Universidade Federal do Rio Grande do Norte

Instituto Metrópole Digital

Programa de Residência em Tecnologia da Informação

Turma - Laboratório de Inovação Tecnológica em Saúde

Disciplina: Banco de Dados

Professor: Gustavo Bezerra Paz Leitão

WESLEY ADAMO COSTA DOS SANTOS

ESTUDO DE CASO : ANÁLISE DE DESEMPENHO ENTRE OS BANCOS DE DADOS NOSQL MONGODB E COUCHDB EM PYTHON

NATAL/RN

2020

SUMÁRIO

1. INTRODUÇÃO

Com necessidade de armazenar volumes cada vez maiores de dados e informações com maior escalabilidade e eficiência, percebeu-se que os bancos de dados relacionais não supriam as necessidades exigidas com grandes volumes de dados. Sendo assim, foram pensadas e desenvolvidas tecnologias capazes de armazenar e analisar esses dados, surgindo assim os bancos de dados NoSQL.

Os bancos de dados NoSQL são divididos em quatro categorias: chave-valor, grafos, coluna e documentos. Esses, por sua vez, armazenam e gerenciam os dados em uma estrutura chamada de documento. Um documento contém todas as informações que o define, sendo recomendados para representar dados semi-estruturados e podem ser armazenados utilizando o padrão de arquivos JSON/BSON, amplamentes disseminados entre os desenvolvedores devido sua fácil interpretação e utilização .

Em relação às opções de bancos de dados orientados a documentos, destacam-se o MongoBD e o CouchDB. Sendo assim, o presente trabalho tem como finalidade analisar as características desses dois importantes bancos de dados, comparando-os e analisando o desempenho nas operações de inserção, atualização, busca e remoção de documentos.

2. OBJETIVOS

O presente trabalho tem como finalidade comparar o desempenho entre o MongoDB e o CouchDB nas operações de CRUD, bem como os seguintes objetivos específicos:

- Descrever as funcionalidades implementadas;
- Apresentar os resultados obtidos e as interpretações ;
- Fornecer uma análise entre dois bancos de dados noSQL orientados a documentos

3. REQUISITOS

Nesta seção são apresentados os requisitos funcionais do estudo de caso, mostrando detalhes de suas especificações. A tabela a seguir apresenta resumidamente os nomes dos requisitos, o grau de importância e complexidade de implementação.

Tabela 1 - Requisitos funcionais

Requisitos Funcionais		
Requisito	**Grau de importância**	**Complexidade**
RF001: Ler json	Médio	Baixa
RF002: Inserir documentos no MongoDB	Alto	Baixa
RF003: Inserir documentos no CouchDB	Alto	Baixa
RF004: Buscar documentos no MongoDB	Alto	Baixa
RF005: Buscar documentos no CouchDB	Alto	Baixa
RF006: Atualizar documentos no MongoDB	Alto	Baixa
RF007: Atualizar documentos no CouchDB	Alto	Baixa
RF008: Remover documentos no MongoDB	Alto	Baixa
RF009: Remover documentos no MongoDB	Alto	Baixa

Fonte: Autor

RF001: Este requisito consiste na leitura do JSON contendo os dados a serem utilizados na comparação entre as tecnologias utilizadas..

RF002: Essa funcionalidade é responsável por permitir a inserção dos documentos no MongoDB. Seu grau de importância é alto, pois é por meio dela que os dados são inseridos na base de dados. Devido utilizar o driver do SGBD para inserir os dados, o grau de complexidade é baixo.

RF003: É por meio dessa funcionalidade que os documentos são inseridos no CouchDB. Sendo assim de grande importância. Já sua implementação é de baixa complexidade, pois é por meio do driver disponibilizado pelo SGBD que os dados serão armazenados no meio físico.

RF004: A busca de documentos armazenados na base de dados do MongoDB é realizada por meio dessa funcionalidade. Seu grau de importância é alto, pois é por meio dela que os dados são recuperados na base de dados e sua complexidade é baixa.

RF005: Essa funcionalidade é responsável por permitir a busca de documentos armazenados na base de dados do CouchDB, o que justifica seu grau de importância ser alto. Já em relação a complexidade, o driver prover todos os mecanismos necessários para tal funcionalidade, o que diminui a complexidade na implementação da funcionalidade.

RF006: Os documentos armazenados na base de dados do MongoDB são atualizados por meio desta funcionalidade. Seu grau de importância é alto e com baixa complexidade em sua implementação.

RF007: Permite a atualização dos documentos armazenados na base de dados do CouchDB. Seu grau de importância é alto e sua complexidade é baixa.

RF008: Responsável por permitir a remoção dos documentos armazenados na base de dados do MongoDB. Seu grau de importância é alto, pois é uma funcionalidade essencial e seu grau de complexidade é baixo..

RF009: Essa funcionalidade é responsável por permitir a remoção dos documentos armazenados na base de dados do CouchDB. Seu grau de importância é alto, pois é por meio dela que os dados serão removidos na base de dados. Já sua implementação é de baixa complexidade, pois é por meio do driver disponibilizado pelo SGBD que os dados serão removidos do meio físico.

4. DESENVOLVIMENTO

4.1 Tecnologias Utilizadas

Na tabela 2 são apresentadas as tecnologias/ferramentas utilizadas no desenvolvimento do estudo.

Tabela 2 - Tecnologias utilizadas no desenvolvimento do estudo

Tecnologia	Versão
Python	3.6
Docker	19.03.12
Imagem MongoDB Docker	4.2.3
Imagem CouchDB Docker	3.1.0
Pymongo (driver MongoDB para python)	3.10.1
Driver CouchDB para Python	1.2
Postman	7.2.3

Fonte: Autor

4.1 Ambiente de testes

A tabela a seguir apresenta as especificações da máquina utilizada nos testes de desempenho entre os SGBDs.

Tabela 3 - Tecnologias utilizadas no desenvolvimento do estudo

Nome	Descrição
Processador	Core i7 8565u
Memória RAM	8GB DDR4 2666MHZ
HD	SSD M.2 Samsung PM981
Sistema Operacional	Linux Mint 19.3

Fonte: Autor

4.2 Implementação

Como mencionado, foi utilizado a linguagem Python na codificação. Primeiramente, usando o pandas, é lido o documento JSON contendo todos os dados que serão utilizados. O arquivo JSON utilizado contém informações acerca dos diplomas expedidos na UFRN desde 2000 (59126 diplomas) e foi criado a partir de um dataset disponível em http://dados.gov.br/dataset/diplomas. Abaixo, um exemplo de um documento salvo no MongoDB.


```
{
        "_id" : "0",
        "nome_discente" : "CLÁUDIO REGIS DOS SANTOS LUCAS",
        "curso" : "DOUTORADO EM CIÊNCIA E ENGENHARIA DE PETRÓLEO",
        "nivel_ensino" : "DOUTORADO",
        "titulacao" : "DOUTOR EM CIÊNCIA E ENGENHARIA DE PETRÓLEO",
        "data_conclusao" : "2020/07/01 00:00:00.000",
        "data_expedicao_diploma" : "2020/07/02 00:00:00.000",
        "nome_reitor" : "JOSÉ DANIEL DINIZ MELO",
        "descricao_reitor" : "Reitor",
        "numero_diploma" : null,
        "data_registro" : "2020/07/01 00:00:00.000",
        "num_processo_registro" : "23077.045592/2020-52",
        "livro" : "S",
        "numero_folha" : 397
}
```


Quanto a funcionalidade de inserção, o código abaixo representa como é feita a inserção dos dados no CouchDB. Na chamada da função é passado um ‘n’ que representa a quantidade de documentos que serão salvos. Como o arquivo JSON já contém os ‘ids’, para não dar erro de id duplicado, a cada chamada da função de inserção são removidos todos os documentos salvos. Em seguida, é iniciada a marcação de tempo. A inserção é realizada por meio de um laço de repetição “for” passando por cada documento contido no arquivo até a quantidade ‘n’ passada por parâmetro. Por fim, é encerrada a marcação de tempo e apresentado o valor obtido.

```
for x in self._arquivo_json.values:
   if i <= n:
       collection.save(x)
       i += 1
   else:
       break
```

Já a atualização dos documentos consiste em atualizar o atributo ‘curso’ para ‘MECATRÔNICA’ e o atributo ‘livro’ para ‘P’ para uma determinada quantidade ‘n’ (enviado na chamada da função) de documentos. Assim como a inserção, utiliza-se o laço de repetição para atualizar os ‘n’ documentos. No caso do CouchDB, são recuperados todos os documentos salvos e atualizados os atributos dos ‘n’ documentos no laço de repetição e armazenados em uma lista temporária que será enviada como parâmetro na chamada da função de atualização de documentos do driver do SGBD . A mesma lógica é utilizado no método do MongoDB. O trecho de código abaixo demonstra a atualização de documentos no CouchDB.

```
for i in range(quantidade):
   doc = list_documents[i]['doc']
   doc['curso'] = "MECATRÔNICA"
   doc['livro'] = "P"
   list_temp.append(doc)

bd.update(list_temp)
```

Em relação a busca de documentos, foi utilizado como critério de busca todos os alunos que sejam da graduação ou o nome do reitor seja ‘José Daniel Diniz Melo’ ou o livro seja ‘S’. Como a quantidade de documentos recuperados é maior que 30 mil, foi utilizado um limite de 5 mil documentos. Na atualização é utilizado um “for” para recuperar os documentos ‘n’ vezes. Sendo assim, se for passado um n=10, serão feitas 10 chamadas ao método de busca do driver do SGBD chamado, como pode ser observado no trecho de código abaixo..

```
"selector": {
   "$or":
       [
           {'nivel_ensino': 'GRADUAÇÃO'},
           {'nome_reitor': 'José Daniel Diniz Melo'},
```

```
            {'livro': 'S'}
        ]
},
"limit": 5000
```

Por fim, a remoção é feita passando um valor ‘n’ por parâmetro na chamada da função definindo a quantidade de documentos que serão removidos. A fim de testar quanto tempo é necessário para executar a remoção, são feitas ‘n’ chamadas ao método de remoção de documento do driver do SGBD passando como parâmetro o documento a ser excluído. O trecho de código abaixo demonstra como é realizada a remoção no CouchDB.

```
for i in range(n):
   bd.delete(list_documents[i]['doc'])
```

5. RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados obtidos nas operações de inserção, atualização, busca e remoção para cada banco de dados comparados de acordo a quantidade de dados e iterações. Para cada operação foram realizadas três execuções. Sendo assim, o valor obtido em segundo(s) representa a média aritmética das três execuções para cada operação analisada. Com o objetivo de verificar apenas o tempo decorrido na execução do método do driver do SGBD, é desconsiderado o tempo de outras instruções. Sendo assim, a marcação de tempo inicia-se assim que é realizada a chamada ao método do driver. Para visualizar todas as execuções e seus respectivos resultados, consultar repositório[1] contendo os prints dos resultados.

5.1 Inserção

A tabela 4 apresenta os valores de tempo para a operação de inserção, considerando a inserção de 1, 10, 100, 1000, 1000 e 59621 documentos (tamanho total do arquivo JSON). Destaca-se a diferença significativa nos valores obtidos entre os dois bancos de dados

Tabela 4 - Resultados obtidos na operação de inserção.

Inserções	**1**	**10**	**100**	**1000**	**10000**	**59621**
MongoDB	0.0015	0.0066	0.0415	0.2843	4.2311	23,1313
CouchDB	0.0140	0.1165	0.8241	13.4920	185.8771	1498.2843
Tempo (s)						

Fonte: Autor

5.2 Busca

Na tabela 5 é apresentada o desempenho dos bancos de dados na operação de busca, considerando a quantidade de iterações realizadas em uma execução do

[1] Repositório : https://github.com/wesleyadamo/trabalho-final-bd

algoritmo. Sendo assim, em 10 buscas realizadas em uma única execução, o MongoDB retornou os dados em 0.0018 segundo e o CouchDB em 34.0843 segundos.

Tabela 5 - Resultados obtidos na operação de busca

Iterações	1	10	100	1000
MongoDB	0.0001	0.0003	0.0018	0.0141
CouchDB	0.6925	2.9622	34.0843	363.6698
Tempo(s)				

Fonte: **Autor**

5.3 Atualização

Para cada execução são apresentados os valores de tempo considerando a quantidade de documentos atualizados, como demonstrado na tabela a seguir.

Tabela 6 - Resultados obtidos na operação atualização

Doc. Atualizado	1	10	100	1000	10000	59621
MongoDB	0.0031	0.0030	0.0047	0.0238	0.1645	0.8361
CouchDB	0.0404	0.0241	0.0366	0.1544	2.7499	107.0716
Tempo(s)						

Fonte: Autor

5.4 Remoção

Por fim, a tabela 7 apresenta o desempenho em segundo de cada banco de dados em relação a quantidade de dados removidos durante a execução do código.

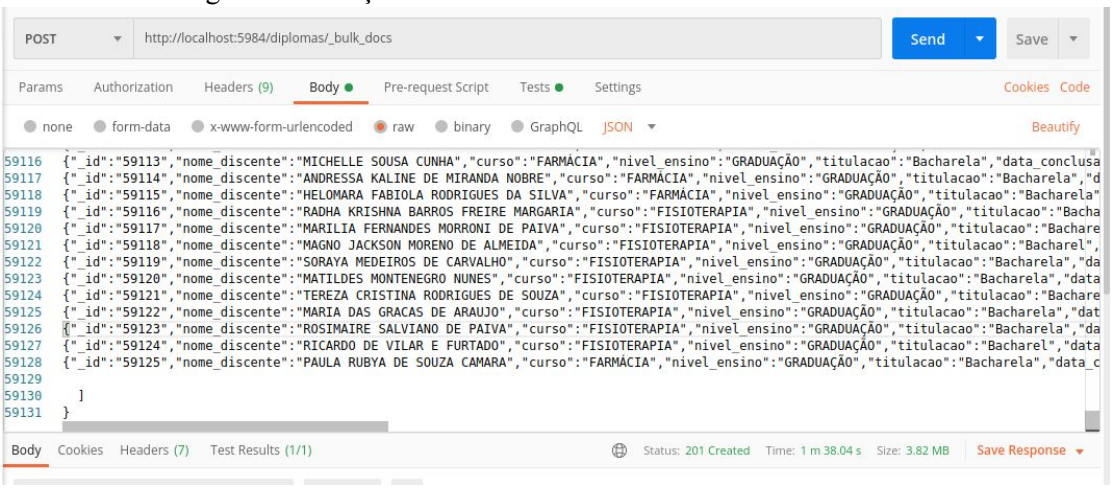
Tabela 6 - Resultados obtidos na operação remoção

Doc. Removidos	1	10	100	1000	10000
MongoDB	0.0031	0.0074	0.0611	0.5042	5.0043
CouchDB	0.0396	0.1151	1.1741	11.6866	157.8869
Tempo(s)					

Fonte: Autor

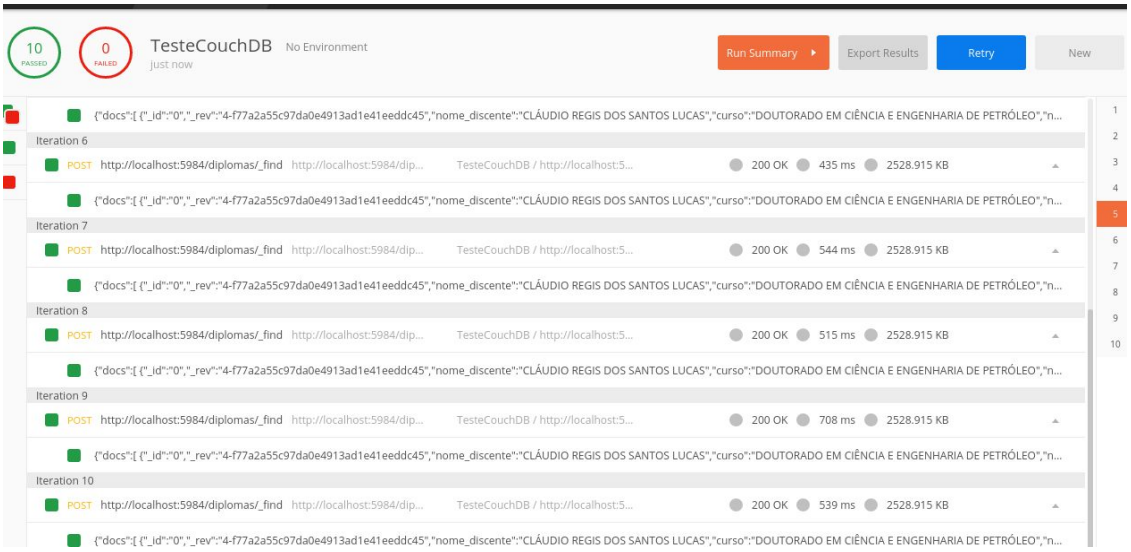
Como demonstrado, o MongoDB apresentou para todas as operações um resultado significativamente melhor. Porém, sendo realizada uma pesquisa em fontes disponíveis na internet, percebeu-se que utilizando o driver o resultado obtido com o CouchDB foi próximo ou até pior ao que demonstrado acima. Com isso, foram feitos testes das duas operações mais demoradas do estudo com a API do CouchDB utilizando o Postman. O resultado da inserção do arquivo completo (59621 documentos) durou aproximadamente 98 segundos contra aproximadamente 1499 segundos do obtido no Python, como pode ser observado na imagem 1. Já para a busca, como demonstrado na imagem 2 abaixo, foi feito um teste com 10 iterações, utilizando o mesmos critérios de buscas do estudo, o resultado no Postman foi aproximadamente 5 segundos contra aproximadamente 2.8 segundos do resultado obtido no estudo.

Imagem 1 - Inserção de todos os documentos utilizando o Postman.



Fonte: Autor

Imagem 1 - Teste de busca com 10 iterações no CouchDB utilizando o Postman..



Fonte: Autor

Imagem 3 - Resultado obtido no teste de busca com o Postman.

Fonte: Autor

6. CONCLUSÃO

Diante do que foi apresentado durante o estudo de caso, para as especificações da máquina e das tecnologias utilizadas, o banco de dados MongoDB apresentou um resultado melhor em todas as análises realizadas, sendo recomendado para ser utilizado como banco NoSQL orientado a documentos em Python, pois, além dos resultados melhores, há mais documentação/artigo disponível para ser consultado, o que garante utilizar o driver de forma mais otimizada. Porém, como mencionado anteriormente, o CouchDB apresentou resultado melhor na busca utilizando requisições HTTP no Postman. Essa diferença significativa utilizando as implementações oficiais do drivers dos dois bancos analisados, pode está relacionada as otimizações feitas durante suas implementações, bem como o desenvolvimento maior de atualizações no driver do MongoDB. Para a continuação das análises, serão realizados testes utilizando o particionamento dos dados em ambos SGBD's e os resultados disponibilizados no repositório do estudo.