CENTRO UNVERSITÁRIO DE ANÁPOLIS – UNIEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

FERNANDO JOSÉ DE SOUZA
PAULO HENRIQUE ALVES DE SANTANA

ESTUDO DE CASO: ANÁLISE ENTRE BANCO DE DADOS RELACIONAL E NÃO RELACIONAL

ANÁPOLIS - GO
2017

FERNANDO JOSÉ DE SOUZA
PAULO HENRIQUE ALVES DE SANTANA

ESTUDO DE CASO: ANÁLISE ENTRE BANCO DE DADOS RELACIONAL E NÃO RELACIONAL

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de bacharel em Engenharia da Computação do Centro Universitário de Anápolis - UniEvangélica.
Orientadora: Prof. Me. Luciana Nishi.

ANÁPOLIS - GO
2017

"Jamais considere seus estudos como uma obrigação, mas como uma oportunidade invejável para aprender a conhecer a influência libertadora da beleza do reino do espírito, para seu próprio prazer pessoal e para proveito da comunidade à qual seu futuro trabalho pertencer".

Albert Einstein

DEDICATÓRIA

Aos nossos pais, pelo incentivo e pelo apoio constantes, aos nossos amigos e às pessoas com quem convivemos ao longo desses anos, e também para a professora Aline Dayany que nos auxiliou quanto a escolha do tema abordado.

AGRADECIMENTOS

Primeiramente queremos agradecer a Deus, pois sem sua força não teríamos conseguido. Agradecemos também aos nossos familiares e aos nossos amigos que foram pessoas importantes durante esse longo processo de aprendizagem. Agradecemos também a instituição a qual junto com nossa orientadora nos proporcionou esse sentimento de realização e sucesso. E eu Fernando, quero fazer menção do meu pai que sempre me apoiou e que foi o meu espelho de homem e que infelizmente já não está mais entre nós.

RESUMO

Devido à quantidade de dados e a necessidade de manipulação dos mesmos, o crescente avanço tecnológico disponibiliza diversos modelos de banco de dados disponíveis no mercado, tendo cada um deles suas próprias especificidades e aplicabilidades. O objetivo deste trabalho é realizar a comparação entre o sistema de banco de dados relacional, tradicionalmente utilizado para armazenamento de dados, e o sistema de banco de dados NoSQL, que surgiu como uma solução alternativa para gerenciar maiores quantidades de dados, baseando-se em seus conceitos, aplicação e armazenamento dos dados. Ao final, demonstra-se através de um estudo a análise de desempenho de cada banco, e também se comprova através dos resultados, que para a aplicação abordada neste trabalho o modelo relacional se sobressai em determinada aplicação em relação ao NoSQL, principalmente com relação a inserção de grande massa de dados.

Palavras Chave: Banco de dados; Modelo Relacional; Armazenamento de dados; NoSQL.

ABSTRACT

Due to the amount of data and the need to manipulate them, the growing technological advance makes available several database models in the market, each with its own specifics and applicabilities. The purpose of this work is to compare the relational database system traditionally used for data storage and the NoSQL database system, which emerged as a workaround to manage larger amounts of data, based on concepts, application and data storage. At the end, it is demonstrated through a study the performance analysis of each bank, and it is also proved through the results, that for the application addressed in this work the relational model excels in certain application in relation to NoSQL, mainly in relation to insert a large mass of data.

Keywords: Database; Relational Model; Data storage; NoSQL.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAS DE ABREVIAÇÕES E SIGLAS

SQL	*Structured Query Language.*
NoSQL	*Not Only SQL* – (Não Somente SQL).
BD	Banco de Dados.
JSON	*JavaScript Object Notation.*
XML	*Extensible Markup Language.*
YAML	*Ain't Another Markup Language.*
HTTP	*HyperText Transfer Protocol.*
IBM	*International Business Machines.*
SEQUEL	*Structured English Query Language.*
ISO	*International Standards Organization.*
ANSI	*Amareican National Standards Institute.*
DDL	*Data Definition Language.*
DML	*Data Manipulation Language.*
SGBD	Sistema de Gerenciamento de Banco de Dados.
SGBDR	Sistema de Gerenciamento de Banco de Dados Relacional.
IDE	*Integrated Development Environment*
HTML	*Hyper Text Markup Language*
ASP	*Active Server Pages*
LINQ	*Language Integrated Queries*
API	*Application Programming Interface*

SUMÁRIO

1 INTRODUÇÃO

Os bancos de dados estão cada vez mais presentes no cotidiano das pessoas, fato este que colabora para o crescente volume de informações, tanto nas organizações quanto na Web. Todavia, há uma grande gama de modelos de banco de dados disponíveis no mercado, tendo cada um deles suas próprias especificações e aplicabilidade, destacando o modelo relacional, por ser comumente utilizado em aplicações, e também o modelo não relacional (NoSQL), devido a sua crescente utilização por suas características de desempenho e escalabilidade em determinadas aplicações de *software*.

O modelo relacional é, comumente, o tipo de banco de dados mais utilizado em aplicações de softwares. Por outro lado, há um crescimento acentuado do volume de dados, principalmente em aplicações web, e certas limitações tem motivado o surgimento de modelos alternativos ao banco de dados relacional. Com isto o modelo não relacional (NoSQL) vem ganhando cada vez mais espaço no mercado.

Os Bancos de Dados NoSQL (*Not Only SQL*) vieram para resolver essa demanda de armazenamento de dados, apresentando uma abordagem diferente de persistência de dados, baseada no paradigma BASE (*Basically Available, Soft-state or Scalable, Eventually Consistency*), desempenho, disponibilidade, escalabilidade e consistência dos dados. A diversidade do Banco de Dados Não Relacionai (NoSQL) é grande, cada um possuindo particularidades e conceitos diferentes proporcionando ao desenvolvedor uma gama enorme, podendo atender às necessidades distintas, de acordo com quatro categorias, orientado a coluna, orientado a documentos, orientado a grafos e orientado a armazéns chave-valor.

De acordo com (FOWLER, 2011) e (LEITE, 2010), banco de dados NoSQL foram desenvolvidos para atender situações especificas. Assim, começa a surgir alguns questionamentos de quando utilizar banco de dados relacionais e quando utilizar banco de dados NoSQL? Ou quando pode-se utilizar os dois?

Com isto, o objetivo deste trabalho é analisar o desempenho do banco de dados relacional (MySQL) e não relacional (RavenDB), fornecendo dados que auxilie na tomada de decisão sobre o uso dos mesmos. E sendo assim elaborado um estudo das principais literaturas acerca destes modelos em questão, definindo seus prontos fortes e fracos, e em seguida apresentando os resultados do estudo comparativo.

Esse tema é relevante, e justifica-se pelos fatores de riscos envolvidos durante a escolha de um modelo de banco de dados a ser utilizado em uma aplicação de software, visto

que cada um deles possuem características distintas, encarecendo ao profissional conhecer a ambos.

Na sequência o trabalho foi particionado como, Capitulo 1 – Referencial teórico, Capitulo 2 – Desenvolvimento, subdivido com os subitens Configuração e Aplicação, e por último o Capitulo 3 – Conclusão, neste é descrito o resultado final sobre o desempenho dos bancos de dados no cenário utilizado.

2 REFERENCIAL TEÓRICO

Nesta seção é feita uma abordagem geral sobre os conceitos básicos necessários para a compreensão dos principais temas abordados nesta pesquisa. De uma forma geral serão trabalhadas as tipologias de dados abordadas, os conceitos fundamentais em torno do tema Banco de Dados - aprofundando-se no que tange as tipologias de Banco de dados Relacionais e Não Relacionais englobadas nos objetivos desta pesquisa – os conceitos em torno dos testes necessários para avaliação de desempenho dos dois Bancos de Dados em estudo e testes estatísticos utilizados na análise dos resultados.

2.1 Big Data

Big Data faz referência a ativos de informação de alto volume, alta velocidade ou alta variedade que exigem formas de processamento de informações econômicas e inovadoras, que permitem uma melhor percepção e tomada de decisões (GARTNER, 2017).

Um dos fatores que culminaram para o crescimento do volume de dados é a difusão dos dispositivos de captação de dados, com armazenamento na ordem de *Terabytes*, e aumento de velocidade de transmissão nas redes. Outro fator importante é a facilidade de geração e aquisição de dados gerados digitalmente, como máquinas fotográficas digitais, smartphones, GPS, etc. (CHAUDHURI, 2012).

Basicamente, podemos resumir as características do contexto Big Data em quatro propriedades (SINGH, 2012):

- Dados na ordem centenas de *Terabytes* ou mais (podendo chegar a ordem de *Pentabytes*): envolve, entre outros aspectos, o requisito de alto poder computacional de armazenamento e processamento dos dados;
- Poder de crescimento elástico: está relacionada ao fato de que a quantidade de dados pode variar de alguns *Megabytes* a vários *Terabytes* (e vice-versa) em um espaço de tempo relativamente curto, fazendo com que a estrutura 10 de software/hardware se adapta sob demanda, e seja alocada/desalocada dinamicamente;
- Distribuição do processamento dos dados: os dados devem ser distribuí- dos de forma transparente em vários nós espalhados de forma a manter a integridade dos dados, o que demanda armazenamento, processamento e controle de falhas distribuído;
- Tipos de dados variados, complexos e/ou semiestruturados: relacionada a adoção de modelos mais apropriados, flexíveis e eficientes para o armazenamento de tipos de dados variados e semiestruturados. Vale ressaltar que o modelo relacional não é o

mais adequado, pois não possui flexibilidade para o armazenamento de dados e evolução no modelo para tais tipos de dados.

Apesar desta enorme quantidade de dados, o BigData em si não garante a qualidade da informação, pois a análise continua, em grande parte, sendo subjetiva. Isso se deve ao fato que os dados não são autoexplicativos, e o processo de limpeza, amostragem, e relacionamento dos dados continua sendo crítico e passível a erros, aproximações e incertezas (FISHER et al., 2012).

2.2 Banco de Dados Relacional

Um sistema de banco de dados pode ser considerado apenas como um sistema computadorizado de manutenção de registros. Pode ser tratado como um recipiente ou repositório para armazenar uma coleção de dados computadorizados (DATE, 2004).

O modelo relacional (MR) é um modelo de dados representativo (ou de implementação) que foi proposto por Ted Codd matemático e pesquisador da IBM (*International Business Machines*), em 1970. O modelo baseia-se em conceitos matemáticos – teoria dos conjuntos e lógica de predicado (TAKAI et al., 2005). Os primeiros sistemas comerciais baseados no modelo relacional foram disponibilizados em 1980 e desde então são implementados em muitos sistemas, como Access, Oracle, MySql, entre outros (ELMASRI et al., 2005).

A estrutura fundamental do banco de dados relacional é a relação, também conhecida como tabela, onde os dados são organizados por um ou diversos atributos que fazem uso da metainformação para traduzir o tipo de dado que será armazenado, esse conjunto de dados por sua vez é organizado em registros (linhas) (PINTAR, 2016).

O modelo relacional caracteriza-se para o banco de dados como uma coleção de relações. Como uma tabela é uma coleção dessas relações, existe uma similaridade entre o conceito de tabela e o conceito de relação matemática. (ELMASRI, 2011; SILBERSCHATZ, 2006).

Outra grande característica importante para o modelo é a utilização de restrições de integridade que fazem o uso de chaves primárias e chaves estrangeiras a fim de garantir a integridade dos dados. A chave primária que permite a identificação única de um registro dentro de uma relação e aumento no desempenho de consultas, e a chave estrangeira, que por sua vez permite com que o usuário realize relações entre tabelas distintas (PINTAR, 2016).

Os sistemas gerenciadores de bancos de dados relacionais trouxeram a liberdade para que o usuário não se preocupe mais com a garantia e integridade de seus dados, controle e coerência, recuperação de falhas e segurança dos dados, fazendo com que todo o trabalho seja realizado e gerenciado pelos SGBDs.

2.2.1 Padrão SQL

A linguagem SQL (*Structured Query Language)* ou Linguagem de Consulta Estruturada foi criada e implementada na IBM *Research* como protótipo de um sistema de banco de dados chamado System R (ELMASRI, 2011).

Tendo como base as linguagens de Álgebra e Cálculo Relacional, e inicialmente declarada como SEQUEL, SQL atualmente é a linguagem padrão para Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR), sendo mais fácil de ser utilizada do que suas linguagens maternas – consideradas bastante técnicas para o usuário (ELMASRI; NAVATHE, 2011).

Através de um esforço conjunto entre a ISO e a ANSI, foi criada uma versão padrão da linguagem SQL. Em 1986 foi lançada a primeira versão denominada SQL-86 (ou SQL1), e desde então ela vem sendo atualizada -SQL-92 (ou SQL2), SQL:1999 (ou SQL3), SQL:2003, SQL:2006 e, ainda, uma outra atualização ocorreu em 2008 (ELMASRI; NAVATHE, 2011).

A SQL possui instruções para definição dos dados, consultas e atualizações, que são os seguintes: DDL (*Data Definition Language)* e DML (*Data Manipulation Language*). Embora seja conhecida como uma "linguagem de consulta", a SQL oferece também recursos para definir a estrutura dos dados, tais como: atualizar – incluir, excluir e alterar – dados, especificar restrições de integridade e outros recursos mais (SILBERSCHATZ, 1999).

2.2.2 Limitações do Modelo Relacional

Os principais problemas encontrados com a utilização do modelo relacional estão comumente relacionados ao grande crescimento do número de aplicações, soluções, recursos e tudo o que se refere a sistemas computacionais nas últimas décadas, desta forma a dificuldade de conciliar o tipo de modelo com a demanda de escalabilidade se torna cada vez mais frequente (BRITO, 2010).

Segundo Ferreira et al. (2000) a escalabilidade do modelo de dados relacional possui um limite pois, à medida que a base de dados é distribuída, a complexidade das

ligações com tabelas, que não estejam na mesma máquina, aumenta. Do mesmo modo, à medida que o número de instâncias da base de dados aumenta, complica-se a gestão da garantia das propriedades ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*) em transações distribuídas.

Com isto, as bases relacionais começaram a demonstrar os seguintes problemas quando lidam com enormes volumes de informação:

- Um aumento na complexidade geral da base de dados e na sua gestão, quando esta é distribuída por várias instancias.
- As propriedades ACID de uma base de dados relacional obrigam que tenham uma distribuição e não isolamento de dados.
- Escalabilidade: as bases de dados escalam, porém não escalam facilmente. É possível escalar vertical e/ou horizontalmente, mas a complexidade exigida na gestão dos múltiplos nós da base de dados, ou o custo do *hardware* para suportar a distribuição, são entraves.

2.2.3 MySQL

O MySQL é um Sistema Gerenciador de Banco de Dados (SGBD) que possui uma licença de distribuição livre e de código aberto, distribuído pela empresa Oracle, e é apoiado por uma comunidade grande e ativa de desenvolvedores (MySQL,2011).

O MySQL e outros produtos foram licenciados de acordo com o GNU (GENERAL PUBLIC), projeto iniciado em 1984, cujo objetivo principal era garantir aos seus usuários a liberdade de compartilhar e alterar seus aplicativos e torna-los livres. A alternativa ao software livre (*Open Source*) tem sido amplamente aderida por usuários comuns ou empresas nos mais diversos segmentos, pelo simples fato de que para sua utilização não há necessidade de efetuar gastos com pagamento de licenças.

O MySQL é um servidor *multi-threaded*, que atende a vários usuários ao mesmo tempo, para isso são criados múltiplos processos de execução em um único processo, e suas tarefas são executadas por controle do usuário. Já quanto a sua distribuição, o MySQL reúne diversos programas e bibliotecas de cliente (*client*), ferramentas administrativas e API´s (*Application Programming Interface*) e muito mais programas desenvolvidos e disponibilizados por colaboradores e parceiros (LIMA, 2003).

A arquitetura do MySQL é pouco diferente da dos demais servidores de banco de dados e é útil para uma grande variedade de objetivos. MySQL é flexível para trabalhar bem

em ambientes exigentes, como aplicações web. Ao mesmo tempo, o MySQL pode potencializar aplicações embutidas, depósitos de dados, indexação de conteúdo e software de distribuições, sistemas redundantes altamente disponíveis, processamento de transação online e muito mais. (SCHWARTZ et al, 2009).

2.3 Banco de Dados Não Relacional (NoSQL)

Conhecido inicialmente como banco de dados relacional de código aberto e não possuir interface, esse termo foi utilizado inicialmente em 1998 pelo seu autor Carlo Strozzi, que alegava que o movimento NoSQL "é completamente diferente do modelo relacional e, portanto, deveria ser mais apropriadamente chamado de 'NoREL' ou algo que produzisse o mesmo efeito" (STROZZI, 2000).

Esse modelo de banco de dados faz o uso de vários modelos de dados que incluem documentos, gráficos, chave-valor e colunas. Em 2006, o conceito NoSQL veio à tona novamente quando o Google publicou um artigo chamando *BigTable: A Distributed Storage System for Structured Data*, onde descreve o *BigTable*, projetado pela Google para atender seus projetos e demandas internas e promover maior escalabilidade e disponibilidade. Esse artigo trouxe grandes contribuições, uma delas foi mostrar a flexibilidade na construção da própria solução em armazenamento e atender os requisitos apresentados (BRITO, 2010).

Um diferencial entre banco de dados NoSQL e o SQL, é que o primeiro ganha no quesito performance e na flexibilidade para atender as características particulares de cada organização. Esse fator foi primordial para algumas organizações em meados dos anos 90, quando os modelos relacionais já não atendiam as suas necessidades. Outro diferencial do banco de dados NoSQL é a escalabilidade de seu sistema, e isso era um problema no modelo relacional, que perdia desempenho devido ao grande aumento no número de usuários, e as alternativas para contornar esses problemas na maioria das vezes não era viável (BRITO, 2010).

As primeiras implementações de um sistema NoSQL que disponibilizava maior escalabilidade foi apresentado em 2004 pelo Google, quando o mesmo lançou o *BigTable*, um banco de dados proprietário de alta performance. Já a escalabilidade se baseia em quatro fases principais: a primeira é projeto de agregados onde busca identificar as classes de objetos agregados em uma aplicação, nessa fase as atividades são guiadas por casos de uso e requisitos funcionais. Na segunda fase vem o particionamento de agregados, os agregados são divididos em dados menores e essa atividade é conduzida pelos requisitos de desempenho e casos de uso. A terceira fase consiste no projeto de banco de dados NoSQL de alto nível, onde

mapeia os agregados e entregue para o modelo de dados intermediário de acordo com a identificação das partições. Na última fase, a implementação recebe a representação intermediária de dados e converte em elementos de modelagem específicos do BD de destino (LIMA, 2015).

O modelo NoSQL é divido em quatro categorias básicas; os sistemas que são baseados em armazenamento chave-valor, que tem como exemplo o banco de dados Amazon Dynamo; os que são orientados a documentos, CouchDB e o MongoDB; os que são orientados a coluna, o Cassandra e o HyperTable, e por fim os que são orientados a grafos que tem como exemplos os Neo4j e o ArangoDB. Outra característica do modelo NoSQL, que são reconhecidos pela facilidade de desenvolvimento, desempenho escalável, alta disponibilidade e resiliência. (BRITO, 2010).

A figura 1 mostra que entre os dez bancos de dados mais utilizados no mundo, sete são relacionais. Assim, os bancos de dados relacionais ainda são a preferência da maioria dos programadores, mostra ainda que o banco de dados NoSQL que vem se destacando e ganhando adeptos é o Orientado a Documentos, mais especificamente o MongoDB. Esse modelo de banco de dados é o mais próximo do modelo relacional, enquanto um banco de dados relacional trabalha com linhas e colunas, o banco de dados orientado a documentos guarda em documentos (VAISH, 2013).

Figura 1 – Ranking dos Bancos de Dados ("DB-Engines Ranking", 2017).

327 systems in ranking, May 2017

Rank			DBMS	Database Model	Score		
May 2017	Apr 2017	May 2016			May 2017	Apr 2017	May 2016
1.	1.	1.	Oracle	Relational DBMS	1354.31	-47.68	-107.71
2.	2.	2.	MySQL	Relational DBMS	1340.03	-24.59	-31.80
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1213.80	+9.03	+70.98
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	365.91	+4.14	+58.30
5.	5.	↓ 4.	MongoDB	Document store	331.58	+6.16	+11.36
6.	6.	6.	DB2	Relational DBMS	188.84	+2.18	+2.88
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	129.87	+1.69	-1.70
8.	8.	↓ 7.	Cassandra	Wide column store	123.11	-3.07	-11.39
9.	9.	9.	Redis	Key-value store	117.45	+3.09	+9.21
10.	10.	10.	SQLite	Relational DBMS	116.07	+2.27	+8.81

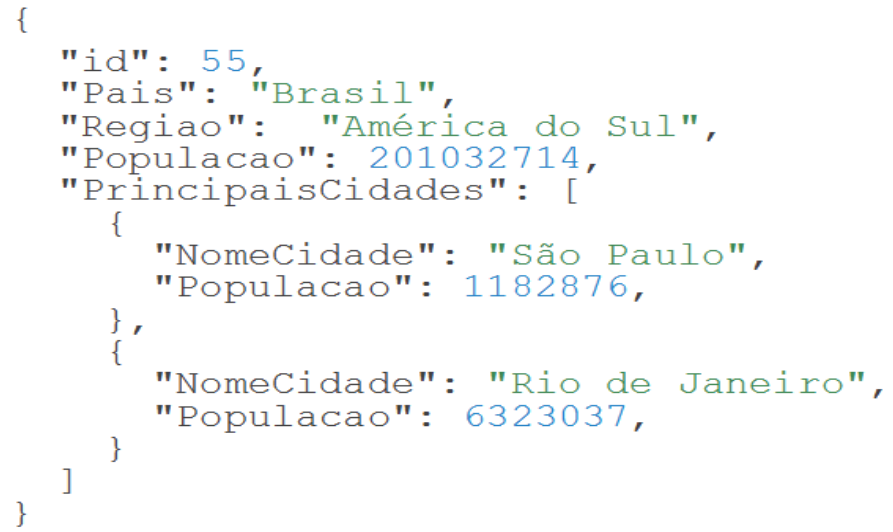
Fonte: https://db-engines.com/en/ranking.

Os bancos de dados orientados a documentos formam uma categoria adequada para aplicações *Web*, pois são projetados para armazenar os dados em forma de documentos, geralmente em formatos JSON, XML ou YAML e não possuem esquemas, ou seja, os documentos armazenados não precisam necessariamente de uma estrutura pré-definida,

normalmente usam o protocolo HTTP com RESTful ou o protocolo Apache Thrift para que as linguagens de programação consigam acessar os dados (IANNI, 2016).

A figura 2 mostra o exemplo de um documento com informações de um local de trabalho onde temos o código, o nome e seu endereço, nesse exemplo mostra um documento no formato padrão JSON. Um documento seria autossuficiente, ou seja, ele contém todas as informações necessárias para existir, sem precisar se relacionar com outro documento. Uma vantagem do banco de dados orientado a documentos é que quando se faz alguma alteração em um dos registros, os demais registros não sofrem nenhum tipo de alteração, permitindo flexibilidade na estrutura do banco, mesmo após a implementação (VAISH, 2013).

Figura 2 - Exemplo de BD Orientado a Documento no formato JSON.



Fonte: https://www.researchgate.net/figure/315779664_fig1_Figura-1-Exemplo-de-documento-no-formato-JSON

Num banco de dados orientado a documentos não existem tabelas e sim uma coleção de documentos e em cada um deles o programador armazena os dados na forma que mais lhe convém, mas ao mesmo tempo precisa se atentar para que ele não se perca em sua base de dados. O que faz esse tipo de banco se destacar em relação ao modelo relacional é que seu conteúdo não possui esquema, ou seja, não possui um diagrama de como o banco de dados é construído, ou é vagamente definido e isso é muito útil em aplicações web, nas quais é necessário armazenar os mais diversos tipos de conteúdo que evoluem ao longo do tempo (VAISH, 2013).

A Figura 3 ilustra a estrutura de um documento no MongoBD, onde o *me* é um atributo e contém a informação básica do registro, o *type* especifica o tipo da entidade e os outros atributos como o *personal*, *financial* e o *criminal* podem ser adicionados em uma outra

ocasião e não precisam ser necessariamente preenchidos quando se está programando, a vantagem de quem faz uso desse tipo de banco é que todos os dados associados à entidade estão em um único registro e isso otimiza as consultas (VAISH, 2013).

Figura 3 - Estrutura de um documento no formato MongoDB.

```
{
   "me": {
     "id"           : "document-uuid",
     "version"      : "1.0.0.0",
     "create_time"  : "2011-11-11T11:11:11Z",
     "last_update"  : "2012-12-12T12:12:12Z"
   },
   "type": "UserProfile",
   "personal": {
     "firstName"  : "Alice",
     "lastName"   : "Matthews",
     "date_of_birth": "1901-01-01T01:01:01Z"
   },
   "financial": {
     "bank"           : { ... },
     "trading"        : { ... },
     "credit-history" : { ... }
   },
   "criminal": {
   }
}
```

Fonte: https://www.mongodb.com/blog/post/creating-single-view-part-3-data-design-and-loading-strategies

2.3.1 Limitações do Modelo Não Relacional

Por mais que as vantagens na utilização deste modelo sejam evidentes, ainda sim existem algumas limitações. Para exemplos como Blog. Nesse exemplo, seria extremamente complicado contabilizar o número de posts existente no site ou o número de replies total, a todos os posts, porque essa informação estaria dispersa, enquanto no modelo relacional, essa informação ser-nos-ia dada quase instantaneamente.

Ou seja, em situações onde o modo como os dados estão estruturados é importante, como é o caso de contas bancárias, uma base de dados relacional é muito mais adequada. Abaixo segue alguns itens que mostram certas limitações.

- Suporte Técnico: Devido ao facto de grande parte dos sistemas NoSQL serem Open Source, o suporte é geralmente oferecido por empresas muito mais pequenas, sem o peso de grandes nomes existentes no mercado das bases de dados relacionais, como é o caso da Oracle, Microsoft ou IBM.

- Dificuldade de instalação e esforço para manutenção.
- Ausência de profissionais com que no modelo Relacional, onde existem milhões de especialistas. Grande parte dos utilizadores de NoSQL ainda estão em processo de aprendizagem.

2.3.2 RavenDB

O RavenDB é um banco de dados orientado a documentos de código aberto para .NET, feito para a persistência e recuperação de documentos. Tal como acontece com todos os bancos de dados orientados a documentos, os documentos são identificados por uma chave única e também podem ser pesquisados em seus atributos.

O RavenDB foi construído para ser um modelo não relacional de primeira classe na plataforma .NET oferecendo aos desenvolvedores a capacidade de estender e incorporar facilmente o banco de dados em suas aplicações (RITCHIE, 2013).

Alguns dos principais recursos que tornam o RavendDB atraente para os desenvolvedores .NET são:

- O RavenDB vem com uma API de cliente totalmente funcional .NET, que implementa a unidade de trabalho, o rastreamento de alterações, as otimizações de leitura, gravações e muito mais. Possui também uma API baseada em REST, para que você possa acessá-lo via *JavaScript.*
- Permite aos desenvolvedores definir índices usando LINQ (*Language Integrated Queries*). Suporta mapa / redução operações em cima de seus documentos usando o LINQ.
- Suporta *System.Transactions* e pode participar de transações distribuídas.
- O servidor pode ser facilmente expandido, adicionando um assembly .NET personalizado.

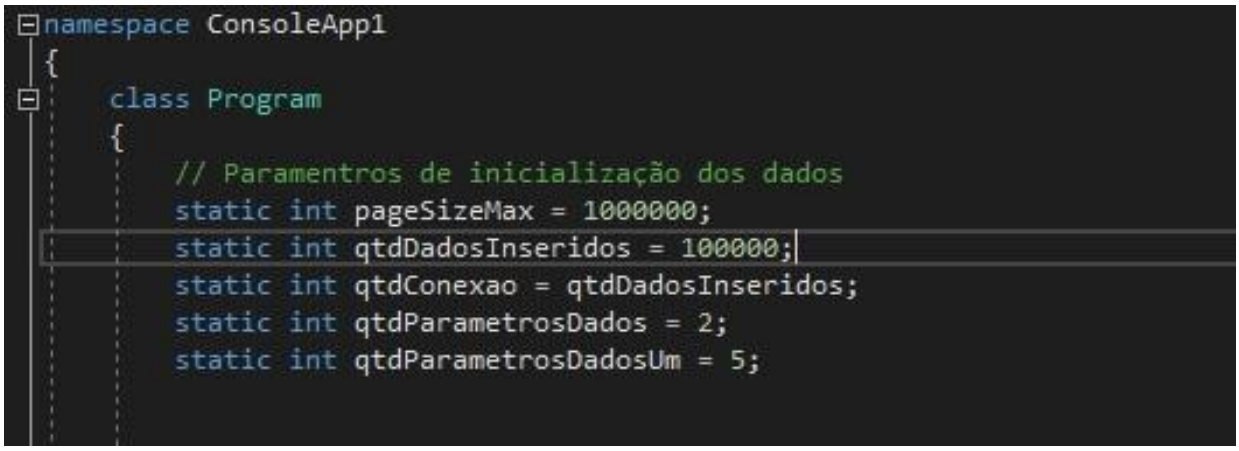
RavenDB aproveita a infraestrutura de armazenamento existente chamada Esent que é conhecida para dimensionar tamanhos surpreendentes. Esent é o mecanismo de armazenamento utilizado por Microsoft Exchange e Active Directory. O mecanismo de armazenamento fornece o armazenamento de dados transacional para os documentos. RavenDB também utiliza outra tecnologia comprovada chamada Lucene.NET para sua indexação de alta velocidade. Lucene.NET é um projeto Apache de código aberto usado para alimentar aplicativos como AutoDesk, StackOverflow, Orchard, Umbraco e muito mais (RITCHIE, 2013).

3 DESENVOLVIMENTO

A princípio definiu-se que seriam utilizados para a pesquisa o SQL Server da Microsoft como modelo relacional e o MongoDB como modelo não relacional, no entanto, encontrou-se dificuldades em trabalhar com esses modelos de bancos de dados por não ter conhecimento suficiente para o desenvolvimento da pesquisa, principalmente o MongoDB, optou-se então pela mudança de ambos, ficando definido que seriam utilizados o MySQL como modelo relacional por já se ter um conhecimento maior e o RavenDB como o modelo não relacional, também por ter-se encontrado um professor que esclarecesse várias dúvidas quanto a este modelo de banco de dados sanando todas as dúvidas existentes. Um dos motivos para utilizar esses dois modelos de bancos de dados, foi por serem *open source* e estarem disponíveis para Sistemas Operacionais Windows e Linux e atenderem as necessidades para o desenvolvimento da pesquisa.

Durante a codificação do modelo NoSQL encontrou-se algumas dificuldades, principalmente na parte do código em que tratava da inserção do arquivo, pois tem-se um arquivo com mais ou menos 370.000 mil objetos e cada objeto com 5 parâmetros, e o RavenDB comporta apenas a inserção de 30 objetos e a execução era abortada. Foi necessário um estudo mais aprofundado sobre a linguagem de programação para sanar o problema. Para facilitar o método de inserção, foi deixado de lado o arquivo que continham os 370.000 mil objetos e foi desenvolvida uma função onde ficaria a cargo dos testadores a quantidade de objetos a serem inseridos nos respectivos bancos. A Figura 4 mostra o trecho do código em que se define a quantidade de arquivos a serem inseridos.

Figura 4 - Função para alterar a quantidade de objetos a serem inseridos.



Fonte – Capturado da ferramenta pelos autores da pesquisa.

A finalidade do desenvolvimento dessa aplicação, é ter acesso aos dois tipos de bancos de dados, SQL e NoSQL, realizando inserções e consultas de registros com o objetivo final de observar o desempenho de ambos os bancos e assim definir qual modelo de banco de dados melhor se aplica a essa situação. O teste de desempenho consiste em testar um sistema visando os requisitos de desempenho. Existem diversos tipos de testes de desempenho, mas os mais comuns são os testes de stress e de carga. Costa (2012) os define como:

- Teste de stress: tem como objetivo determinar o comportamento de uma aplicação quando é submetido além de seus limites normais de carga. Busca determinar os pontos de defeito do sistema;
- Teste de carga: tem como objetivo validar o comportamento de uma aplicação sob condições normais de carga. Verifica se a aplicação cumpre os requisitos de desempenho.

3.1 Configuração

No ambiente de teste, a máquina utilizada foi um Notebook Samsung, com Processador Intel(R) Core(TM) i3-6006 CPU @2.00GHz 1.99GHz, Memória (RAM) 4,00 GB, Sistema Operacional Windows 10 de 64 bits e HD 1 TB. Os testes foram efetuados em banco local sem nenhum tipo de componente *on-line*, e não foi instalado nenhum programa, a não ser o próprio banco de dados a ser executado, evitando assim excesso de processos consumindo memória e recursos que poderiam interferir nos resultados dos testes. Os testes realizados foram operações básicas de inserção e consulta.

A Tabela 1 mostra quais foram os bancos de dados utilizados, e também suas respectivas ferramentas administrativas. Em nenhum dos testes, o processamento e memória não interferiram no desempenho dos processos.

Tabela 1 - Função para alterar a quantidade de objetos a serem inseridos.

Banco de Dados	**Ferramenta Administrativa**
MySQL 5.7	MySQL Workbench 6.3 CE
RavenDB 3.5.4	Management Studio

3.2 Aplicação

A aplicação foi desenvolvida na linguagem C# sem interface gráfica. Para o desenvolvimento do código foi utilizado o software Visual Studio 2017 no modelo NoSQL e o Visual Studio Code para o modelo SQL. A decisão em desenvolver as aplicações utilizando o Visual Studio 2017 foi tomada por ser uma ferramenta *open source*, eficiente e atual que suporta todas as necessidades do projeto.

O Visual Studio 2017 é uma IDE da Microsoft que pode ser usada para desenvolvimento de aplicações. Essa ferramenta oferece suporte a diversas linguagens de programação como: C, C++, C#, Visual Basic .NET, HTML, ASP, entre outras (MICROSOFT, 2017).

Segundo (GUIMARÃES, 2007) não existe um meio universal com o qual avalia-se o desempenho de vários sistemas computacionais, existe a necessidade de definir uma métrica em especifico para cada caso, pois assim como os softwares, os bancos de dados são bastante complexos, pois existe uma grande variação de implementação.

Desenvolvedores de aplicações podem encontrar algumas dificuldades quando compararem diversos sistemas SGBDs com a intenção de encontrarem qual deles é o mais adequado. As bases de dados NoSQL são um excelente complemento para o modelo relacional. A escolha por um modelo não relacional deve estar diretamente relacionada à compatibilidade da sua modelagem com uma ou outra categoria mais do que o mero ganho de performance (LOBO, 2012).

Os arquivos inseridos nos respectivos bancos foram dois arquivos de diferentes tamanhos, um arquivo com dois parâmetros com tamanho de 51 bytes e um com 5 parâmetros com tamanho de 128 bytes. Os arquivos estão no formato JSON. A Figura 5 e 6 mostram os modelos dos respectivos objetos que foram inseridos no banco de dados SQL e NoSQL, um com 2 parâmetros e outro com 5 parâmetros.

Figura 5 - Modelo de um objeto com 2 parâmetros.

```
{
    "id": 1,
    "descricao": "descricao: 0"
}
```

Figura 6 - Modelo de um objeto com 5 parâmetros.

```
{
  "id": 1,
  "cep": "75080320",
  "logradouro": "Rua 200 Setor Jaiara",
  "complemento": "Casa"
  "cidade": "Anapolis"
}
```

A Figura 7 apresenta uma tabela com o tempo gasto durante a inserção e consulta nos respectivos bancos.

Figura 7 - Resultado de inserção e busca na base de dados dos respectivos bancos.

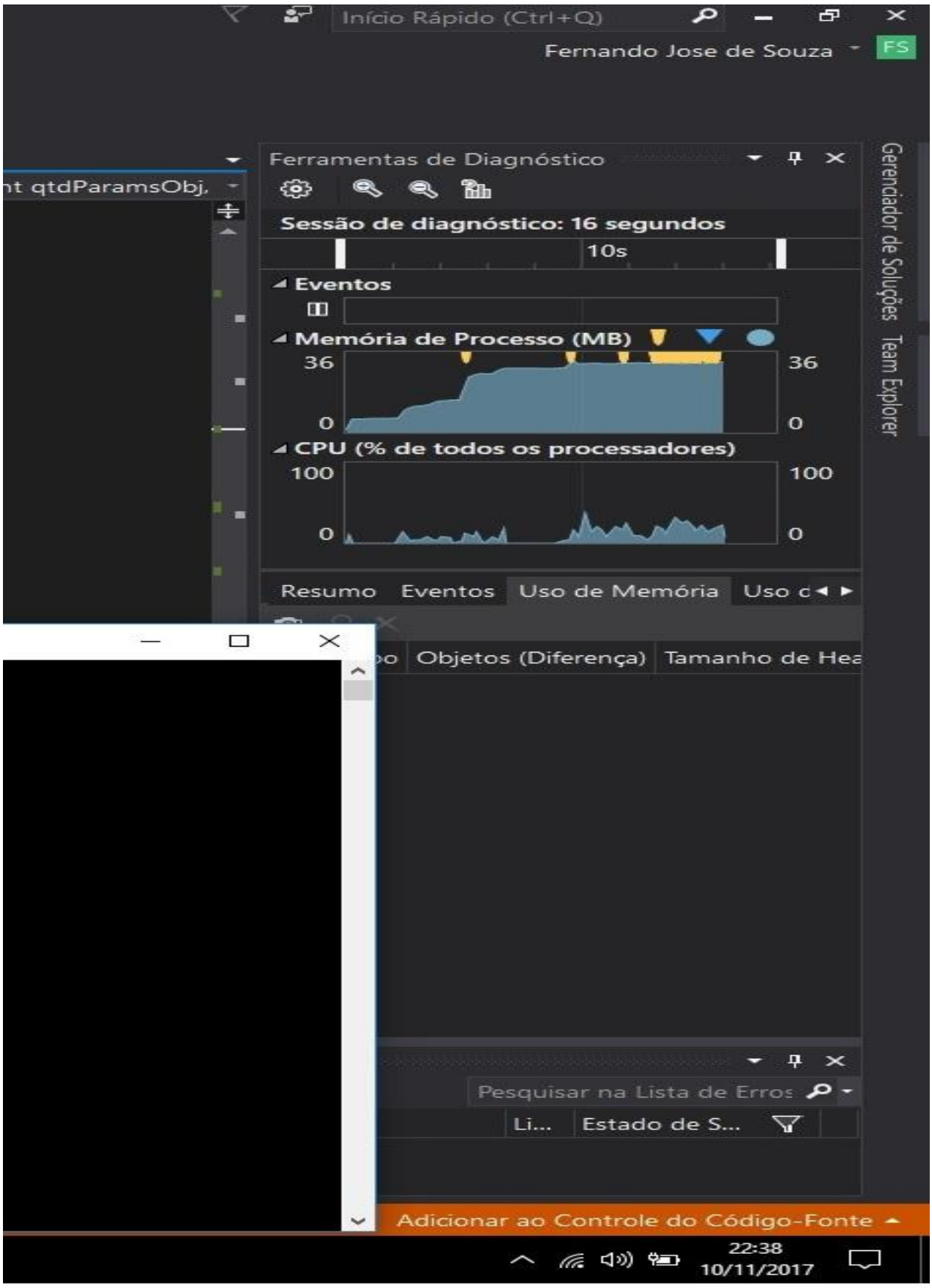
Teste SQL			
Qtd de arquivos inseridos	Nº de parâmetros de cada arquivo	Tempo gasto (inserção)	Tempo gasto (consulta)
100	2	00:00:00.4	00:00:00.2
	5	00:00:00.5	00:00:00.2
500	2	00:00:02.7	00:00:00.2
	5	00:00:03.1	00:00:00.2
1.000	2	00:00:07.4	00:00:00.2
	5	00:00:07.5	00:00:00.3
5.000	2	00:01:13.3	00:00:00.5
	5	00:01:44.9	00:00:00.2
10.000	2	00:03:53.9	00:00:00.4
	5	00:05:05.0	00:00:00.3
25.000	2	00:15:35.0	00:00:00.3
	5	00:22:34.2	00:00:00.3
50.000	2	01:00:10.6	00:00:00.8
	5	01:25:53.6	00:00:00.5
100.000	2	03:31:01.5	00:00:00.7
	5	05:55:04.3	00:00:05.3

Teste NoSQL			
Qtd de arquivos inseridos	Nº de parâmetros de cada arquivo	Tempo gasto (inserção)	Tempo gasto (consulta)
100	2	00:00:02.2	00:00:00.1
	5	00:00:00.6	00:00:00.02
500	2	00:00:08.9	00:00:00.3
	5	00:00:11.8	00:00:00.1
1.000	2	00:00:40.3	00:00:00.9
	5	00:00:40.3	00:00:00.4
5.000	2	00:05:33.7	00:00:01.9
	5	00:07:07.3	00:00:01.3
10.000	2	00:21:27.2	00:00:04.0
	5	00:28:16.9	00:00:03.0
25.000	2	01:52:56.1	00:00:09.6
	5	02:36:12.8	00:00:06.7
50.000	2	07:24:58.0	00:00:18.8
	5	09:52:55.8	00:00:17.0
100.000	2	-	-
	5	-	-

Fonte – Capturado da ferramenta pelos autores da pesquisa.

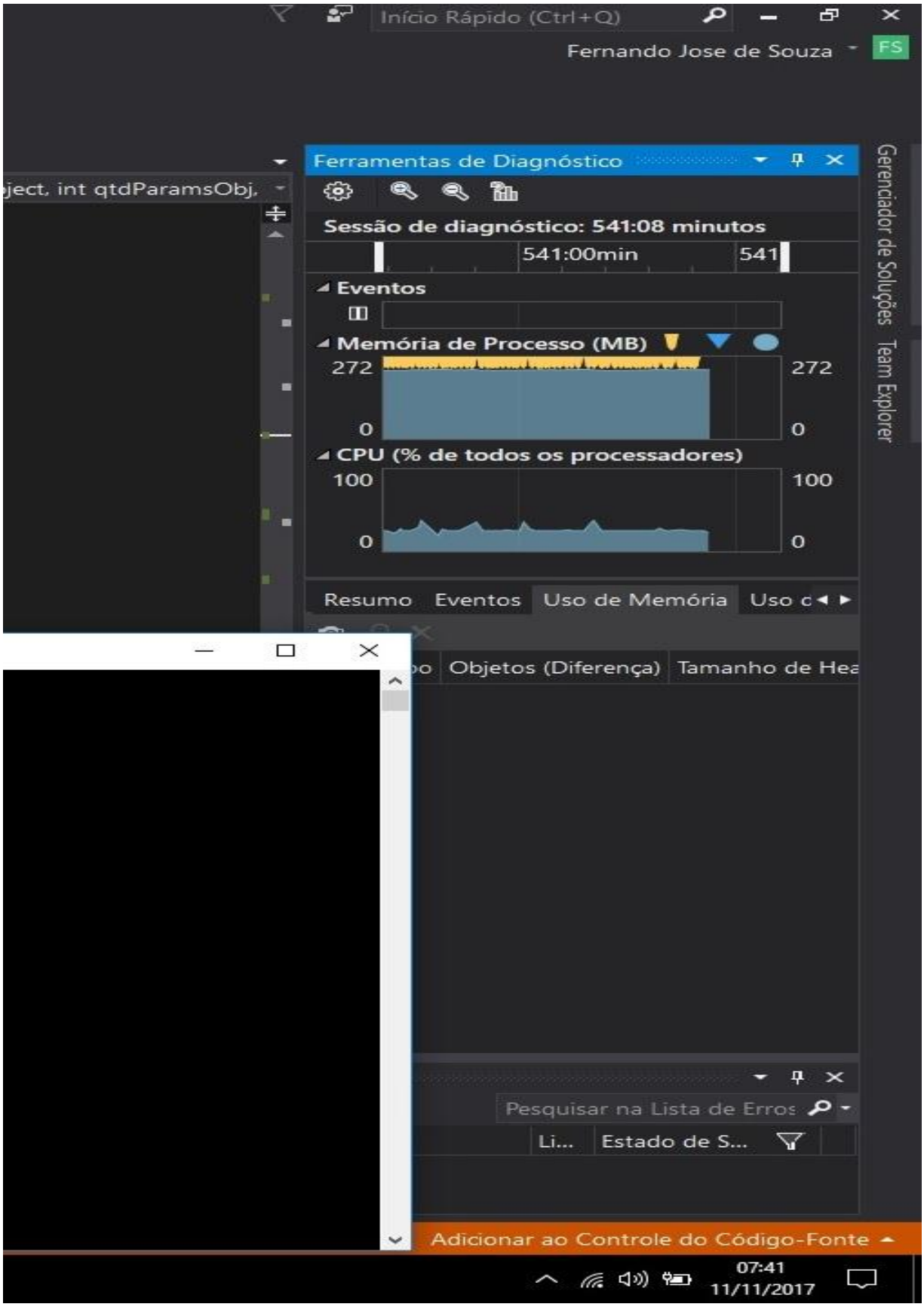
No modelo NoSQL, ao realizar o teste com os 100.000 mil registros (inserção e consulta), o processo foi abortado durante a execução, nas Figuras 8 e 9 mostram o tempo de execução que foi gasto e sem finalizar o processo por completo. A Figura 10 mostra a quantidade de objetos inseridos no banco de dados NoSQL durante o processo. A execução durou um total de 541 minutos, pouco mais de 9 horas, e a quantidade de objetos inseridos foram de 57.533 mil objetos com 2 parâmetros, sendo que restava a inserção do objeto com 5 parâmetros, de um total de 100.000 mil objetos pré-definidos.

Figura 8 - Início do processo de inserção com 100.000 registros.



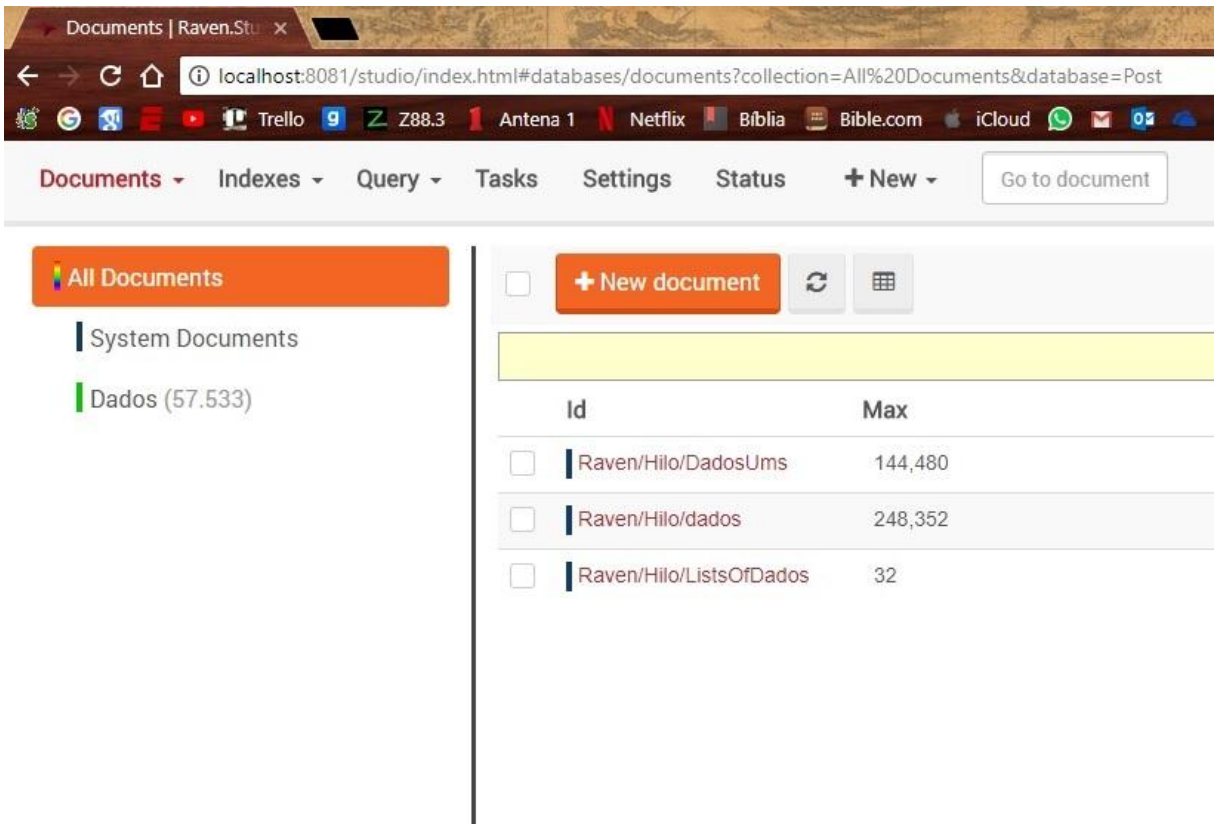
Fonte – Capturado da ferramenta pelos autores da pesquisa.

Figura 9 - Fim do processo de inserção depois de 541 minutos de execução



Fonte – Capturado da ferramenta pelos autores da pesquisa.

Figura 10 - Quantidade de objetos inseridos após 541 minutos de execução.



Fonte – Capturado da ferramenta pelos autores da pesquisa.

O NoSQL tem muitas vantagens para ser utilizado, entretanto, isso não justifica sua utilização em todas as situações. Em muitos sistemas, você pode-se usar o modelo relacional.

O NoSQL não veio para substituir o SQL, mas para oferecer mais uma alternativa de um banco de dados mais flexível no suporte de dados e cada caso dever ser estudado com cuidado. Algumas aplicações podem usar o modelo NoSQL para uma escrita e leitura temporária e atualizando o modelo SQL de tempo em tempo, sendo assim, pode-se aplicar ambas as soluções para diferentes casos de uso. Por isso, o mais comum em soluções escalares de sucesso é a utilização de uma arquitetura híbrida, aproveitando o melhor dos dois modelos (CAELUM, 2009).

4 CONSIDERÇÕES FINAIS

A utilização do modelo de banco de dados NoSQL vem se tornando mais comum, uma saída para aquelas organizações que gerenciam grande fluxo de dados, a facilidade em se trabalhar com esse modelo de banco de dados, faz com que um grande volume de dados possa ser armazenado sem muitos problemas. Mesmo com seu crescimento, o modelo relacional ainda lidera como banco de dados mais utilizado no mundo, como mostra a Figura 1.

O objetivo principal deste trabalho foi comparar o desempenho do MySQL e o RavenDB utilizando operações básicas de inserção e consulta. Com o resultado dos testes, fica claro que apesar do NoSQL ter o intuito de amenizar problemas de performance, o modelo relacional se sobressaiu em relação ao não-relacional, levando em conta o cenário e os bancos de dados escolhidos. Como os testes realizados foram executados em um microcomputador (*notebook*), os resultados não foram muitos satisfatórios, acredita-se que, se os mesmos testes forem realizados em um *Mainframe,* os resultados seriam diferentes e mais satisfatório, pois um *Mainframe* tem um poder de processamento maior do que de um *notebook.*

No teste de inserção, o RavenDB teve um péssimo desempenho em relação ao MySQL, o RavenDB utilizou cerca de 8 vezes mais de tempo do que o MySQL para fazer a inserção dos registros. A grande diferença de tempo pode estar relacionada com o RavenDB criar documentos a cada registro inserido. No teste de busca, o MySQL também teve um desempenho melhor que o RavenDB, o MySQL manteve um bom desempenho, com regularidade em todos os níveis de testes.

Não foi utilizado nenhum tipo de otimização para a realização dos testes, o que pode ter influenciado desempenho dos testes. Desta forma conclui-se que para o cenário adotado nesse trabalho (um *notebook*), o MySQL apresentou melhor estabilidade e desempenho nos testes de inserção e busca em relação ao RavenDB.

5 REFERÊNCIAS BIBLIOGRÁFICAS

BHATT, Chintan. **Internet of Things and Big Data Analytics Toward Next-Generation Intelligence.** Publisher: Springer Series of Big Data, 2017.

BRITO, R. W. **Banco de Dados NoSQL x SGBDs Relacionais: Análise Comparativa**. Disponível em: <http://www.infobrasil.inf.br/userfiles/27-05-S4-1-68840-Bancos de Dados NoSQL.pdf>. Acesso em: 10 out. 2016.

CAELUM. **Bancos de dados não relacionais e o movimento NoSQL.** Disponivel em: <http://blog.caelum.com.br/bancos-de-dados-nao-relacionais-e-o-movimento-nosql/> Acesso em: 15 out. 2017.

DB-Engines Ranking. Disponível em: <https://db-engines.com/en/ranking>. Acesso em: 6 maio. 2017.

DATE, C. J. Introdução a sistemas de banco de dados, 8ª ed., Campus - RJ Inativar, 2004.

DEVMEDIA. **Modelagem SQL x NoSQL**. Disponível em: <http://www.devmedia.com.br/modelagem-sql-x-nosql/29446>. Acesso em: 2 nov. 2016.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de banco de dados. 6ª ed. São Paulo: Editora Pearson, 2011.

FOWLER, M. **Polyglot Persistence**. Disponível em: <https://martinfowler.com/bliki/PolyglotPersistence.html>. Acesso em: 10 abr. 2017.

GARTNET. Gartner. **Data & Analytics.** Disponível em <https://www.gartner.com/>. Acesso em: 30 out. 2017.

GUIMARÃES, L. **Um Aplicativo para Teste de Performances de Bancos de Dados Através dos Benchmarks TPC-C e TPC-HL**. Disponível em: <http://www.lia.ufc.br/site>. Acesso em: 18 out. 2017.

IANNI, V. **Introdução aos bancos de dados NoSQL**. Disponível em: <http://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044>. Acesso em: 6 maio. 2017.

LEITE, G. **Análise Comparativa do Teorema de CAP Entre Banco de Dados NoSQL e Banco de Dados Relacionais**. Disponível em: <http://www.ffb.edu.br/sites/default/files/tcc-20102-gleidson-sobreira-leite.pdf>. Acesso em: 10 abr. 2017.

LOBO, H. **NoSQL ou SQL? Quando uso um, outro ou ambos**. Disponível em: <http://www.itexto.net/devkico/?p=1199>. Acesso em: 3 nov. 2017.

LIMA, C. **Um Estudo sobre Modelagem Lógica para Banco de Dados NoSQL**. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erbd/2015/002.pdf>. Acesso em: 3 nov. 2016.

MICROSOFT. **Tour pelos recursos do IDE do Visual Studio**. Disponível em: <https://docs.microsoft.com/pt-br/visualstudio/ide/visual-studio-ide>. Acesso em: 15 set. 2017.

MYSQL. **MySQL 5.5 Reference Manual.** MySQL. Disponível em: <https://dev.mysql.com/doc/refman/5.5/en/index.html.> Acesso em: 01 out. 2017.

RAMEZ, E. **Sistemas de Banco de Dados**. 6ª ed. 2011.

RITCHIE, Brian. **RavenDB High Performance.** Packt Publishing, 2013.

SCHWARTZ, Baron. et al. **"Alto Desempenho em MYSQL"**. Rio de janeiro: Alta Books, 2009.

SINGH, Sachchidanand; SINGH, Nirmala. **Big Data Analytics.** In: INTERNATIONAL CONFERENCE ON COMMUNICATION, 2012, Ottawa. Anais eletrônicos. Ottawa: Information & Computing Technology (ICCICT), Oct. 19-20, 2012. p.1-4. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6398180&tag=1> Acesso em: 15 out. 2017.

SEVERINO, Antônio Joaquim. Metodologia do trabalho científico. 23 ed. São Paulo: Cortez,2007.

SILBERSCHATZ, Abraham. **Sistema de banco de dados**. 5. ed. Rio de Janeiro: Elsevier, 2006.760 p.

STROZZI. **NoSQL**. Disponível em: <http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home Page>. Acesso em: 3 abr. 2017.

VAISH, G. **Getting Started with NoSQL**. Birmingham: 2013.