UFMT

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE ESPECIALIZAÇÃO EM
BANCO DE DADOS

MODELAGEM DE BANCO DE DADOS NÃO RELACIONAL EM PLATAFORMA BIG DATA VISANDO DADOS DE INTERNET DAS COISAS

CLEITON VITOR FERNANDES

CUIABÁ - MT

2017

UFMT

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE ESPECIALIZAÇÃO EM
BANCO DE DADOS

MODELAGEM DE BANCO DE DADOS NÃO RELACIONAL EM PLATAFORMA BIG DATA VISANDO DADOS DE INTERNET DAS COISAS

CLEITON VITOR FERNANDES

Orientador: Prof. Dr. Roberto Benedito de Oliveira Pereira

Coorientador: Prof. MSc. Nilton Hideki Takagi

Monografia apresentada ao Curso de Especialização em Banco de Dados, do Instituto de Computação da Universidade Federal de Mato Grosso, como requisito para obtenção do título de Especialista em Banco de Dados.

CUIABÁ - MT

2017

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE ESPECIALIZAÇÃO EM BANCO DE DADOS

CERTIFICADO DE APROVAÇÃO

Título: Modelagem de banco de dados Não Relacional em plataforma Big Data visando dados de Internet das Coisas

Autor: Cleiton Vitor Fernandes

Trabalho aprovado em 17 de Fevereiro de 2017.

Comissão examinadora:

Prof. Dr. Roberto Benedito de Oliveira Pereira
Orientador

Prof. MSc. Nilton Hideki Takagi
Instituto de Computação - UFMT

Prof. MSc. Nielsen Cassiano Simões
Instituto de Computação - UFMT

- Dedico, primeiramente ao meu pai, Vitor Mauro Alvellos Fernandes, que me incentivou, acreditou, investiu, ajudou e jamais me deixou desistir ou desanimar nessa missão de estudar, aprender e trabalhar na área que escolhi. Foi ele a primeira pessoa a me dar apoio e que me colocou desde meus 10 anos de idade a seguir na área da informática. Posteriormente, a minha mãe Carmelinda Almeida Fernandes, que na ausência de meu pai foi quem continuou dando força em todos os sentidos para que eu pudesse conquistar tudo que eu tenho até agora.

AGRADECIMENTOS

Agradeço a Deus por tudo que tenho conquistado em minha vida e a todas as pessoas envolvidas diretamente e indiretamente neste trabalho.

Agradecimentos especiais aos professores Dr. Roberto Benedito de Oliveira Pereira e MSc. Nilton Hideki Takagi por me orientar e acreditar neste trabalho e a minha colega de sala e de trabalho Thais Fernanda Bueno da Silva que tanto me incentivou desde o inicio deste projeto até a sua conclusão.

Sem orientação, opinião, correção e empréstimos destas pessoas teria sido muito mais difícil realizar este trabalho.

RESUMO

A Internet das Coisas trabalha com uma grande quantidade de dispositivos ligados a internet e o constate crescimento de usuários que utilizam estes dispositivos gera uma massa gigantesca de dados que cresce a cada ano, por esta razão, o Big Data tem sido o mais recomendado para armazenar os dados gerados. Este trabalho visou ajudar na necessidade de melhorar o armazenamento dos dados e disponibiliza-los de forma mais rápida através de uma modelagem de um banco de dados não relacional baseado em Big Data. Testes de exportação de dados foram realizados em um banco de dados relacional PostgreSQL e importação destes dados em um banco de dados não relacional MongoDB de duas fontes de dados meteorológicos diferentes. Nestes testes foi constatado a flexibilidade e escalabilidade, da modelagem. Ainda foram realizados testes que constataram a performance do banco de dados com esta modelagem através de consultas realizadas diretamente no banco de dados MongoDB, que encontrou uma dificuldade na consulta com mais de 50 mil registros executado na interface gráfica Robomongo. Dificuldade esta, que não inviabiliza a utilização da modelagem para o seu fim principal que é o armazenamento, flexibilidade, adaptabilidade e escalabilidade.

Palavras-chaves: Modelagem de Banco de Dados Não Relacional, MongoDB, Internet das Coisas.

ABSTRACT

Internet of Things works with a great amount of devices connected to Internet and the constant growth of users who use these devices, generates a huge mass of data that grows every year, for this reason Big Data has been the most recommended to store the generated data. This work aimed to help in this need to improve the storage of the data and to make it available more quickly through a non-relational database modeling based on Big Data. Data export tests were performed in PostgreSQL relational database and imported this data into a non-relational database MongoDB from two different meteorological data sources. These tests verified the flexibility and scalability of the modeling. Was also performs tests that verified the performance of the database with this modeling through queries performed directly in the MongoDB database. Tests that also found a difficulty in the query with more than 50 thousand records executed in the graphical interface Robomongo. Difficulty is this that doesn't prevent the use of modeling for its main purpose that is storage, flexible, adaptable and scalable

Keywords: Non-Relational Database Modeling, MongoDB, Internet of Things.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BSON Binary JSON - JSON Binário

CAP *Consistency, Availability and Partition Tolerence* - Consistência, Disponibilidade e Tolerância a Partição

CERN *Conseil Européen pour la Recherche Nucléaire* - Organização Européia para Pesquisa Nuclear

DDL *Data-Definition-Language* - Linguagem de Definição de Dados

DML *Data-Manipulation-Language* - Linguagem de Manipulação de Dados

EMA Estação Meteorológica Automática

GB *Gigabyte*

INMET Instituto Nacional de Meteorologia

IoT *Internet of Things* - Internet das Coisas

JSON *JavaScript Object Notation* - Notação de Objeto de JavaScript

NoSQL *Not Only SQL* - Não Somente SQL

PB *Petabyte*

PGFA Programa de Pós-Graduação da Física Ambiental

RAM *Random Access Memory*

RFID *Radio-Frequency IDentification* - Identificação por Radiofrequência

SGBD Sistema Gerenciador de Banco de dados

SQL *Structured Query Language* - Linguagem de Consulta Estruturada

TE *Terabyte*

TI Tecnologia da Informação

VM *Virtual Machine* - Máquina Virtual

XML *eXtensible Markup Language* - Linguagem de Marcação Extensível

ZB *Zettabyte*

CAPÍTULO 1

INTRODUÇÃO

Vivi-se hoje, na era da informação, como diz Negroponte (2003), na qual a cada dia mais dispositivos estão conectados e possuem cada vez mais sensores que coletam, por sua vez, mais informações. Em 2010, a quantidade total de dados gerados por estes dispositivos ultrapassou a marca de 1 zettabyte (ZB), ao final de 2011, o número cresceu para 1,8ZB, além disso, espera-se que esse número chegue a 35ZB em 2020 (ZASLAVSKY; PERERA; GEORGAKOPOULOS, 2012).

O gerenciamento de grandes volumes de dados, como os gerados por esses dispositivos, é um requisito importante de uma plataforma de sistema, permitindo que ela possa acompanhar a demanda de coleta e análise de dados e, consequentemente, prover respostas, decisões e/ou atuações de maneira eficiente.

Neste contexto, surgem desafios para a persistência, consulta, indexação, processamento e manipulação de transações. Recentemente, soluções baseadas em Big Data têm surgido como uma potencial resposta a alguns desses desafios, a fim de permitir lidar com um imenso volume de dados, diverso e não estruturado (SOLDATOS; SERRANO; HAUSWIRTH, 2012).

Não existe uma definição exata do que é Big Data, contudo, no intuito de tentar definir, são usados como base algumas de suas características principais. A Gartner é uma empresa americana de pesquisa e consultoria que fornece informações sobre tecnologia da

informação para líderes de TI e outros líderes de negócios localizados em todo o mundo, e a mesma convencionou junto a indústria de tecnologia três características que podem ser usadas para melhor definir Big Data conhecidas como 3V: volume, variedade e velocidade (ZASLAVSKY; PERERA; GEORGAKOPOULOS, 2012), conforme Figura 1.

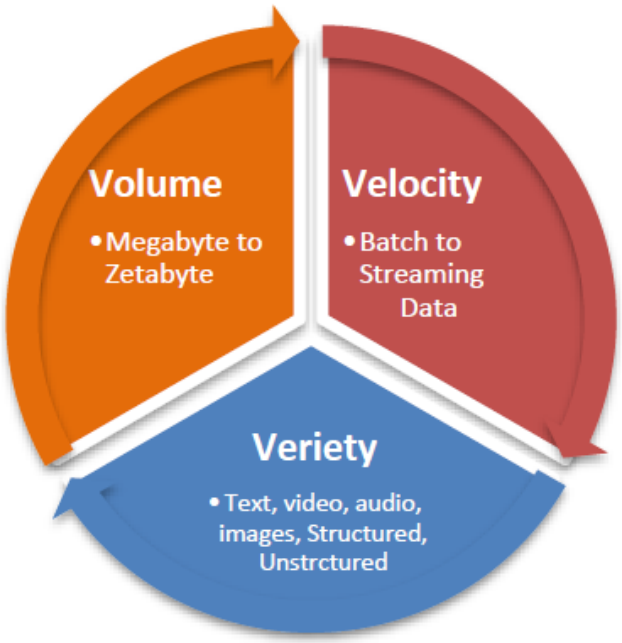

Figura 1 – Características do Big Data Fonte: (LI et al., 2012)

Contudo (LI et al., 2012) define essas características da seguinte forma:

- Volume: refere-se ao tamanho dos dados, tais como terabytes (TB), petabytes (PB), zettabytes (ZB) e assim por diante.
- Variedade: é tipos de dados, por exemplo, os dados podem ser logs da web, leituras de sensores RFID, dados de redes sociais não estruturados, *streaming* de vídeo e áudio.
- Velocidade: significa a frequência com que os dados são gerados. Por exemplo, cada milissegundo, segundo, minuto, hora, dia, semana, mês, ano. Alguns dados precisam ser processados em tempo real, já outros podem ser processados quando necessário. Tipicamente, podemos identificar três categorias principais: ocasionais, frequentes e em tempo real.

Segundo (LI et al., 2012), há uma série de desafios relacionados a grande massa dados. Devido às suas características alguns dos desafios herdados são captura, armazenamento, pesquisa, análise e virtualização.

> Atualmente, Big Data considera volume de dados que podem chegar até Terabytes, em um futuro não muito distante Big Data irá considerar volumes de dados a partir de Petabytes. Além disto, a proposta deve ser capaz de dar suporte ao processamento desses dados, (...) com uma modelagem capaz de facilitar tanto as consultas quanto as inferências sobre os dados, ou seja, uma modelagem adaptável capaz de suportar dados heterogêneos de forma simples. (PERNAMBUCO, 2014)

Dadas as características da proposta de modelagem citadas, é necessário escolher um banco de dados que atenda de forma mais simples e eficiente. Os bancos de dados relacionais são estruturados e muito rígidos, dificultando uma modelagem com estas características.

Em comparação com os bancos de dados relacionais, os bancos de dados não relacionais, atualmente também chamado de NoSQL, são mais flexíveis e escaláveis horizontalmente. Uma das principais vantagens das bases de dados NoSQL é representada pela inexistência de uma estrutura de dados rígida que, nos bancos de dados relacionais, deve ser definida antes que os dados possam ser armazenados. O banco de dados NoSQL permite o gerenciamento de aplicativos mais fácil e remove a necessidade de modificação de aplicativos ou mudança de esquema de banco de dados e, além disso, é melhor e mais fácil em escalabilidade horizontal (Veronika Abramova; Jorge Bernardino and Pedro Furtado, 2005).

No entanto, há ainda uma série de desafios a serem superados para alavancar a ampla disseminação desse paradigma, principalmente com relação ao desenvolvimento de aplicações e à alta heterogeneidade decorrente da inerente diversidade de tecnologias de *hardware* e *software* desse ambiente. (PIRES et al., 2015)

Apesar de todos estes desafios, existe um crescimento da produção de dispositivos e sistemas inteligentes que cada vez mais se conectam através de redes locais e pela internet e que juntos estes dispositivos formam o que hoje é chamado de Internet das Coisas. A grande quantidade de conectividade entre esses dispositivos tem criado uma grande massa de dados e, para poder trabalhar com tal volume, é necessário projetar, modelar e implantar soluções usando uma tecnologia como Big Data, que pode utilizar dados estruturados e não estruturados (SOUZA; SANTOS, 2015).

Baseado na análise das características dos dados da Internet das Coisas, Li et al. (2012) propõe uma solução de gerenciamento de armazenamento diferente, com base em NoSQL como solução para os armazenamentos atuais que não suporta muito bem o armazenamento de dados massivos e heterogêneos da Internet das coisas.

Para se ter uma ideia da importância da Internet das Coisas, a IBM anunciou a compra da empresa de informações meteorológicas Weather.com e mais um investimento de 3 bilhões de dólares em serviços relacionados à Internet das Coisas. No caso da transação com a Weather Company, o que interessa à IBM em particular é a plataforma dinâmica de dados em nuvem, que é o motor do seu aplicativo móvel - o quarto aplicativo mais usado nos Estados Unidos - e que lida com 26 bilhões de requisições por dia no seu serviço baseado em nuvem. A aquisição faz parte da estratégia da IBM de aumentar sua presença no mercado de Internet das Coisas (IoT) e vai integrar a nova divisão *Watson IoT Unit* e a plataforma *Watson IoT Cloud* Booton (2016).

Para atender a demanda de tamanho, volume, variedade e velocidade da internet das coisas é necessário, entre outras coisas, um banco de dados NoSQL que, em conjunto com uma arquitetura integrada de Big Data, revolve boa parte desses itens. Talvez a solução que chegue mais próxima, mesmo assim muito longe do que deve atingir a Internet das Coisas em um futuro próximo, é a arquitetura mista baseada em uma modelagem de banco de dados NoSQL adequada com computação distribuída em nuvem. Como principal objeto de proposta deste trabalho é tão somente a Modelagem de Banco de Dados NoSQL, não será abordado as outras camadas da arquitetura de uma solução de Internet das Coisas.

O objetivo geral desse trabalho, visando atender uma necessidade da Internet das Coias, se concentra na modelagem de dados estrutural em banco de dados NoSQL baseado em Big Data.

A modelagem proposta deve ser escalável, adaptável e que seja mais adequada para utilização de dados pré-processados gerados por sensores meteorológicos.

Para atingir tal objetivo foram propostos os seguintes objetivos específicos:

- Investigar os modelos de banco de dados não relacional disponíveis no mercado que seja escalável e adaptável;
- Investigar o armazenamento de dados oriundos da Internet das Coisas;
- Desenvolver um estudo de caso utilizando dados sensoriais meteorológicos do Instituto Nacional de Meteorologia e do programa de pós-graduação da Física Ambiental da Universidade Federal de Mato Grosso;
- Avaliar e homologar o estudo de caso 1: Modelagem dos dados, onde será realizado a modelagem do banco de dados, estudo de caso 2: Popular base de dados, onde os dados serão preparados e populados no banco de dados com a modelagem deste trabalho e estudo de caso 3: Verificação de performance, onde serão realizados testes de performance através de consultas.

Para todos os objetivos propostos neste trabalho serão realizados os seguintes procedimentos:

Pesquisa bibliográfica: consiste na busca e leitura de material de caráter cientifico por meio impresso ou digital. Este material é fundamental para embasamento teórico do projeto. Pesquisa experimental: será utilizado um banco de dados não relacional para desenvolver e aplicar uma modelagem voltado para dados sensoriais. A modelagem será testada com dados meteorológicos fornecidos pelo programa de pós-graduação em Física Ambiental da Universidade Federal de Mato Grosso e do site do INMET (Instituto Nacional de Meteorologia).

A partir dos objetivos definidos, este trabalho foi organizado em 5 capítulos.

No Capítulo 2, Fundamentação Teórica, é apresentado o resultado da pesquisa bibliográfica realizada sobre todos os principais itens de estudo envolvidos como os conceitos de Big Data, banco de dados relacional e não relacional ou NoSQL, modelagem de banco de dados NoSQL e Internet das Coisas, para fundamentar os estudos de caso aqui propostos.

No capítulo 3, Desenvolvimento da Modelagem de banco de dados Não Relacional em plataforma Big Data visando dados de Internet das Coisas, serão descritos todos os componentes de hardware e software escolhidos para realização de cada estudo de caso e os detalhes dos cenários dos testes propostos, como os scripts a serem utilizados no desenvolvimento de cada estudo de caso.

No capítulo 4, Resultados e Discussões, serão discutidos os resultados do cenário de cada estudo de caso proposto.

No Capitulo 5, Conclusões, serão revistas as hipóteses iniciais comparadas aos resultados e explicado se os resultados conseguiram atingir de forma satisfatória ou não os objetivos propostos .

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Este capitulo se dedica a apresentar o resultado dos estudos bibliográficos realizados, inciando com o conceito de modelagem de dados, descrevendo os modelos de dados relacional e não relacional, conceito de banco de dados, demostrando um sistema gerenciador de banco de dados e principais características de Big Data que foram pesquisados em livros, artigos, documentação de software para fundamentar os objetivos deste trabalho.

2.1 Modelagem de Dados

Modelagem é o processo de criação de um modelo de dados para um sistema de informação através da aplicação de técnicas de modelagem de dados. É um processo usado para definir e analisar os requisitos necessários para suportar os processos de negócios no âmbito dos sistemas de informação correspondentes nas organizações (SILVERSTON, 1997).

A modelagem conceitual é uma fase muito importante no projeto de uma aplicação de banco de dados, em muitas ferramentas de projeto de software, as metodologias de projeto de banco de dados e de engenharia de software são interligadas, pois essas atividades estão fortemente relacionadas (ELMASRI; NAVATHE, 2011).

O projeto de um banco de dados é dividido em algumas etapas como a de levantamento e análise de requisitos, projeto conceitual, projeto lógico ou mapeamento do modelo de dados e projeto físico, conforme a Figura 2.

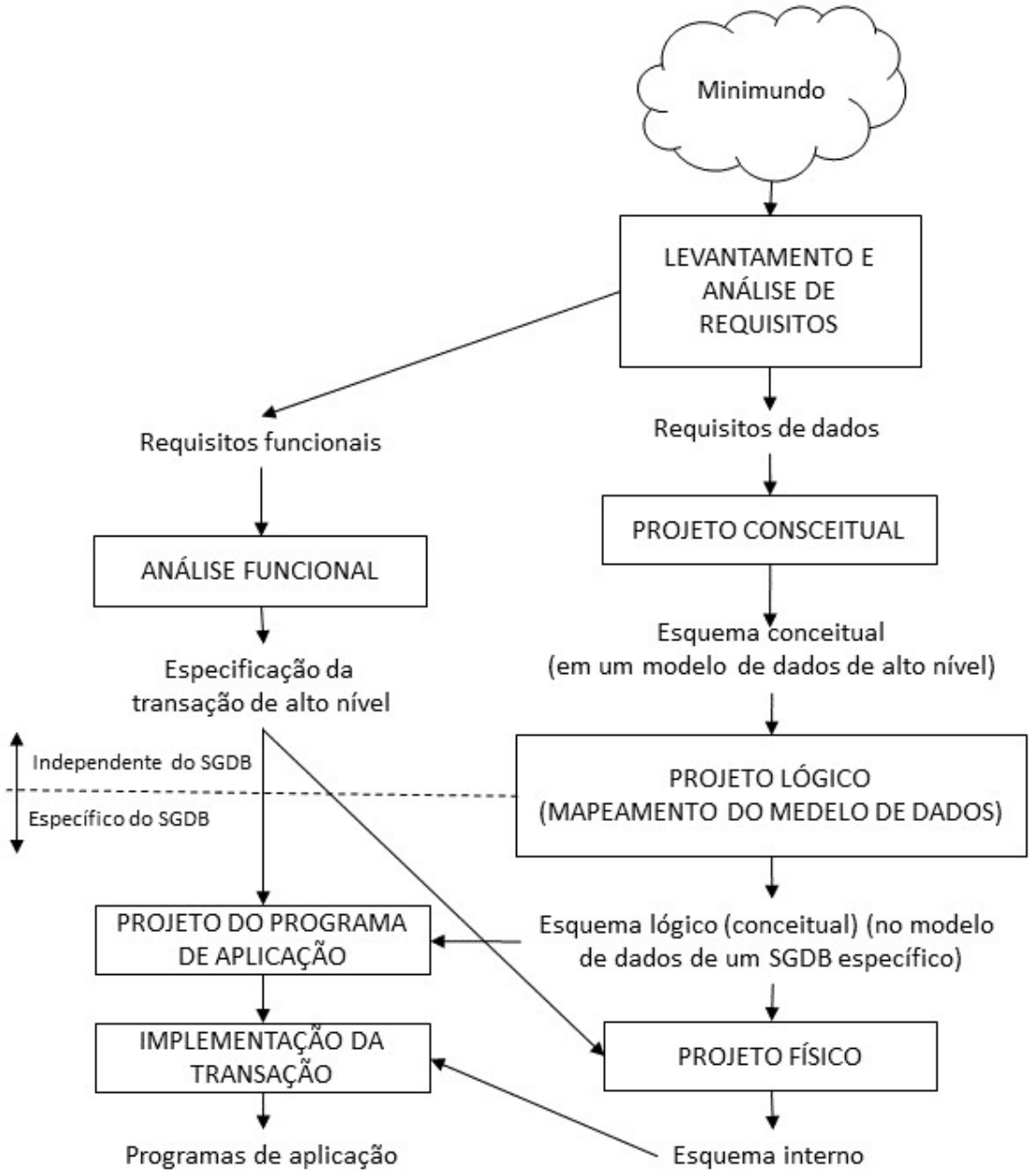

Figura 2 – Diagrama simplificado para ilustrar as principais fases do projeto de banco de dados. Fonte: (ELMASRI; NAVATHE, 2011)

Embora existam muitas maneiras de criar modelos de dados, de acordo com (SILVERSTON, 1997), apenas duas metodologias de modelagem se destacam: *bottom-up* e *top-down*:

Os modelos *Bottom-up* ou *View Integration* geralmente começam com formulários de estruturas de dados existentes, campos em telas de aplicativos ou relatórios.

Esses modelos são geralmente físicos, específicos da aplicação e incompletos do ponto de vista empresarial. Eles não podem promover a partilha de dados, especialmente se forem construídos sem referência a outras partes da organização.

Modelos de dados lógicos *Top-down*, por outro lado, são criados de forma abstrata obtendo informações de pessoas que conhecem a área de assunto. Um sistema pode não implementar todas as entidades em um modelo lógico, mas o modelo serve como um ponto de referência.

O modelo de dados é um conjunto de ferramentas conceituais para a descrição, relacionamento, semântica de dados e regras de consistência. Os modelos mais conhecidos são: modelos lógicos com base em objetos, modelos lógicos com base em registros e modelo físicos de dados. (Abraham Silberschatz, Henry Korth, 1999).

2.1.1 Modelos Lógicos com Base em Objetos

São usados na descrição de dados no nível lógico e de visões. Existem vários modelos, mas os mais conhecidos são:

2.1.1.1 Modelo Entidade-Relacionamento

Há várias anotações para modelagem de dados. O modelo real é frequentemente chamado de "Modelo de Entidade-Relacionamento"porque retrata dados em termos das entidades e relacionamentos descritos nos dados (WHITTEN; BENTLEY; DITTMAN, 1998).

A modelagem Entidade-Relacionamentos é um método de modelagem de banco de dados de esquema relacional, usado na engenharia de software para produzir um modelo de dados conceitual ou modelo de dados semântico de um sistema, muitas vezes um banco de dados relacional e seus requisitos. Esses modelos são usados na primeira fase do projeto do sistema de informações durante a análise de requisitos para descrever as necessidades de informação ou o tipo de informação que deve ser armazenada em um banco de dados. As técnicas de modelagem de dados podem ser usadas para descrever qualquer ontologia, isto é, uma visão geral e classificações de termos usados e suas relações para um determinado universo de discurso ou área de interesse (WHITTEN; BENTLEY; DITTMAN, 1998).

O modelo Entidade-Relacionamento tem por base a percepção do mundo real como um conjunto de objetos chamados entidade. Entidade é um objeto do mundo real que pode se relacionar com outros objetos através dos seus atributos (Abraham Silberschatz, Henry Korth, 1999).

Por exemplo, um relacionamento é uma associação entre entidades, o depositante associa um cliente a cada conta que ele possui. Uma linha pode-se armazenar uma entidade como um cliente e em cada coluna ou atributo desta entidade pode ser armazenado informações do mesmo, como nome e endereço. Toda estrutura lógica do banco de dados pode ser expressa graficamente por meio do diagrama Entidade Relacionamento, cujos construtores dos seguintes componentes são: (Abraham Silberschatz, Henry Korth, 1999):

- Retângulo: representa os conjuntos de entidades;
- Elipses: representam os atributos;
- Losango: representam os relacionamentos entre os conjuntos de entidades;
- Linhas: unem os atributos aos conjuntos de entidades e o conjunto de entidades aos seus relacionamentos.

Cada componente é rotulado com o nome da entidade ou relacionamento que representa, conforme Figura 3.

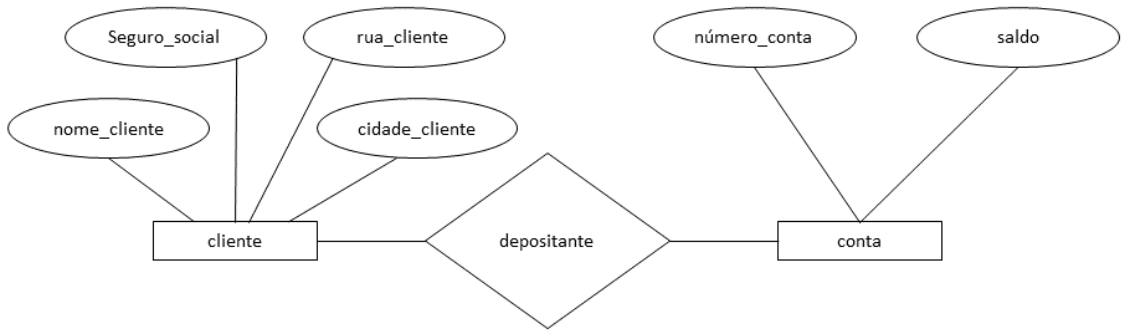

Figura 3 – Diagrama Entidade-Relacionamento representando uma conta bancaria fonte: (Abraham Silberschatz, Henry Korth, 1999)

2.1.1.2 Modelo Orientado a Objeto

O modelo orientado a objetos tem por base um conjunto de objetos que contém valores armazenados em variáveis instâncias e um conjunto de códigos chamados métodos (Abraham Silberschatz, Henry Korth, 1999).

2.1.1.3 Modelo Semântico de Dados

Nos modelos semânticos, o processo de combinação de documentos com determinada consulta é baseado no nível de conceito e combinação semântica. As Abordagens

semânticas incluem diferentes níveis de análise, como as análises morfológica, sintática e semântica, para recuperar documentos com mais eficiência (ELMASRI; NAVATHE, 2011).

2.1.1.4 Modelo Funcional de Dados

O modelo funcional de dados é uma representação estruturada das funções (atividades, ações, processos, operações) dentro do sistema modelado (ELMASRI; NAVATHE, 2011).

2.1.2 Modelos Lógicos com Base em Registros

Modelos lógicos com base em registros são usados para descrever os dados no nível lógico e de visão.

2.1.2.1 Modelo Relacional

O modelo relacional usa um conjunto de tabelas para representar tanto os dados como a relação entre eles. Cada tabela possui uma ou mais colunas e cada uma possui um nome único. A maior vantagem deste tipo de banco de dados é a habilidade de descrever relacionamento entre os dados utilizando junções entre tabelas distintas fortemente tipadas, chaves primárias e chaves estrangeiras, entre outros.

A chave primaria é unívoca, o atributo da chave primária têm um valor único, não redundante e não nulo. A chave estrangeira que determina o relacionamento é um subconjunto de atributos que constituem a chave primária de uma outra relação (Abraham Silberschatz, Henry Korth, 1999).

No modelo relacional uma linha é chamada de tupla, um cabeçalho da coluna é chamado de atributo e a tabela é chamada de relação. O modelo relacional representa o banco de dados como uma coleção de relações. Quando uma relação é considera uma tabela de valores, cada linha na tabela representa uma coleção de valores de dados relacionados. Uma linha representa um fato que corresponde a um entidade ou relacionamento do mundo real conforme Figura 4.

Uma Entidade pode ser concreta como uma pessoa ou livro, ou abstrata, como um empréstimo ou viagem. Ela pode ser representada por um conjunto de atributos, que são propriedades descritivas de cada membro de um conjunto entidade (Abraham Silberschatz, Henry Korth, 1999).

Os valores de um atributo que descrevem as entidades podem ser simples ou compostos, monovalorados ou multivalorados, nulos e derivados.

- Valores simples, quando não são divididos em partes, como, por exemplo, nome_cliente, endereço_cliente.
- valores compostos, quando são divididos em partes como, por, exemplo, prenome, nome_intermediário, sobrenome, rua, numero, bairro, cidade, uf, cep.
- Valores monovalorados, quando um atributo simples pode possuir apenas um valor.
- valores multivalorados, quando um atributo possui um, nenhum ou mais de um valor, por exemplo, um funcionário pode possuir um dependente, nenhum dependente ou mais de um dependente.
- valores nulos, quando um valor não é preenchido, por exemplo, quando um funcionário não possui nenhum dependente, nenhum valor é aplicado fazendo com que o atributo assuma o valor nulo.
- Valores derivados, quando o valor é dependente de outro valor como, por exemplo, um funcionário possui um atributo chamado data_contratação e outro tempo_casa, em que o atributo tempo_casa depende do valor armazenado no atributo data_contratação e a data atual.

Um relacionamento é uma associação entre uma ou várias entidades. A função que uma entidade desempenha em um relacionamento é chamada de papel.

nome_cliente
Johnson
Smith
Hayes
Turner
Johnson
Jones
Lindsay
Smith

seguro_social
192-83-7465
019-28-3746
677-89-9011
182-73-6091
192-83-7465
321-12-3123
336-66-9999
019-28-3746

rua_cliente
Alma
North
Main
Putnam
Alma
Main
Park
North

cidade_cliente
Palo Alto
Rye
Harrison
Stamford
Palo Alto
Harrison
Pittsfield
Rye

numero_conta
A-101
A-215
A-102
A-305
A-201
A-217
A-222
A-201

numero_conta
A-101
A-215
A-102
A-305
A-201
A-217
A-222

saldo
500
700
400
350
900
750
700

Figura 4 – Exemplo de um banco de dados relacional. Fonte: (Abraham Silberschatz, Henry Korth, 1999)

Normalmente, os papeis são distintos, porém quando o mesmo conjunto de entidades participa de um conjunto de relacionamento mais de uma vez pode exercer diferentes papéis. Assim, podemos obter o grau dos relacionamentos, sendo de grau 2 o relacionamento binário e de grau 3 o relacionamento ternário. Para o funcionamento destes conjuntos de relacionamentos é necessário definir algumas restrições , as quais o conteúdo do banco de dados deve respeitar.

O mapeamento das cardinalidades expressa o número de entidades às quais outras entidades podem estar associadas via um conjunto de relacionamentos. Um exemplo de relacionamento R binário entre os conjuntos de entidades A e B, o mapeamento das cardinalidades deve seguir uma das instruções abaixo (Abraham Silberschatz, Henry Korth, 1999):

- Um para Um, uma entidade em A esta associada no máximo a uma entidade em B, e uma entidade em B está associada a no máximo uma entidade em A conforme a Figura 5(a).
- Um para muitos, uma entidade em A está associada a várias entidades em B. Uma entidade em B, entretanto, deve estar associada a no máximo uma entidade em A conforme a Figura 5(b).
- Muitos para um, uma entidade em A está associada a no máximo uma entidade em B. Uma entidade em B, entretanto, pode estar associada a um número qualquer de entidades em A conforme a Figura 6(a).
- Muitos para muitos, uma entidade em A está associada a qualquer número de entidades em B e uma entidade em B está associada a um número qualquer de entidade em A conforme a Figura 6(b).

O rateio de cardinalidades de um relacionamento pode afetar a colocação dos atributos nos relacionamentos. Um esquema de banco de dados relacional tem por base tabelas derivadas de Entidade-Relacionamento, onde é possível determinar a chave primária para um esquema de relação das chaves primárias dos conjuntos de relacionamentos ou entidades cujos esquemas são (Abraham Silberschatz, Henry Korth, 1999):

- Conjunto de entidade forte, onde a chave primária do conjunto de relacionamentos torna-se a chave primária da relação.
- Conjunto de entidades fracas, onde a chave primária do conjunto de entidades fortes do qual o conjunto de entidades fracas é dependente.

- Conjunto de relacionamentos, quando a união das chaves primárias dos conjuntos de entidades relacionados tornam-se superchaves da relação.
- Tabelas combinadas, quando a chave primária do conjunto de entidades "muitos"torna-se a chave primária da relação.

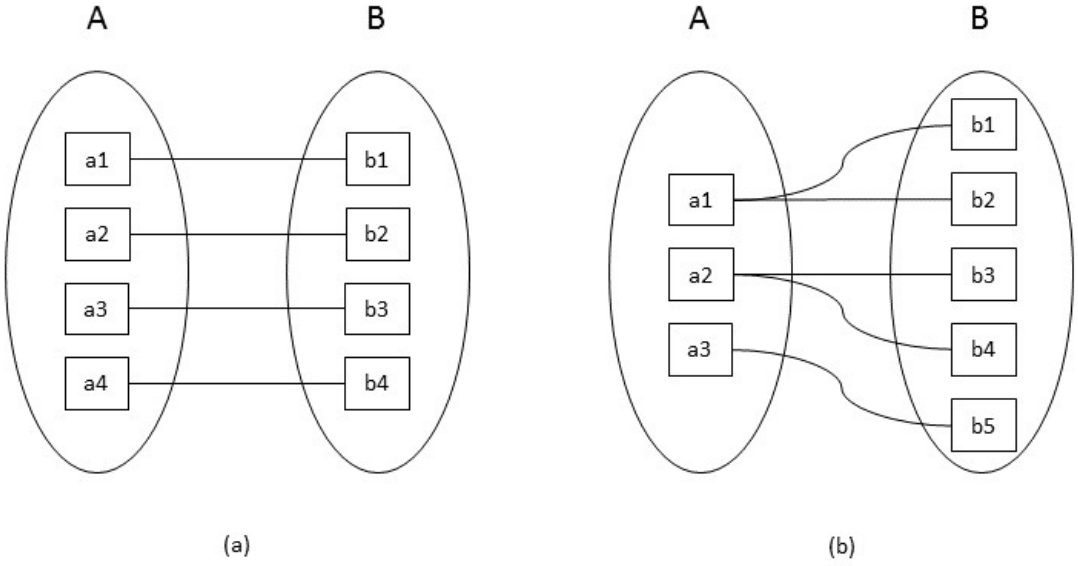

Figura 5 – Mapeamento das cardinalidades. (a) Um para Um. (b) Um para muitos. Fonte: (Abraham Silberschatz, Henry Korth, 1999)

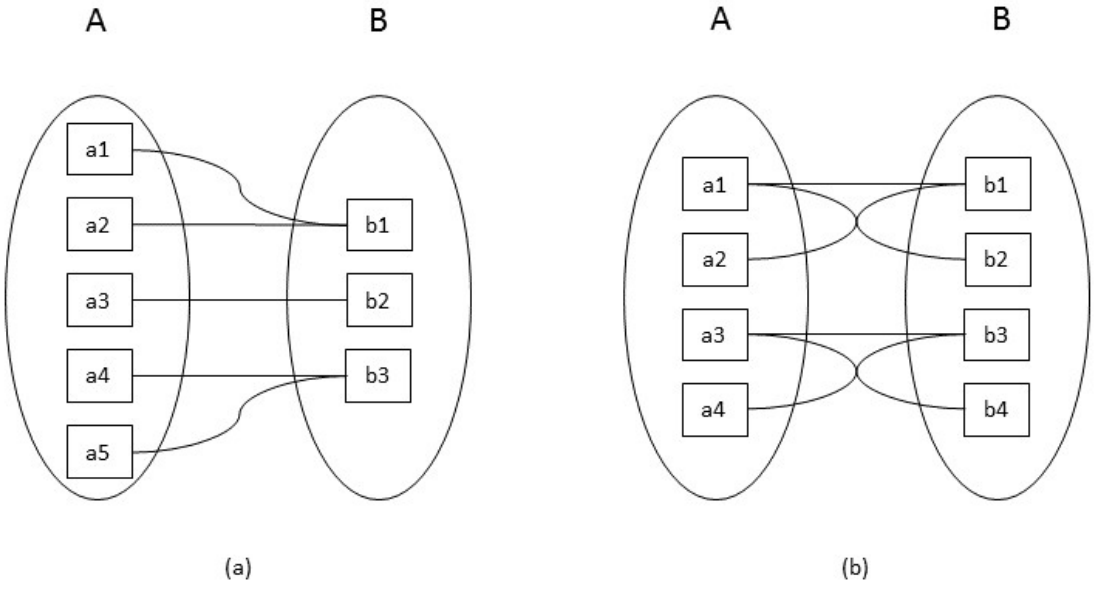

Figura 6 – Mapeamento das cardinalidades. (a) Muitos para Um. (b) Muitos para muitos. Fonte: (Abraham Silberschatz, Henry Korth, 1999)

Da lista descrita, se verifica que um esquema de relação pode incluir entre seus atributos a chave primária de outro esquema, essa chave é chamada chave estrangeira. Com estas informações sobre relacionamentos e chaves é possível realizar operações de consultas através da álgebra relacional que é uma linguagem de consultas procedural. Consiste em um conjunto de operações tendo como entrada uma ou mais relações e produzindo como resultado uma nova relação. A álgebra relacional é considerada a base para a linguagem SQL (ELMASRI; NAVATHE, 2011).

Porém a linguagem SQL é basicamente relacional, não sendo possível utilizá-la em modelagem não relacional(STROZZI, 2007).

2.1.2.2 Modelo de Rede

Modelo de rede são representados por um conjunto de registros e as relações entre estes registros são representados por *links* organizados no banco de dados por um conjunto arbitrário de gráficos.

2.1.2.3 Modelo Hierárquico

Modelo hierárquico é similar ao modelo em rede, pois os dados e suas relações são representados, respectivamente, por registros e *links*, porém no modelo hierárquico os registros estão organizados em árvores.

2.1.2.4 Modelo Orientado a Objetos

Modelo orientado a objetos tem por base um conjunto de objetos. Um objeto contém valores armazenados em variáveis, conjunto de códigos chamados de métodos que operam esse objeto e os objetos que contêm os mesmos tipos de valores e métodos são agrupados em classes.

2.1.3 Modelos Físicos de Dados

Os modelos físicos de dados descrevem o nível mais baixo do banco de dados e são poucos, sendo os mais conhecidos: modelo unificado e modelo de partição de memória. (Abraham Silberschatz, Henry Korth, 1999)

Porém, para esse trabalho o modelo de dados mais importante é o Modelo Não Relacional.

2.1.4 Modelo Não Relacional

Segundo (SADALAGE; FOWLER, 2012), o banco de dados NoSQL surgiu motivada pela necessidade de trabalhar com grandes volumes de dados. Seus defensores afirmam que os sistemas desenvolvidos com banco de dados Não Relacionais são mais flexíveis do que os bancos de dados Relacionais, já que são livres de esquemas rígidos, são distribuídos, escaláveis, possuem melhor desempenho e não necessitam de uma infraestrutura robusta para armazenar seus dados.

Carlo Strozzi introduziu este nome em 1998 como o de um banco de dados relacional de código aberto que não possuía uma interface SQL. O termo NoSQL foi re-introduzido no início de 2009 por um funcionário do Rackspace, Eric Evans, quando Johan Oskarsson da *Last.fm* queria organizar um evento para discutir bancos de dados open source distribuídos.(STROZZI, 2007)

Dentre os banco de dados NoSQL existem várias categorias com características e abordagens diferentes para o armazenamento, relacionadas principalmente com o modelo de dados utilizado, as principais categorias são (PERNAMBUCO, 2014):

2.1.4.1 Chave/Valor

Os dados são armazenados sem esquema pré-definido, no formato de pares Chave/Valor, onde tem-se uma Chave que é responsável por identificar o dado e seu valor que corresponde ao armazenamento do dado em sí.

2.1.4.2 Orientado a Colunas

Os dados são armazenados em famílias de colunas com a adição de alguns atributos dinâmicos. Por exemplo, são utilizadas chaves estrangeiras, mas que apontam para diversas tabelas diferentes, sendo assim este banco de dados não é relacional.

2.1.4.3 Orientado a Documentos

Cada documento contém uma informação única pertinente a um único objeto, mantendo a estrutura dos objetos armazenados. Em geral, todos os dados são assumidos como documentos, que por sua vez são encapsulados e codificados em algum formato padrão, podendo ser estes formatos XML, YAML, JSON, BSON, assim como formatos binários como PDF e formatos do Microsoft Office.

2.1.4.4 Grafos

Os dados são armazenados em uma estrutura de nós, arestas e propriedades, com os nós representando as entidades em si, as arestas representando os relacionamentos entre os nós e as propriedades representando os atributos das entidades, podendo estas serem representadas tanto nos nós quanto nas arestas do grafo.

Esse tipo de banco de dados utiliza teorias matemáticas dos grafos, muito utilizadas nas mais diversas aplicações, com uma modelagem mais natural dos dados. É possível realizar consultas com um alto nível de abstração. Por esses motivos, esta categoria é indicada para aplicações em redes sociais e que necessitam de processamento inteligente como inferências a partir dos dados armazenados.

2.2 Banco de Dados

Banco de dados é uma coleção organizada de dados relacionados (ELMASRI; NAVATHE, 2011). Um banco de dados representa algum aspecto do mundo real logicamente coerente com algum significado inerente. Sistemas de banco de dados são projetados, construídos e populados com dados para uma finalidade especifica. O gerenciamento desses dados implica na definição das estruturas de armazenamento e dos mecanismos de manipulação dessas informações e ainda na garantia de segurança dos dados tanto no armazenamento quanto no acesso de usuários não autorizados.(Abraham Silberschatz, Henry Korth, 1999).

Um banco de dados pode ter qualquer tamanho, complexidade e pode ser gerado e mantido manualmente ou computadorizado. Um catálogo de fichas clinicas de pacientes de uma clinica odontológica pode ser criado e mantido manualmente em papel e armazenado em gavetas de armário, já um banco de dados computadorizado pode ser criado e mantido por um grupo de aplicações específicas para esta tarefa ou por um sistema gerenciador de banco de dados (ELMASRI; NAVATHE, 2011).

2.3 Sistema Gerenciador de Banco de Dados

Um Sistema Gerenciador de Banco de Dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados (ELMASRI; NAVATHE, 2011). O principal objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações no banco de dados e facilitar o processo de definição, construção, manipulação e compartilhamento de banco de dados entre usuários e aplicações (Abraham Silberschatz, Henry Korth, 1999).

A definição ou informação descritiva do banco de dados também é armazenada pelo SGDB na forma de catálogo ou dicionário, chamado de metadados. A manipulação de um banco de dados inclui funções como consulta ao banco de dados para recuperar dados específicos. O compartilhamento de um banco de dados permite que usuários e programas acessem simultaneamente. Um programa de aplicação acessa o banco de dados ao enviar consultas ou solicitações de dados ao SGDB (ELMASRI; NAVATHE, 2011).

Outras funções importantes fornecidas pelo SGDB incluem proteção do sistema contra falhas de hardware ou software, através do seu subsistema de *backup* e *restore*. O subsistema de recuperação é responsável por garantir que o banco de dados seja restaurado ao estado em que estava antes da transação ser executada. O *backup* ou cópia de segurança de disco também é necessário no caso de uma falha catastrófica de disco. Inclui também proteção de segurança contra acesso não autorizado ou malicioso, uma vez que diversos tipos de usuários ou grupo de usuários compartilham a mesma base de dados. O SGBD deve oferecer vários níveis de acesso através de uma conta de usuário protegida por senha. O subsistema de segurança é responsável pelas restrições de cada conta de usuário responsável pela administração do sistema terá privilégios que um usuário comum não tem, pois deve apenas manipular os dados ou até mesmo somente consultar algumas informações sem permissão para realizar qualquer tipo de alteração (ELMASRI; NAVATHE, 2011).

Há muitos tipos de SGBD, desde pequenos sistemas que funcionam em computadores pessoais, a sistemas enormes que estão associados a mainframes. Um modelo de dados para um SGBD define como os dados serão armazenados no banco de dados(Abraham Silberschatz, Henry Korth, 1999).

O maior benefício de um banco de dados é proporcionar ao usuário uma visão abstrata dos dados (Abraham Silberschatz, Henry Korth, 1999). O sistema oculta determinados detalhes sobre a forma de armazenamento e manutenção desses dados. O SGBD age como interface entre os programas de aplicação e os ficheiros de dados físicos e separa as visões lógica e de concepção dos dados. Assim sendo, são basicamente três os componentes de um SGBD:

- Linguagem de definição de dados (específica conteúdos, estrutura a base de dados e define os elementos de dados);
- Linguagem de manipulação de dados (para poder alterar os dados na base);
- Dicionário de dados (guarda definições de elementos de dados e respectivas características e descreve os dados.

Existem muitos tipos de ferramentas completas e com funcionalidades acrescidas que elevam para outros níveis a capacidade operacional de gerar e gerenciar informação de valor para a organização, segue exemplo de algumas destas ferramentas SGBD: Firebird, HSQLDB, IBM DB2, IBM Informix, JADE, MariaDB, Microsoft Visual Foxpro, MongoDB, mSQL, MySQL, Oracle, PostgreSQL, SQL-Server, Sybase, TinySQL, ZODB.

Para completar a definição inicial do SGDB, é uma coleção de arquivos e programas inter-relacionados ilustrado na Figura 7.

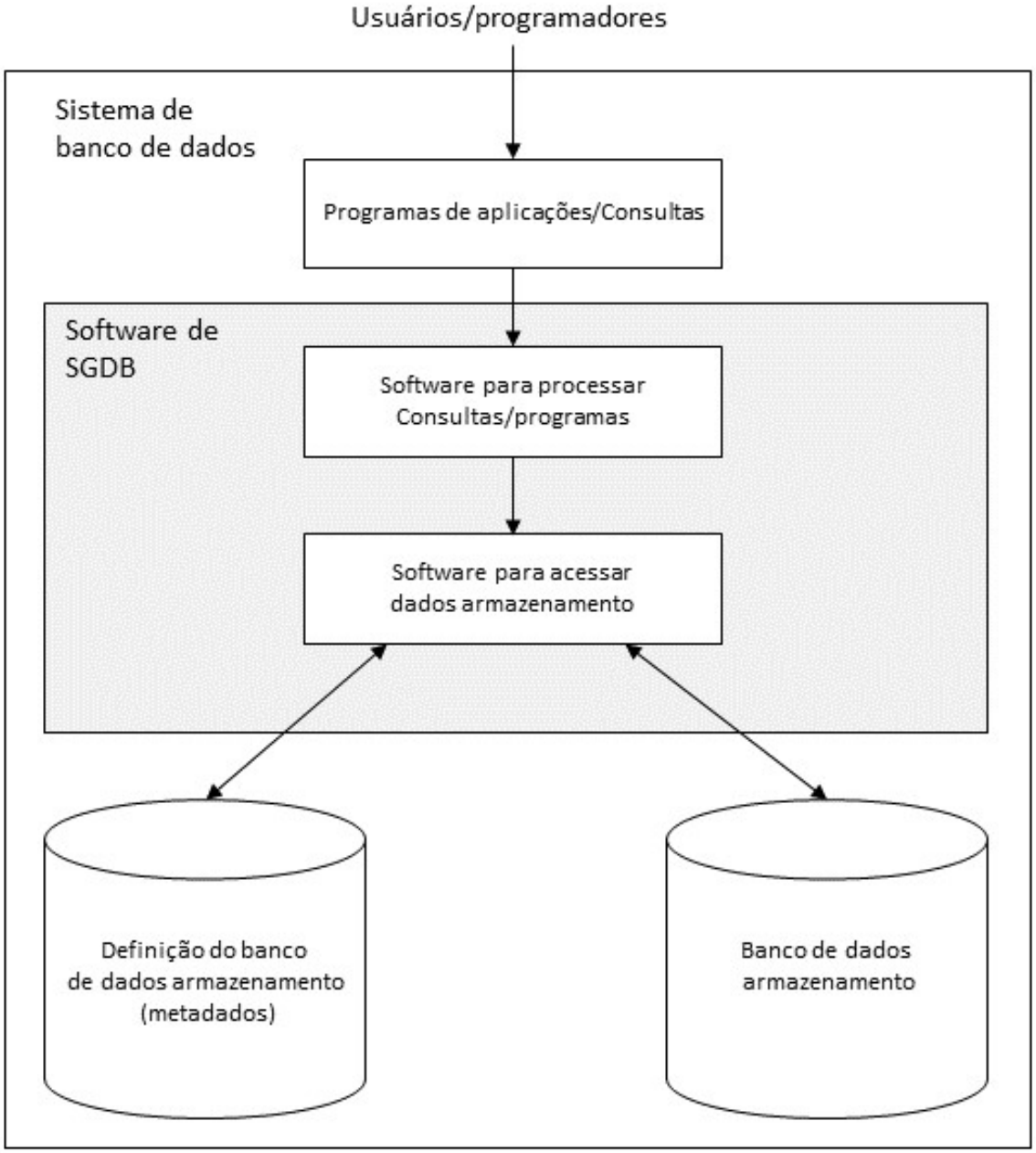

Figura 7 – Diagrama simplificado de um ambiente de sistema de banco de dados. Fonte: (ELMASRI; NAVATHE, 2011)

2.3.1 SGBD - Relacional

Para que se possa usar um sistema, ele precisa ser eficiente na recuperação das informações. Esta eficiência está relacionada à forma pela qual foram projetadas as complexas estruturas de representação desses dados no banco de dados (Abraham Silberschatz, Henry Korth, 1999).

Assim, o projeto do banco de dados deve considerar a interface entre o sistema de banco de dados e o sistema operacional. Os componentes funcionais do sistema de banco de dados podem ser divididos pelos componentes de processamento de consultas e pelo componentes de administração de memória (Abraham Silberschatz, Henry Korth, 1999).

Os componentes de processamento de consultas são :

- Compilador DML traduz comandos DML da linguagem de consultas em instruções de baixo nível. DML é uma família de linguagem de manipulação de dados dividas em duas categorias: procedural e declarativa. A procedural especifica como os dados devem ser obtidos do banco e a declarativa não necessita especificar o como os dados serão obtidos do banco. Um exemplo de linguagem declarativa mais conhecida é o SQL.
- Pré-compilador para comandos DML, inseridos em programas de aplicação que convertem comandos DML em chamadas de procedimentos normais da linguagem hospedeira.
- Interpretador DDL interpreta os comandos DLL e registra-os em um conjunto de tabelas que contêm metadados.
- Componentes para o tratamento de consultas executam instruções de baixo nível geradas pelo compilador DML.

Os componentes de administração de armazenamento de dados são:

- Gerenciamento de autorizações e integridade, testa o funcionamento das regras de integridade e a permissão de acesso aos dados pelo usuário.
- Gerenciamento de transações, garante que o banco de dados permanecerá em estado consistente.
- Administração de arquivos, gerencia a alocação de espaço no armazenamento em disco.

- Administração de *buffer*, responsável pela intermediação de dados do disco para a memória principal.

Além disso, algumas estruturas de dados são exigidas como parte da implementação física do sistema:

- Arquivo de dados, armazena o próprio banco de dados.
- Dicionário de dados, armazena os metadados relativos à estrutura do banco de dados.
- Índices, proporcionam acesso rápido aos itens de dados que são associados a valores determinados.
- Estatísticas de dados, armazenam informações estatísticas relativas aos dados contidos no banco de dados.

Os componentes e a conexão entre eles são mostrados conforme Figura 8.

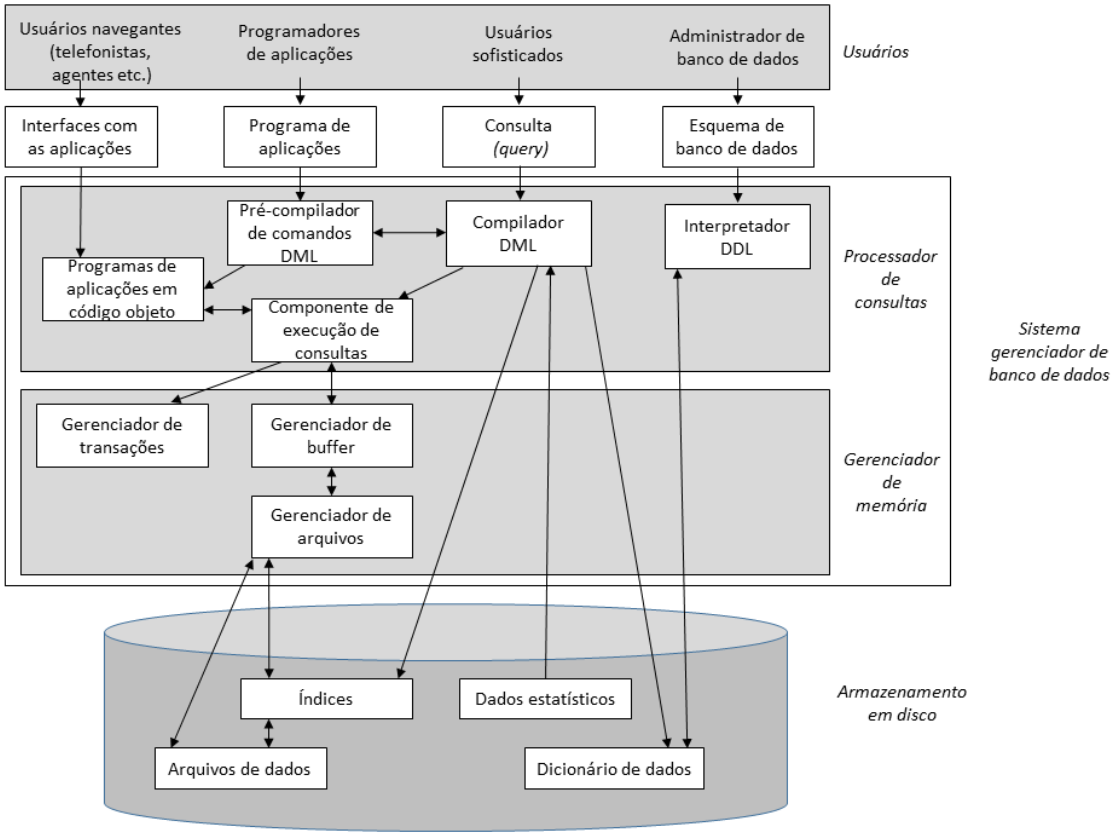

Figura 8 – Estrutura detalhada do Sistema de Gerenciamento Banco de dados Fonte: (Abraham Silberschatz, Henry Korth, 1999)

Conforme os componentes de processamento de consultas, um esquema de dados é especificado por um conjunto de definições expressas por uma linguagem especial chamada linguagem de definição de dados (data-definition language - DDL). O resultado da compilação dos parâmetros DDLs é armazenado em um conjunto de tabelas que constituem um arquivo especial chamado dicionário de dados ou diretório de dados.

Para expressar consulta e atualizações é utilizado uma linguagem chamada linguagem de manipulação de dados (data-manipulation-language - DML). Ela é utilizada para viabilizar o acesso ou a manipulação dos dados de forma compatível ao modelo de dados apropriado. A DML responsável pela recuperação de informações é chamada de linguagem de consulta estruturada (Structured Query Language - SQL) (Abraham Silberschatz, Henry Korth, 1999).

A versão Original dessa linguagem foi desenvolvida em San José no laboratório de pesquisas da IBM nos anos 70. A linguagem SQL proporciona comandos para definição, exclusão e alteração de esquemas de relações, consultas baseadas em álgebra relacional e calculo relacional de tuplas. Ainda possui comandos para definição de visões, especificação de direito de acesso, especificação de regras de integridade, especificação de inicialização e finalização de transações(Abraham Silberschatz, Henry Korth, 1999).

Para garantir essas regras, o SGBD relacional utiliza um conceito de controle transacional chamado ACID (Atomicidade, Consistência, Isolamento e Durabilidade) do inglês *Atomicity, Consistency, Isolation, Durability*(ELMASRI; NAVATHE, 2003):

- Atomicidade: A transação deve ter todas as suas operações executadas em caso de sucesso ou nenhum resultado, em caso de falha, ou seja, todo o trabalho é feito, ou nada é feito. Neste caso, não existe resultados parciais da transação.
- Consistência: A execução de uma transação deve levar o banco de dados de um estado consistente a um outro estado consistente, ou seja, uma transação deve terminar respeitando as regras de integridade e restrições de integridade lógica previamente assumidas.
- Isolamento: É um conjunto de técnicas que tentam evitar que transações concorrentes interfiram umas nas outras.
- Durabilidade: Os efeitos de uma transação em caso de sucesso devem persistir no banco de dados, mesmo em casos de quedas de energia, travamentos ou erros. Garante que os dados estarão disponíveis em definitivo.

Os SGBDs relacionais mais conhecidos e utilizados pelo mercado são: Oracle, Microsoft SQL Server, PostgreSQL, MySQL, entre outros.

As arquiteturas de SGBD relacional, têm como possível dificuldade tornar os bancos de dados convencionais escaláveis e mais baratos, além de manter um esquema rígido na modelagem dos dados (SADALAGE; FOWLER, 2012).

Em 1998, surgiu o termo Banco de Dados Não-Relacional, baseado em uma solução de banco de dados que não oferecia uma interface SQL, mas esse sistema ainda era baseado na arquitetura relacional. Posteriormente, o termo passou a representar soluções que promoviam uma alternativa ao Modelo Relacional, tornando se uma abreviação de Not Only SQL (não apenas SQL) (BRITO, 2010).

2.3.2 SGBD - Não Relacional

Banco de Dados Não-Relacional tem algumas características em comum, tais como serem livres de esquema, promoverem alta disponibilidade e maior escalabilidade (BRITO, 2010). Os primeiros começará a surgir como o SGBD orientado a objetos, que tem como principais características, armazenar os dados como objetos, linguagem de programação e o esquema do banco de dados usam o mesmo modo de definição de tipos e os bancos de dados orientados oferecem suporte a versionamento de objetos.

O SGBD Não Relacional, atualmente, mais conhecido como SGBD NoSQL tem como principais características: primeiro e mais óbvio, não utilizar a linguagem SQL, embora não seja regra, mas geralmente são projetos de código aberto e oferecem uma gama de opções para consistência e distribuição. Outra característica é operar sem um esquema, permitindo-lhe livremente adicionar campos para registros de banco de dados sem ter que definir quaisquer alterações na estrutura em primeiro lugar (SADALAGE; FOWLER, 2012).

Os SGBDs NoSQL apresentam algumas características fundamentais que os diferenciam dos tradicionais SGBDs relacionais, tornando-os adequados para armazenamento de grandes volumes de dados não estruturados ou semiestruturados. Segue algumas destas características:

- Escalabilidade horizontal. A ausência de controle de bloqueios é uma característica dos SGBDs NoSQL que torna esta tecnologia adequada para solucionar problemas de gerenciamento de volumes de dados que crescem exponencialmente.
- Ausência de esquema ou esquema flexível. Uma característica evidente é a ausência completa ou quase total do esquema que define a estrutura dos dados modelados. Esta ausência facilita tanto a escalabilidade quanto contribui para um maior aumento da disponibilidade. Em contrapartida, não há garantias da integridade dos dados, o que ocorre nos bancos de dados relacionais, devido à sua estrutura rígida.

- Permite replicação de forma nativa. Diminui o tempo gasto para recuperar informações.
- Consistência eventual. A consistência nem sempre é mantida entre os diversos pontos de distribuição de dados. Esta característica tem como princípio o teorema CAP (*Consistency, Availability and Partition Tolerence*), que diz que, em um dado momento, só é possível garantir duas de três propriedades entre consistência, disponibilidade e tolerância à partição (LI et al., 2012).

Por mais que o SGBD NoSQL não possua esquemas rígidos e inflexíveis, a base de dados, geralmente, há um esquema implícito presente. Esse esquema implícito é um conjunto de suposições sobre a estrutura dos dados no código que manipula os dados. Por exemplo, uma modelagem usando um armazenamento do tipo Chave/Valor conforme Figura 9.

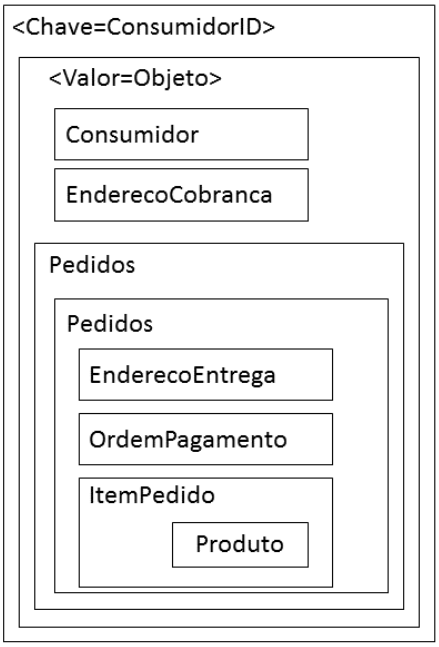

Figura 9 – Modelagem do Sistema de Gerenciamento Banco de dados NoSQL para consumidor e seus pedidos Fonte: (SADALAGE; FOWLER, 2012)

Porém, essa estrutura de dados usada pelos bancos de dados NoSQL valor-chave, coluna larga, gráfico ou documento, é diferente das usadas por padrão em bancos de dados relacionais, tornando algumas operações mais rápidas no NoSQL. Apesar de todas as vantagens de escalabilidade, flexibilidade e velocidade, o banco de dados NoSQL enfrenta grandes desafios como a consistência dos dados processados em transações distribuídas como as que existem em banco de dados relacional como PostgreSQL utilizado nesse trabalho.

2.4 PostgreSQL

Para preparar os dados para realização dos estudos de caso foi necessário a utilização de um banco de dados relacional e o escolhido, por conta de já haver um tipo de dados JSON, foi o PostgreSQL.

O PostgreSQL é um poderoso sistema de banco de dados objeto-relacional de código aberto, derivado do pacote POSTGRES escrito na Universidade da Califórnia em Berkeley. Com mais de uma década de desenvolvimento por trás, o PostgreSQL é, atualmente, o mais avançado banco de dados de código aberto disponível.

O projeto POSTGRES, liderado pelo Professor Michael Stonebraker,começou em 1986, atualmente, possui uma arquitetura comprovada que lhe confere uma reputação de confiabilidade, integridade de dados e correção. Ele é executado em todos os principais sistemas operacionais, incluindo Linux, UNIX, Mac OS X, Solaris, e Windows. Inclui a maioria dos tipos de dados SQL:2008, também suporta o armazenamento de objetos binários, incluindo imagens, sons ou vídeo. Possui interfaces de programação nativas para C / C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros.

Um banco de dados de classe empresarial, o PostgreSQL possui recursos sofisticados como o MVCC (*Multi-Version Concurrency Control*), *tablespaces*, replicação assíncrona, transações aninhadas, *backups on-line*, um planejador e otimizador de consultas sofisticado. É altamente escalável tanto na grande quantidade de dados que pode gerenciar quanto no número de usuários simultâneos que pode acomodar. Existem sistemas ativos do PostgreSQL em ambientes de produção que gerenciam mais de 4 *terabytes* de dados (POSTGRES, 2017).

Porém, para obter o processamento de Big Data provenientes da Internet das Coisas será necessário adotar um banco de dados que suporte além do grande volume de dados, fazer isso de forma paralela e que ofereça fácil escalabilidade (LI et al., 2012).

Para se escolher um banco de dados líder de mercado, o Gartner o define da seguinte forma. "Os líderes demonstram, geralmente, mais suporte para uma ampla gama de aplicações operacionais, tipos de dados e vários casos de uso. Esses fornecedores demonstraram a satisfação do cliente consistente com forte apoio ao cliente. Por isso, os líderes, geralmente, representam o menor risco para clientes nas áreas de desempenho, escalabilidade, confiabilidade e suporte. Como as demandas do mercado mudam com o tempo, então, os líderes demonstram forte visão em apoio não só das necessidades atuais do mercado, mas também das tendências emergentes"(GARTNER, 2015).

Para escolher um banco de dados que atenda a estas características de Big Data, o Gartner, considera um SGBD comercial definido como sistemas que também suportam múltiplas estruturas e tipos de dados, como XML, texto, JavaScript Object

Notation (JSON), áudio, imagem e vídeo. Eles devem incluir mecanismos para isolar os recursos de carga de trabalho e controlar vários parâmetros de acesso do usuário final dentro de instâncias gestão dos dados.

Diante das características apresentadas, foi utilizado o Quadrante Mágico da Gartner (2015) para selecionar o banco de dados NoSQL para realizar o estudo de caso da modelagem, conforme Figura 10.

Figura 10 – Quadrante de Gartner Fonte:(GARTNER, 2015)

Por ser um dos bancos de dados melhor ranqueado entre os lideres no Quadrante Mágico da Gartner, o MongoDB foi o escolhido.

2.5 MongoDB

MongoDB é uma aplicação de código aberto, de alta performance, alta disponibilidade e escalonamento automático. O seu desenvolvimento começou em outubro de 2007 pela 10gen. A primeira versão pública foi lançada em fevereiro de 2009 (ELIOT, 2010).

O MongoDB, a partir da versão 3.0, introduziu novos mecanismos de armazenamento que ampliam as capacidades do banco de dados, permitindo-lhe escolher as tecnologias ideais para as diferentes cargas de trabalho em sua organização, como por exemplo, o mecanismo MMAPv1 padrão. Uma versão melhorada do motor usado em versões anteriores do MongoDB e o novo motor de armazenamento *WiredTiger*, que proporciona melhor controle de concorrência, compressão nativa dos dados gerando benefícios significativos nas áreas de menores custos de armazenagem e melhorando o desempenho do *hardware* (MONGODB, 2015).

Executar vários mecanismos de armazenamento em uma implantação e alavancar a mesma linguagem de consulta, escalando métodos e ferramentas operacionais em todos eles para reduzir representativamente o desenvolvimento e complexidade operacional tornado ele um dos bancos de dados NoSQL líder de mercado.

No MongoDB, a menor unidade é um documento. Os documentos são armazenados em uma coleção, conforme Figura 11, que por sua vez compõem um banco de dados. Os documentos são análogos a linhas em uma tabela SQL, mas há uma grande diferença: nem todos os documentos precisam ter a mesma estrutura. Os documentos de uma coleção única não necessitam ter o mesmo conjunto de campos e tipo de dados de um campo pode diferir entre os documentos dentro de uma coleção. Outra característica do MongoDB é que os campos em um documento podem conter matrizes e / ou sub-documentos (às vezes chamados de documentos aninhados ou incorporados) conforme a Figura 12.

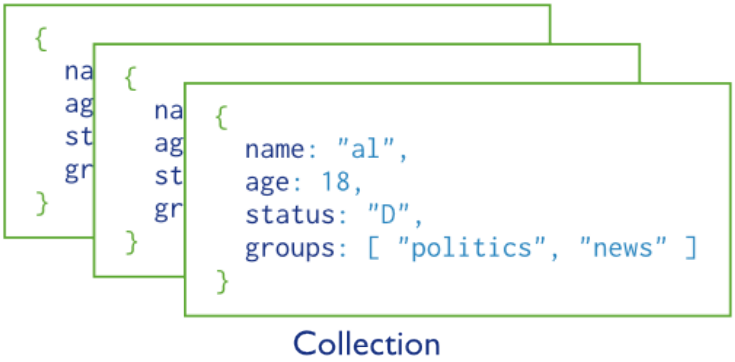

Figura 11 – Exemplo de uma Coleção Fonte: Mongo (2016)

Figura 12 – Exemplo de um Documento Fonte: (MONGODB, 2016)

O MongoDB fornece a capacidade de consulta em qualquer campo dentro de um documento. Fornece também um rico conjunto de opções de indexação para otimizar uma grande variedade de consultas, incluindo índices de texto, índices geoespaciais, índices compostos, índices dispersos, índices de tempo de vida, índices exclusivos e outros. Além disso, fornece a capacidade de analisar dados no local, sem que precise ser replicado para análise dedicada ou motores de busca.

Os documentos MongoDB são semelhantes aos objetos *JavaScript Object Notation* mais conhecidos como JSON. Os valores dos campos podem incluir outros documentos, *arrays* e *arrays* de documentos.

2.5.1 JSON

JSON é um formato de troca de dados simples de fácil escrita pelos humanos e de fácil interpretação pelas máquinas. Desenvolvido a partir de um subconjunto de linguagem de programação JavaScript (CROCKFORD, 2006).

JSON é construído em duas estruturas:

- Uma coleção de pares nome/valor, reconhecido como objeto.
- Uma lista ordenada de valores, reconhecido como uma matriz, vetor, lista ou sequência.

Um objeto é um conjunto não ordenado de pares nome/valor. Um objeto começa com "{"e termina com "}". Cada nome é seguido por ":"e o nome/valor pares são separados por ",".

Uma matriz é uma coleção ordenada de valores. Começa com "["e termina com "]". Os valores são separados por ",". Exemplo de uma estrutura JSON na Figura 13. Este formato não suporta campos binários, para isso o MongoDB utiliza o formato BSON.

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Figura 13 – Exemplo de um arquivo Json Fonte:(CROCKFORD, 2006)

2.5.2 BSON

BSON é uma representação binária de documentos JSON. BSON é um formato de serialização binária usado para armazenar documentos e fazer chamadas de procedimento remoto no MongoDB. O valor de um campo pode ser qualquer um dos tipos de dados BSON, incluindo outros documentos, matrizes e arrays de documentos conforme Figura 14.

O banco de dados MongoDB foi escolhido por possuir além de todas as características apresentada pela Gartner, mas também por atender algumas características do Big Data.

```
var mydoc = {
              _id: ObjectId("5099803df3f4948bd2f98391"),
              name: { first: "Alan", last: "Turing" },
              birth: new Date('Jun 23, 1912'),
              death: new Date('Jun 07, 1954'),
              contribs: [ "Turing machine", "Turing test", "Turingery" ],
              views : NumberLong(1250000)
            }
```

Figura 14 – Exemplo de um arquivo Bson Fonte:(MONGODB, 2016)

2.6 Big Data

Big Data é um termo amplamente utilizado para nomear conjuntos de dados muito grandes ou complexos. Pode-se considerar que o Big Data é uma quantidade enorme de informações nos servidores de bancos de dados e que funcionam dentro de diversos servidores de rede de computadores utilizando um sistema operacional de rede, interligados entre si.

As primeiras noções do Big Data foram limitadas a algumas organizações como Google, Yahoo, Microsoft e Organização Européia para Pesquisa Nuclear (CERN). No entanto, com os recentes desenvolvimentos em tecnologias como sensores, *hardware* de computador e da nuvem, o aumento de poder de armazenamento e processamento o custo cai rapidamente (ZASLAVSKY; PERERA; GEORGAKOPOULOS, 2012).

Entretanto, os principais desafios incluem: análise, captura, curadoria de dados, pesquisa, compartilhamento, armazenamento, transferência, visualização e informações sobre privacidade dos dados.

Para ajudar no processamento dos dados oriundos do Big Data, a Google criou um modelo de programação desenhado para processar grandes volumes de dados em paralelo, chamado MapReduce. O MapReduce é um modelo que foi proposto para o processamento de grandes conjuntos de dados, através da aplicação de tarefas, capazes de realizar este processamento de forma paralela, dividindo o trabalho em um conjunto de tarefas independentes. (Dean, Jeffrey;Ghemawat, 2004).

Composto por duas fases, esse processo se divide na fase de Map, onde ocorre sumarização dos dados, como por exemplo, uma contagem de uma determinada palavra presente em uma palavra menor, é nesta fase onde os elementos individuais são transformados e quebrados em pares do tipo Chave-Valor. Na fase de Reduce, a mesma recebe

como entrada a saída da fase Map, a partir disto, são realizados procedimentos de filtragem e ordenamento dos dados, que agrupam os pares semelhantes em conjuntos menores.

Na utilização da plataforma Big Data neste trabalho foi definido a utilização dos bancos de dados PostgreSQL e MongoDB, e se faz necessário entender algumas terminologias e conceitos.

2.6.1 PostgreSQL x MongoDB

Como o banco de dados de origem, onde foram preparados os dados foi o PostgreSQL, um banco de dados relacional utilizando a linguagem SQL e o banco de dados a receber as informações foi o MongoDB utilizando linguagem JavaScript, segue abaixo algumas terminologias conforme Figura 15.

Termos - SQL	Termos - MongoDB
Banco de Dados	Banco de Dados
Tabela	Coleção
Linha	Documento
Coluna	Campo
Índice	Índice
Junção de Tabelas	Vinculação de Documentos
Chave Primária	Chave Primária

Figura 15 – Terminologias SQL utilizado no PostgreSQL e do MongoDB

Assim como as terminologias, os comandos também são diferentes. Segue um comparativo dos comandos conforme Figura 16.

SQL Statement	Mongo Query Language Statement
CREATE TABLE USERS (a Number, b Number)	Implicit or use MongoDB::createCollection().
INSERT INTO USERS VALUES(1,1)	$db->users->insert(array("a" => 1, "b" => 1));
SELECT a,b FROM users	$db->users->find(array(), array("a" => 1, "b" => 1));
SELECT * FROM users WHERE age=33	$db->users->find(array("age" => 33));
SELECT a,b FROM users WHERE age=33	$db->users->find(array("age" => 33), array("a" => 1, "b" => 1));
SELECT a,b FROM users WHERE age=33 ORDER BY name	$db->users->find(array("age" => 33), array("a" => 1, "b" => 1))->sort(array("name" => 1));
SELECT * FROM users WHERE age>33	$db->users->find(array("age" => array('$gt' => 33)));
SELECT * FROM users WHERE age<33	$db->users->find(array("age" => array('$lt' => 33)));
SELECT * FROM users WHERE name LIKE "%Joe%"	$db->users->find(array("name" => new MongoRegex("/Joe/")));
SELECT * FROM users WHERE name LIKE "Joe%"	$db->users->find(array("name" => new MongoRegex("/^Joe/")));
SELECT * FROM users WHERE age>33 AND age<=40	$db->users->find(array("age" => array('$gt' => 33, '$lte' => 40)));
SELECT * FROM users ORDER BY name DESC	$db->users->find()->sort(array("name" => -1));
CREATE INDEX myindexname ON users(name)	$db->users->ensureIndex(array("name" => 1));
CREATE INDEX myindexname ON users(name,ts DESC)	$db->users->ensureIndex(array("name" => 1, "ts" => -1));
SELECT * FROM users WHERE a=1 and b='q'	$db->users->find(array("a" => 1, "b" => "q"));
SELECT * FROM users LIMIT 20, 10	$db->users->find()->limit(10)->skip(20);
SELECT * FROM users WHERE a=1 or b=2	$db->users->find(array('$or' => array(array("a" => 1), array("b" => 2))));
SELECT * FROM users LIMIT 1	$db->users->find()->limit(1);
EXPLAIN SELECT * FROM users WHERE z=3	$db->users->find(array("z" => 3))->explain()
SELECT DISTINCT last_name FROM users	$db->command(array("distinct" => "users", "key" => "last_name"));
SELECT COUNT(*y) FROM users	$db->users->count();
SELECT COUNT(*y) FROM users where AGE > 30	$db->users->find(array("age" => array('$gt' => 30)))->count();
SELECT COUNT(AGE) from users	$db->users->find(array("age" => array('$exists' => true)))->count();
UPDATE users SET a=1 WHERE b='q'	$db->users->update(array("b" => "q"), array('$set' => array("a" => 1)));
UPDATE users SET a=a+2 WHERE b='q'	$db->users->update(array("b" => "q"), array('$inc' => array("a" => 2)));
DELETE FROM users WHERE z="abc"	$db->users->remove(array("z" => "abc"));

Figura 16 – Comparação entre os comandos SQL utilizado pelo PostgreSQL e os utilizados pelo MongoDB

2.7 Resumo do Capítulo

Neste capítulo foi descrito os assuntos que fazem parte dos estudos de caso propostos. Big Data é o assunto principal deste trabalho, sendo muito importante compreender seu funcionamento e suas necessidades que serão sanadas com uma modelagem de banco de dados Não Relacional baseada em arquivo JSON, que será armazenado e gerenciando no banco de dados MongoDB. Esta modelagem, por si só, ajuda a sanar alguns problemas ocasionados pela internet das coisas.

A Modelagem de Banco de dados não relacional é muito utilizada para tratar grande massa de dados não estruturados. O banco de dados MongoDB é um banco de dados amplamente utilizado por ser um dos que melhor oferece suporte a modelagem Não Relacional segundo a Gartner. Para compreender melhor o banco de dados e modelagem não relacional foi necessário descrever, com detalhes, o banco de dados e os modelos relacionais.

CAPÍTULO 3

DESENVOLVIMENTO DO TRABALHO

Este capítulo se dedica a apresentar os dados utilizados nos estudos de caso, de onde eles se originaram, como foi a sua preparação antes de sua importação para o banco de dados com a modelagem aqui proposta, os *softwares* e *hardwares* utilizados para realização de cada estudos de caso. Também sera descrito o passo a passo de cada estudo de caso proposto por este trabalho.

3.1 Origem dos Dados

Os dados utilizados neste trabalho foi fornecido por duas fontes diferentes sendo elas o INMET (Instituto Nacional de Meteorologia) e do PGFA (Programa de Pós-graduação de Física Ambiental) da Universidade Federal de Mato Grosso. Os dados foram entregues em planilhas eletrônicas. Os dados utilizados nos estudos de caso proposto neste trabalho foram obtidos originalmente de sensores, que estão ou estiveram instalados em estações de meteorologia.

3.1.1 Dados do Instituto Nacional de Meteorologia

A coleta de dados é feita através de sensores para medição dos parâmetros meteorológicos a serem observados. As medidas tomadas, em intervalos de minuto a minuto, e integralizadas para no período de uma hora, para serem transmitidas, (INMET, 2011)

são: Temperatura Instantânea do Ar, Temperatura Máxima do Ar, Temperatura Mínima do Ar, Umidade Relativa Instantânea do Ar, Umidade Relativa Máxima do Ar, Umidade Relativa Mínima do Ar, Temperatura Instantânea do Ponto de Orvalho, Temperatura Máxima do Ponto de Orvalho, Temperatura Mínima do Ponto de Orvalho, Pressão Atmosférica Instantânea do Ar, Pressão Atmosférica Máxima do Ar, Pressão Atmosférica Mínima do Ar, Velocidade Instantânea do Vento, Direção do Vento, Intensidade da Rajada do Vento, Radiação Solar e Precipitação Acumulada no Período.

Estes dados são armazenados em uma memória não volátil que os mantém medidos por um período especificado. Os dados são captados e mantidos temporariamente em uma rede de estações meteorológicas que os disponibilizam para serem transmitidos, via satélite ou telefonia celular, para a sede do INMET.

Os sensores ficam instalados em uma estação meteorológica automática (EMA), que nada mais é do que um instrumento de coleta automática de informações ambientais locais (meteorológicas, hidrológicas ou oceânicas) composta pelos seguintes elementos: Sub-sistema de coleta de dados, Sub-sistema de controle e armazenamento, Sub-sistema de energia (painel solar e bateria) e Sub-sistema de comunicação.

Além dos sub-sistemas mencionados, a estação deve ser instalada em uma base física, numa área livre de obstruções naturais e prediais, situada em área gramada mínima de 14m por 18m, cercada por tela metálica (para evitar entrada de animais). Os sensores e demais instrumentos são fixados em um mastro metálico de 10 metros de altura, aterrado eletricamente (malha de cobre) e protegido por pára-raios. Os aparelhos para as medições de chuva (pluviômetro) e de radiação solar, bem como a antena para a comunicação, ficam situados fora do mastro, mas dentro do cercado conforme Figura 17.

Figura 17 – Estação Meteorológica Automática Fonte:(INMET, 2011)

3.1.2 Dados do Programa de Pós-Graduação da Física Ambiental da Universidade Federal de Mato Grosso

Assim como o INMET, os dados do Programa de Pós-Graduação da Física Ambiental (PGFA) da Universidade Federal de Mato Grosso (UFMT) foram coletados através de sensores que captaram dados micrometeorológicos da Torre micrometeorológica Cambarazal conforme Figura 18. Por se tratar de dados micrometeorológicos, os dados do PGFA foram coletados na ordem de segundos, o que gera uma grande quantidade de dados.

Figura 18 – Torre Micrometeorológica Cambarazal Fonte:(FEDERAL et al., 2009)

Devido a grande quantidade de dados, é necessário realizar um processo de limpeza antes da disponibilização dos dados para que se aproveite somente os dados úteis.

No PGFA além do processo de análise e buscas dos dados mais úteis, outro problema é o de armazenamento desses dados. Na grande maioria das vezes, cada pesquisador mantém os dados em planilhas eletrônicas no computador pessoal dificultando o compartilhamento da informação. Devido a esta dificuldade, as informações foram cedidas por um dos pesquisadores do PGFA. Porém, para que os dados pudessem ser armazenados para realização dos estudos de caso, fez-se necessário transformar as informações que estavam em planilha eletrônica para o formato de documento JSON.

As informações cedidas em formato de planilha eletrônica pelo INMET e pelo PGFA foram modelados no formato JSON.

3.2 Cenário

O Cenário utilizado para realizar os estudos de caso da modelagem é um conjunto de dados meteorológicos que, primeiramente, foram inseridos em um banco de dados relacional SQL Server 2016 instalado em uma máquina virtual com sistema

operacional Windows. Por conta dos poucos recursos e componentes em relação ao tipo de dados no formato Json, foi muito difícil gerar os arquivos na modelagem proposta.

Os arquivos que foram possíveis de gerar no SQL Server causou um grave problema de desempenho, fazendo com que fosse necessário escolher outro banco de dados. Após análise criteriosa, foi utilizado o banco de dados PostgreSQL instalado em uma máquina virtual com sistema operacional Windows e posteriormente os dados foram extraídos do banco de dados relacional para arquivos no formato JSON. Os arquivos físicos foram copiados para outra máquina virtual com sistema operacional Linux para serem importados no banco de dados Não Relacional MongoDB. Ambas as máquinas virtuais foram instaladas dentro da mesma máquina física compartilhando o mesmo *hardware*.

Como os dados fornecidos estavam em um banco de dados relacional, precisavam ser tratados antes de importar para o banco de dados Não Relacional,sendo necessário a realização de uma preparação dos dados.

3.2.1 Preparação dos Dados

Para realizar a preparação dos dados, foi escolhido o banco de dados PostgreSQL que possui compatibilidade com o tipo de dados JSON e, além disso, a credibilidade de ser um banco de dados robusto e amplamente utilizado pelo mercado. Após a escolha do banco de dados, o primeiro passo é criar as tabelas para receber os dados das planilhas eletrônicas de cada fonte de dados, onde foi incluído o atributo chamado "fonte", conforme Figuras 19 e 20 para identificar a origem dos dados.

```
-- Table: public.pgfaa

-- DROP TABLE public.pgfaa;

CREATE TABLE public.pgfaa
(
  fonte character(10),
  ano bigint,
  mes integer,
  dia integer,
  hora numeric(10,2),
  global numeric(10,2),
  tsolo1 numeric(10,2),
  tsolo3 numeric(10,2),
  tsolo7 numeric(10,2),
  tsolo15 numeric(10,2),
  tsolo30 numeric(10,2),
  temp3 numeric(10,2),
  velocv3 numeric(10,3),
  net numeric(10,3),
  ur3 numeric(10,3),
  ppt numeric(10,3)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE public.pgfaa
  OWNER TO postgres;
```

Figura 19 – Script para criação da tabela de dados para receber os dados originais do PGFA

```
    -- Table: public.inmet

-- DROP TABLE public.inmet;

CREATE TABLE public.inmet
(
  fonte character(10),
  ano bigint,
  mes integer,
  dia integer,
  hora integer,
  temp_inst numeric(10,2),
  temp_max numeric(10,2),
  temp_min numeric(10,2),
  umid_inst numeric(10,2),
  umid_max numeric(10,2),
  umid_min numeric(10,2),
  pto_orvalho_inst numeric(10,2),
  pto_orvalho_max numeric(10,2),
  pto_orvalho_min numeric(10,2),
  pressao numeric(10,2),
  pressao_max numeric(10,2),
  pressao_min numeric(10,2),
  vento_direcao numeric(10,2),
  vento_vel numeric(10,2),
  vento_rajada numeric(10,2),
  radiacao numeric(10,3),
  precipitacao numeric(10,2)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE public.inmet
  OWNER TO postgres;
```

Figura 20 – Script para criação da tabela de dados para receber os dados originais do INMET

Feito isso, foi necessário realizar a importação dos dados de cada fonte de dados através da funcionalidade de importação da ferramenta de interface gráfica PgAdmin conforme Figura 21.

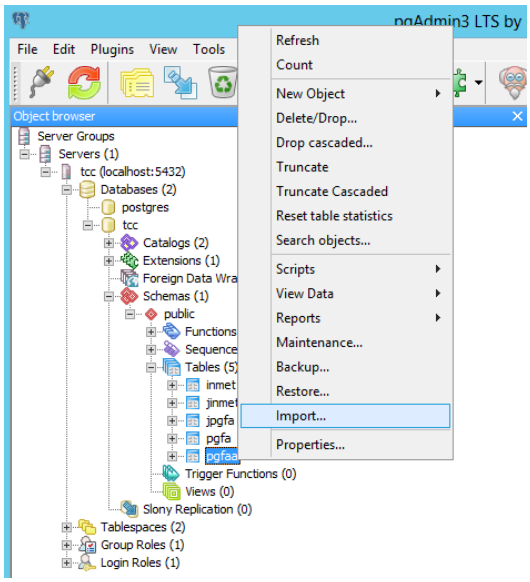

Figura 21 – Tela mostrando a funcionalidade de importação da ferramenta de interface gráfica PgAdmin

Para que a funcionalidade funcione corretamente, é preciso realizar algumas verificações como o formato de data, campos nulos e linhas quebradas que precisaram ser corrigidas antes da realização da importação para o banco de dados.

Após a importação dos dados de origem nas tabelas criadas, conforme Figura 22, foram realizados varias tentativas de exportação direto das tabelas de origem para os arquivos no formato JSON, que resultaram em scripts complexos e de execução muito lenta chegando a gerar um arquivo de 10 mil registros por hora. Observando esta dificuldade, foi necessário otimizar o processo e para isso houve a necessidade de criar tabelas auxiliares para ajudar na transformação do formato dos dados originais para o formato JSON no próprio banco de dados relacional. Sendo assim, então, foram criados as tabelas auxiliares para cada fonte de dados incluindo em sua estrutura um atributo chamado "id"para identificar cada registro e auxiliar no momento da exportação destes dados. Também foi criado um atributo do tipo JSON chamado "info"para guardar todos os outros atributos de cada registro da tabela de origem. As Figura 23 e 24 representam os scripts de criação de cada tabela auxiliar.

Output pane [Query 1]

Data Output | Explain | Messages | History

	fonte character(10)	ano bigint	mes integer	dia integer	hora numeric(10,2)	global numeric(10,2)	tsolo1 numeric(10,2)	tsolo3 numeric(10,2)	tsolo7 numeric(10,2)	tsolo15 numeric(10,2)	tsolo30 numeric(10,2)	temp3 numeric(10,2)	velocv3 numeric(10,3)	net numeric(10,3)	ur3 numeric(10,3)	ppt numeric(10,3)
1	PGFA	2008	1	1	0.00	0.00	28.04	29.07	29.88	30.25	29.61	22.77	97.700	1.478		-10.870
2	PGFA	2008	1	1	15.00	0.00	27.99	29.03	29.86	30.24	29.62	22.72	97.700	1.487		-6.663
3	PGFA	2008	1	1	30.00	0.00	27.95	29.00	29.83	30.23	29.62	22.61	97.800	1.614		-5.837
4	PGFA	2008	1	1	45.00	0.00	27.93	28.97	29.82	30.23	29.64	22.57	97.700	1.056		-6.000
5	PGFA	2008	1	1	100.00	0.00	27.95	28.93	29.78	30.21	29.64	22.53	97.700	1.150		-5.620
6	PGFA	2008	1	1	115.00	0.00	27.95	28.90	29.75	30.19	29.64	22.56	97.700	0.593		-10.350
7	PGFA	2008	1	1	130.00	0.00	27.94	28.88	29.74	30.19	29.65	22.57	97.600	0.000		-6.584
8	PGFA	2008	1	1	145.00	0.00	27.91	28.83	29.69	30.16	29.64	22.60	97.700	0.000		-6.122
9	PGFA	2008	1	1	200.00	0.00	27.85	28.78	29.64	30.12	29.63	22.61	97.800	0.000		-6.109
10	PGFA	2008	1	1	215.00	0.00	27.81	28.74	29.61	30.10	29.63	22.61	97.800	0.000		-7.980
11	PGFA	2008	1	1	230.00	0.00	27.80	28.71	29.58	30.09	29.63	22.59	97.900	0.000		-8.980
12	PGFA	2008	1	1	245.00	0.00	27.81	28.68	29.54	30.09	29.64	22.55	97.900	0.000		-9.100
13	PGFA	2008	1	1	300.00	0.00	27.75	28.64	29.48	30.07	29.64	22.53	98.000	0.399		-8.930
14	PGFA	2008	1	1	315.00	0.00	27.69	28.58	29.42	30.04	29.63	22.57	97.900	1.547		-9.510
15	PGFA	2008	1	1	330.00	0.00	27.61	28.50	29.33	29.99	29.60	22.69	98.100	1.127		-10.460
16	PGFA	2008	1	1	345.00	0.00	27.59	28.43	29.26	29.93	29.55	22.69	98.100	0.573		-10.810

Figura 22 – Tela mostrando alguns dados importados no PostgreSQL

```
-- Table: public.jpgfa

-- DROP TABLE public.jpgfa;

CREATE TABLE public.jpgfa
(
  id integer NOT NULL DEFAULT nextval('jpgfa_id_seq'::regclass),
  info json NOT NULL,
  CONSTRAINT jpgfa_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE public.jpgfa
  OWNER TO postgres;
```

Figura 23 – Script de criação de tabela auxiliar para transformação do formato dos dados da PGFA

```
-- Table: public.jinmet

-- DROP TABLE public.jinmet;

CREATE TABLE public.jinmet
(
  id integer NOT NULL,
  info json,
  CONSTRAINT jinmet_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE public.jinmet
  OWNER TO postgres;
```

Figura 24 – Script de criação de tabela auxiliar para transformação do formato dos dados do INMET

Em seguida, os dados foram transferidos das tabelas onde estão os dados originais para as tabelas auxiliares e para isso foi desenvolvido um script para cada fonte de dados conforme as Figuras 25 e 26.

```
insert into jpgfa
select row_number() over() as id, row_to_json(a)
from (
select
row_to_json(keys.*) as keys,
row_to_json(dados.*) as dados
from
(SELECT rtrim(fonte) as fonte, ano, mes, dia, hora
    FROM pgfaa
    )  keys
inner join
(SELECT ano, mes, dia, hora, global, tsolo1, tsolo3,
tsolo7, tsolo15, tsolo30, temp3, velocv3, net, ur3, ppt
    FROM public.pgfaa
    ) dados
on keys.ano = dados.ano and keys.mes = dados.mes
and keys.dia = dados.dia and keys.hora = dados.hora
) a
```

Figura 25 – Script de inserção de dados na tabela auxiliar do PGFA

```
insert into jinmet
select row_number() over() as id, row_to_json(a)
from (
select
row_to_json(keys.*) as keys,
row_to_json(dados.*) as dados
from
(SELECT rtrim(fonte) as fonte, ano, mes, dia, hora
    FROM public.inmet
    )  keys
inner join
(SELECT ano, mes, dia, hora, temp_inst, temp_max,
    temp_min, umid_inst, umid_max, umid_min,
    pto_orvalho_inst, pto_orvalho_max, pto_orvalho_min,
    pressao, pressao_max, pressao_min, vento_direcao,
    vento_vel, vento_rajada, radiacao, precipitacao
    FROM public.inmet
    ) dados
on keys.ano = dados.ano and keys.mes = dados.mes and
keys.dia = dados.dia and keys.hora = dados.hora
) a
```

Figura 26 – Script de inserção de dados na tabela auxiliar do INMET

Após transferir os dados da tabela de origem para a tabela com o formato JSON, os dados se apresentam conforme Figura 27.

Data Output | Explain | Messages | History

	id integer	info json
1	1	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":0},"dados":{"ano":2000,"mes":1,"dia":1,"hora":0,"temp inst":27.50,"temp max":29.20,"temp min":27.50,"
2	2	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":0},"dados":{"ano":2000,"mes":1,"dia":1,"hora":0,"temp inst":27.50,"temp max":29.20,"temp min":27.50,"
3	3	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":0},"dados":{"ano":2000,"mes":1,"dia":1,"hora":0,"temp inst":27.50,"temp max":29.20,"temp min":27.50,"
4	4	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":0},"dados":{"ano":2000,"mes":1,"dia":1,"hora":0,"temp inst":27.50,"temp max":29.20,"temp min":27.50,"
5	5	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":1},"dados":{"ano":2000,"mes":1,"dia":1,"hora":1,"temp inst":26.70,"temp max":27.40,"temp min":26.70,"
6	6	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":1},"dados":{"ano":2000,"mes":1,"dia":1,"hora":1,"temp inst":26.70,"temp max":27.40,"temp min":26.70,"
7	7	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":1},"dados":{"ano":2000,"mes":1,"dia":1,"hora":1,"temp inst":26.70,"temp max":27.40,"temp min":26.70,"
8	8	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":1},"dados":{"ano":2000,"mes":1,"dia":1,"hora":1,"temp inst":26.70,"temp max":27.40,"temp min":26.70,"
9	9	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":2},"dados":{"ano":2000,"mes":1,"dia":1,"hora":2,"temp inst":26.50,"temp max":26.80,"temp min":26.50,"
10	10	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":2},"dados":{"ano":2000,"mes":1,"dia":1,"hora":2,"temp inst":26.50,"temp max":26.80,"temp min":26.50,"
11	11	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":2},"dados":{"ano":2000,"mes":1,"dia":1,"hora":2,"temp inst":26.50,"temp max":26.80,"temp min":26.50,"
12	12	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":2},"dados":{"ano":2000,"mes":1,"dia":1,"hora":2,"temp inst":26.50,"temp max":26.80,"temp min":26.50,"
13	13	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":3},"dados":{"ano":2000,"mes":1,"dia":1,"hora":3,"temp inst":26.60,"temp max":27.00,"temp min":26.50,"
14	14	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":3},"dados":{"ano":2000,"mes":1,"dia":1,"hora":3,"temp inst":26.60,"temp max":27.00,"temp min":26.50,"
15	15	{"keys":{"fonte":"INMET","ano":2000,"mes":1,"dia":1,"hora":3},"dados":{"ano":2000,"mes":1,"dia":1,"hora":3,"temp inst":26.60,"temp max":27.00,"temp min":26.50,"

Figura 27 – Tela mostrando alguns dados importados no banco de dados PostgreSQL com formato JSON

Depois dos dados transferidos para as tabelas auxiliares, foi possível gerar os arquivos em formato JSON, desta vez, gerando aproximadamente 50 mil registros por arquivo no tempo médio de 4,5 segundos por arquivo conforme Figura 28.

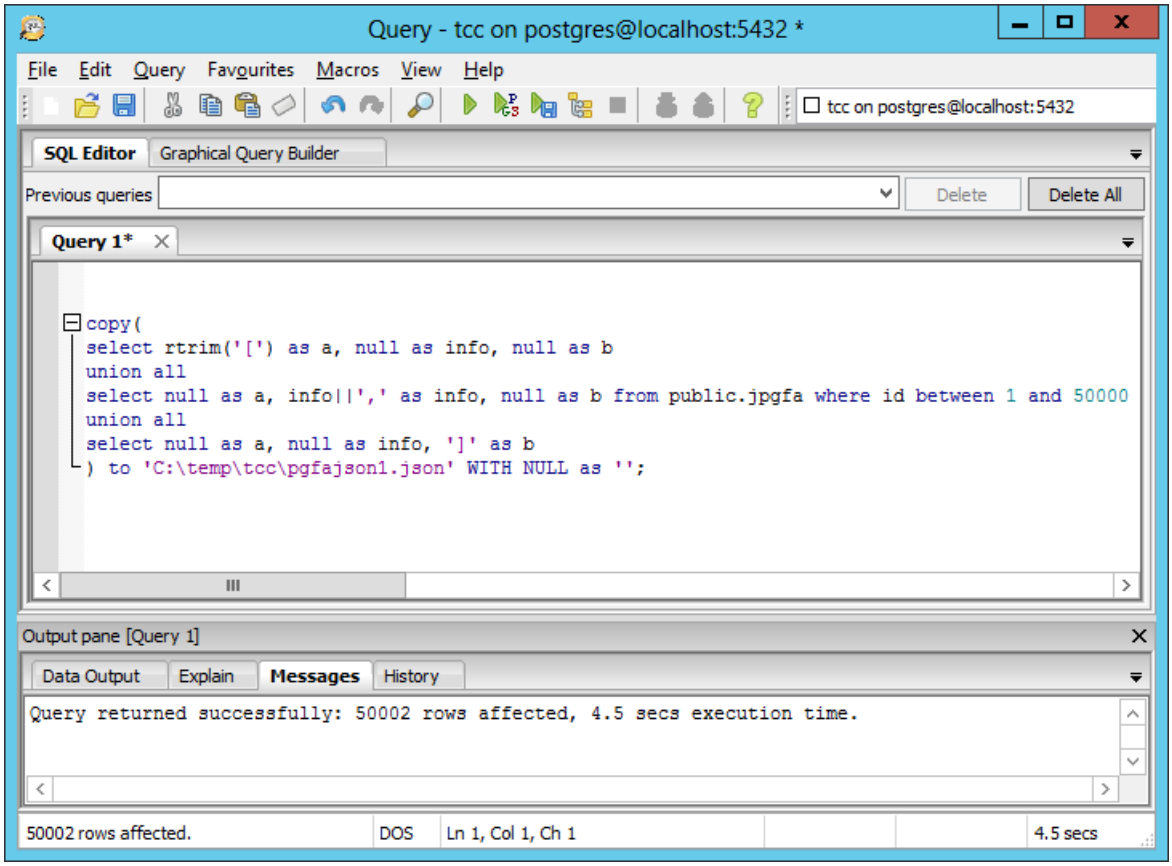

Figura 28 – Tela mostrando script de geração dos arquivos JSON e o tempo de execução do script

Os arquivos devem ser gerados na modelagem aqui proposta, que será detalhada neste capítulo, e para gerar os arquivos nesta modelagem foram utilizados alguns equipamentos virtuais e físicos.

3.2.2 Equipamentos Utilizados

Para realizar a inclusão das planilhas eletrônicas na base de dados foram necessários a criação de servidores virtualizados no sistema de virtualização Oracle VirtualBox versão 5.1.10. A máquina hospedeira possui processador Intel Core i7-4500U, 16GB de memória RAM com sistema operacional *Windows* 10.

Foram criadas duas máquinas virtuais (VM) para o desenvolvimento deste trabalho a fim de que as configurações do SGBD de conversão dos dados não afete as configurações do SGBD de recepção dos dados por possíveis erros decorrentes de instalação ou parametrizações.

Todas as máquinas virtualizadas tem a mesmas configurações de *hardware* sendo elas: 1 núcleo de Processador Intel Core i7-4500, Disco Rígido 60GB, 8GB de Memória RAM e Placa de Rede em modo NAT.

Na máquina que foi construída para realizar a conversão dos dados, foi instalado o banco de dados PostgreSQL versão 9.6.1 64bits e o SGBD PgAdmin3 versão 1.23.0a de código aberto e o sistema operacional foi o *Windows Server* 2012 64bits.

Na máquina que foi construída para realizar a recepção dos dados, foi instalado o banco de dados MongoDB versão 3.2.8 e a ferramenta de interface gráfica Robomongo 0.9.0-RC9, principalmente, por ser nativo [1] do MongoDB e o sistema operacional Linux Ubuntu 16.04 LTS 64-bit.

3.3 Estudo de Caso

Neste trabalho foram desenvolvidos três estudos de caso: uma modelagem dos dados em JSON para extrair os dados do banco de dados relacional em formato de documento para ser utilizado no segundo estudo de caso, que é popular o banco de dados NoSQL com os documentos gerados pela modelagem e o terceiro estudo de caso é realizar consultas para verificação de performance do banco de dados com massa de aproximadamente 1 milhão de registros.

3.3.1 Estudo de Caso 1: Modelagem dos dados

Para validar o estudo de caso, foi necessário realizar a modelagem através de implementação de regras em *JavaScript*, que é trabalhado em uma camada anterior ao banco de dados. Na verdade, a modelagem é um documento JSON que posteriormente é persistido no banco de dados.

A primeira fonte de dados é a do Programa de Pós-Graduação em Física Ambiental da Universidade Federal de Mato Grosso que compreende a análise de solo. A segunda fonte de dados é do Instituto Nacional de Meteorologia. Utilizando este cenário foi desenvolvido uma modelagem não relacional para atender os dados compreendidos como dados da Internet das coisas.

Os dados disponibilizados em planilhas eletrônicas, primeiramente, foram importados para o banco de dados relacional PostgreSQL, que foi escolhido por possuir funções nativas do banco de dados, para tratamento de documentos JSON. Utilizando estas funções nativas do PostgreSQL, foi desenvolvido um script para gerar os documentos JSON para gerar os documentos de cada fonte de dado conforme Figura 28.

Como no cenário proposto grande parte dos registros estão relacionados a informações temporais e vem de fontes de dados diferentes, a modelagem foi construída em formato JSON com um conjunto de regras em javascript.

[1] referencia sobre a palavra nativo http://ausland.com.br/erp-diferenca-entre-software-integrado-embarcado-e-nativo/

A ideia da modelagem é ser o mais flexível possível, sendo assim, não é necessário a criação de tabelas ou no caso do MongoDB coleções antes da inserção dos dados. Caso a coleção não exista, o próprio MongoDB se encarrega de criar antes da inserção dos documentos. Os dados serão armazenados conforme a modelagem em JSON, que constitui em armazenar um documento com dois documentos internos ou também chamados de sub-documentos.

O primeiro documento interno possui um conjunto de dados denominados KEYS, onde ficam no mínimo um campo que será utilizado como chave podendo possuir mais campos chaves. Estes campos chaves devem ser campos que existam também no segundo documento interno.

O segundo documento interno possui um conjunto de dados denominado DADOS, onde ficam os atributos do documento podendo incluir qualquer tipo de dados conforme Figura 29.

```
{
    keys: {

    },
    dados: {

    }
}
```

Figura 29 – Exemplo da modelagem vazia

Porém para o preenchimento correto da modelagem é importante seguir algumas regras obrigatórias.

Regras

Primeiramente, é necessário que o documento possua os dois conjuntos de dados KEYS e DADOS citados acima.

No conjunto KEYS, é necessário que se informe no mínimo um campo que possa ser utilizado como chave ou um conjunto de campos e que possa facilitar a busca pelo documento.

No conjunto DADOS, pode possuir qualquer tipo de dados inclusive os dados incluídos no conjunto KEYS.

No cenário proposto, foram criados documentos com o primeiro conjunto de dados possuindo cinco campos iniciais essenciais sendo eles: fonte, ano, mês, dia e hora. No segundo conjunto de dados, foi colocado os dados temporais extraídos dos dados dos

sensores das estações meteorológicas disponibilizados pelo INMET e PGFA. Dados estes que já foram processados.

Os dados foram disponibilizados no formato de documento de texto e para realizar o estudo de caso foi necessário transformar do formato texto para o formato JSON. Formato este utilizado para atender a modelagem proposta.

Utilizando os dados disponibilizados no contexto deste trabalho, ambos documentos possuem os mesmos atributos no conjunto KEYS, que é o campo necessário para sua localização e identificação dentro do banco de dados. Já, o segundo conjunto de dados é apresentado com os atributos diferentes. Os documentos possuem no conjunto DADOS, cada um os seus atributos disponibilizados por cada fonte fornecedora dos dados meteorológicos. Após a geração dos documentos é possível realizar a população da base de dados no MongoDB.

3.3.2 Estudo de Caso 2: Popular base de dados

Para realizar a população dos dados, o importante, inicialmente, é realizar uma busca no banco de dados com os dados do conjunto KEYS do documento a ser inserido.

No caso da busca encontrar no banco de dados um documento com exatamente os mesmos campos e valores do conjunto KEYS do documento a ser inserido, a regra atualizará o documento do banco de dados com os dados do documento a ser inserido.

No caso da busca encontrar no banco de dados um documento com os mesmos campos, porém qualquer valor diferente do conjunto KEYS do documento a ser inserido, a regra cria um novo documento no banco de dados com os atributos contidos no documento a ser inserido.

No caso da busca não encontrar nenhum documento no banco de dados com o conjunto KEYS que contenham os mesmos campos que o conjunto KEYS do documento a ser inserido, a regra cria um novo documento no banco de dados com os atributos contidos no documento a ser inserido.

As regras citadas, são representadas pela linha de comando representada na Figura 30.

```
db.teste.update({ keys: input.keys },
{ $set: { keys: input.keys, dados: input.dados } },
{ upsert: true })
```

Figura 30 – Script para realizar a população dos documentos JSON na base de dados.

O trecho do comando: db.teste.update identifica o banco de dados, a coleção a ser atualizada ou inserida. o comando update em conjunto com outros parâmetros realiza uma busca no banco, atualiza ou insere dados na coleção escolhida.

O trecho do comando: { keys: input.keys } representa a pesquisa pelos documentos existentes.

O trecho do comando: { $set: { keys: input.keys, dados: input.dados} } aloca os dados a serem atualizados ou inseridos.

O trecho do comando: { upsert: true } é o parâmetro que permite o comando update inserir um novo documento caso o documento não exista no banco de dados.

Após a realização da população dos dados na base de dados MongoDB, são realizados verificações de performance.

3.3.3 Estudo de Caso 3: Verificação de performance

Para realizar a verificação de performance dos bancos de dados, serão utilizados 2 tipos de consulta em cada banco de dados, uma simples e uma complexa. Cada consulta será executada com 4 limites de registros, sendo uma com 1 mil registros, uma com 10 mil registros, uma com 50 mil registros e a última com 100 mil registros. As consultas serão repetidas 10 vezes cada uma e o resultado será a média aritmética simples dos resultados de cada consulta exclusiva.

Exemplo, a consulta simples será executada 10 vezes com 1 mil registros, serão somados os tempos dos resultados das 10 consultas e posteriormente dividido por 10, o resultado final é o tempo médio de execução da consulta simples com mil registros.

Para as operações realizadas no banco de dados PostgreSQL, o código da consulta foi estruturado da seguinte forma:

- Consulta simples com 1 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes limit 1000;

- Consulta simples com 10 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes limit 10000;

- Consulta simples com 50 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes limit 50000;

- Consulta simples com 100 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes limit 100000;

- Consulta complexa com 1 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes where p.ano < 2015 and p.ano > 1995 and p.mes != 1 and p.mes != 12 limit 1000;

- Consulta complexa com 10 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes where p.ano < 2015 and p.ano > 1995 and p.mes != 1 and p.mes != 12 limit 10000;

- Consulta complexa com 50 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes where p.ano < 2015 and p.ano > 1995 and p.mes != 1 and p.mes != 12 limit 50000;

- Consulta complexa com 100 mil registros

  SELECT * FROM public.pgfaa p inner join public.inmet i on p.mes = i.mes where p.ano < 2015 and p.ano > 1995 and p.mes != 1 and p.mes != 12 limit 100000;

Para as operações realizadas no banco de dados MongoDB o código da consulta foi estruturado da seguinte forma:

- Consulta simples com 1 mil registros

  db.getCollection('tcc').find({}).limit(1000);

- Consulta simples com 10 mil registros

  db.getCollection('tcc').find({}).limit(10000);

- Consulta simples com 50 mil registros

  db.getCollection('tcc').find({}).limit(50000);

- Consulta simples com 100 mil registros

  db.getCollection('tcc').find({}).limit(100000);

- Consulta complexa com 1 mil registros

  db.getCollection('tcc').find({"keys.ano":{$lte:2015},"keys.ano":{$lte:1995},

  "dados.mes":{$ne:1},"dados.mes":{$ne:12}}).limit(1000);

- Consulta complexa com 10 mil registros

  db.getCollection('tcc').find({"keys.ano":{$lte:2015},"keys.ano":{$lte:1995},
  "dados.mes":{$ne:1},"dados.mes":{$ne:12}}).limit(10000);

- Consulta complexa com 50 mil registros

  db.getCollection('tcc').find({"keys.ano":{$lte:2015},"keys.ano":{$lte:1995},
  "dados.mes":{$ne:1},"dados.mes":{$ne:12}}).limit(50000);

- Consulta complexa com 100 mil registros

  db.getCollection('tcc').find({"keys.ano":{$lte:2015},"keys.ano":{$lte:1995},
  "dados.mes":{$ne:1},"dados.mes":{$ne:12}}).limit(100000);

3.4 Resumo do Capítulo

Neste capítulo foi descrito a origem dos dados, a preparação desses dados, em qual cenário serão realizados cada um dos estudos de caso e foi descrito com detalhes a configuração das máquinas, sistemas operacionais e banco de dados nelas instalados.

CAPÍTULO 4

RESULTADOS E DISCUSSÕES

A seguir, serão apresentados os resultados de todos os estudos de caso da modelagem dos dados, população da base de dados e verificação de performance.

4.1 Resultado do Estudo de Caso 1: Modelagem dos dados

Utilizando a modelagem proposta após executar o script de geração dos documentos em formato JSON, cada arquivo JSON possui vários documentos e cada um deles preenchidos com os dados do contexto que se apresenta conforme os exemplos representados pela Figura 31 com os dados disponibilizados pelo INMET, e na figura 32 com os dados disponibilizados pelo PGFA. Estes são exemplos de como os documentos ficam com a modelagem proposta, assim os documentos podem ser utilizados para realizar a população na base de dados MongoDB.

```
{
    "keys": {
        "fonte":"INMET"
        "ano": 2016,
        "mes": 7,
        "dia": 26,
        "hora": 2
    },
    "dados": {
    "codigo_estacao" : "A901",
    "data" : "26/07/2016",
    "hora" : 2,
    "temp_inst" : 27.9,
    "temp_max" : 29,
    "temp_min" : 27.9,
    "umid_inst" : 30,
    "umid_max" : 30,
    "umid_min" : 28,
    "pto_orvalho_inst" : 8.6,
    "pto_orvalho_max" : 8.7,
    "pto_orvalho_min" : 8.3,
    "pressao" : 984.8,
    "pressao_max" : 984.9,
    "pressao_min" : 984.8,
    "vento_direcao" : 1.5,
    "vento_vel" : 27,
    "vento_rajada" : 3.9,
    "radiacao" : -2.95,
    "precipitacao" : 0
    }
}
```


Figura 31 – Exemplo da modelagem preenchida com os dados da INMET Fonte: codigo Json com os dados do INMET.

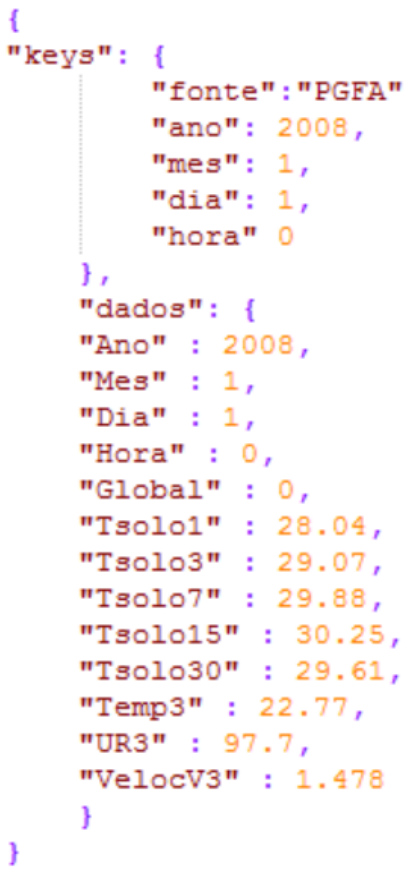

Figura 32 – Exemplo da modelagem preenchida com os dados da PGFA Fonte: codigo Json com os dados do PGFA.

4.2 Resultado do Estudo de Caso 2: Popular base de dados

Após a população dos documentos na coleção, é possível verificar se os dados foram inseridos corretamente, executando na interface gráfica RoboMongo o seguinte script: "db.getCollection('nome_coleção').find({})", conforme a Figura 33 e verificar como ficou cada documento, abrindo o registro diretamente no RoboMongo conforme Figuras 34 com o resultado da inserção dos dados da PGFA e Figura 35 com o resultado da inserção dos dados do INMET.

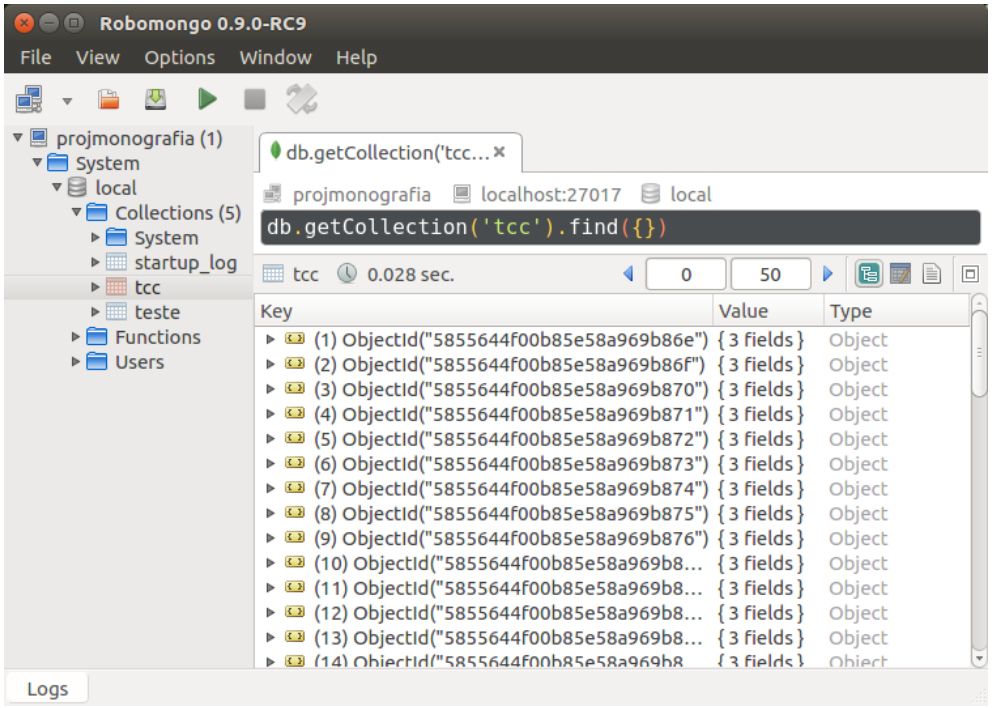

Figura 33 – Exemplos de documentos inseridos no banco de dados MongoDB.

_id	ObjectId("57f94b015635a93c7a3ce797")
keys	{ 5 fields }
fonte	PGFA
ano	2008.0
mes	1.0
dia	1.0
hora	0.0
dados	{ 13 fields }
Ano	2008.0
Mes	1.0
Dia	1.0
Hora	0.0
Global	0.0
Tsolo1	28.04
Tsolo3	29.07
Tsolo7	29.88
Tsolo15	30.25
Tsolo30	29.61
Temp3	22.77
UR3	97.7
VelocV3	1.478

Figura 34 – Dados da PGFA inseridos no banco de dados Fonte:tela com o resultado da inserção dos dados do PGFA.

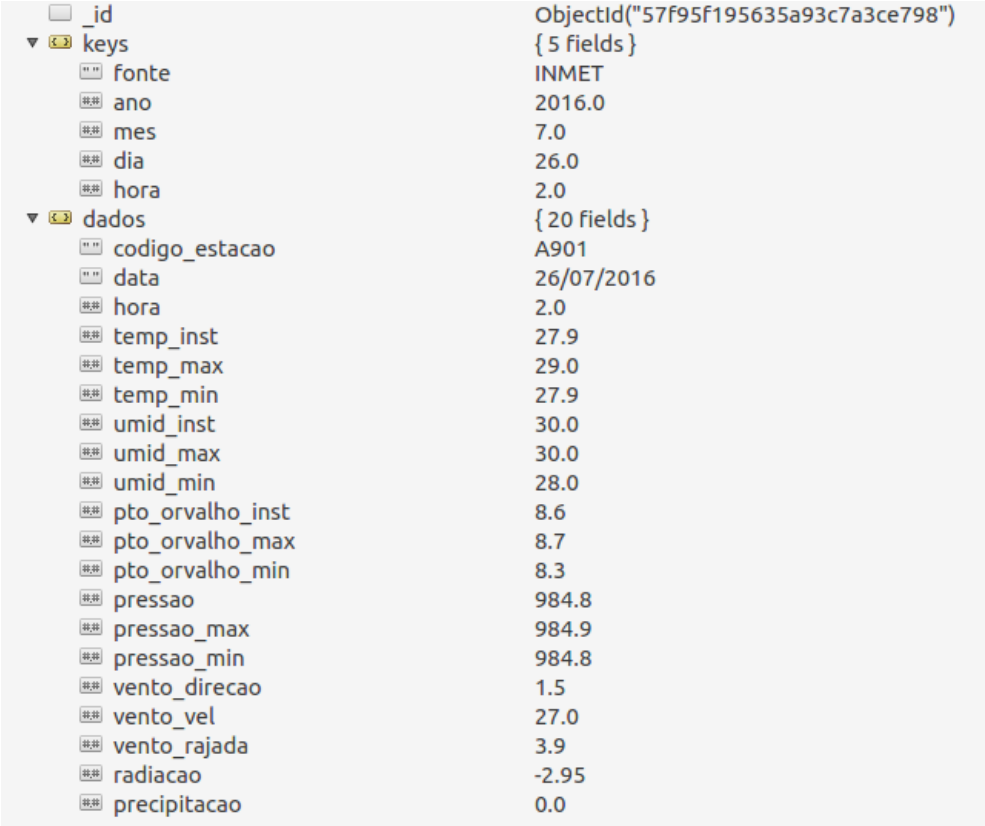

Figura 35 – Dados da INMET inseridos no banco de dados Fonte:tela com o resultado da inserção dos dados do INMET.

4.3 Resultado do Estudo de Caso 3: Verificação de performance

Após verificar que os dados foram gravados no banco de dados MongoDB, foram realizados os testes de performance. Os testes foram realizados conforme o item 3.3.3 desse trabalho, porém o banco de dados MongoDB não conseguiu concluir a apresentação dos dados ao selecionar 100 mil registros, tanto para consulta simples como para consulta complexa conforme resultados apresentados na Figura36. A quantidade máxima de registros apresentadas pelo banco de dados MongoDB foi de 50 mil registros. Ao ultrapassar a quantidade de 50 mil registros durante as consultas no MongoDB, a interface gráfica Robomongo fechava após alguns segundos de execução.

Banco de Dados	Simples				Complexo			
	1.000	10.000	50.000	100.000	1.000	10.000	50.000	100.000
PostgreSQL	00:00:00,648	00:00:06,550	00:00:19,760	00:00:47,700	00:06:23,500	00:04:58,400	00:00:19,760	00:00:56,220
MongoDB	00:00:00,492	00:00:01,524	00:00:03,347		00:00:01,114	00:00:02,004	00:00:03,434	

Figura 36 – Tabela com o resultado dos tempos médios das consultas dos banco de dados PostgreSQL e MongoDB.

Mesmo o banco de dados MongoDB não conseguindo apresentar os resultados das consultas de 100 mil registros para as consultas simples, alcançando o limite de 50 mil registros, o banco de dados MongoDB quase sempre apresentou resultados próximo de 0 segundos enquanto o PostgreSQL aumentou o tempo de resposta a cada quantidade de registros que foi consultada conforme o gráfico da Figura 37 atingindo uma média superior a 50 segundos com a consulta de 100 mil registros.

Figura 37 – Gráfico com o resultado dos tempos médios das consultas simples realizados no banco de dados PostgreSQL e MongoDB.

Assim, como nas consultas simples, o banco de dados MongoDB sofreu o mesmo problema nas consultas complexas, não marcando tempo com a consulta de 100 mil registros e registrando novamente a marca limite dos 50 mil registros. Repetindo o que aconteceu nas consultas simples, o banco de dados MongoDB registrou para todas as consultas um tempo médio abaixo de 0 segundos, já ao contrario das consultas simples, o banco de dados PostgreSQL apresentou uma resposta mais rápida para cada consulta que era feita porém, sempre mantendo uma média muito superior ao resultado apresentado

pelo MongoDB. O resultado mais baixo apresentado pelo banco de dados PostgreSQL foi a marca de aproximadamente 40 segundos conforme Figura 38 voltando a aumentar o tempo de consulta quando consultado 100 mil registros.

Figura 38 – Gráfico com o resultado dos tempos médios das consultas complexas realizados no banco de dados PostgreSQL e MongoDB.

A fim de entender esse problema nas consultas de 100 mil registros encontrados na execução realizada na interface gráfica Robomongo foi realizado uma pesquisa em fórum na internet, e através do site Stack Overflow, que é a maior comunidade on-line para programadores compartilhem seus conhecimentos e promover suas carreiras, fundado em 2008 com mais de 6 milhões de programadores registrados e que recebe mais de 40 milhões de visitantes por mês, foi encontrado um caso semelhante.

Usando Mono 3.2.1 no Ubuntu 12.04 em um projeto CSharp que se conecta ao MongoDB e tenta consultar e atualizar os documentos executando 4 scripts diferentes esperando receber 10 mil documentos de resultado, sendo que em um deles não apresenta nenhum resultado. Caso esse consultado no dia 06/02/2017 e que pode ser verificado através do seguinte link: "http://stackoverflow.com/questions/19067189/strange-performance-test-results-with-different-write-concerns-in-mongodb-c-sh?rq=1".

CAPÍTULO 5

CONCLUSÕES

Com este trabalho realizou-se a investigação dos modelos de banco de dados não relacional, conhecendo seus conceitos e suas diferenças para outros padrões, atualmente, existentes. Dos modelos investigados, o modelo orientado a documento foi o escolhido por ser mais flexível, escalável, adaptável, aceitar dados textuais e binários em vários formatos diferentes.

Após esta escolha, foi possível investigar e testar o armazenamento de dados oriundos da Internet das Coisas. Os dados foram previamente preparados em um banco de dados relacional e posteriormente foi facilmente armazenado no banco de dados não relacional permitindo dar sequência ao trabalho.

Feito os testes de armazenamento, foram desenvolvidos os estudos de caso utilizando dados sensoriais meteorológicos do Instituto Nacional de Meteorologia e do programa de pós-graduação da Física Ambiental da Universidade Federal de Mato Grosso que sofreram uma prévia preparação.

Com os dados preparados, foram realizados a execução, avaliação e homologação de cada estudo de caso. Estudo de caso 1 - Modelagem dos dados: foi realizado a modelagem dos dados através do modelo orientado a documentos desenvolvido em linguagem JavaScript conforme regras estabelecidas no item 3.3.1 deste trabalho e gravados no formato de arquivo JSON.

Estudo de caso 2 - Popular base de dados: com os documentos salvos em arquivo foi possível importar os mesmos para dentro do banco de dados seguindo as regras estabelecidas no item 3.3.2 deste trabalho.

Estudo de caso 3 - Verificação de performance: foi realizado testes de performance através de várias consultas em quantidade diferentes de registros em 2 bancos de dados diferentes sendo eles o PostgreSQL e o MongoDB e o resultado apresentou um problema de resposta da consulta com limite de 100 mil registros quando realizado no MongoDB através de sua interface gráfica Robomongo, porém não foi possível até o momento identificar se o problema ocorreu no banco de dados ou na interface gráfica. Mesmo com o problema ocorrido na consulta dos 100 mil registros realizada no MongoDB, o banco de dados PostgreSQL através de sua interface gráfica PgAdmin apresentou resultado de tempo de resposta muito superior ao tempo de resposta apresentado pelo MongoDB, concluindo que mesmo com os problemas apresentados o banco de dados não relacional obteve um desempenho melhor nos testes efetuados.

O resultado final demonstra que, assim como o cenário utilizado para os testes, é possível utilizar o mesmo esquema para diversos tipos de cenários diferentes onde ao menos um atributo do sub-documento KEYS exista tanto em uma fonte como em outra fonte de dados envolvidas como no sub-documento DADOS do próprio documento principal.

De forma geral, através dos estudos de caso percebe-se que os objetivos foram alcançados sendo que, a modelagem construída possibilita o armazenamento de grande massa de dados como as dos sensores meteorológicos. Por não possuir uma coleção pré-definida, possibilita a inserção de qualquer tipo de dados obedecendo as regras da modelagem fazendo com que a modelagem seja escalável e flexível. Como a modelagem necessita que ao menos que um atributo seja igual entre todos os sub-documentos KEYS, a modelagem permite o cruzamento de dados entre dados de contextos diferentes possibilitando a inserção na mesma coleção e a recuperação dos documentos dentro da coleção se torna mais rápida, porém foi identificado um problema apresentado na interface gráfica Robomongo que ao precisar realizar uma consulta que traga de uma só vez mais de 50 mil registros, a aplicação acaba fechando.

Para trabalhos futuros existe uma grande quantidade de possibilidades como a de resolver o problema aqui encontrado sobre a consulta com mais de 50 mil registros na interface gráfica Robomongo, além de dados textuais, testar a modelagem com dados binários e a realização de testes da modelagem em outros bancos de dados NoSQL. Construir sistemas para sua utilização, onde será realizada inserção, consultas e alterações podendo assim avaliar melhor sua flexibilidade, adaptabilidade, escalabilidade e seu desempenho.

REFERÊNCIAS

Abraham Silberschatz, Henry Korth, S. S. *Sistema de Banco de Dados*. Terceira e. São Paulo: Makron Books, 1999. 778 p. vii, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 20, 21

BOOTON, J. *IBM finally reveals why it bought The Weather Company*. 2016. Disponível em: <http://www.marketwatch.com/story/ibm-finally-reveals-why-it-bought-the-weather-company-2016-06-15>. 4

BRITO, R. W. Bancos de dados NoSQL x SGBDs relacionais: análise comparativa. *Faculdade Farias Brito e Universidade de Fortaleza*, 2010. Disponível em: <http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Bancos+de+Dados+NoSQL+x+SGBDs+Relacionais+:+An?lise+Comparati>. 22

CROCKFORD, D. *The application/json Media Type for JavaScript Object Notation (JSON)*. 2006. Disponível em: <http://www.ietf.org/rfc/rfc4627.txt?number=4627>. vii, 27, 28

Dean, Jeffrey;Ghemawat, S. *MapReduce: Simplified Data Processing on Large Clusters*. 2004. 1 p. Disponível em: <http://research.google.com/archive/mapreduce.html>. 29

ELIOT. *State of MongoDB*. 2010. Disponível em: <http://blog.mongodb.org/post/434865639/state-of-mongodb-march-2010>. 26

ELMASRI, R.; NAVATHE, S. B. Fundamentals of Database Systems. *Database*, v. 28, p. 1029, 2003. ISSN 19406029. 21

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados - 6ª Edição.pdf*. [s.n.], 2011. 790 p. ISBN 978-85-7936-085-5. Disponível em: <http://fileshare3590.depositfiles.org/auth-1426943513b5db19f23dd9b11ebf50d2-187.39.203.223-1997336327-152949425-guest/FS359-5/SistBD6Ed...rar>. vii, 6, 7, 10, 14, 16, 17, 18

FEDERAL, U. et al. Simulação da evapotranspiração e otimização computacional do modelo de ritchie com clusters de bancos de dados. 2009. vii, 35

GARTNER. *Quadrante Mágico da Gartner para o DBMS Operacional*. 2015. Disponível em: <http://www.intersystems.com/br/produtos/cache/gartners-gartner-dbms/>. vii, 24, 25

INMET, I. N. d. M. Nota Técnina No. 001/2011/SEGER/LAIME/CSC/INMET. Rede de estações meteorológicas automáticas do INMET. n. 001, p. 1–11, 2011. vii, 32, 34

LI, T. et al. A Storage Solution for Massive IoT Data Based on NoSQL. *2012 IEEE International Conference on Green Computing and Communications*, p. 50–57, 2012. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6468294>. vii, 2, 3, 23, 24

MONGO. *Databases and Collections*. 2016. 1 p. Disponível em: <https://docs.mongodb.com/manual/core/databases-and-collections/>. vii, 26

MONGODB. Top 5 Considerations When Evaluating NoSQL Databases. n. June, p. 1–7, 2015. 26

MONGODB. *Documents*. 2016. Disponível em: <https://docs.mongodb.com/manual/core/document/>. vii, 27, 29

PERNAMBUCO, U. F. D. Uma Arquitetura de Cloud Computing para análise de Big Data provenientes da Internet Of Things Uma Arquitetura de Cloud Computing para análise de Big Data provenientes da Internet Of Things. 2014. 3, 15

PIRES, P. F. et al. Plataformas para a Internet das Coisas. *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 110–169, 2015. 3

POSTGRES. *PostgreSQL*. 2017. Disponível em: <https://www.postgresql.org/>. 24

SADALAGE, P.; FOWLER, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. [s.n.], 2012. 192 p. ISSN 1098-. ISBN 9780321826626. Disponível em: <http://medcontent.metapress.com/index/A65RM03P4874243N.pdf$\delimiter"026E30F$nhttp://books.google.com/books?hl=en{&}lr={&}id=AyY1a6-k3PIC{&}oi=fnd{&}pg=PT23{&}dq=NoSQL+Distilled:+A+Brief+Guide+to+the+Emerging+World+of+Polyglot+Persistence{&}ots=6GAoDEGKNi{&}sig=mz6P>. vii, 15, 22, 23

SILVERSTON, L. *The Data Model Resource Book*. [S.l.: s.n.], 1997. ISBN 0-471-15364-8. 6, 7

SOLDATOS, J.; SERRANO, M.; HAUSWIRTH, M. Convergence of utility computing with the Internet-of-things. In: *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*. [S.l.: s.n.], 2012. p. 874–879. ISBN 9780769546841. 1

SOUZA, V. C. O.; SANTOS, M. V. C. Amadurecimento, Consolidação e Performance de SGBDs NoSQL - Estudo Comparativo. *XI Brazilian Symposium on Information System*, p. 235–242, 2015. 3

STROZZI, C. *NoSQL - A Relational Database Management System*. 2007. Disponível em: <http://www.strozzi.it/cgi-bin/CSA/tw7/I/en{_}US/NoSQL/HomeP>. 14, 15

Veronika Abramova; Jorge Bernardino and Pedro Furtado. EXPERIMENTAL EVALUATION OF NOSQL DATABASES. *International Journal of Database Management Systems (IJDMS) Vol.6, No.3, June 2014*, v. 6, n. 100, p. 1–16, 2005. 3

WHITTEN, J.; BENTLEY, L.; DITTMAN, K. *Systems analysis and design methods*. Irwin/McGraw-Hill, 1998. ISBN 9780256199062. Disponível em: <https://books.google.com.br/books?id=0OEJAQAAMAAJ>. 8

ZASLAVSKY, A.; PERERA, C.; GEORGAKOPOULOS, D. Sensing as a Service and Big Data. *Proceedings of the International Conference on Advances in Cloud Computing (ACC-2012)*, p. 21–29, 2012. ISSN 9788173717789. 1, 2, 29