Análise comparativa de desempenho de aplicação Java com persistência em Banco de Dados MySQL e MongoDB

Gerson Viana Marques[1]; Claudio R. de Medeiros[1]; Jeferson Queiroga Pereira[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)

BR 405KM 154, S/N, Bairro Chico Cajá Pau dos Ferros – RN – Brasil

{claudiorodrigozh,jefersonqueiroga,vianagerson3029}@gmail.com

Abstract. *Doubts between SQL and NoSQL relational databases permeate the computing environment, Making it difficult to choose the database model to use in an application. To assist in this choice this paper describes a brief theoretical and practical comparison between the NoSQL Document Oriented database models and the SQL Relational databases, based on a real application development for enrollment management of an institution; Where the results show that the NoSQL database showed better results in all tested conditions.*

Resumo. *Dúvidas entre os bancos Relacionais SQL e os Orientados a Documentos NoSQL permeiam o cenário da computação, tornando difícil a escolha do modelo de banco para se utilizar em uma aplicação. Para auxiliar nessa escolha este trabalho descreve uma breve comparação teórica e também prática entre os modelos de banco de dados Orientado a Documentos NoSQL e os bancos de dados Relacionais SQL, baseado em um desenvolvimento de uma aplicação real para gerenciamento de matrículas de uma instituição; onde os resultados mostram que o* banco de dados NoSQL mostrou melhores resultados em todas as condições testadas.

Palavras chave: *Banco de Dados, Desempenho, Aplicação Java.*

1. Introdução

Atualmente o mundo trabalha com grandes volumes de dados. As empresas sobrevivem e mantêm tudo armazenado de forma segura para utilizarem posteriormente ao seu favor. O alto 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].

Com o passar dos tempos ampliou-se a utilização dos modelos de banco de dados relacionais, de modo que propiciou sua aplicação em diversos sistemas de armazenamento de dados. Porém, com o surgimento de novos paradigmas de armazenamentos, e o aumento maciço de dados de certas organizações, essa hegemonia vem perdendo espaço. Um desses novos paradigmas é o banco de dados Orientado a Documentos NoSQL, que surgiu como uma solução para o armazenamento e processamento de dados na *Web* 2.0 [Chang et al 2006], [Decandia et al 2007], [Cooper et al 2008]. Com a crescente popularização das redes sociais, a geração de conteúdo por dispositivos conectados faz com que o trabalho de armazenamento de dados com o objetivo de utilizá-los em ferramentas analíticas comece a esbarrar nas questões de

escalabilidade e custos de manutenção desses dados [Ianni e Vinicius 2013]. Mesmo com a necessidade e com o aumento da popularidade, a tecnologia NoSQL ainda é contestada em algumas áreas. Dessa forma, este trabalho tem como objetivo relatar um pouco sobre essa nova tecnologia e realizar uma comparação de desempenho e implantação entre os bancos Orientados a Documentos e os Relacionais. A ideia é apresentar dois bancos, um de cada modelo, que são gratuitos e populares entre os usuários, auxiliando assim um programador, ou uma equipe na escolha do melhor modelo de banco de dados.

Para esse trabalho, foram escolhidos o banco relacional MySQL e o Orientado a Documentos MongoDB. Com intuito de obter os resultados da pesquisa foi criada uma aplicação para gerenciamento de matrículas, que foi submetida a migração de dados de um arquivo TXT (arquivo de texto) com média de 30 mil registros de alunos do IFRN (Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte), contendo o nome, matrícula, curso, de cada um aluno. Uma aplicação simples, contendo um esquema de banco de dados simples, onde já foi bastante perceptivo a diferença entre ambos os bancos. Após essa etapa foi possível realizar as comparações, analisar o desempenho e tirar as conclusões.

O artigo está estruturado como descrito a seguir. Na Seção 2, são apresentados trabalhos relacionados e seus resultados. Na Seção 3, são apresentadas as características dos bancos de dados testados. A Seção 4, apresenta a etapa de preparação do ambiente para a execução dos testes. Na Seção 5, são apresentados os resultados. Por fim, a Seção 6 apresenta as conclusões finais deste trabalho.

2. Trabalhos Relacionados

Um fator importante para a criação deste trabalho é a pouca quantidade de trabalhos relacionados. Politowski e Maran (2014) fazem uma comparação de performance entre MongoDB e PostgreSQL, mostrando diferenças e fazendo testes de inserção e busca de dados. Como resultado houve uma grande diferença de desempenho, em favor do banco de dados MongoDB, principalmente quando o número de registros no banco de dados e grande.

Claudino, Souza e Salgado (2015) descrevem os modelos de bancos relacionais e não-relacionais, em termos de conceitos e estruturas, e apresenta um estudo comparativo apontando possíveis mapeamentos conceituais entre eles.

Li e Manoharan (2013) em seus trabalhos, faz testes com vários bancos NoSQL e com o banco relacional MS SQL Server [Microsoft 2018], fazendo testes de escrita, remoção, leitura e instanciação. Com resultados obtidos, foi possível afirmar que os bancos NoSQL foi súpero ao relacionais, porém, apresentou algumas variações quando submetidas a outras ferramentas.

Como citado anteriormente, encontra-se trabalhos relacionados que apresentam comparativos entre os modelos abordados neste artigo, mas são poucos e ainda não são hábeis para total escolha de um modelo. Este trabalho irá agregar experiências para contribuir em conjunto com os demais na escolha do modelo adequado para cada situação.

3. Bancos de dados utilizados para a comparação

Abaixo será apresentado os modelos de banco de NoSQL MongoDB e o modelo Relacional MySQL, elencando suas características. Tendo em vista que ambos são gratuitos e bastante populares entre os usuários. Vale também ressaltar que segundo a DB-Engines (2017), ambos estão no *top five* dos bancos mais utilizados.

3.1. MongoDB

O MongoDB é um banco de dados NoSQL de código aberto sob a licença GNU AGPL (Affero General Public License) versão 3.0, escrito em C++ e Orientado a Documentos. Apesar de ter sido iniciado em 2007, só foi concluído em 2009. Diversas linguagens e plataformas já possuem *drivers* para o MongoDB. Além disso, o MongoDB possui binários para diversas plataformas como Windows, Mac OS X, Linux e Solaris. Atualmente ele é um dos mais populares bancos de dados NoSQL e está na versão 3.6.

A persistência dos documentos no MongoDB é feita no formato BSON (lista ordenada de elementos), que são objetos JSON (acrônimo para "*JavaScript Object Notation*", é um formato leve para intercâmbio de dados computacionais) binários. BSON por sua vez, suporta estruturas como arrays e *embedded objects* assim como JSON.

Os documentos podem ser armazenados em coleções, onde serão feitas as operações de busca e indexação. Para persistência, MongoDB usa arquivos mapeados em memória, deixando o gerenciador de memória virtual do sistema operacional decidir quais partes irão para o disco e quais ficam na memória. Isto faz com que o MongoDB não tenha controle sobre o momento onde os dados são escritos no disco [Orend 2010].

3.2. MySQL

O MySQL é um banco relacional de código aberto, escrito em C++ desenvolvido e distribuído com licenças GNU/GLP (*General Public Licence*). Além do programa, o seu código-fonte é disponibilizado para que possa ser alterado de acordo com as necessidades dos seus usuários [Milani 2007].

O MySQL possibilita diversos tipos de tabela para armazenamento dos dados, tendo em conta que cada tipo tem suas próprias características. Dependendo da plataforma em que seja usado, as tabelas poderão armazenar grandes volumes de dados. Utilizando as tabelas do tipo *InooDB* o armazenamento pode ser equivalente a *Terabytes* de tamanho [Ricardo 2013].

Como já sabemos o MySQL trabalha com linguagem SQL (*Structured Query Language*) para fazer requisições ao banco. É um excelente banco de dados para ser utilizado em aplicações que trabalham com grandes volumes de dados, além de ser compatível com diversos sistemas operacionais vem em constante evolução ao decorrer de cada nova versão lançada [Ricardo 2013].

4. Ambiente de testes

Os testes de comparação entre os modelos foram feitos em um computador com processador Intel Core i3-4005U CPU de 1.7 GHz com 4 GB de RAM e HD de 500 GB com o sistema operacional Linux Ubuntu 16.04 de 64 bits. A versão do MySQL usada foi a 5.7 a do MongoDB foi a 3.2.

A implementação foi realizada com a linguagem Java, que segundo o índice da

Comunidade de Programação TIOBE (2018), é a linguagem de programação mais prestigiada da atualidade. Para representação gráfica foi utilizado a API (Application Programming Interface) do JavaFX, que atualmente é a API mantida pela Oracle [Oracle 2018], sendo assim atualizada constantemente, propiciando melhor suporte e usabilidade. Para utilização dos bancos de dados foram necessários a aplicação dos *drives* MongoDB Java Driver 3.6.3 e MySQL Connector/J 5.1.46.

A estrutura do banco montada é a mesma para ambos os bancos e foi estabelecida de acordo com o diagrama de classe apresentado na Figura 1. O diagrama pode ser entendido como uma modelagem em um sistema de matrícula de alunos.

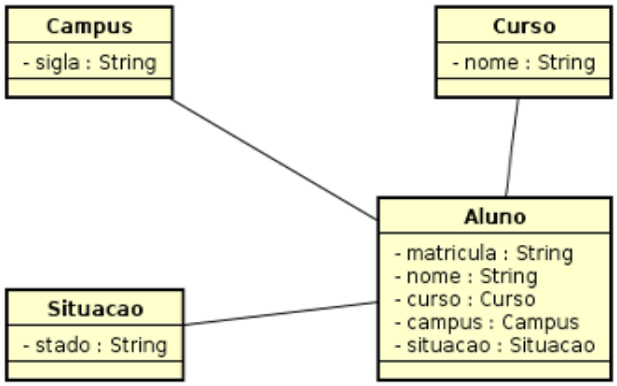

Figura 1. Diagrama de Classes do caso de tese.

A partir do diagrama acima, foi adotado a seguinte arquitetura: A aplicação Java ler o arquivo TXT, prepara os dados e insere nos respectivos bancos. Essa representação pode ser melhor compreendida na Figura 2.

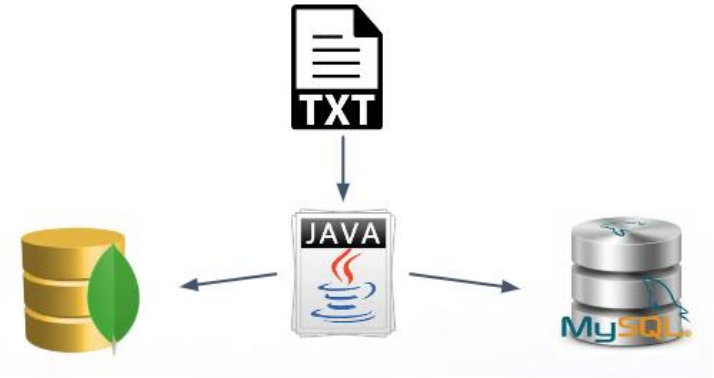

Figura 2. Arquitetura do caso de teste.

A avaliação é realizada levando em consideração a implementação, e o comportamento da aplicação quando submetida a cada operação. Para calcular o tempo médio de cada operação, o método Java `currentTimeMillis` foi adotado. Este método retorna em milissegundos o tempo gasto para execução de uma tarefa [Oracle 2017].

4.1. Operações com MongoDB

Para as operações com MongoDB foi adotada a seguinte estrutura.

```
mongo =  Mongo("localhost", 27017);
db = mongo.getDB("BancoIFRN");
# <querys>
mongo.close();
```

Figura 3. Estrutura de código Java adotada para operações com o MongoDB.

É criada uma instância do MongoDB. Após, é criado/recuperado o banco de dados de nome BancoIFRN. Logo após a execução da query da instância é fechada.

Para fazer inserção é criado um Objeto do tipo BasicDBObject que através do método put(chave,valor), monta o documento no formato JSON. Após montado o objeto é inserido através da função insert() dentro de uma coleção de documentos (*collection*). Exemplo da inserção de um Aluno:

```
BasicDBObject bbdo = new BasicDBObject();
bdbo.put("nome", "fulano da silva");
bdbo.put("matricula", "2015464");
bdbo.put("idCurso", ObjectId("1234242sddsww3ae"));
bdbo.put("idCampus", ObjectId("234rdsf2s34w3ae"));
bdbo.put("idSituacao", ObjectId("56fgty78113ae"));
alunos.insert(bdbo);
```

Figura 4. Exemplo de inserção de um objeto na sua respectiva coleção.

Para listar um documento é criado um DBcursor que recebe um coleção através chamada do método find() em uma coleção existente. Exemplo de uma listagem de Campus.

```
DBCursor cursor = campus.find();
while (cursor.hasNext()) {
    BasicDBObject bObject = (BasicDBObject) cursor.next();
        cmps.add(
           new Campus( bObject.getString("_id"), bObject.getString("sigla"))
        );
}
```

Figura 5. Exemplo de listagem de uma coleção.

4.2. Operações com MySQL

Para as operações no banco de dados MySQL, o código foi estruturado da seguinte maneira:

```
conn = (Connection) DriverManager.getConnection(
    "jdbc:mysql://localhost/"+ BancoIFRN, root, root
);
# <querys>
conn.close();
```

Figura 6. Estrutura de código Java adotada para operações com MySQL.

Na primeira linha a conexão é feita passando o banco, usuário e senha. Na linha dois são colocadas as consultas SQL, e após executadas é fechada a conexão.

A inserção é feita através de uma String SQL formada com os gets do objeto e executada através do método executeUpdate() de um objeto *Statement*. Exemplo da inserção de um Curso:

```
Curso curso = new Curso (" curso ");
sql = "INSERT INTO Curso(nome) VALUES('" + curso.getNome() + "')";
stm = conn.getConn().createStatement();
stm.executeUpdate(sql);
```

Figura 7. Exemplo de inserção de um objeto da sua respectiva tabela.

A busca por uma lista de objetos em uma base de dados MySQL é mais complexa. É criado como nas anteriores uma *String* com o SQL correspondente a busca. Além disso é preciso criar um objeto do tipo *ResultSet*, que receberá o resultado do banco. Exemplo de uma listagem de Campus:

```
List<Campus> campus = new ArrayList<>();
sql = "SELECT * FROM Campus";
pst = (PreparedStatement) conexao.getConn().prepareStatement(sql);
pst.executeQuery();
rs = pst.executeQuery(sql);
while (rs.next()) {
    String id = String.valueOf(rs.getInt("idCampus"));
    String nome = rs.getString("sigla");
    campus.add(new Campus(id, nome));
};
```

Figura 8. Exemplo de listagem de uma tabela.

5. Resultados

Os resultados foram obtidos a partir da inserção do arquivo de texto em ambos os bancos de dados. Cada linha do arquivo é composta por um registro específico de um aluno. Com isso foi implementada uma aplicação que lê e separa os dados corretamente da linha e o processa, para que eles possam ser adicionadas no banco.

As tabelas 1 e 2 apresentam os resultados dos testes de inserção e de busca feitos no MongoDB e no MySQL, onde os valores estão representados em segundos, levando em consideração todos os tratamentos necessários em relação aos dados.

Tabela 1. Teste de inserção com os bancos MySQL e MongoDB

Banco	100 registros	1.000 registros	30.000 registros
MongoDB	0.023	0.903	47.78
MySQL	4,7	48.39	1.364,701

Na tabela 1, percebe-se que para as inserção de 100, 1000 e 30.000 registro, o banco de dados NoSQL(MongoDB) teve melhores resultados, sendo mais notório para

o teste de 30 mil registros. Na tabela 2, mostra o tempo necessário para listar todos os elementos da tabela utilizada. Tanto para consultas simples de um único objeto quanto para consultas compostas fazendo ligações de todos os objetos o MongoDB obtiver melhores resultados.

Tabela 2. Teste de busca com os bancos MySQL e MongoDB

Banco	Consulta simples	Consulta composta
MongoDB	0.059	0.806
MySQL	0.579	1.654

O banco MongoDB mostrou melhores resultados em termos de velocidade tanto em relação às consultas de recuperação de documentos, como também mostrou um desempenho bastante superior quando submetido a inserção dos dados do arquivo. Já o MySQL, apesar da possível otimização das *querys,* utilizando um código SQL diferente, demonstrou resultados piores e a abaixo do esperado.

6. Conclusão

Persistir dados, e nos retornar quando necessários são propósitos de ambos os bancos. Os Bancos de dados NoSQL e os SQL possuem finalidades diferentes, paradigmas diferentes, mas continuam com o mesmo objetivo. De acordo com os testes realizados, quanto maior a carga de inserções e de consultas a base de dados, sugere-se a utilização do MongoDB, devido mostrar melhor comportamento em relação ao MySQL.

O MongoDB é mais indicado para aqueles sistemas que tenham necessidades maiores de armazenamento e desempenho, mas não indica que devemos utilizá-lo em todas as situações. Toda escolha de banco depende do problema que está sendo submetido.

Os bancos NoSQL não foram criados com o intuito de substituir o SQL, mas para acrescentar e oferecer mais uma opção de banco de dados. Podendo até em uma solução de seu problema, utilizar os dois bancos, aproveitando as vantagens de ambas e sanando os déficits de cada um.

7. Referências

CLAUDINO, Myller; SOUZA, Damires; SALGADO, Ana Carolina (2015). Mapeamentos Conceituais entre os modelos Relacional e NoSQL. **Uma abordagem comparativa**.

CHANG, Fay et al (2018). Bigtable: A distributed storage system for structured data. **ACM Transactions on Computer Systems (TOCS)**, v. 26, n. 2, p. 4, 2008.

ENGINES, D. B (2017). DB Engines ranking. 2017.

GUEDES, Marylene (2018). **SQL vs NoSQL**, qual usar?. Disponível em : <www.treinaweb.com.br/blog/sql-vs-nosql-qual-usar/>. Acesso em: 4 fev. 2018.

IANNI, Vinicius (2013). Introdução aos bancos de dados NoSQL. 2013.

JavaFX (2018). Disponível em : <www.oracle.com/technetwork/pt/java/javafx/>. Acesso em 5 fev. 2018.

LI, Yishan; MANOHARAN, Sathiamoorthy. A performance comparison of SQL and NoSQL databases. In: **Communications, computers and signal processing (PACRIM), 2013 IEEE pacific rim conference on**. IEEE, 2013. p. 15-19.

MEDEIROS, Higor (2018). **Introdução ao MongoDB.** Disponível em : <www.devmedia.com.br/introducao-ao-mongodb/30792> . Acesso em 4 fev. 2018.

MILANI, André (2007). **MySQL-guia do programador**. Novatec Editora, 2007.

MongoDB (2018). Disponível em : <www.mongodb.com>. Acesso em 5 fev. 2018.

MySQL (2018). Disponível em: <www.mysql.com>. Acesso em 5 fev. 2018.

Microsoft (2018). Disponível em : <www.microsoft.com/pt-br/sql-server/sql-server-2016>. Acesso em 18 mar. 2018.

NASCIMENTO, Matheus Bellio (2014). MongoDB: Um Estudo Teórico-Prático do Conceito de Banco de Dados NoSQL.

Oracle (2018). Disponível em: <www.oracle.com>. Acesso em 24 mar. 2018.

OREND, Kai (2010). Analysis and classification of NoSQL databases and evaluation of their ability to replace an object-relational Persistence Layer. **Architecture**, v. 1, 2010.

POLITOWSKI, Cristiano; MARAN, Vinıcius (2014). Comparaç ao de Performance entre PostgreSQL e MongoDB. **X Escola Regional de Banco de Dados. SBC**, p. 1-10, 2014.

TIOBE Index for March 2018 (2018). Disponível em: <www.tiobe.com/tiobe-index>. Acesso em 24 mar. 2018

Ubuntu (2018). Disponível em : <www.ubuntu.com>. Acesso em 5 fev. 2018.

VOGELS, Werner. Eventually consistent (2009). **Communications of the ACM**, v. 52, n. 1, p. 40-44, 2009.