Análise comparativa entre um banco de dados relacional (PostgreSQL) e um banco de dados não somente SQL (MongoDB)

Trabalho de Conclusão de Curso

Engenharia da Computação

Irandré Sales de Menezes Junior
Orientador: Prof. Eliane Maria Loiola

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

IRANDRÉ SALES DE MENEZES JUNIOR

Análise comparativa entre um banco de dados relacional (PostgreSQL) e um banco de dados não somente SQL (MongoDB)

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, novembro de 2016.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 22 de 12 de 2016, às 10:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente **IRANDRE SALES DE MENEZES JUNIOR**, orientado pelo professor **Eliane Maria Loiola**, sob título **Análise comparativa entre bancos de dados relacionais e bancos de dados não somente SQL (NoSQL – not only SQL)**, a banca composta pelos professores:

Tarciana Dias da Silva
Eliane Maria Loiola

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☐ Aprovada ☒ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 7,5 (sete e meio)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 30 dias para entrega da versão final da monografia a contar da data deste documento.

Tarciana Dias da Silva
TARCIANA DIAS DA SILVA

Eliane Maria Loiola
ELIANE MARIA LOIOLA

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Dedico esta monografia aos meus pais e a Meg, que sempre me incentivaram em todos esses anos de estudos.

Agradecimentos

Agradeço primeiramente a Deus, o criador e mantenedor de todas as coisas, obrigado por me manter firme durante todos esses anos de estudos, obrigado pelo consolo nos momentos difíceis, obrigado por me dar paciência e discernimento.

Aos meus pais agradeço pelo esforço que tiveram para me educar em todos esses anos, sou hoje uma boa pessoa, pela boa educação que me deram, obrigado por acreditarem em mim, obrigado por me motivar nos momentos difíceis, obrigado pelos momentos alegres que me proporcionaram, e por todas as ajudas que pude ter vindas de vocês. Agradeço também ao meu irmão Igleiton por todos os momentos alegres que tivemos juntos, obrigado ser esse irmão que posso contar sempre.

Obrigado meu amigo Wandson por me fazer rir, e me ensinar a sair mais, obrigado pelos ensinamentos e por sempre estar disponível para me ajudar, muito sucesso.

Aos meus amigos Carlos Alberto e John Kennedy, obrigado por toda ajuda que tive de vocês, nos momentos de dúvidas, nas vezes que fizemos trabalhos juntos, vocês são ótimas pessoas e merecem todo o sucesso.

A professora Eliane, agradeço pelas orientações que me deu, para que este trabalho fosse feito de maneira coerente.

A Sinha e Meg, obrigado por todos os momentos de felicidades que me proporcionaram, vocês são as coisinhas mais fofas do mundo, amo vocês.

Resumo

A quantidade de informação que está disponível para a humanidade é enorme e à medida que o conhecimento humano se expande, maior é a quantidade dessa informação que precisa ser armazenada e analisada. O banco de dados relacional pode ser pesado para aplicações que precisam de velocidade, como aplicações *web* e *blogs* que possuem diversos tipos de atributos. Para resolver requisitos não funcionais como estes relacionados à agilidade é necessária uma base de dados igualmente ágil. O NoSQL foi projetado para armazenamentos de dados distribuídos, onde há necessidade de armazenar grandes quantidades de dados. Modelo de dados NoSQL não exigem um tipo fixo para seus elementos, não sendo necessário defini-lo previamente. O NoSQL não foi criado para extinguir a linguagem SQL, em outras palavras, podemos usar o SQL e banco de dados NoSQL para alcançar os melhores resultados. Este estudo mostra uma comparação entre os dois paradigmas de bancos de dados, apresentando uma análise de tempo de inserção, seleção atualização e remoção de dados, com esta comparação, é possível notar que o MongoDB (NoSQL) tem melhor desempenho em inserir, selecionar, atualizar e remover dados do que o PostgreSQL (relacional), também é possível observar que o PostgreSQL (relacional) tem uma plataforma mais completa que o MongoDB (NoSQL) com interfaces para o usuário e administrador. Por fim, o estudo apresenta as principais diferenças entre os dois paradigmas e diferenças de sintaxe entre as duas linguagens de consultas utilizadas.

Abstract

The amount of information that is available to mankind is enormous, and as human knowledge expands, the greater the amount of information that needs to be stored and analyzed. The relational database can be heavy for applications that need speed, such as web applications and blogs that have several types of attributes, to solve this problem requires an equally agile database. NoSQL is designed for distributed data stores where there is a need to store large amounts of data. The data model does not require a fixed type, and it does not need to be previously defined. NoSQL was not created to extinguish the SQL language, in other words we can use SQL and NoSQL database to achieve the best results. This study shows a comparison between the database paradoxes, presenting an analysis of insertion time, update selection and data removal, with this comparison, it is possible to notice that MongoDB (NoSQL) performed better in the tests than PostgreSQL (relational), it is also note that PostgreSQL has a more complete platform with user and administrator interfaces. The study also presents the main differences between the two paradoxes and syntax differences between the two query languages used.

Sumário

Índice de Figuras

Índice de Tabelas

Índice de Gráficos

Tabela de Símbolos e Siglas

ACID - *Atomic, Consistent, Isolated e Durable.*

ACM - *Association for Computing Machinery.*

APIs - *Application Programming Interfaces.*

BASE - *Basically Available, Soft-State, Eventually Consistent.*

BD – Banco de Dados.

BDs - Bancos de dados.

CAP - *Consistency, Availability, Partition tolerance.*

TotalSeleçãoFinal - Quantidade de estudos selecionados somente sobre as diferenças entre BDs NoSQL e BDs relacionais.

TotalPréSeleção - Quantidade de estudos selecionados sobre bancos NoSQL como um todo.

GHz – GigaHertz.

HTML - *HiperText Markup Language.*

IEEE - *Institute of Electrical and Electronics Engineers.*

JSON - *Java Script Object Notation.*

NOSQL- *Not Only SQL.*

RAM - Memória de Acesso Aleatório.

REST - Representational State Transfer.

SGBDR - Sistemas de Gerenciamento de Banco de Dados Relacional.

SQL - *Structured Query Language.*

TCP - *Transmission Control Protocol.*

XML - *eXtensible Markup Language.*

XPATH - *XML Path Language.*

Capítulo 1

Introdução

A quantidade de informação que está disponível para a humanidade é enorme e à medida que o conhecimento humano se expande, maior é a quantidade dessa informação que precisa ser armazenada e analisada. Além da quantidade, o fluxo e a variedade dessas informações constantemente desafiam a indústria e a academia pois a quantidade de dados manipulados aumenta exponencialmente. Nos bancos de dados relacionais a necessidade de transformar os dados em tabelas pode aumentar a complexidade das operações de seleção de dados e requerer o uso de algoritmos complexos de mapeamento e estrutura. Mesmo quando uma base de dados pode ser coberta pelo modelo relacional, as diversas garantias providas por este modelo podem gerar uma sobrecarga desnecessária para tarefas simples [1].

O banco de dados (BD) relacional pode ser pesado para aplicações que precisam de velocidade, como aplicações *web* e *blogs* que possuem diversos tipos de atributos. Textos, comentários, imagens, vídeos, código fonte e outras informações precisam ser armazenadas em diversas tabelas, e como as aplicações na *web* são muito ágeis, precisam ser amparadas por uma base de dados igualmente ágil e com um *schema* de fácil adaptação [1].

O termo NoSQL (Não Somente SQL) foi usado pela primeira vez por Strozzi em 1998 para citar um BD relacional *open-source* que não utilizava SQL (*Structured Query Language*), o Strozzi *NoSQL* [2]. Em 2009, na conferência conhecida como "*NoSQL Meetup*" e organizada por Johan Oskarsdon [3], o termo foi utilizado novamente, porém referenciando BDs não relacionais. O NoSQL foi projetado para armazenamentos de dados distribuídos, onde há necessidade de armazenar grandes quantidades de dados. O modelo de dados não exige um tipo fixo, não sendo necessário defini-lo previamente. O NoSQL não foi criado para extinguir a linguagem SQL, em outras palavras, podemos usar o BD SQL e NoSQL para alcançar os melhores resultados. Por exemplo, podemos usar BD NoSQL para armazenar enormes quantidades de dados não estruturados e armazenar dados estruturados usando BD SQL, de modo que seja possível fazer bom uso da sintaxe do SQL [4].

Portanto para suprir algumas dificuldades dos BDs relacionais, o uso de BDs No*SQL* torna-se cada vez maior, pois estes têm como prioridade o desempenho e a disponibilidade, características importantes quando é necessário manipular grandes quantidades de dados. Sabe-se ainda que novos paradigmas podem vir acompanhados de alguma resistência o que pode ser modificado com análises comparativa entre o novo paradigma e outro amplamente conhecido, estas análises devem ser realizadas de maneira ampla e clara.

1.1 Objetivos

O objetivo do trabalho é realizar uma pesquisa sobre BDs relacionais e NoSQL, apresentar seus principais conceitos e suas utilizações, realizar comparações de sintaxe e desempenho de leitura e escrita, também serão realizados testes de inserção, seleção, atualização e remoção de dados, sendo assim o trabalho tem como meta oferecer mais um mecanismo de comparação entre os dois modelos de BD.

1.1.1 Objetivos específicos

A seguir é apresentado um conjunto de objetivos específicos deste trabalho:

- Apresentar as principais diferenças entre os BDs relacionais e NoSQL, bem como as diferenças de sintaxe;
- Apresentar um estudo de caso, utilizando os dois paradigmas;
- Apresentar em que casos é melhor utilizar cada um dos paradigmas;

1.2 Estrutura do trabalho

Este trabalho está organizado a partir de uma breve introdução sobre os BDs relacionais e NoSQL, no Capítulo 2 é apresentada uma revisão sistemática cujo objetivo é identificar quais as diferenças entre os BDs relacionais e NoSQL. O Capítulo 3 apresenta uma análise comparativa entre os bancos de dados relacionais e NoSQL. O Capítulo 4 apresenta um estudo de caso para comparação dos dois paradigmas. Por fim, o Capítulo 5 apresenta algumas considerações finais e as perspectivas de trabalhos futuros a respeito deste estudo.

Capítulo 2

Revisão sistemática

Esta revisão sistemática busca identificar quais as diferenças entre os BDs relacionais e os NoSQL, analisando suas características, vantagens e desvantagens.

Segundo Biolchini *et al.* [15], os principais objetivos de uma revisão sistemática são identificar, avaliar e interpretar todas as pesquisas disponíveis em relação ao tema, identificar lacunas na pesquisa atual e fornecer recursos para possíveis novas pesquisas.

2.1 Planejamento da revisão sistemática

O protocolo utilizado para revisão sistemática se baseia no protocolo proposto por Biolchini *et al*. (2005). Logo abaixo será mostrado o planejamento de acordo com esse protocolo:

1. A primeira etapa consiste de uma revisão da literatura a procura de estudos primários utilizando como base a questão principal, que será apresentada a seguir, além de construção das definições que serão usadas para selecionar os estudos ao longo do projeto;
2. Na segunda etapa, definem-se critérios de qualificação (inclusão e exclusão) claros e reproduzíveis para selecionar os estudos da primeira pesquisa feita que abordem o tema com ênfase;
3. Na terceira etapa os critérios eleitos na etapa anterior serão aplicados aos estudos pré-selecionados para que sejam escolhidos aqueles que melhor atendem ao propósito da revisão sistemática;
4. Após aplicar os critérios de qualificação nos estudos pré-selecionados, o material restante será analisado e interpretado, onde o foco principal é catalogar de onde as informações foram retiradas;
5. Esta é a última etapa da revisão sistemática, onde os resultados da revisão serão apresentados;

O resultado da revisão sistemática envolve uma análise dos estudos mais relevantes encontrados e serão utilizados como referências importantes na condução deste trabalho.

2.1.1 Etapa 1. Elaboração da Questão Principal

Nesta etapa, os objetivos da revisão sistemática são definidos através da elaboração da questão central e da qualidade e amplitude desta questão. A questão primária usada nesta revisão sistemática é:

Quais as diferenças entre bancos de dados relacionais e NoSQL?

A qualidade e amplitude da questão são controladas através de alguns dos mecanismos citados em Biolchini *et al*. (2005) e que são apresentados a seguir:

Tamanho da População

O tamanho da população se refere a quantidade de estudos selecionados e leva em consideração duas variáveis:

- TotalPreSeleção (Quantidade de estudos selecionados sobre bancos NoSQL como um todo.);
- TotalSeleçãoFinal (Quantidade de estudos selecionados somente sobre as diferenças entre BDs NoSQL e BDs relacionais.);

Características da População

Os temas dos estudos selecionados nas buscas estão diretamente relacionados aos temas abaixo:

- NoSQL;
- BDs relacionais;

Resultado Esperado

Como consequência direta da extração de dados dos estudos selecionados esperamos alcançar o seguinte resultado:

- Comparar os dois tipos de BDs, os relacionais e os NoSQL, identificando suas características, as vantagens e desvantagens de utilizar cada tipo e identificar em que contextos eles são melhores para utilização;

- A amplitude da questão apresentada alcança empresas que necessitam armazenar grandes quantidades de informações obtidas através de seus produtos desenvolvidos;

2.1.2 Etapa 2. Seleção das Fontes de Dados

Para garantir uma ampla disponibilidade de estudos nas fontes de dados, foi definido o inglês, como idioma em que foram feitos os estudos.

As buscas foram realizadas por meio da submissão de *strings* nas fontes de dados eletrônicas e nas máquinas de buscas conforme a lista a seguir:

Bases de dados eletrônicas indexadas:

- *Association for Computing Machinery* (ACM);
- *Institute of Electrical and Electronics Engineers* (IEEE);

As palavras-chave utilizadas são todas pertinentes ao tema central, ou seja, BDs NoSQL e BDs relacionais, como mostradas na Tabela 1:

Tabela 1. Palavras-chave.

Palavras-chave
NoSQL
SQL databases
Relational Databases
Analysing
Unstructured
Diferences between
Flexible schema

Só foram utilizadas palavras no idioma inglês para construção da *string* de busca. A *string* de busca utilizada foi a seguinte:

(nosql AND relational databases) OR (diferences AND between AND sql AND databases AND no AND sql) OR (analysing AND nosql AND unstructured) OR (nosql AND flexible AND schema)

2.1.3 Etapa 3. Seleção dos estudos

Nesta etapa, os critérios para seleção dos estudos são apresentados. Serão descritos os critérios de inclusão dos estudos, os critérios de exclusão dos estudos, a

definição dos estudos que serão considerados, além dos procedimentos para seleção preliminar.

O critério para inclusão e exclusão de estudos nos resultados das pesquisas é atender a questão primária: Quais as diferenças entre os BDs relacionais e os bancos NoSQL.

Os critérios usados para inclusão ou exclusão dos estudos selecionados são descritos na Tabela 2.

Tabela 2. Critérios para inclusão/exclusão dos estudos pré-selecionados.

Item	Descrição
1	O estudo possui mais de três páginas?
2	O estudo está escrito nos idiomas escolhidos para a pesquisa?
3	O estudo está completo?
4	O estudo se estende até 20 páginas?
5	O estudo possui as referências?
6	A data de publicação é atual?

Os critérios usados para avaliar os estudos pré-selecionados são descritos na Tabela 3.

Tabela 3. Critérios para avaliar os estudos pré-selecionados.

Item	Descrição
1	O estudo menciona os temas que serão utilizados no trabalho, como NoSQL e BDs Relacionais?
2	O estudo se baseia em alguma pesquisa?

O processo para seleção preliminar foi definido por meio da preparação da *string* de busca a partir das palavras-chaves e dos termos relacionados, apresentados anteriormente, para posterior submissão desta *string* realizada nas fontes de estudos primários. Para saber se um determinado estudo é relevante à pesquisa, foi realizada a leitura dos resumos (*abstracts*) e, em caso positivo, estes foram selecionados para posterior leitura completa, segundo os critérios de inclusão e exclusão.

O processo para seleção final considera a leitura de todos os artigos relevantes identificados na seleção preliminar usando como base os critérios da Tabela 3.

2.1.4 Etapa 4. Extração dos resultados

Ao término da seleção final foi realizada uma análise de cada estudo onde foram considerados alguns itens como: identificação, explanação sobre o assunto e relevância ao tema proposto que são explorados mais à frente. O formulário usado na extração dos dados é mostrado na Tabela 4.

Tabela 4. Campos do formulário para extração de dados.

Identificação	**Título do artigo ou texto em questão**
Autor(es)	Indicar pessoa, grupo ou organização autora do artigo ou texto.
Características extraídas [Número da referência]	
Síntese do item em questão, apresentando as suas características principais extraídas.	

2.2 Condução da revisão sistemática

Nesta seção será apresentado a fase de buscas dos artigos que foram utilizados como base teórica para compor este trabalho. Será apresentado separadamente o processo executado em cada uma das livrarias digitais utilizadas.

2.2.1 Processo de busca

A busca na biblioteca digital *Institute of Electrical and Electronics Engineers* (IEEE), foi realizada através do endereço eletrônico http://ieeexplore.ieee.org/search/advsearch.jsp [19], no mês de outubro de 2016. A Figura 1 apresenta o mecanismo de busca do IEEE.

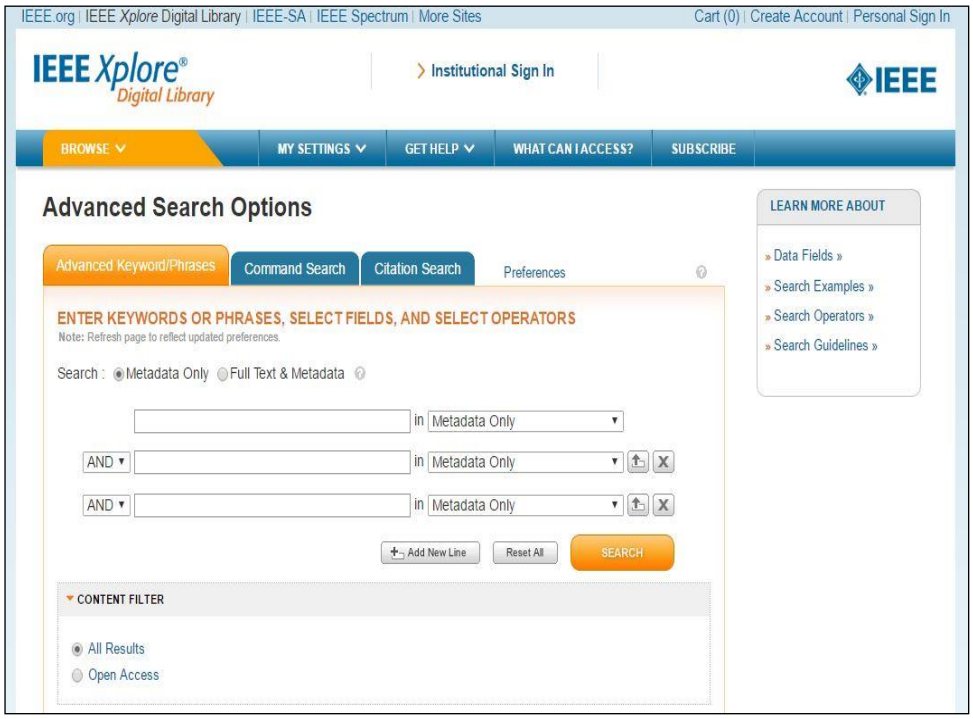

Figura 1. Busca no *Institute of Electrical and Electronics Engineers* (IEEE).

Foram retornados 142 resultados, a maioria disponíveis para *download*, dentre esses foram selecionados 79 através das leituras dos resumos.

A busca na biblioteca digital *Association for Computing Machinery* (ACM) foi realizada no mês de outubro de 2016, por meio do endereço eletrônico http://portal.acm.org/advsearch.cfm [20]. A Figura 2 apresenta o mecanismo de buscas da ACM. Utilizando a *string* de busca foram retornados 74 resultados. Dentre eles foram escolhidos 28 através da leitura dos abstracts.

Para a elaboração deste trabalho foram selecionados cerca de cento e sete estudos, portanto TotalPréSeleção = 107. Do total de 107 estudos selecionados apenas um grupo com 17 estudos tratava especificamente das diferenças e análises entre bancos relacionais e NoSQL, portanto TotalSeleçãoFinal = 17.

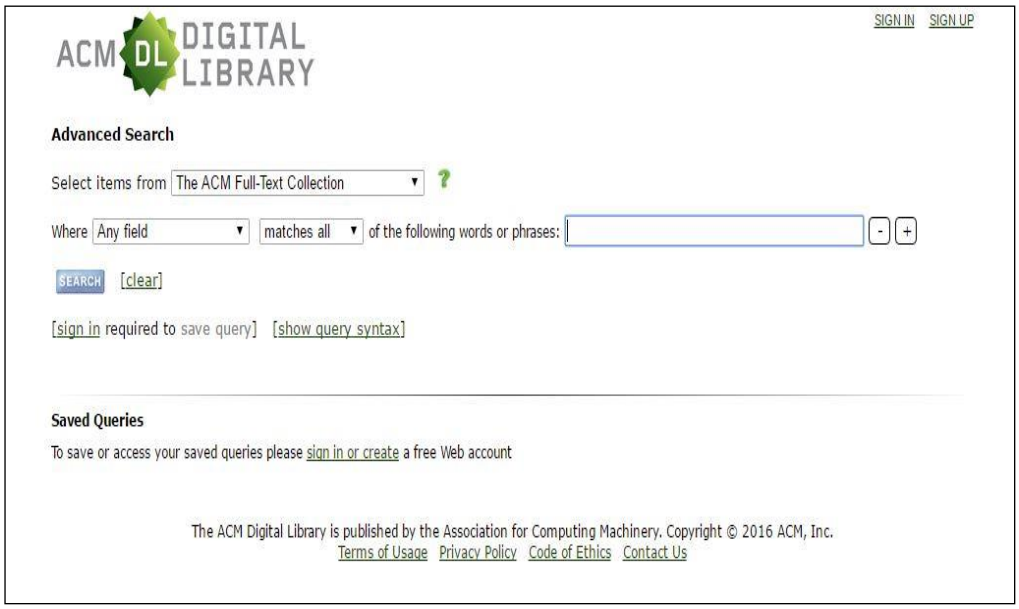

Figura 2. Busca na Biblioteca Digital *Association for Computing Machinery* (ACM).

2.2.2 Seleção preliminar

Conforme o planejamento da revisão sistemática para esta etapa, todos os 17 estudos selecionados na etapa anterior foram lidos por completo, dentre os 17, 10 foram selecionados para serem utilizados como referência, os outros não foram selecionados, porque repetiam conteúdo de outros artigos, por isso foram escolhidos os artigos mais completos dentre os que apresentavam conteúdo semelhante. A lista a seguir apresenta a lista de títulos candidatos à leitura completa:

1. *MongoDB vs Oracle -- Database Comparison* [5];
2. *A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment* [6];
3. *A Comparative Study: MongoDB vs. MySQL* [7];
4. *QODM: A Query-Oriented Data Modeling Approach for NoSQL Databases* [8];
5. *NoSQL evaluation: A use case oriented survey* [9];
6. *A study into the capabilities of NoSQL databases in handling a highly heterogeneous tree* [10];
7. *Data modelling for discrete time series data using Cassandra and MongoDB* [11];
8. *Analysis of various NoSql database* [12];
9. *Comparing NoSQL MongoDB to an SQL DB* [13];
10. *NoSQL Databases: MongoDB vs Cassandra* [4]*;*

2.2.3 Extração de dados

Nesta etapa da revisão sistemática foi realizada a extração dos dados a partir da leitura dos textos selecionados na etapa anterior. Para isso foi utilizado o formulário de extração de dados mostrado anteriormente com o objetivo de organizar as informações obtidas na leitura dos textos. Todos os textos selecionados na seção anterior foram lidos por completo. A Tabela 5 apresenta os dados quem foram extraídos através da leitura dos artigos:

Tabela 5. Dados extraídos da revisão sistemática.

Estudo nº 1	*MongoDB vs Oracle – Database Comparison*
Autor(es)	Alexandru Boicea, Florin Radulescu, Laura Ioana Agapin
Características extraídas [5]	
Principais diferenças A principal diferença é que o NoSQL é uma categoria de mecanismos de BD que não suportam o SQL, a fim de obter recursos de desempenho ou confiabilidade incompatíveis com a flexibilidade do SQL. Esses mecanismos geralmente fornecem uma linguagem de consulta que fornece um subconjunto do que o SQL pode fazer, além de alguns recursos adicionais. *JOIN*, *TRANSACTION* e *LIMIT* normalmente não são suportados pelos bancos NoSQL. Alguns NoSQL têm importação automática de recursos SQL. Os dados SQL são armazenados em tabelas com uma estrutura fixa. Pode-se definir relações entre as tabelas a partir da mesma base de dados. As relações entre as tabelas podem ter relações com chaves estrangeiras. É possível definir a restrição *PRIMARY KEY* nas tabelas. Uma tabela pode ter uma chave primária composta de uma ou mais colunas da tabela. Pode haver outras restrições declaradas nas colunas de uma tabela, como *UNIQUE*, *FOREIGN KEY ou NOT NULL*. Nos BDs NoSQL os dados são armazenados usando coleções. Por exemplo, as coleções do MongoDB não têm restrições quanto aos dados armazenados nelas. As coleções não têm uma estrutura fixa. Os campos podem ter diferentes tipos de dados. O artigo também apresenta as diferenças de sintaxe das linguagens de consultas utilizadas, na inserção, seleção, atualização e remoção de registros, entre um banco de dados relacional e um BD NoSQL orientado à documentos.	
Estudo nº 2	*A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment*
Autor(es)	Min-Gyue Jung, Seon-A Youn, Jayon Bae, Yong-Lak Choi

Características extraídas [6]
O conceito de *Big data*, inicialmente recebeu atenção pela capacidade de manipular grande volume de dados. No entanto, devido ao desenvolvimento de vários dispositivos eletrônicos, não só os dados de texto, mas também vários dados - dados SNS e dados de sensores de fluxo que não se encaixam nos formatos de relatório existentes - estão sendo gerados em tempo real. O Sistema de Gerenciamento de Bando de Dados Relacional (SGBDR) que tem sido utilizado tem problemas na estruturação de dados não estruturados e tem problemas de desempenho e custo no processamento de dados maciços. Tendo função de mapeamento de memória, NoSQL executa leitura e gravação rápida, o que torna NoSQL adequado para o processamento de dados grandes. Além disso, ao contrário do RDBMS que processava principalmente dados estruturados, o NoSQL pode manipular dados não estruturados com mais facilidade. Assim, muitas companhias que constroem o sistema de dados grande que inclui dados não estruturados e de sensor, tendem a fazer análise da vantagem de características de NoSQL.

Estudo nº 3	*A Comparative Study: MongoDB vs. MySQL*
Autor(es)	Cornelia Gyorödi, Robert Gyorödi, George Pecherle, Andrada Olah
Características extraídas [7]	
NoSQL, não é uma ferramenta, mas uma metodologia composta de várias ferramentas complementares e concorrentes. A principal vantagem de um BD NoSQL é que, ao contrário de um BD relacional, ele pode lidar com dados não estruturados, como documentos, e-mail, multimídia e mídia social eficientemente. Bases de dados não relacionais não utilizam os princípios RDBMS (*Relational Data Base Management System*) e não armazenam dados em tabelas, o esquema não é fixo e tem um modelo de dados muito simples. Em vez disso, eles usam chaves de identificação e os dados podem ser encontrados com base nas chaves atribuídas.	

Estudo nº 4	*QODM: A Query-Oriented Data Modeling Approach for NoSQL Databases*
Autor(es)	Xiang Li, Zhiyi Ma, Hongjie Chen
Características extraídas [8]	
Principais vantagens do NoSQL são os seguintes aspectos: 1) a capacidade de escala horizontal 2) a capacidade de replicar e distribuir dados (partição) em muitos servidores; 3) uma interface ou protocolo de nível de chamada simples (em contraste com uma chamada SQL); 4) um modelo de concorrência mais fraco do que as transações ACID (*Atomic, Consistent, Isolated and Durable)* da maioria dos sistemas de BD relacional (SQL); 5) uso eficiente de índices distribuídos e RAM (Memória de Acesso Aleatório) para armazenamento	

de dados; 6) a capacidade de adicionar dinamicamente novos atributos a registros de dados. As bases de dados NoSQL não cumprem o princípio ACID, mas mantêm os princípios do CAP (Consistência, Disponibilidade, e tolerância a falhas) e BASE (Basicamente disponível, Estado leve e Eventualmente consistente).

Estudo nº 5	*NoSQL Evaluation A Use Case Oriented Survey*
Autor(es)	Robin Hecht; Stefan Jablonski

Características extraídas [9]

Possibilidades de consultas

Devido ao seu modelo de dados simples, as APIs de armazenamentos de chave-valor fornecem somente operações baseadas em *put*, *get* e *delete*. Qualquer linguagem de consulta seria uma sobrecarga desnecessária para esses tipos. Se forem necessárias funcionalidades de consulta adicionais, elas precisam ser implementadas na camada de aplicação, o que pode levar rapidamente a muito mais complexidade do sistema e perda de desempenho. Portanto, os armazenamentos de chave-valor não devem ser usados, se forem necessárias consultas mais complexas.

Os NoSQL orientado à documentos oferecem APIs muito mais ricas. As consultas de intervalo em valores, índices secundários, consulta de documentos aninhados e operações como "e", "ou", "entre" são recursos, que podem ser usados convenientemente. As consultas dos NoSQL Riak e MongoDB podem ser estendidas com expressão regular. Riak oferece funcionalidades para atravessar ligações entre documentos facilmente. As interfaces REST também são suportadas por NoSQL baseados em documentos.

Os NoSQL orientados a colunas fornecem apenas consultas de intervalo e algumas operações como "in", "e / ou" e expressão regular, se forem aplicadas em chaves de linha ou valores indexados. Mesmo que cada NoSQL da família de colunas ofereça um SQL como linguagem de consulta para fornecer uma interação mais conveniente do usuário, somente as chaves de linha e os valores indexados podem ser considerados nessas consultas. Porém ainda não existe uma linguagem de consulta comum para os NoSQL da família de colunas disponíveis.

Os BDs de grafo podem ser consultados de duas maneiras diferentes. As estratégias de correspondência de grafos tentam encontrar partes do grafo original, que correspondem a um padrão de grafo definido. A travessia do grafo, por outro lado, começa a partir de um nó escolhido e percorre o grafo de acordo com uma descrição. As estratégias transversais diferem na detecção de um nó correspondente o mais rápido possível (largura primeiro) e em encontrar o caminho mais curto (profundidade primeiro). A maioria dos BDs baseados

em grafos oferece APIs REST, interfaces específicas de linguagem de programa e armazena linguagens de consulta específicas para acessar dados usando uma das duas estratégias descritas. Em contraste com outros BDs NoSQL, existem algumas linguagens de consulta, que são suportadas por mais de um BD grafo. SPARQL é uma linguagem de consulta popular, declarativa com uma sintaxe muito simples fornecendo correspondência de padrão de grafo. É suportado pela maioria das dos NoSQL baseados em grafos e por Ne04j. Gremlin é uma linguagem de programação imperativa usada para executar travessias gráficas baseadas em XPATH.

Estudo nº 6	*A Study Into the Capabilities of NoSQL Databases in Handling a Highly Heterogeneous Tree*
Autor(es)	Dileepa Jayathilake, Charith Sooriaarachchi, Thilok Gunawardena, Buddhika Kulasuriya e Thusitha Dayaratne

Características extraídas [10]

NoSQL baseado em grafos

Um dos tipos de BD NoSQL mais populares é o BD baseado em grafos. Eles se assemelham a estruturas semelhantes a grafos encontradas na vida cotidiana, como redes humanas e redes rodoviárias. Um BD orientado a grafos pode ser definido como uma estrutura, que consiste de nós, arestas e propriedades para representar e armazenar dados. Os nós representam registros na analogia RDBMS. Bordas ou relações são derivadas de alguma coluna predefinida em um nó. Isso permite que os BDs de orientados a grafos gerenciem complexos relacionamentos muitos-para-muitos que são difíceis de serem absorvidos na maioria dos outros paradigmas de BD. Com essa força inerente, os BDs baseado em grafos são super rápidos ao lidar com dados associativos. Como os BDs baseado em grafos não precisam de operações caras como as junções no RDBMS, eles são altamente escaláveis e os dados *ad-hoc* podem ser facilmente gerenciados com um BD baseado em grafos.

Estudo nº 7	*Data Modelling for Discrete Time Series Data Using Cassandra and MongoDB*
Autor(es)	Dharavath Ramesh, Ashay Sinha, Suraj Singh

Características extraídas [11]

O teorema CAP afirma que em um ambiente distribuído, um sistema pode garantir no máximo duas das três propriedades a seguir:

- Consistência (C): O resultado de alterações feitas a qualquer parte do BD deve ser refletido em toda a base de dados.
- Disponibilidade (A): Se o BD estiver disponível, os usuários do BD devem ser capazes de acessar os dados.

- Tolerância de Partição (P): Em caso de partições arbitrárias resultantes devido a falha de rede, o sistema continua a funcionar sem problemas.

O Teorema CAP é o princípio básico de base de dados NoSQL. Este teorema também é conhecido como o Brewer's *Theorem* .

Estudo nº 8	*Analysis of Various NoSql Database*
Autor(es)	Pragati Prakash Srivastava; Saumya Goyal; Anil Kumar

Características extraídas [12]

O artigo apresenta os seguintes tipos de NoSQL:

Chave-valor

Essas bases de dados mapeiam uma chave para um valor ou um conjunto de valores. As chaves são únicas e atômicas, o que significa que elas são indestrutíveis. Eles são usados para consultar as entradas nos BDs de armazenamento chave-valor. O armazenamento de chave-valores fornece uma implementação de tabela *hash* com pares de valor-chave espalhados por vários servidores remotos em um cluster distributivo. Assim, eles podem alcançar a eficiência requerida fornecendo solicitações de leitura e gravação aleatórias extremamente rápidas e a flexibilidade para armazenar dados em um formato sem esquema. Uma vez que diferentes pares de valor de chave armazenam um conjunto de dados não relacionados isso evita as operações de junção SQL e GROUP BY, bem como referências de chaves estrangeiras.

Família de colunas

A ideia por trás de BDs de armazenamento relacionais era que eles armazenavam todas as entradas de tabela associadas a uma única linha em conjunto no disco, ou seja, todas as entradas de coluna associadas a uma identificação de linha específica seriam armazenadas em conjunto. No caso de organizações bancárias ou de financiamento que tinham de manter um conjunto enorme de registros relacionados, nem sempre era garantido que todos os valores fossem armazenados de forma consecutiva. No caso do BD da família de colunas, uma coluna inteira de uma tabela é armazenada em conjunto e mapeada para uma única chave. Uma vez que todas as entradas em colunas têm índices, é possível pesquisar apenas uma parte da tabela. Também uma coluna pode ter hierarquias de colunas dentro dele tornando-se a super coluna. Isso fornece pesquisas fáceis e acesso rápido, evitando gastos gerais desnecessários para procurar a chave individual de um registro.

Orientado a documentos

O BD de documentos é uma alternativa aos BDs relacionais e é usado para armazenar dados semiestruturados existentes na forma de XML (*Extensible Markup Language*), JSON (*Java Script Object Notation*) ou em outros formatos semelhantes. Um documento pode ser comparado a uma linha de um BD relacional que contém todas as informações relacionadas a documentos. Uma coleção compreende vários documentos onde cada documento pode ter esquemas diferentes e diferem no número e tipo de dados que estão sendo armazenados. É especialmente otimizado para armazenar informações textuais. Uma vez que o conjunto de dados relacionados é armazenado em conjunto, salva os overheads da operação SQL JOIN. Embora o BD tenha um design livre de esquema, os registros armazenados são semiestruturados e existem na forma de hierarquias.

Baseado em grafos

Os BDs baseados em grafos são mais adequados para percorrer e pesquisar aplicativos, como encontrar links relacionados no *LinkedIn*, procurar amigos no *Facebook*, etc. Isso dá mais importância à relação entre itens de dados e não dados. Eles são altamente otimizados para percorrer rápido e fazer uso eficiente dos algoritmos de grafos, como o caminho mais curto primeiro, a fim de encontrar a ligação entre as informações.

O artigo também apresenta exemplos de cada tipo de NoSQL disponíveis no mercado.

Estudo nº 9	*Comparing NoSQL MongoDB to an SQL DB*
Autor(es)	Zachary Parker, Scott Poe, Susan V. Vrbsky

Características extraídas [13]

Os dados no modelo relacional são geralmente representados por um esquema de BD, a fim de capturar a semântica do BD. Objetos no BD com o mesmo número de características, tipo e formato são agrupados, tornando-os dados estruturados. O modelo relacional baseia-se nesta suposição de dados estruturados, com os seus dados armazenados nas linhas e colunas de uma tabela, pelo que cada linha tem o mesmo número e tipo de colunas de dados.

Tabelas em BDs relacionais normalmente são normalizadas, o que resulta na criação de várias tabelas. A consulta dessas tabelas requer a busca e combinação de informações de muitas tabelas diferentes. A combinação de informações com base em um valor de correspondência para uma chave primária e uma chave estrangeira em várias tabelas no modelo relacional requer o uso de uma operação de associação. Quanto maior o esquema e mais tabelas que precisam ser associadas, mais demora para o BD relacional para buscar os dados.

O NoSQL pode ajudar a lidar com dados que não estão estruturados. Os dados podem ser semiestruturados, de tal forma que objetos de dados semelhantes podem ser agrupados, mas os objetos podem ter características diferentes. Informações de esquema também podem ser misturadas com valores de dados em dados semiestruturados, como encontrados em dados XML. Dados não estruturados podem ser de qualquer tipo e não podem ter formato. Esses dados não podem ser representados por qualquer tipo de esquema, como páginas da *Web* em HTML.

Os recursos típicos de BDs SQL, como as propriedades ACID, exigem uma certa sobrecarga e são relaxados ou eliminados em BDs NoSQL para maximizar o desempenho. Muitas bases de dados NoSQL organizam os dados em pares chave-valor. A chave é usada para identificar de forma exclusiva um item de dados específico e o valor pode ser uma palavra simples, número ou uma estrutura complexa com semântica exclusiva. O desenvolvimento de consultas é mais complexo, não há nenhuma linguagem de consulta padrão, e há limites para as operações. Especificamente, não existe nenhuma operação de associação. No entanto, em geral o processamento é mais simples, mais acessível e mais flexível.

Estudo nº 10	*NoSQL Databases: MongoDB vs Cassandra*
Autor(es)	Veronika Abramova, Jorge Bernardino

Características extraídas [14]

Bases de dados relacionais armazenam dados como um conjunto de tabelas, cada uma com informações diferentes. Todos os dados estão relacionados para que seja possível acessar informações de tabelas diferentes simultaneamente. O modelo relacional baseia-se no conceito de "relação". Então, basicamente, uma relação é uma tabela organizada em colunas e linhas. Cada tabela é formada por um conjunto de tuplas com os mesmos atributos. Esses atributos contêm informações sobre algum objeto. Mais complexo BD contém um monte de tabelas com milhões de entradas. Essas tabelas são conectadas para que os dados de uma tabela possam estar relacionados a outros por chave. Existem diferentes tipos de chaves, mas essencialmente existem de dois tipos: chave primária e chave estrangeira. A chave primária é usada para identificar cada tabela inteira, tupla, como única. Chave externa é usada para tabelas de referência cruzada. Chave estrangeira em uma tabela representa uma chave primária na outra.

Enquanto o volume de dados aumenta exponencialmente, alguns problemas se tornaram evidentes. Um deles é o desempenho de BD relacionado ao acesso a dados e estrutura básica do modelo relacional. O SQL permite a extração fácil de dados, mas quando o volume de informações é enorme, o tempo de execução da consulta pode se tornar lento. Qualquer

aplicação com grande quantidade de dados inevitavelmente perderá o desempenho. Para superar esses problemas de eficácia, emergiram diferentes tipos de BDs. Um deles é conhecido como NoSQL correspondente *"Not Only SQL".* NoSQL foi introduzido por Carlo Strozzi em 1980 para referir um BD de código aberto que não estava usando a interface SQL.

ACID vs BASE

Bases de dados relacionais são baseadas em um conjunto de princípios para otimizar o desempenho. Princípios usados pelas bases de dados relacionais ou NoSQL são derivados do teorema CAP. De acordo com este teorema, as seguintes garantias podem ser definidas:

- **Consistência** - todos os nós têm os mesmos dados ao mesmo tempo;
- **Disponibilidade** - todos os pedidos têm resposta;
- **Tolerância de partição** - se parte do sistema falhar, todo o sistema não entrará em colapso.

ACID é um princípio baseado no teorema CAP e usado como conjunto de regras para transações de BD relacional. As garantias ACID são:

- **Atômico** - uma transação é concluída quando todas as operações são concluídas, caso contrário, a reversão é executada;
- **Consistente** - uma transação não pode recolher o BD, caso contrário, se a operação for ilegal, a reversão é executada;
- **Isolado** - todas as transações são independentes e não podem afetar-se mutuamente;
- **Durabilidade** - quando o *commit* é executado, as transações não podem ser desfeitas.

É notório que, para ter BD robusto e correto, essas garantias são importantes. Mas quando a quantidade de dados é grande, ACID pode ser difícil de alcançar. Por isso, NoSQL se concentra no princípio BASE:

- **Basicamente disponível** - todos os dados são distribuídos, mesmo quando há uma falha o sistema continua a funcionar;
- **Estado leve** - não há garantia de consistência;
- **Eventualmente consistente** - o sistema garante que, mesmo quando os dados não são consistentes, eventualmente será.

É importante notar que a BASE segue o teorema CAP e se o sistema é distribuído, duas das três garantias devem ser escolhidas. O que escolher depende de necessidades pessoais e

propósito de BD. BASE é mais flexível que ACID e a grande diferença é sobre a consistência. Se a consistência for crucial, os BDs relacionais podem ser uma solução melhor, mas quando há centenas de nós em um cluster, a consistência se torna muito difícil de realizar.

Capítulo 3

Bancos de Dados Relacionais vs NoSQL

Nos últimos anos muito tem se falado do conceito de *BigData* principalmente pelo grande volume de dados que precisam ser armazenados, e muitos destes dados não são estruturados. No entanto, devido ao desenvolvimento de vários dispositivos eletrônicos, estão sendo gerados em tempo real, não só os dados de texto, mas dados de sensores de fluxo que não se encaixam nos formatos de relatório existentes [6].

O Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR) que tem sido utilizado tem problemas na estruturação de dados não estruturados e tem problemas de desempenho e custo no processamento de dados maciços. Tendo função de mapeamento de memória, NoSQL executa a leitura e gravação rápida, o que torna NoSQL adequado para o processamento de dados grandes. Além disso, ao contrário do SGBDR que processava principalmente dados estruturados, o NoSQL pode manipular dados não estruturados com mais facilidade. Assim, muitas companhias que constroem um sistema de dados grande que inclui dados não estruturados, tendem a fazer uma consulta da vantagem de características de NoSQL[6]. Este capítulo apresenta as características dos dois conceitos e suas diferenças.

3.1 Banco de Dados Relacionais

Bases de dados relacionais armazenam dados como um conjunto de tabelas, cada uma com informações diferentes. Todos os dados estão relacionados para que seja possível acessar informações de tabelas diferentes simultaneamente. O modelo relacional baseia-se no conceito de "relação". Então, basicamente, uma relação é uma tabela organizada em colunas e linhas. Cada tabela é formada por um conjunto de tuplas com os mesmos atributos, como é mostrado na Tabela 6. Esses atributos

contêm informações sobre algum objeto. Mais complexo BD contém um monte de tabelas com milhões de entradas [14].

Tabela 6. Tabela de um BD Relacional.

IDdoProduto	NomeProduto	IDFornecedor	IDCategoria	QuantidadePorUnidade	PreçoUnidade	UnidadeEstoque	UnidadesEmOrdem	NivelDeReposicao	Descontinuado
1	Dove	472	3	1	2.00	300	100	50	T
2	Tang	827	2	1	1.50	600	500	100	F
3	Coca	453	2	1	6.50	640	200	150	T
4	Sundown	647	3	1	45.0	300	100	30	T

Essas tabelas são conectadas para que os dados de uma tabela possam estar relacionados a outros por chave. Existem diferentes tipos de chaves, mas essencialmente existem dois tipos: chave primária e chave estrangeira. A chave primária é usada para identificar na tabela inteira, cada tupla, como única. Chave estrangeira é usada para tabelas de referência cruzada. Chave estrangeira em uma tabela pode representar uma chave primária na outra [14].

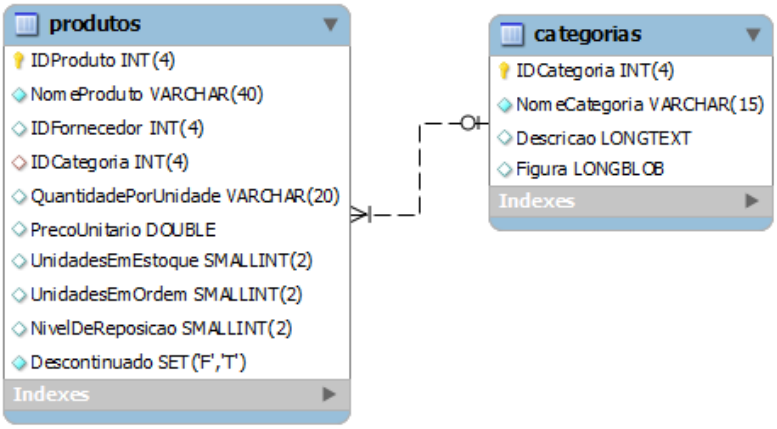

Figura 3. Modelo de um BD relacional.

A Figura 3 mostra duas tabelas uma de produtos e outra de categorias, onde IDProduto e IDCategoria são as chaves de produtos e categoria respectivamente, e o atributo IDCategoria que está contido na tabela de produtos é uma chave estrangeira de categorias.

Os dados no modelo relacional são geralmente representados por um esquema de BD, os objetos no BD com o mesmo número de características, tipo e formato são agrupados, tornando-os dados estruturados. O modelo relacional baseia-se nesta

suposição de dados estruturados, com os seus dados armazenados nas linhas e colunas de uma tabela, pelo que cada linha tem o mesmo número e tipo de colunas de dados [13].

Tabelas em BDs relacionais normalmente são normalizadas, o que resulta na criação de várias tabelas. A consulta dessas tabelas requer a busca e combinação de informações de muitas tabelas diferentes. A combinação de informações com base em um valor de correspondência para uma chave primária e uma chave estrangeira em várias tabelas no modelo relacional requer o uso de uma operação de associação. Quanto maior o esquema e mais tabelas que precisam ser associadas, mais demora para o BD relacional para buscar os dados [13].

3.2 NoSQL

Enquanto o volume de dados aumenta exponencialmente, alguns problemas se tornaram evidentes. Um deles é o desempenho de BD relacionado ao acesso a dados e estrutura básica do modelo relacional. O SQL permite a extração fácil de dados, mas quando o volume de informações é enorme, o tempo de execução da consulta pode se tornar lento. Qualquer aplicação com grande quantidade de dados inevitavelmente perderá o desempenho. Para superar esses problemas de eficácia, emergiram diferentes tipos de BDs. Um deles é conhecido como NoSQL correspondente "*Not Only SQL*". NoSQL foi introduzido por Carlo Strozzi em 1980 para referir um BD de código aberto que não estava usando a linguagem SQL [14].

NoSQL, não é uma ferramenta, mas uma metodologia composta de várias ferramentas complementares e concorrentes. A principal vantagem de um BD NoSQL é que, ao contrário de um BD relacional, ele pode lidar com dados não estruturados, como documentos, e-mail, multimídia e mídia social eficientemente. Bases de dados não relacionais não utilizam os princípios dos SGBDR e não armazenam dados em tabelas, o esquema não é fixo e tem um modelo de dados muito simples. Em vez disso, eles usam chaves de identificação e os dados podem ser encontrados através dessas chaves [7].

A implementação de BD mais comum hoje é baseada no modelo relacional que usa SQL como sua linguagem de consulta. No entanto, as soluções de BD NoSQL

estão se tornando mais proeminentes à medida que quantidades maciças de dados em rápido crescimento estão sendo coletadas hoje. Estes dados são tipicamente não estruturados, complexos e não se encaixam bem no modelo relacional. Exemplos deste tipo de dados são registros de telefones inteligentes em que o local é transmitido a cada poucos segundos, alimentação de câmeras de vídeo em espaços públicos e até mesmo o enorme número de páginas e documentos na *web*. No entanto, existe também uma enorme quantidade de BDs de tamanho modesto com dados estruturados que ainda precisam ser armazenados e processados em um BD [13].

Tendo função de mapeamento de memória, NoSQL executa leitura e gravação rápida, o que torna NoSQL adequado para o processamento de dados grandes. Além disso, ao contrário do SGBDR que processa principalmente dados estruturados, o NoSQL pode manipular dados não estruturados com mais facilidade [6].

Principais vantagens do NoSQL são os seguintes aspectos:

- A capacidade de escala horizontal
- A capacidade de replicar e distribuir dados (partição) em muitos servidores;
- Uma interface ou protocolo de nível de chamada simples (em contraste com uma chamada SQL);
- Um modelo de concorrência mais fraco do que as transações da maioria dos sistemas de BD relacional (SQL);
- Uso eficiente de índices distribuídos e RAM (Memória de Acesso Aleatório) para armazenamento de dados;
- A capacidade de adicionar dinamicamente novos atributos a registros de dados.

O NoSQL pode ajudar a lidar com dados que não estão estruturados. Os dados podem ser semiestruturados, de tal forma que objetos de dados semelhantes podem ser agrupados, mas os objetos podem ter características diferentes. Informações de esquema também podem ser misturadas com valores de dados em dados semiestruturados, como encontrados em dados XML (*eXtensible Markup Language*). Dados não estruturados podem ser de qualquer tipo e não podem ter formato. Esses dados não podem ser representados por qualquer tipo de esquema, como páginas da *Web* em HTML (*HiperText Markup Language*) [13].

A escalabilidade horizontal dos NoSQL é baseada em uma arquitetura do tipo memória distribuída com replicação e fragmentação dos dados em diferentes servidores (CATTELL, 2010) [16], permitindo suportar um grande número de operações de I/O (*Input /Output*) por segundo. A maioria desses bancos não fornecem suporte às propriedades transacionais ACID, para conseguirem um desempenho e uma escalabilidade maior. Os SGBD NoSQL operam sobre a relação desempenho e complexidade do modelo, tendendo sempre a aumentar a primeira. CATTELL (2010) define seis das características mais importantes dos bancos de dados NoSQL [16]:

- Aumento de desempenho de operações simples quando do aumento do número de nós;
- Replicação e distribuição de dados em diferentes nós;
- No lugar do SQL, um protocolo simples de comunicação com o SGBD;
- Um modelo de controle de concorrência mais “relaxado” do que os utilizados nos SGBDR tradicionais, chamados consistência tardia;
- Uma distribuição eficiente dos índices e utilização de memória RAM para armazenamento de dados;
- Adição dinâmica de atributos aos registros já existentes na base (pela não-obrigação de ter esquemas fixos).

O termo "escalabilidade horizontal" significa a capacidade de distribuir tanto os dados como as cargas das operações em muitos servidores, sem RAM ou disco compartilhado entre os servidores. A escala horizontal difere da escala "vertical", onde um sistema de banco de dados utiliza muitos núcleos e/ou CPUs que compartilham RAM e discos [16].

Os recursos típicos de BDs SQL, como as propriedades ACID (*Atomic, Consistent, Isolated and Durable*) exigem uma certa sobrecarga e são relaxados ou eliminados em BDs NoSQL para maximizar o desempenho. Muitas bases de dados NoSQL organizam os dados em pares chave-valor. A chave é usada para identificar de forma exclusiva um item de dados específico e o valor pode ser uma palavra simples, número ou uma estrutura complexa com semântica exclusiva. O desenvolvimento de consultas é mais complexo, não há nenhuma linguagem de consulta padrão, e há limites para as operações. Especificamente, não existe

nenhuma operação de associação. No entanto, em geral o processamento é mais simples, mais acessível e mais flexível [13].

3.2.1 Tipos de NoSQL

Chave-valor

Essas bases de dados mapeiam uma chave para um valor ou um conjunto de valores, como apresentado na Figura 4. As chaves são únicas e atômicas, o que significa que elas são indestrutíveis. Eles são usados para consultar as entradas nos BDs de armazenamento chave-valor. O armazenamento de chaves-valores fornece uma implementação de tabela *hash* com pares de chave-valor espalhados por vários servidores remotos em um cluster distributivo. Assim, eles podem alcançar a eficiência requerida fornecendo solicitações de leitura e gravação aleatórias extremamente rápidas e a flexibilidade para armazenar dados em um formato sem esquema. Uma vez que diferentes pares de valor de chave armazenam um conjunto de dados não relacionados evita as operações de junção SQL e *GROUP BY*, bem como referências de chaves estrangeiras [12].

Figura 4. NoSQL chave-valor.

Devido ao seu modelo de dados simples, as APIs (*Application Programming Interface*) de armazenamentos de chave-valor fornecem somente operações baseadas em *put*, *get* e *delete*. Qualquer linguagem de consulta seria uma sobrecarga desnecessária para esses tipos. Se forem necessárias funcionalidades de consulta adicionais, elas precisam ser implementadas na camada de aplicativo, o que pode levar rapidamente a muito mais complexidade do sistema e penalidades de

desempenho. Portanto, os armazenamentos de chaves-valores não devem ser usados, se forem necessárias consultas ou consultas mais complexas [9].

Redis é um sistema de armazenamento do tipo chave-valor. É um dos BDs mais rápidos em operações na memória. Esse BD oferece velocidade rápida para suportar leitura e gravação em grandes cargas de trabalho. Ao contrário de outras bases de dados NoSQL Redis também é chamado como o servidor de estrutura de dados, uma vez que oferece uma grande diversidade de estruturas de dados no armazenamento de chaves. As chaves podem conter *Strings* (*int*, *long*, *char array*, lista vinculada), *Hashes* (*long int*, *double, char array*), listas, conjuntos e conjuntos ordenados. Alguns recursos do Redis incluem [12]:

- Durante o armazenamento de dados no servidor, os usuários podem especificar o tempo após o qual esse conjunto de registros será descartado automaticamente;
- Uma vez que é um BD na memória, ele suporta bloqueio e escrita simultânea de dados e suporta o processamento em lote de dados usando comandos como EXEC, MULTI e WATCH. Uma vez que todas as tarefas são concluídas, toda a transação é acrescentada ao disco;
- O protocolo binário que ele usa para se comunicar com outros nós é altamente eficiente; no entanto o seu valor é limitado por um tamanho de 512 *Mbytes*;
- Um único sistema com 2,5 GHz (Gigahertz) RAM pode executar 1000-4000 escritas por segundo no Redis;

Redis é mais recomendado e adequado para o seguinte:

- Para realizar análises rápidas em tempo real, o Redis fornece o recurso avançado de análise de visualização da contagem exata de usuários usando um site, em menos de um segundo;
- Para distribuição e gerenciamento de conteúdo, cache e armazenamento de *cookies*;
- Para aplicações que envolvem um número muito elevado de gravações que irá entregar alto desempenho, no entanto o tamanho do conjunto de dados não pode ser maior do que o tamanho da memória principal;

Redis não é muito recomendado nem adequado para o seguinte:

- Operações transacionais altamente seguras no *dataset* já que clientes não confiáveis também podem acessar o soquete TCP (*Transmission Control Protocol)* / Unix;
- Redis compromete a disponibilidade de dados, pois no caso de uma falha de rede se o endereço (número de porta) do nó mestre mudar, uma intervenção manual é necessária e a configuração precisa ser reconfigurada;

Família de colunas

A ideia por trás de BDs de armazenamento relacionais era que eles armazenavam todas as entradas de tabela associadas a uma única linha em conjunto no disco, ou seja, todas as entradas de coluna associadas a uma identificação de linha específica seriam armazenadas em conjunto. No caso de organizações bancárias ou de financiamento que tinham de manter um conjunto enorme de registros relacionados, nem sempre era garantido que todos os valores fossem armazenados de forma consecutiva. No caso do BD da família de colunas, uma coluna inteira de uma tabela é armazenada em conjunto e mapeada para uma única chave, como apresentado na Figura 5. Uma vez que todas as entradas em colunas têm índices, é possível pesquisar apenas uma parte da tabela. Também uma coluna pode ter hierarquias de colunas dentro dela tornando-se a super coluna. Isso fornece pesquisas fáceis e acesso rápido, evitando gastos gerais desnecessários para procurar a chave individual de um registro [12].

Chaves de linha **Colunas**

Recife	**Nome**	**Idade**	**Estado**
	Lucas	33	Pernambuco
São Paulo	**Nome**	**Idade**	**Estado**
	Franco	45	São Paulo
Belém	**Nome**	**Idade**	**Estado**
	Mario	25	Pará
Curitiba	**Nome**	**Idade**	**Estado**
	Francisco	18	Paraná

Figura 5. NoSQL família de colunas.

Os NoSQL orientados a colunas fornecem apenas consultas de intervalo e algumas operações como "*in*", "e / ou" e expressão regular, se forem aplicadas em chaves de linha ou valores indexados. Mesmo que cada NoSQL da família de colunas ofereça um SQL como linguagem de consulta para fornecer uma interação mais conveniente do usuário, somente as chaves de linha e os valores indexados podem ser considerados nessas consultas. Porém ainda não existe uma linguagem de consulta comum para os NoSQL da família de colunas disponíveis [9].

Cassandra é um BD *open source* completamente orientado a colunas desenvolvido por Apache. Segue-se uma arquitetura em anel onde todos os nós são independentes iguais. Garante tolerância e disponibilidade de partições, que não oferecem nenhum ponto de falha. Cassandra fornece balanceamento automático de carga e armazena dados em uma forma não normalizada [12].

- Ele oferece escalabilidade linear não importa quão grande seja a carga de trabalho o seu desempenho de transferência permanece inalterado enquanto processa um conjunto extremamente grande de dados. (O desempenho de outros BDs NoSQL como o HBase e o MongoDB podem se degradar ao processar quantidades tão grandes de dados);
- Cassandra usa protocolo para comunicar uma mensagem de atualização para todas as réplicas simultaneamente;
- Ler, escrever, atualizar é extremamente simples em Cassandra. Ele oferece uma boa experiência de usuário e é quase perfeitamente tolerante a falhas;

Cassandra deve ser selecionado para o seguinte:

- Um aplicativo em que o número de leituras é maior do que o número de gravações. Por exemplo. Organizações como *Airship, Twitter* usam Cassandra;
- Aplicações onde a consistência imediata não é uma grande preocupação. (No caso de hotéis on-line e usuários de reservas de voos estão interessados em ver apenas o menor preço, o que pode alterar ligeiramente no momento da reserva);

- Aplicações *Web* que têm de fornecer esquema dinâmico e conteúdo aos utilizadores, exemplo *Netflix*;
- Aplicações onde é necessária alta manutenção de código;

Cassandra não deve ser selecionado para o seguinte:

- Para operações transacionais e relacionais onde é necessária alta consistência;
- Para consultas dinâmicas envolvendo operações *JOIN* e *Aggregat*e;

Orientado a documentos

O BD de documentos é uma alternativa aos BDs relacionais e é usado para armazenar dados semiestruturados existentes na forma de XML, JSON (*Java Script Object Notation*) ou em outros formatos semelhantes. Um documento pode ser comparado a uma linha de um BD relacional que contém todas as informações relacionadas a documentos. Uma coleção compreende vários documentos onde cada documento pode ter esquemas diferentes e diferem no número e tipo de dados que estão sendo armazenados, como ilustrado na Figura 6. É especialmente otimizado para armazenar informações textuais. Embora o BD tenha um *design* livre de esquema, os registros armazenados são semiestruturados e existem na forma de hierarquias [12].

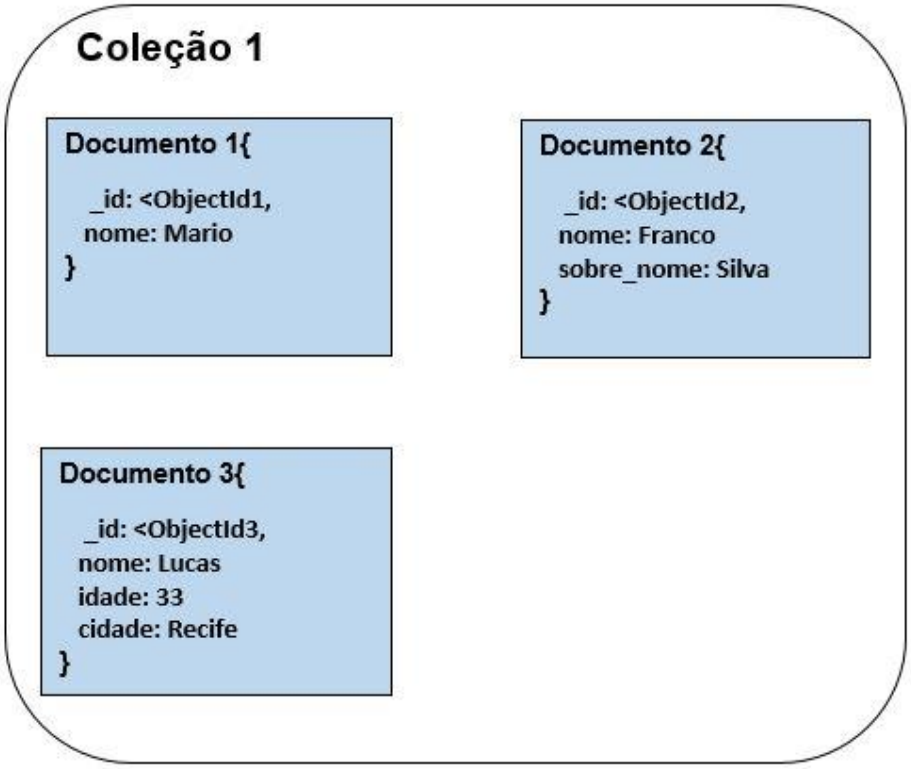

Figura 6. NoSQL orientado a documentos.

Os NoSQL orientado à documentos oferecem APIs muito mais ricas. As consultas de intervalo em valores, índices secundários, consulta de documentos aninhados e operações como "e", "ou", "entre" são recursos, que podem ser usados convenientemente. As consultas dos NoSQL Riak e MongoDB podem ser estendidas com expressão regular. Enquanto MongoDB suporta operações adicionais como contagem e distinto, Riak oferece funcionalidades para atravessar ligações entre documentos facilmente. As interfaces REST (*Representational State Transfer*) também são suportadas por NoSQL baseados em documentos [9].

MongoDB é um BD de armazenamento de documentos que armazena dados semiestruturados escritos em formato BSON (*Binary* JSON). Oferece consistência forte após uma escrita com sucesso à memória, a operação de leitura entregará resultados atualizados. Também é altamente escalável. No entanto, pode oferecer disponibilidade limitada de dados, uma vez que ele tem apenas o mestre principal em um cluster. Assim, no caso de uma falha de rede os dados podem não estar disponíveis. Ele pode trocar entre disponibilidade e consistência por ter mais de um mestre ativo em um cluster. Este BD foi desenvolvido pela 10gen e oferece esquemas dinâmicos, suporta o mapeamento de documentos usando abordagens tanto

embutidas quanto referenciadas. 10gen oferece tanto a versão gratuita e edição empresarial sob licença comercial. Algumas características importantes do MongoDB incluem [12]:

- Ele fornece indexação de cada atributo em um documento e, portanto, oferece alto desempenho;
- Ele usa o sistema de arquivos GridFS para *sharding* automático e armazenar arquivos maiores, especialmente os dados multimídia não estruturados, por exemplo vídeos. Oferece suporte especial para consultas de localização e de intervalo baseadas em latitude, longitude;
- Ele oferece desempenho de alta taxa de transferência;

MongoDB é mais adequado para o seguinte:

- Como alternativa aos aplicativos da web que usam RDBMS;
- Para fornecer escalabilidade e operações de armazenamento em cache;
- Para o gerenciamento de conteúdo de dados semiestruturados;

MongoDB não é adequado para o seguinte:

- Aplicações que requerem excessivas operações *JOIN* e referências de chaves estrangeiras;
- Aplicações que exigem alta conformidade com as propriedades do ACID;

Baseado em Grafos

Os BDs baseados em grafos são mais adequados para percorrer e pesquisar aplicativos, como encontrar links relacionados no *LinkedIn*, procurar amigos no *Facebook*, etc. Isso dá mais importância à relação entre itens de dados e não dados. Eles são altamente otimizados para percorrer rápido e fazer uso eficiente dos algoritmos de grafos, como o caminho mais curto primeiro, a fim de encontrar a ligação entre as informações [12]. Eles se assemelham a estruturas semelhantes a grafos encontradas na vida cotidiana, como redes humanas e redes rodoviárias.

Um BD baseado em grafos pode ser definido como uma estrutura, que consiste de nós, arestas e propriedades para representar e armazenar dados, como ilustrado na Figura 7. Os nós representam registros na analogia RDBMS. Bordas ou relações

são derivadas de alguma coluna predefinida em um nó. Isso permite que os BDs baseados em grafos gerenciem complexos relacionamentos muitos-para-muitos que são difíceis de serem absorvidos na maioria dos outros paradigmas de BD. Com essa força inerente, os BDs baseados em grafos são super-rápidos ao lidar com dados associativos. Como estes BDs não precisam de operações caras como as junções no SGBDR, eles são altamente escaláveis e os dados ad-hoc podem ser facilmente gerenciados com um BD baseado em grafos [10].

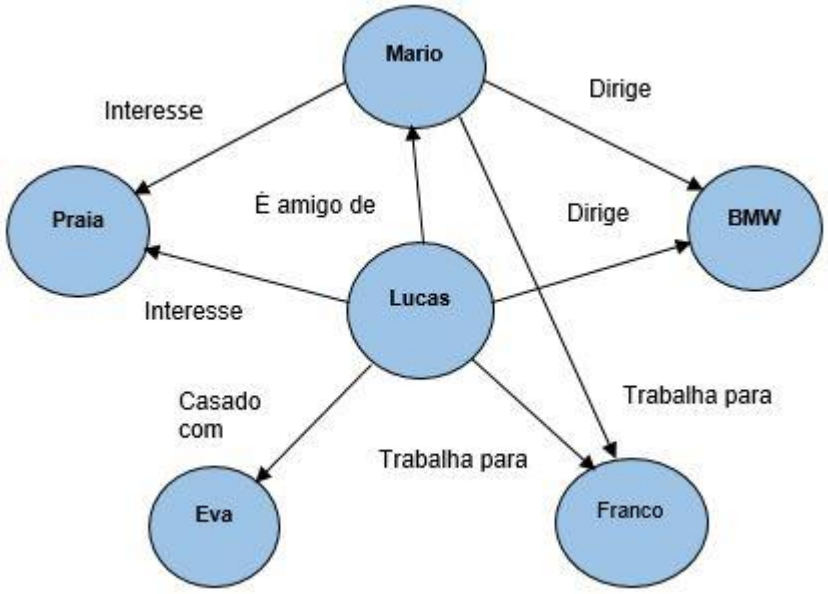

Figura 7. NoSQL baseado em grafos.

Os BDs baseados em grafos podem ser consultados de duas maneiras diferentes. As estratégias de correspondência de grafos tentam encontrar partes do grafo original, que correspondem a um padrão de grafo definido. A travessia do grafo, por outro lado, começa a partir de um nó escolhido e percorre o grafo de acordo com uma descrição. As estratégias transversais diferem na detecção de um nó correspondente o mais rápido possível (largura primeiro) e em encontrar o caminho mais curto (profundidade primeiro). A maioria dos BDs baseado em grafos oferece APIs REST, interfaces específicas de linguagem de programa e armazena linguagens de consulta específicas para acessar dados usando uma das duas estratégias descritas. Em contraste com outros BDs NoSQL, existem algumas linguagens de consulta, que são suportadas por mais de um BD baseado em grafos. SPARQL é uma linguagem de consulta popular, declarativa com uma sintaxe muito simples fornecendo correspondência de padrão de grafo. É suportado pela maioria das dos NoSQL baseado em grafos e por Ne04j. Gremlin é uma linguagem de programação

imperativa usada para executar travessias gráficas baseadas em XPATH (XML *Path Language)* [9].

O **Neo4j** é um BD baseado em grafos altamente escalável, construído especificamente para alavancar não apenas dados, mas também seus relacionamentos.

O mecanismo de processamento e armazenamento de grafos nativos do Neo4j oferece desempenho constante em tempo real, ajudando as empresas a criar aplicativos inteligentes para atender aos atuais desafios de dados em constante evolução. Algumas características importantes do Neo4j são [18]:

- Processamento baseado em grafos, para otimizar o processamento em tempo real de relacionamentos de dados, em escala;
- Adequado para dados mestre, transações e em todos os lugares onde você precisa de confiabilidade de dados;
- Linguagem de consulta poderosa Cypher exige freqüentemente um código de 10x a 100x menos do que o SQL;
- Escalabilidade e alta disponibilidade, tempos de resposta de consulta consistentes e integridade de dados robustos;
- Importação de dados a partir de BDs relacionais e outros: seja milhões de linhas ou bilhões;
- Suporte à integração para linguagens e *frameworks* populares;

3.1 Principais diferenças

A principal diferença é que o NoSQL é uma categoria de mecanismos de BD que não suportam o SQL, a fim de obter recursos de desempenho ou confiabilidade incompatíveis com a flexibilidade do SQL. Esses mecanismos geralmente fornecem uma linguagem de consulta que fornece um subconjunto do que o SQL pode fazer, além de alguns recursos adicionais. *JOIN*, *TRANSACTION*, *LIMIT* e não indexados, normalmente não são suportados pelos bancos NoSQL. Alguns NoSQL têm importação automática de recursos SQL [5].

Os dados SQL são armazenados em tabelas com uma estrutura fixa. Pode-se definir relações entre as tabelas a partir da mesma base de dados. As relações entre as tabelas podem ter relações com chaves estrangeiras. É possível definir a restrição *PRIMARY KEY* nas tabelas. Uma tabela pode ter uma chave primária composta de uma ou mais colunas da tabela. Pode haver outras restrições declaradas nas colunas de uma tabela, como *UNIQUE*, *FOREIGN KEY ou NOT NULL* [5].

Nos BDs NoSQL os dados são armazenados usando coleções. As coleções do MongoDB não têm restrições quanto aos dados armazenados nelas. As coleções não têm uma estrutura fixa. Os campos podem ter diferentes tipos de dados [5].

3.3.1 ACID vs BASE

Bases de dados relacionais são baseadas em um conjunto de princípios para otimizar o desempenho. Princípios usados pelas bases de dados Relacionais ou NoSQL são derivados do teorema CAP. Ele é conhecido também como o *Brewer's Theorem* [11], de acordo com este teorema, as seguintes garantias podem ser definidas:

- **Consistência** (C): O resultado de alterações feitas a qualquer parte do BD deve ser refletido em toda a base de dados;
- **Disponibilidade** (A): Se o BD estiver disponível, os usuários do BD devem ser capazes de acessar os dados;
- **Tolerância à falhas** (P): Em caso de partições arbitrárias resultantes devido a falha de rede, o sistema continua a funcionar sem problemas;

O Teorema CAP afirma que em um ambiente distribuído, um sistema pode garantir no máximo duas das três propriedades a seguir como ilustrado na Figura 8 [11]:

Figura 8. Diagrama representando teorema CAP.

CA: Todos os nós em um *cluster* estão constantemente se comunicando uns com os outros para garantir uma consistência forte após cada operação de leitura. Os dados armazenados estão sempre disponíveis para atender a solicitação do cliente, no entanto, quando o particionamento ou escalonamento horizontal é executado, o cluster fica bloqueado.

CP: Esse sistema oferece consistência de dados após cada transação, ao mesmo tempo que fornece escalabilidade sem interromper o aplicativo. No entanto, alguns dados podem não estar acessíveis para atender o pedido do cliente sempre.

AP: Este sistema garante disponibilidade de dados para atender a solicitação do cliente e alta tolerância à falhas, mas oferece consistência eventual, isto é, após a conclusão de uma transação todas as réplicas dos dados podem não estar no mesmo estado. No entanto, dada nenhuma entrada todas as réplicas acabará por se tornar consistente em todo o cluster [12].

ACID é um princípio baseado no teorema CAP e usado como conjunto de regras para transações de BD relacional. As garantias ACID são:

- **Atomicidade** - uma transação é concluída quando todas as operações são concluídas, caso contrário, a reversão é executada;
- **Consistência** - uma transação não pode recolher o BD, caso contrário, se a operação for ilegal, a reversão é executada;
- **Isolamento** - todas as transações são independentes e não podem afetar-se mutuamente;

- **Durabilidade** - quando o *commit* é executado, as transações não podem ser desfeitas.

É notório que, para ter um BD robusto e correto, essas garantias são importantes. Mas quando a quantidade de dados é grande, ACID pode ser difícil de alcançar. Por isso, NoSQL se concentra no princípio BASE:

- **Basicamente disponível** - todos os dados são distribuídos, mesmo quando há uma falha o sistema continua a funcionar;
- **Estado leve** - não há garantia de consistência, o sistema não tem de ser disponível o tempo todo;
- **Eventualmente consistente** - o sistema garante que, mesmo quando os dados não são consistentes, eventualmente será, o sistema tornasse consistente no momento devido.

3.3.2 Diferenças de sintaxe

Para fazer a comparação de sintaxe entre os dois tipos de BDs foram escolhidos os dois BDs descritos acima, o MongoDB representando o NoSQL e o PostgreSQL representando os BDs relacionais. Esta seção tem como referência o artigo desenvolvido por (Boicea, A, 2012) [5], neste artigo o autor apresenta uma comparação de sintaxe entre o MongoDB (NoSQL) e o Oracle (Relacional). O PostGreSQL foi escolhido para representar os bancos relacionais, porque ele é um banco de dados gratuito que pertence a Oracle.

O MongoDB é relativamente novo no mercado de BD e é usado em muitos projetos e produtos importantes, tais como: *MTV Networks, Craiglist, Disney Interactive Media Group, Sourceforge, Wordnik, Yabblr, SecondMarket, The Guardian, Forbes, The New York Times, bit.ly, GitHub, FourSquare, Disqus, PiCloud* etc. Cada um desta lista usa MongoDB porque eles encontraram algo que se encaixa muito bem com seus projetos. Por exemplo, *the* Craiglist usa o MongoDB para arquivar bilhões de registros, o *MTV* usa o MongoDB como repositório principal para *MTV Networks*, no *SourceForge*, o MongoDB é usado para o armazenamento de *back-end*, o *The Guardian* usa o MongoDB porque descobriu que eles podem armazenar documentos muito simples usando Este tipo de sistema de gerenciamento de BD, o *The New York*

Times usa MongoDB em um formulário de construção de aplicativos para fotos enviadas, *bit.ly* usa MongoDB para armazenar o histórico do usuário [5].

PostgreSQL é um sistema de gerenciamento de BD relacional (SGBDR) baseado em POSTGRES, Versão 4.2, desenvolvido na Universidade da Califórnia no *Berkeley Computer Science Department*. A POSTGRES foi pioneira em muitos conceitos que só se tornaram disponíveis em alguns sistemas de BD comerciais muito mais tarde [21].

Existem grandes diferenças de sintaxe entre os dois sistemas de gerenciamento de BD. O PostgreSQL usa linguagem SQL comum. Ele usa instruções de manipulação de dados como *SELECT, UPDATE, DELETE* [21].

O MongoDB usa funções para as operações de adicionar novos registros, atualizar e excluir registros existentes[5].

Para melhor entender as diferenças entre a sintaxe de dois BDs, vamos dar alguns exemplos. Em primeiro lugar vamos falar sobre a criação de um objeto no qual podemos inserir dados. Este objeto é chamado tabela no PostgreSQL e coleção no MongoDB. No BD PostgreSQL, a criação de uma tabela é feita usando a instrução CREATE [21]:

```
CREATE TABLE usuarios (
    usuario_id INT NOT NULL,
    primeiro_nome VARCHAR2(50),
    sobre_nome VARCHAR2(50)
);
```

No MongoDB, a criação da coleção é feita na primeira inserção. Devido à flexibilidade da coleção MongoDB, a estrutura não precisa ser corrigida[5].

Excluir uma tabela com todas os *constraints* e índices pode ser muito fácil. No PostgreSQL é uma função que irá eliminar uma tabela se não houver outras tabelas relacionadas a esta [21]:

```
DROP TABLE usuarios;
```

No MongoDB não há dependências entre os registros para que você possa facilmente descartar qualquer registro e os índices atribuídos a ele [5]:

db.usuarios.drop();

Em relação à inserção de dados, os BDs PostgreSQL têm vários tipos de restrições nas tabelas. Quando você está tentando inserir um novo registro, você tem que ter cuidado para não violar nenhuma das restrições. Por outro lado, MongoDB é mais flexível com os dados. Você não tem quaisquer restrições quanto aos dados das coleções. Esta é a principal diferença entre um BD SQL e um NoSQL [5][21].

Os dados SQL são armazenados em tabelas com uma estrutura fixa. Pode-se definir relações entre as tabelas a partir da mesma base de dados. As relações entre as tabelas podem ser relações chave estrangeiras. Você pode definir a restrição *PRIMARY KEY* nas tabelas. Uma tabela pode ter uma chave primária composta de uma ou mais colunas da tabela. Pode haver outras restrições declaradas nas colunas de uma tabela, como *UNIQUE, FOREIGN KEY ou NOT NULL*[5].

Nos BDs NoSQL os dados são armazenados usando coleções. As coleções do MongoDB não têm restrições quanto aos dados armazenados nelas. As coleções não têm uma estrutura fixa. Os campos podem ter diferentes tipos de dados [5].

No PostgreSQL um novo registro é salvo no BD usando a instrução INSERT. Por exemplo, se você deseja inserir um novo registro na tabela "usuários", a instrução é desta forma [21]:

INSERT INTO usuario(id_usuario, primeiro_nome, sobre_nome)

VALUES(1, 'Fernando', 'Monteiro');

No MongoDB a inserção de um novo registro é feita usando funções e objetos BSON. As funções utilizadas para inserir novos dados são *'insert'* e *'save'*. Ao inserir um novo objeto em uma coleção MongoDB, um novo campo é adicionado automaticamente ao seu novo objeto. Esse campo é chamado '_id' e é único e usado como um índice. Por exemplo, se você quiser adicionar o mesmo usuário na coleção 'usuário' do BD MongoDB, você terá que escrever isto [5]:

db.usuario.insert({ 'usuario_id': 1, 'primeiro_nome': 'Fernando', 'sobre_nome': 'Monteiro' });

Ou

db.usuario.save({ ‘usuario_id’: 1, ‘primeiro_nome’: ‘Fernando’, ‘sobre_nome’: ‘Monteiro’ });

A recuperação de dados básicos no BD PostgreSQL é feita usando a instrução SELECT simples. É preciso especificar os campos que se deseja selecionar e as tabelas a partir do qual ele quer recuperar dados. Aqui está um exemplo de uma instrução SELECT simples de uma única tabela [21]:

SELECT usuario_id, primeiro_nome, sobre_nome FROM usuarios;

A saída para esta instrução é uma tabela contendo todos os usuários da tabela de usuários. Se você quiser refinar a pesquisa, você pode incluir uma instrução WHERE [21].

Por exemplo, se você quiser recuperar apenas os usuários cujo sobrenome é Silva, você deve incluir uma cláusula WHERE como a seguinte [21]:

WHERE sobre_nome LIKE 'Silva';

Você pode ordenar os resultados usando a cláusula ORDER BY. Os resultados podem ser ordenados ascendente ou descendente [21].

No MongoDB, você pode recuperar resultados usando a função *find*. No MongoDB, não existem tabelas. Os dados são armazenados usando coleções de dados [5].

Estas coleções não têm uma estrutura fixa. Os itens armazenados nesta coleção podem ter campos diferentes. Por exemplo, supondo que temos uma coleção de usuários nomeados "usuários". Se você quiser recuperar todos os itens desta coleção você deve usar o *find* desta maneira [5]:

db.usuarios.find();

Se você quiser recuperar apenas alguns dos itens da coleção, por exemplo, apenas os usuários que têm o nome "Fernando" você usa a função *find* com um objeto extra [5]:

db.usuarios.find({ primeiro_nome: “Fernando” });

Se você não quiser que sua consulta exiba todos os campos, você pode adicionar um objeto para especificar quais campos serão exibidos. Para o campo "_id", que MongoDB adiciona automaticamente a novos objetos inseridos nas coleções, você tem que especificar não exibi-lo. Para os outros campos que você deseja consultar precisa especificar para exibi-los. Por exemplo, se você quiser exibir apenas o "sobre_nome" e "primeiro_nome" e não o "usuario_id" e o "_id" [5]:

db.usuarios.find({}, { “_id”: 0, “sobre_nome”: 1, “primeiro_nome”: 1 });

No MongoDB você pode classificar os dados. Você pode usar a função de classificação onde você especifica o campo que você deseja classificar os dados e a ordem de classificação: ascendente (1) ou descendente (-1):

db.usuarios.find().sort({ “sobre_nome”: 1 });

A remoção de dados geralmente é feita usando alguns critérios ou não. No BD PostgreSQL, os dados removidos de uma tabela são feitos usando a instrução DELETE:

DELETE FROM usuarios;

Se você quiser excluir, por exemplo, apenas as linhas que têm o 'id do usuário' maior que 10, a instrução deve ser assim [21]:

DELETE FROM usuarios WHERE usuario_id > 10;

Nas coleções MongoDB, a função usada para excluir o conteúdo é 'remove' [5]:

db.usuarios.remove();

Se você quiser remover apenas algumas entradas, como as que têm o 'ID do usuário' maior que 10, então você deve escrever isto [5]:

db.usuarios.remove({ ‘user_id’: { ‘$gt’: 10 } });

Você pode facilmente atualizar uma entrada ou parte de uma entrada em BDs. No PostgreSQL se você quiser atualizar uma tabela inteira, use a instrução [21]:

UPDATE usuarios SET sobre_nome = ‘sobre_nome’;

Se você deseja atualizar apenas os usuários com o 'usuario_id' maior que 100, a instrução é esta [21]:

UPDATE usuarios

SET sobre_nome = ‘sobre_nome’

WHERE usuario_id > 100;

No MongoDB você pode facilmente atualizar um registro ou muitos registros. O comando se você deseja atualizar a coleção inteira é [5]:

db.usuarios.update({}, {‘$set’: {‘sobre_nome’: ‘sobre_nome’}});

O primeiro objeto é usado para selecionar as entradas que você deseja atualizar. Nesse caso, a consulta retorna todas as entradas da coleção. Se você deseja atualizar apenas as entradas que têm o 'usuario_id' maior que 100, a função de atualização tem esta aparência [5]:

db.usuarios.update({‘usuario_id’: {‘$gt’: 100}},

{‘$set’: {‘sobre_nome’: ‘sobre_nome’}});

Com a função de atualização você pode substituir um objeto da coleção, você pode puxar um item de uma *array* você pode empurrar um item para uma *array*, você pode definir um novo campo, você pode atualizar um conjunto de campos, você pode fazer muitas coisas [5].

Embora existam diferenças de sintaxe, trabalhar com ambos os BDs é leve e intuitivo. Ambos os BDs têm suas vantagens e desvantagens, mas no que diz respeito à sintaxe, ambos são fáceis de usar e fáceis de aprende.

Capítulo 4

Estudo de caso

Neste capítulo será apresentado um estudo de caso envolvendo os dos paradigmas de BDs, bem como uma breve introdução sobre os dois BDs escolhidos.

Este estudo de caso foi baseado no artigo desenvolvido por (Boicea A., 2012) [5], onde o autor apresenta comparações de desempenho entre os bancos MongoDB e Oracle, no estudo o autor conclui que na atualização e remoção dos dados, os tempos do MongoDB são bem menores do que os tempos do Oracle, e ainda conclui que estes tempos sem mantêm praticamente constantes tanto para poucos registros quanto para muitos. No estudo desenvolvido por Jung, Min-Gyue *et al.* (2015) [6], foi feita uma comparação de desempenho entre o MongoDB e o PostgreSQL, onde ele conclui que as velocidades das operações de inserir, selecionar, atualizar e remover, no MongoDB foram mais rápidas.

No estudo desenvolvido por Carniel *et al.* (2012), foram feitas comparações entre BDs relacionais e NoSQL, no contexto de *Data warehouse*, considerando o Star Schema Benchmark. Os resultados mostraram que o modelo orientado por coluna do software FastBit apresentou melhor desempenho, com ganhos de 25,4% a 99,4% se comparado com outros modelos NoSQL e o modelo relacional, no processamento de consultas em Data warehouse [17].

Para realização do estudo de caso foram necessárias as seguintes ferramentas:

- PostgreSQL versão 9.5;
- MongoDB versão 3.3.0;
- RoboMongo versão 0.9.0;
- PgAdmin 3;
- BrModelo versão 2.0;

4.1 PostgreSQL

O sistema de gerenciamento de BD objeto-relacional agora conhecido como PostgreSQL é derivado do projeto POSTGRES escrito na Universidade da Califórnia em Berkeley em 1986. Com mais de duas décadas de desenvolvimento, o PostgreSQL é agora o BD *open source* mais avançado disponível. Ele suporta uma grande parte do padrão SQL e oferece muitos recursos modernos [21]:

- Consultas complexas
- Chaves estrangeiras
- Triggers
- Visualizações atualizáveis
- Integridade transacional
- Controle de concorrência

Além disso, o PostgreSQL pode ser estendido pelo usuário de várias maneiras, por exemplo, adicionando [21]:

- Tipos de dados
- Funções
- Operadores
- Funções agregadas
- Métodos de índice
- Linguagens processuais

E por causa da licença *free*, o PostgreSQL pode ser usado, modificado e distribuído por qualquer pessoa gratuitamente para qualquer finalidade, seja ela privada, comercial ou acadêmica [21].

O PostgreSQL pode ser baixado no site https://www.postgresql.org/download/ , e atualmente está disponível na versão 9.5 para os sistemas operacionais *Windows Linux* e *Mac OS* entre outros. A instalação é fácil, basta executar o instalador e seguir as instruções [21].

Este instalador inclui o servidor PostgreSQL, pgAdmin que é uma ferramenta gráfica para gerenciar e desenvolver seus BDs como ilustrado na Figura 9, e StackBuilder, um gerenciador de pacotes que pode ser usado para baixar e instalar

ferramentas e drivers adicionais do PostgreSQL. Stackbuilder incluem gerenciamento, integração, migração, replicação, geoespacial, conectores e outras ferramentas [21].

Os usuários avançados também podem baixar um arquivo zip dos binários, sem o instalador. Este download destina-se aos usuários que desejam incluir o PostgreSQL como parte de outro instalador de aplicativos [21].

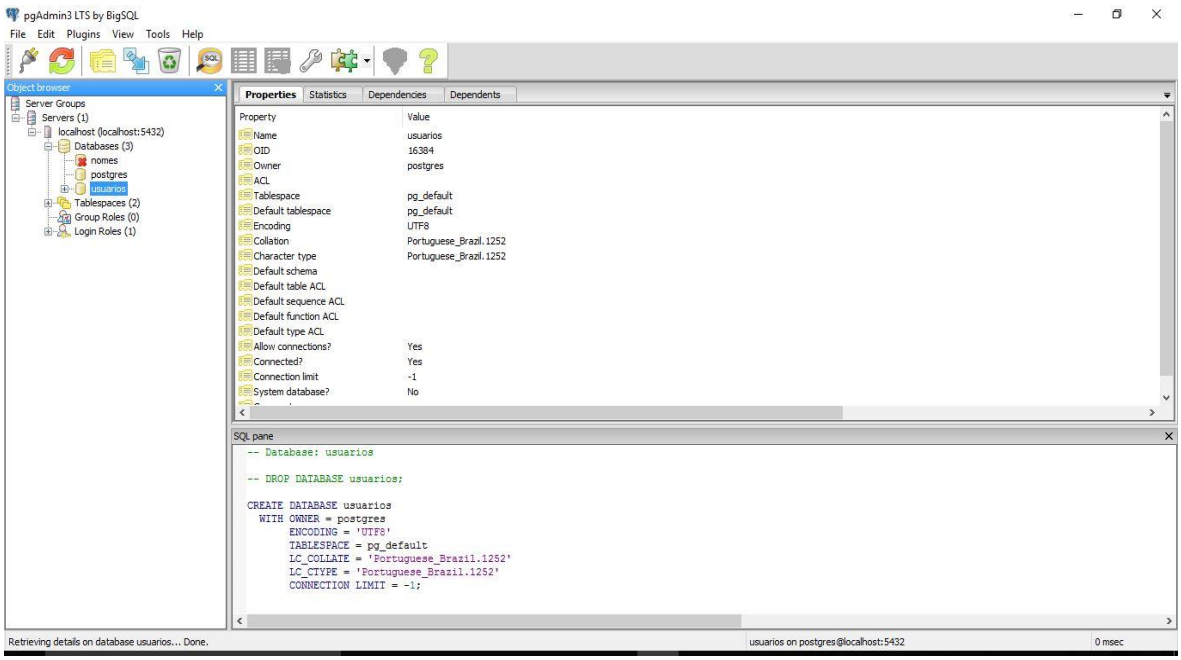

Figura 9. PgAdmin 3.

4.2 MongoDB

O MongoDB é um BD de documentos de código aberto que fornece alta performance, alta disponibilidade e escalonamento automático.

Um registro no MongoDB é um documento, que é uma estrutura de dados composta de pares de campo e valor. Os documentos MongoDB são semelhantes aos objetos JSON, como apresentado na Figura 10. Os valores dos campos podem incluir outros documentos, *arrays* e *arrays* de documentos.

Figura 10. Documentos do MongoDB.

As vantagens do uso de documentos são:

- Os documentos (isto é, objetos) correspondem a tipos de dados nativos em muitas linguagens de programação.
- Documentos incorporados e matrizes reduzem a necessidade de junções caras.
- Esquema dinâmico suporta polimorfismo fluente.

4.2.1. Características chaves do MongoDB

Alta performance

O MongoDB fornece persistência de dados de alto desempenho. Em particular,

- Suporte para modelos de dados incorporados reduz a atividade de Entrada e Saída no sistema de BD.
- Os índices suportam consultas mais rápidas e podem incluir chaves de documentos incorporados e matrizes.

Rica linguagem de consulta

MongoDB suporta uma linguagem de consulta rica para suportar operações de leitura e gravação, bem como:

- Agregação de Dados;
- Pesquisa de Texto e Consultas Geoespaciais;

Alta disponibilidade

O recurso de replicação MongoDB, chamado conjunto de réplicas, fornece:

- *Failover* automático (mudança de servidor após uma falha) ;
- Redundância de dados;

Um conjunto de réplicas é um grupo de servidores MongoDB que mantêm o mesmo conjunto de dados, fornecendo redundância e aumentando a disponibilidade de dados.

Escalabilidade horizontal

O MongoDB fornece escalabilidade horizontal como parte de sua funcionalidade principal, através do método *Sharding*:

- Distribui dados através de um cluster de máquinas;
- *Sharding* é um método para distribuir dados em várias máquinas. O MongoDB usa este método para suportar implementações com conjuntos de dados muito grandes e operações de alto rendimento;

Suporte para Múltiplos Armazenadores

O MongoDB suporta múltiplos mecanismos de armazenamento, tais como:

- WiredTiger;
- MMAPv1;

Além disso, o MongoDB fornece API de mecanismo de armazenamento plugável que permite que terceiros desenvolvam mecanismos de armazenamento para o MongoDB.

MongoDB armazena registros de dados como documentos BSON. BSON é uma representação binária de documentos JSON, usado para armazenar documentos e fazer chamadas de procedimento remoto no MongoDB.

BSON suporta os seguintes tipos de dados como valores em documentos conforma a Tabela 7. Cada tipo de dados tem um número correspondente e um alias de cadeia que pode ser usado com o operador $ type para consultar documentos por tipo BSON.

Tabela 7. Tipos de dados suportados por BSON.

Tipo	Número	Alias	Notas
Double	1	“double”	

String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Obsoleto.
ObjectId	7	“objectId”	
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	
DBPointer	12	“dbPointer”	Obsoleto.
JavaScript	13	“javascript”	
Symbol	14	“symbol”	Obsoleto.
JavaScript(with scope)	15	“javascriptWithScope”	
32-bit integer	16	“int”	
Timestamp	17	“timestamp”	
64-bit integer	18	“long”	
Min key	-1	“minKey”	
Max key	127	“maxKey”	

O MongoDB pode ser baixado no site https://www.mongodb.com/download-center?jmp=hero#community, e atualmente está disponível na versão 3.2.11 para os sistemas operacionais Windows, Linux, Mac OS e Solares. A instalação é fácil, basta fazer o *download* e seguir as instruções mostradas no site [22].

Para realizar as inserções e consultas no MongoDB foi utilizado o sistema RoboMongo ilustrado na Figura 11. O RoboMongo pode ser baixado através do site https://robomongo.org/download.

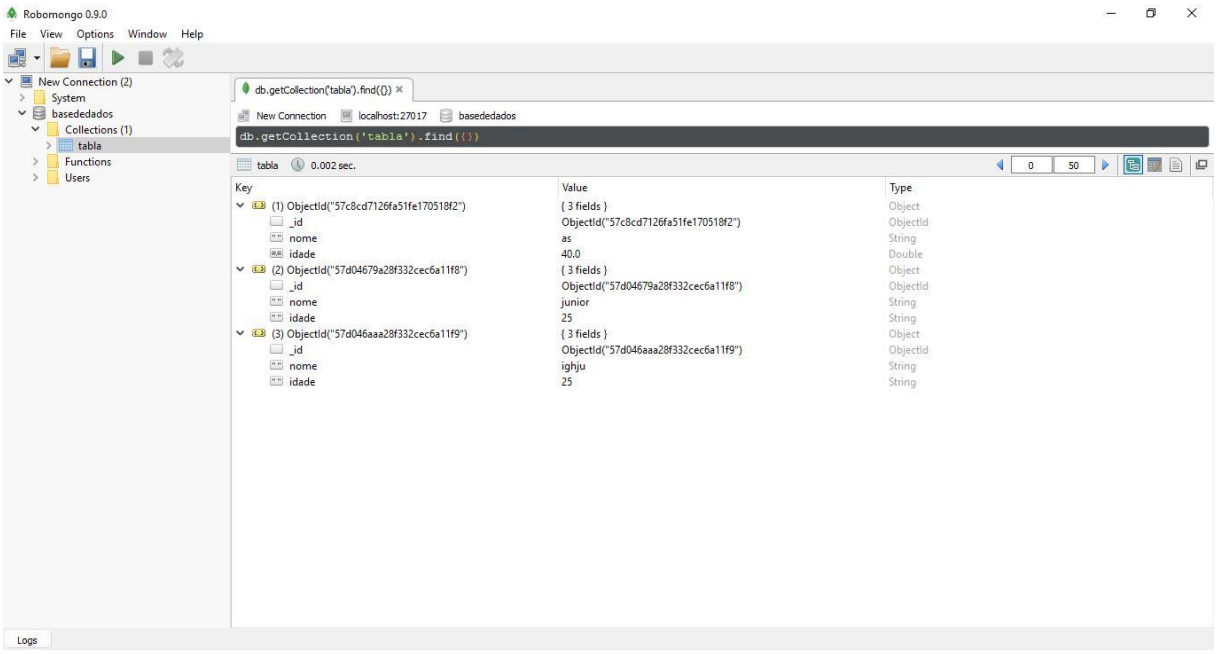

Figura 11. RoboMongo.

4.3 Experimentos

Para comparar o desempenho do PostgreSQL e do MongoDB, foram necessárias realizar as operações de inserir, selecionar, atualizar e excluir. As operações Inserir, Selecionar, Atualizar e Excluir foram executadas para cada um dos 10, 100, 1000, 10000 casos de dados com base nas informações dos funcionários de uma empresa.

Para a configuração inicial, foi intalado o MongoDB e o PostgreSQL em uma máquina com um processador Intel i3 1.50 GHz com 4GB de RAM e sistema operacional *Windows* 10. Ambos os BDs foram instalados e armazenados seus dados no disco C do computador.

O modelo do BD que foi utilizado foi feito através da ferramenta BrModelo [23], a Figura 12, apresenta a estrutura do modelo de BDs relacional utilizado neste estudo de caso, uma tabela de funcionários que contém matrícula, o primeiro nome, sobre nome, telefone, email, setor onde funcionário atua, e data de vínculo com a empresa.

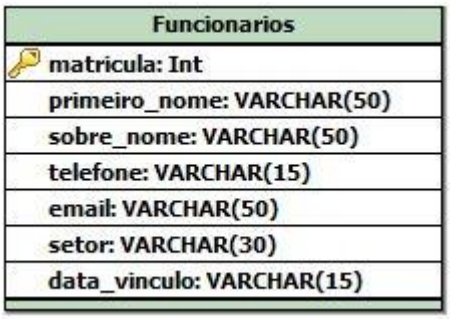

Figura 12. Modelo da tabela Funcionários.

Também foi desenvolvido um modelo do documento que será utilizado no MongoDB, a Figura 13 mostra a estrutura de um registro que foi utilizado neste estudo de caso.

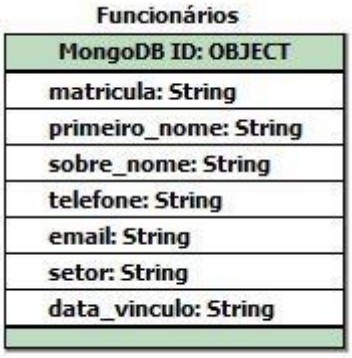

Figura 13. Modelo de um registro de Funcionários.

No PostgreSQL a tabela foi criada da seguinte forma utilizando a linguagem SQL:

```
CREATE TABLE usuarios(
matricula INT NOT NULL,
primeiro_nome VARCHAR2(50),
sobre_nome VARCHAR2(50) ,
telefone VARCHAR(15),
email VARCHAR(50),
setor VARCHAR(30),
data_vinculo VARCHAR(15))
```

No MongoDB, a criação da coleção é feita na primeira inserção. Devido à flexibilidade da coleção MongoDB, portanto a criação da coleção foi feita na operação de *insert*.

4.3.1 Inserção dos dados

O MongoDB apresentou uma limitação na inserção quando inserido 10000 registros com todos os campos propostos, o sistema retornou uma mensagem avisando que operação havia ultrapassado o limite de 64MB por inserção, como mostrado na Figura 14, por isso houve a necessidade de diminuir os atributos, neste caso só foram considerados dois atributos, a matricula e o primeiro_nome.

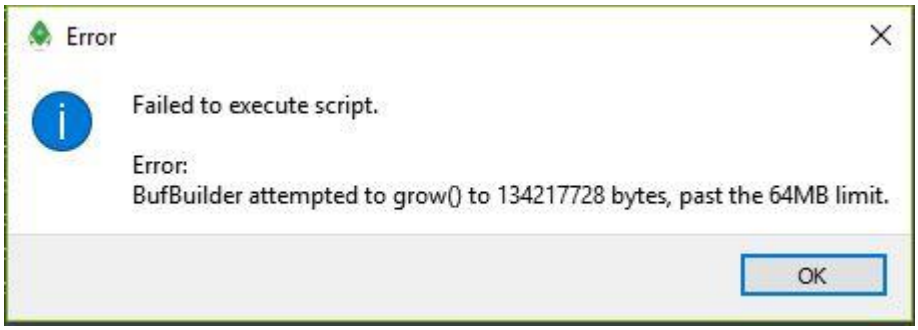

Figura 14. Erro ao inserir 10000 registros com todos os campos.

No PostgreSQL a inserção foi feita como o modelo abaixo:

```
INSERT INTO usuarios( matricula, primeiro_nome)
VALUES( 1, 'primeiro_nome');
```

No MongoDB a inserção foi feita como o modelo abaixo:

```
db.usuarios.insert( { 'matricula': 1, 'primeiro_nome': 'primeiro_nome' );
```

Foram realizados 4 casos de inserção de dados, com 10, 100, 1000 e 10000 linhas por registro, como apresentado na Tabela 8, em todos os casos o tempo de inserção no MongoDB foram menores.

Tabela 8. Tempo de inserção em milissegundos.

Quantidade de inserções	PostgreSQL	MongoDB
10	48 mseg	**4 mseg**
100	**31 mseg**	**9 mseg**
1000	**131 mseg**	**65 mseg**
10000	**770 mseg**	**650 mseg**

O Gráfico 1 apresenta todos os casos de inserção, para ambos os BDs, embora que no caso das 10000 inserções o tempo dos dois ficarem próximos, conclui-se que nestes casos, o MongoDB é mais eficiente na inserção de dados.

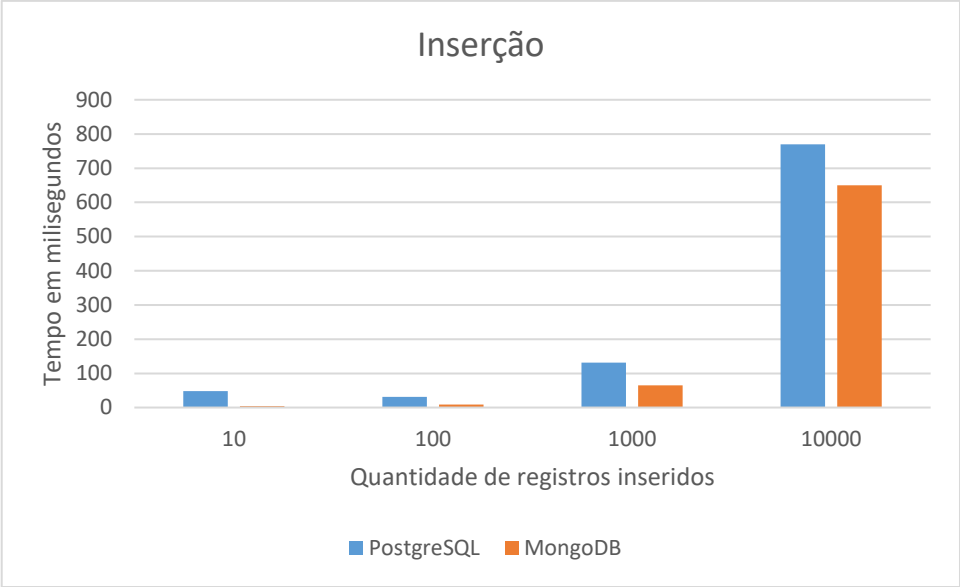

Gráfico 1. Comparação de tempos de inserção.

4.3.2 Seleção dos dados

No PostgreSQL o *select* foi feito descrito abaixo:

Select from usuarios;

No MongoDB o *find* foi feito como o modelo abaixo:

db.usuarios.find();

Foram realizados 4 casos de seleção de dados, com 10, 100, 1000 e 10000 linhas por registro, como apresentado na Tabela 9, em todos os casos o tempo de seleção no MongoDB foram menores.

Tabela 9. Tempo de seleção em milissegundos.

Quantidade de dados	PostgreSQL	MongoDB
10	31 mseg	**14 mseg**
100	**31 mseg**	**13 mseg**
1000	**31 mseg**	**13 mseg**
10000	**100 mseg**	**57 mseg**

O Gráfico 2 apresenta todos os casos de seleções, para ambos os BDs, é importante observar que em todos os casos os tempos de seleção do MongoDB foram bem menores do que os tempos do PostgreSQL, e nos três primeiros casos os tempos do MongoDB se mantiveram próximos a 13 milissegundos. Portanto o MongoDB é mais eficiente em selecionar poucas e grandes quantidades de dados, porém o PostgreSQL lida muito bem com pouca quantidade de dados.

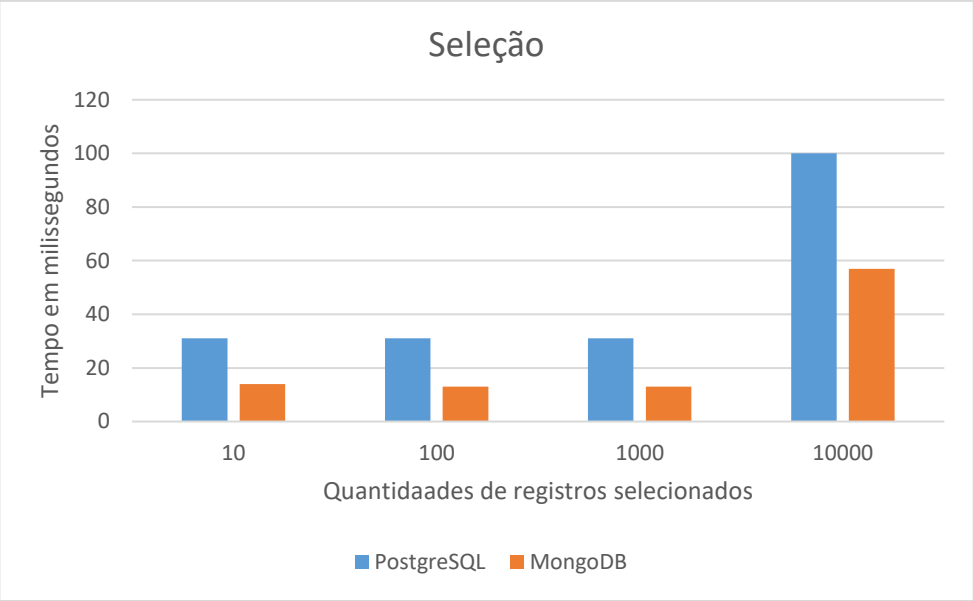

Gráfico 2. Comparação de tempos de seleção.

4.3.3 Atualização dos dados

No PostgreSQL o *update* foi feito como o modelo abaixo:

```
UPDATE usuarios
SET nome = 'nome100'
WHERE matricula > 10
AND matricula <= 110
```

No MongoDB o *update* foi feito como o modelo abaixo:

```
db.usuarios.update(  { matricula: { '$gt': 10, '$lte': 110 } },  { '$set': { nome:
'nome100' } } );
```

Foram realizados 4 casos de atualização de dados, para os registros com matrícula < = 10, matrícula entre 11 e 111, matrícula entre 112 e 1112, e matrícula entre 1113 e 11113, como apresentado na Tabela 10, em todos os casos o tempo de atualização no MongoDB foram menores.

Tabela 10. Tempo de atualização em milissegundos.

Quantidade de atualizações	PostgreSQL	MongoDB
10	38 mseg	**2 mseg**
100	**33 mseg**	**3 mseg**
1000	**53 mseg**	**3 mseg**
10000	**69 mseg**	**5 mseg**

O Gráfico 3 apresenta todos os casos de atualizações para ambos os BDs, é importante observar que em todos os casos os tempos de atualização do MongoDB se manteve menor do que 6 milissegundos. Em comparação com os tempos do PostgreSQL, os tempos do MongoDB foram muito menores, portanto o MongoDB é mais eficiente na atualização tanto de grandes quanto de pequenas quantidades de dados.

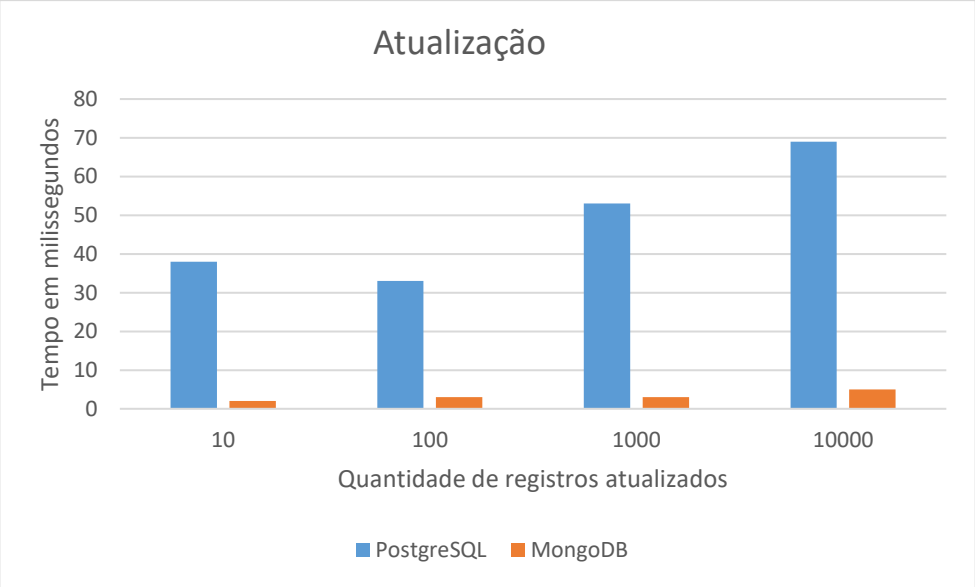

Gráfico 3. Comparação de tempos de atualização.

4.3.4 Remoção dos dados

No PostgreSQL o *delete* foi feito como o modelo abaixo:

```
DELETE FROM usuarios
WHERE matricula > 10
AND matricula <= 110;
```

No MongoDB o *remove* foi feito como o modelo abaixo:

```
db.users.remove(  { matricula: { '$gt': 10, '$lte': 110 } } );
```

Foram realizados 4 casos de remoções de dados, para os registros com matrícula < = 10, matrícula entre 11 e 111, matrícula entre 112 e 1112, e matrícula entre 1113 e 1113, como apresentado na Tabela 11, em todos os casos o tempo de remoção no MongoDB foram menores. Portanto o MongoDB é mais eficiente em remover grandes e pequenas quantidades de dados.

Tabela 11. Tempo de remoção dos dados em milissegundos.

Quantidade de remoções	PostgreSQL	MongoDB
10	54 mseg	**2 mseg**
100	**32 mseg**	**2 mseg**
1000	**32 mseg**	**3 mseg**
10000	**33 mseg**	**2 mseg**

O Gráfico 4 apresenta todos os casos de remoções para ambos os BDs, é importante observar que em todos os casos os tempos de remoção do MongoDB se manteve menor do que 4 milissegundos. Em comparação com os tempos do PostgreSQL, os tempos do MongoDB foram muito menores, portanto o MongoDB é mais eficiente na remoção tanto de grandes quanto de pequenas quantidades de dados.

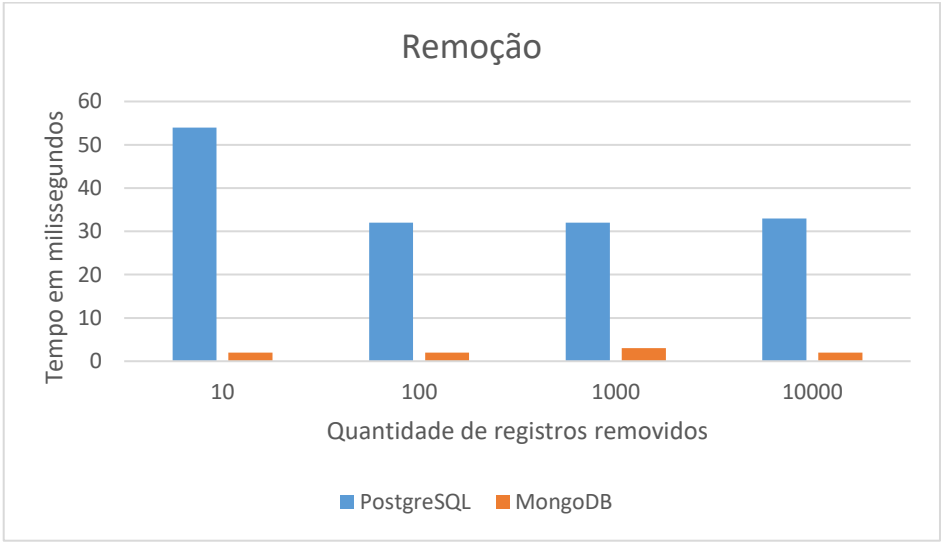

Gráfico 4. Comparação de tempos de remoção.

Capítulo 5

Conclusões e trabalhos futuros

O uso do protocolo da revisão sistemática em si é uma contribuição importante deste trabalho e, o tema estudado tem relevância para a comunidade acadêmica e para a indústria de software. Com a conclusão de todas as etapas da revisão sistemática, com buscas de artigos em livrarias digitais IEEE e ACM foi possível ter uma base teórica rica em conteúdo, para construção deste estudo.

Com a construção do modelo de dados foi possível perceber que os BDs relacionais precisam ser estruturados antes de serem criadas as tabelas, tendo que ser definidos os tipos e tamanhos dos atributos. Diferentes dos modelos relacionais os NoSQL orientados a documentos não precisam ser totalmente estruturados antes da criação das coleções, foi percebido que os tamanhos dos atributos não precisam ser definidos. Outro aspecto notado foi que as tabelas dos BDs relacionais precisam ser criadas antes das inserções dos dados, já no MongoDB a coleção é criada na primeira inserção de um documento.

Em relação a sintaxe dos dois BDs utilizados, o PostgreSQL utiliza a linguagem SQL e o MongoDB utiliza BJSON, embora existam diferenças entre as duas, trabalhar com ambos os BDs é leve e intuitivo. Ambos os BDs têm suas vantagens e desvantagens, mas no que diz respeito à sintaxe, ambos são fáceis de usar e fáceis de aprender. Vale ressaltar que o MongoDB tem uma estrutura flexível, onde uma coleção pode ter um número diferente de atributos, pode ter diferentes tipos de atributos em cada registro e não é preciso criar uma coleção antes de inserir os dados, pois a criação da coleção é feita na primeira inserção. Já o PostgreSQL como todos os outros SGBDR apresenta limitações de estrutura, onde é preciso criar as tabelas antes de serem inseridos os dados, as linhas devem ter a mesma quantidade de atributos, e os tipos dos atributos são previamente definidos.

Com a condução dos testes do caso de uso proposto notou-se que apesar de os NoSQL serem desenvolvidos para armazenar grandes quantidades de dados, houve um erro ao tentar inserir um documento com 10000 registros com 7 campos, pois ultrapassou os 64MB por inserção, entretanto isto não ocorreu no PostgreSQL,

mas para uma comparação mais justa foi limitado 10000 como sendo o número máximo de inserções, e os campos foram diminuídos para dois campos.

Após as execuções dos testes, foi possível observar que o MongoDB se saiu melhor em todas os testes de inserção, seleção, atualização e remoção de dados, também foi possível observar que na inserção de 10000 registros o tempo gasto nos dois bancos foram bem parecidos, é importante ressaltar que o PostgreSQL se sai bem nos casos de manipulação de dados menores.

As bases de dados relacionais são baseadas em um conjunto de princípios para otimizar o desempenho, no entanto quando há uma quantidade muito grande de dados fica difícil seguir esses princípios, e isso leva a uma perda de desempenho, devido as muitas junções de tabelas que são necessárias para realizar as operações, já o NoSQL não tem essas operações de junções isso faz com que ganhe mais velocidade nas operações.

NoSQL não é um substituto para SGBDR. Podemos destacar, entretanto, que o NoSQL é benéfico para sistemas de tecnologia da *web* como o *Facebook* e o *Google* que guardam muitas informações de seus usuários o tempo todo. Enquanto isso, SGBDR é mais adequado para empresas, especificamente para empresas que exigem relatórios

Portanto o NoSQL é melhor para ser usado quando é preciso ter uma maior velocidade nas transações, e também quando há muitos dados não estruturados para serem armazenados, e não é bom utilizá-lo quando o sistema exige que ajam junções de registros, já os BDs relacionais são melhores utilizados quando se há necessidade de realizar junções de tabelas e para sistemas que exigem Atomicidade, Consistência, Isolabilidade e Durabilidade dos dados.

Como trabalhos futuros pretende-se criar uma aplicação que acesse um BD NoSQL para manipular dados de uma empresa, nota-se que são muitos os documentos utilizados por profissionais de recursos humanos, então esta aplicação tem como finalidade armazenar os dados e documentos dos funcionários de uma empresa, propondo aos profissionais de recursos humanos uma ferramenta para ajudar nos seus trabalhos.

Referências

[1] Duarte, Denio, ***SQLtoKeyNoSQL: A Layer for Relational to Key-based NoSQL Database Mapping.*** *In: International Conference on Information Integration and Web-based Applications & Services*, 2015, Brussels, Belgium.

[2] Tamane, Sharvari, ***Non-Relational Databases in Big Data.*** *In: Italian Conference on Theoretical Computer Science*, 2016, Udaipur, India.

[3] Cattell, Rick, ***Scalable SQL and NoSQL Data Stores.*** ACM SIGMOD *Record*, Vol. 39, No. 4 ,Dezembro de 2010.

[4] Abramova, Veronika; Bernardino,Jorge, ***NoSQL Databases: MongoDB vs Cassandra.*** *In: Conference* C3S2E, 10-12 de Julho, 2013, Porto, Portugal.

[5] Boicea, A., Radulescu, F., Ioana Agapin, L., ***MongoDB vs Oracle – Database Comparison.*** *In: Third International Conference on Emerging Intelligent Data and Web Technologies*, 2012.

[6] Jung, Min-Gyue; Youn, Seon-A; Bae, Jayon; Choi, Yong-Lak, ***A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment.*** *In: 8th International Conference on Database Theory and Application*, 2015.

[7] Gyorödi, C., Gyorödi, R., Pecherle, G., Olah, A.*,* ***A Comparative Study: MongoDB vs. MySQL.*** *In: 13th International Conference on Engineering of Modern Electric Systems (EMES)*, 2015.

[8] Li, Xiang; Ma**,** Zhiyi; Chen Hongjie, ***QODM: A Query-Oriented Data Modeling Approach for NoSQL Databases.*** *In: IEEE Workshop on Advanced Research and Technology in Industry Applications* (WARTIA), 2014.

[9] Hecht, Robin; Jablonski, Stefan, ***NoSQL Evaluation A Use Case Oriented Survey.*** *In: International Conference on Cloud and Service Computing*, 2011.

[10] Jayathilake**,** D., Sooriaarachchi, C., Gunawardena, T., Kulasuriya, B., Dayaratne, T.*,* ***A Study into the Capabilities of NoSQL Databases in Handling a Highly Heterogeneous Tree.*** *In: 99X Technology Colombo*, Sri lanka, 2012.

[11] Ramesh**,** D., Sinha, A., Singh, S., ***Data Modelling for Discrete Time Series Data Using Cassandra and MongoDB.*** *In: 3rd Int'l Conf. on Recent Advances in Information Technolog*, 2016.

[12] Prakash Srivastava, P., Goyal, S., Kumar, A., ***Analysis of Various NoSql Database.*** *In: Dept. of Computer Science and Engineering Mody University of Science and Technology Rajasthan*, India, 2015.

[13] Parker, Z., Poe, S., V. Vrbsky, S., ***Comparing NoSQL MongoDB to an SQL DB.*** *In: ACMSE'13, April* 4-6, *Savannah, GA, USA,* 2013.

[14] Gunter, D., Govindaraju, M., Ramakrishnan, L., Shane Canon, R., Dede, E., ***Performance Evaluation of a MongoDB and Hadoop Platform for Scientific Data Analysis.*** *In: ScienceCloud'13, June 17, 2013, New York, NY, USA.*

[15] BIOLCHINI J., ***Systematic Review in Software Engineering****. Systems Engineering and Computer Science Department* COPPE/UFRJ. Rio de Janeiro. 2005.

[16] Cattell, R., **Scalable SQL and NoSQL data stores,** New York, NY, USA, 2010.

[17] Carniel, A. C., Aguiar, A., Porto, V. H., **Query processing over data warehouse using relational databases and NoSQL. In:** XXXVIII Conferencia Latinoamericana En**,** 2012, Departamento de computação, Universidade Federal de São Carlos.

[18] Neo4j product disponível em: https://neo4j.com/product/. Acesso em: 14 dez. 2016.

[19] IEEE disponível em: http://ieeexplore.ieee.org/search/advsearch.jsp. Acesso em: 2 out. 2016.

[20] ACM disponível em: http://portal.acm.org/advsearch.cfm. Acesso em: 3 out. 2016.

[21] PostgreSQL Documentation 9.5.5 disponível em: https://www.postgresql.org/docs/9.5/static/queries.html. Acesso em: 14 dez. 2016.

[22] MongoDB versão 3.2.11 disponível em: https://www.mongodb.com/download-center?jmp=hero#community. Acesso em: 14 dez. 2016.

[23] BrModelo versão 2.0 disponível em: http://sis4.com/brModelo/download.aspx. Acesso em: 14 dez. 2016.