
O modelo de grafos é bastante utilizado em redes sociais. Nesse ambiente o usuário com seu perfil cria uma rede de amigos, gerando diversos relacionamentos com outras entidades. O modelo de grafos cria essa rede de amigos em um grafo. A maturidade do conceito de grafos permite buscas complexas nesta estrutura. São

exemplos de bancos de dados orientado a grafos o Neo4j (NEO4J, 2019) e o OrientDB (ORIENTDB, 2019).

2.4 MODELO ORIENTADO A FAMILIA DE COLUNAS

Neste modelo os dados são armazenados em linhas dentro de uma coluna (tabela), diferente dos bancos de dados relacionais que armazenam os dados em uma única linha. O modelo orientado a família de colunas possui um identificador exclusivo em cada tabela, mais especificamente uma chave primária. Cada linha dessa tabela é um atributo, podendo haver mais de uma linha e relação entre as tabelas. Na Figura 3 ilustra-se a representação dos dados no modelo família de colunas.

Figura 3 – Representação do Modelo Família de Colunas



Fonte: Autoria própria

No banco de orientado a colunas, os dados são indexados por uma tripla formada por linha, coluna e *timestamp* (LÓSCIO *et al*.,2011). Linhas e colunas são identificadas por chaves. O *timestamp* é o identificador do tipo de dado. O conceito família de colunas é utilizado para o agrupamento de colunas do mesmo tipo.

A vantagem do modelo é a compressão de dados. Cada tabela contém o mesmo tipo de dado em suas linhas. Este modelo torna-se útil em arquiteturas contendo processadores multinúcleos (múltiplos núcleo), particionando o processamento dos dados. São exemplos de bancos de dados orientados a família de colunas o Cassandra (CASSANDRA, 2019) e o HBase (HBASE, 2019).

2.5 MODELO ORIENTADO A DOCUMENTOS

O modelo orientado a documentos é projetado para armazenar os dados encapsulados em pares de chave-valor no formato JSON (*JavaScript Object Notation*) ou em outro padrão semelhante (WANZELLER, 2013). Segundo Sadalage e Fowler (2013) o documento é o conceito principal do modelo. Assim como características dos modelos não relacionais o documento em si não possui esquema pré-definido. A ausência de esquemas permite o armazenamento de dados semi-estruturados nos documentos. Segundo Ronaldo *et al*. (2000) dados semi-estruturados apresentam uma representação estrutural heterogênea. Essa estrutura é caracterizada por dados irregulares fora de um padrão estrutural. Os dados semi-estruturados apresentam uma estrutura fortemente evolutiva, podem ser modificados constantemente.

O armazenamento JSON traz uma vantagem adicional que é o suporte a tipos de dados, o que torna a forma de armazenamento mais amigável para os desenvolvedores (BRITO, 2010). A estrutura dos documentos JSON segue o modelo chave-valor (JSON, 2019). Cada objeto do documento é delimitado por uma chave { } no início e no fim respectivamente. O delimitador entre os pares chave-valor é representado pelo símbolo “:” (dois pontos). Os pares chave-valor são separados por vírgula. No documento uma chave precisa ser única. Pode-se criar um campo _id dentro de cada objeto tornando-o exclusivo. A Figura 4 ilustra a representação dos dados em um documento JSON.

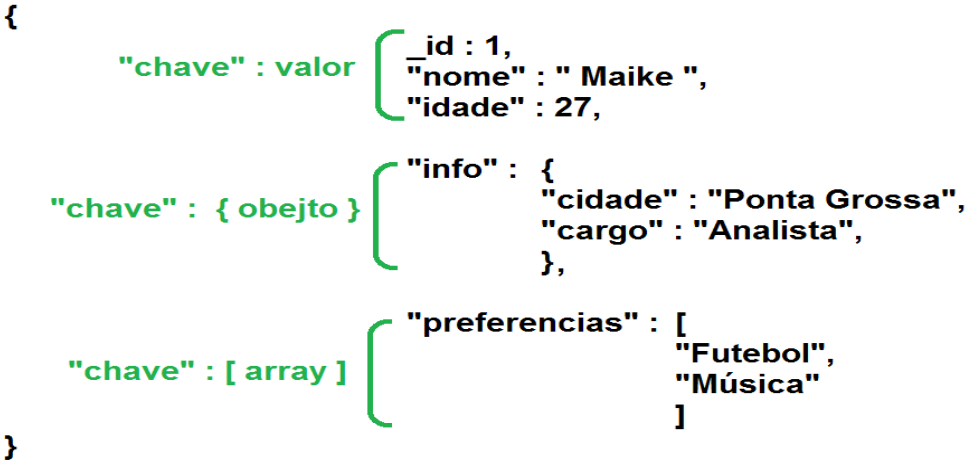
Figura 4 – Representação do Modelo Orientado a Documentos

Fonte: Autoria própria

Nos pares chave-valor, o campo valor permite o armazenamento de diferentes tipos de dados: texto, data, coordenadas geométricas, números, entre outros. Pode-se

armazenar estruturas de dados do tipo objeto ou *array* (matriz) no campo valor, conforme ilustra a Figura 5. São exemplos de bancos de dados orientados a documentos o MongoDB (MONGODB, 2019) e o CouchDB (COUCHDB, 2019).

Figura 5 – Formato Documento JSON



Fonte: Autoria própria

2.6 CONSIDERAÇÕES

Os SGBDRs são amplamente utilizados atualmente, sendo padrão de mercado. Sua principal vantagem está na garantia de integridade dos dados, tal característica é assegurada pelas propriedades ACID. Porém, mantê-las diante de um elevado volume de dados, tornou-se uma tarefa onerosa. O processamento desse volume de dados em tempo hábil considerando sistemas relacionais, revelou-se uma limitação. Tal problema impulsionou novos estudos na área, surgindo assim os bancos de dados *NoSQL,* também chamados de *não-convencionais*. Essa classe de sistemas pode renunciar das propriedades ACID e de esquemas pré-definidos em prol do desempenho. Um dos tipos de banco de dados *NoSQL* são os orientados a documentos, detalhados no Capitulo 3.

3 SGBD ORIENTADO A DOCUMENTOS

Uma característica marcante do modelo orientado a documentos é a flexibilidade do esquema na criação de um documento. Tal característica foi essencial na criação de uma poderosa ferramenta de monitoramento para a cidade de Chicago (CHICAGO, 2019). Dados de diversos setores como transporte, polícia e bombeiros, entre outros são armazenados em um único banco de dados para serem processados em tempo real. Ao cruzar os dados obtidos e fazendo uso de ferramentas de geoprocessamento, foi possível melhorar o policiamento de regiões com altos índices de criminalidade. Dados oriundos de redes sociais tornaram possível prever e evitar incidentes. O mapeamento da utilização de energia elétrica permitiu a tomada de medidas preventivas, como a instalação de novas linhas de rede elétrica. Enfim, tal ferramenta contribuiu para melhoria da qualidade de vida dos habitantes além de tornar Chicago uma cidade integrada.

Outra característica do modelo orientado a documentos é a escabilidade. Pois permitiu a *EA GAMES* (*Electronic Arts Inc*), empresa desenvolvedora e distribuidora de jogos eletrônicos o escalonamento do jogo FIFA *Online 3,* para milhares de usuários *online* em tempo real (EA GAMES, 2019). A *EA GAMES* processa seu jogo espalhado em cerca de 250 servidores que executam um SGBD orientado a documentos resultando em uma solução inovadora e de alto desempenho.

Atualmente existem diferentes bancos de dados nativos orientados a documentos. Esses SGBDs são elencados conforme *ranking* da *DB-Engines* (DB-ENGINES, 2019). Em seu *ranking*, o *site DB-Engines* calcula a pontuação dos SGBDs considerando dados como: popularidade, interesse geral no sistema, número de ofertas de empregos, relevância nas redes sociais, entre outros. A Figura 6 ilustra o *ranking DB-Engines* para SGBDs que tomam por base o modelo orientado a documentos.

Figura 6 – Ranking de popularidade dos SGBD's.

Rank			DBMS	Database Model	Score		
Sep 2019	Aug 2019	Sep 2018			Sep 2019	Aug 2019	Sep 2018
1.	1.	1.	MongoDB	Document	410.06	+5.50	+51.27
2.	2.	2.	Amazon DynamoDB	Multi-model	57.82	+1.25	+4.47
3.	3.	3.	Couchbase	Document, Multi-model	31.29	-2.54	-3.26
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model	30.87	+0.93	+11.69
5.	5.	5.	CouchDB	Document	18.40	-1.37	-0.25
6.	6.	6.	MarkLogic	Multi-model	13.42	-1.04	+1.89
7.	7.	7.	Firebase Realtime Database	Document	11.25	-0.39	+3.14
8.	8.	8.	OrientDB	Multi-model	5.06	-1.22	-0.41
9.	9.	↑ 16.	Google Cloud Firestore	Document	4.94	-1.32	+2.52
10.	10.	↑ 11.	Google Cloud Datastore	Document	4.72	-1.18	+0.92

Fonte: DB-ENGINES RANKING (2019).

Conforme a Figura 6, é possível observar que o MongoDB é o primeiro colocado no *ranking,* sendo o SGBD mais popular dentre os que suportam o modelo de armazenamento orientado a documentos. Dentre os 4 primeiros colocados, o MongoDB é o único SGBD nativo, cuja estrutura de armazenamento foi desenvolvida especificamente para tratar documentos. Devido a essas duas importantes características, o MongoDB foi escolhido para realização dos experimentos neste trabalho.

3.1 MONGODB

O MongoDB é um SGBD de código-fonte aberto, desenvolvido em linguagem C++. Diferente do modelo relacional, o MongoDB armazena dados em *collection* (coleções), equivalentes às relações do modelo relacional fazendo uso do formato BSON (*Binary* JSON), uma extensão do formato JSON (WANZELLER, 2013). Uma *collection* pode ter um ou mais documentos que são equivalentes as tuplas no modelo relacional (WANZELLER, 2013). Segundo Oliveira (2017), o modelo orientado a documentos não depende de um esquema rígido, não demanda estruturação fixa dos dados como acontece nos bancos relacionais.

Para garantir velocidade nas transações os modelos não relacionais abdicaram das propriedades ACID. O MongoDB não obedece a todas as propriedades ACID nas transações. A exceção é a operação de gravação de dados no documento que é atômica.

Devido aos modelos não relacionais não adotarem as regras de integridade referencial com intuito de obter-se flexibilidade na modelagem dos dados, foi possível a concepção de um esquema livre, sem regras. Em contrapartida, determinadas aplicações podem exigir regras específicas para determinados valores dentro do documento. Consequentemente necessita-se realizar validações nos valores antes de armazená-los. Para atender essa demanda, o MongoDB, a partir da versão 3.2 oferece recursos de validação do esquema, que permitem um controle dos dados armazenados no banco (DOCUMENTAÇÃO MONGODB, 2019).

Para criação de regras internas no esquema, validações são extremamente úteis. Por exemplo, no momento que se cria uma *collection*, é possível definir que um determinado campo só aceitará os seguintes valores de entrada: “casa” ou “apartamento”. Caso o usuário insira dados nessa *collection* onde o valor do campo possua como entrada “condomínio”, o SGBD não validará a transação. O MongoDB possui outras validações como: lista de valores permitidos, valores máximos e mínimos, tipo de dado, obrigatoriedade de preenchimento, entre outros. As validações podem ocorrer durante as transações de atualização e inserção de dados. As que contêm erros ou *warnings* (alertas) são registradas no *log* do MongoDB.

3.1.1 Estrutura dos Documentos

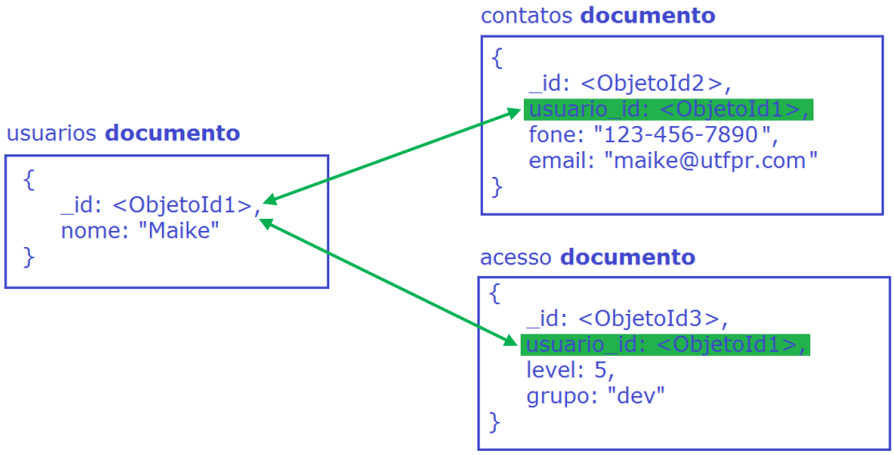
Segundo Chodorow (2013), o modelo orientado a documentos substitui o conceito de “linha” oriundo do modelo relacional, por um modelo mais flexível, o “documento”. Cada documento tem um campo chamado **_id,** utilizado como chave primária. Esse campo possui um valor exclusivo, pois identifica o objeto, cujo conteúdo é limitado pelas chaves “{“ e “}”.

O MongoDB permite relacionar objetos a partir do campo exclusivo **_id** em diferentes documentos, semelhante ao relacionamento de tabelas no banco de dados relacional, onde utiliza-se chave estrangeira. Existem dois tipos de relacionamentos, por referência ou referência embutida no documento.

O relacionamento por referência assemelha-se a definição de chave estrangeira. A Figura 7 ilustra três documentos, onde **contatos e acessos** se referenciam a **usuários** por meio do objeto **_id**. Essa referência não é um recurso do MongoDB, a flexibilidade do modelo permite tal modelagem. Porém, a desvantagem do

relacionamento por referência é observada em consultas, pois deve-se ler todos os documentos e traduzir as referências para obter o resultado, tornando a tarefa mais onerosa.

Figura 7 – Relacionamento por Referência MongoDB



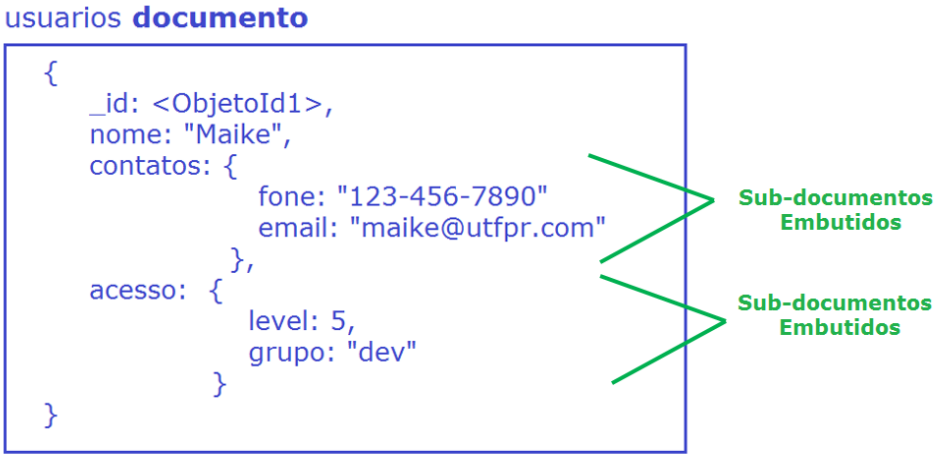
Fonte: Autoria própria

O relacionamento por referência embora útil é pouco aplicado, pois o tempo de leitura dos documentos não justifica ramificar os relacionamentos. Quando utilizado, esse relacionamento pode gerar inconsistências na operação de gravação, devido a leitura e gravação de vários documentos ao mesmo tempo.

A documentação do MongoDB recomenda unir todos os relacionamentos em apenas um único documento. Na Figura 8 pode-se observar que os relacionamentos dos documentos **contatos** e **acesso** estão dispostos no documento **usuários**. Armazena-se dados redundantes em um único documento, com o intuito de otimizar o armazenamento e a consulta, conforme descreve (OLIVEIRA, 2017).

O MongoDB permite armazenar os relacionamentos por referências em sub-documentos ou *arrays* de documentos, descreve (WANZELLER, 2013). A Figura 8 ilustra o relacionamento por meio de sub-documentos. Observa-se que o campo **usuario_id** presente na Figura 7 não precisa ser reescrito na Figura 8, pois os valores já estão contidos na chave dos campos: **contatos e acesso**.

Figura 8 – Referencias Embutidas MongoDB



Fonte: Autoria própria

3.1.2 Linguagem de Consulta

O MongoDB possui uma linguagem de consulta exclusiva, que difere da sintaxe da SQL (*Structured Query Language*), utilizada nos SGBDRs. A utilização das chaves "{" e "}" entre o comando torna intuitiva a consulta, pois assemelha-se a estrutura de um arquivo JSON. A Figura 9 ilustra a diferença entre uma consulta no MongoDB e uma equivalente em SQL.

Figura 9 – Comparativo de consultas SQL x MongoDB

```
db. inventario . find ( { status : "A" , qty : { $lt : 30 } } )
```

A operação corresponde à seguinte instrução SQL:

```
SELECT * FROM inventario WHERE status = "A" AND qty < 30
```

Fonte: Autoria própria

Conforme a Figura 9, o texto em cor verde representa os comandos reservados das linguagens. Em amarelo estão os valores a serem investigados. A palavra "**inventario**" representa uma *collection* no MongoDB, em SQL por sua vez se refere a uma relação. A consulta em questão filtra duas chaves, **status** e **qty.** No MongoDB as cláusulas devem sempre estar dentro de chaves "{" e "}", separados pelos operadores

lógicos. Na Figura 9, a consulta em questão possui o operador logico “**AND**” que é representado pela virgula “,” no MongoDB.

Outra característica do MongoDB refere-se aos operadores relacionais que possuem diferentes sintaxes em relação a SQL. Quando utilizados, devem estar sempre entre chaves “{“ e “}”. Na Figura 9, o operador “**<**” (menor que), é representado por **$lt** no MongoDB. Observa-se que o mesmo está entre as chaves **{ $lt:30 }**, evidenciando uma nova operação. O Quadro 1 representa os demais operadores relacionais e seus equivalentes em SQL, conforme documentação do MongoDB (DOCUMENTAÇÃO MONGODB, 2019).

Quadro 1 – Equivalência de Operadores Relacionais entre o MongoDB e a SQL

Operadores	Descrição	Equivalência SQL
$eq	Valor igual a	**=**
$gt	Valor maior que	**>**
$gte	Valor maior que ou igual a	**>=**
$lt	Valor menor que	**<**
$lte	Valor menor que ou igual a	**<=**
$ne	Valor diferente de	**<>**
$in	Procura valor em uma matriz	**in**
$nin	Nenhum valor específico em uma matriz	**not in**

Fonte: Autoria própria

3.1.3 Processamento de Consultas

Para o eficiente processamento de consultas é exigido dos SGBDs mecanismos extremamente complexos. Os mesmos são responsáveis pelo planejamento, execução da consulta sobre a base de dados e apresentação dos resultados aos usuários (BINI, 2014).

Para cada consulta submetida ao MongoDB, é gerado um ou mais planos de execução. Cada plano descreve as etapas, para que a consulta seja executada e os resultados obtidos. Para a elaboração de boas estratégias de otimização, necessita-se de uma análise minuciosa dos planos gerados. Essas estratégias têm com finalidade, minimizar o tempo de execução e reduzir a utilização de recursos computacionais empregados na execução da consulta. Conhecer a estrutura dos planos de execução

torna-se premissa essencial para realizar ajustes buscando a melhoria de desempenho do SGBD.

O MongoDB possui um planejador lógico de consultas, responsável por gerenciar os planos de execução. Seu objetivo é encontrar uma solução eficiente para a consulta, no que diz respeito à escolha do plano que leva o menor tempo para executá-la. Dessa forma, quando uma consulta é submetida ao MongoDB, o planejador primeiramente verifica se o seu plano de execução não se encontra em *cache*, como resultado da submissão anterior da mesma consulta. Caso não encontre, planos de execução candidatos serão gerados e submetidos a avaliação de desempenho. Na etapa de avaliação de desempenho, o plano que apresentar o menor tempo de execução será escolhido, carregado em *cache* e os demais descartados.

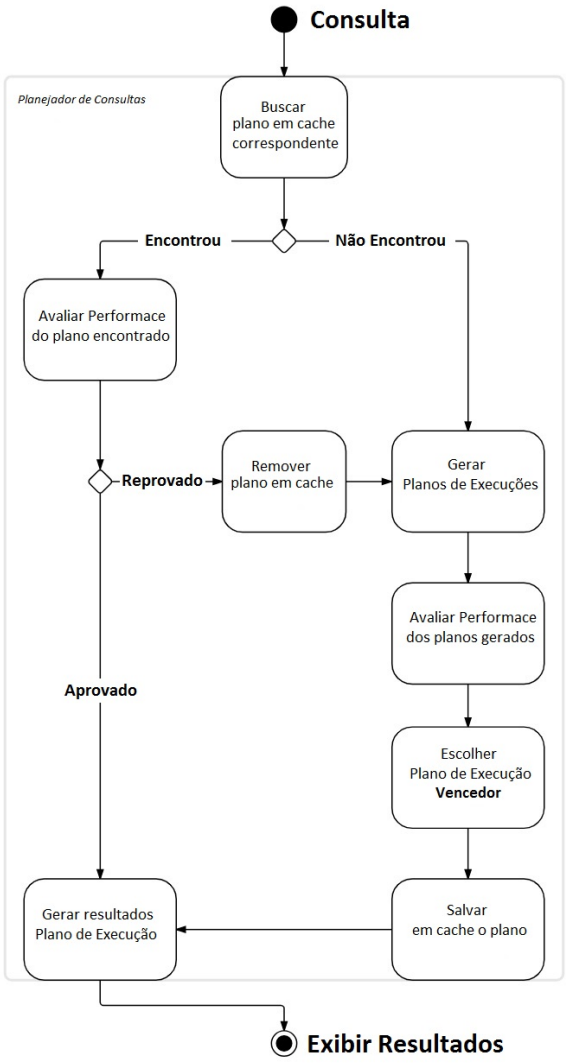
Quando uma consulta é submetida ao SGBD, e a mesma já possui um plano armazenado em *cache*, este será reavaliado por meio de um mecanismo chamado *replanning* (replanejamento). Caso o plano seja considerado eficiente, o mesmo segue para a etapa de execução e é mantido em *cache*. Caso contrário, o plano será excluído do *cache* e novos planos de execução candidatos serão gerados para a consulta conforme já descrito. A Figura 10 representa o fluxo do planejador de consultas do MongoDB.

Para se obter detalhes do plano de execução de uma consulta, utiliza-se o comando *Explain***.** Dados relevantes contidos no plano serão apresentados como: número de documentos retornados; número de documentos lidos; índices utilizados; tempo necessário para execução de cada estágio; tempo necessário para executar cada estágio da consulta e os planos reprovados. O comando *Explain* aceita três parâmetros a saber:

- **queryPlanner:** Realiza avaliações de desempenho e retorna o plano de execução mais eficiente, considerando o menor tempo de execução. Utilizando esse parâmetro, a consulta não é executada, apenas avaliada.
- **executionStats:** Realiza avaliações de desempenho e retorna o plano de execução mais eficiente, considerando o menor tempo de execução sendo que a consulta é realmente executada sobre o banco de dados. Os planos reprovados não são exibidos usando este parâmetro.
- **allPlansExecution:** Realiza avaliações de desempenho e retorna o plano de execução mais eficiente, considerando o menor tempo de execução sendo que a consulta é realmente executada sobre o banco de dados. Todos os planos de

execução são retornados inclusive os planos reprovados por desempenho, caso existam.

Figura 10 – Fluxograma do Planejador Lógico de Consultas do MongoDB



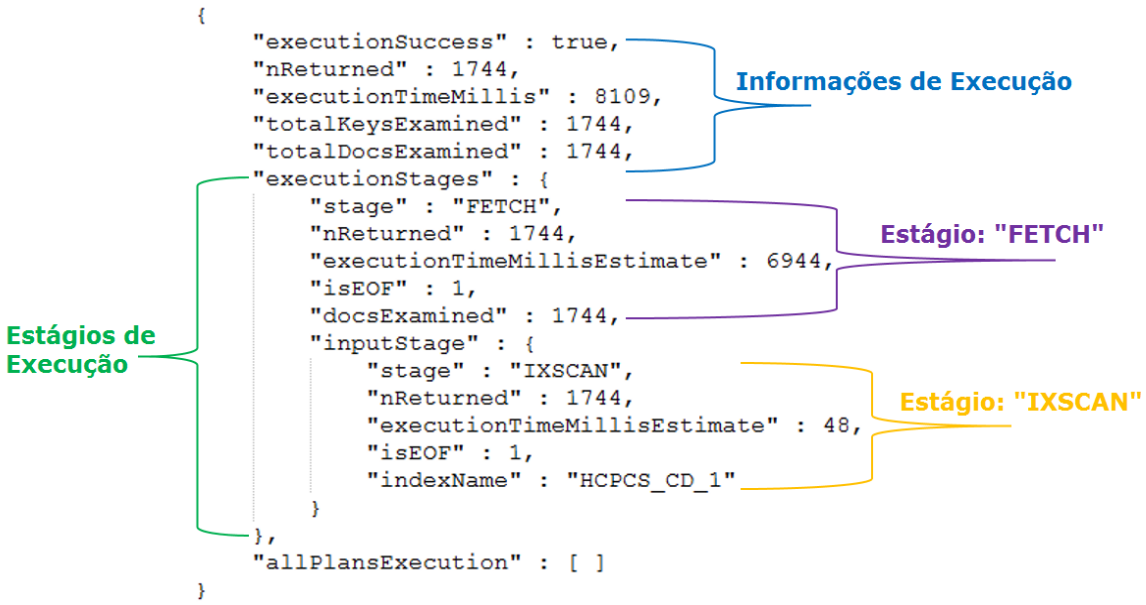
Fonte: Autoria própria

Por padrão o comando *Explain* utiliza o parâmetro *queryPlanner*. Porém para se obter informações úteis aos ajustes de desempenho propostos neste trabalho, utilizou-se o parâmetro *executionStats*. Tais informações são detalhadas:

- **nReturned:** Número de objetos retornados;
- **executionTimeMillis:** Tempo total em milissegundos para a escolha do plano mais eficiente, assim como sua execução até a exibição dos resultados;

- **totalKeysExamined:** Número de chave de índices verificadas;
- **totalDocsExamined:** Número de objetos examinados;
- **executionStages:** Apresenta as etapas de execução do plano mais eficiente. Podem haver diversas etapas de execução, conforme ilustra a Figura 11.

Figura 11 – Exemplo de um plano de execução de consulta



Fonte: Documentação MongoDB (DOCUMENTAÇÃO MONGODB, 2019)

Informações detalhadas dos estágios da execução da consulta também estão presentes na Figura 11 e estão contidas no objeto *inputStage*. Cada estágio descreve o tipo de operação que será processada pelo SGBD para execução da consulta. Os estágios presentes nos planos de execução deste trabalho são:

- **COLLSCAN:** Realiza a varredura completa na *collection*. Todos os dados dos documentos são lidos;
- **IXSCAN:** Realiza a leitura das chaves dos índices na memória principal;
- **FETCH:** Realiza a recuperação de um documento dentro da *collection*;
- **TEXT:** Realiza comparações do tipo texto.

Pode-se observar na Figura 11, que o plano em questão contém dois estágios de execução, um *FETCH* e outro *IXSCAN*. Destaca-se algumas informações especificas de cada estágio, conforme documentação do MongoDB (DOCUMENTAÇÃO MONGODB, 2019):

- **stage**: Especifica o nome do estágio de execução;
- **isEOF:** Especifica o estágio de execução, se o mesmo atingiu o fim do fluxo. Se *true* ou 1, o estágio de execução atingiu o final do fluxo. Se for *false* ou 0, o estágio ainda poderá ter resultados para retornar;
- **keysExamined:** Para estágios de execução de consulta que varrem um índice (por exemplo, *IXSCAN*), é o número total de chaves de índices examinadas no estágio;
- **docsExamined:** Especifica o número de objetos verificados no estágio de execução;
- **indexName:** Indica o campo com índice utilizado para processar a consulta.

3.2 CONSIDERAÇÕES

O MongoDB é o SGBD mais utilizado dentre os bancos de dados nativos orientados a documentos. Suas características permitem sua utilização em diversas aplicações. A flexibilidade do esquema de dados é um atrativo marcante desse SGBD, justificando sua utilização. Tal modelagem contribui para o armazenamento da base de dados em um único documento, tornando a execução de consultas mais ágil.

Prover o processamento rápido e eficiente de grandes volumes de dados é um dos propósitos do SGBD MongoDB. Determinadas aplicações demandam respostas rápidas às solicitações. Para atender a essas exigências, podem ser aplicadas ao SGBD técnicas de *tuning*, assunto abordado no Capítulo 4 deste documento.

4 TUNING NO SGDB

Na busca por melhoria de desempenho em SGBDs, é comum o emprego de técnicas de *tuning*. Segundo Dominico (2013) e Tramontina (2008) a prática do *tuning* em SGBDs é definida como o ajuste fino em seus parâmetros de configuração, objetivando maximizar seu desempenho, reduzindo o tempo de resposta às solicitações. Amplamente utilizada em SGBDRs, também pode ser aplicada em bases de dados não-convencionais.

O emprego de técnicas de *tuning* requer grande conhecimento por parte do DBA*,* gera alto custo às organizações devido ao tempo empregado para testar sua eficiência. Tais técnicas, segundo Shasha e Bonnet (2002) podem envolver mudanças nas configurações de *hardware* do hospedeiro, alterações em parâmetros do sistema operacional, como a quantidade de memória compartilhada utilizada pelo SGBD, além de configurações nos seus próprios parâmetros de configuração.

Para facilitar o estudo e compreensão dos diversos parâmetros empregados na prática de *tuning,* Soror *et al*. (2008) e Soror *et al*. (2007) os classificam em dois tipos: *prescriptive parameters***:** parâmetros que afetam diretamente o sistema de banco de dados, como a memória compartilhada pelo SGBD. D*escriptive parameters***:** parâmetros que afetam indiretamente o SGBD, como os relacionados ao sistema operacional.

Na realização deste trabalho utilizou-se uma estrutura interna de dados disponível no MongoDB, os ***indexs*** (índices), classificado como *prescriptive parameters*. Como parâmetro externo ao SGBD, foi utilizado o ***THP (Transparent Huge Pages)***, classificado como *descriptive parameters.* Tal parâmetro é ajustável e disponível no sistema operacional *Linux*. Tanto a escolha de índices quanto do parâmetro THP para realização dos experimentos são indicações da documentação do MongoDB (DOCUMENTAÇÃO MONGODB, 2019).

4.1 INDEX

Quando submetida uma consulta ao MongoDB, pode-se apenas projetar um campo específico do documento sendo os demais descartados. Apesar dessa restrição em um único campo, o SGBD por padrão irá percorrer todas as instâncias, para retornar

à correspondência solicitada. Essa ação não é eficiente, pois exige acesso constante ao disco rígido para a leitura dos dados. Como é de conhecimento segundo Korth *et al*. (2006), esta operação gera queda no desempenho das aplicações. Uma técnica eficiente para contornar o problema é catalogar as informações do banco de dados em chaves. Desta forma, a busca é efetuada pelas chaves, para recuperar os dados correspondentes em disco. O resultado é um único acesso ao disco rígido. Essa técnica é chamada de *index*. Segundo Korth *et al*. (2006) um *index* em banco de dados funciona como índice de um livro. O sistema pesquisa no *index* e traz o bloco de disco contendo as informações. Os *index* são estruturas ágeis, pois são armazenados em memória principal RAM (*Random Access Memory*), tornando a pesquisa rápida e eficiente.

Os *indexs* no MongoDB são armazenados utilizando a estrutura de dados Árvore B (*B-Tree*)[1]. Essa técnica foi desenvolvida em 1971, utilizando o conceito de estrutura em árvore (BAYER; MCCREIGHT, 1971). É amplamente aplicada em sistemas de banco de dados, sendo projetada inicialmente para ser manipulada em memória secundaria (disco rígido). Porém, atualmente utiliza-se também em memória principal (RAM). A principal vantagem da utilização de estruturas Árvore B é manter as chaves classificadas em ordem crescente ou decrescente. Segundo Lightstone *et al.* (2007) essa classificação torna a consulta mais ágil, por isso é a estrutura mais utilizada para melhoria de desempenho em consultas.

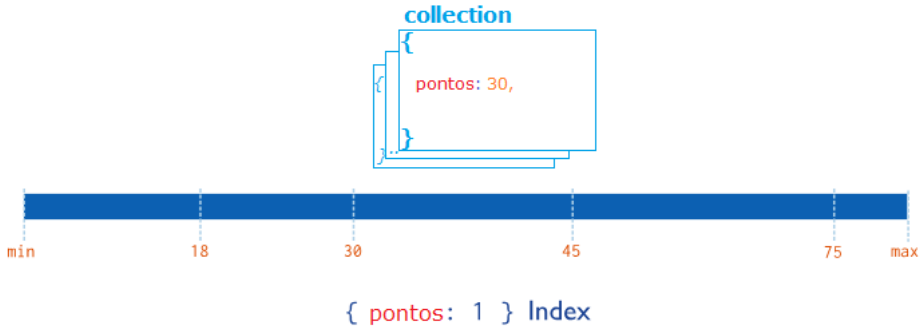
No MongoDB os *indexs* são definidos na *collection*, sendo suportados em qualquer campo ou subcampo. Quando se cria uma *collection* é gerado automaticamente um campo **_id** para cada novo documento. Tal índice **_id** identifica os documentos como exclusivo**,** não permitindo duplicidade. O campo **_id** se assemelha ao conceito de chave primária dos SGBDRs. O MongoDB suporta os seguintes tipos de índices utilizados neste trabalho:

- **Campo Único**: Além do índice **_id,** o MongoDB suporta a criação de índice para um campo único de uma *collection*, ordenados de forma crescente e decrescente**.** A Figura 12 representa um índice no campo único denominado **pontos**. Utiliza-se o parâmetro (1) para indicar a ordenação crescente ou parâmetro (-1) indicando ordenação decrescente. O comando para criação desse tipo de índice é:

[1] Arvore *B-Trees*: estrutura de dados no formato de árvore. Utilizada para armazenar chaves, equivalentes aos *indexs*.

db.<*collection*>.createIndex({ <*campo*>: < ordem classificação (1 ou -1)> }).

Figura 12 – Representação Index - Campo Único

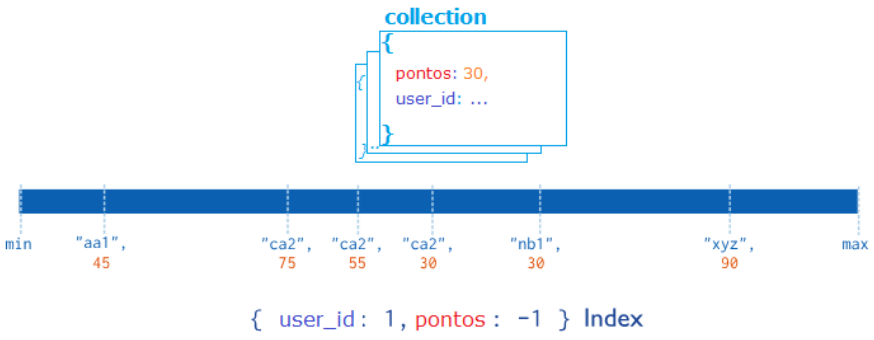


Fonte: Adaptado de (DOCUMENTAÇÃO MONGODB, 2019)

- **Índice Composto:** O MongoDB suporta a criação de índices compostos, utilizando dois campos de uma *collection*. Pode-se ordenar de forma crescente ou decrescente alternando os campos. A Figura 13 ilustra um índice composto, no campo ***userid*** com ordem de classificação crescente. Outro índice no campo ***pontos*** com ordem de classificação decrescente. O comando para criação desse tipo de índice é:

db.<*collection*>.createIndex({ <*campo1*>: <ordem classificação (1 ou -1)>, <campo2> : <ordem classificação (1 ou -1)> }).

Figura 13 – Representação Index Composto



Fonte: Adaptado de (DOCUMENTAÇÃO MONGODB, 2019)

- **Índice Texto**: O índice texto suporta a pesquisa de conteúdo em campos do tipo ***string*** em uma *collection*. A consulta é realizada na cadeia de caracteres. Esse tipo de *index* permite a criação de índices composto, ou seja, um índice em dois campos do tipo ***string***. A Figura 14 ilustra o comando de criação para esse tipo de *index*.

Figura 14 – Comando para criação de um Index Text

```
db.<collection>.createIndex(
{
  <campo1> : "text"
  <campo2> : "text"
 }
)
```

Fonte: Autoria própria

4.2 THP (*TRANSPARENT HUGE PAGES*)

O *THP* é um gerenciador de páginas de memória, disponível nos sistemas operacionais *Linux* (UBUNTU, 2019). É uma camada de abstração, que gerencia e automatiza as *huge pages* (páginas grandes). A memória principal possui o armazenamento baseado em blocos de 4.096 *bytes* chamados de *pages* (páginas). As *pages* podem sofrer alterações no tamanho, com limite máximo de 1 *gigabytes* (GB). Quando aumentado o tamanho dessas *pages* além do padrão, elas se tornam *huge pages* (CORBET, 2010). A alteração ocorre devido ao Sistema Operacional (SO) reservar uma porção maior de memória para um determinado aplicativo, com a finalidade de obter melhor desempenho. Quem gerencia o tamanho das *pages* que serão disponibilizadas para cada aplicativo é o THP (TRANSPARENT HUGE PAGES, 2019). As *huge pages* são sempre alocadas em memória física não entrando na lista de blocos candidatos a *swap*[2].

Segundo Gorman (2004) no gerenciamento da memória virtual, o SO gera endereços lógicos que são traduzidos para endereços físicos e estão contidos em uma tabela de mapeamento de endereços. Para cada transação sobre uma página, o sistema necessita carregar a tabela de mapeamento. Caso o aplicativo possua *pages* de tamanho pequeno, é necessária a tradução de endereços de cada página. Essa

[2] *Swap*: é uma memória virtual do Linux. Uma memória emergencial, caso a memória principal venha a se esgotar.

solicitação gera uma perda de desempenho (*overhead*) em relação as *pages* maiores, que necessitam de menos traduções de endereços.

O TLB (*Translation Lookaside Buffer*) é um *hardware* capaz de minimizar a tradução de endereços físico e lógicos na memória, evitando uma consulta excessiva à tabela de páginas na memória principal (GORMAN, 2004). A necessidade de uma página verificar a tabela de mapeamento de endereços acarreta em perda de desempenho, pois a cada solicitação gera-se o acesso à memória principal, justificando assim, a utilização do TLB. Quando necessário o acesso à tabela de mapeamento, a CPU solicita ao TLB o número da página e o mesmo realiza uma busca em uma tabela interna, armazenada em *cache*. Essa tabela agrega informações das *pages*, consequentemente evita-se o acesso a memória principal.

O THP atua minimizando a sobrecarga de pesquisas do TLB. Ele gerencia as *pages* conforme o tamanho da memória. Em máquinas com grande quantidade de memória, o THP usa *huge pages* que realizam menos consultas na tabela de mapeamento de endereços na memória principal.

O conceito de *huge page*s é frequentemente utilizado em banco de dados relacionais com a implementação do THP. Em banco de dados não relacionais devido as altas cargas de trabalhos (leituras/escritas) o THP pode obter um pior desempenho. Essa piora se dá ao fato dos padrões de acesso a memória principal serem aleatórios em vez de sequenciais.

4.3 CONSIDERAÇÔES

As técnicas de *tuning* podem ser aplicadas tanto em sistemas relacionais quanto em sistemas não convencionais. Tais técnicas envolvem alterações em parâmetros que podem afetar diretamente ou indiretamente o SGBD, conforme detalhado. Para analisar a eficiência da aplicação de técnicas de *tuning* no SGBD MongoDB foi necessário a criação de um ambiente experimental para realização de testes de desempenho. Esse ambiente assim como os resultados obtidos, serão apresentados no Capítulo 5 deste trabalho.

5 AMBIENTE EXPERIMENTAL E RESULTADOS

Para realização dos experimentos, utilizou-se um computador dotado de um processador Intel Core I7-5500U de 3.00 GHz (*gigahertz*), com 4 MB (*megabytes)* de memória *cache*. O processador possuía dois núcleos físicos e quatro núcleos simulados. E um total de 16 GB (*gigabytes)* de memória RAM. A memória secundária era composta por um disco rígido SATA (*Serial Advanced Technology Attachment*) de 1 TB (*terabytes)* de 7200 RPM (Rotação por Minuto). O SO utilizado foi GNU/Linux, distribuição Ubuntu 18.04.3 LTS (*Long Term Support*) *Bionic Beaver* com *kernel* 4.18.0-18x86_64 (UBUNTU, 2019). O SGBD empregado foi o MongoDB versão 4.0.9 (versão estável) (MONGODB, 2019).

Para garantir a confiabilidade dos resultados, elaborou-se um *checklist* inicial antes da execução de cada experimento: Cada consulta foi submetida ao SGBD por 3 vezes e logo após extraída a média dos tempos de execução, sendo este valor ponderado na escala de milissegundos (ms). Após cada submissão de consulta o serviço do SGBD MongoDB era interrompido. Seguia-se a limpeza da *cache* do SO e retomava-se a execução do SGBD. Iniciada a execução do cliente MongoDB, foi realizada a limpeza do *cache* interno utilizado pelo planejador de consultas. Durante a submissão das consultas não haviam processos concorrentes executando no ambiente computacional. Os comandos empregados e suas respectivas funcionalidades foram:

1. **sudo service mongod stop:** Interrompe o serviço do MongoDB;
2. **sync; echo 3 > /procs/sys/vm/drop_caches**: Esvazia a memória *cache* do SO;
3. **sudo service mongod start:** Inicia o serviço do MongoDB;
4. **db.collection.getPlanCache().clear()**: Limpa os planos de execução salvos em cache.

Para as consultas que utilizavam estruturas *indexs*, elaborou-se um *checklist* diferenciado garantindo a confiabilidade dos resultados. Além da limpeza da memória *cache* do SO, eram removidas as entradas de *indexs* das bases de dados. Com esse procedimento garantia-se o esvaziamento da memória principal. Para isso os seguintes comandos foram necessários:

1. **db.<collection>.dropIndexes()**: Remove todos os índice da *collection*;
2. **sudo service mongod stop:** Interrompe o serviço do MongoDB;
3. **sudo service mongod start:** Inicia o serviço do MongoDB;

4. **db.collection.getPlanCache().clear():** Limpa os planos de execução das consultas.

5.1 BASE DE DADOS

Para realização dos experimentos foi necessária a utilização de duas bases de dados, com a finalidade de avaliar a efetividade das técnicas de *tuning* empregadas. As bases de dados foram geradas por meio da importação de documentos do tipo JSON, CSV (*Comma-Separated Values*) ou TSV (*Tab Separated Values*) utilizando o comando **mongoimport**.

A Figura 15 ilustra o comando **mongoimport**. Onde "--db" é o parâmetro que indica o nome da base de dados utilizada, neste caso *dados_medicos*. O parâmetro "--*collection*" nomeia a *collection*, chamada *medicina.* O parâmetro *"--type*" indica a origem do tipo de arquivo importado, neste exemplo o arquivo possui a extensão CSV. Para especificar o nome dos campos no arquivo CSV utiliza-se sua primeira linha, através do parâmetro *"--headerline"*. Por fim, o parâmetro *"--file"* indica o caminho do arquivo a ser importado.

Figura 15 – Exemplo de aplicação do comando "mongoimport"

```
mongoimport --db "dados_medicos" --collection "medicina" --type CSV --headerline --file "home/medicina.csv"
```

Fonte: Autoria própria

Atualmente a Internet possui diversos *sites* públicos que oferecem *datasets*[3] gratuitos, destinados a fins acadêmicos e divulgação de dados. Um exemplo é o *Kaggle* (KAGGLE, 2019), *site* de uma comunidade *online* de cientistas de dados com propriedade do *Google*. Fundada em 2010, permite que usuários publiquem dados relacionados a área de ciência de dados. A partir do *Kaggle* foram selecionados dois *datasets* utilizados para realizar os experimentos neste trabalho.

O primeiro *dataset* selecionado foi disponibilizado pela *Centers for Medicare & Medicaid Services* (CMS). O mesmo contém um resumo das atividades anuais da

[3] *Datasets*: conjunto ou coleção de dados tabulados.

Medicare Part B (Seguro Médico Privado), com informações sobre taxa de serviços, equipamentos médicos e recursos utilizados, entre outros. Os atributos relevantes são: HCPCS_CD que indica o código dos suprimentos; PLACE_OF_SERVICE_CD que indica o número do local dos serviços; PSPS_NCH_PAYMENT_AMT que se refere a contagem do número de serviços que foram aceitos pela (CMS). Por fim o PSPS_HCPCS_ASC_IND_CD que indica se um procedimento de liberação de itens médicos está aprovado ou reprovado. Esse *dataset* foi importado e identificado como **dados_medicos** no MongoDB, utilizado para analisar o comportamento dos *indexs* únicos e compostos. Sua *collection* recebeu o nome de "**medicina**".

Outro *dataset* escolhido possui uma lista de endereços (ruas e avenidas) de diversas cidades dentro do território nacional. Foi utilizado para analisar o comportamento de um *index* do tipo texto. Os atributos relevantes são REGGION que contém a sigla de um estado e CITY que especifica o nome de uma cidade. Tal *dataset* foi importado e identificado como **dados_brasil** e sua *collection* recebeu o nome de "**enderecos**".

Depois de concluída a importação dos arquivos CSV e a criação das *collecitons*, foi possível observar conforme o Quadro 2, a forma como as bases de dados e suas respectivas *collecitons* ficaram dispostas no MongoDB. Isso foi possível fazendo o uso dos comandos:

- **show *databases*:** Exibe o nome das bases de dados disponíveis;
- **show *collection*:** Exibe as *collections* disponíveis para cada base de dados;
- **use <database>:** Seleciona uma base de dados para manipulação. Utilizou-se o comando *use* seguido do nome da base.

Quadro 2 – Comandos MongoDB

Comando	Resultado
show databases	- dados_medicos (Importado do CSV) - dados_brasil (Importado do CSV)
use dados_medicos show collection	medicina
use dados_brasil show collection	enderecos

Fonte: Autoria própria

Utilizou-se o comando "**db.<collection>.<parameter>**", onde "*collection*" é o nome da coleção e "*parameter*" é o parâmetro solicitado, neste caso **dataSize()**, que retorna o tamanho da *collection* em *bytes.* Por meio dos comandos ilustrados no Quadro 3 – Parâmetro da collection no MongoDBQuadro 3, pode-se observar o espaço em disco necessário para armazenar as collections.

Quadro 3 – Parâmetro da collection no MongoDB

Comando com os Parâmetros	Resultado
db.medicina.datasize()	9.12 GB (Convertido de *bytes* para *gigabytes*)
db.enderecos.dataSize()	11.2 GB (Convertido de *bytes* para *gigabytes*)

Fonte: Autoria própria

5.2 DISCUSSÃO DE RESULTADOS

Este tópico apresenta as discussões sobre os resultados obtidos por meio dos experimentos. Como forma de padronização os nomes das bases de dados utilizadas foram abreviados: "**dados_medicos"** abreviou-se pela letra **A**; "**dados_brasil"** optou-se pela letra **B**. As consultas submetidas foram catalogadas seguindo o padrão: <abreviação da base de dados>.Q<número da consulta>. Também utilizou-se o parâmetro *explain(true).executionStats* do comando *Explain()*, para obter detalhes do plano de execução, necessários para as discussões deste tópico.

Os planos de execução foram aferidos em milissegundos (ms). Para discussão e análise de cada parâmetro utilizou-se como referência o plano gerado pela execução da primeira consulta. Devido ao tamanho dos planos gerados, torna-se inviável introduzi-los em meio a esse Capítulo, estando disponíveis no APÊNDICE A - Plano de Execução das Consultas.

5.2.1 Index Único

As consultas que fizeram uso do parâmetro *Index* Único são apresentadas no Quadro 4.

Quadro 4 – Formato das Consultas – *INDEX* ÚNICO

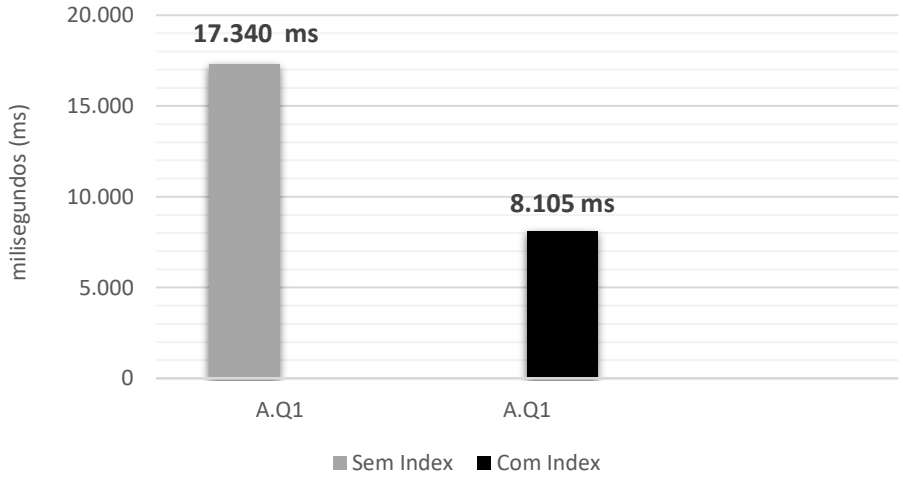
Nome Consulta	Comando
A.Q1	db.medicina.find({"HCPCS_CD":52648})
A.Q2	db.medicina.find({"HCPCS_CD":52648,"PLACE_OF_SERVICE_CD":11})
A.Q3	db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100}, "PSPS_HCPCS_ASC_IND_CD":"Y"})

Fonte: Autoria própria.

Para investigar o parâmetro *Index* Único, inicialmente submeteu-se a consulta **A.Q1,** que filtra apenas o campo HCPCS_CD. Como resultado obteve-se um tempo médio de 17.340 (ms), ilustrado no gráfico da Figura 16. Foi constado apenas um estágio *COLLSCAN* no componente "executionStages" do plano de execução da consulta (Apêndice A página 57). Isso obrigou o SGBD a realizar a leitura de todos os objetos presentes na *collection* (um total de 18.040.704 documentos) para realizar o filtro estipulado. O elemento "totalDocsExamined" do plano apresenta tal valor. Esse processo foi ineficiente, pois foram retornados apenas 1.744 documentos referente ao filtro "HCPCS_CD":52648**,** indicado pelo campo "nReturned".

Na tentativa de agilizar o tempo de resposta da consulta **A.Q1**, foi criado um *index* ordenado de forma crescente (1), para o campo **HCPCS_CD.** Submetendo-se a mesma consulta, agora fazendo uso do *index,* obteve-se um tempo médio de 8.105 (ms) ou seja, uma redução de mais de 9 segundos (s), conforme observado no gráfico da Figura 16

Figura 16 – Tempo médio de execução da consulta A.Q1



Fonte: Autoria própria

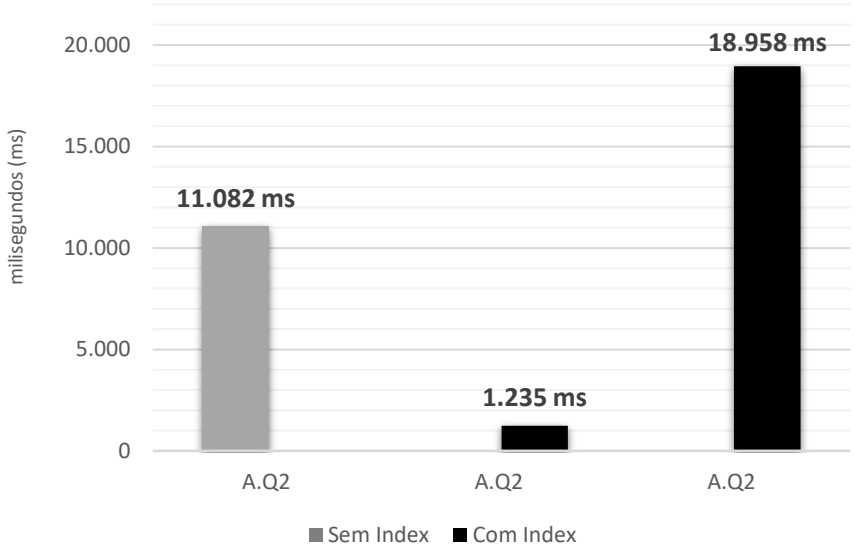
Os resultados evidenciaram a ineficiência do estágio *COLLSCAN* em uma consulta sem *index*. No plano de execução da consulta **A.Q1** já com *index*, constatou-se que o estágio *IXSCAN* necessitou de apenas 48 (ms) para buscar as chaves na memória que atendem ao filtro estipulado (Apêndice A página 58). Depois de recuperadas, realizou-se um acesso ao disco para recuperar dessa vez, o conteúdo das chaves e assim retornar o resultado final da consulta. Esse processo é representado no estágio *FETCH*, o qual despendeu um total de 6.944 (ms) para execução (Apêndice A página 58).

O estágio *FETCH* é o gargalo das consultas no MongoDB, pois os acessos ao disco são lentos. Além de recuperar o conteúdo das chaves de *indexs* em disco, pode-se ainda realizar um filtro sobre esses dados. Para um efetivo ganho de desempenho, deve-se utilizar esse estágio para recuperar a menor quantidade possível de documentos em disco. Para isso deve-se atentar aos seguintes elementos do plano de execução: “totalKeysExamined” (chaves de *indexs* examinadas) e “nReturned” (documentos retornados). Quanto mais próximo o valor de “totalKeysExamined” em relação a “nReturned”, mais eficiente será a consulta. Pois utilizar os *indexs* é extremamente ágil, assim acessa-se o disco somente para recuperar os documentos que precisarão ser retornados.

Em um novo cenário de testes submeteu-se a consulta **A.Q2** ao SGBD. Dessa vez foram filtrados dois campos, HCPCS_CD e PLACE_OF_SERVICE_CD**.** Obteve-se um tempo médio de execução de 11.082 (ms) conforme observado no gráfico da Figura 17 (Página 39). Na tentativa de minimizar o tempo de resposta, criou-se um *index* ordenado de forma crescente (1) no campo PLACE_OF_SERVICE_CD. Submetida novamente a consulta, obteve-se um tempo médio de 18.958 (ms) conforme a Figura 17 (Página 39). Constatou-se perda de desempenho em relação a mesma consulta sem a utilização de *index*. Em seu plano de execução verificou-se o valor 6.035.417 em “totalKeysExamined” (Apêndice A página 60), mesmo sendo um valor expressivo de chaves, o estágio *IXSCAN* necessitou de apenas 355 (ms) para buscá-las em memória. Porém recuperar em disco todos esses documentos e ainda realizar o filtro para atender a condição “HCPCS_CD”: 52648, indicada no elemento "filter" do plano, despendeu-se um tempo total de 5.969 (ms). Todo esse processo para retornar apenas 203 documentos, valor de “nReturned”. Em resumo, constatou-se uma grande amplitude entre os valores de “nReturned” e “totalKeysExamined”, essa observação comprova a perda de desempenho no tempo de resposta da consulta.

Ciente da perda de desempenho causado pela utilização de um *index* único no campo PLACE_OF_SERVICE_CD, submeteu-se novamente a consulta **A.Q2**, porém com o campo HCPCS_CD indexado de forma crescente (1). Como resultado obteve-se um tempo médio de 1.235 (ms) conforme ilustrado no gráfico da Figura 17 (Página 39). No plano de execução, observou-se que “resolver” a consulta pelo campo HCPCS_CD exigiu recuperar apenas 1.744 chaves de *indexs* no estágio *IXSCAN* (Apêndice A página 61). Esse valor é extremamente inferior comparado às 6.035.417 chaves recuperadas na consulta **A.Q2,** com *index* no campo PLACE_OF_SERVICE_CD (Apêndice A página 60). Esse ganho de desempenho é justificado pelo fato do valor de “totalKeysExamined”, nesse caso 1.744, estar próximo do valor do elemento “nReturned”, que apresentou 203 (Apêndice A página 61).

Figura 17 – Tempo médio de execução da consulta A.Q2

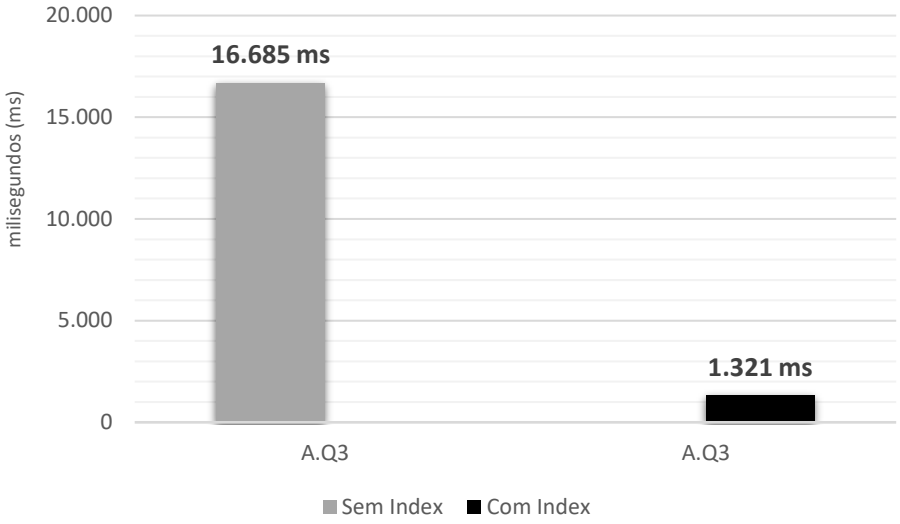


Fonte: Autoria própria

Ciente do desempenho de um *index* único, submeteu-se uma nova consulta **A.Q3,** dessa vez exigiu-se mais campos a serem filtrados em relação as consultas **A.Q1** e **A.Q2**. Essa consulta resultou em um tempo médio de 16.685 (ms) conforme gráfico da Figura 18. Para investigar esse cenário, submeteu-se a consulta **A.Q3**, porém com campo HCPCS_CD indexado, resultando em um tempo médio de 1.321 (ms), ilustrado no gráfico da Figura 18. Em resumo, elevar a quantidade de campos filtrados na consulta não acarretou em perda significativa de desempenho, levando em

consideração o tempo médio de resposta das consultas **A.Q1** e **A.Q2** com *índexs* únicos.

Figura 18 – Tempo médio de execução da consulta A.Q3



Fonte: Autoria própria

Nos cenários apresentados neste tópico, a utilização de um *index* único resultou em uma melhora expressiva de desempenho, reduzindo os tempos de respostas das consultas. Deve ser levado em consideração que esta estrutura quando utilizada de forma incorreta pode reduzir o desempenho, elevando o tempo de execução da consulta. Além da melhoria de desempenho, a utilização de *index* único pode resultar na utilização eficiente de recursos computacionais como memória principal, processador e disco.

5.2.2 Index Texto

As consultas que fizeram uso do parâmetro *Index* Texto são apresentadas no Quadro 5. As consultas **B.Q1**, **B.Q2** e **B.Q3** presentes no quadro são equivalentes entre si, a diferença é a forma da escrita. No MongoDB quando um campo faz uso de um *index* texto, a sintaxe para realizar a busca, utilizando este campo indexado é a seguinte: $text: {$search:"<”*texto”*>"}, onde <“texto”> é a cadeia de caracteres a ser buscadas.

Quadro 5 – Formato das Consultas – *INDEX* TEXTO

Nome Consulta	Comando
B.Q1	db.enderecos.find({"CITY":"Ponta Grossa","REGION":"PR"})
B.Q2	db.enderecos.find({"CITY":"Ponta Grossa", $text: {$search:"PR"}})
B.Q3	db.enderecos.find({ $text: {$search:"Ponta Grossa"},"REGION":"PR"})
A.Q4	db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100}, "PSPS_HCPCS_ASC_IND_CD":"Y"}

Fonte: Autoria própria.

Para investigar o parâmetro *index t*exto submeteu-se a consulta **B.Q1**, que filtra dois campos, CITY e REGION, ambos do tipo texto. Tal consulta apresentou um tempo médio de execução de 32.284 (ms) conforme o gráfico da Figura 19. Com intuito de analisar o desempenho desse tipo de estrutura, submeteu-se a consulta **B.Q2** ao SGBD, dessa vez com o campo REGION indexado, o tempo médio de execução foi de 15.930 (ms) ilustrado na Figura 19. Observou-se assim, uma redução significativa de cerca de 16 segundos (s) com a utilização desta estrutura.

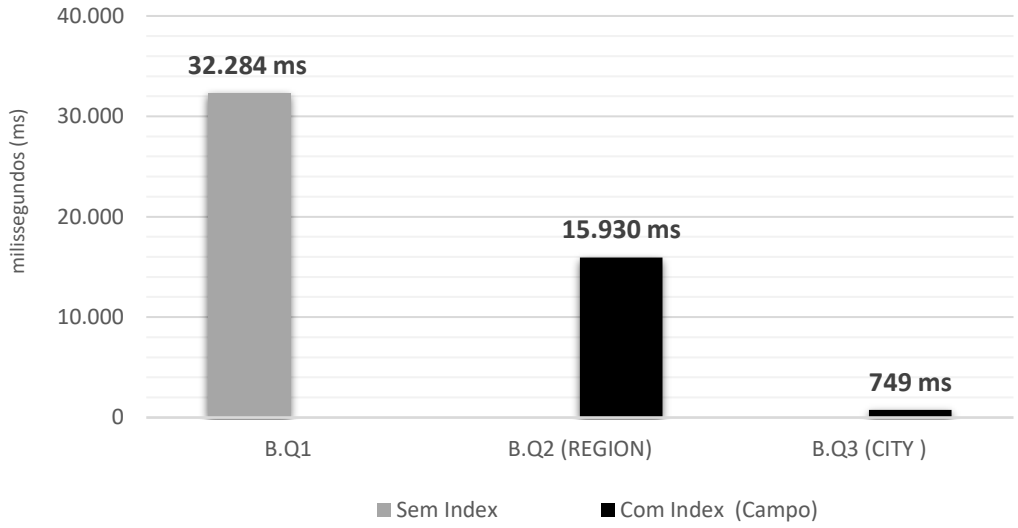
Considerando o plano de execução da consulta **B.Q2** (Apêndice A página 70), percebe-se que o campo REGION indexado possui um total de 4.203.822 correspondências referentes ao filtro “PR”, indicado em “nReturned” no estágio *IXSCAN*. Para verificar tal equivalência foi preciso uma comparação do tipo texto em toda essas chaves. Essa operação demanda certo tempo de processamento quando utilizados com uma grande quantidade de documentos, devido à complexidade dessa operação como: identificar letras maiúsculas, minúsculas, idiomas, entre outros.

Mesmo sendo ágil armazenar os *indexs* na memória, o estágio *IXSCAN* do plano de execução da consulta **B.Q2** (Apêndice A página 70) despendeu um total de 3.411 (ms) para execução, além de recuperar em memória as chaves, necessitou-se realizar ainda as comparações do tipo texto nas mesmas. Esse estágio é identificado como *TEXT* no elemento “inputStages”. Essas etapas totalizadas despenderam um total de 11.575 (ms), praticamente metade do tempo de execução da consulta. O restante do tempo foi utilizado pelo estágio *FETCH*, que ainda verificou o filtro "CITY": "Ponta Grossa". Essa operação precisou ainda acessar novamente o disco para realizar o filtro estipulado em CITY. Esse duplo acesso ao disco é constado pelo valor do campo “totalDocsExamined**”,** sendo 8.407.644, o dobro do valor de “totalKeysExamined”, que foi 4.203.822**.**

Na tentativa de melhorar o desempenho da consulta **B.Q2** submeteu-se a consulta **B.Q3** ao SGBD, dessa vez com um *index* texto sobre o campo CITY. Verificou-

se que o tempo médio de execução dessa consulta foi de apenas 749 (ms) conforme ilustrado na Figura 19. Uma redução de 15 segundos (s) em relação a **B.Q2** e cerca de 31 segundos (s) em relação a consulta **B.Q1** sem *index*. Observou-se também no plano de execução da consulta **B.Q3** (Apêndice A página 71), que somado os estágios *IXSCAN* e *TEXT* despenderam em um total de 509 (ms). Outra observação é o valor do elemento "totalKeysExamined", um total de 263.099 chaves referentes ao filtro "Ponta Grossa", sendo inferior ao valor de 4.203.822 observado no plano da consulta **B.Q2** (Apêndice A página 70) com *index* texto no campo REGION. Além de uma baixa amplitude em relação ao elemento "nReturned", que apresentou o valor de 104.428, assim justificando a significativa melhoria no tempo de resposta.

Figura 19 – Tempo médio de execução das consultas B.Q1, B.Q2 e B.Q3

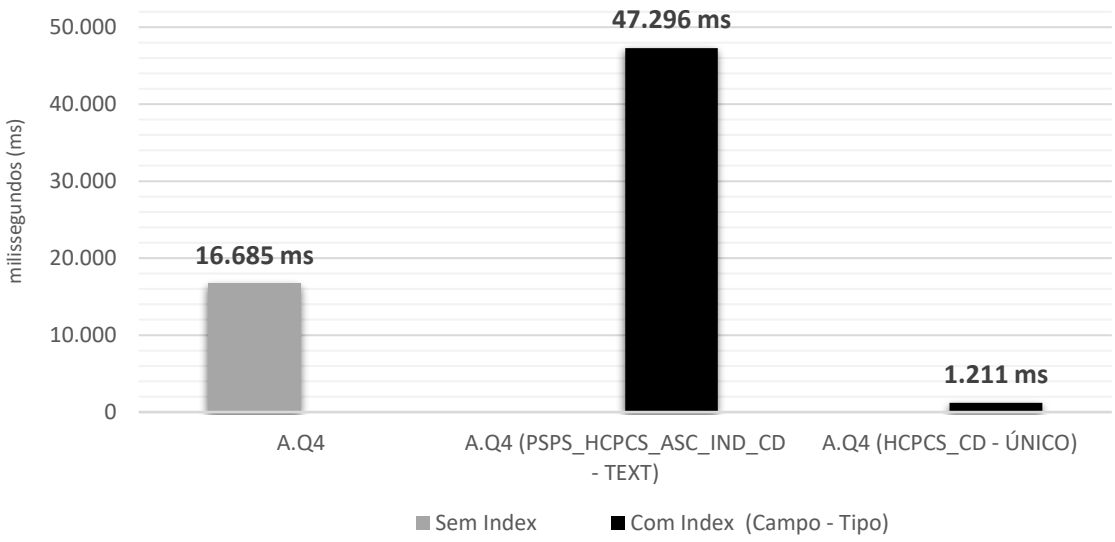


Fonte: Autoria própria

Em outro cenário submeteu-se a consulta **A.Q4** exigindo uma comparação do tipo texto no campo PSPS_HCPCS_ASC_IND_CD. Esse campo possui apenas dois valores, "Y" ou "N". Sabe-se que comparações do tipo texto são custosas quando processada em grandes quantidades. Nesse caso submeteu-se a consulta **A.Q4** com *index* texto no campo PSPS_HCPCS_ASC_IND_CD, a mesma resultou em um tempo médio de 47.296 (ms) conforme ilustrado no gráfico da Figura 20. Esse tempo foi significativamente pior considerando a consulta **A.Q3** (Equivalente a A.Q4) sem a utilização de *index*, que resultou em um tempo médio de 16.685 (ms), ilustrado na Figura 20. No plano de execução os estágios *IXSCAN* e *TEXT* somados despenderam um total de 72 segundos (s) (Apêndice A página 64). Esse declínio no desempenho

pode ser observado nos valores dos elementos “totalKeysExamined”, que apresentou o total de 8.195.511 e “nReturned” com 152 documentos retornados, o que demonstra uma alta amplitude nos valores dos elementos. A perda de desempenho se deve ao fato de consultas que envolvem comparações do tipo texto realizarem dois acessos ao disco conforme observado no plano de execução (Apêndice A página 64). No cenário de testes utilizando apenas o conceito de *index* único no campo HCPCS_CD, obteve-se um tempo médio de 1.211 (ms), ilustrado no gráfico da Figura 20.

Figura 20 – Tempo médio de execução da consulta A.Q4



Fonte: Autoria própria

Utilizar o conceito de *index* texto para determinado campo, que possui uma grande quantidade de dados exige tempo elevado de processamento, pois é necessário realizar comparações do tipo texto. A escolha de um *index* texto assertivo requer um grande conhecimento da base de dados e uma análise minuciosa dos planos de execução das consultas.

5.2.3 Index Composto

A consulta que fez uso do parâmetro *Index* composto são apresentadas no Quadro 6.

Quadro 6 – Formato das Consultas – *INDEX* COMPOSTO

Nome Consulta	Comando
A.Q5	db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100}, "PSPS_HCPCS_ASC_IND_CD":"Y"})

Fonte: Autoria própria

Para comprovar a eficiência dos *indexs* composto submeteu-se a consulta **A.Q5** ao SGBD, que possui três campos a serem filtrados: HCPCS_CD, PSPS_NCH_PAYMENT_AMT e PSPS_HCPCS_ASC_IND_CD. Esta consulta submetida resultou em um tempo médio de execução de 16.685 (ms) conforme ilustra a Figura 21. Como descrito, os *index* de campo único são eficientes quando utilizados de forma correta. Assim pode-se pensar que criar um *index* único para diversos campos pode ser uma técnica eficiente.

Para investigar a afirmação descrita anteriormente submeteu-se a consulta **A.Q5** fazendo uso de dois *indexs* únicos, nos campos HCPCS_CD e PSPS_NCH_PAYMENT_AMT, resultando em um tempo médio de execução de 1.185 (ms) ilustrado na Figura 21. No plano de execução (Apêndice A página 66) observou-se que a consulta foi "resolvida" utilizando apenas um *index,* descartando os demais. O elemento "indexName" indica qual campo indexado foi utilizado, nesse caso o campo HCPCS_CD. Isso ocorre pelo fato do MongoDB utilizar apenas um *index* para executar a consulta. O mesmo seleciona o campo indexado mais eficiente dentre os disponíveis. Consequentemente criar diversos *index* único não é coerente. Para essas situações o MongoDB permite a criação de *indexs* composto com vários campos.

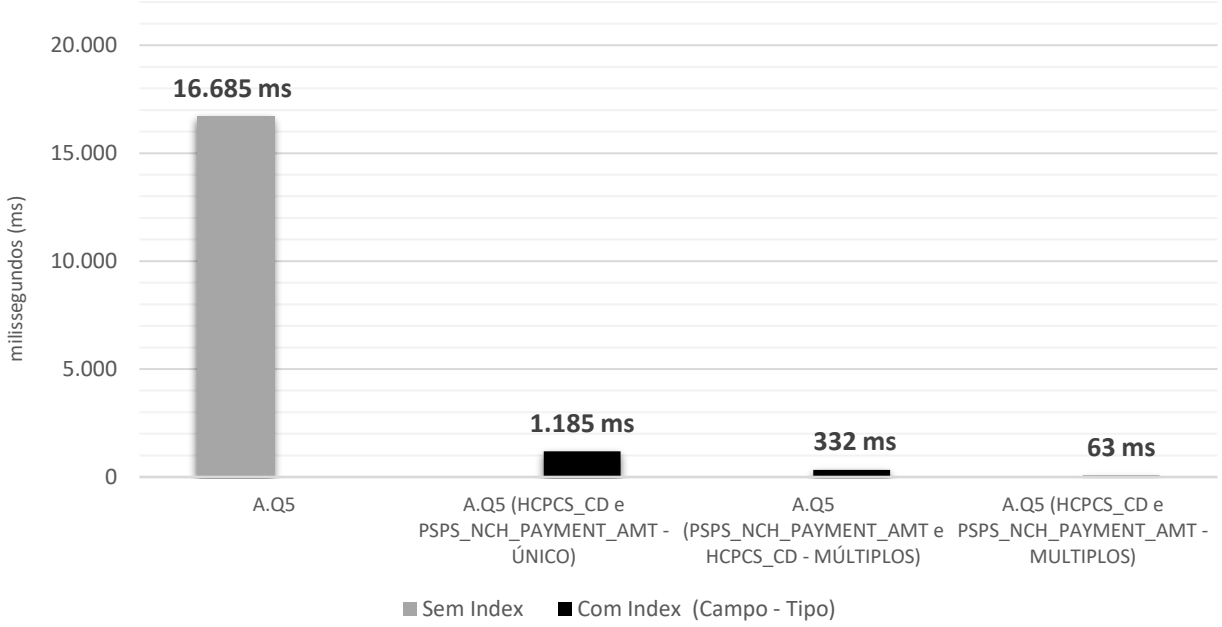
Para análise de desempenho, submeteu-se a consulta **A.Q5**, dessa vez utilizando um *index* composto nos campos PSPS_NCH_PAYMENT_AMT e HCPCS_CD, nessa ordem. O tempo médio de execução foi de 332 (ms) ilustrado na Figura 21, percebeu-se assim, um ganho de desempenho significativo. Porém analisando o plano de execução (Apêndice A página 67), identificou-se uma ineficiência na "resolução" da consulta. Para retornar o valor 152 em "nReturned" foi necessário ler somente 152 documentos, representado pelo campo "totalDocsExamined". Até esse ponto, pode-se afirmar que a consulta foi eficiente, pois leu somente os documentos necessários para retornar o resultado final. Em contrapartida o campo "totalKeysExamined" apresentou o valor de 20.153 chaves. Observa-se assim uma quantidade elevada de chaves pesquisadas para retornar apenas 152 documentos. Apesar do ganho de desempenho, pode-se afirmar que a sua execução não foi

eficiente. As chaves de *indexs* são armazenadas na memória principal, utilizá-la de maneira desnecessária não é uma boa prática.

A ineficiência observada na consulta **A.Q5** com *index* composto é justificada pela ordem de criação dos *indexs*. Incialmente ler o campo PSPS_NCH_PAYMENT_AMT exige um número maior de correspondências em relação ao campo HCPCS_CD. Já tratado anteriormente, a eficiência das consultas com o *index* no campo HCPCS_CD, justifica usá-lo primeiro e em seguida indexar os demais campos. Assim evita-se leituras desnecessárias de chaves de *indexs* na memória.

Com a finalidade de comprovar a eficiência dos *indexs* composto sobre os campos na ordem correta, submeteu-se novamente a consulta **A.Q5**, desta vez com um *index* composto na seguinte ordem dos campos HCPCS_CD e PSPS_NCH_PAYMENT_AMT. No plano de execução (Apêndice A página 68), observa-se a maior eficiência da consulta. O valor dos campos “totalKeysExamined” e “nReturned” foram iguais (152), ou seja, o SGBD executou a consulta somente com os *indexs* composto. Tal execução apresentou um tempo médio de 63 (ms), uma redução no tempo de resposta em relação a consulta com os *indexs* na ordem inversa, que foi de 332 (ms) conforme ilustrado no gráfico da Figura 21.

Figura 21 – Tempo médio de execução da consulta A.Q5



Fonte: Autoria própria

Os *indexs* composto são poderosos no MongoDB, porém precisam ser manipulados com cautela. Exigem grande quantidade de memória RAM, dependendo do número de campos indexados. Assim torna-se imprescindível utilizá-los de maneira assertiva, e os testes de desempenho evidenciaram essa afirmação.

5.2.4 THP (Transparent Huge Pages)

As consultas que fizeram uso do parâmetro *THP* são apresentadas no Quadro 7. Salienta-se o fato desse parâmetro assumir os seguintes valores: habilitado e desabilitado. Sendo a opção padrão do SO configurado como habilitado.

Quadro 7 – Formato das Consultas THP

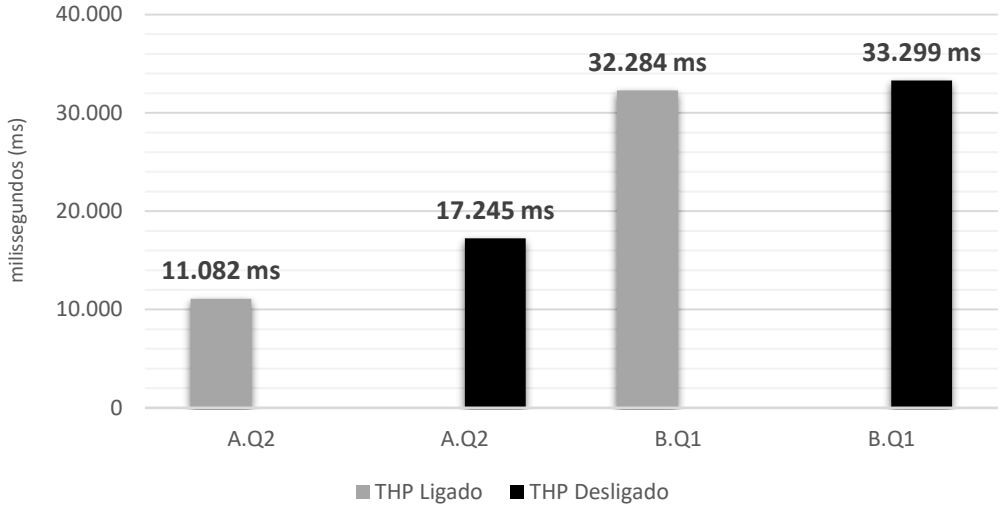
Nome Consulta	Comando
A.Q2	db.medicina.find({"HCPCS_CD":52648,PLACE_OF_SERVICE_CD:11})
B.Q1	db.enderecos.find({"CITY":"Ponta Grossa","REGION":"PR"})
B.Q3	db.enderecos.find({ $text: {$search:"Ponta Grossa"},"REGION":"PR"})

Fonte: Autoria própria

Com intuito de investigar o parâmetro THP habilitado, inicialmente utilizou-se como referências as consultas **A.Q2**, cujo tempo médio de execução foi de 11.082 (ms) e **B.Q1** que submetida obteve-se um tempo médio de 32.284 (ms), conforme ilustrado no gráfico da Figura 22. Em seguida desabilitou-se o parâmetro THP, obtendo os seguintes tempos médios: 17.245 (ms) em **A.Q2** e 33.299 (ms) em **B.Q1**, ilustrados na Figura 22. Observou-se perda de desempenho significativa em **A.Q2**, de cerca de 6 segundos (s), já em **B.Q1** verificou-se um aumento de 1 segundos (s) no tempo de execução da consulta, conforme o gráfico da Figura 22.

Considerando a carga de trabalho de leitura, aliado a ausência de *indexs* torna-se danoso ao desempenho do SGBD a utilização do parâmetro THP desabilitado. Esse fato deve-se principalmente à essas consultas exigirem acessos sequenciais ao disco, tornando-se lentas. Além da lentidão, uma piora no desempenho não justifica desabilitar o parâmetro THP.

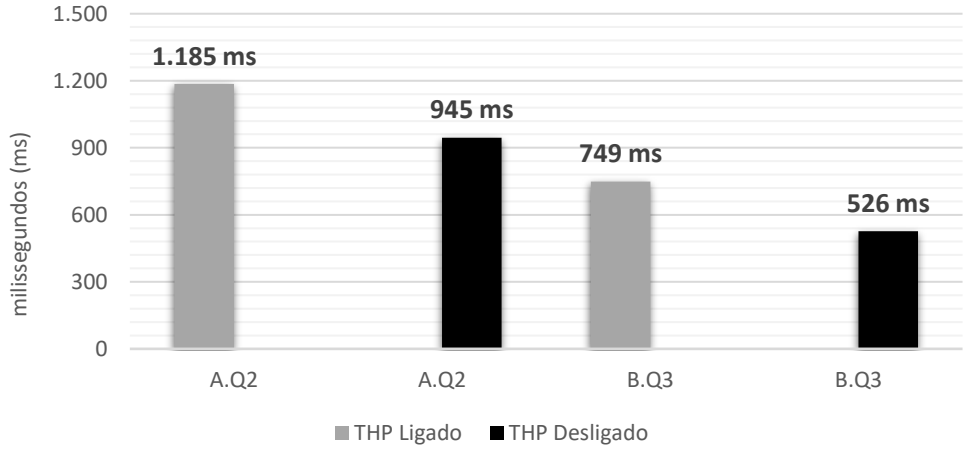
Figura 22 – Tempo médio de execução das consultas A.Q2 e B.Q1 sem Index com o parâmetro THP (Habilitado/Desabilitado)



Fonte: Autoria própria

Fazendo o uso da estrutura de dados *index*, submeteu-se novamente as consultas **A.Q2** e **B.Q3**. Dessa vez com o parâmetro THP habilitado obteve-se o tempo médio de 1.185 (ms) em **A.Q2.** Já na consulta **B.Q3** o tempo médio foi de 749 (ms), conforme ilustrado no gráfico da Figura 23. Em seguida desabilitado o parâmetro THP e submetidas novamente as consultas, obteve-se os seguintes tempos: 945 (ms) em **A.Q2** e 526 (ms) em **B.Q3** respectivamente, todos esses tempos estão ilustrados na Figura 23.

Figura 23 – Tempo médio de execução das consultas A.Q2 e B.Q3 com Index com o parâmetro THP (Habilitado/Desabilitado)



Fonte: Autoria própria

Em resumo observou melhoria de desempenho nos tempos de respostas de todas as consultas com *indexs*. A utilização dos *indexs* de forma assertiva combinado com o parâmetro THP desabilitado apresentou resultados positivos. Tal combinação é significativa quanto ao desempenho das consultas, assim justifica desabilitar o parâmetro THP do SO.

6 CONCLUSÃO

O modelo orientado a documentos possui diversas vantagens, tornando atrativo a sua utilização quando há necessidade de processamento e armazenamento de grandes quantidades de dados. Tomando o documento como formato de armazenamento, sua flexibilidade de esquema permite buscas menos complexas, sem a utilização de uniões e junções entre documentos. Ademais abdicar das propriedades ACID garante melhor desempenho, pois assim evita-se verificações de integridade referencial. Todas essas características empregadas foram a solução encontrada para suprir as limitações de desempenho encontradas em sistemas que utilizam o modelo relacional como forma de armazenamento.

Técnicas de *tuning* quando aplicadas em sistemas de banco de dados são poderosas formas de obter ganhos de desempenho. Expandidas para os sistemas não convencionais, sua utilização de forma assertiva contribui positivamente para a melhoria no tempo de execução das consultas. Porém deve-se considerar que a aplicação de tais técnicas de forma equivocada, pode acarretar em perda de desempenho significativa do SGBD.

Neste trabalho a aplicação do *tuning* em um SGBD orientado a documentos apresentou resultados positivos. Os experimentos focaram em uma carga de trabalho específica de consultas. A utilização da estrutura de dados *index* quando aplicada de forma assertiva obtém significativa redução no tempo de resposta das consultas, em relação a não utilização dessa estrutura. Porém o emprego de forma equivocada de um *index* pode ser danoso ao desempenho do SGBD, quando por exemplo, a utilização dessa estrutura do tipo texto sobre um campo que exigiu um número elevado de correspondências. Essas comparações demandaram alto tempo de processamento do SGBD em relação a consulta que não utilizou esta estrutura.

Além da comprovação da eficiência dos *indexs*, este trabalho evidenciou por meio dos planos de execução das consultas, como devem ser realizados ajustes assertivos na base de dados utilizando esta estrutura, além de evidenciar ajustes equivocados a serem evitados. Tendo o conhecimento do processamento de consultas do SGBD, os resultados comprovaram que ajustar os *indexs* para executar as etapas que exigem o maior esforço computacional na consulta é uma tarefa eficiente. Consequentemente culminou em um melhor tempo de resposta pelo SGBD na

execução das consultas. Para alcançar esses resultados exigiu-se um conhecimento profundo dos planos de execução gerado pelo MongoDB.

Além dos resultados com a estrutura de dados *indexs*, comprovou-se que desabilitar o parâmetro THP foi positivo, apenas nas consultas que utilizavam esta estrutura de dados. Observou ainda uma perda de desempenho considerável nas consultas sem a utilização de *index*. Assim conclui-se que desabilitar o parâmetro THP é danoso quando utilizado em uma base de dados que não possui campos indexados.

6.1 TRABALHOS FUTUROS

Trabalhos futuros podem ser desenvolvidos considerando cargas de trabalhos voltadas às operações de escritas no MongoDB: Pode-se verificar o comportamento do SGBD combinando-se parâmetros *prescriptive* e *descriptive* devidamente ajustados.

A possibilidade de aplicação e análise da eficiência de técnicas de *tuning* sobre SGBDs *NoSQL* orientados a grafos, chave-valor e família de colunas são outros trabalhos que podem ser desenvolvidos.

Realizar o *tuning* de SGBDs *NoSQL* em ambientes virtualizados é outra possibilidade de pesquisa. Verificar se as regras de *tuning* aplicadas em ambientes dedicados são realmente eficientes em ambientes virtualizados, é uma pesquisa de interesse estratégico às organizações.

REFERÊNCIAS

BAYER, R; MCCREIGHT, E. (1972). **Organization and Maintenance of Large Ordered Indexes**. Acta Informatica, Vol. 1, Fasc. 3, 1972.

BINI, T**. Aplicação do Algoritmo de Kruskal na otimização de consultas com Múltiplas Junções Relacionais**, Capítulo 2, 2009. 97 f. Dissertação (Mestrado em Informática) – Programa de Pós-graduação em Informática da Universidade Federal do Paraná, 2009.

BINI, T. **ANALISE DA APLICABILIDADE DAS REGRAS DE OURO AO TUNING DE SISTEMAS GERENCIADORES DE BANCOS DE DADOS RELACIONAIS EM AMBIENTES DE COMPUTACÃO EM NUVEM**. Dissertação (Doutorado em Ciência da Computação) – Programa de Pós-Graduação em Informática, Setor de Ciências Exatas da Universidade Federal do Paraná, 2014.

BRITO, W. **Bancos de Dados NoSQL x SGBDs Relacionais: Análise Comparativa**. FFB, 2010. Disponível em: <http://docplayer.com.br/433629-Bancos-de-dados-nosql-x-sgbds-relacionais-analise-comparativa.html> Acesso 22 Ago. 2019.

CASSANDRA, 2019. Disponível em: <ww.cassandra.apache.org/> Acesso 22 Ago. 2019

CHICAGO, 2015. Disponível em: <https://www.mongodb.com/customers/city-of-chicago> Acesso 02 Set. 2019.

CHODOROW, K**. MongoDB: The Definitive Guide.** O'Reilly Media Inc., USA (2013).

CODD, F. **Relational completeness of data base sublanguages**.
R. Rustin (Ed.), Data Base Systems, Englewood Cliffs, USA, Prentice-Hall (1972).

CORBET, J. 2010. **Memory compaction**. Disponível em: <lwn.net/Articles/368869/>.

COUCHDB, 2019. Disponível em: <www.couchdb.apache.org/> Acesso 22 Ago. 2019.

DATE, J. **Introdução a sistemas de bancos de dados.** Tradução [da 7. Ed. Americana] Vanderberg Dantas de Souza, Plubicare Consultoria e Serviços. – Rio de Janeiro: campus, 2004.

DB-ENGINES RANKING, 2019. Disponível em: <www.db-engines.com/en/ranking> Acesso 22 Ago. 2019.

DOCUMENTAÇÃO MONGODB, 2019. Disponível em: <www.docs.mongodb.com/> Acesso 22 Ago. 2019.

DOMINICO, S. **TUNING: UM ESTUDO SOBRE A OTIMIZAÇÃO DE DESEMPENHO DE SISTEMAS GERENCIADORES DE BANCO DE DADOS RELACIONAIS SOB CARGA DE TRABALHO DE SUPORTE A DECISÃO**. Dissertação (Graduação em Tecnologia em Sistemas para Internet) – Universidade Tecnológica Federal do Paraná - UTFPR, 2013.

EA GAMES, 2014. Disponível em: < https://www.mongodb.com/blog/post/ea-scores-mongodb-based-fifa-online-3?c=1534ec3d9c> Acesso: 02/09/2019.

ELMASRI, R; NAVATHE, B. **Sistemas de Bancos de Dados**. Addison-Wesley, 4a. 2011, edição em português.

GORMAN, M. **Understanding the Linux Virtual Memory Manager**. PrenticeHall PTR, Upper Saddle River, NJ, USA, 2004.

HBASE, 2019. Disponível em: <https://hbase.apache.org/> Acesso 22 Ago. 2019.

JSON, 2019. Disponível em: <www.json.org/json-pt.html> Acesso 22 Ago. 2019.

KAGGLE, 2019. Disponível em: <www.kaggle.com/datasets> Acesso 12 Ago. 2019.

KORTH, F; SILBERCHATZ, A; SUDARSHAN, S. **Sistema de banco de dados**. 5.ed. Rio de Janeiro: Campus, 2006.

KUSS, L. **Otimização de desempenho do Hadoop MapReduce: um caso prático**. Dissertação (Graduação em Bacharelado em Ciência da Computação) – Universidade Tecnológica Federal do Paraná - UTFPR, 2017.

LEAVITT, N. **"Will NoSQL Databases Live Up to Their Promise?"** Computer, vol. 43, no. 2. 2010

LIGHTSTONE, S; TEOREY, J; NADEAU, T. **Physical Database Design: The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 978-0-12-369389-1.

LÓSCIO, F; OLIVEIRA, D; PONTES, S. **Nosql no desenvolvimento de aplicações web colaborativas VIII Simpósio Brasileiro de Sistemas Colaborativos, Brasil**. 2011. Disponível em: <www.addlabs.uff.br/sbsc_site/SBSC2011_NoSQL.pdf> Acesso: 22/08/2019.

MONGODB, 2019. Disponível em: <www.mongodb.com/> Acesso 01 Ago. 2019.

MYSQL, 2019. Disponível em: <www.mysql.com/> Acesso 01 Ago. 2019.

NEO4J, 2019. Disponível em: <www.neo4j.com/product/?ref=home-banner/> Acesso 22 Ago. 2019.

OLIVEIRA, F. **Avaliação de desempenho de gerenciadores de bancos de dados multi-modelo em aplicações com persistência poliglota**. Dissertação (Dissertação de Mestrado em Ciência da Computação) – Faculdade Campo Limpo Paulista - UNIFACCAMP, 2017.

ORACLE, 2019. Disponível em: <www.oracle.com/br/database/> Acesso 22 Ago. 2019.

ORIENTDB, 2019. Disponível em: <www.orientdb.com/> Acesso 23 Ago. 2019.

PENTEADO, R. *et al*. **Um estudo sobre bancos de dados em grafos nativos**. 2014. Disponível em: <www.inf.ufpr.br/carmem/pub/erbd2014-artigo.pdf> Acesso 22 Ago. 2019.

POSTGRESQL, 2019. Disponível em: <www.postgresql.org//> Acesso 22 Ago. 2019.

PRIGOL, E. **Redes Sociais com Banco de Dados Orientado a Grafos**. Dissertação (Graduação em Ciência da Computação) – Universidade Regional do Noroeste do Estado do Rio Grande do Sul – UNIJUÍ. 2016.

RAMAKRISHNAN, R; GEHRKE, J. **Sistema Gerenciador de Banco de Dados**. 3.ed. Editora Bookman. Porto Alegre, 2008. ISBN 978-85-63308-77-1.

REDIS, 2019. Disponível em: <www.redis.io/> Acesso 22 Ago. 2019.

RIAK, 2019. Disponível em: <www.riak.com/> Acesso 26 Ago. 2019.

RONALDO, M. *et al*. **Dados Semi-estruturados**. 2000. Disponível em: <www.ime.usp.br/~jef/semi-estruturado.pdf> Acesso: 12/08/2019.

SADALAGE, J; FOWLER, M. **NoSQL distilled: a brief guide to the emerging world of polyglot persistence**. Pearson Education, Inc., 2013.

SHASHA, D; BONNET P. **Database Tuning: Principles, Experiments, and Troubleshooting Techniques**. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2002.

SOROR, A; ABOULNAGA, A; SALEM, K. **Database virtualization: A new frontier for database tuning and physical design**. In Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop, ICDEW ’07, Washington, DC, USA. IEEE Computer Society.

SOROR, A; MINHAS, F; ABOULNAGA, A; SALEM, K; KOKOSIELIS, P; KAMATH, S. **Automatic virtual machine configuration for database workloads**. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD ’08, New York, NY, USA. ACM.

TRAMONTINA, B. **Database Tuning: Configurando o Interbase e o PostgreSQL. 2008**. Disponível em: <www.ic.unicamp.br/~geovane/mo410-091/Ch20-ConfigInterbasePosgres-art.pdf>. Acesso 25 Set. 2019.

TRANSPARENT HUGE PAGES, 2011**.** Disponível em: <www.lwn.net/Articles/423584/> Acesso 01 Set. 2019.

UBUNTU, 2019. Disponível em: <www.help.ubuntu.com/stable/ubuntu-help/> Acesso 22 Ago. 2019.

WANZELLER, D**. Investigando o Uso de Bancos de Dados Orientados a Documentos para Gerenciar Informações da Administração Pública**. Dissertação (Graduação em Computação - Licenciatura) – Universidade de Brasília – UnB. 2013.

APÊNDICE A - Plano de Execução das Consultas

A.Q1 – Sem *Index*

```
> db.medicina.find({"HCPCS_CD":52648}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 1744,
    "executionTimeMillis" : 32402,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 18040704,
    "executionStages" : {
        "stage" : "COLLSCAN",
        "filter" : {
            "HCPCS_CD" : {
                "$eq" : 52648
            }
        },
        "nReturned" : 1744,
        "executionTimeMillisEstimate" : 4558,
        "works" : 18040706,
        "advanced" : 1744,
        "needTime" : 18038961,
        "needYield" : 0,
        "saveState" : 140965,
        "restoreState" : 140965,
        "isEOF" : 1,
        "direction" : "forward",
        "docsExamined" : 18040704
    },
    "allPlansExecution" : [ ]
}
```

A.Q1 – Com *Index* Único no campo HCPCS_CD

```
> db.medicina.find({"HCPCS_CD":52648}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 1744,
    "executionTimeMillis" : 8109,
    "totalKeysExamined" : 1744,
    "totalDocsExamined" : 1744,
    "executionStages" : {
        "stage" : "FETCH",
        "nReturned" : 1744,
        "executionTimeMillisEstimate" : 6944,
        "works" : 1745,
        "advanced" : 1744,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 380,
        "restoreState" : 380,
        "isEOF" : 1,
        "docsExamined" : 1744,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 1744,
            "executionTimeMillisEstimate" : 48,
            "works" : 1745,
            "advanced" : 1744,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 380,
            "restoreState" : 380,
            "isEOF" : 1,
            "keyPattern" : {
                "HCPCS_CD" : 1
            },
            "indexName" : "HCPCS_CD_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "HCPCS_CD" : [ ]
            },
            "isUnique" : false,
            "isSparse" : false,
            "isPartial" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "HCPCS_CD" : [
                    "[52648.0, 52648.0]"
                ]
            },
            "keysExamined" : 1744,
            "seeks" : 1,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [ ]
}
>
```

A.Q2 – Sem *Index*

```
> db.medicina.find({"HCPCS_CD":52648,PLACE_OF_SERVICE_CD:11}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 203,
    "executionTimeMillis" : 10200,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 18040704,
    "executionStages" : {
        "stage" : "COLLSCAN",
        "filter" : {
            "$and" : [
                {
                    "HCPCS_CD" : {
                        "$eq" : 52648
                    }
                },
                {
                    "PLACE_OF_SERVICE_CD" : {
                        "$eq" : 11
                    }
                }
            ]
        },
        "nReturned" : 203,
        "executionTimeMillisEstimate" : 223,
        "works" : 18040706,
        "advanced" : 203,
        "needTime" : 18040502,
        "needYield" : 0,
        "saveState" : 140943,
        "restoreState" : 140943,
        "isEOF" : 1,
        "direction" : "forward",
        "docsExamined" : 18040704
    },
    "allPlansExecution" : [ ]
}
```

A.Q2 – Com *Index* Único no campo PLACE_OS_SERVICE_CD

```
> db.medicina.find({"HCPCS_CD":52648,PLACE_OF_SERVICE_CD:11}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 203,
    "executionTimeMillis" : 33318,
    "totalKeysExamined" : 6035417,
    "totalDocsExamined" : 6035417,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "HCPCS_CD" : {
                "$eq" : 52648
            }
        },
        "nReturned" : 203,
        "executionTimeMillisEstimate" : 5969,
        "works" : 6035418,
        "advanced" : 203,
        "needTime" : 6035214,
        "needYield" : 0,
        "saveState" : 47186,
        "restoreState" : 47186,
        "isEOF" : 1,
        "docsExamined" : 6035417,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 6035417,
            "executionTimeMillisEstimate" : 355,
            "works" : 6035418,
            "advanced" : 6035417,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 47186,
            "restoreState" : 47186,
            "isEOF" : 1,
            "keyPattern" : {
                "PLACE_OF_SERVICE_CD" : 1
            },
            "indexName" : "PLACE_OF_SERVICE_CD_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "PLACE_OF_SERVICE_CD" : [ ]
            },
            "isUnique" : false,
            "isSparse" : false,
            "isPartial" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "PLACE_OF_SERVICE_CD" : [
                    "[11.0, 11.0]"
                ]
            },
            "keysExamined" : 6035417,
            "seeks" : 1,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [ ]
}
```

A.Q2 – Com *Index* Único no campo HCPCS_CD

```
> db.medicina.find({"HCPCS_CD":52648,PLACE_OF_SERVICE_CD:11}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 203,
    "executionTimeMillis" : 1338,
    "totalKeysExamined" : 1744,
    "totalDocsExamined" : 1744,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "PLACE_OF_SERVICE_CD" : {
                "$eq" : 11
            }
        },
        "nReturned" : 203,
        "executionTimeMillisEstimate" : 1308,
        "works" : 1745,
        "advanced" : 203,
        "needTime" : 1541,
        "needYield" : 0,
        "saveState" : 66,
        "restoreState" : 66,
        "isEOF" : 1,
        "docsExamined" : 1744,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 1744,
            "executionTimeMillisEstimate" : 23,
            "works" : 1745,
            "advanced" : 1744,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 66,
            "restoreState" : 66,
            "isEOF" : 1,
            "keyPattern" : {
                "HCPCS_CD" : 1
            },
            "indexName" : "HCPCS_CD_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "HCPCS_CD" : [ ]
            },
            "isUnique" : false,
            "isSparse" : false,
            "isPartial" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "HCPCS_CD" : [
                    "[52648.0, 52648.0]"
                ]
            },
            "keysExamined" : 1744,
            "seeks" : 1,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [ ]
}
```

A.Q3 – Sem *Index*

```
> db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},"PSPS_HCPCS_ASC_IND_CD":"Y"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 29157,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 18040704,
    "executionStages" : {
        "stage" : "COLLSCAN",
        "filter" : {
            "$and" : [
                {
                    "HCPCS_CD" : {
                        "$eq" : 52648
                    }
                },
                {
                    "PSPS_HCPCS_ASC_IND_CD" : {
                        "$eq" : "Y"
                    }
                },
                {
                    "PSPS_NCH_PAYMENT_AMT" : {
                        "$lt" : 100
                    }
                }
            ]
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 3483,
        "works" : 18040706,
        "advanced" : 152,
        "needTime" : 18040553,
        "needYield" : 0,
        "saveState" : 140956,
        "restoreState" : 140956,
        "isEOF" : 1,
        "direction" : "forward",
        "docsExamined" : 18040704
    },
    "allPlansExecution" : [ ]
}
>
```

A.Q3 – Com *Index* Único no campo HCPSCD

```
> db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},"PSPS_HCPCS_ASC_IND_CD":"Y"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 1321,
    "totalKeysExamined" : 1744,
    "totalDocsExamined" : 1744,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "$and" : [
                {
                    "PSPS_HCPCS_ASC_IND_CD" : {
                        "$eq" : "Y"
                    }
                },
                {
                    "PSPS_NCH_PAYMENT_AMT" : {
                        "$lt" : 100
                    }
                }
            ]
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 1315,
        "works" : 1745,
        "advanced" : 152,
        "needTime" : 1592,
        "needYield" : 0,
        "saveState" : 64,
        "restoreState" : 64,
        "isEOF" : 1,
        "docsExamined" : 1744,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 1744,
            "executionTimeMillisEstimate" : 66,
            "works" : 1745,
            "advanced" : 1744,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 64,
            "restoreState" : 64,
            "isEOF" : 1,
            "keyPattern" : {
                "HCPCS_CD" : 1
            },
            "indexName" : "HCPCS_CD_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "HCPCS_CD" : [ ]
            },
            "isUnique" : false,
            "isSparse" : false,
            "isPartial" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "HCPCS_CD" : [
                    "[52648.0, 52648.0]"
                ]
            },
            "keysExamined" : 1744,
            "seeks" : 1,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [ ]
}
>
```

A.Q4 – Com *Index* Texto no campo PSPS_HCPCS_ASC_IND_CD

```
> db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},$text:{$search:"Y"}}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 101404,
    "totalKeysExamined" : 8195511,
    "totalDocsExamined" : 16391022,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "$and" : [
                {
                    "HCPCS_CD" : {
                        "$eq" : 52648
                    }
                },
                {
                    "PSPS_NCH_PAYMENT_AMT" : {
                        "$lt" : 100
                    }
                }
            ]
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 29026,
        "works" : 8195512,
        "advanced" : 152,
        "needTime" : 8195359,
        "needYield" : 0,
        "saveState" : 64189,
        "restoreState" : 64189,
        "isEOF" : 1,
        "docsExamined" : 8195511,
        "alreadyHasObj" : 8195511,
        "inputStage" : {
            "stage" : "TEXT",
            "nReturned" : 8195511,
            "executionTimeMillisEstimate" : 28578,
            "works" : 8195512,
            "advanced" : 8195511,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 64189,
            "restoreState" : 64189,
            "isEOF" : 1,
            "indexPrefix" : {

            },
            "indexName" : "PSPS_HCPCS_ASC_IND_CD_text",
            "parsedTextQuery" : {
                "terms" : [
                    "y"
                ],
                "negatedTerms" : [ ],
                "phrases" : [ ],
                "negatedPhrases" : [ ]
            },
            "textIndexVersion" : 3,
            "inputStage" : {
                "stage" : "TEXT_MATCH",
                "nReturned" : 8195511,
                "executionTimeMillisEstimate" : 28545,
                "works" : 8195512,
                "advanced" : 8195511,
                "needYield" : 0,
                "saveState" : 64189,
                "restoreState" : 64189,
                "isEOF" : 1,
                "docsRejected" : 0,
                "inputStage" : {
                    "stage" : "FETCH",
                    "nReturned" : 8195511,
                    "executionTimeMillisEstimate" : 28503,
                    "works" : 8195512,
                    "advanced" : 8195511,
                    "needTime" : 0,
                    "needYield" : 0,
                    "saveState" : 64189,
                    "restoreState" : 64189,
                    "isEOF" : 1,
                    "docsExamined" : 8195511,
                    "alreadyHasObj" : 0,
                    "inputStage" : {
                        "stage" : "OR",
                        "nReturned" : 8195511,
                        "executionTimeMillisEstimate" : 5472,
                        "works" : 8195512,
                        "advanced" : 8195511,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 64189,
                        "restoreState" : 64189,
                        "isEOF" : 1,
                        "dupsTested" : 8195511,
                        "dupsDropped" : 0,
                        "inputStage" : {
                            "stage" : "IXSCAN",
                            "nReturned" : 8195511,
                            "executionTimeMillisEstimate" : 3752,
                            "works" : 8195512,
                            "advanced" : 8195511,
                            "needTime" : 0,
                            "needYield" : 0,
                            "saveState" : 64189,
                            "restoreState" : 64189,
                            "isEOF" : 1,
                            "keyPattern" : {
                                "_fts" : "text",
                                "_ftsx" : 1
                            },
                            "indexName" : "PSPS_HCPCS_ASC_IND_CD_text",
                            "isMultiKey" : false,
                            "isUnique" : false,
                            "isSparse" : false,
                            "isPartial" : false,
                            "indexVersion" : 2,
                            "direction" : "backward",
                            "indexBounds" : {

                            },
                            "keysExamined" : 8195511,
                            "seeks" : 1,
                            "dupsTested" : 0,
                            "dupsDropped" : 0
                        }
                    }
                }
            }
        }
    },
    "allPlansExecution" : [ ]
}
```

A.Q5 – Sem *Index*

```
> db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},"PSPS_HCPCS_ASC_IND_CD":"Y"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 29157,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 18040704,
    "executionStages" : {
        "stage" : "COLLSCAN",
        "filter" : {
            "$and" : [
                {
                    "HCPCS_CD" : {
                        "$eq" : 52648
                    }
                },
                {
                    "PSPS_HCPCS_ASC_IND_CD" : {
                        "$eq" : "Y"
                    }
                },
                {
                    "PSPS_NCH_PAYMENT_AMT" : {
                        "$lt" : 100
                    }
                }
            ]
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 3483,
        "works" : 18040706,
        "advanced" : 152,
        "needTime" : 18040553,
        "needYield" : 0,
        "saveState" : 140956,
        "restoreState" : 140956,
        "isEOF" : 1,
        "direction" : "forward",
        "docsExamined" : 18040704
    },
    "allPlansExecution" : [ ]
}
>
```

A.Q5 – Com *Index* Único nos campos HCPCS_CD e PSPS_NCH_PAYMENT_AMT

```
db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},"PSPS_HCPCS_ASC_IND_CD":"Y"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 1302,
    "totalKeysExamined" : 1744,
    "totalDocsExamined" : 1744,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "$and" : [
                {
                    "PSPS_HCPCS_ASC_IND_CD" : {
                        "$eq" : "Y"
                    }
                },
                {
                    "PSPS_NCH_PAYMENT_AMT" : {
                        "$lt" : 100
                    }
                }
            ]
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 1244,
        "works" : 1745,
        "advanced" : 152,
        "needTime" : 1592,
        "needYield" : 0,
        "saveState" : 64,
        "restoreState" : 64,
        "isEOF" : 1,
        "docsExamined" : 1744,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 1744,
            "executionTimeMillisEstimate" : 68,
            "works" : 1745,
            "advanced" : 1744,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 64,
            "restoreState" : 64,
            "isEOF" : 1,
            "keyPattern" : {
                "HCPCS_CD" : 1
            },
            "indexName" : "HCPCS_CD_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "HCPCS_CD" : [ ]
            },
            "isUnique" : false,
            "isSparse" : false,
            "isPartial" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "HCPCS_CD" : [
                    "[52648.0, 52648.0]"
                ]
            },
            "keysExamined" : 1744,
            "seeks" : 1,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [
        {
            "nReturned" : 101,
            "executionTimeMillisEstimate" : 756,
            "totalKeysExamined" : 1078,
            "totalDocsExamined" : 1078,
            "executionStages" : {
                "stage" : "FETCH",
                "filter" : {
                    "$and" : [
                        {
                            "PSPS_HCPCS_ASC_IND_CD" : {
                                "$eq" : "Y"
                            }
                        },
                        {
                            "PSPS_NCH_PAYMENT_AMT" : {
                                "$lt" : 100
                            }
                        }
                    ]
                },
                "nReturned" : 101,
                "executionTimeMillisEstimate" : 756,
                "works" : 1078,
                "advanced" : 101,
                "needTime" : 977,
                "needYield" : 0,
                "saveState" : 37,
                "restoreState" : 37,
                "isEOF" : 0,
                "docsExamined" : 1078,
                "alreadyHasObj" : 0,
                "inputStage" : {
                    "stage" : "IXSCAN",
                    "nReturned" : 1078,
                    "executionTimeMillisEstimate" : 68,
                    "works" : 1078,
                    "advanced" : 1078,
                    "needTime" : 0,
                    "needYield" : 0,
                    "saveState" : 37,
                    "restoreState" : 37,
                    "isEOF" : 0,
                    "keyPattern" : {
                        "HCPCS_CD" : 1
                    },
                    "indexName" : "HCPCS_CD_1",
                    "isMultiKey" : false,
                    "multiKeyPaths" : {
                        "HCPCS_CD" : [ ]
                    },
                    "isUnique" : false,
                    "isSparse" : false,
                    "isPartial" : false,
                    "indexVersion" : 2,
                    "direction" : "forward",
                    "indexBounds" : {
                        "HCPCS_CD" : [
                            "[52648.0, 52648.0]"
                        ]
                    },
                    "keysExamined" : 1078,
                    "seeks" : 1,
                    "dupsTested" : 0,
                    "dupsDropped" : 0
                }
            }
        },
        {
            "nReturned" : 0,
            "executionTimeMillisEstimate" : 29,
            "totalKeysExamined" : 1078,
            "totalDocsExamined" : 1078,
            "executionStages" : {
                "stage" : "FETCH",
                "filter" : {
                    "$and" : [
                        {
                            "HCPCS_CD" : {
                                "$eq" : 52648
                            }
                        },
                        {
                            "PSPS_HCPCS_ASC_IND_CD" : {
                                "$eq" : "Y"
                            }
                        }
                    ]
                },
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 23,
                "works" : 1078,
                "advanced" : 0,
                "needTime" : 1078,
                "needYield" : 0,
                "saveState" : 64,
                "restoreState" : 64,
                "isEOF" : 0,
                "dupsExamined" : 1078,
                "docsExamined" : 1078,
                "alreadyHasObj" : 0,
                "inputStage" : {
                    "stage" : "IXSCAN",
                    "nReturned" : 1078,
                    "executionTimeMillisEstimate" : 23,
                    "works" : 1078,
                    "advanced" : 1078,
                    "needTime" : 0,
                    "needYield" : 0,
                    "saveState" : 64,
                    "restoreState" : 64,
                    "isEOF" : 0,
                    "keyPattern" : {
                        "PSPS_NCH_PAYMENT_AMT" : 1
                    },
                    "indexName" : "PSPS_NCH_PAYMENT_AMT_1",
                    "isMultiKey" : false,
                    "multiKeyPaths" : {
                        "PSPS_NCH_PAYMENT_AMT" : [ ]
                    },
                    "isUnique" : false,
                    "isSparse" : false,
                    "isPartial" : false,
                    "indexVersion" : 2,
                    "direction" : "forward",
                    "indexBounds" : {
                        "PSPS_NCH_PAYMENT_AMT" : [
                            "[-inf.0, 100.0)"
                        ]
                    },
                    "keysExamined" : 1078,
                    "seeks" : 1,
                    "dupsTested" : 0,
                    "dupsDropped" : 0
                }
            }
        }
    ]
}
```

A.Q5 – Com *Indexs* Compostos nos campos PSPS_NCH_PAYMENT_AMT e HCPCS_CD

```
> db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},"PSPS_HCPCS_ASC_IND_CD":"Y"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 901,
    "totalKeysExamined" : 20153,
    "totalDocsExamined" : 152,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "PSPS_HCPCS_ASC_IND_CD" : {
                "$eq" : "Y"
            }
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 387,
        "works" : 20153,
        "advanced" : 152,
        "needTime" : 20000,
        "needYield" : 0,
        "saveState" : 166,
        "restoreState" : 166,
        "isEOF" : 1,
        "docsExamined" : 152,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 152,
            "executionTimeMillisEstimate" : 238,
            "works" : 20153,
            "advanced" : 152,
            "needTime" : 20000,
            "needYield" : 0,
            "saveState" : 166,
            "restoreState" : 166,
            "isEOF" : 1,
            "keyPattern" : {
                "PSPS_NCH_PAYMENT_AMT" : 1,
                "HCPCS_CD" : 1
            },
            "indexName" : "PSPS_NCH_PAYMENT_AMT_1_HCPCS_CD_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "PSPS_NCH_PAYMENT_AMT" : [ ],
                "HCPCS_CD" : [ ]
            },
            "isUnique" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "PSPS_NCH_PAYMENT_AMT" : [
                    "[-inf.0, 100.0)"
                ],
                "HCPCS_CD" : [
                    "[52648.0, 52648.0]"
                ]
            },
            "keysExamined" : 20153,
            "seeks" : 20001,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [ ]
}
```

A.Q5 – Com *Index* Composto nos campos HCPCS_CD e PSPS_NCH_PAYMENT_AMT

```
> db.medicina.find({"HCPCS_CD":52648,"PSPS_NCH_PAYMENT_AMT":{$lt:100},"PSPS_HCPCS_ASC_IND_CD":"Y"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 152,
    "executionTimeMillis" : 186,
    "totalKeysExamined" : 152,
    "totalDocsExamined" : 152,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "PSPS_HCPCS_ASC_IND_CD" : {
                "$eq" : "Y"
            }
        },
        "nReturned" : 152,
        "executionTimeMillisEstimate" : 179,
        "works" : 153,
        "advanced" : 152,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 8,
        "restoreState" : 8,
        "isEOF" : 1,
        "docsExamined" : 152,
        "alreadyHasObj" : 0,
        "inputStage" : {
            "stage" : "IXSCAN",
            "nReturned" : 152,
            "executionTimeMillisEstimate" : 32,
            "works" : 153,
            "advanced" : 152,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 8,
            "restoreState" : 8,
            "isEOF" : 1,
            "keyPattern" : {
                "HCPCS_CD" : 1,
                "PSPS_NCH_PAYMENT_AMT" : 1
            },
            "indexName" : "HCPCS_CD_1_PSPS_NCH_PAYMENT_AMT_1",
            "isMultiKey" : false,
            "multiKeyPaths" : {
                "HCPCS_CD" : [ ],
                "PSPS_NCH_PAYMENT_AMT" : [ ]
            },
            "isUnique" : false,
            "isSparse" : false,
            "isPartial" : false,
            "indexVersion" : 2,
            "direction" : "forward",
            "indexBounds" : {
                "HCPCS_CD" : [
                    "[52648.0, 52648.0]"
                ],
                "PSPS_NCH_PAYMENT_AMT" : [
                    "[-inf.0, 100.0)"
                ]
            },
            "keysExamined" : 152,
            "seeks" : 1,
            "dupsTested" : 0,
            "dupsDropped" : 0
        }
    },
    "allPlansExecution" : [ ]
}
```

B.Q1 – Sem *Index*

```
> db.enderecos.find({"CITY":"Ponta Grossa","REGION":"PR" }).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 104428,
    "executionTimeMillis" : 39766,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 50092000,
    "executionStages" : {
        "stage" : "COLLSCAN",
        "filter" : {
            "$and" : [
                {
                    "CITY" : {
                        "$eq" : "Ponta Grossa"
                    }
                },
                {
                    "REGION" : {
                        "$eq" : "PR"
                    }
                }
            ]
        },
        "nReturned" : 104428,
        "executionTimeMillisEstimate" : 3202,
        "works" : 50092002,
        "advanced" : 104428,
        "needTime" : 49987573,
        "needYield" : 0,
        "saveState" : 391363,
        "restoreState" : 391363,
        "isEOF" : 1,
        "direction" : "forward",
        "docsExamined" : 50092000
    },
    "allPlansExecution" : [ ]
}
>
```

B.Q2 – Com *Index* Texto em REGION

```
> db.enderecos.find({"CITY":"Ponta Grossa", $text: {$search:"PR"}}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 104428,
    "executionTimeMillis" : 31366,
    "totalKeysExamined" : 4203822,
    "totalDocsExamined" : 8407644,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "CITY" : {
                "$eq" : "Ponta Grossa"
            }
        },
        "nReturned" : 104428,
        "executionTimeMillisEstimate" : 11781,
        "works" : 4203823,
        "advanced" : 104428,
        "needTime" : 4099394,
        "needYield" : 0,
        "saveState" : 32950,
        "restoreState" : 32950,
        "isEOF" : 1,
        "docsExamined" : 4203822,
        "alreadyHasObj" : 4203822,
        "inputStage" : {
            "stage" : "TEXT",
            "nReturned" : 4203822,
            "executionTimeMillisEstimate" : 11575,
            "works" : 4203823,
            "advanced" : 4203822,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 32950,
            "restoreState" : 32950,
            "isEOF" : 1,
            "indexPrefix" : {

            },
            "indexName" : "REGION_text",
            "parsedTextQuery" : {
                "terms" : [
                    "pr"
                ],
                "negatedTerms" : [ ],
                "phrases" : [ ],
                "negatedPhrases" : [ ]
            },
            "textIndexVersion" : 3,
            "inputStage" : {
                "stage" : "TEXT_MATCH",
                "nReturned" : 4203822,
                "executionTimeMillisEstimate" : 11567,
                "works" : 4203823,
                "advanced" : 4203822,
                "needTime" : 0,
                "needYield" : 0,
                "saveState" : 32950,
                "restoreState" : 32950,
                "isEOF" : 1,
                "docsRejected" : 0,
                "inputStage" : {
                    "stage" : "FETCH",
                    "nReturned" : 4203822,
                    "executionTimeMillisEstimate" : 11544,
                    "works" : 4203823,
                    "advanced" : 4203822,
                    "needTime" : 0,
                    "needYield" : 0,
                    "saveState" : 32950,
                    "restoreState" : 32950,
                    "isEOF" : 1,
                    "docsExamined" : 4203822,
                    "alreadyHasObj" : 0,
                    "inputStage" : {
                        "stage" : "OR",
                        "nReturned" : 4203822,
                        "executionTimeMillisEstimate" : 4098,
                        "works" : 4203823,
                        "advanced" : 4203822,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 32950,
                        "restoreState" : 32950,
                        "isEOF" : 1,
                        "dupsTested" : 4203822,
                        "dupsDropped" : 0,
                        "inputStage" : {
                            "stage" : "IXSCAN",
                            "nReturned" : 4203822,
                            "executionTimeMillisEstimate" : 3411,
                            "works" : 4203823,
                            "advanced" : 4203822,
                            "needTime" : 0,
                            "needYield" : 0,
                            "saveState" : 32950,
                            "restoreState" : 32950,
                            "isEOF" : 1,
                            "keyPattern" : {
                                "_fts" : "text",
                                "_ftsx" : 1
                            },
                            "indexName" : "REGION_text",
                            "isMultiKey" : false,
                            "isUnique" : false,
                            "isSparse" : false,
                            "isPartial" : false,
                            "indexVersion" : 2,
                            "direction" : "backward",
                            "indexBounds" : {

                            },
                            "keysExamined" : 4203822,
                            "seeks" : 1,
                            "dupsTested" : 0,
                            "dupsDropped" : 0
                        }
                    }
                }
            }
        }
    },
    "allPlansExecution" : [ ]
}
```

B.Q3 – Com *Index* Texto em CITY


```
> db.enderecos.find({ $text: {$search:"Ponta Grossa"},"REGION":"PR"}).explain(true).executionStats
{
    "executionSuccess" : true,
    "nReturned" : 104428,
    "executionTimeMillis" : 1465,
    "totalKeysExamined" : 263099,
    "totalDocsExamined" : 317342,
    "executionStages" : {
        "stage" : "FETCH",
        "filter" : {
            "REGION" : {
                "$eq" : "PR"
            }
        },
        "nReturned" : 104428,
        "executionTimeMillisEstimate" : 515,
        "works" : 263101,
        "advanced" : 104428,
        "needTime" : 158672,
        "needYield" : 0,
        "saveState" : 2061,
        "restoreState" : 2061,
        "isEOF" : 1,
        "docsExamined" : 158671,
        "alreadyHasObj" : 158671,
        "inputStage" : {
            "stage" : "TEXT",
            "nReturned" : 158671,
            "executionTimeMillisEstimate" : 509,
            "works" : 263101,
            "advanced" : 158671,
            "needTime" : 104429,
            "needYield" : 0,
            "saveState" : 2061,
            "restoreState" : 2061,
            "isEOF" : 1,
            "indexPrefix" : {

            },
            "indexName" : "CITY_text",
            "parsedTextQuery" : {
                "terms" : [
                    "grossa",
                    "ponta"
                ],
                "negatedTerms" : [ ],
                "phrases" : [ ],
                "negatedPhrases" : [ ]
            },
            "textIndexVersion" : 3,
            "inputStage" : {
                "stage" : "TEXT_MATCH",
                "nReturned" : 158671,
                "executionTimeMillisEstimate" : 509,
                "works" : 263101,
                "advanced" : 158671,
                "needTime" : 104429,
                "needYield" : 0,
                "saveState" : 2061,
                "restoreState" : 2061,
                "isEOF" : 1,
                "docsRejected" : 0,
                "inputStage" : {
                    "stage" : "FETCH",
                    "nReturned" : 158671,
                    "executionTimeMillisEstimate" : 509,
                    "works" : 263101,
                    "advanced" : 158671,
                    "needTime" : 104429,
                    "needYield" : 0,
                    "saveState" : 2061,
                    "restoreState" : 2061,
                    "isEOF" : 1,
                    "docsExamined" : 158671,
                    "alreadyHasObj" : 0,
                    "inputStage" : {
                        "stage" : "OR",
                        "nReturned" : 158671,
                        "executionTimeMillisEstimate" : 195,
                        "works" : 263101,
                        "advanced" : 158671,
                        "needTime" : 104429,
                        "needYield" : 0,
                        "saveState" : 2061,
                        "restoreState" : 2061,
                        "isEOF" : 1,
                        "dupsTested" : 263099,
                        "dupsDropped" : 104428,
                        "inputStages" : [
                            {
                                "stage" : "IXSCAN",
                                "nReturned" : 104428,
                                "executionTimeMillisEstimate" : 145,
                                "works" : 104429,
                                "advanced" : 104428,
                                "needTime" : 0,
                                "needYield" : 0,
                                "saveState" : 2061,
                                "restoreState" : 2061,
                                "isEOF" : 1,
                                "keyPattern" : {
                                    "_fts" : "text",
                                    "_ftsx" : 1
                                },
                                "indexName" : "CITY_text",
                                "isMultiKey" : true,
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "backward",
                                "indexBounds" : {

                                },
                                "keysExamined" : 104428,
                                "seeks" : 1,
                                "dupsTested" : 104428,
                                "dupsDropped" : 0
                            },
                            {
                                "stage" : "IXSCAN",
                                "nReturned" : 158671,
                                "executionTimeMillisEstimate" : 46,
                                "works" : 158672,
                                "advanced" : 158671,
                                "needTime" : 0,
                                "needYield" : 0,
                                "saveState" : 2061,
                                "restoreState" : 2061,
                                "isEOF" : 1,
                                "keyPattern" : {
                                    "_fts" : "text",
                                    "_ftsx" : 1
                                },
                                "indexName" : "CITY_text",
                                "isMultiKey" : true,
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "backward",
                                "indexBounds" : {

                                },
                                "keysExamined" : 158671,
                                "seeks" : 1,
                                "dupsTested" : 158671,
                                "dupsDropped" : 0
                            }
                        ]
                    }
                }
            }
        }
    },
    "allPlansExecution" : [ ]
}
>
```