Comparação de Performance de Processamento entre Bases de Dados Relacionais e Bases de Dados NoSql

Matheus Henrique da Silva Muniz[1], Ricardo Costa P Santos[2]

[1]Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas – Campus Juiz de Fora – MG – Brasil

[2]Orientador - Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas– Campus Juiz de Fora – MG – Brasil

matheushenrique-sm1995@hotmail.com, ricardo.cpsantos@gmail.com

Abstract. *This paper aims to compare performance between relational database and NoSQL database technologies, covering typically inserted issues within the relational context. The final goal is to create two equivalent databases: one relational and the other one into the NoSQL context, in order to develop a system that forwards the connection between these two bases, feeding them with the same data set, providing the execution of several operations and, therefore, the computational cost measurement, in order to create information which allows performance comparison between those two database architectures, supported by statistical methods.*

Resumo*. Este trabalho tem por objetivo fazer a comparação de performance entre as tecnologias de banco de dados Relacional e banco de dados NoSQL, com problemas tipicamente inseridos dentro do contexto relacional. O objetivo final desse trabalho é criar 2 bases de dados equivalentes: uma relacional e a outra dentro do contexto NoSQL, desenvolver um sistema que promova a conexão com essas 2 bases, alimentando-as com o mesmo conjunto de dados, proporcionando a execução de diversas operações e, consequentemente, a mensuração do custo computacional, de forma a gerar informações que permitam a comparação de desempenho entre as 2 arquiteturas de banco de dados, amparada por métodos estatísticos.*

1. Introdução

Bancos de dados relacionais foram desenvolvidos em meados da década de 70 com o propósito de serem mecanismos de persistência de dados, que através de um software (SGBD – Sistema Gerenciador de Banco de Dados) manipula as informações, armazenando-as de forma estruturada em tabelas, usando uma linguagem de consulta própria (SQL – Structured Query Language) [1,2]. Logo, a utilização de bases de dados relacionais dentro das grandes organizações passou a ser um fator crucial, tornando esta uma ferramenta utilizada em todo o mundo. Porém, estudos realizados pela IBM [3] indicam que, atualmente são gerados dados na ordem dos petabytes todos os dias e estima-se que a cada 18 meses esse número tende a dobrar, uma vez que, o modelo relacional possui uma série de limitações como, a perda de eficiência em consultas devido ao grande volume de dados e a análise desta informação. Esse aumento deve-se ao fato do crescimento do uso de tecnologias que compartilham informações entre si, como por exemplo, redes sociais e serviços de streaming. Esse conjunto de dados

complexos e demasiadamente grandes e que modelos tradicionais não conseguem tratar de uma forma muito eficiente, podem ser nomeados pelo termo “Big Data” [3].

Para tratar esses conjuntos de dados, surgem novas alternativas de modelos de banco, como por exemplo, softwares de bancos de dados NoSql, desenhados para suportar imensos volumes de dados.

Sendo assim, torna-se importante saber se essa nova tecnologia terá uma eficiência superior à Relacional, mesmo em problemas tipicamente relacionais, ou seja, onde as informações estão devidamente estruturadas, não contendo a variedade de dados normalmente encontrada em sistemas já desenvolvidos dentro do contexto do big data.

2. Bancos de dados NoSQL

O propósito da criação dos bancos de dados NoSQL foi para fornecer um conjunto de novos recursos de gerenciamento de dados e também superar algumas limitações presentes nos bancos de dados relacionais [4].

Com o objetivo de vencer essas limitações, o NoSQL se baseia no BASE(basicamente disponível, estado leve e eventualmente consistente) caracterizado por ter uma grande disponibilidade dos dados e umas das desvantagens desse modelo que é a falta de consistência devido ao fato de não possui esquema, diferente do modelo relacional que se baseia no princípio ACID (atomicidade, consistência, isolamento e durabilidade) o qual garante uma consistência forte mas gera uma escalabilidade mais difícil em comparação ao BASE [5] . De acordo com o teorema CAP (Consistência, disponibilidade e tolerância a particionamento) esses problemas existem devido ao fato que só é possível em um sistema distribuído ter dois dos três elementos desse teorema [6].

Quando comparados estes dois modelos de dados, nota-se que o modelo NoSQL possui uma maior flexibilidade por diversos motivos, como por exemplo, o fato de não possuir uma estrutura fixa, podendo os dados serem armazenados de forma não estruturada, semiestruturada ou estruturada, garantindo uma das principais vantagens em relação ao modelo relacional uma vez que, cerca de 80% de toda informação hoje gerada é não estruturada[7].

A utilização de Bases não relacionais contribui para a diminuição de gastos da empresa, pois a alta escalabilidade horizontal permite a utilização de diversos servidores com baixo custo, diferentemente do cenário visto no relacional que possui uma alta escalabilidade vertical, sendo necessário muitas das vezes a construção de mainframes com elevada capacidade de processamento [2, 8].

Atualmente os bancos de dados NoSQL são divididos em quatro grupos principais, cada um com um tipo específico de armazenamento [9], que são:

- Key-Value Store: este tipo de base de dados é o mais simples, consiste em uma chave com um valor atribuído a ela permitindo a sua visualização através de uma tabela hash, mas mesmo sendo o mais simples dentre os quatro é o que possui a maior escalabilidade.

- Document Store: este modelo em relação ao primeiro é mais complexo e utiliza como modo de armazenamento coleções e documentos através de formatos padronizados como o XML e o JSON.

- Column Family: este tipo de banco de dados foi criado com o objetivo de armazenar e processar um grande volume de dados distribuídos.

- Graph Database: este modelo utiliza três componentes, que seriam os nós, os relacionamentos e as propriedades, sendo cada nó ser conectado com diversas arestas. Ele torna-se muito útil quando é necessário realizar consultas muito complexas.

Para esse experimento foi escolhido o modelo "Document store", mais especificamente o MongoDB, pela facilidade de implementação, disponibilidade de ferramentas que auxiliam a otimização das bases de dados, suporte a diversas linguagens de programação através de drivers e também pelo formato de armazenamento em documentos ser muito semelhante a objetos, facilitando e muito a criação da camada de persistência no desenvolvimento de sistemas [10].

3. MongoDB

O MongoDB possui código aberto e fornece alta performance, armazenando dados em forma de documentos JSON. Este utiliza um Sistema de gerenciamento de banco de dados NoSQL do tipo "Document Store", que permite o agrupamento de todas informações de um determinado conjunto em um único documento. Outra característica do "Document Store" é a utilização de redundância (cópias de segurança) com a finalidade de atenuar o problema da inconsistência, proveniente da distribuição dos registros em cluters de computadores (quanto maior o número de computadores maior é a probabilidade de falha em algum ponto, mas caso haja alguma falha o impacto é menor, sendo mais passível a recuperação) [19,20]. Uma base de dados do MongoDB é composta por um conjunto desses documentos agrupados em forma de coleções [11].

Para o aumento da performance, o modelo orientado a documento (Document Store) utiliza a ferramenta "MapReduce" que a partir de métodos avançados possibilita uma maior eficiência no agrupamento e filtragem dos dados possibilitando a leitura /escrita dos dados de forma paralela [19]. O "MapReduce" é um modelo de programação criado para algoritmos paralelos e distribuídos buscando o processamento em massa de dados, usando o paradigma de programação funcional através das funções "Map" e "Reduce" [20,21].

A partir da tabela 1 pode-se observar uma comparação entre o MongoDB e o modelo relacional mostrando os tipos correspondentes.

Tabela 1. Comparação entre os tipos correspondentes em um banco de dados relacional e um banco de dados NoSQL orientado a documento: MongoDB

BD Relacional	MongoDB
Base de Dados	Base de Dados
Tabela	Coleção
Registro/ linha	Documento JSON
Coluna	Atributo

3.1. Documentos JSON

Documentos JSON utilizam um formato muito semelhante ao XML. Sua estrutura é delimitada por chaves e o seu interior possui um conjunto de pares, sendo cada par caracterizado com o nome do atributo cercado por aspas e o outro o valor estipulado a este, as possibilidades de atribuição são um número, uma cadeia de caracteres, um conjunto de valores (vetor) ou até mesmo outro documento JSON [11,12]. Para um conjunto de valores, este deve ser cercado por colchetes com os valores separados por vírgula. Esses detalhes podem ser observados na Figura 1.

```
{
    "id": 1,
    "nome": "Rodrigo",
    "cpf": "12345678912",
    "telefone": ["912345678", "912345679"]
}
```

Figura 1. Exemplo de um documento JSON

A falta de consistência como já mencionado é um dos problemas do NoSQL, no MongoDB um dos problemas está presente no armazenamento de um documento JSON, pois este não possui uma estrutura amarrada, sendo possível um registro quando inserido, conter campos não presentes ou campos escritos de forma errada em relação ao modelo conceitual. Outro possível problema ainda em relação ao Documento JSON está relacionado à tipagem de uma variável, que pode variar entre todos os possíveis tipos, sem acusar erros de inserção.

Para tratar a falta de consistência, os desenvolveres responsáveis por construir aplicações que se relacionam com MongoDB, necessitam de um maior cuidado na hora de fazer esses relacionamentos, o que implica, muita das vezes, em um custo mais alto no desenvolvimento. Outra alternativa é a utilização de um framework na linguagem em que o programada está baseado, para restringir os atributos (quais serão usados e seus respectivos nomes) e a tipo de cada um deles. Para o modelo relacional foi utilizado o banco de dados MySQL.

4. Operações do MongoDB e MySQL

O objetivo dessa seção é mostrar uma série de operações similares entre os dois modelos de bancos de dados (MongoDB e MySQL) [12,18].

Na tabela 2 foi feita uma comparação entre operações de criação de uma base de dados e de criação de uma tabela/coleção, com a diferença que no MongoDB não é definido uma estrutura e também a destruição de uma tabela/coleção e uma base de dados. No MongoDB a mesma operação de criação do banco de dados é a mesma utilizada para selecionar uma base de dados para o uso. Para deletar uma base de dados é necessário que esta esteja selecionada.

Tabela 2. Exemplos de operações de criação de base de dados, tabela e coleção e a destruição de uma tabela/coleção e uma base de dados

Operação	MySQL	MongoDB
criar a base de dados	**CREATE DATABASE** biblioteca;	**USE** biblioteca
Criar tabela/coleção	**CREATE TABLE** pessoa (cpf **NUMBER(11) PRIMARY KEY**, nome **VARCHAR(30) NOT NULL**, sexo **NUMBER(1) NOT NULL**);	**db.createCollection**("pessoa")
deletar tabela/coleção	**DROP TABLE** livro;	**db**.livro.**drop()**
deletar base de dados	**DROP DATABASE** biblioteca	**db.dropDatabase**()

Na tabela 3 foi feita uma comparação entre operandos [14,15] dos dois bancos de dados. Alguns desses operandos podem ser visto em exemplos da tabela 4.

Tabela 3. Comparação entre operandos das duas bases de dados

Operação	MySQL	MongoDB
Igual a	=	$eq
Menor que	>	$gt
Menor ou Igual	>=	$gte
Maior que	<	$lt
Maior ou Igual	<=	$lte
Diferente de	!= ou <>	$ne

Na tabela 4 foi feita uma comparação entre operações de inserção, alteração e destruição de registro.

Tabela 4. Exemplos de operações de inserção, alteração, destruição de registros e pesquisa

Operação	MySQL	MongoDB
Inserir registro	**INSERT INTO** pessoa **VALUES**(12345678912, 'rodrigo',1);)	**db**.pessoa.**insert**({ "cpf": 12345678912, nome: "Rodrigo", "sexo":1})
Alterar registro	**UPDATE** pessoa **SET** nome = "Rodrigo Silva" **WHERE** cpf=12345678912;	**db**.pessoa.**update**({ "cpf":12345678912}, {**$set**: {"nome": "Rodrigo Silva"}})
Deletar registro	**DELETE FROM** pessoa **WHERE** cpf=12345678912;	**db**.pessoa.**remove**({ "cpf": {**$gt**:12345678912}})
Pesquisar	**SELECT** * **FROM** pessoa **WHERE** sexo=1;	**db**.pessoa.**find**({"sexo":1})

No MongoDB o atributo usado como o identificador de um registro (chave primária) é o "_id", esse pode ser inserido junto do documento como um campo normal ou gerado através da função "ObjectId()", que pode ser uma string Hexadecimal de 24 Bytes, uma string de 12 bytes binária ou simplesmente um número[16].

5 – Metodologia

Foram criadas duas bases de dados para realização dos testes. Na figura 2 observa-se a modelagem do banco de dados relacional, enquanto na figura 3 é representado o modelo de inserção de registros em cada documento, pois MongoDB não apresenta uma estrutura fixa.

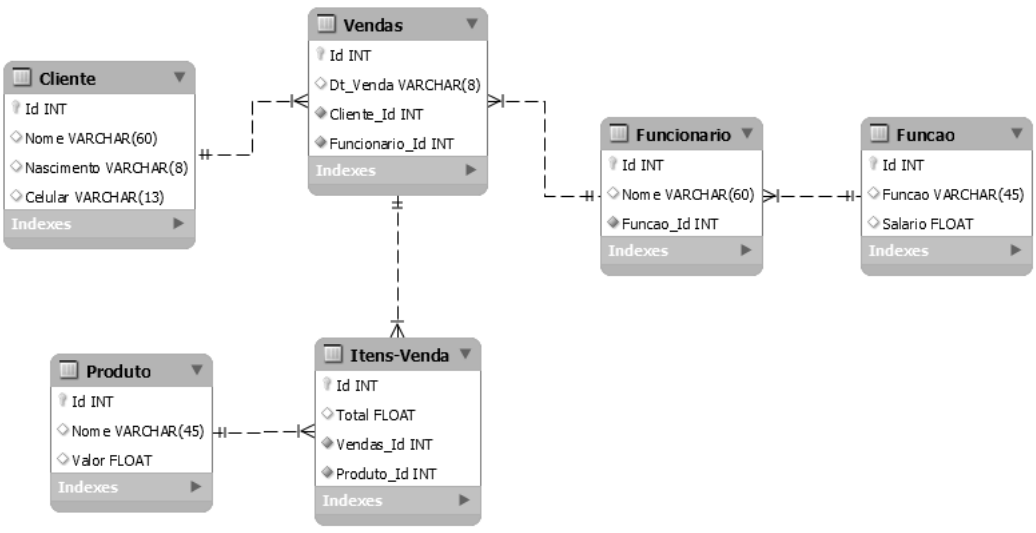

Figure 2. Modelagem do Banco de dados relacional.

```
db.cliente.insert({
      nome: 'Anderson Viera',
      nascimento: '20/08/1990',
      venda: [{
            dt_venda:'10/01/2016',
            funcionario_id: [ ObjectId('52ffc33cd85242f436000001')],
            itens:[
              {Obid: ObjectId('52ffc4a5d85242602e000000'),total:'20'},
              {Obid: ObjectId("52ffc4a5d85242602e000001"),total:'10'}
            ]
      }]
})
```

```
db.produto.insert({
      nome: 'borracha',
      valor: 0.25
})
```

```
db.funcao.insert({
      Funcao: 'vendedor',
      Salario: 1000
})
```

```
db.funcionario.insert({
      Nome: 'Adriano',
      Funcao_id: ObjectId("58482f682453ccad32a16aad")
})
```

Figure 3. Exemplos de inserção de registro no MongoDB.

O registro do "cliente" no MongoDB em comparação ao relacional contém também os registros das suas vendas que por sua vez contém os produtos de cada venda.

5.1 - Contextos utilizados para os testes.

Foram utilizados dois contextos para a realização dos testes. O primeiro consiste nas operações em cima de registros em que as dados já são estruturadas, ou seja, no modelo relacional é necessário apenas uma tabela para a sua representação, e consequentemente os testes consistiram em cima de uma única tabela/coleção. O segundo leva em consideração dados que não são estruturados e que precisam de mais de uma tabela para o armazenamento no relacional e também duas operações comuns usadas para análise dos dados, no MongoDB este pode ser armazenado também em várias coleções ou simplesmente em uma.

5.2 - Configuração

Para a realização desse experimento foi utilizado uma adaptação da metodologia empregada em YCSB (Yahoo Cloud Benchmark) [17] que consiste em medir performance através de diversos testes caracterizados como "Workloads" (conjunto de operações) separados em dois grupos (referente aos dois contextos já mencionados no tópico 5.1). Para realização desse experimento foram criados dois softwares, um responsável por gerar os dados tanto para o banco de dados relacional quanto para o MongoDB e outro para realizar todos os testes.

Para o primeiro grupo, foram criados os workloads de 1 a 7, sendo que cada um destes simula uma determinada situação, sendo executados na tabela/coleção cliente. Cada teste executa 1.000 operações podendo ser esta homogênea (um tipo) ou heterogênea (dois tipos). As possibilidades de tipos usadas são de leitura (read), alteração (update) e inserção (insert). Os worloads usados foram:

- Workload 1: Consiste em 50% de leitura e 50% de alteração;
- Workload 2: Consiste em 95% de leitura e 5% de alteração;
- Workload 3: Esse workload é 100% de leitura;
- Workload 4: Consiste em 95% de leitura e 5% de inserção;
- Workload 5: Consiste em 5% de leitura e 95% de alteração;

- Workload 6: Esse workload é 100% de alteração;

- Workload 7: Este workload é 100% de inserção;

O segundo grupo se restringe aos workloads de 8 a 11, e tem o objetivo de testar consultas que envolvem a união com outras tabelas/coleções e operações em cima desses resultados.

- Workload 8: O objetivo desse é realizar uma consulta com um nível de acoplamento entre duas tabelas/coleções com a restrição de mostrar as vendas realizadas pelo funcionário que possua uma determinada chave estrangeira/referência para tabela/coleção "função". Para o SQL foi utilizado um "Inner Join" entre as tabelas "venda" e "funcionario" e a condição através do "where". Para o MongoDB o processo consiste em duas etapas realizadas com ajuda da aplicação (visto que esse modelo não possui uma operação nativa semelhante ao "Inner Join").Na primeira é realizada uma recuperação dos funcionários com a referência específica que atende a restrição e o armazenamento desses em uma lista, na segunda etapa é realizado uma consulta na coleção "cliente" retornando as vendas efetuadas pelos funcionários que estão na lista gerada na primeira etapa.

- Wokload 9: É a implementação do anterior com mais um nível de acoplamento totalizando dois níveis entre três tabelas/coleções e a restrição baseada na variável "Salario" da tabela/coleção "funcão". No SQL a consulta possui um "Inner Join" entre as tabelas "venda" e "funcionario" e outra entre "funcionario" e "função" e a condição através do "where". Já no MongoDB foi necessário o uso de três etapas, na primeira é realizado uma busca das funções com um determinado salário e o armazenamento dessas em uma lista. Na segunda a busca é realizada na coleção "funcionario" que possuísse a sua função na lista gerada na primeira etapa e também o armazenamento desse resultado, a última é a recuperação dos registros de vendas na coleção "cliente" que tenham sido realizadas pelos funcionários retornados na segunda etapa.

- Workload 10: Consiste em executar uma única query. Para o relacional a consulta tem como objetivo de recuperar os registros dos clientes e a quantidade de produtos comprada por esse (uso da operação "Count()" em cima da chave estrangeira "Produto_Id" na tabela "Carrinho"), a execução foi feita a partir da tabela de venda fazendo dois "Inner Join" com as tabelas "Cliente" e "Carrinho" (um nível de profundidade em cada). No MongoDB a consulta similar foi realizada somente na coleção "cliente" pois esse já contém as vendas e produtos relativos a essa. O objetivo desse teste é averiguar uma das principais vantagens do MongoDB, que seria o armazenamento de registros dentro de registros.

- Workload 11: Parecido com o anterior, com a diferença de um nível de profundidade a mais no acoplamento das tabelas/coleções e também a inclusão de uma condição (mostrar registros de clientes que tenham comprado pelo menos um produto acima de um determinado valor) em cima da tabela "Produto". No MongoDB esse novo nível foi implementado em duas partes, a primeira parte consiste na recuperação dos registro dos Produtos que satisfaçam a condição "Acima de um determinado valor". A segunda parte é a recuperação dos registros de clientes que tenham comprado algum destes produtos. O objetivo desse teste é avaliar a vantagem do banco de dados relacional em cima

do MongoDB, pois o banco de dados relacional possui uma operação nativa (Inner Join) enquanto o outro não.

Em relação aos "worloads" 8 e 9 as consultas das vendas no MongoDB foram realizadas na coleção "cliente", pois os registros referentes as vendas são armazenadas dentro dos registros dos clientes que realizaram essa compra. Para estes workloads foi realizando uma filtragem nos registros dos clientes, retornando apenas os dados referentes às vendas e o "Id" deste. Essa filtragem foi utilizada para simular um ambiente aonde a coleção "venda" é separada do cliente.

Todos os testes foram executados em uma máquina com um processador Intel(R) Core I5-4200U com velocidade de clock de 1.60 Ghz de mínima e 2.30GHz de máxima com um sistema operacional Windows 10 de arquitetura 64 bits e 4 GB de RAM . Em relação aos Bancos de dados, foi usado no relacional o MySQL Community Server 5.6.20, e no NoSQL, o MongoDB Community Server versão 3.4.

6. Experimentos

Para os experimentos foram utilizados dois tipos de base, uma relacional e outra não, em três situações diferentes, a primeira com 10.000 registros de clientes para cada modelo, a segunda com 100.000 e a terceira com 300.000, todos esses dados gerados através do software desenvolvido. Para a base relacional, foram gerados também para cada registro um registro de venda e quatro produtos relacionados à mesma, totalizando respectivamente nos três casos apresentados 10.000, 100.000 e 300.000 registros de venda e 40.000, 400.000 e 1.200.000 de produtos vendidos. O mesmo ocorre para o não relacional, com a diferença que a venda e os quatro produtos vendidos estão no mesmo registro, o registro do cliente. Outros dados inseridos são 1.000 produtos, 100 funcionários e 30 funções, estes inseridos da mesma forma nas três situações. Os registros gerados para os dois modelos são os mesmos, sendo somente armazenados de forma diferente.

Os testes consistiram em executar todos os workloads nas três situações para, além de verificar a diferença entre os dois modelos, averiguar se a diferença na quantidade de dados influencia significativamente, até os 300.000 registros de cliente em conjunto com os outros registros presentes em outras tabelas/coleções do modelo. Para aumentar a confiabilidade dos testes, essa sequência foi realizada 30 vezes e, a partir dos dados gerados, foi realizada uma média para cada workload de cada situação, sendo essa média submetida a uma função que calcula o desvio padrão para eliminar os pontos fora da curva.

6.1 – Workload 1

A tabela 5 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1.000 operações nos três casos analisados, sendo as operações 50% leitura e 50% alteração.

Tabela 5.Tempo de execução em milissegundos do workload 1

Tamanho da base	10.000	100.000	300.000
MySQL	35521	35920	36380
MongoDB	134	152	152

A partir dos resultados demonstrados na tabela 5, pode-se observar que o MongoDB obteve uma grande superioridade em questão de desempenho nas três situações estudadas, sendo a sua diferença de tempo de execução ao banco de dados relacional de aproximadamente 265 vezes menor na base com 10.000 clientes, 236 vezes menor na base com 100.000 e 239 vezes menor na base com 300.000.

6.2 – Workload 2

A tabela 6 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1.000 operações nos três casos analisados, sendo as operações 95% leitura e 5% alteração.

Tabela 6. Tempo de execução em milissegundos do workload 2

Tamanho da base	10.000	100.000	300.000
MySQL	3444	3573	3621
MongoDB	116	134	137

No tabela 6 pode-se observar que em um ambiente que 95% das operações são de leitura e outros 5% são de alteração, o MongoDB novamente demostra ser superior em questão de desempenho, sendo a sua diferença de tempo de execução ao relacional de aproximadamente 30 vezes menor na base com 10.000 clientes, 27 vezes menor na base com 100.000 e 26 vezes menor na base com 300.000.

6.3 – Workload 3

A tabela 7 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1.000 operações nos três casos analisados, sendo as operações 100% leitura.

Tabela 7. Tempo de execução em milissegundos do workload 3

Tamanho da base	10.000	100.000	300.000
MySQL	101	102	104
MongoDB	103	124	125

O objetivo do workload 3 é testar exclusivamente uma única operação, a de leitura. Na tabela 7 nota-se que os resultados favoreceram o SQL nas três situações, a diferença do tempo de execução em relação ao MongoDB foi de aproximadamente de 2 milissegundos com 10.000 registros, 22 milissegundos com 100.000 e 21 milissegundos com 300.000.

6.4 – Workload 4

A tabela 8 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1000 operações nos três casos analisados, sendo as operações 95% leitura e 5% inserção.

Tabela 8. Tempo de execução em milissegundos do workload 4

Tamanho da base	10.000	100.000	300.000
MySQL	3720	4047	4467
MongoDB	112	141	142

No tabela 6 pode-se observar que o MongoDB novamente obteve um melhor desempenho, possuindo uma diferença de tempo de execução ao banco de dados relacional de aproximadamente 33 vezes menor na base com 10.000 clientes, 29 vezes menor na base com 100.000 e 31 vezes menor na base com 300.000.

6.5 – Workload 5

A tabela 9 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1000 operações nos três casos analisados, sendo as operações 5% leitura e 95% alteração.

Tabela 9. Tempo de execução em milissegundos do workload 5

Tamanho da base	10.000	100.000	300.000
MySQL	67799	67935	68164
MongoDB	138	159	163

A partir dos resultados gerados pelo workload 5, pode-se observar que o MongoDB demonstrou ter um desempenho significativamente maior nas três situações estudadas, sendo a sua divergência de tempo de execução ao SQL de aproximadamente 491 vezes menor na base com 10.000 clientes, 439 vezes menor na base com 100.000 e 418 vezes menor na base com 300.000.

6.6 – Workload 6

A tabela 10 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1000 operações nos três casos analisados, sendo as operações 100% alteração.

Tabela 10. Tempo de execução em milissegundos do workload 6

Tamanho da base	10.000	100.000	300.000
MySQL	69959	70548	71424
MongoDB	141	159	161

No workload 6 foi testado exclusivamente uma única operação, a de alteração. A partir da tabela 10 observa-se que em um conjunto de operações exclusivamente de alteração, o MongoDB tem um desempenho muito maior nas três situações, a diferença do tempo de execução em relação ao banco de dados relacional foi de aproximadamente 496 vezes menor com 10.000 registros, 444 com 100.000 e 444 vezes com 300.000.

6.7 – Workload 7

A tabela 11 mostra os resultados baseados nas médias obtidas na execução de 30 vezes a sequência de 1000 operações nos três casos analisados, sendo as operações 100% inserção.

Tabela 11. Tempo de execução em milissegundos do workload 7

Tamanho da base	10.000	100.000	300.000
MySQL	82075	82766	83754
MongoDB	254	314	320

Os resultados registrados na tabela 11 demonstram que de todos wokloads do primeiro grupo (de 1 ao 7), este workload é o que exige o maior tempo de processamento, sendo executado apenas um tipo de operação, a de inserção. Pode-se notar que novamente que o MongoDB demonstrou uma superioridade no quesito de desempenho, a diferença do tempo de execução em relação ao SQL foi de aproximadamente de 323 vezes menor com 10.000 registros, 264 vezes menor com 100.000 e 262 vezes menor com 300.000.

6.8 – Workload 8

A tabela 12 mostra os resultados baseados nas médias obtidas na execução de 30 vezes uma única operação nos três casos analisados.

Tabela 12. Tempo de execução em milissegundos do workload 8

Tamanho da base	10.000	100.000	300.000
MySQL	7533	7721	7830
MongoDB	30	34	34

O objetivo do workload 8 é testar o acoplamento de um nível entre duas tabelas/coleções utilizando uma condição “where”. Na tabela 12 os resultados demonstram que quando executamos esse tipo de operação, o MongoDB se torna mais viável em questão de tempo de execução, possuindo uma divergência em relação ao SQL de aproximadamente 251 vezes para 10.000, 227 para 100.000 e 230 para 300.000.

6.9 – Workload 9

A tabela 13 mostra os resultados baseados nas médias obtidas na execução de 30 vezes uma única operação nos três casos analisados.

Tabela 13. Tempo de execução em milissegundos do workload 9

Tamanho da base	10.000	100.000	300.000
MySQL	7784	7933	8079
MongoDB	30	33	33

Na tabela 13 são demonstrados os resultados gerados pelo acoplamento de dois níveis entre três tabelas/coleções, com o objetivo de interligar essas e também a utilização de uma condição "where". Através dos resultados gerados por esse workload, observa-se que mesmo com um nível a mais, o MongoDB ainda é mais viável em questão de tempo de execução, não tendo praticamente quase nenhuma diferença em relação ao tempo do workload anterior. A diferença em relação ao banco de dados relacional é de aproximadamente 259 vezes para 10.000, 240 para 100.000 e 245 para 300.000.

6.10 – Workload 10

A tabela 14 mostra os resultados baseados nas médias obtidas na execução de 30 vezes uma única operação nos três casos analisados.

Tabela 14. Tempo de execução em milissegundos do workload 10

Tamanho da base	10.000	100.000	300.000
MySQL	845	3117	17992
MongoDB	30	39	44

O objetivo deste workload é testar a diferença de desempenho quando no MongoDB existir coleções dentro de coleções e sua representação em um banco de dados relacional através da ligação a duas outras tabelas (dois "Inner join" de um nível).Foi utilizado também um contador. Na tabela 14 os resultados demonstram que o tempo das execuções no MongoDB obtiveram um melhor desempenho, possuindo uma dissensão em relação ao SQL de, aproximadamente 28 vezes para 10.000, 80 vezes para 100.000 e 409 vezes para 300.000.

Um comportamento que pode ser observado no banco de dados relacional, é que mesmo com dois acoplamentos de um nível, com comando "count()", ele demonstra ter um desempenho melhor do que quando possui um nível usando uma instrução "where". Esse comportamento é observado até a situação com 100.000 registros.

6.11 – Workload 11

A tabela 15 mostra os resultados baseados nas médias obtidas na execução de 30 vezes uma única operação nos três casos analisados.

Tabela 15. Tempo de execução em milissegundos do workload 11

Tamanho da base	10.000	100.000	300.000
MySQL	1455	5369	25911
MongoDB	57	59	61

Este workload é uma extensão do anterior, sendo que neste a coleção composta está ligada a uma coleção exterior, no relacional para a representação, foi adicionado mais um nível a um dos "Inner Join". A partir dos resultados gerados por esse workload, percebe-se que o MongoDB obteve um melhor desempenho nas três situações. A diferença em relação ao banco de dados relacional é de aproximadamente 25 vezes para 10.000, 91 vezes para 100.000 e 425 vezes para 300.000.

7 - Conclusão

A partir do cenário estudado, fica nítido que dos 11 testes executados, 10 desses o MongoDB demonstrou ser muito mais eficaz. O único wokload que favoreceu o SQL foi o que possui somente operações de busca (workload 3), mas mesmo assim, a diferença foi pequena e, quando está dentro de um conjunto de operações (wokload 1, 2, 4 e 5), mesmo que seja 95% destas, o MongoDB se sobressai novamente.

Em relação ao segundo grupo, observa-se que no modelo NoSQL estudado, não existe alteração significante entre as três variações de volume das bases de dados, mesmo que as buscas sejam acompanhadas de operações de cálculo (workload 10 e 11) ou condições (workload 8 e 9). Já no SQL, quando existe um acoplamento entre tabelas nas pesquisas, uma condição atribuída a esta gera um esforço computacional muito maior que uma simples operação "count()" nas situações com 10.000 e 100.000. Por outro lado, a variação entre os três volumes em uma busca com esta operação é muito mais expressivo, como pode ser visto nos resultados dos workloads 10 e 11.

No banco de dados relacional observa-se que até os 300.000 registros de clientes associados a outros registros presentes em tabelas/coleções distintas, no primeiro grupo (workload do 1 ao 7) não existe uma grande diferença no tempo de execução, Já o modelo NoSql demonstra uma variação significativa entre 10.000 e 100.000 (90.000 a mais), mas esta variação torna-se pequena a partir 100.000, como poder ser visto entre 100.000 e 300.000 (200.000 a mais).

Referências

[1]Cláudio, A: "Bancos de Dados Relacionais", http://www.devmedia.com.br/bancos-de-dados-relacionais-artigo-revista-sql-magazine-86/20401

[2]Abramova, V., Bernardino, J., Furtado, P.(2014) :"EXPERIMENTAL EVALUATION OF NOSQL DATABASES" Vol. 6, No. 3, June.

[3]Chede, C. (2012): “Você realmente sabe o que é Big Data?”, https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/voce_realmente_sabe_o_que_e_big_data?lang=en, Abril.

[4]Vimala, S., Khanna Nehemiah, H., Bhuvaneswaran, R. S., and Saranya, G. (2013): “Design Methodology” for No.3, Junho.

[5]Sasaki, B. M. (2015): “Graph Databases for Beginners: ACID vs. BASE Explained”, https://neo4j.com/blog/acid-vs-base-consistency-models-explained/

[6]Browne, J. (2009): “Brewer's CAP Theorem”, http://www.julianbrowne.com/article/viewer/brewers-captheorem

[7]Data Science Academy: “Big data fundamentos”,

http://www.datascienceacademy.com.br/path-player?courseid=big-data-fundamentos

[8]Brito, R. :“Bancos de Dados NoSQL x SGBDs Relacionais:Análise Comparativa”.

[9]Kokay, M. C. (2012): “Banco de dados NoSQL: Um novo paradigma - Revista SQL Magazine 102” , http://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918, Agosto.

[10]Lennon, J. (2011): “Explore o MongoDB”, https://www.ibm.com/developerworks/br/library/os-mongodb4/, Julho

[11]Arboleda, F. J.M. , Rendón, J. E. Q., Vásquez, R. R. (2016):”UNA COMPARACIÓN DE RENDIMIENTO ENTRE ORACLE Y MONGODB” . Ciencia e Ingeniería Neogranadina, Março.

[12]JSON: “Introdução ao JSON”, http://www.json.org/json-pt.html

[13]Code Academy: “Learn SQL”, https://www.codecademy.com/learn/learn-sql

[14]MongoDB query Comparison: “Comparison Query Operators”, https://docs.mongodb.com/manual/reference/operator/query-comparison/

[15]Microsoft Developer Network: “Operadores de comparação”, https://msdn.microsoft.com/pt-br/library/ms188074.aspx

[16]MongoDB: “ObjectId()”, https://mongodb.github.io/node-mongodb-native/api-bson-generated/objectid.html

[17]Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2014): “Benchmarking cloud serving systems with YCSB”, June.

[18]MongoDB Operations: “MongoDB CRUD Operations”, https://docs.mongodb.com/manual/crud/

[19]Medeiros, H.: “Introdução ao MongoDB”, http://www.devmedia.com.br/introducao-ao-mongodb/30792

[20]Machado, H.: “Hadoop MapReduce: Introdução a Big Data” , http://www.devmedia.com.br/hadoop-mapreduce-introducao-a-big-data/30034

[21]DevMedia: “Big Data: MapReduce na prática”, http://www.devmedia.com.br/big-data-mapreduce-na-pratica/32812