UM ESTUDO COMPARATIVO ENTRE O MONGODB E O POSTGRESQL[1]

Emanuele Araujo Solagna[2]

Alexandre Tagliari Lazzaretti[3]

RESUMO

Nos últimos anos houve um aumento significativo do uso de aplicações não-convencionais. A utilização de bancos de dados NoSQL (Not Only SQL) vem crescendo significativamente, principalmente pelo aumento do volume de dados gerados e a da necessidade de suprir a demanda de aplicativos web escaláveis. Por esta razão, os modelos tradicionais de bancos de dados apresentam fatores limitantes, pois não possuem muita flexibilidade na estruturação e são limitados quanto ao processamento dos grandes volumes de dados. No entanto, os Bancos de Dados NoSQL são recentes e ainda pouco conhecidos, o que pode ser um empecilho no momento de escolher um destes para gerenciamento de uma base de dados. Assim, este trabalho realizou, baseado em um conjunto de dados estruturados, um estudo sobre o banco de dados NoSQL MongoDB e uma comparação através de diversos comandos de manipulação com o SGBD PostgreSQL. Como resultado, pode-se concluir que o banco de dados MongoDB demonstrou ter uma melhor performance ao ser submetido aos testes propostos.

Palavras-chave: NoSQL, Bancos de Dados, Dados, Escalabilidade[4].

1 INTRODUÇÃO

O aumento no tráfego de dados exige que empresas procurem soluções para poder suprir a demanda cada vez maior de performance e disponibilidade que fazem com que outros requisitos até então indiscutíveis, como consistência de dados, sejam revistos (VOGELS, 2009). Como opção para a solução destes e outros problemas, tecnologias emergentes, recentemente chamadas de movimento NoSQL (Not Only SQL), surgiram como uma nova classe de SGBDs, que utilizam outros

---

[1] Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, Campus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

[2] Graduanda em Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense, Campus Passo Fundo (IFSul – Campus Passo Fundo/RS). E-mail: emanuelesolagna@hotmail.com.

[3] Orientador: professor do IFSul. E-mail: alexandre.lazzaretti@passofundo.ifsul.edu.br.

[4] Habilidade de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer.

modelos, diferentes do convencional e que vêm evoluindo trazendo um paradigma diferente de armazenamento de dados, como, por exemplo, armazenamento de documentos sem esquema definido.

O termo NoSQL foi utilizado pela primeira vez em 1998, como um banco de dados relacional de código aberto que não possuía interface SQL. "O movimento NoSQL é completamente distinto do modelo relacional e, portanto, deveria ser mais apropriadamente chamado NoREL (STROZZI, 2007). Somente em 2009 o termo foi reintroduzido em um evento que visava discutir banco de dados open source distribuídos, promovido por Johan Oskarsson. Posteriormente surgiram projetos NoSQL como MongoDB, CouchDB e Cassandra (EVANS, 2009).

O principal objetivo deste trabalho é realizar uma comparação de desempenho entre dois bancos de dados gratuitos e que vem sendo utilizados atualmente. Para realizar este comparativo, foram escolhidas as ferramentas PostgreSQL (POSTGRESQL, 2016) e MongoDB (MONGODB, 2016).

Através da implementação de um ambiente de testes, foi possível realizar a comparação, analisar os resultados e tirar conclusões sobre o desempenho das duas tecnologias no cenário utilizado.

2 REFERENCIAL TEÓRICO

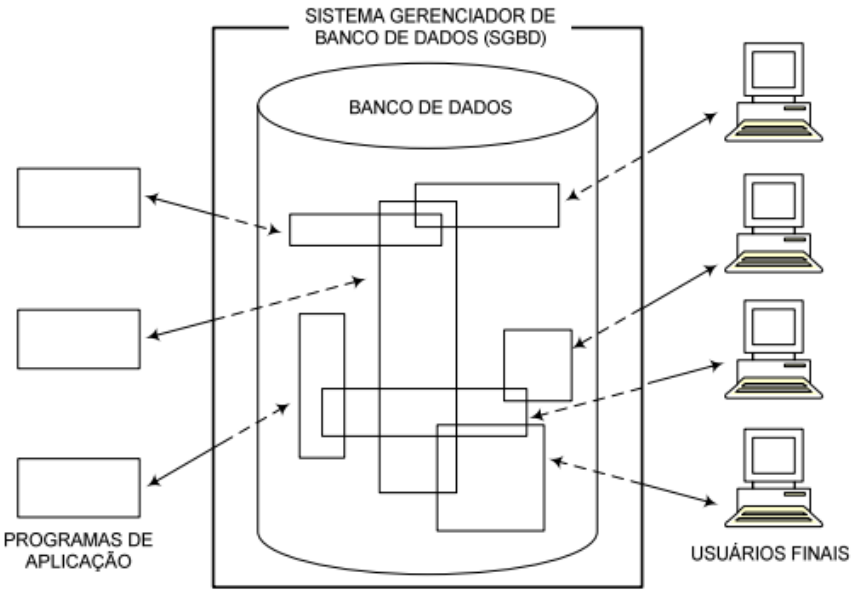
Nesta seção são detalhados os conceitos e definições necessárias para o entendimento do trabalho realizado.

2.1 BANCO DE DADOS

Segundo Date (2004, p. 6), “Um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa”, ou seja, um montante de informações pertinentes a determinado sistema de aplicação. Em outras palavras, pode ser definido como um local onde são armazenados dados necessários à manutenção das atividades de determinada organização, que possuem vários níveis de interação com o mundo real e com o público interessado em seu conteúdo.

Os componentes de um banco de dados são: o próprio banco de dados e o Software Gerenciador do Banco de Dados. E, quem tem acesso aos seus componentes são os usuários e as aplicações (Figura 1).

Figura 1. Representação de um Sistema de Banco de Dados.



Fonte: DATE, 2004 (ADAPTADO)

Para Elmasri e Navathe (2011, p. 3), na expressão Banco de Dados, subentende-se que:

> Um banco de dados representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD – Universe of Discourse). As mudanças no minimundo são refletidas no Banco de Dados. Um banco de dados é uma coleção logicamente coerente de dados com algum significado inerente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados. Um banco de dados é projetado, construído e populado com dados para uma finalidade específica. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados.

Entre o banco de dados armazenado e os usuários há um conjunto de programas denominado Software Gerenciador de Banco de dados (SGBD) (DATE, 2004). Então, a gestão dos bancos de dados se dá graças aos SGBDs.

“O principal objetivo de um SGDB é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações

do banco de dados“, (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 1). Para tanto, o Software Gerenciador de Banco de Dados disponibiliza recursos para definir, construir, manipular, compartilhar, proteger e manter bancos de dados (ELMASRI; NAVATHE, 2011).

Portanto, podemos entender o SGBD como o conjunto de programas responsável por gerenciar o banco de dados, pois fornecem suporte aos métodos de acesso, ou seja, maneiras de armazenar e recuperar informações de forma apropriada.

2.2 MODELO DE DADOS

Na definição de Elmasri e Navathe (2011, p. 19), um modelo de dados é “uma coleção de conceitos que podem ser usados para descrever a estrutura de um banco de dados”. Então, o modelo de um banco de dados pode ser entendido como a forma de descrever os seus dados, relacionamentos, semântica e restrições de consistência. O modelo de dados descreve o projeto do banco nas formas física, lógica e de visão (SILBERSCHATZ, 2006).

Cada SGBD segue um modelo de banco de dados, de forma a diferenciar esses dados de acordo com sua representação.

2.2.1 Modelo Relacional

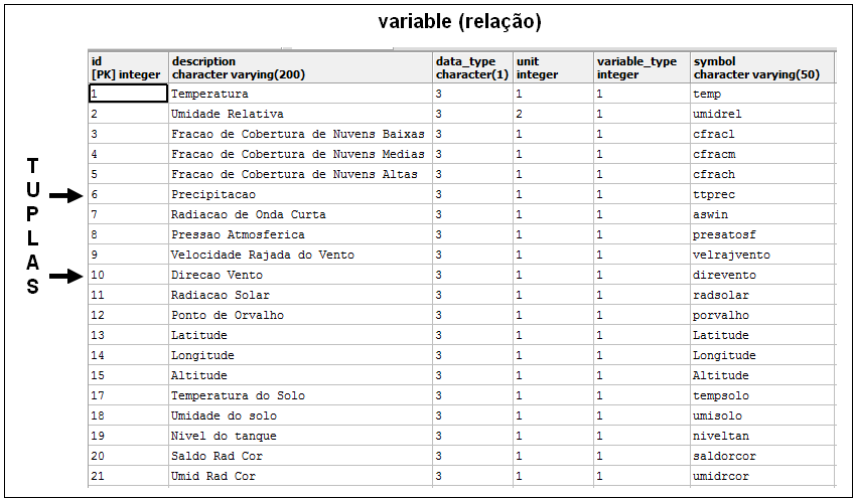
Para Date (2004, p. 67), o modelo relacional refere-se a “três aspectos principais dos dados: a estrutura de dados, a integridade de dados e a manipulação de dados”.

Um banco de dados que segue o Modelo Relacional, representa seus dados como um conjunto de relações. Considerando que uma relação é, de certo modo, similar a uma tabela de valores e aplicando a terminologia do Modelo Relacional diz-se que as linhas denominam-se tuplas; as colunas, atributos; e a tabela em si, relação (ELMASRI; NAVATHE, 2011).

Relações, ou tabelas, são constituídas de uma série de elementos (atributos), que possuem características comuns. Um exemplo é mostrado na Figura 2, na tabela denominada “variable”. Na mesma figura, pode-se observar o conceito de atributo ou campo. Cada atributo indica uma determinada característica associada a

tabela. Nesse caso, tem-se os atributos*: id, description, data_type, unit, variable_type* e *symbol*. O domínio é caracterizado como sendo os valores possíveis do atributo. As tuplas, ou registros, indicam as linhas da relação**.**

Figura 2. Exemplo de tabela do Modelo Relacional



id [PK] integer	description character varying(200)	data_type character(1)	unit integer	variable_type integer	symbol character varying(50)
1	Temperatura	3	1	1	temp
2	Umidade Relativa	3	2	1	umidrel
3	Fracao de Cobertura de Nuvens Baixas	3	1	1	cfracl
4	Fracao de Cobertura de Nuvens Medias	3	1	1	cfracm
5	Fracao de Cobertura de Nuvens Altas	3	1	1	cfrach
6	Precipitacao	3	1	1	ttprec
7	Radiacao de Onda Curta	3	1	1	aswin
8	Pressao Atmosferica	3	1	1	presatosf
9	Velocidade Rajada do Vento	3	1	1	velrajvento
10	Direcao Vento	3	1	1	direvento
11	Radiacao Solar	3	1	1	radsolar
12	Ponto de Orvalho	3	1	1	porvalho
13	Latitude	3	1	1	Latitude
14	Longitude	3	1	1	Longitude
15	Altitude	3	1	1	Altitude
17	Temperatura do Solo	3	1	1	tempsolo
18	Umidade do solo	3	1	1	umisolo
19	Nivel do tanque	3	1	1	niveltan
20	Saldo Rad Cor	3	1	1	saldorcor
21	Umid Rad Cor	3	1	1	umidrcor

Fonte: O Autor

2.2.2 Modelo NoSQL (Not Only SQL)

Segundo Suissa (2010) o termo NoSQL, foi utilizado pela primeira vez em 1998, por Carlo Strozzi, como nome de seu SGBD, baseado no Modelo Relacional, sem interface SQL. NoSQL é um termo genérico para uma classe definida de banco de dados que rompe o paradigma dos bancos de dados baseados nos Modelos Relacionais.

Segundo Sadalage e Fowler (2013), os bancos de dados NoSQL baseiam-se nas necessidades da web no século XXI, de modo que, geralmente, apenas sistemas desenvolvidos nesse período são chamados NoSQL, excluindo, dessa forma, muitos dos bancos de dados criados antes do novo milênio.

O NoSQL foi projetado para promover uma alternativa de armazenamento com velocidade e disponibilidade elevada. E começaram a ser desenvolvido por grandes empresas para suprir a necessidade de alto armazenamento e escalabilidade. Atualmente são tratados como o futuro de grande armazenamento de informações, conseguindo assim trabalhar com esse novo cenário que os bancos de dados Relacionais não atendem como desejado.

Os bancos de dados NoSQL são classificados de acordo com a forma de armazenar os dados. Existem 4 tipos de categorias: chave/valor, orientado a colunas, orientado a documentos e baseados em grafos (POPESCU, 2010).

1. Armazenamento chave-valor (Key/Value): O armazenamento do tipo chave/valor é semelhante ao uso de mapas ou de dicionários, os dados são endereçados por uma única chave, permitindo aos clientes colocar e solicitar valores por chaves. Esses sistemas podem conter dados estruturados ou não estruturados. Esses bancos são úteis para operações simples, baseadas somente em atributos chave. Por exemplo, sistemas com rápida troca de dados e com frequente escrita. Como a maioria dos armazenamentos chave/valor mantém seu conjunto de dados em memória, eles são bastante usados para cache de consultas SQL (LEAVITT, 2010).
2. Armazenamento em colunas: Um SGBD orientado a colunas armazena seu conteúdo de forma inversa aos bancos de dados orientados a linhas. Este formato de armazenar as informações torna-se vantajoso para Data Warehouses[5], onde agregações são processadas sobre uma quantidade de dados de características similares.
3. Armazenamento baseado em Grafos: O armazenamento baseado em grafos fundamenta-se na teoria dos grafos. Em geral, grafos consistes de nós, propriedades e arestas. Os nós representam as entidades, as propriedades representam os atributos e as arestas representam as relações (LEAVITT, 2010).
4. Armazenamento orientados a documentos: Segundo Lennon (2013), o MongoDB armazena dados dentro de documentos semelhantes a JSON (usando BSON — uma versão binária de JSON), que retém os dados usando pares de chave/valor. Um banco de dados orientado a documentos armazena, recupera e gerencia dados semiestruturados. O elemento dos dados é chamado documento. Os documentos são endereçados no banco de dados via uma chave única que representa o documento. Uma das características de um banco de dados orientado a documentos é que, além da simples

[5] Depósito de dados digitais que serve para armazenar informações detalhadas relativamente a uma empresa.

chave/valor de pesquisa, é possível recuperar um documento através de uma API, ou linguagem de consulta disponibilizada pelo banco. Todos os bancos de dados correntes fornecem suporte a documentos no formato JSON (LEAVITT, 2010).

O JSON é um padrão aberto de estruturação de dados baseado em texto e legível por humanos, utilizado para a serialização de dados. Foi projetado com o objetivo de ser simples, portátil, textual, e um subconjunto do JavaScript e tem se mostrado uma ótima alternativa ao XML, inclusive sendo nativo de armazenamento em alguns bancos de dados (JSON, 2016). Na Figura 3 está um exemplo de documento JSON.

Figura 3. Exemplo de documento JSON.

```
{
        "title": "Example Schema",
        "type": "object",
        "properties": {
                "firstName": {
                        "type": "string"
                },
                "lastName": {
                        "type": "string"
                },
                "age": {
                        "description": "Age in years",
                        "type": "integer",
                        "minimum": 0
                }
        },
        "required": ["firstName", "lastName"]
}
```

Fonte: JSON Schema (2016)

Outro formato de documento utilizado é o BSON, semelhante ao JSON, porém com armazenado em binário, consistindo em uma lista de elementos chave/valor, onde a chave é uma string e o valor pode ser string, inteiro, data, booleano, nulo, objeto BSON ou matriz BSON (BSON, 2012).

2.3 MONGODB

O MongoDB inicialmente foi desenvolvido como um componente de serviço pela empresa 10gen em outubro de 2007, passando a ser um software open source em 2009.

MongoDB (do inglês humongous, "gigantesco") é uma aplicação de código aberto, de alta performance, sem esquemas, orientado a documentos. Foi escrito na linguagem de programação C++ (BATISTA et al., 2013).

Os bancos de dados orientados a documentos são bastante diferentes dos tradicionais bancos de dados relacionais. Em vez de armazenar dados em estruturas rígidas, como tabelas, eles os armazenam em documentos vagamente definidos (CUNHA, 2011). Um documento é a unidade de dados armazenável. Possui uma estrutura comparável a um documento XML ou JSON. De fato, o MongoDB baseia os documentos em um formato chamado BSON, o qual é muito similar ao JSON.

O MongoDB é considerado um dos projetos mais notáveis NoSQL devido sua linguagem de consulta de alta performance, baseada em documentos e facilidade de migração de um banco de dados relacional, pelo fato de suas consultas serem convertidas fácilmente.

Para gerenciar o MongoDB, foi utilizada a interface gráfica Robomongo 0.9.0-RC6, uma aplicação nativa do MongoDB, criada pela empresa de desenvolvimento Paralect, a qual publicou a primeira versão do Robomongo em 28 de maio de 2012.

2.4 POSTGRESQL

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (SGBDOR).

Foi desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley em 1998. À época, um programador chamado Michael Stonebraker liderou um projeto para a criação de um servidor de banco de dados relacionais chamado Postgres, oriundo de um outro projeto da mesma instituição denominado Ingres. Esse projeto foi pioneiro em vários conceitos, que somente se tornaram disponíveis muito mais tarde em alguns sistemas de banco de dados comerciais.

O PostgreSQL é um descendente de código fonte aberto deste código original de Berkeley, que é compatível com a linguagem SQL ANSI, uma linguagem de consulta declarativa, que é a padrão para SGBD relacionais. Apresenta suporte a chave estrangeira, operações join, views, triggers, procedures, entre outros. Além

disso, suporta a maioria dos tipos de dados da especificação *SQL*[6]:2008, dentre eles estão: INTEGER, VARCHAR, BOOLEAN, NUMERIC, CHAR, DATE, INTERVAL e TIMESTAMP.

A plataforma de desenvolvimento utilizada para manusear o PostgreSQL foi o pgAdmin, uma ferramenta open source bastante popular no mercado. Ela pode ser utilizada em sistemas como o Linux, Mac OSX, Windows, entre outros e gerencia as versões do PostgreSQL acima da 7.3.

3 RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados obtidos e as discussões levantadas a partir da aplicação dos testes.

3.1 AMBIENTE E APLICAÇÃO DE TESTES

Os testes foram realizados em um computador com processador Intel Core i7-3517U CPU de 2.4 GHz, 8 GB de RAM, com o sistema operacional Windows 10 Pro de 64 bits. A versão do PostgreSQL usada foi a 9.3 e a do MongoDB foi a 3.2.3

Para implementar os testes, a linguagem Java (Java 2016) foi utilizada juntamente com os respectivos drivers necessários para a conexão com os bancos. Para o PostgreSQL, foi utilizado o driver PostgreSQL JDBC e para o MongoDB foi utilizado o driver Java Driver MongoDB.

Para medir o tempo de execução, foi usado o método *System.currentTimeMillis()*[7] da linguagem Java.

Na figura 4 é representada a estruturada do código de conexão da linguagem JAVA com o banco de dados MongoDB:

Figura 4. Código de conexão com o MongoDB.

```
1 MongoClient mongoClient = new MongoClient("localhost", 27017);
2 DB db = mongoClient.getDB("TCC");
```

Fonte: O Autor

[6] Linguagem de Consulta Estruturada.

[7] Método que retorna a hora atual.

São criadas as variáveis, as quais foram utilizadas para a conexão (*mongoClient* e *DB*). É feita a conexão no endereço *localhost*, na porta 27017 (Linha 1), padrão do MongoDB e então é recuperado o banco *TCC* (Linha 2).

O código de conexão da linguagem JAVA com o banco de dados PostgreSQL é representado na Figura 5.

Figura 5. Código de conexão com o PostgreSQL

```
1 String url = "jdbc:postgresql://localhost:5432/TCC1";
2 String usuario = "postgres";
3 String senha = "postgres";
4 Class.forName("org.postgresql.Driver");
5 con = DriverManager.getConnection(url, usuario, senha);
```

Fonte: O Autor

Na primeira linha é definida a URL de conexão no endereço localhost e porta 5432, as próximas duas linhas são as linhas de usuário e senha do banco. Após, a quarta linha é o registro do driver do PostgreSQL através do método *Class.forName()*[8]. Na quinta linha é feita a conexão através da variável con.

As métricas de comparação foram definidas com base nas métricas utilizadas no artigo Comparação de Performance entre PostgreSQL e MongoDB (POLITOWSKI e MARAN, 2014). Os testes foram divididos em cinco categorias: busca com a utilização de índices, busca sem a utilização de índices, inserção, atualização e exclusão. Os comandos de busca foram executados em loops de 1, 1.000, 10.000 repetições, bem como os comandos de inserção, ambos foram efetuados três vezes cada, sendo considerada a média aritmética dos mesmos. As demais operações (atualização e exclusão), foram realizadas em apenas uma execução, sem loops.

3.2 MODELOS DO ESTUDO DE CASO

O banco de dados AgroDB (LAZZARETTI, 2013) foi utilizado para realização dos testes, em específico, o conjunto de dados meteorológicos já existente. A base

[8] Método que inicializa automaticamente o driver do banco PostgreSQL.

de dados utilizada no PostgreSQL, possui aproximadamente 89.000.000 (oitenta e nove milhões) de registros sendo que 45.000.000 (quarenta e cinco milhões) desses dados foram transferidos para a base de dados no MongoDB, para que os testes pudessem ser realizados. Nas Figuras 6, e 7 estão representados os códigos utilizados para que fosse efetuada a transferência dos dados do banco de dados PosgreSQL para o banco o banco de dados MongoDB.

Figura 6. Consulta dos dados no PostgreSQL.

```
1  ConexaoPostgreSQL c = new ConexaoPostgreSQL();
2  c.conectar();
3
4  ResultSet res = c.executaConsulta("SELECT * FROM weather_data_variable");
5  List<Objeto> lista = new ArrayList();
6
7  Objeto o = null;
8      while (res.next()) {
9            o = new Objeto(res.getInt("id"),
10                   res.getString("data_type"),
11                   res.getDate("date"),
12                   res.getTime("time"),
13                   res.getDouble("data_value"),
14                   res.getInt("var_id"),
15                   res.getString("var_description"),
16                   res.getInt("weather_id"),
17                   res.getInt("weather_station"),
18                   res.getDate("weather_date_import"));
19           lista.add(o);
20       }
```

Fonte: O Autor

Nas linhas 1 e 2 é feita a conexão ao banco PostgreSQL, as linhas 4 e 5 representam respectivamente a execução do comando de consulta e a criação da lista onde ficaram os dados antes de serem inseridos no MongoDB. Da linha 8 a 19 é feita a inserção dos dados consultados anteriormente, na lista.

Figura 7. Inserção dos dados no MongoDB.

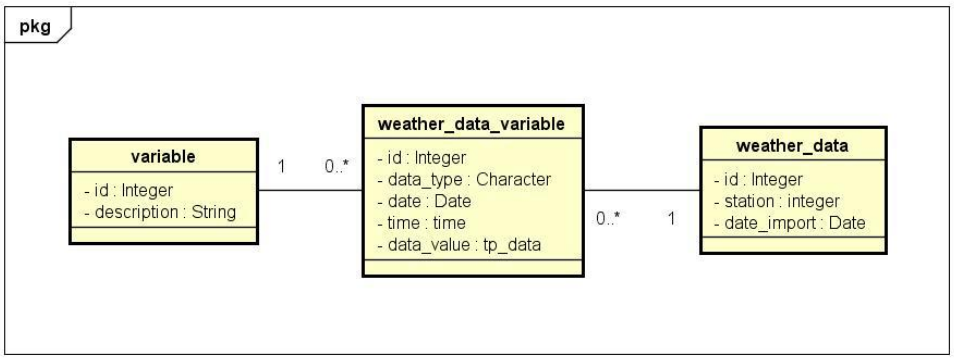
```
1  ConexãoMongo cm = new ConexãoMongo();
2  cm.conectar();
3  DBCollection coll = cm.getColecao("dados");
4  for (int i = 0; i <= lista.size(); i++) {
5      BasicDBObject doc = new BasicDBObject();
6      doc.put("id", lista.get(i).getId());
7      doc.put("data_type", lista.get(i).getData_type());
8      doc.put("date", lista.get(i).getDate());
9      doc.put("time", lista.get(i).getTime());
10     doc.put("data_value", lista.get(i).getData_value());
11     BasicDBObject variable = new BasicDBObject();
12     variable.put("id", lista.get(i).getVar_id());
13     variable.put("description", lista.get(i).getVar_description());
14     doc.put("variable", variable);
15    BasicDBObject weather_data = new BasicDBObject();
16    weather_data.put("id", lista.get(i).getWeather_id());
17    weather_data.put("station", lista.get(i).getWeather_station());
18    weather_data.put("date_import", lista.get(i).getWeather_date_import());
19    doc.put("weather_data", weather_data);
20    coll.insert(doc);
```

Fonte: o Autor

Nas linhas 1 e 2 é feita a conexão ao banco MongoDB, a linha 3 representa o comando que recupera a coleção *dados,* onde serão inseridos os dados recuperados anteriormente. Da linha 5 a 19 é criado e estruturado o documento que será inserido. Por fim, na linha 20, os dados são inseridos no documento, populando assim o banco de dados no MongoDB.

A estrutura do banco no PostgreSQL foi estabelecida através do diagrama de classe, apresentado na Figura 8.

Figura 8. Diagrama de classes.



Fonte: O Autor

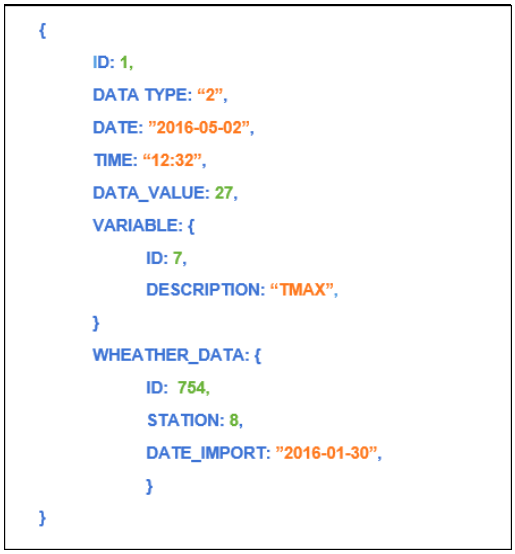
A tabela *variable* possui um atributo identificador (id) e um atributo para descrever o valor do registro (*description*), neste campo encontram-se valores como por exemplo: temperatura, radiação solar, latitude, entre outros. Já na tabela weather_data estão presentes, além do atributo identificador, os atributos *station*, onde é armazenado o número da estação onde dado registro foi coletado e o atributo *date_import* que representa a data de importação do registro.

A tabela *weather_data_variable* possui os seguintes atributos:

1. *Id* – atributo identificador do registro no banco;
2. *Data_type* – atributo que faz referência ao tipo de dado armazenado no atributo data_value;
3. *Date* – data em que o registro foi armazenado no banco;
4. Time – hora em que o registro foi armazenado no banco;
5. Data_value – campo de tipo abstrato "tp_data", “com ele, é possível armazenar em um mesmo campo dados com formatos diferentes” (LAZZARETTI, 2013. p, 43);

A estrutura do documento no MongoDB foi construída utilizando as 3 tabelas do banco presentes no PostgreSQL (Figura 9).

Figura 9. Estrutura do documento MongoDB.



Fonte: O Autor

Na modelagem orientada a documentos, a tabela weather_data_variable passou a ser a *collection* e seus registros, os documentos. Já as tabelas variable e weather_data estão inseridas em cada documento como subdocumentos.

3.3 COMANDOS DE TESTE

Os testes foram divididos em cinco categorias, descritas a seguir:

1. Busca com a utilização de indices;
2. Busca sem a utilização de indices;
3. Inserção;
4. Atualização;
5. Exclusão.

A Figura 10 apresenta o código para realizar as operações no banco de dados PostgreSQL.

Figura 10. Código para execução das consultas.

```
1  long tempoInicial = System.currentTimeMillis();
2
3  ConexaoPostgreSQL c = new ConexaoPostgreSQL();
4  c.conectar();
5
6  c.executaConsulta("<QUERIES AQUI>");
7
8  System.out.println("TEMPO: "
9          + (System.currentTimeMillis() - tempoInicial));
10
11 c.desconectar();
```

Fonte: O Autor

Na primeira linha, é criada a variável *tempoInicial*, que através do método JAVA para a captura do horário em que foi iniciada a execução da aplicação. As linhas 3 e 4 são respectivamente a linha onde é criada a instância da classe de conexão e a linha onde a conexão é realizada. Na sexta linha, em *<QUERIES AQUI>* são colocadas as consultas na linguagem SQL pura, sem frameworks. A

linha 8 e a linha 9 representam o cálculo do tempo de execução da aplicação. E por fim a linha que encerra a conexão com o banco.

Para realizar as operações no banco de dados MongoDB, o código foi estruturado da seguinte maneira (Figura 11):

Figura 11. Código para execução das consultas.

```
1  long tempoInicial = System.currentTimeMillis();
2
3  ConexãoMongo cm = new ConexãoMongo();
4  cm.conectar();
5  DBCollection colecao = cm.getColecao("dados");
6
7  BasicDBObject query = new BasicDBObject();
8  query.put("<CHAVE>", "<VALOR>");
9
10 colecao.<MÉTODO>(query);
11
12 System.out.println("TEMPO: " +
13         (System.currentTimeMillis() - tempoInicial));
```

Fonte: O Autor

Na primeira linha, é criada a variável para a captura o tempo em que foi iniciada a execução da aplicação. A terceira e quarta linhas são as linhas criadas para a realizar a conexão. A quinta linha é onde é feita a recuperação da coleção *dados*. Na sétima e oitava linhas são inseridas as queries que serão executadas. E na décima linha são executadas as operações, sendo que em *<MÉTODOS>* são colocados os métodos que serão utilizados para cada operação (*find, update, remove* e *insert*).

As **inserções** foram feitas englobando todas as tabelas no PostgreSQL um documento completo e seus subdocumentos com todos os dados preenchidos no MongoDB. Os dados foram inseridos em duas execuções, na primeira foram inseridos 1.000 (mil) registros e na segunda 10.000 (dez mil) registros, sendo inseridos no total 11.000 (onze mil) registros em ambos os bancos. Na Figura 12, estão os comandos utilizados para inserção.

Figura 12: Comandos de inserção.

POSTGRESQL

```
INSERT INTO weather_data_variable
(id,weather_data,  variable, data_type,
date, data_value.value_double)
VALUES  (1000000000 ,106,2,'3','2016-05-30',80);
```

MONGODB

```
db.collection.insert({id : "1000000000", data_type: "2",
date: "2016-05-31', time: "12:32", data_value: 27,
variable: {id: 7, description: "TMAX" },
weather_data: {id: 765, station: 4,
date_import: "2016-05-31"} })
```

Fonte: O Autor

Os resutados dos comandos de inserção são representados nas tabelas a seguir (Tabela 1 e Tabela 2):

Tabela 1 - Inserções no Postgresql.

REGISTROS	1.000	10.000
TEMPO (seg)	2,015	15,993

Fonte: O Autor (2016).

Tabela 2 - Inserções no MongoDB.

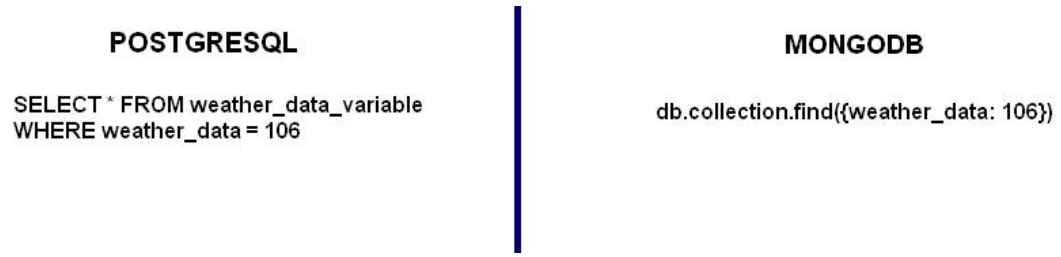
REGISTROS	1.000	10.000
TEMPO (seg)	1,608	5,663

Fonte: O Autor (2016).

As Tabelas 1 e 2 representam os resultados das operações de inserção, onde ambos os bancos de dados obtiveram bons resultados. No PostgreSQL, em aproximadamente 2 segundos foi realizada a inserção de 1.000 (mil) registros e com cerca de 16 segundos pode-se efetuar a inserção de 10.000 (dez mil) registros. O banco MongoDB apresentou tempos de inserção ainda menores. Sendo assim, em aproximadamente 1 segundo e meio foram inseridos 1.000 (mil) registros e em 5 segundos e meio 10.000 (dez mil) registros.

A **busca utilizando índices** consiste na busca de todos os registros/documentos da tabela/coleção *weather_data_variable* que possuem 106 como o valor do campo *weather_data (weather_data.id* no caso do MongoDB*).* Na Figura 13, estão os comandos utilizados para a busca.

Figura 13. Comandos de busca com indices.



Fonte: O Autor

Os resultados das buscas no PostgreSQL estão na Tabela 3 e do MongoDB na Tabela 4.

Tabela 3 – Busca com índices no PostgreSQL.

EXECUÇÕES	1	1.000	10.000
TEMPO (seg)	0,410	7,695	54191,975

Fonte: O Autor.

Tabela 4 – Busca com índices no MongoDB.

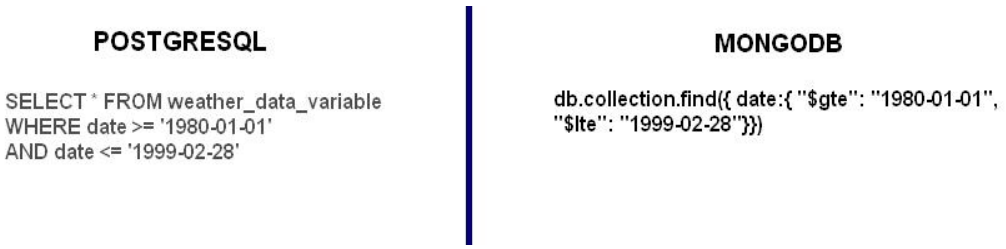
EXECUÇÕES	1	1.000	10.000
TEMPO (seg)	0,350	0,613	4,020

Fonte: O Autor.

Nas Tabelas 3 e 4 estão os resultados da busca por registros utilizando índices, onde o PostgreSQL obteve um melhor resultado em relação ao MongoDB quando a consulta foi executada apenas uma vez. O PostgreSQL executou a consulta 1.000 (mil) vezes em *7 segundos e meio*, e o MongoDB em *meio segundo*. A mesma operação, com 10.000 (dez mil) execuções, foi finalizada após *15 horas* no PostgreSQL, já no MongoDB, foram necessários apenas *4 segundos.*

Já na **busca sem índices** o resultado deve retornar todos os registros/documentos que possuem o campo *date* com valor entre *1980-01-01* e *1999-12-31*. Na Figura 14, estão os comandos utilizados para busca.

Figura 14. Comandos de busca sem índices.



Fonte: O Autor.

Os parâmetros *$gte*[9] e *$lte*[10] no comando do MongoDB são operadores de comparação, selecionam os documentos onde o valor do campo é maior do que ou igual a, e menor ou igual a (isto é, >= e <=) determinado valor.

Os resultados nas estão Tabelas 5 e 6:

Tabela 5 – Busca sem índices no PostgreSQL.

EXECUÇÕES	1	1.000	10.000
TEMPO (seg)	3,125	8,791	66555,960

Fonte: O Autor.

Tabela 6 – Busca sem índices no MongoDB.

EXECUÇÕES	1	1.000	10.000
TEMPO (seg)	0,466	0,926	3,500

Fonte: O Autor.

O MongoDB obteve melhores resultados que o PostgreSQL, quando índices não foram utilizados. Foram 18 horas para finalizar a consulta 10.000 (dez mil) vezes, sendo que no MongoDB a mesma consulta foi executava em *3 segundos e meio.*

Tanto as consultas com índices, quanto as que os índices não foram utilizados, retornaram aproximadamente 1.100.000 (um milhão e cem mil) registros/documentos por execução.

[9] Greater than or equals (maior ou igual).

[10] Less Than or equals (menor ou igual).

Nas **atualizações**, foram modificados 1.200.000 (um milhão e duzentos mil) registros/documentos. A operação atualizou o campo *data_value* para 0.95 em todos os registros/documentos da tabela/coleção weather_data_variable onde o valor do campo *variable* fosse igual a 2 e o valor do campo *weather_data* estivesse entre 98 e 109. Na Figura 15 estão os comandos utilizados para atualização.

Figura 15. Comandos de atualização.

POSTGRESQL

```
UPDATE weather_data_variable
SET data_value.value_double = 0.95
WHERE variable = 2
AND weather_data >= 98
AND weather_data <=109
```

MONGODB

```
db.collection.update({
"weather_data.id": {$gte: 98, $lte: 109},
"variable.id" : 2}, {$set:{"data_value" : 0.95}})
```

Fonte: O Autor

Os resultados das atualizações estão presentes nas Tabelas 7 e 8:

Tabela 7 – Atualizações no PostgreSQL.

EXECUÇÕES	1
TEMPO (seg)	725.340

Fonte: O Autor.

Tabela 8 - Atualizações no MongoDB.

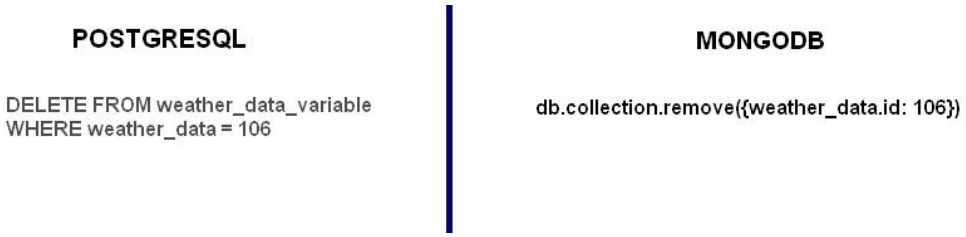
EXECUÇÕES	1
TEMPO (seg)	300,540

Fonte: O Autor.

As Tabelas 7 e 8 representam os resultados das atualizações. Os registros foram atualizados em 12 minutos no PostgreSQL e em 5 minutos no MongoDB.

Nos testes de **exclusão** foram apagados cerca de 1.100.000 (um milhão e cem mil) registros/documentos. A operação excluiu todos os registros/documentos que possuíam 106 como o valor do campo *weather_data (weather_data.id* no caso do MongoDB*).* Na Figura 16 estão os comandos utilizados para exclusão.

Figura 16. Comandos de exclusão.



Fonte: O Autor

As Tabelas 9 e 10 apresentam os resultados dos comandos de exclusão.

Tabela 9 - Exclusões no PostgreSQL.

EXECUÇÕES	1
TEMPO (seg)	163,932

Fonte: O Autor.

Tabela 10 - Exclusões no MongoDB.

EXECUÇÕES	1
TEMPO (seg)	61,212

Fonte: O Autor.

Em *2 (dois)* minutos a operação foi concluída no banco PostgreSQL e em *1 (um) minuto* a mesma operação foi efetuada no MongoDB.

4 CONCLUSÃO

O presente trabalho apresentou uma comparação de desempenho do tempo de execução de instruções que incluem, removem, alteram e consultam dados nos bancos de dados SQL e NoSQL

Utilizando os dois bancos de dados na configuração padrão, sem ajustes de otimização e baseados no ambiente de testes proposto, pode-se concluir que o MongoDB obteve melhores resultados. Isso se dá pelo fato de que os bancos NoSQL terem sido desenvolvidos para suprir a demanda por performance.

Portanto, o NoSQL não veio para substituir os bancos de dados tradicionais, e sim para servir de possível opção para pontos aonde estes possuem maiores

dificuldades. Bancos de dados NoSQL e Relacionais utilizam paradigmas diferentes e, por sua vez, possuem diferentes finalidades.

Acredita-se que pode-se atribuir a melhor performance do SGBD MongoDB, mostrada neste trabalho, pelo fato de sua modelagem estar de acordo com o propósito e os padrões das aplicações NoSQL, evitando a existência de junções.

Como trabalho futuro, é interessante realizar e avaliar testes de comparação de performance utilizando outras alternativas de modelagem para o modelo NoSQL, verificando assim qual a sua influência no resultado final dos testes.

ABSTRACT

In recent years there has been a significant increase in the use of non-conventional applications. The use of NoSQL databases (Not Only SQL) has increased significantly, mainly due to the increased volume of data generated and the need to meet the demand for scalable web applications. For this reason, the traditional models of databases have limiting factors, they do not have a lot of flexibility in structuring and are limited as to the processing of large volumes of data. However, NoSQL databases are recent and still little known, which can be a hindrance when choosing one of these for managing a database. This work done, based on a set of structured data, a study on the NoSQL database MongoDB and a comparison, through various command handling with the DBMS. As a result, could be concluded that MongoDB database shown to have improved performance when subjected to the proposed testing.

Keywords: NoSQL. Data Bases. Data, Scalability

REFERÊNCIAS

BATISTA, F. et al. **MongoDB: Banco de dados orientado a documento**. 2012. Disponível em: <http://MongoDB.machinario.com/>. Acesso em: 5 abr. 2016.

BSON. **Binary JSON**. Disponível em <http://bsonspec.org/> Acesso em maio de 2012.

CUNHA, T. M. de A. **Escalabilidade de Sistemas com Banco de Dados NoSQL: um Estudo de Caso Comparativo com MongoDB e MySQL**. 2011. 85 f. Centro Universitário da Bahia – Estácio, Salvador.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados.** 8. ed. Rio de Janeiro: Elsevier, 2004.

ELMASRI, R; NAVATHE, S, B. **Sistemas de Bancos de dados**. 6. ed. São Paulo: Addison Wesley, 2011.

EVANS, E. **NoSQL,** 2009. Disponível em: < http://blog.symlink.com/2009/05/12/nosql_2009.html> Acesso em: 22 mai, 2016.

JSON. **Introducing JSON**. Disponível em http://www.json.org/. Acesso em 22 mai. 2016.

JSON Schema. Disponivel em : <http://json-schema.org/examples.html> Acesso em jun. 2016.

LAZZARETTI, T. A. **Integração de Banco de Dados e Modelos de Simulação de Culturas Para Estimar o Impacto de Mudanças do Clima no Rendimento de Grãos e na Severidade da Giberela em Trigo**. 2013. 174f. Tese (Doutorado em Fitopatologia) - Universidade de Passo Fundo, Passo Fundo, RS, Maio. 2013.

LEAVITT, N. **Will NoSQL Databases Live Up to Their Promise?** 2010. Computer, 12. ISSN: 0018-9162, p. 12 – 14

LENNON, J. (2013). **MongoDB.** Disponível em: http://alfavillecode.blogspot.com.br/2013/01/MongoDB.html. Acesso em: 06 mai, 2016.

MONGODB. Disponível em: <https://www.mongodb.com/>. Acesso em 10 mar. 2016.

POLITOWSKI, C.; MARAN, V . **Comparação de Performance entre PostgreSQL e MongoDB**.. In: X Esc. Reg. de Banco de Dados (ERBD), 2014, São Francisco do Sul, SC. Disponível em < https://www.researchgate.net/publication/261871960_Comparacao_de_Performance_entre_PostgreSQL_e_MongoDB > Acesso em: 22. Jun. 2016.

POPESCU, A. **Presentation: NoSQL at CodeMash - An Interesting NoSQL categorization.** Disponível em http://nosql.mypopescu.com/post/396337069/presentation-nosql-codemashan-interesting-nosql. Acesso em 12 Jun. 2016.

POSTGRESQL. Disponível em: <http://www.postgresql.org/>. Acesso em 10 mar. 2016.

SADALAGE, P; FOWLER, M. **NoSQL Essentials.** 1. ed. São Paulo: Novatec, 2013.

SILBERSCHATZ, A; KORTH, H F.; SUDARSHAN, S. **Sistema de Bancos de Dados**. 3. ed. São Paulo: Makron Books, 1999.

SILBERSCHATZ, A.; KORTH, H.. F., SUDARSHAN, S. **Sistema de banco de dados**, Rio de Janeiro: Editora Elsevier, 2006. ISBN: 978-85-352-1107-8.

STROZZI. C. (2007). [on-line]. Disponível em <http://www.strozzi.it/cgibin/CSA/tw7/I/en_US/nosql/Home%20Page>. Acesso em: 18 abr, 2016.

SUISSA, J. **Introdução ao NoSQL**. NoSQL Br. 2010. Disponível em: <http://www.nosqlbr.com.br/introducao-ao-nosql.html.>. Acesso em: 20 abr 2016.

VOGELS, W. (2009). **Eventually consistent. Commun**. ACM, 52(1). P. 40–44. Disponivel em: <http://cs.stanford.edu/people/chrismre/cs345/rl/eventually-consistent.pdf> Acesso em: 21. Mai 2016.