Banco de dados NoSQL

Abner S. Kaizer[1], **André M. S. Neves**[1],
Habacuque B. B. Neves[1], **Leonam T. Vasconcelos**[1]

[1]Centro Federal de Educação Tecnológica de Minas Gerais – campus Timóteo (CEFET)
35180-008 – Timóteo – MG – Brasil

Abstract. *This paper aims to show a little fraction of the so called NoSQL database, a not so new technology that has not been used a lot if compared to the traditional relational databases. In this article, the difference of implementations, performance and usage will be demonstrated. Also some advantages and disadvantages of each technology are going to be shown, just to make sure the comparison is fair. Moreover, a benchmark test will be analized in the end of this paper, for a proper demonstration of the differences between the technologies showed in the text.*

Resumo. *Este artigo tem como objetivo mostrar uma pequena fração dos assim chamados banco de dados NoSQL, tecnologia que, apesar de não ser muito recente, possuía pouca utilização, quando comparada aos tradicionais bancos de dados relacionais. Nesse trabalho, as diferenças de implementação, performance e uso entre ambos os paradigmas serão demonstradas, juntamente com suas vantagens e desvantagens em algumas possíveis aplicações. Além disso, um teste de desempenho será analisado no final deste material, para demonstrar propriamente as diferenças entre as tecnologias mostradas no texto.*

1. Lista de abreviaturas e siglas

SQL	*Structured Query Language* (Linguagem de Consulta Estruturada);
NoSQL	*Not only SQL* (Não apenas SQL);
SGBD	Sistema de Gerencialmento de Banco de Dados;
IDC	International Data Coporation;
ACID	*Atomicity, Consistency, Isolation and Durability* (Atomicidade, Consistência, Isolameno e Durabilidade);
CAP	*Consistency, Availability and Partition tolerance* (Consistência, Disponibilidade e Tolerância a partição);
BASE	*Basically Available, Soft state, Eventual consistency* (Eventualmente Disponíıvel, Estado Suave e Consistência Eventual);
AWS	*Amazon Web Services*;
Redis	*Remote Dictionary Server*;
SDK	*Software Development Kit* (Kit de Desenvolvimento de Software).

2. Introdução

A cada dia que se passa, a ciência e engenharia de computação estão cada vez mais avançadas, com um contínuo e incessante desenvolvimento de modelagens e técnicas inovadoras capazes de realizar tarefas cada vez mais complexas e massivas. Com uma taxa

de evolução e crescimento exorbitante quando comparada com outros ramos da ciência, a computação foi capaz de sair de um dispositivo de 15m^2 de área ocupada, com 17468 válvulas (o ENIAC, *Electronic Numerical Integrator and Computer*, em português, computador integrador numérico eletrônico) [Bernard Cohen et al. 1990] para microprocessadores com bilhões de transistores que cabem na palma da mão em curto período.

Tal ascensão científica permitiu que cientistas desenvolvessem aplicações gigantescas, disponíveis para todo o mundo através da internet, como sites de venda (e.g., Amazon, Ebay), grandes ferramentas de busca (e.g., Google, DuckDuckGo), redes sociais (e.g., Instagram, Reddit), entre outros. Contudo, atrelada à essa evolução computacional, aumentou-se também o volume de dados processados e trafegados nas grandes aplicações, com uma previsão para aumentar cada vez mais, em escala exponencial. Em relação a esse cenário, foi publicado pela *International Data Corporation (IDC)* um relatório desenvolvido por [Reinsel et al. 2018], o qual afirma que em 2018 os dados globais atingiram 33 *Zettabytes* ($33 \times 10^{12}GB$), e que há uma previsão de que atingirão 127 *Zettabytes* ($127 \times 10^{12}GB$) em 2025. Daí surge o termo **BigData**, que vem sido bastante comentado e estudado nos últimos anos. Segundo Lincon Neves e Evandro Jardini [Neves and Jardini], BigData é todo aquele volume de dados que está além da capacidade da maioria das aplicações comuns. Junto com o crescimento da quantidade de dados com a qual lidamos, cresceu-se também a necessidade de desenvolver novas ferramentas capazes de lidar com esta imensa quantidade de dados.

Para atender a essa demanda, novos paradigmas e conceitos foram teorizados e desenvolvidos, para que tamanho volume de informação fosse tratado com eficiência. Por conseguinte, foi iniciado um movimento denominado **NoSQL** (*Not only SQL* - Não apenas SQL), um termo genérico usado pela primeira vez em 1998 [Marcos Silva and Bellio Nascimento 2014], com a intenção de referir-se a qualquer sistema ou processo de armazenamento de dados que não segue o tradicional modelo de bancos de dados relacionais [Razu Ahmed et al. 2018]. Com isso, esse paradigma proporciona e propõe novas técnicas satisfatórias para manipulação de dados em massa, e segundo [Vieira et al. 2014], essa filosofia e/ou modelo foram criados com o intuito inicial de solucionar problemas de aplicações modernas, como escalabilidade.

2.1. Justificativa

Por utilizar uma representação conveniente e compatível com as informações existente no mundo real, o modelo relacional tem se mostrado eficiente para armazenamento de dados estruturados desde a sua concepção em 1970 [Berg et al. 2012], juntamente com as tecnologias de SGBD's (Sistemas de gerenciamento de banco de dados) que promovem recursos que garantem a consistência prometida pela definição de esquemas. No entanto, devido ao acelerado crescimento da quantidade e fluxo de informações, os SGBD's relacionais demonstraram deficiência ou falta de suporte a alguns recursos que tornaram-se essenciais para garantir desempenho das aplicações [Marcos Silva and Bellio Nascimento 2014]. Segundo [Gomes and Basso 2015], algumas dessas características são:

- Escalabilidade horizontal: Com o crescimento da quantidade e fluxo de dados armazenados, torna-se necessário aumentar os recursos computacionais que sejam

capazes de lidar com essa demanda. Isso pode ser realizado por meio da escalabilidade vertical, processo no qual os recursos de *hardware* são otimizados.
Contudo, esse artifício é limitado e torna-se cada vez mais caro ter um hardware com maior potencial. Para isso, é utilizado o conceito de **escalabilidade horizontal**, processo que realiza a divisão da carga de trabalho em distintos nós de processamento, de forma a garantir um desempenho aceitável nas operações.
No entanto, os bancos de dados relacionais não foram projetados para essa operação, que torna-se complexa nesse contexto por causa da estrita exigência de transações e consistência de dados. Essa característica traz junto uma necessidade de concorrência entre operações no banco, de forma que a utilização de escalabilidade horizontal eficiente é dificultada.

- Ausência de esquema: Nos bancos de dados *NoSQL*, não é obrigatória a definição de um padrão para estrutura dos dados, assim como não há definições de integridades referenciais ou restrições de consistência. Isso facilita o desenvolvimento de aplicações mais dinâmicas, que lidam com dados de variados formatos (livre de um esquema rígido), além de tornar mais simples o processo de escalabilidade horizontal, viabilizada pela tolerância a partições dos dados armazenados.

- Consistência eventual: Os bancos de dados relacionais tradicionais tem como característica geral garantir consistência de acordo com as especificações do modelo do esquema, o que inclui verificações de integridade referencial e restrições de valores.
Na tentativa de ter os dados sempre disponíveis simultaneamente com informações distribuídas horizontalmente, não é possível garantir consistência em tempo integral, pois por alguma falha ou atraso de comunicação, os nós de processamento podem ter versões diferentes dos dados.

2.1.1. Big Data

Big Data são grande volumes de dados pouco estruturados, variados e que são gerados em grande velocidade, inclusive em tempo real. Podem consistir nos dados coletados de redes sociais ou dispositivos internet das coisas por exemplo [Oracle Brasil 2019]. Observa-se que este tipo de dado é melhor armazenado em bancos de dados *NoSql*, pois terão a velocidade, escalabilidade e maleabilidade necessárias.

Contudo, a partir dessa imensa quantidade de dados, é necessário retirar conclusões, isto é, gerar modelos da realidade que ajudarão a solucionar problemas, o que chama-se *Data Mining* [Two Crows Consulting 2019]. Nesse quesito além do armazenamento, o recursos oferecidos por bancos de dados *NoSql* também pode facilitar esta análise, como por exemplo, um banco orientado a grafos seria adequado para facilitar algumas análises de interações.

Apesar de serem claramente melhores para alguns casos, *NoSql* pode competir com o SGBDs tradicionais quando trata-se de *Data Warehouse*. Esse conceito envolve uma arquitetura envolvida em agregar dados já preparados e contextualizados para análise [Oracle 2019]. Quando leva-se em conta que o *data warehouse* em si envolve dados

organizados, os SGBDs tradicionais ganham espaço, até porque são usadas técnicas de *clustering* para aumentar a eficiência.

Com isso, percebe-se que os bancos não relacionais tem uma forte ligação com *Big Data*, que é algo de extrema importância e utilização e que representando uma das necessidades que vieram a tornar os bancos *NoSql* populares.

2.2. Fundamentação teórica

As explicações a respeito das tecnologias *NoSQL* são fundamentadas em três conceitos principais: modelo **ACID**, modelo **BASE** e o teorema **CAP**. O modelo ACID é um paradigma seguido por SGBD's (Sistemas de gerenciamento de banco de dados) tradicionais, que significa a garantia de Atomicidade, Consistência, Integridade e Disponibilidade dos dados. O teorema CAP consiste em uma hipótese realizada sobre o sistemas distribuídos. O modelo BASE, por sua vez, surgiu a partir do ACID, por meio da modificação de alguns de seus pilares.

2.2.1. Modelo ACID

ACID é a sigla para *Atomicity*, *Consistency*, *Isolation* e *Durability* (i.e., atomicidade, consistência, isolação e durabilidade). De acordo com [Molina Toth], os SGBD's relacionais se baseiam no modelo *ACID*, para garantirem que as transações sejam feitas com segurança, que é obtida quando existe atomicidade, consistência, isolamento e integridade das operações. Contudo, para que possamos desfrutar destes recursos, são utilizados algoritmos que executam operações complexas sobre os dados gerenciados, o que leva, em alguns casos, à perda de desempenho.

O modelo *ACID*, apesar da possível perda de desempenho, como relatado anteriormente, funciona muito bem em sistemas monolíticos, isso é, todo o banco de dados está concentrado em uma única máquina. Em contextos de sistemas distribuídos, obter a coordenação e organização dos dados à ponto de manter integralmente as propriedades postuladas para o modelo em questão torna-se proibitivo, pois para isso teríamos que assegurar que:

1. Todo nó da rede deve ter a mesma versão do dado;
2. Todo cliente que fizer uma solicitação deve encontrar ao menos uma cópia do dado requerido;
3. A base de dados mantêm seus dados mesmo se for instalado em máquinas diferentes.

2.2.2. Teorema CAP

Os tópicos acima se referem às propriedades de consistência (tópico 1), disponibilidade (tópico 2) e tolerância à partições (tópico 3) [Molina Toth]. Nesse contexto, com a necessidade de trabalhar com sistemas distribuídos, é considerado o teorema CAP, cujo nome é a sigla para *Consistency*, *Availability* e *Partition tolerance*, propriedades das quais o teorema trata. É feita a afirmativa de que *em uma rede sujeita a falhas de comunicação,*

é impossível que um sistema mantenha uma resposta consistente ou mesmo produza uma resposta para cada requisição ([Gilbert and Lynch], traduzido e adaptado). Portanto, com a necessidade cada vez maior da distribuição de sistemas, aplicar o modelo ACID para a gerência dos dados tende ao impossível.

De acordo com o teorema CAP, apenas duas das três propriedades são possíveis de serem garantidas integralmente durante o funcionamento de um sistema. Com isso, em um momento de instabilidade, uma das três é deixada de lado.

Cada implementação de banco de dados distribuído leva em conta esse teorema, e prioriza duas propriedades em detrimento da terceira. Dessa forma, existem as seguintes escolhas possíveis:

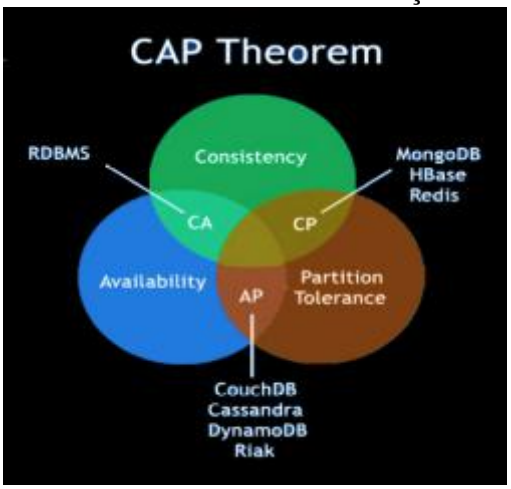
- Consistência e disponibilidade: Sistemas que priorizam essas propriedades necessitam abrir mão da tolerância à partições, pois na tentativa de mantê-la, em um momento de instabilidade da rede existem dois caminhos a serem seguidos:
  - Permitir inconsistência: Situação onde o conjunto distribuído poderá apresentar irregularidades nas informações, seja irregularidade lógica, ou seja pelo motivo de dois nós terem versões diferentes de alguns dados.
  - Tornar-se indisponível: Caso não seja permitida a consistência eventual, o sistema precisa estar indisponível até que a comunicação em rede seja estabelecida

  Como ambas as propriedades são prioridades, não é possível ter tolerância à partições.

- Consistência e tolerância à partição: Em caso de uma falha de rede, é priorizada a consistência dos dados distribuídos, de forma que o sistema força-se a estar indisponível.

- Disponibilidade e tolerância a partição: Em situações crítica, abre-se mão da consistência.

Figura 1. Diferentes bancos de dados em relação ao teorema CAP.



Fonte: [Esha Awasthi et al.]

2.2.3. Modelo BASE

Com base no modelo *ACID* e com respeito às premissas do teorema CAP, surge um novo modelo: o *BASE* (*Basically Available, Soft state, Eventual consistency* - eventualmente disponível, estado suave e consistência eventual), que foi desenvolvido para distribuição de sistemas com propriedades de tolerância à falhas de consistência e escalabilidade [Molina Toth].

Tomando como referência o modelo BASE, surgem os bancos de dados não relacionais – NoSQL. Como no modelo apresentado, a consistência não é garantida da mesma forma que acontece no modelo ACID. Mas é possível manter particionamento dos dados para sistemas distribuídos [Molina Toth], o que possibilita que quantidades de dados gigantescas sejam processadas e analisadas.

Segundo [MongoDB for absolute beginners], a utilização do modelo *BASE* tem as seguintes implicações, causadas por três características :

- Eventualmente disponível - Uma requisição pode resultar em falha (indisponibilidade), porém na menor parte das vezes;
- *Soft state* (Estado suave) - Em um contexto de estado estrito, o estado do sistema só pode ser alterado caso haja manipulação do banco de dados. Esse é o caso dos bancos de dados relacionais: Ao inserir, modificar ou excluir um registro, todas as operações são transacionais, o que envolve verificação de consistência perante ao esquema e replicação dos dados em todos os nós distribuídos, caso haja. Todas as operações devem ser executadas antes da a resposta de confirmação ser enviada.
  Pela característica do *soft state*, isso não ocorre, pois prioriza-se a eficiência das operações em um ambiente de alta demanda. Com isso, o sistema pode tornar-se inconsistente após manipulações de usuários, mas após isso, poderá novamente retornar ao estado de consistência.
  Isso viola a característica do estado estrito, citada anteriormente, na qual o estado do sistema é alterado sem intervenção do usuário.
- Consistência eventual: Pela existência do *soft state*, o sistema poderá tornar-se inconsistente em algum momento de alta demanda, no sentido de que os nós de processamento estarão dessincronizados. Contudo, retornará à consistência em um momento posterior, quando os dados puderem ser propagados.

3. Desenvolvimento

Atualmente temos uma ampla variedade de ferramentas NoSQL no mercado, como MongoDB, Apache CouchDB, Hadoop, Amazon SimpleDB, Redis, Neo4J, entre outros [Vieira et al. 2014]. Tais ferramentas se diferenciam, entre outros aspectos, pela forma como mapeiam e manipulam dados: existem sistemas baseados em documentos (e.g., MongoDB, Apache CouchDB), baseados em Coluna (e.g., Hadoop, Amazon SimpleDB), alguns baseados em *chave-valor* (e.g., Redis) e outros baseados em grafos como o Neo4J.

Cada uma das estruturas acima apresentadas possuem seus respectivos desempenhos baseados no contexto e no tipo de dados e problemas com o qual estão lidando.

3.1. Tipos de Bancos de Dados NoSql

Numa classificação mais geral, os bancos NoSql podem ser divididos entre orientados a agregação e não orientados a agregação [Ganesh Chandra 2015]. Agregação é o recurso

que existe para que o banco de dados compute informações a partir de dados primitivos, como por exemplo, o valor total de uma compra [MongoDb Docs 2019a].

Os tipos de bancos *NoSql* que possuem agregação são chave-valor, coluna e baseado em documentos. Os bancos *NoSql* orientados a grafos não possuem agregação, contudo são mais compatíveis com o modelo *ACID* e são ideais para consultas complexas.[Ganesh Chandra 2015]

Os bancos de dados NoSql também podem ser divididos sob a perspectiva do teorema *CAP*, onde diferentes software seguem um subconjunto de duas das três características evidenciadas pelo teorema, conforme mostrado na figura 1.

3.1.1. Chave-Valor

Este tipo de banco armazena os dados sem uma definição de esquema, mas armazena cada registro associados a uma chave. Geralmente são operados por comandos simples de recuperação e inserção, de forma que não permitem consultas avançadas. Em alguns cenários, é necessário ter redundância de dados. Contudo, apresentam alto desempenho e disponibilidade, devido a sua natureza redundante [Molina Toth].

Os bancos baseados em pares de chave e valor são vantajosos para situações onde o armazenamento é simples, como por exemplo, por ser usado como *cache* de aplicações *web* ou como memória compartilhada de nós de processamento, que trabalham concorrentemente. A maioria desses bancos é executada na RAM (*Random Access Memory* - Memória de acesso randômico), de forma que é possível aproveitar melhor sua simplicidade de armazenamento e atingir velocidades ainda maiores [AWS Amazon 2019b].

Dentre alguns exemplos desses bancos, está o *Amazon DynamoDb*, *Apache Cassandra* e *Redis* [AWS Amazon 2019b].

3.1.2. Orientado a Documentos

Documentos são estruturas de dados que possuem o formato *XML* (*Extensible Markup Language* - Linguagem de marcação extendível) ou *JSON* (*Javascript Object Notation* - Notação de objetos javascript), que são caracterizadas pela flexibilidade de estruturação de dados lineares e/ou hierárquicos [Molina Toth].

Diferentemente dos bancos de dados relacionais, não existe esquema rígido, mas é utilizada uma estrutura que respeite a semânticas dos dados armazenados. É o modelo ideal para sistemas onde os haverão dados independentes estruturalmente, que possuem diferentes atributos. Em contrapartida com o modelo relacional, apresenta também a vantagem de desnormalização estruturada, onde um único registro pode conter atributos multivalorados e compostos sem que semântica da estrutura seja prejudicada. Nessa situação onde os dados estão localizados em um único registro, existe maior agilidade para recuperação de determinadas informações.

Geralmente este tipo de banco possui maior facilidade de integração com *software*, principalmente se é seguido o modelo de orientação a objetos, pois existe uma maior compatibilidade de estrutura de dados, quando comparado ao modelo relacional.

Alguns exemplos deste tipo de banco são: *MongoDb*, *Apache CouchDB* e *Google Cloud FireStore* [AWS Amazon 2019a].

3.1.3. Orientado a Grafos

Nesse tipo de banco, existem obrigatoriamente nós (entidades) que podem possuir rótulos (identificações do nó) e atributos (dados da entidade). Com base no modelo matemático de grafos, são armazenadas as relações entre nós, o que caracteriza informações semântica de como os dados estão relacionados [Neo4j Docs 2019a, Sachdeva et al. 2018]

Apesar de representar as relações entre os dados, difere dos bancos relacionais por não utilizar tabelas, e sim um modelo simplificado. Sendo assim, é ideal para aplicações onde dados possuem um grande volume de interconexões, o que permite realizar consultas complexas, de forma que seriam inviáveis de serem executadas em bancos relacionais tradicionais. [De et al.]

Esse tipo de banco de dados é recomendado para aplicações de redes sociais, mineração de dados, algoritmos de recomendação, inteligência artificial , dentre outros [Neo4j Docs 2019b].

Alguns exemplos desse tipo de banco são: *Neo4j*, *JanusGraph* e *Amazon Neptune*[Db Engines 2019].

3.1.4. Coluna

Os bancos de dados NoSQL do tipo coluna são reconhecidos por indexarem as informações na forma de uma tripla (coluna,linha,timestamp), de forma que chaves apontam para atributos ou colunas múltiplas. Na tripla, as colunas e linhas são identificadas por chaves e o *timestamp* permite a diferenciação entre versões de um mesmo dado.

Sua estrutura assemelha-se à forma de organização em tabelas de um banco relacional, porém é otimizado para que as informações sejam organizadas em colunas, ao invés de linhas. Os bancos classificados nesse modelo foram influenciados pelo *Big-Table* da *Google* [Chang et al. 2008].

A diferença entre este modelo e o modelo relacional é que a organização é feita em arquiteturas distribuídas massivamente, e não no formato de tabelas. No armazenamento em colunas, cada chave está associada a um ou mais atributos (colunas) e as informações ficam guardadas de um modo tal que as informações possam ser agregadas rapidamente com menor atividade de E/S (Entrada e saída) [Nayak et al. 2013].

Estes bancos encorajam o uso de desnormalização por meio da duplicação de dados. Com isso, é possível obter um ganho na velocidade de leitura, pois não são utilizados relacionamentos que demandam um trabalho de reconstituição dos dados [Fowler 2015], mas existe perca de desempenho na alteração dos dados [Nayak et al. 2013].

Alguns exemplos de bancos com está característica são: *Cassandra*, *MariaDB* e *HBase*.

3.2. Alguns Dos Principais SBGD's de Bancos de Dados Nosql

3.2.1. Redis

Redis(REmote DIctionary Server) é um banco de código aberto do tipo chave-valor que executa na memória RAM , escrito em C e otimizado para performance [Redis Lab 2019].

Possui sua própria linguagem de acesso (Domain Specific Language ou DSL), que permite atomicidade e consistência. Também suporta vários tipos de dados, sendo que uma chave pode ser de qualquer um desses tipos.

Especificamente projetado para simplicidade de uso, pode até mesmo ter um microcontrolador como cliente, além de permitir o uso de *scripts* da linguagem Lua no lado do servidor.

Apesar de ser executado na RAM, possui recursos de segurança, o que permite: Replicação de dados de forma assíncrona, Persistência de dados no disco, deleção automática de dados na RAM, clusterização e criptografia de comunicação entre cliente e servidor.

Redis é modular, isto é, permite o desenvolver escrever extensões para o banco, e inclusive é possível implementar novos tipos de dados.

3.2.2. Firestore (Firebase)

É um serviço de banco de dados orientado a documentos oferecido pelo *Google*, que faz parte das tecnologias *Firebase* [Google Firebase 2019],

Tem intenção de ser eficiente e de fácil uso por parte dos programadores, pois inclui vários recursos úteis e importantes, como: Segurança, cache offline e sincronização em tempo real. Possui *SDKs* (*Software development kit* - Kit de desenvolvimento de software) e biblioteca escritas várias linguagens para vários dispositivos diferentes, tanto para aplicações de cliente como servidor, o que permite um fácil acesso e manipulação do banco de dados.

3.2.3. MongoDb

O MongoDb é uma base de dados orientada a documentos do tipo *JSON*, que oferece dinamicidade na estrutura interna para que a mesma seja alterada durante a execução da aplicação [MongoDb Docs 2019b]. Foi primeiramente lançado em 2009, e a ferramenta tinha sido planejada para experimentar uma nova forma de manipulação e gerenciamento de dados distribuída, não relacional e baseada na arquitetura de documentos [1]. A base de dados é gratuita para o uso e de código aberto, com todo seu código fonte disponível em seu repositório oficial no *github* [2].

Segundo o próprio site oficial, o MongoDb se encaixa no grupo de base de dados **horizontalmente escaláveis**, ou seja, com a arquitetura e os ajustes corretos, podemos

[1] `https://www.mongodb.com/evolved`
[2] `https://github.com/mongodb/mongo`

elevar o desempenho da base de dados aumentando o números de nós de processamentos, ou máquinas na rede, para que assim o sistema se distribua.

Ainda de acordo com o site oficial, o banco de dados é considerado o primeiro a implementar transações ACID multi-documento. Tal funcionalidade garante que possamos realizar *queries* ou *mutations* (alterações no banco – inserção, deleção ou atualização) em vários documentos diferentes sob uma transação, o que mantém as restrições de consistência e integridade de dados do modelo *ACID*.

Assim como boa parte das bases de dados relacionais clássicas, o Mongo permite que vários usuários modifiquem tuplas de uma mesma coleção, de forma a permitir concorrência a nível de documentos baseada em *locks*. Isso impede da realização de uma ação por um grupo de usuários enquanto um procedimento ou rotina é executado. O SGBD também suporta funções de agregação na *query* como agrupamentos, além de possuir no próprio sistema a implementação do algoritmo de *MapReduce*, muito utilizado em análise de dados.

3.3. Estudo de caso

Para uma pequena, porém concisa demonstração, foram realizados alguns testes comparativos no estilo *benchmark* nos bancos de dados *MySQL* e *MongoDB*, para avaliar o desempenho dos *SGBDs* em consultas simples e complexas.

Para garantir a justiça e o equilíbrio dos testes, algumas consultas foram realizadas várias vezes, para que assim possamos tirar a média aritmética dos tempos finais. Dessa forma, foram evitados quaisquer desvio de dados temporais provocados pelo estado da máquina na qual os testes serão executados.

Os *SGDBs* foram povoados a partir de uma tabela (arquivo csv contendo os dados não normalizados) que contém a lista de participantes do ENEM (Exame Nacional do Ensino Médio) de 2017 [Portal INEP 2019][3], a qual possui diversas informações, como:

- Informações sobre condições especiais:
  - Surdez;
  - Déficit de atenção;
  - Dislexia;
  - Discalculia;
  - Entre outras.
- Informações sobre atendimento especializado:
  - Prova em *Braille*;
  - Prova ampliada;
  - Ledor;
  - Auxílio para transcrição;
  - Utilização de computador;

[3] `http://portal.inep.gov.br/microdados`

  - Sala individual;
  - Sala especial com até 20 participantes;
  - Entre outras.
- Situação da conclusão do Ensino Médio;
- Nota em cada uma das provas;
- Respostas sobre o questionário sócio-econômico;

Em ambos os SGBD's, os dados foram normalizados da maneira mais estrita possível. No entanto, no *MongoDb*, foram aproveitados os recursos de atributos compostos e multivalorados em situações for semanticamente compatíveis.

Foram realizados testes com consultas simples e complexas, de forma que envolva a explorar e avaliar a qualidade dos recursos oferecidos pelos SGBD's para tais operações, como índices e agregação.

O objetivo dos testes foi mostrar as diferenças de desempenho que obtemos em alguns cenários específicos, ao escolher a ferramenta mais apropriada para atuar nos mesmos. Contudo, o resultado final independe dos números obtidos. Temos como o principal foco, demonstrar como os tempos das buscas crescem à medida que a quantidade de dados cresce nos bancos relacionais (representados pelo MySQL) em comparação com a variação dos tempos de busca na base de dados em bancos NoSQL, sendo representados pelo MongoDb.

3.3.1. Análise de desempenho

Foram definidas 4 *queries* para realização dos testes:

- Para cada UF, nota média de cada uma das provas objetivas;
- Para cada UF e estado civil, nota máxima geral dos alunos;
- Agrupar alunos por quantidades de condições especiais (deficiência, situação ou requerimento especial) e exibir a nota média de cada grupo;
- Para cada estado, nota média em matemética dos alunos com discalculia.

Ao realizarmos as consultas nos bancos importados, capturamos os tempos obtidos nas 4 *queries*. Os resultados estão dispostos na tabela abaixo:

Tabela 2. Tabela de resultados contendo a média dos tempos de 4 execuções da mesma query.

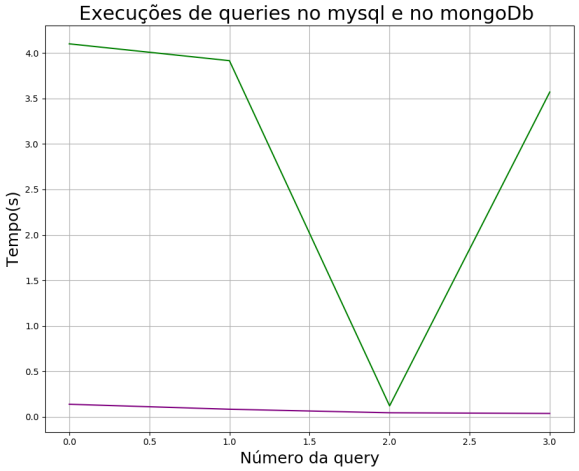
Query	Mysql	Mongo
1	0.13865	4.10000
2	0.083975	3.915
3	0.0451	0.121
4	0.037725	3.5700000

Fonte: autores.

Para melhor visualização dos dados da tabela 2, tomamos o gráfico abaixo:

Pela simples análise dos resultados acima, podemos claramente perceber uma informação importante: Por mais que as bases de dados *NoSQL* são aclamadas pelo

Figura 2. Gráfico dos resultados das consultas nos bancos de dados. Onde a curva roxa representa os tempos de consulta no MySQL e a curva verde, os tempos de consulta no MongoDb.



Fonte: autores.

magnífico desempenho, descrito em diversos artigos e matérias, como mostrado nos trabalhos de [Araujo Solagna and Lazzaretti] e [Gomes and Basso 2015], em alguns casos o clássico banco de dados relacional se sobressai, como foi o caso do teste executado nesse artigo.

A base de dados utilizada tinha cerca de 60000 tuplas na tabela de participantes, porém os dados ainda não chegam nem perto da quantidade absurda de informação processada pelos grandes centros de processamento de dados.

4. Considerações Finais

Os bancos de dados NoSQL constituem um ferramental muito importante para o desenvolvimento da computação e para o tão falado BigData. As necessidades computacionais de hoje em dia demandam que os dados possam ser manipulados de maneira eficiente e escalável, isto é, podendo ser distribuídos entre várias máquinas ou nós de processamentos. Tal demanda, pode ser satisfeita caso a ferramenta correta seja implementada no ambiente apropriado sob as condições favoráveis.

Contudo, por último mas não menos importante, é necessário constatarmos que o paradigma NoSQL não substitui o paradigma relacional. Como no próprio teste prático pôde-se visualizar, nem sempre um SGBD NoSQL irá ter um desempenho melhor que um SGBD Relacional, seja por implementação inconsistente, seja por estarmos lidando ainda com uma quantidade de dados irrisória, se comparado ao volume de dados no qual a nova tecnologia foi projetada para trabalhar.

Assim como em muitos algoritmos complexos, o crescimento matemático do tempo de complexidade das operações realizadas pelos mecanismos de busca de ambos os paradigmas principais analisados pelo teste prático têm uma discrepância mais acentuada quando as dimensões dos dados e das estruturas envolvidas aumentam.

Referências

[Araujo Solagna and Lazzaretti] Araujo Solagna, E. and Lazzaretti, A. T. UM ESTUDO COMPARATIVO ENTRE O MONGODB E O POSTGRESQL 1. Technical report.

[AWS Amazon 2019a] AWS Amazon (2019a). What Is a Document Database? Disponível em: https://aws.amazon.com/nosql/document/. Acesso em: 09/06/2019.

[AWS Amazon 2019b] AWS Amazon (2019b). What Is a Key-Value Database? Disponível em: https://aws.amazon.com/nosql/key-value/. Acesso em: 09/06/2019.

[Berg et al. 2012] Berg, K. L., Seymour, T., and Goel, R. (2012). History Of Databases. *International Journal of Management & Information Systems (IJMIS)*, 17(1):29.

[Bernard Cohen et al. 1990] Bernard Cohen, I., Aspray, W., Editorial Board, e., Galler, B., Randell, B., Williams, M., Zemanek, H., Bashe, C. J., Johnson, L. R., Palmer, J. H., and Pugh, E. W. (1990). John von Neumann and the Origins of Modern Computing. Technical report.

[Db Engines 2019] Db Engines (2019). DB-Engines Ranking - popularity ranking of graph DBMS. Disponível em: https://db-engines.com/en/ranking/graph+dbms. Acesso em: 09/06/2019.

[De et al.] De, T., Cypriano De Souza, P., Bernardes De Paula, L., Augusto, M., Santos, S., and Rothenberg, C. E. Modelos Semânticos em Bancos de Dados Baseados em Grafos para Aplicações de Controle de Redes Definidas por Software. Technical report.

[Esha Awasthi et al.] Esha Awasthi, Shikha Agrawal, and Rajeev Pandey. A SURVEY ON NOSQL AND NEWSQL DATA STORES FOR BIG DATA MANAGEMENT.

[Ganesh Chandra 2015] Ganesh Chandra, D. (2015). BASE analysis of NoSQL database. *Future Generation Computer Systems*, 52:13–21.

[Gilbert and Lynch] Gilbert, S. and Lynch, N. A. Perspectives on the CAP Theorem. Technical report.

[Gomes and Basso 2015] Gomes, B. C. K. and Basso, C. d. A. M. (2015). Desempenho De Banco De Dados Não Relacionais Com Big Data. TECSI.

[Google Firebase 2019] Google Firebase (2019). Firebase. Disponível em: https://firebase.google.com/?hl=pt-BR. Acesso em: 09/06/2019.

[Marcos Silva and Bellio Nascimento 2014] Marcos Silva, A. and Bellio Nascimento, M. (2014). MongoDB: Um Estudo Teórico-Prático do Conceito de Banco de Dados NoSQL - Trabalho de Diplomação. Technical report.

[Molina Toth] Molina Toth, R. Abordagem NoSQL – uma real alternativa. Technical report. Disponível em: https://dcomp.sor.ufscar.br/verdi/topicosCloud/nosql_artigo.pdf. Acesso em: 09/06/2019.

[MongoDb Docs 2019a] MongoDb Docs (2019a). Aggregation — MongoDB Manual. Disponível em: https://docs.mongodb.com/manual/aggregation/. Acesso em: 09/06/2019.

[MongoDb Docs 2019b] MongoDb Docs (2019b). What Is MongoDB? — MongoDB. Disponível em: https://www.mongodb.com/what-is-mongodb. Acesso em: 09/06/2019.

[MongoDB for absolute beginners] MongoDB for absolute beginners. MongoDB for absolute beginners: ACID and CAP. Disponível em: https://mongodbforabsolutebeginners.blogspot.com/2016/06/acid-and-cap-theroems.html . Acesso em: 09/06/2019.

[Nayak et al. 2013] Nayak, A., Poriya, A., and Poojary, D. (2013). Type of NOSQL Databases and its Comparison with Relational Databases. Technical Report 4.

[Neo4j Docs 2019a] Neo4j Docs (2019a). Data Modeling Concepts and Techniques — Neo4j. Disponível em: https://neo4j.com/developer/guide-data-modeling/ . Acesso em: 09/06/2019.

[Neo4j Docs 2019b] Neo4j Docs (2019b). Graph Database Use Cases and Solutions. Disponível em: https://neo4j.com/use-cases/ . Acesso em: 09/06/2019.

[Neves and Jardini] Neves, L. and Jardini, E. d. A. ESTUDO DE FERRAMENTAS DE BIG DATA PARA DEFINIÇÃO DE INDICADORES EM TOMADA DE DECISÃO. (1996):1–21.

[Oracle 2019] Oracle (2019). What Is a Data Warehouse. Disponível em: https://www.oracle.com/database/what-is-a-data-warehouse/. Acesso em: 09/06/2019.

[Oracle Brasil 2019] Oracle Brasil (2019). O que é Big Data? Disponível em: https://www.oracle.com/br/big-data/guide/what-is-big-data.html . Acesso em: 20/06/2019.

[Portal INEP 2019] Portal INEP (2019). Microdados - INEP. Disponível em: http://portal.inep.gov.br/microdados. Acesso em: 09/06/2019.

[Razu Ahmed et al. 2018] Razu Ahmed, M., Arifa Khatun, M., Asraf Ali, M., and Sundaraj, K. (2018). A literature review on NoSQL database for big data processing. *International Journal of Engineering & Technology*, 7(2):902.

[Redis Lab 2019] Redis Lab (2019). Redis — Redis Labs. Disponível em: https://redislabs.com/redis-features/redis. Acesso em: 09/06/2019.

[Reinsel et al. 2018] Reinsel, D., Gantz, J., and Rydning, J. (2018). The Digitization of the World From Edge to Core. Technical report.

[Sachdeva et al. 2018] Sachdeva, V., Sharma, S., Saini, P., and Mrinal Pandey, D. (2018). Comparative Study of RDBMS, NOSQL and Graph Databases. Technical report.

[Two Crows Consulting 2019] Two Crows Consulting (2019). About data mining. Disponível em: http://twocrows.com/data-mining/. Acesso em: 09/06/2019.

[Vieira et al. 2014] Vieira, M. R., Figueiredo, J. M. D., Liberatti, G., Fellipe, A., and Viebrantz, M. (2014). Bases de datos NoSQL, MongoDB y GIS. *MappingGIS*, (1):1–30.