UNIVERSIDADE DO SUL DE SANTA CATARINA
JOÃO LUIZ MONTEIRO JOAQUIM

ANÁLISE DE DESEMPENHO ENTRE BANCO DE DADOS RELACIONAL E BANCO DE DADOS ORIENTADO A DOCUMENTOS PARA DADOS GEOESPACIAIS

Florianópolis
2015

JOÃO LUIZ MONTEIRO JOAQUIM

ANÁLISE DE DESEMPENHO ENTRE BANCO DE DADOS RELACIONAL E BANCO DE DADOS ORIENTADO A DOCUMENTOS PARA DADOS GEOESPACIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Flávio Ceci, M. Eng.

Florianópolis

2015

JOÃO LUIZ MONTEIRO JOAQUIM

ANÁLISE DE DESEMPENHO ENTRE BANCO DE DADOS RELACIONAL E BANCO DE DADOS ORIENTADO A DOCUMENTOS PARA DADOS GEOESPACIAIS

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação da Universidade do Sul de Santa Catarina.

Florianópolis, 17 de junho de 2015.

Professor e orientador Prof. Flávio Ceci, M. Eng.
Universidade do Sul de Santa Catarina

Prof. Aran Bey Tcholakian Morales, Dr. Eng.
Universidade do Sul de Santa Catarina

Prof. Júlio Gonçalves Reinaldo, B. Sc.
Institulo Stela

Dedico este trabalho aos meus pais, Luiz Sergio da Silva Joaquim e Rosana Monteiro Joaquim que foram os maiores incentivadores pela minha formação acadêmica, tornando também responsáveis por esta conquista.

AGRADECIMENTOS

Agradeço aos meus pais, Luiz Sergio da Silva Joaquim e Rosana Monteiro Joaquim pelo apoio e motivação dos estudos e seguir em frente, independente das dificuldades que a vida apresenta.

Agradeço aos meus orientadores deste trabalho, Júlio Gonçalves Reinaldo e Flávio Ceci pelo apoio e esclarecimento de dúvidas no desenvolvimento deste trabalho.

Agradeço ao Fernando Quadro, no qual inspirou conhecer a área de geoprocessamento e descobrir uma nova área com muitas oportunidades e desafios.

Agradeço aos amigos que fiz em no curso de Sistemas de Informação, unidade Dib Mussi pelas novas amizades que fiz.

RESUMO

Com o aumento do acesso da população a tecnologias de informação e comunicação (TIC), entre eles a internet, também houve um aumento na utilização dos sistemas de informação geográficos para os mais diversos fins, desde aprendizagem escolar até no auxílio nas tomadas de decisões. Para realizar a tarefa de armazenar as informações geradas pela utilização das TIC, é utilizado um software chamado banco de dados, entre eles o mais utilizado o banco de dados relacionais. Entretanto, esse tipo de banco de dados existe algumas desvantagens, entre elas o desempenho na manipulação de grandes volumes de dados, entre elas, o tipo de dado geográfico. Portanto, cria-se uma demanda por alternativas de banco de dados para realizar a manipulação de dados geográficos. Entre as alternativas que surgiram, podem-se destacar os bancos de dados orientado a documentos, entretanto não se têm conhecimento do comportamento desse tipo de banco de dados, na manipulação de dados geoespaciais. Com esse cenário de mudança na área de banco de dados, este trabalho tem como objetivo de analisar quais destes tipos de banco de dados tem melhor desempenho para dados geoespaciais. Para isso, foi desenvolvido um comparativo baseado em um determinado cenário especificado neste trabalho para alcançar este objetivo. Os resultados apresentados pelos testes foram satisfatórios fornecendo informações necessárias para alcançar o objetivo de saber qual tipo de banco de dados tem melhor desempenho para dados geoespaciais.

Palavras-chave: Sistema de Informação Geográfico. Banco de Dados Orientado a Documentos. Banco de Dados Relacional.

SUMÁRIO

1 INTRODUÇÃO

Os mapas estão no cotidiano da humanidade desde os tempos em que o homem deixou de ser nômade, e são utilizados das mais diversas formas. Os primeiros mapas eram entalhes nas pedras, pinturas em cavernas ou desenhos na argila cozida (LONGO, 2011). A cartografia (ciência que realiza a representação do espaço físico da terra através de mapas, cartas, entre outros) teve sua origem por volta de 2500 A.C., quando os sumérios criaram um mapa para representar uma parte da Mesopotâmia, que incluía o rio Eufrates e o Monte Zagros (LONGO, 2011). Os mapas tiveram grande importância na história da humanidade, principalmente na questão de poder utilizá-los como forma de registrar espaços/territórios e no suporte para tomada de decisões.

Ao longo do tempo, houve descobertas em diversas áreas que aperfeiçoaram a precisão e a qualidade da confecção de mapas. Entre essas descobertas, pode-se citar a área da matemática na qual os gregos, segundo Longo (2011), a partir da observação e de métodos científicos, como a utilização da trigonometria, permitiu a conclusão da esfericidade da terra, e a definição da latitude e longitude (sistema de coordenadas geográficas).

Até a segunda metade do século XX, a análise e criação dos mapas eram feitas de forma manual a qual apresentava problemas, como o relatado por Câmara, Davis e Monteiro (2014, p. 1) sobre a dificuldade de se realizar análises combinando vários mapas. A partir de então, com o advento da computação, foi possível representar as informações e armazenar tudo em um ambiente computacional, surgindo o geoprocessamento (CÂMARA, DAVIS e MONTEIRO; cap. 1)[1].

Sobre o geoprocessamento, Câmara, Davis e Monteiro (2014, p. 1) escrevem:

> [...] o termo Geoprocessamento denota a disciplina do conhecimento que utiliza técnicas matemáticas e computacionais para o tratamento da informação geográfica e que vem influenciando de maneira crescente as áreas de Cartografia, Análise de Recursos Naturais, Transportes, Comunicações, Energia e planejamento Urbano e Regional. As ferramentas computacionais para Geoprocessamento, chamadas de Sistemas de Informação Geográfica (GIS), permitem realizar análises complexas, ao integrar dados de diversas fontes e ao criar bancos de dados geo-referenciados. Tornam ainda possível automatizar a produção de documentos cartográficos.

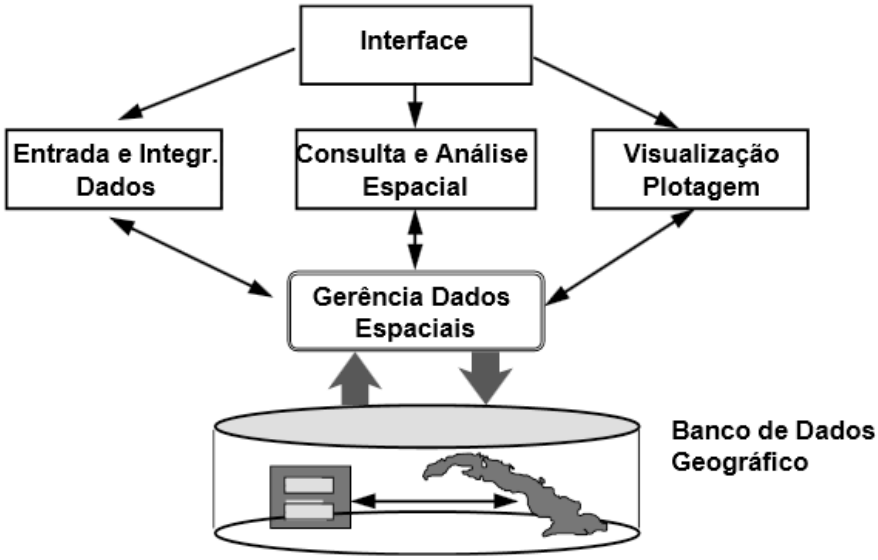
A partir do conceito de geoprocessamento, fica clara a vantagem da utilização dos sistemas de informação geográficos (SIG), também conhecido pela sua sigla inglês GIS (*Geographic Information Systems*). Sistemas de Informação Geográficos são responsáveis por

armazenar, manipular, processar grandes quantidades de dados geográficos, permitindo ainda mostrá-los como mapas eletrônicos. Câmara, Davis e Monteiro (2014, cap. 3, p. 1), sobre os SIG, escrevem:

> O termo Sistemas de Informação Geográfica (SIG) é aplicado para sistemas que realizam o tratamento computacional de dados geográficos e recuperam informações não apenas com base em suas características alfanuméricas, mas também através de sua localização espacial; oferecem ao administrador (urbanista, planejador, engenheiro) uma visão inédita de seu ambiente de trabalho, em que todas as informações disponíveis sobre um determinado assunto estão ao seu alcance, inter-relacionadas com base no que lhes é fundamentalmente comum a localização geográfica. Para que isto seja possível, a geometria e os atributos dos dados num SIG devem estar georreferenciados , isto é, localizados na superfície terrestre e representados numa projeção cartográfica.

Além do conceito, deve-se considerar os componentes que constituem um SIG que são, segundo Câmara, Davis e Monteiro (2014, cap. 3, p. 2): "[...] interface com o usuário; entrada e integração de dados; funções de consulta e análise espacial; visualização e plotagem; Gerência dos Dados Espaciais e Banco de dados Geográfico". A figura 1 representa o relacionamento entre os componentes de um SIG.

Figura 1 - Representação da estrutura geral do sistema de informação geográfico.



Fonte: Câmara, Davis e Monteiro (2014, cap. 3, p. 3).

No componente banco de dados geográfico, é utilizado um banco de dados que é descrito por Elmasri e Navathe (2005) como uma coleção de dados que possuem um significado para um determinado domínio no contexto do usuário, entretanto, para gerenciar

[1] Devido essa referência ser um livro ser disponibilizado na internet com uma estrutura diferente de um livro

esses dados, é utilizado um sistema gerenciador de banco de dados (SGBD) que é coleção de programas que permite a criação e manipulação de um banco de dados. Um banco de dados utiliza um modelo de dados para organizar o armazenamento dos dados, como, por exemplo, o modelo de dados relacionais que, segundo Elmasri e Navathe (2005), é descrito como um modelo que representa o banco de dados como um conjunto de relações ou tabelas, que representam coleções de valores e entre essas tabelas estão suas relações.

1.1 PROBLEMÁTICA

As informações relativas a recursos minerais, demarcação de terra, rotas comerciais, etc. foram criadas e armazenadas de forma manual e o desenho dos mapas poderia ser, além de impreciso para representar detalhes, pouco prático para análises mais detalhadas. Entretanto, com os Sistemas de Informação Geográficos (SIG) que, segundo Ozemoy, Smith e Sicherman (1981), são um conjunto de funções automatizadas, que fornecem aos profissionais capacidades avançadas de armazenamento, acesso, manipulação e visualização de informação georreferenciada, muitas vantagens e facilidades para os profissionais de geografia e cartografia, entre outros, foram disponibilizadas.

A importância de um SIG difere para cada área que o utiliza. Segundo Lobato (2008), para a cartografia tradicional, a agilidade de um SIG, nos dias atuais, em que as grandes mudanças na forma que a sociedade se comunica, tendo acesso a informações instantâneas a todo momento, em qualquer lugar no planeta é imprescindível. Sobre essa necessidade Lobato escreve: “Aqui se impõe o desafio à Cartografia: como acompanhar esses rearranjos espaciais, cada vez mais frequente e intensos, a que o espaço é constantemente estimulado?”. (LOBATO, 2008, p.01). Além de Lobato, Passos (2011) descreve a importância do SIG no ensino de cartografia para desenvolver habilidades de análise espacial e síntese de pensamento, consideradas competências inerente à cartografia.

Ao analisar a estrutura de um SIG, deve-se observar que o componente responsável por armazenar e manipular os dados para o SIG é o banco de dados geográficos, atualmente sendo o mais utilizado para essa responsabilidade os bancos de dados relacionais, entretanto esse tipo de banco de dados foi projetado para gerenciar dados alfanuméricos, logo

---

convencional, foi proposto adicionar o número do capítulo para facilitar na busca das citações.

o desempenho em realizar consultas baseados em dados geográficos podem não ser satisfatórios. (BATTY, 1990).

Com o avanço tecnológico e o aumento no número de pessoas utilizando essas tecnologias, houve o surgimento de alternativas aos bancos de dados relacionais para suprir as deficiências que ele apresentava, logo para suprir essas deficiências, surgiram os bancos de dados não relacionais, dentre eles os bancos de dados orientados a documentos, sendo uma das alternativas para solucionar os problemas relacionados anteriormente. (ALVAREZ, 2013).

Dada à importância de um SIG em várias áreas de utilização (como na educação, nos negócios, na administração pública, entre outros) e o aumento da utilização das tecnologias, chega-se a seguinte dúvida: a utilização de banco de dados orientado a documentos pode ser uma boa alternativa para trabalhar com dados geoespaciais comparados aos bancos de dados relacionais?

1.2 OBJETIVOS

Nesta seção, pretende-se apresentar os objetivos gerais e específicos deste trabalho.

1.2.1 Objetivo geral

Analisar o desempenho dos bancos de dados relacional e orientado a documentos para tratar de dados geoespaciais.

1.2.2 Objetivos específicos

- introduzir os Sistemas de Informação Geográficos (SIG);
- descrever os bancos de dados relacionais e orientado a documentos;
- desenvolver um experimento utilizado para comparar os bancos de dados;
- apresentar os resultados do experimento dos dados geoespaciais no banco de dados relacionais e orientado a documentos.

1.3 JUSTIFICATIVA

Com o aumento dos dispositivos de tecnologias e informação e o acesso dos mesmos pela população, houve uma necessidade de aumentar a interatividade do usuário com a informação especificamente na parte de dados geoespaciais, saber onde fica o hospital mais próximo, rota mais rápida para chegar a determinado destino, quantos quilômetros tem determinada área, qual a cidade a partir de um determinado ponto, entre outras necessidades. A partir dessas necessidades exemplificadas, Câmara, Davis e Monteiro (2014, cap. 1) afirmam que quando aparece a palavra "onde" dentre as questões e problemas que precisam ser resolvidos por um sistema informatizado, os SIGs têm uma grande chance de ser adotada. Scussel (2011) escreve sobre o aumento da importância dos SIGs nas organizações, escrevendo que só, no ano de 2011, houve um aumento da aplicação de sistemas SIGs nas áreas de planejamento urbano e da terra (56%) e Geomarketing (33%), como, também, para as organizações, os SIGs têm grande importância para a tomada de decisões, aumento da eficiência, reduzindo gastos com tarefas críticas, aumento da confiabilidade e capacidade de controle.

Com os avanços das tecnologias de informação, barateamento dos equipamentos de tecnologia e aumento no número de usuários da internet, houve a necessidade de banco de dados que atendessem problemas de escalabilidade, desempenho e armazenar dados de forma flexível, logo surgiram os banco de dados NoSQL (Not Only SQL) que são definidos por

Fowler (2012) como bancos de dados que seguem as seguintes características: não usam o modelo de dados relacional; são escritos em código aberto; são desenhados para rodar em grande número de computadores; são baseados nas necessidades de propriedades da internet do século 21; não possuem esquema para permitir que os campos sejam adicionados sem qualquer controle, portanto, podendo ser uma alternativa para solucionar o problema que os bancos de dados relacionais tem relacionado ao desempenho de consultas complexas envolvendo dados geográficos. (BATTY, 1990). Dos bancos de dados NoSQL, o banco de dados orientado a documentos, é o que será utilizado neste trabalho devido a sua forma de armazenar os dados de forma flexível e desempenho aceitável. Um banco de dados orientado a documentos é definido por Simões (2014) como um tipo de banco de dados que armazena coleções de documentos representados por objetos com identificadores únicos e um conjunto de dados, sendo objeto de estudo desta monografia.

O desempenho é o requisito não funcional comparado nessa monografia, pois, conforme Alvarez (2013), desempenho é uma das grandes preocupações da engenharia de software, pois o desempenho é um dos requisitos para que uma aplicação se torne sucesso comercial.

1.4 ESTRUTURA DA MONOGRAFIA

Esta monografia está dividida em introdução, revisão bibliográfica, método, desenvolvimento do comparativo e conclusão. No capítulo da introdução, são descritos brevemente os assuntos abordados nesta monografia, o problema, os objetivos gerais e específicos e sua justificativa.

No capítulo 2, revisão bibliográfica, os assuntos são apresentados com maior detalhe, como os sistemas de informação geográficos (SIG), banco de dados relacionais e banco de dados orientado a documentos.

No capítulo 3, método, são escritos os tipos de pesquisa utilizados pelo autor, as etapas metodológicas, a arquitetura da solução e as delimitações desta monografia.

No capítulo 4, experimento, são escritos as ferramentas utilizadas neste trabalho, o cenário do experimento, a preparação das bases de dados para realização do experimento, execução do experimento e apresentação.

No capítulo 5, conclusão, são descritos as conclusões realizadas pelo autor desta monografia baseado nos resultados do experimento executado no capítulo anterior. Também neste capítulo e descrito os supostos motivos por ter chego no resultado apresentado e as sugestões de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

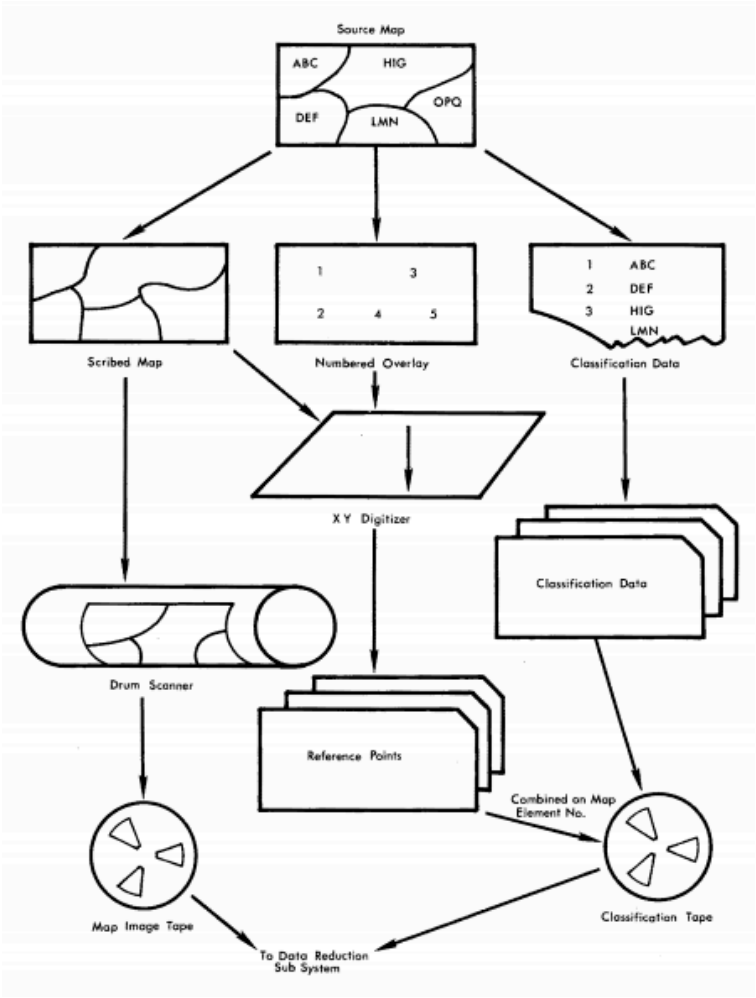
Nesta seção, são apresentados os assuntos temas desta monografia, sistemas de informação geográfica, banco de dados relacionais e banco de dados orientado a documentos.

2.1 SISTEMAS DE INFORMAÇÃO GEOGRÁFICA

Sistemas de informação geográfica (SIG), também, conhecidos pelo nome em inglês *Geographic Information System* (GIS) são definidos como um conjunto de sistemas voltados ao tratamento de informações geográficas, utilizando recursos computacionais com o objetivo de capturar, gerenciar, analisar e mostrar os dados geoespaciais como forma de auxiliar o usuário na tomada de decisões (GARTNER, 2014); (CÂMARA, DAVIS e MONTEIRO, 2014, cap. 3). O termo SIG foi criado por Roger Tomlinson[2], para nomear parte de um programa governamental criado pelo governo Canadense, que tinha como objetivo a criação de um inventário dos recursos naturais no início na década de 1960. Entretanto, naquela época, os recursos computacionais e mão de obra eram escassos e caros, tornando os SIGs possibilidades remotas. (CÂMARA, DAVIS e MONTEIRO, 2014, cap. 1) (MORAIS, 2012). A figura 2 mostra o fluxo de processamento dos dados proposto por Roger Tomlinson no artigo *An Introduction to the geo-information system of the Canada land territory.*

[2] Roger Tomlinson: Conhecido como pai do SIG, geógrafo que conceituou e desenvolveu o primeiro SIG para uso do inventário territorial do Canadá na década de 1960 (UCGIS, 2014).

Figura 2 – Representação do fluxo do processamento de dados proposto por Roger Tomlinson na década de 1960.



Fonte: Tomlinson (1967)

Os SIGs começaram a ser viáveis para utilização comercial na década de 1970 com o aumento dos recursos computacionais e consequente barateamento, portanto, também, na mesma década, surgiram os primeiros sistemas comerciais de CAD (Computer Aided Design, ou projeto assistido por computador), melhorando as condições para a produção de desenhos e plantas utilizados nos campos engenharia, servindo de base para os primeiros sistemas de cartografia automatizada, entretanto, na época como esses sistemas só podiam ser utilizadas em computadores de grande porte, somente grandes organizações tinham acesso (CÂMARA, DAVIS e MONTEIRO, 2014, cap. 1).

A partir da década de 1980, começa um crescimento acelerado dos SIGs, crescimento este que dura até os dias atuais. Nos Estados Unidos, foi criado o centro de

pesquisa, centro nacional de informação e análise geográfica (NCGIA[3]), tornando o geoprocessamento uma disciplina científica independente. Esse crescimento levou ao surgimento de várias aplicações SIGs por diversos motivos, como, o barateamento dos recursos, a disponibilização dos computadores pessoais, o desenvolvimento dos SGBDs relacionais, entre outros (CÂMARA, DAVIS e MONTEIRO, 2014, cap. 1).

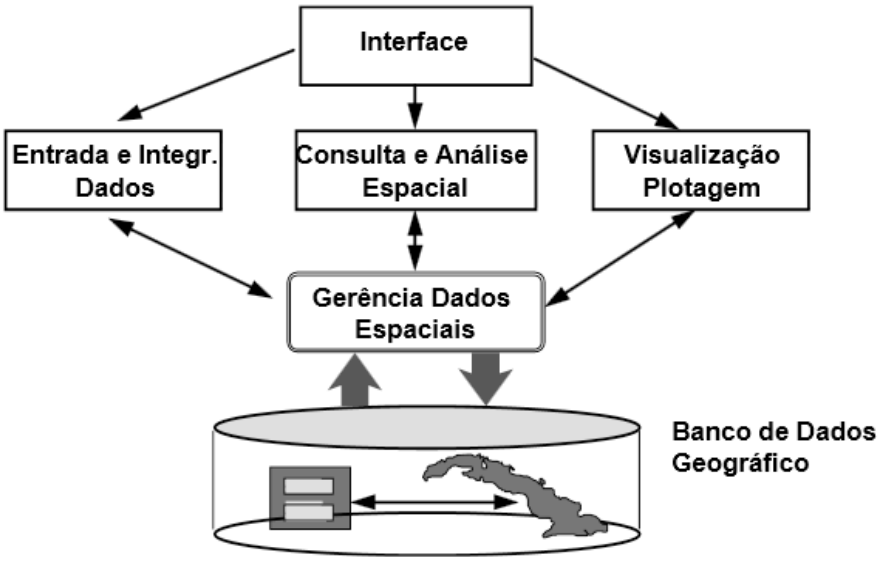
No Brasil, o geoprocessamento foi conhecido no início da década de 1980 a partir da divulgação e formação de pessoal pelo professor Jorge Xavier da Silva na Universidade Federal do Rio de Janeiro (UFRJ). Em 1982, a vinda de Roger Tomlinson ao Brasil, para divulgação dos SIGs, incentivou o surgimento de vários grupos acadêmicos interessados em desenvolver a tecnologia (CÂMARA, DAVIS e MONTEIRO, 2014, cap. 1).

2.1.1 Estrutura

A analisar a estrutura de um SIG, deve-se considerar que, devido à ampla utilização dele em diversas áreas, não segue um padrão rígido de como toda a estrutura deve ser, mas define um conjunto mínimo de componentes que devem estar presentes, esses componentes mínimos são: componente(s) de interface com o usuário, componente(s) de processamento de dados espaciais e componente(s) de armazenamento e recuperação de dados espaciais e não espaciais. A figura 3 representa um exemplo da estrutura geral de um SIG.

[3] NCGIA – Centro Nacional de Informação e Análise Geográfica (National Centre of Geographic Information and Analysis) é um centro de pesquisa independente dedicado a pesquisa básica, educação e tecnologias relacionadas aos SIGs (NCGIA,2014).

Figura 3 – Exemplo teórico da estrutura de um SIG.



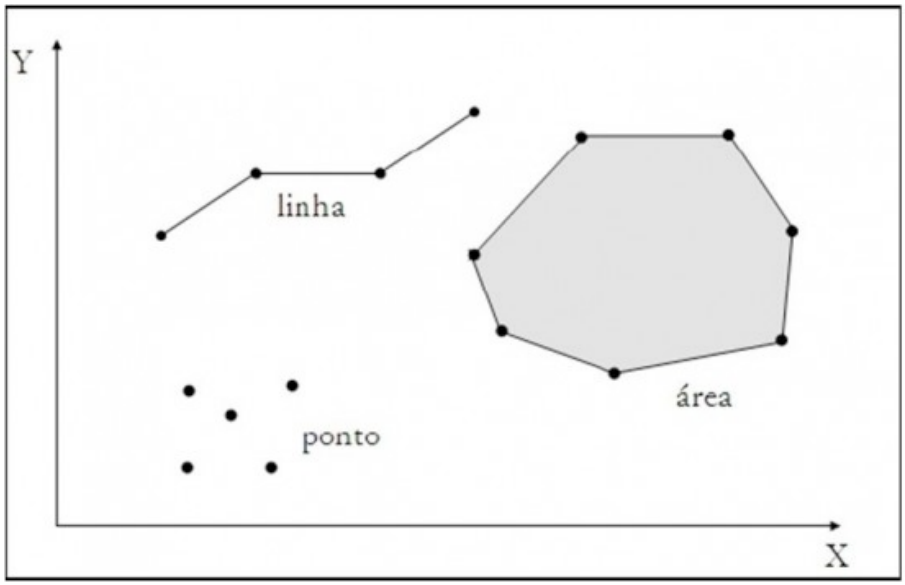
Fonte: Câmara, Davis e Monteiro (2014, cap. 3, p. 3).

Conforme escrito anteriormente, esses componentes são organizados de forma hierárquico sendo o nível mais externo ou nível mais próximo do usuário é responsável pela iteração homem-máquina, como o sistema é operado e controlado, podendo ser essa iteração ser realizada de diversas formas, um navegador de internet, um programa de linha de comando, aplicativo desktop, entre outros. O nível intermediário tem como responsabilidade o processamento de dados geoespaciais, podendo ser composto de diversos componentes para adequar o SIG à necessidade do usuário, exemplos de componentes nesse nível são mecanismos de conversão de dados, análise e consulta espacial, álgebra de mapas, estatística espacial entre outros. O nível interno tem como responsabilidade o armazenamento e manipulação dos dados espaciais e não espaciais composta por um sistema de gerência de banco de dados geográficos capaz de manipular dados espaciais e não espaciais. (CÂMARA, DAVIS e MONTEIRO, 2014, cap. 3) (CASANOVA, 2014, cap. 1).

2.1.2 Tipos de dados geográficos

Dados geográficos ou também chamado dados geoespaciais são utilizados dentro dos bancos de dados geográficos e se diferenciam dos demais devido ao seu componente espacial, isso significa que esse tipo de dado pode ter uma localização no espaço geográfico, podendo ter como base suas coordenadas. Esse tipo de dado pode ser classificado em dados vetoriais e dados matriciais (ou também referenciado de varredura). Dados vetoriais representam os dados por pelo menos um par de coordenadas (latitude e longitude), podendo haver três representações: ponto, linha e polígono (área). Pontos são utilizados para representar dados sem dimensões (sem largura e comprimento), exemplos são localização de escolas para um aplicativo de zoneamento, localização das ocorrências de crimes de determinada cidade, entre outros. Linhas são utilizadas para representar dados com única dimensão (somente comprimento), exemplo seria o comprimento de um rio, estradas, ferrovias, entre outros, e polígonos representam dados com duas dimensões (largura e comprimento), exemplo seria demarcação territorial de determinado país, um lago, uma floresta, entre outros (MEDEIROS, 2015) (MORAIS, 2000). A figura 4 demonstra um exemplo de cada tipo de dado vetorial.

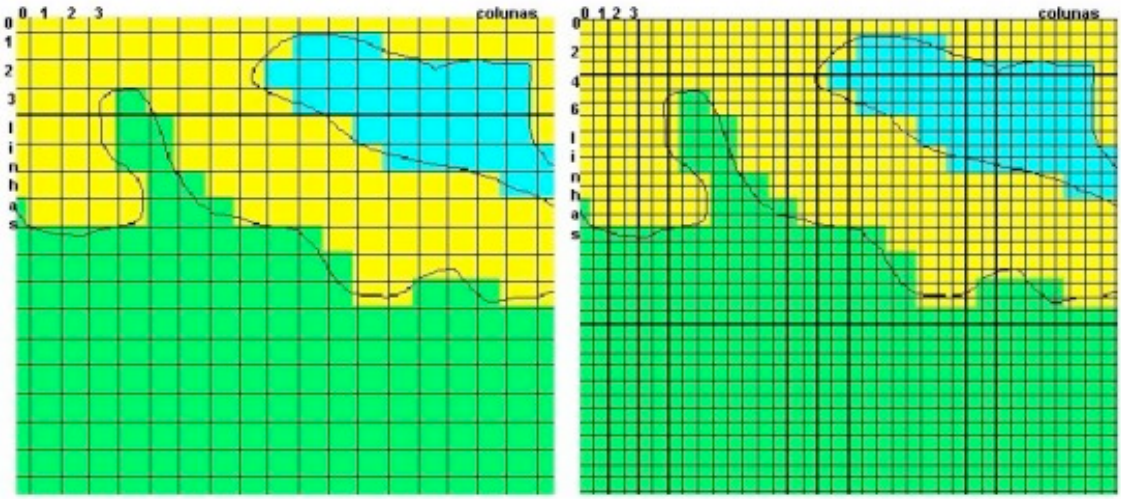
Figura 4 – Exemplo dos tipos de dados vetoriais.



Fonte: Medeiros (2015)

Dados matriciais, conforme mencionado anteriormente, representam os dados a partir de uma superfície montada em forma de um conjunto de linhas e colunas nos quais, para cada célula, há seu valor atribuído. (MEDEIROS, 2015) (MORAIS, 2000). A figura 5 mostra um exemplo de um mapa, utilizando os dados matriciais.

Figura 5 – Exemplo do tipo de dado matricial.



Fonte: Medeiros (2015).

A figura 5 demonstra um mapa a partir de uma matriz e para cada célula, há seu respectivo valor e pinta com uma determinada cor, representando, assim, os dados matriciais.

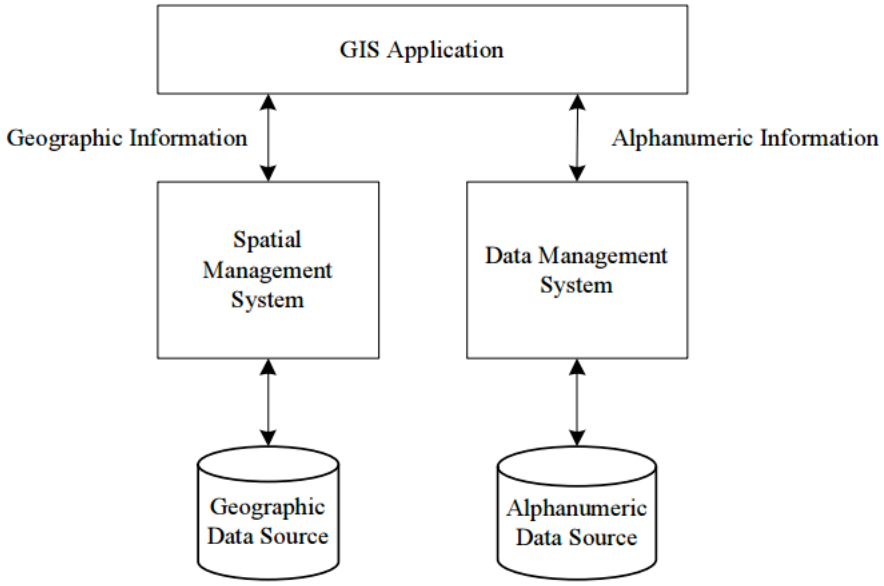
2.1.3 Tipos de arquitetura para gerenciamento dos dados

Com o aumento da utilização dos SIGs, houve um aumento da importância da utilização dos SGBDs devido ao grande volume de dados que os mesmos devem manipular, entretanto os dados geográficos são dados complexos a que foram propostos arquiteturas para realizar essa tarefa, essa arquitetura pode dividi-las por geração, sendo a primeira a arquitetura dual ou também conhecida como híbrida, a segunda arquitetura integrada ou em camadas e a terceira arquitetura extensível (SLATANOVA, 2014) (LUACES, 2004).

Inicialmente, a arquitetura que era utilizada nas aplicações SIG era a arquitetura dual ou também referenciada por outros autores como arquitetura híbrida. Esse tipo de arquitetura consistia nos dados geográficos e não geográficos ser armazenados em lugares

diferentes, sendo que os dados não geográficos são armazenados num SGBD relacional, enquanto que os dados geográficos, num arquivo externo, entretanto a ligação desses dados é feita a partir de uma ligação lógica no qual é utilizado um identificador único. Esse tipo de arquitetura era utilizado devido aos SGBDs não suportarem manipulação de dados geográficos, logo, sem o suporte de um SGBD, não é possível utilizar os recursos do mesmo, como gerenciamento de transações, integridade, etc. (SLATANOVA, 2014) (LUACES, 2004). A figura 6 demonstra um exemplo da arquitetura dual, no qual o SIG para os dados não espaciais apontam para uma fonte de dados e os dados geoespaciais para outra fonte.

Figura 6 – Exemplo de arquitetura dual no SIG



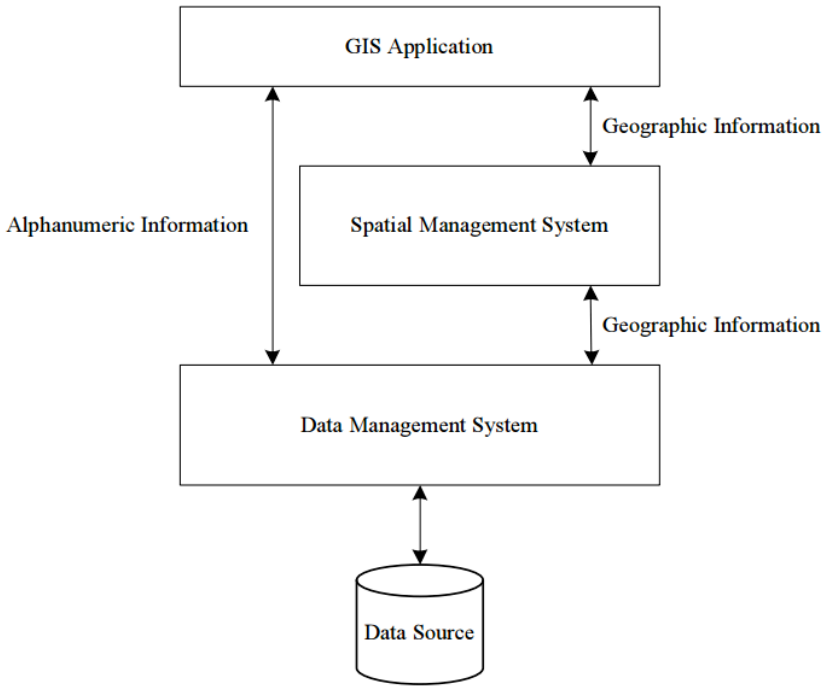
Fonte: Luaces (2004, p. 68).

A arquitetura integrada ou também conhecida como em camada foi criada com o objetivo de solucionar o problema que a arquitetura dual tinha na parte de armazenamento dos dados reunindo ambos os tipos de dados no SGBD. Entretanto, o processamento dos dados geográficos é realizado separado a partir de um módulo ou como escrito por outros autores numa camada separada. Esse tipo de arquitetura trouxe a vantagem, conforme mencionado anteriormente de armazenar ambos os tipos de dados (geográficos ou não geográficos) num SGBD. Contudo, os dados geográficos são armazenados num formato específico dos SGBDs,

chamado BLOB[4], e, devido à limitação do SGBD de manipular esse tipo de dado, transfere a responsabilidade para o aplicativo SIG em realizar essa tarefa, logo, transferindo essa responsabilidade às consultas que não podem ser otimizadas para trazer maior desempenho. (SLATANOVA, 2014) (LUACES, 2004).

A figura 7 demonstra a arquitetura integrada no SIG, mostrando ambos os tipos de dados numa única fonte de dado e módulo separado responsável por realizar o processamento dos dados geográficos.

Figura 7 – Exemplo de arquitetura integrada no SIG



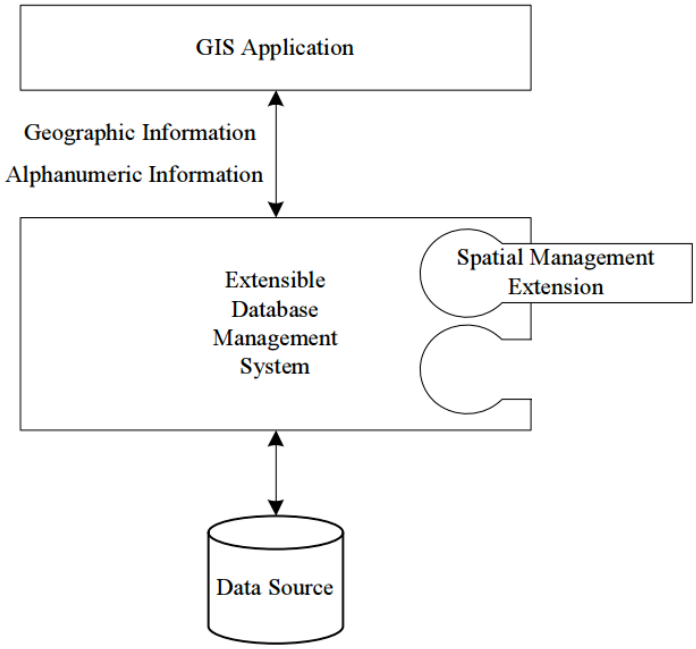
Fonte: Luaces (2004, p. 69).

A arquitetura extensível, ou também referenciada por outros autores como sendo uma melhoria da arquitetura integrada, e uma arquitetura no qual os dados geográficos ou não ainda continuam na mesma fonte de dados como arquitetura integrada. Entretanto ela soluciona o problema que ela tinha devido aos SGBDs de mercado que oferecerem um módulo especializado para trabalhar com dados geográficos, além de funções específicas para trabalharem com eles. Exemplo de SGBDs que disponibilizam módulos são Informix

[4] BLOB - Grande Objeto Binário (Binary Large Object), tipo de formato utilizado em formato de dados para armazenar tipos de dados binários (Oracle, 2015).

Universal Server, Oracle, PostgreSQL, entre outros. (LUACES, 2004). A figura 8 demonstra por meio de diagrama como a arquitetura extensível funciona.

Figura 8 – Exemplo de arquitetura extensível no SIG



Fonte: Luaces (2004, p. 71).

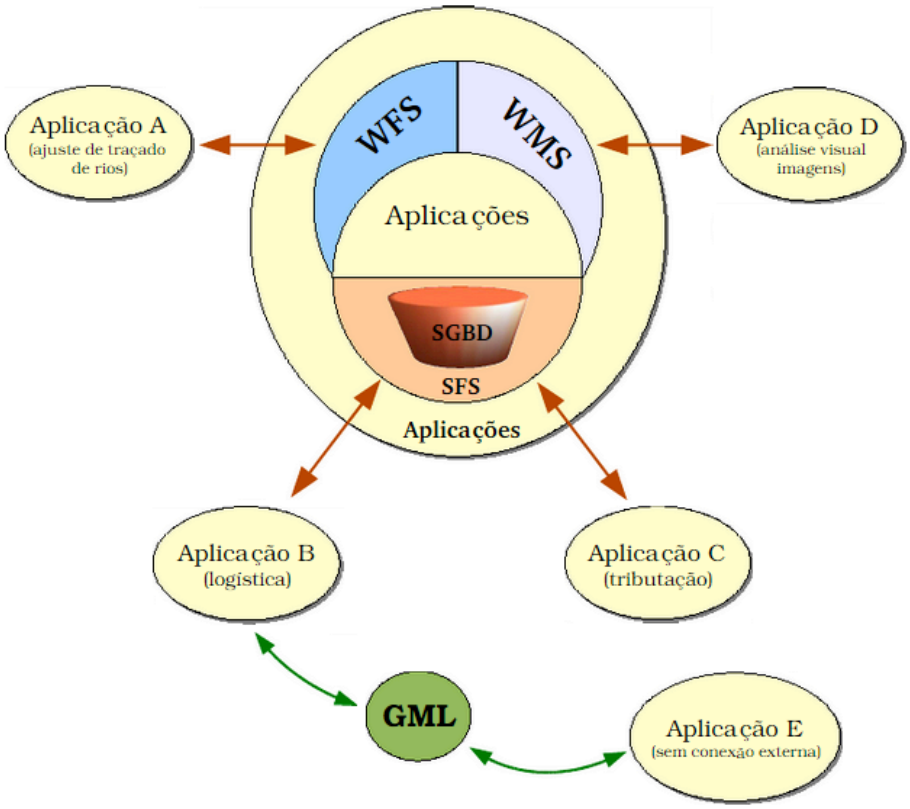
Sobre as vantagens que esse tipo de arquitetura disponibiliza, Luaces (2004) escreve que por o processamento e os dados estar no mesmo local (SGBD), além de estarem disponíveis todos os benefícios que um SGBD oferece, também as requisições podem ser otimizadas via linguagem de consulta que é utilizada pelo banco de dados, assim, disponibilizando um maior desempenho no processamento das consultas.

2.1.4 Consórcio open geospatial consortium

Criado em 1994, o consórcio internacional open geoespatial (OGC – Open Geospatial Consortium), antes denominado OpenGIS, é um consórcio formado por centenas de empresas, instituições de ensino e pesquisa ao redor do mundo com a missão criar especificações abertas ao público tendo como objetivo permitir a interoperabilidade dos

sistemas voltados para a área de geotecnologias. As especificações que a OGC definiu tem, além da uma ampla adoção pelo mercado, também é bastante amplo devido a diversas especificações criadas, exemplos de especificações mais utilizadas são a Simple Features Specification (SFS), Web Feature Service (WFS), Web Map Service (WMS) e Geography Markup Language (GML). A SFS é uma especificação relacionada a manipular dados geográficos para o SGBD, usando a linguagem de consulta utilizada neles, a WFS define uma forma de acesso aos dados geográficos via protocolo HTTP[5], a especificação WMS define 4 protocolos que permitem a visualização de dados geográficos em forma de camadas, contendo vetores e imagens, e GML define um formato via XML para transporte e armazenamento de dados geográficos. (UCHOA, 2004) (UCHOA, 2010). A figura 9 demonstra como essas especificações se relacionam com as aplicações SIG.

Figura 9 – Demonstração da utilização das especificações da OGC num SIG.



Fonte: Uchoa (2004, p. 13).

[5] HTTP – Protocolo transferência de hipertexto (Hypertext Transfer Protocol), é um protocolo de rede utilizado na internet para transferência de arquivos entre duas máquinas (MITCHELL, 2015).

Conforme na figura e comentado por Uchoa (2004), dentre as várias especificações definidas pela OGC, pode-se considerar a especificação SFS a mais importante, pois ela é o padrão de organização no qual os dados geográficos devem ser armazenados e as funções que os bancos devem disponibilizar.

2.2 BANCO DE DADOS

Pode-se definir Banco de dados como uma coleção de dados inter-relacionados e persistentes, que representam informações de um determinado domínio, usados por sistemas de aplicação para visualização e manipulação desses dados pelos usuários. (DATE, 2004) (KORTH, 1995).

As aplicações pioneiras de banco de dados mantinham os registros de grandes organizações, contendo em muitas dessas aplicações estruturas de dados semelhantes, entretanto, os problemas com esses bancos de dados eram vários e, entre eles, podemos citar a misturados relacionamentos conceituais, armazenamento físico e a localização de registros no disco, inflexibilidade para os acessos a registro quando novas consultas e transações eram necessárias, dificuldade em reorganizar o banco de dados, quando as mudanças eram necessárias para atender novos requisitos da aplicação. As consultas aos bancos de dados eram feitas através de interfaces para a linguagem de programação, fazendo com que fosse necessário implementar novos programas para cada uma. Esses bancos de dados eram implementados em computadores de grande porte (mainframes) e foram utilizados desde meados dos anos 1960 até os anos de 1970 e 1980 e eram baseados nos modelos de banco de dados hierárquico, de rede e de arquivos invertidos (ELMASRI e NAVATHI, 2005).

O banco de dados relacional teve seu início no ano de 1970, pelo cientista da computação Edgar Codd que publicou um artigo chamado "A Relational Model of Data for Large Shared Data Banks" ("Um modelo relacional de dados para grandes bancos de dados compartilhados"). Esse artigo introduziu a ideia de modelo relacional, utilizando a álgebra relacional, enfatizando a importância da separação do armazenamento físico da sua representação conceitual. Ele expôs ainda uma linguagem simples de consulta de alto nível para acesso aos dados, permitindo que desenvolvedores utilizassem operações com um conjunto completo de dados de uma única vez (CAMILO, 2014).

Posteriormente, foram criados dois bancos de dados baseados nas ideias de Edgar Codd, o primeiro foi o “System R”, um projeto da IBM que, além do desenvolvimento do banco de dados, apresentou a linguagem de consulta conhecida como SQL (Structured Query Language), entretanto, mesmo tendo sucesso na criação do banco de dados e do SQL, ele não teve um sucesso comercial tornando-se a base de outro banco de dados da IBM, chamado SQL/DS, que teve sucesso comercial (CAMILO, 2014). O segundo banco de dados baseado nas ideias de Edgar Codd foi o Ingress, desenvolvido pela Universidade da Califórnia em Berkeley utilizado por agências de pesquisa e pelas forças armadas dos Estados Unidos da América. Muito similar ao System R, embora funcionasse em outra plataforma, tendo como vantagem que seu código fonte foi disponibilizado publicamente, mesmo mais tarde sido comprado e comercializado pela Computer Associates na década de 1980 (CAMILO, 2014).

2.2.1 Linguagem SQL

Linguagem de consulta estruturada (Structured Query Language), ou também conhecida pela sua sigla SQL, é uma linguagem de consulta criada por Edgar Frank Codd no início da década de 1970 no laboratório de pesquisa da IBM em San José, no estado da Califórnia, Estados Unidos da América, com o objetivo de ser uma linguagem amigável para o usuário final realizar consulta aos dados que estão armazenados nos SGBDs, sendo utilizada pela primeira vez no banco de dados System R da IBM. (DATE, 2004) (KORTH, 1995).

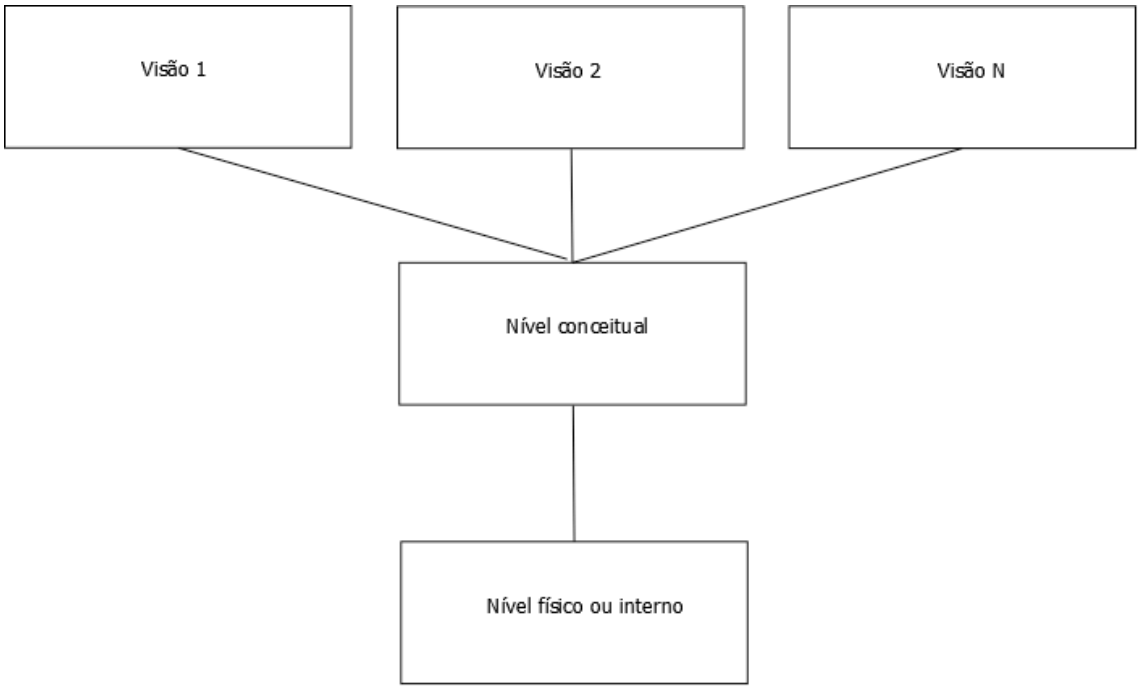
Por ser uma linguagem amigável de se utilizar, várias corporações começaram a aplicá-la, criando adaptações da SQL para seus bancos de dados comerciais e, com isso, logo houve a necessidade de padronização da SQL, sendo gerenciada por comitês da ISO (Organização Internacional de Normalização) e da ANSI (Instituto Nacional Americano de Padrões). (CAMILO, 2014).

O processo de padronização começou em 1983, entretanto, somente em 1987 foi concluído sendo chamada de SQL1, em 1989, foi liberada uma nova versão que possuía suporte a chaves primárias e chaves estrangeiras. Em 1992, uma nova versão chamada SQL92 foi disponibilizada com novas instruções e extensões, nos anos de 1999 e 2003 foram criados as novas versões SQL99 e SQL2003, entretanto, a SQL99 se impõe como padrão entre os fabricantes de banco de dados. (CAMILO, 2014).

2.2.2 Abstração de Dados

Um dos principais objetivos dos sistemas de banco de dados é prover aos usuários as informações que buscam de forma eficiente, sem precisar saber dos detalhes da forma como os dados são armazenados e mantidos, oferecendo uma visão abstrata do mesmo (KORTH, 1995). Entretanto, esse objetivo tem levado o projeto das estruturas de dados a serem complexas, uma vez que os usuários dos bancos de dados não têm conhecimentos técnicos na área de computação, tornando necessário esconder essa complexidade entre diversos níveis de abstração com o objetivo de simplificar a interação do usuário com o SGBD (KORTH, 1995). Essa abstração, conhecida como arquitetura ANSI/SPARC, divide-se em três níveis: nível interno ou físico, nível conceitual e nível externo ou de visões (DATE, 2004) (KORTH, 1995). A figura 10 demonstra os três níveis de abstração, sua hierarquia e relacionamento entre os níveis.

Figura 10 – Três níveis de abstração de dados



Fonte: Adaptado de: (KORTH, 1995, p. 5) (DATE, 2004, p. 29).

O nível físico representa o mais baixo da abstração, descrevendo como os dados estão de fato armazenados, tendo suas complexas estruturas de dados em detalhes. O nível conceitual descreve quais dados estão armazenados e os relacionamentos existentes entre eles

e, no nível de visões, são descritas apenas partes do banco de dados, utilizando estruturas mais simples comparadas ao nível conceitual, entretanto, mantém alguma complexidade devido ao grande tamanho do banco de dados. (KORTH, 1995).

2.2.3 Modelos de Dados

Segundo Korth (1995, p. 6), os modelos de dados são "uma coleção de ferramentas conceituais para descrição, relacionamentos, semântica e restrições de consistência de dados". Os modelos de dados são divididos em três grupos: modelos físicos de dados, modelos lógicos baseados em objetos e modelos lógicos baseados em registros. (KORTH, 1995) (GEREMIA, 2010).

Modelos físicos de dados são utilizados para a descrição de dados no nível físico, sendo que, em comparação aos outros modelos de dados, é o menos utilizado e existem poucos modelos de dados para esse grupo, que seriam: Modelo unificador e Estrutura de memória. (KORTH, 1995).

Modelos lógicos baseados em objetos são utilizados para a descrição de dados nos níveis conceituais e de visões, tendo como característica a capacidade de estruturação flexível, contendo nesse grupo o modelo entidade-relacionamento, modelo orientado a objeto, modelo binário, modelo semântico de dados, modelo infológico, entre outros. (KORTH, 1995). Dentre esses modelos, destaca-se o modelo orientado a objetos, que é um modelo baseado no modelo de programação orientada a objetos (OO). Esse modelo de dados é muito utilizado em aplicações mais complexas devido aos seus requisitos diferentes, como estrutura de dados mais complexa, transações de longa duração, novos tipos de dados (específicos para armazenamento de imagens e textos longos), definição de operações não convencionais específicas da aplicação, logo, oferecendo esse modelo de dados mais flexibilidade sem estar limitado pelos tipos de dados e linguagens de consultas de outros modelos de dados. (ELMASRI e NAVATHE, 2005).

Modelos lógicos baseados em registros são utilizados para a descrição de dados nos níveis conceituais e de visões assim como o modelo lógico baseado em objetos, mas são chamados assim devido ao banco de dados ser estruturado em registros de formato fixo de diversos tipos, sendo que cada tipo de registro define um número fixo de campos, atributos

um tamanho fixo (KORTH, 1995). Portanto, além de serem utilizados para descrição de dados, fornecem uma descrição de alto nível da implementação. (KORTH, 1995).

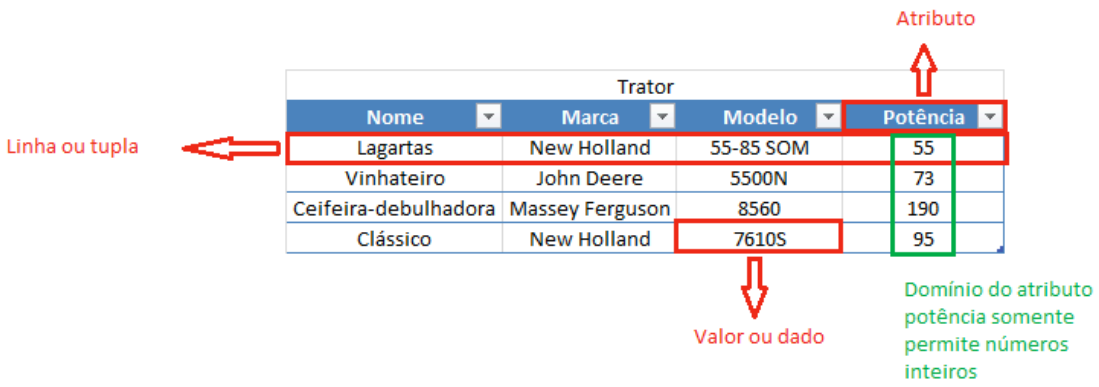
Os modelos de dados deste grupo mais conhecidos são o relacional, de rede e o hierárquico. O modelo de dados hierárquico organiza os dados baseados na estrutura de dados tipo árvore, essa estrutura pode ser pensada numa raiz e nela podem existir vários ramos (galhos) e para cada um podem existir várias folhas, logo, somente existe um caminho único para chegar numa determinada folha. O modelo de dados em rede seria uma extensão do modelo hierárquico, tendo somente como diferença, nesse tipo de modelo de dados, a folha pode ter vários registros pai, enquanto que no hierárquico somente é permitida um registro pai. (KORTH, 1995) (GEREMIA, 2010).

2.2.3.1 Modelo de dados relacionais

O modelo relacional surgiu a partir de um resultado teórico de Ted Codd na década de 1970, utilizando a teoria dos conjuntos e álgebra relacional para solucionar algumas necessidades: independência dos dados dos SGBDs; processamento *ad hoc;* disponibilizar um conjunto de operações baseadas em álgebra relacional para armazenamento e recuperação dos dados (BONFIOLI, 2006).

No modelo relacional, a estrutura fundamental, como o nome sugere, é a relação, ou tabela como sugere Cacho (2014), a qual é uma representação da informação com um conjunto de propriedades definidas para serem armazenadas (CALDEIRA, 2014) (BONFIOLI, 2006). Para Caldeira (2014), o modelo relacional é composto por três conceitos básicos: domínio, relação e atributo. Os atributos, propriedades ou campos são as informações que representam uma determinada tabela (CACHO, 2014) (BONFIOLI, 2006). Para cada atributo, é obrigatório participar de um domínio, sendo que um domínio é definido como o conjunto de dados válidos para o atributo, como, por exemplo, um atributo chamado telefone, que pode aceitar um conjunto número de nove dígitos (CACHO, 2014). Uma tupla é definida por Cacho (2014), como representação dos valores de uma determinada tabela. A figura 11 mostra um exemplo dos conceitos apresentados anteriormente:

Figura 11 – Exemplo de uma estrutura de tabela



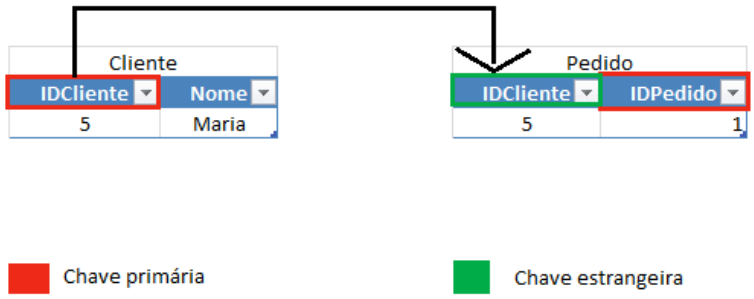
Fonte: Adaptado de: (CALDEIRA, 2014, p. 14).

No modelo relacional, Caldeira (2014, p. 12-13) escreve sobre as propriedades de que uma tabela é caracterizada:

> Ter um nome único dentro do mesmo diagrama de modelo de dados relacional; Ter de zero a *n* linhas, cuja ordenação é indiferente dado que não são identificadas pela sua posição. Uma relação sem nenhuma linha diz-se vazia. Ter de zero a *n* linhas, cuja ordenação é indiferente dado que não são identificadas pela sua posição. Uma relação sem nenhuma linha diz-se vazia. Cada um dos atributos contém valores retirados dum domínio particular, o que quer dizer que num mesmo atributo os dados são obrigatoriamente todos do mesmo tipo; Numa mesma relação não podem existir dois atributos com o mesmo nome; Cada relação tem que ter uma *chave primária*. Uma chave primária é um atributo, ou combinação de atributos, cujos valores proporcionam uma identificação unívoca das tuplas duma relação, ou seja, um certo valor para a chave só pode aparecer uma única vez em cada relação. Significa ainda que as tuplas duma relação são todas diferentes entre si, i.e., não são permitidas linhas duplicadas. Considerando a relação representada na Figura 2-2 não faz qualquer sentido que seja permitida a duplicação de uma linha dessa relação, pois os dados perderiam a sua coerência; A intersecção de uma coluna com uma linha corresponde a um dado ou valor. Não é permitida a existência de grupos de atributos repetidos; Os dados, ou valores, são sempre de tipo atómico.

Date (2004 apud BONFIOLI, 2006) define integridade como: "[...] a precisão ou correção de dados no banco de dados. Nesse contexto, "integridade" significa semântica e são as restrições de integridade que representam o significado dos dados." A figura 12 mostra um exemplo de uma tabela com uma chave primária e chave estrangeira:

Figura 12 – Exemplo de relacionamento entre tabelas de um banco de dados relacional



Fonte: Adaptado de: (BONFIOLI, 2006, p. 7).

Conforme demonstra a figura anterior, existem duas tabelas chamadas Cliente e Pedido. Na tabela Pedido, para realizar o relacionamento, existe uma coluna chamada IDCliente com o valor do respectivo cliente ao qual está relacionado.

2.2.4 Banco de Dados Orientado a Documentos

Com o advento de novas fontes de dados como sensores, GPS (Global Positioning Systems), entre outros, foi desencadeado um aumento rápido no volume e os tipos de dados gerados, trazendo novos desafios e oportunidades ao tratamento desse grande volume de dados. Para resolver esse novos desafios, houve o surgimento de novos tipos de banco de dados conhecidos como NoSQL (No Only SQL), que consistem num grande número de tipos de bancos de dados, utilizando outras estruturas de dados fora do modelo relacional, como, por exemplo, banco de dados orientado a documentos, banco de dados chave/valor, banco de dados orientado a grafos, entre outros (TIWARI, 2011).

Os bancos de dados orientados a documento têm como seu conceito principal o documento, o documento nesse tipo de banco de dados consiste num conjunto de informações

baseadas em chave valor, podendo ser armazenados como arquivo XML, JSON[6], BSON, entre outros. Tanto o banco de dados relacionais e orientados apresentam similaridades, como o fato de ambos obrigatoriamente ter um campo que identifica unicamente o registro na mesma coleção, entretanto, que diferencia esse tipo de banco de dados dos bancos de dados objeto-relacionais está no fato desse tipo de banco de dados armazenarem os dados de forma flexível, sem precisar que os documentos sigam uma estrutura rígida comparada aos bancos de dados relacionais, essa característica é descrita por algumas referências por ter seu schema flexível. (TIWARI, 2011) (SADALAGE; FOWLER, 2012). A figura 13 demonstra um exemplo de um registro armazenado em determinado banco de dados orientado a documentos, baseiam esse registro no formato JSON.

Figura 13 – Exemplo de registro no formato JSON

```
{ "firstname": "Martin",
  "likes": [ "Biking",
             "Photography" ],
  "lastcity": "Boston",
  "lastVisited":
}
```

Fonte: Sadalage e Fowler, 2012.

A figura 14 demonstra outro registro no mesmo formato (JSON), armazenado na mesma coleção do mesmo banco de dados orientado a documentos.

[6] JSON (J*avaScript Object Notation*) é uma estrutura de dados baseadas em texto com o objetivo de trocar dados entre linguagens de programação (ECMA, 2013, p. 4).

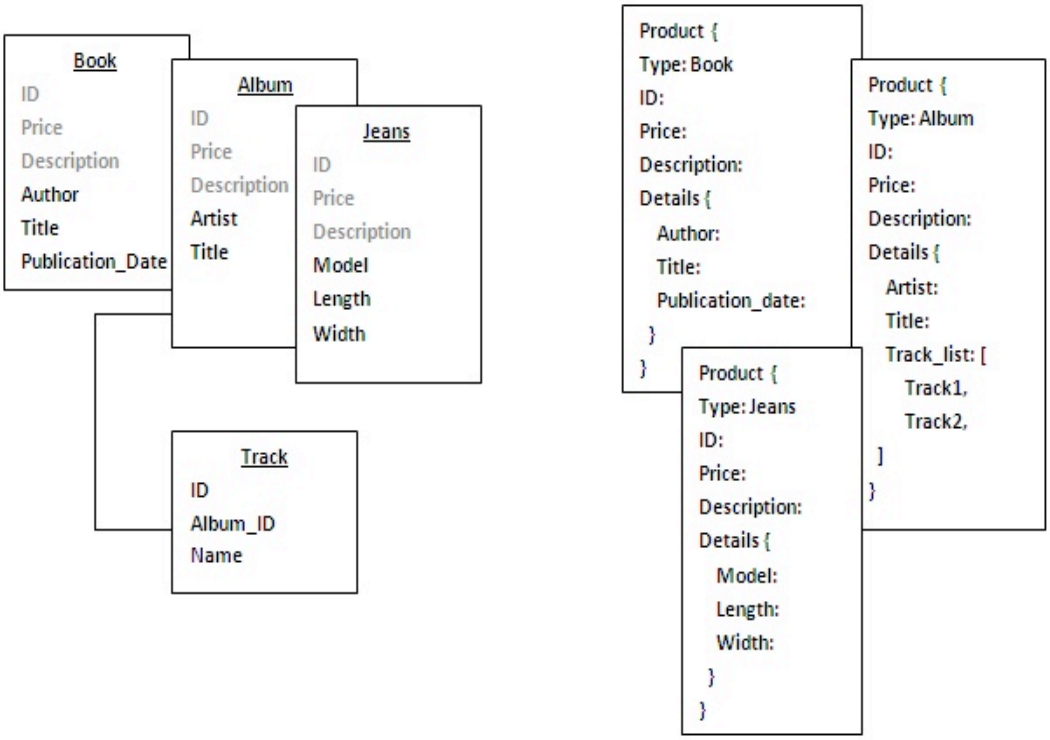
Figura 14 – Exemplo de registro no formato JSON semelhante

```
{
  "firstname": "Pramod",
  "citiesvisited": [ "Chicago", "London", "Pune", "Bangalore" ],
  "addresses": [
    { "state": "AK",
      "city": "DILLINGHAM",
      "type": "R"
    },
    { "state": "MH",
      "city": "PUNE",
      "type": "R" }
  ],
  "lastcity": "Chicago"
}
```

Fonte: Sadalage e Fowler, 2012.

Essa diferença dos bancos de dados relacionais e orientados a documentos, Sadalage e Fowler (2012), escrevem que essa diferença está definida, pois nos bancos de dados orientados a documentos não existir atributos vazios. A figura 15 demonstra um exemplo de modelagem de dados no banco de dados relacional e orientado a documento.

Figura 15 – Exemplo de modelagem no banco de dados relacional e banco de dados orientado a documentos.



Fonte: Pichiliani, 2013.

A figura 15 demonstra a diferença de modelagem de um relacionamento entre as entidades “Livro” (Book), “Album” e “Jeans”, pode-se destacar que no modelo relacional (lado esquerdo) é modelado a partir das regras de normatização, tornando sua modelagem rígida enquanto que no modelo orientado a documentos (lado direito), os dados seguem um modelo mais flexível. Essa abordagem mais flexível dos bancos de dados orientado a documentos Harrison (2010) escreve que é um fator que atrai desenvolvedores a utilizar esse tipo de banco de dados, pois a facilidade de integrar um sistema desenvolvido numa linguagem orientada a objetos é maior comparado aos banco de dados relacionais e utilizando os bancos de dados relacionais, muitas vezes, o sistema deve se adequar a modelagem rígida dele, tornando improdutivo o trabalho de um desenvolvedor.

3 MÉTODO

Ao definir método, Gil (1999) e Galliano (1979) descrevem como um conjunto de procedimentos baseados em raciocínio lógico, dispostos sobre etapas ordenadamente dispostas para se chegar em determinado objetivo. Nesta seção, descreve-se o tipo de pesquisa do trabalho e o porquê desta classificação, etapas metodológicas, arquitetura da solução e suas delimitações.

3.1 CARACTERIZAÇÃO DO TIPO DE PESQUISA

Ao descrever o que é pesquisa científica, Andrade (2001) descreve como um tipo de pesquisa com base em raciocínio lógico e com um conjunto de procedimentos sistemáticos com objetivo de encontrar soluções para os problemas propostos.

A pesquisa científica pode ser classificada de diversas formas, pela natureza pode ser dividida em básica e aplicada. A pesquisa básica tem como seu objetivo gerar novos conhecimentos sem uma aplicação prática, enquanto que a pesquisa aplicada tem o mesmo objetivo, mas com fins práticos e para solução de determinado problema, logo, a pesquisa aplicada melhor se encaixa neste trabalho devido ao seu objetivo alinhar com este trabalho. (SILVA; MENEZES, 2005) (ALMEIDA, 2011).

Ao analisar a pesquisa pela sua forma de abordagem, pode ser dividida em quantitativa e qualitativa. A pesquisa qualitativa é um tipo de pesquisa descritiva, analisando os dados intuitivamente não requerendo a utilização de métodos e técnicas estatísticas, enquanto que na pesquisa quantitativa considera que todos os dados coletados podem ser quantificados para ser classificados e analisados, utilizando métodos e técnicas estatísticas. (SILVA; MENEZES, 2005).

Do ponto de vista dos objetivos, a pesquisa pode ser dividida em explicativa, que visa a identificar os fatores no qual determinado fato acontece, descritiva que tem como objetivo descrever as características de determinado fato ou o relacionamento entre os fatos, e exploratória visa a estudar profundamente determinado fato, tornando ela explícita ou construir hipóteses na tentativa de explicar o fato. Baseado pelos procedimentos técnicos, é

dividida em pesquisa bibliográfica, documental, experimental, expost-facto, levantamento, estudo de caso, pesquisa-ação e pesquisa participante. (SILVA; MENEZES, 2005) (ALMEIDA, 2011).

Neste trabalho, utiliza-se, do ponto de vista dos objetivos, a pesquisa exploratória, e do ponto de vista dos procedimentos, a pesquisa bibliográfica, pesquisa experimental e estudo de caso, pois envolve pesquisa bibliográfica e uma análise comparativa do uso de um banco de dados relacional e um banco de dados orientado a documentos, tendo como experimento dados geoespaciais. Pela forma de abordagem, será utilizada a quantitativa, pois, a partir de um experimento, utiliza-se o tempo de resposta que cada banco de dados levou para executar o experimento e, assim, analisar o resultado.

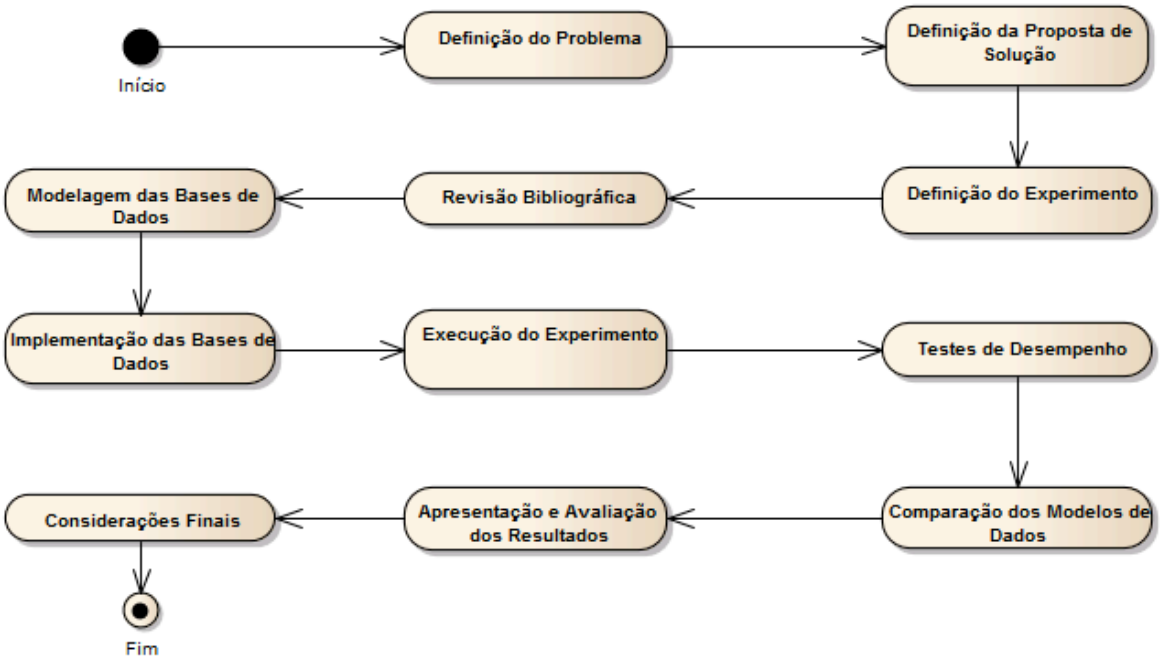
3.2 ETAPAS METODOLÓGICAS

O processo de elaboração deste trabalho se organiza a partir das seguintes etapas:

1. Definição do Problema;
2. Definição da Proposta da Solução;
3. Definição do Experimento;
4. Revisão Bibliográfica;
5. Modelagem das Bases de Dados;
6. Implementação das Bases de Dados;
7. Execução do Experimento;
8. Testes de Desempenho;
9. Comparação dos Modelos de Dados;
10. Apresentação e Avaliação dos Resultados;
11. Considerações Finais;

A figura 15 demonstra um fluxograma das atividades executadas por este trabalho.

Figura 15 – Fluxograma das etapas metodológicas



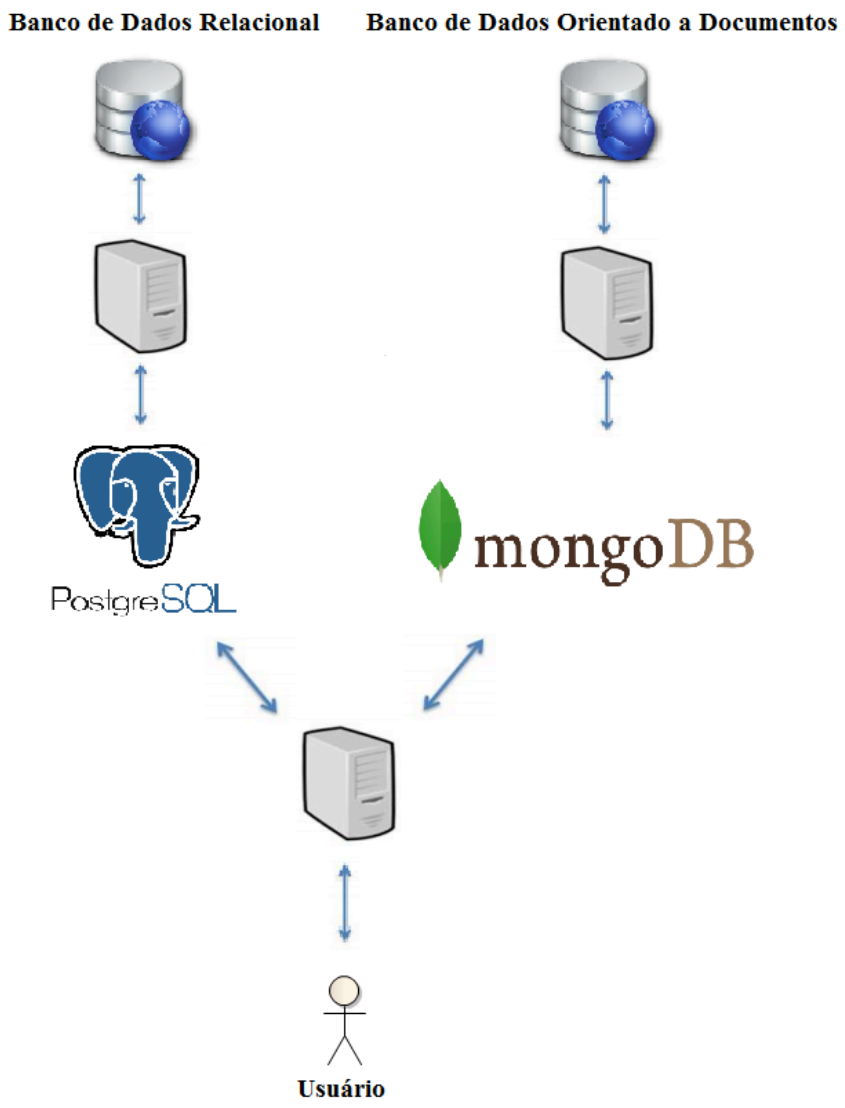
Fonte: O Autor (2015).

A figura 15 demonstra, a partir de um fluxograma, o processo de pesquisa e desenvolvimento deste trabalho.

3.3 DESENHO DA SOLUÇÃO

Considerando os objetivos propostos, o experimento e as tecnologias escolhidas, foi desenvolvida uma proposta da solução, conforme a figura 17.

Figura 17 – Proposta da solução



Fonte: O Autor (2015).

A solução desenvolvida utiliza os dados da malha geométrica dos municípios brasileiros a partir do censo realizado em 2010 pelo Instituto Brasileiro de Geografia e Estatística (IBGE) para o desenvolvimento dos bancos de dados relacional e orientado a documentos. Essas informações relacionadas aos municípios, regiões e estados são modeladas, conforme o modelo de dados relacional e orientado a documentos para seus respectivos bancos de dados utilizados nesta monografia.

O banco de dados relacional utiliza o PostgreSQL com a extensão PostGIS e o banco de dados orientado a documentos utiliza o MongoDB. Para cada banco de dados é criada uma base de dados com as mesmas informações, entretanto, no PostgreSQL, os dados

são modelados a partir da modelagem relacional e, no MongoDB, a modelagem orientada a documentos. Ambos os bancos de dados ficarão na mesma máquina para que tenham as mesmas condições.

Após a conclusão do desenvolvimento das bases de dados, são aplicadas consultas a partir dos casos de uso determinados e, a partir dos resultados obtidos, são analisados e comparados.

3.4 DELIMITAÇÕES

O banco de dados relacional escolhido para ser utilizado é o PostgreSQL, versão 9.4, com a extensão PostGIS para manipulação de dados geoespaciais. O motivo de utilizar esse banco de dados e essa extensão foi devido ao fato de ser o mais utilizado para manipulação de dados geoespaciais, baseados em produtos software-livre/código aberto.

O banco de dados orientado a documentos escolhido é o MongoDB, versão 3.0, sem nenhuma extensão para manipulação de dados geoespaciais, pois já se encontra disponível. O motivo da escolha desse banco foi pela sua flexibilidade na inserção dos dados, facilidade de instalação e suporte a diversos sistemas operacionais.

Não foi desenvolvida nenhuma aplicação para inserção dos registros nas bases de dados. Os dados são provenientes do IBGE a partir do censo realizado em 2010 das delimitações territoriais dos municípios brasileiros. Os registros são inseridos no PostgreSQL através de importação de dados de arquivos chamados *shapefile* (arquivos com as extensões shx, shp, prj e dbf) por meio de uma aplicação disponibilizada pelo mesmo. Os registros são inseridos no MongoDB através de comandos disponibilizados pelo PostgreSQL para criar arquivos JSON e assim realizando a importação desses dados para o banco de dados orientado a documentos escolhido.

4 EXPERIMENTO

Nesta seção, são apresentados os passos realizados no experimento com o objetivo de validar, a partir de determinado cenário, qual tipo de banco de dados tem melhor desempenho com dados geoespaciais. Primeiro são apresentadas as ferramentas utilizadas para o experimento, seguido do cenário do experimento, preparação das bases de dados para ambos banco de dados, realização do experimento e apresentação do resultado.

4.1 FERRAMENTAS

Nesta seção, são apresentadas as ferramentas e tecnologias utilizadas no desenvolvimento do experimento desta monografia. A figura 18 demonstra as ferramentas e tecnologias utilizadas.

Figura 18 – Ferramentas e tecnologias utilizadas.

Fonte: O Autor (2015).

As ferramentas e tecnologias apresentadas na figura 18 foram utilizadas para a captura, transformação dos dados para ambas as bases de dados e execução do experimento.

4.1.1 PostgreSQL

Segundo Bonfioli (2006, p. 20), o PostgreSQL é um SGBD do tipo de modelo de dados objeto-relacional, descendente do Postgress, com suporte as linguagens SQL92/SQL99 desenvolvido no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley com o código fonte aberto.

O PostgreSQL possui diversas funcionalidades que foram incorporadas ao longo do tempo e essas características, por serem muitas, são divididas em grupos que, segundo PostgreSQL (2014, tradução nossa) se define em: backend, perfomance, segurança, rede, internacionalização, plataforma, tipos de dados, funções e gatilhos, linguagens procedurais, módulos adicionais.

4.1.2 PostGIS

PostGIS é um módulo com o objetivo de adicionar funcionalidades na manipulação e armazenamentos de dados geográficos no banco de dados PostgreSQL, diferente do PostgreSQL que também realiza esse trabalho, o PosGIS também implementa funcionalidades topológicas, seguindo a especificação SFS (Simple Features Specification) do consórcio OGC. Para o tratamento de grandes volumes de dados geoespaciais, o PostGIS implementa a indexação dos dados pelo Rtree sobre a indexação GiST (Generalized Search Trees) nativa do PostgreSQL, mesmo possuindo três tipos de indexação (B-Tree, R-Tree e GiST), elas não apresentam a robustez necessária para a manipulação e armazenamentos de dados geoespaciais (JÚNIOR, 2010).

4.1.3 MongoDB

MongoDB é um banco de dados do tipo orientado a documentos, sendo o foco desse banco a estrutura de um documento comparado ao banco de dados relacional de uma tabela, voltado a aplicações que utilizam a infraestrutura da internet, suporte à grande volume de leitura e escrita, fácil de ser escalável, flexível e tolerante a falhas (BARASUOL, 2012).

O MongoDB foi criado por volta de 2007 pela empresa 10gen a partir de um software tipo plataforma como serviço, sendo composta por uma aplicação rodando em determinado servidor e uma base de dados, hospedando aplicações web e podendo ser realizado o escalonamento, quando for necessário, entretanto, a própria empresa percebeu que

seus usuários estavam descontentes por perder o controle sobre as tecnologias nas quais utilizadas, querendo uma nova base de dados, portanto, ela direcionou seus esforços em criar uma de uma base de dados que atendessem a demanda, chamado MongoDB.

4.1.4 GeoKettle

GeoKettle é uma ferramenta do tipo ETL7, criada pela Spatialytics, baseada na ferramenta Kettle mantida pela Pentaho Data Integration, entretanto, o GeoKettle tem como foco na manipulação de dados geográficos devido a ausência de suporte a esse tipo de dado pela ferramenta na qual ela foi baseada. (GEOKETTLE, 2014).

A ferramenta GeoKettle é mantida sob a licença LGPL, portanto, o produto de código aberto e sem custo. Por ter o foco para dados geográficos, o Geokettle fornece suporte a várias bibliotecas de código aberto (JTS, GeoTools, OGR) e plugins (através do Sextante). (GEOKETTLE, 2014).

4.1.5 GeoJSON

GeoJson é um formato com o objetivo de codificar estruturas de dados geográficos (ponto, linha, polígonos, entre outros) baseado no formato Json. Esse formato foi criado em 2008 por Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub, Christopher Schmidt. (GEOJSON, 2015).

Registrado no IETF[8], está na sua 5ª versão com data de criação de 28 de janeiro de 2015, com data de expiração para 1º de agosto de 2015. (THE GEOJSON, 2015).

---

[7] ETL (*Extract, Transformation* e *Load*) é um termo utilizado para referenciar ferramentas com o objetivo de extrair dados de determinada fonte de dados, transformar esses dados e carregar ela para outras fontes de dados. (BEAL, 2015)

[8] IETF (*Internet Engineering Task Force*) é uma comunidade internacional que definem a padronização de protocolos utilizados na internet. (IETF, 2015)

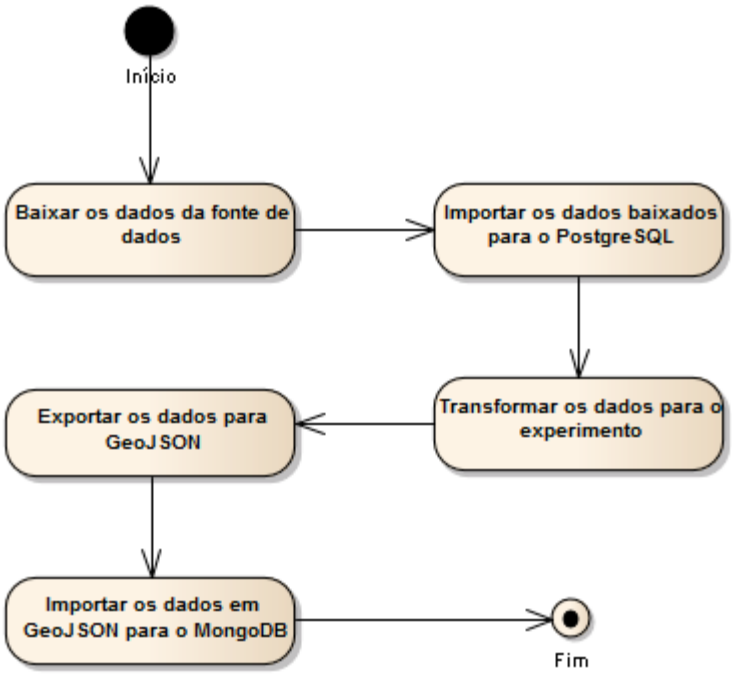
4.2 CENÁRIO DO EXPERIMENTO

Nesta monografia, são utilizados como cenário da aplicação os dados do censo demográfico realizado pelo IBGE no ano de 2010 com as demarcações territoriais do Brasil, sendo apresentadas todas as unidades territoriais a partir de municípios, microrregiões, mesorregiões e unidades da federação. (MALHA, 2015). Esses dados são utilizados neste trabalho devido a ser disponibilizado de uma fonte de dados pública, entretanto, esses dados para esse experimento devem ser realizados em um trabalho de conversão e armazenamento dos dados nos bancos de dados utilizados para o experimento deste trabalho.

No experimento, com ambos os bancos de dados já com os dados armazenados neles, são descritos determinados cenários de consultas envolvendo tipos de dados geográficos (ponto, linha e polígono) para cada tipo de dados armazenado nos bancos de dados (municípios, microrregiões, mesorregiões e estados), logo, com as consultas realizadas, é obtido o tempo de resposta que cada banco de dados levou para retornar as informações e assim realizar conclusões baseadas nesses resultados.

A figura 19 apresenta os passos a partir de um fluxograma os passos a ser realizados para montagem do cenário do experimento.

Figura 19 – Cenário do experimento.



Fonte: O Autor (2015)

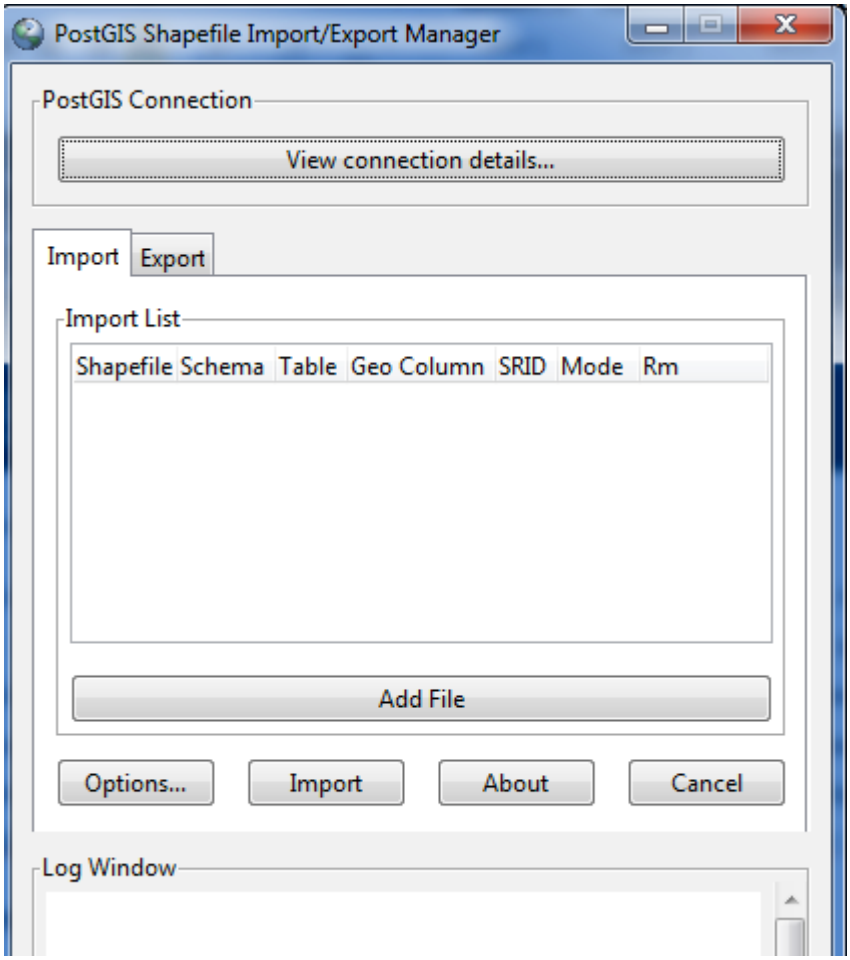
A fonte de dados utilizada neste trabalho disponibiliza esses dados de cada tipo de representação territorial em arquivos do tipo shapefile, entretanto são agrupados por estados e colocados dentro de um arquivo do tipo zip, para facilitar a disponibilidade desses dados. Depois, para realizar a exportação dos dados para o banco de dados relacional (PostgreSQL), é utilizado um programa disponibilizado pelo PostGIS chamado PostGIS Shapefile Import/Export Manager e armazenado esses dados no PostgreSQL, posteriormente, é realizado a transformação dos dados, utilizando a ferramenta GeoKettle, armazenando em outro banco de dados dentro do PostgreSQL, com os dados armazenados no PostgreSQL, é utilizando novamente o GeoKettle para exportar os dados para um arquivo GeoJSON e, por fim, exportando esse tipo de arquivo para o MongoDB.

4.3 PREPARAÇÃO DAS BASES DE DADOS

Com os arquivos shapefile baixados da fonte de dados escolhida, o primeiro passo é realizar a importação desses dados para um banco de dados, no qual nesse passo será o PostgreSQL. (MALHA, 2015). Antes de realizar a importação, é necessário criar a base de dados para armazenar esses dados, logo para realizar esse procedimento, é necessário acessar o banco de dados com um usuário que tenha permissão de criar base de dados e realizar os seguintes passos conforme apresentado no apêndice A.

Conforme apresentado no apêndice A, além da criação da base de dados, é habilitado à extensão que adiciona funcionalidades geoespaciais para a base de dados criada. Após a criação da base de dados, agora é necessário importar os dados em arquivos shapefile para dentro da base de dados criada anteriormente, logo para realizar essa tarefa é utilizado um programa disponibilizado pelo PostGIS para realizar essa tarefa, chamado PostGIS Shapefile Import/Export Manager. A figura 21 apresenta a interface gráfica do programa mencionado anteriormente.

Figura 21 – Interface gráfica do PostGIS Shapefile Import/Export Manager.



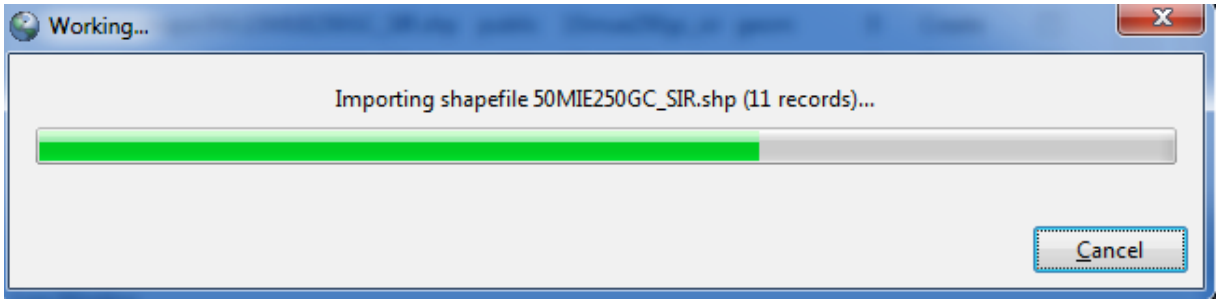
Fonte: O Autor (2015)

A figura 21 apresenta a tela inicial do programa que realiza o processo de capturar as informações dos arquivos shapefile e armazenar numa base de dados relacional (no caso deste trabalho, dentro do PostgreSQL numa base de dados com o PostGIS habilitado).

Para realizar o processo de importar as informações para a base de dados, é necessário configurar os dados de conexão com o banco de dados, digitando valores como o endereço do servidor, porta, base de dados, usuário e senha, sendo realizado esse procedimento clicando no botão View connection details. O passo seguinte é necessário adicionar os arquivos shapefile na lista de arquivos para realização da importação dos dados, logo para realizar esse procedimento, é necessário clicar no botão Add File. O próximo passo é configurar a importação considerar que os arquivos adicionados estão codificados a partir do LATIN1 e habilitar as opções “Do not create ‘bigint’ columns”, “Create spatial index automatically after load”, ”Load data using COPY rather than INSERT”, logo esses procedimentos é realizado ao clicar no botão Options. O último passo é clicar no botão Import

e o programa inicializa o processo de importação de dados. A figura 22 apresenta o programa realizando a importação dos dados para a base de dados escolhida.

Figura 22 – PostGIS Shapefile Import/Export Manager realizando o processo de importação.



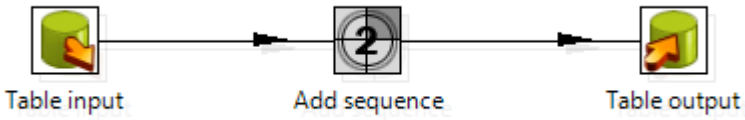
Fonte: O Autor (2015)

Conforme apresentado na figura 22, após realizar os passos de configuração, o programa mostra o processo da importação dos dados com o nome do arquivo que está sendo importado e a quantidade de registros que contém no arquivo.

Com os dados armazenados na base de dados relacionais, agora é necessário realizar a transformação dos dados para que o modelo de dados de ambas as bases sejam mais similar possível para que o experimento seja imparcial. Para realizar esse processo de transformação dos dados, é necessário criar outra base de dados relacional para armazenar esses dados, logo para realizar esse processo, o apêndice B apresenta os comandos SQL para criação da base de dados.

Com o banco de dados criado, é necessário realizar a transformação dos dados para a nova base de dados, logo para essa tarefa, é utilizado o programa chamado GeoKettle. Os procedimentos para realizar a transformação dos dados são na sua maioria parecidos, na figura 23 é demonstrado o fluxograma criado no Geokettle para essa tarefa.

Figura 23 – Fluxograma no Geokettle que realiza a transformação dos dados.



Fonte: O Autor (2015)

Conforme apresentado na figura 23, o fluxograma utilizado neste trabalho para realizar a transformação dos dados. Na tarefa chamada “Table input” é realizado a consulta na

base de dados shapefile_brasil utilizando comandos SQL conforme no apêndice C. No próximo passo, chamado “Add sequence” não é realizada nenhuma configuração especial, entretanto, na tarefa “Table output”, além de configurar a conexão com a base de dados que armazena os dados transformados, na hora de realizar o mapeamento das colunas, apontar a coluna “valuename” para “gid”.

Após realizar a transformação dos dados, agora é necessário armazenar os mesmos dados no banco de dados orientado a documento, no caso deste trabalho, é armazenado no banco de dados MongoDB. Para realizar esse procedimento, é necessário primeiro exportar os dados que estão na base de dados relacional com os dados transformados para GeoJSON, pois o MongoDB aceita somente importação de dados nos padrões JSON. A exportação dos dados para GeoJSON é utilizado um comando SQL que monta os arquivos de estados, micro regiões, meso regiões e municípios conforme apêndice D. Devido a problemas em determinados dados geoespaciais, foi necessário remover um registro de munícipio, no qual o apêndice E é utilizado para remover de ambas as bases.

Após exportar os dados para arquivos GeoJSON, é necessário importar esses dados para o MongoDB, logo para realizar essa tarefa é necessário executar determinados comandos conforme apresentados no apêndice F. Com os dados exportados para o MongoDB, é necessários criar os índices nas propriedades que estão as coordenadas geográficas para ser realizado as consultas geoespaciais, logo o apêndice G apresenta os comandos necessários para realizar esta tarefa.

4.4 EXECUÇÃO DO EXPERIMENTO

A execução do experimento desta monografia deve-se basear no tempo de resposta dos seguintes requisitos descritos a seguir:

- RF001 - A partir de determinado ponto especificado, retornar qual município está naquele ponto.
- RF002 - A partir de determinada linha especificada, retornar quais municípios estão naquela linha.

- RF003 - A partir de determinado polígono especificado, retornar quais municípios contém dentro desse polígono.
- RF004 - A partir de determinado ponto especificado, retornar qual meso região está naquele ponto.
- RF005 - A partir de determinada linha especificada, retornar quais meso regiões estão naquela linha.
- RF006 - A partir de determinado polígono especificado, retornar quais meso regiões contém dentro desse polígono.
- RF007 - A partir de determinado ponto especificado, retornar qual micro região está naquele ponto.
- RF008 - A partir de determinada linha especificada, retornar quais micro regiões estão naquela linha.
- RF009 - A partir de determinado polígono especificado, retornar quais micro regiões contém dentro desse polígono.
- RF010 - A partir de determinado ponto especificado, retornar qual município está naquele ponto.
- RF011 - A partir de determinada linha especificada, retornar quais municípios estão naquela linha.
- RF012 - A partir de determinado polígono especificado, retornar quais municípios contém dentro desse polígono.

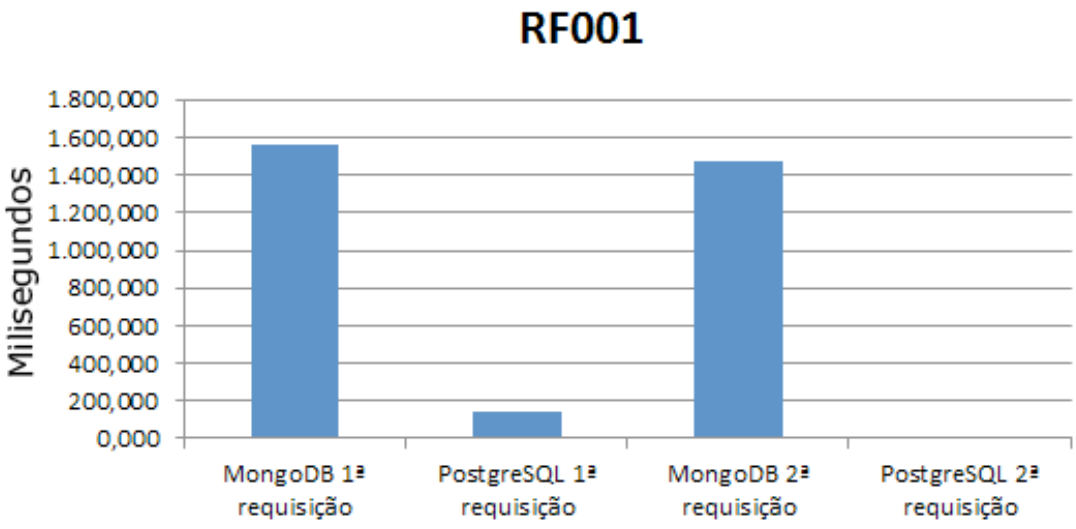
Para cada requisito descrito anteriormente, ambas as bases de dados devem ter funcionalidades que permitam que elas sejam realizadas. Neste experimento, o sistema de coordenadas geográficas utilizado nos dados foi o WGS 84, com o objetivo de que os dados retornados em ambas as bases sejam as mesmas.

Nos requisitos que utiliza o tipo de seleção ponto, a coordenada geográfica escolhida (longitude e latitude) é -48.56146 e -27.54014. Nos requisitos que utiliza o tipo de seleção linha, a coordenada geográfica escolhida é -53.63637 e -26.31376, -51.17683 e -26.79483, -50.58058 e -28.02121, -48.70375 e -26.20535. Nos requisitos que utiliza o tipo de seleção polígono, a coordenada geográfica escolhida é -51.55627 e -26.75418, -48.98832 e -26.97099, -49.02220 e -28.59713, -49.02220 e -28.59713, -51.55627 e -26.75418.

4.4.1 RF001 - A partir de determinado ponto especificado, retornar qual município está naquele ponto.

Neste item do experimento, o resultado deve retornar somente um único registro, no caso dos estados, retorna Santa Catarina. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice H, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 1 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 1 – Resultado das consultas dos bancos de dados para realizar a RF001



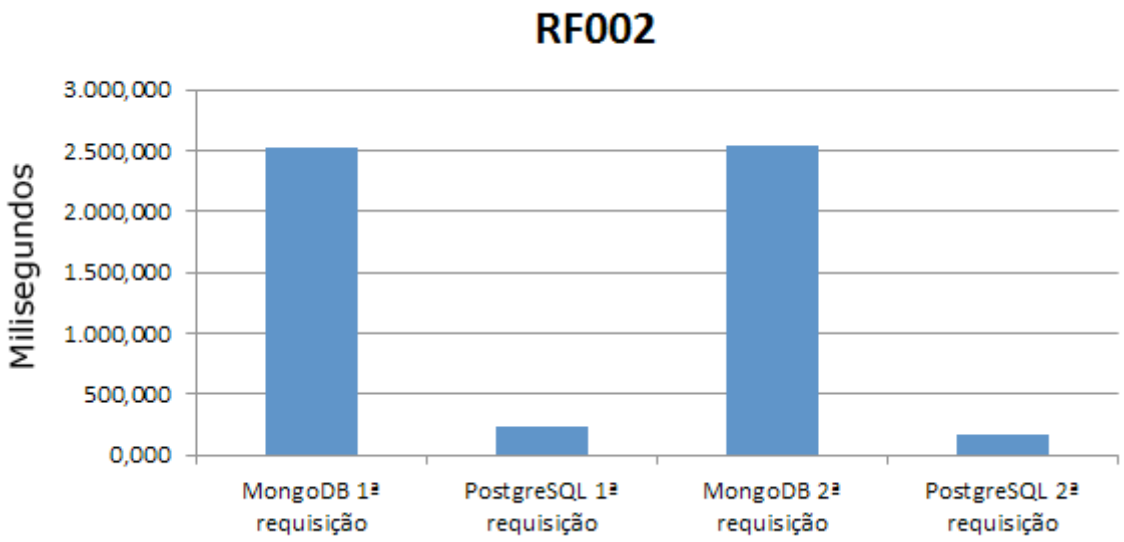
Fonte: O Autor (2015)

Conforme apresentado no gráfico 1, no MongoDB na primeira requisição o tempo de resposta foi de 1556 milissegundos, na segunda requisição foi de 1478 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 137,47 milissegundos, na segunda consulta o tempo de resposta diminuiu para 9,036 milissegundos.

4.4.2 RF002 - A partir de determinada linha especificada, retornar quais municípios estão naquela linha.

Neste item do experimento, o resultado deve retornar somente um único registro, no caso dos estados, retorna Santa Catarina, mesmo o tipo de seleção sendo uma linha. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice I, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 2 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 2 – Resultado das consultas dos bancos de dados para realizar a RF002



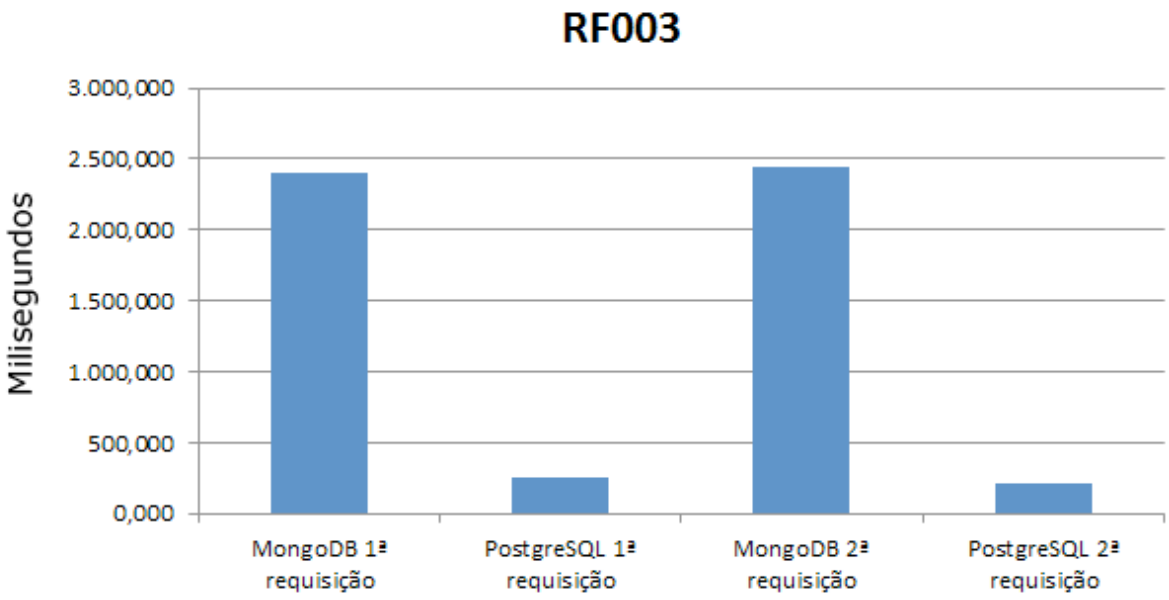
Fonte: O Autor (2015)

Conforme apresentado no gráfico 2, no MongoDB na primeira requisição o tempo de resposta foi de 2520 milissegundos, na segunda requisição foi de 2543 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 228,104 milissegundos, na segunda consulta o tempo de resposta diminuiu para 178,188 milissegundos.

4.4.3 RF003 - A partir de determinado polígono especificado, retornar quais municípios contém dentro desse polígono.

Neste item do experimento, o resultado deve retornar somente um único registro, no caso dos estados, retorna Santa Catarina, mesmo o tipo de seleção sendo um polígono. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice J, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 3 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 3 – Resultado das consultas dos bancos de dados para realizar a RF003



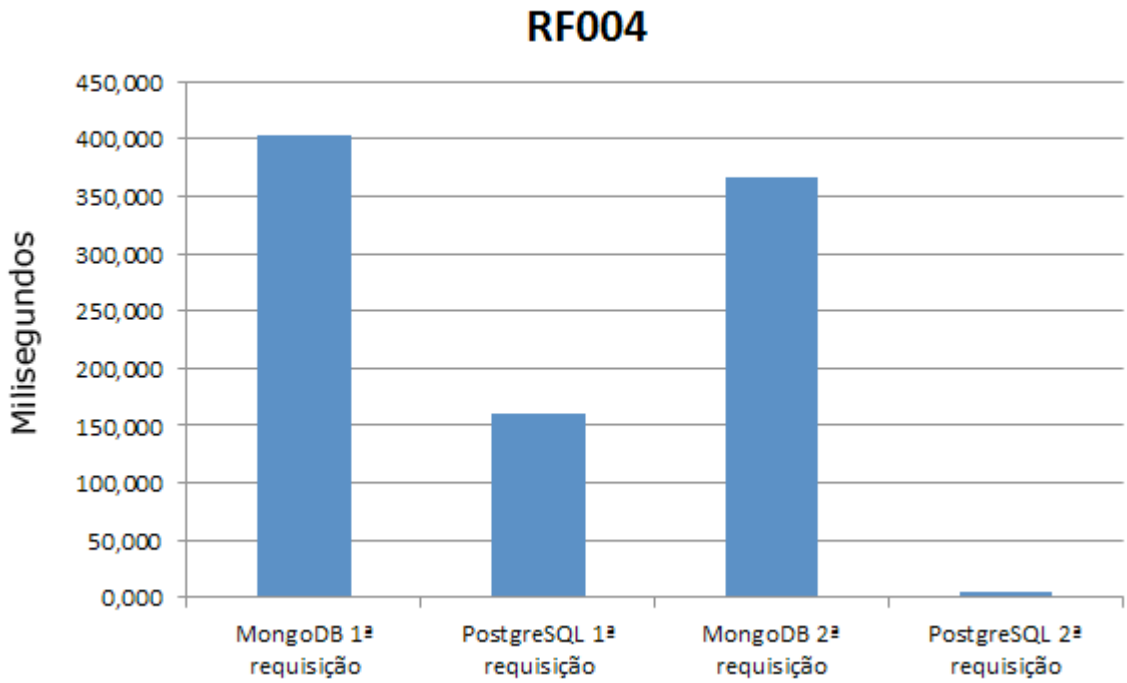
Fonte: O Autor (2015)

Conforme apresentado no gráfico 3, no MongoDB na primeira requisição o tempo de resposta foi de 2398 milissegundos, na segunda requisição foi de 2443 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 256,707 milissegundos, na segunda consulta o tempo de resposta diminuiu para 214,577 milissegundos.

4.4.4 RF004 - A partir de determinado ponto especificado, retornar qual meso região está naquele ponto.

Neste item do experimento, o resultado deve retornar somente um único registro, no caso das meso regiões, retorna Grande Florianópolis. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice K, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 4 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 4 – Resultado das consultas dos bancos de dados para realizar a RF004



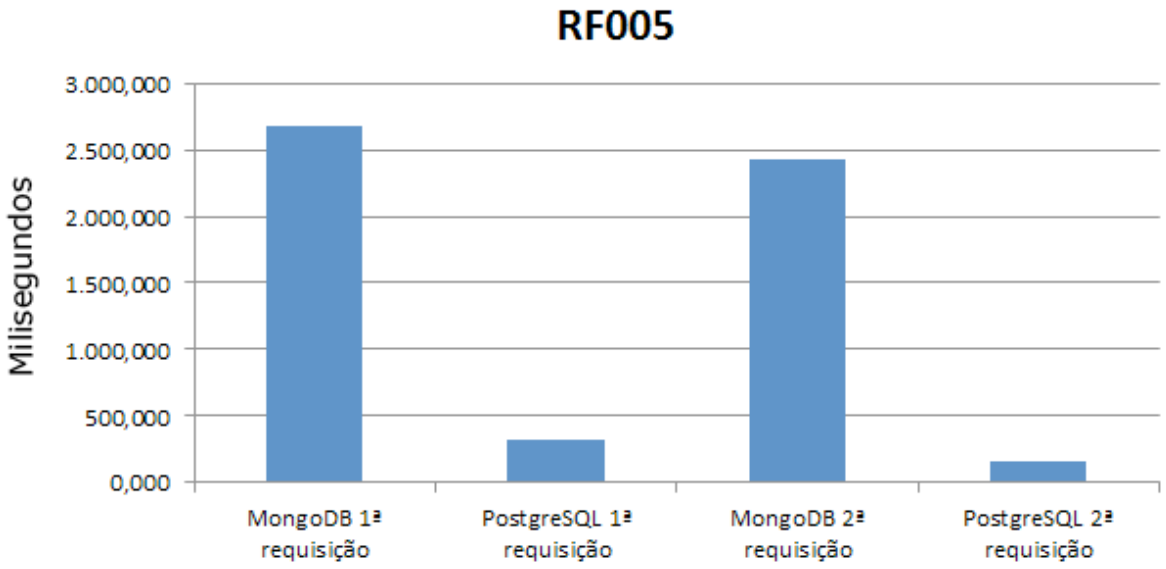
Fonte: O Autor (2015)

Conforme apresentado no gráfico 4, no MongoDB na primeira requisição o tempo de resposta foi de 403 milissegundos, na segunda requisição foi de 367 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 160,155 milissegundos, na segunda consulta o tempo de resposta diminuiu para 4,229 milissegundos.

4.4.5 RF005 - A partir de determinada linha especificada, retornar quais meso regiões estão naquela linha.

Neste item do experimento, o resultado deve retornar 5 registros, Vale do Itajaí, Grande Florianópolis, Norte Catarinense, Serrana, Oeste Catarinense. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice L, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 5 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 5 – Resultado das consultas dos bancos de dados para realizar a RF005



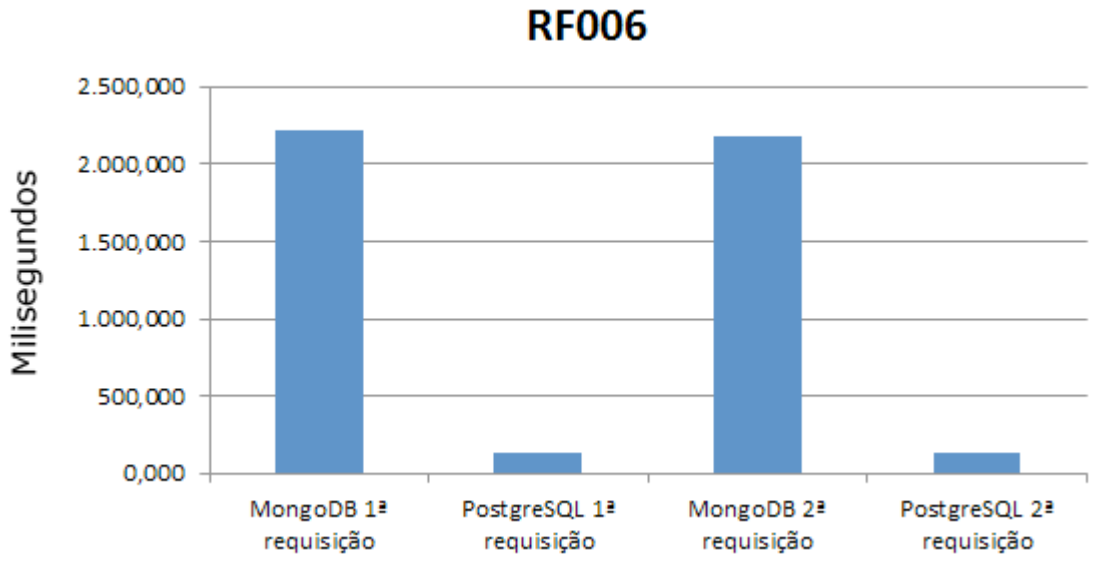
Fonte: O Autor (2015)

Conforme apresentado no gráfico 5, no MongoDB na primeira requisição o tempo de resposta foi de 2675 milissegundos, na segunda requisição foi de 2429 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 328,183 milissegundos, na segunda consulta o tempo de resposta diminuiu para 160,287 milissegundos.

4.4.6 RF006 - A partir de determinado polígono especificado, retornar quais meso regiões contém dentro desse polígono.

Neste item do experimento, o resultado deve retornar 5 registros, Vale do Itajaí, Grande Florianópolis, Serrana, Oeste Catarinense, Sul Catarinense. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice M, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 6 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 6 – Resultado das consultas dos bancos de dados para realizar a RF006



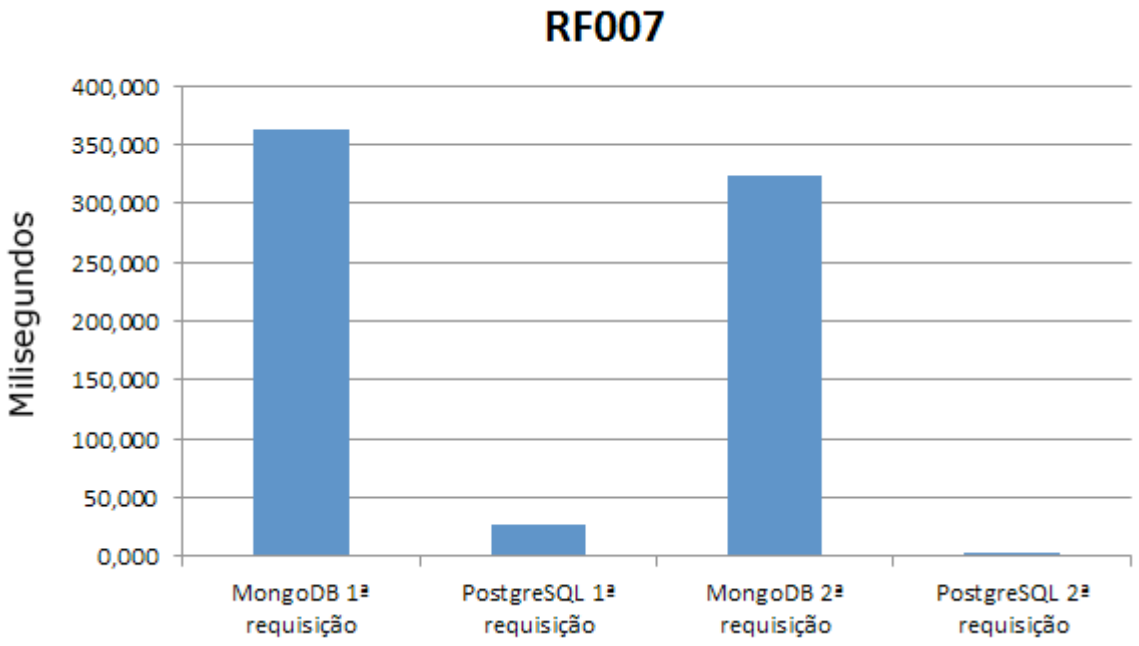
Fonte: O Autor (2015)

Conforme apresentado no gráfico 6, no MongoDB na primeira requisição o tempo de resposta foi de 2211 milissegundos, na segunda requisição foi de 2211 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 128,562 milissegundos, na segunda consulta o tempo de resposta diminuiu para 132,924 milissegundos.

4.4.7 RF007 - A partir de determinado ponto especificado, retornar qual micro região está naquele ponto.

Neste item do experimento, o resultado deve retornar somente um único registro, no caso das micros regiões, retorna Florianópolis. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice N, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 7 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 7 – Resultado das consultas dos bancos de dados para realizar a RF007



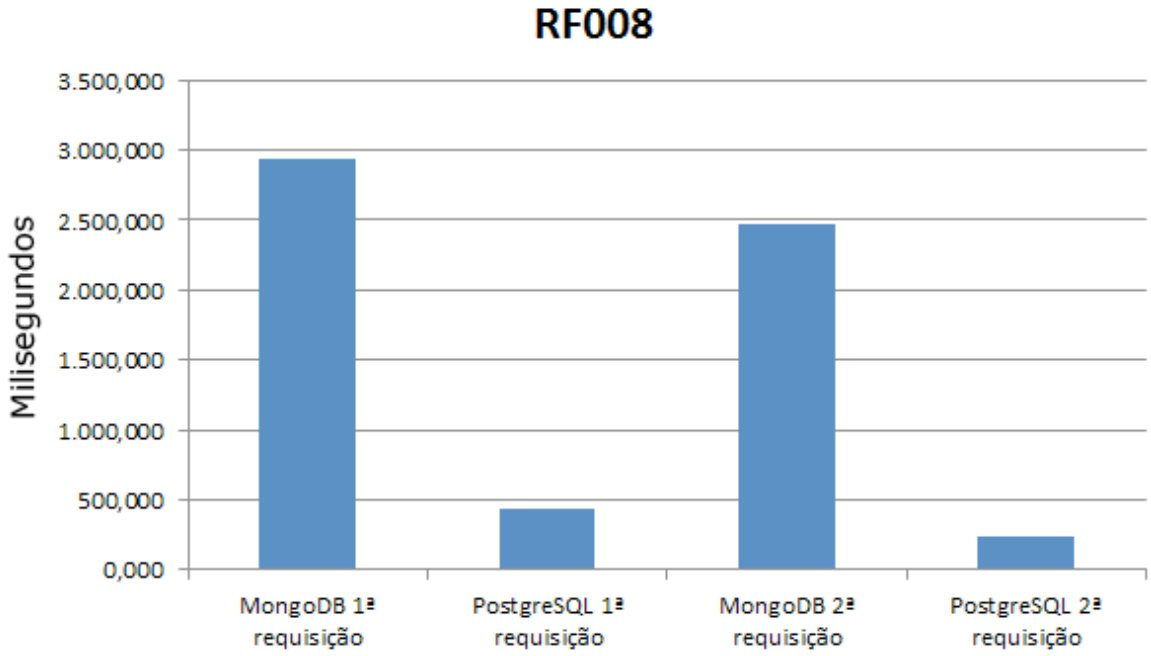
Fonte: O Autor (2015)

Conforme apresentado no gráfico 7, no MongoDB na primeira requisição o tempo de resposta foi de 363 milissegundos, na segunda requisição foi de 323 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 26,919 milissegundos, na segunda consulta o tempo de resposta diminuiu para 2,685 milissegundos.

4.4.8 RF008 - A partir de determinada linha especificada, retornar quais micro regiões estão naquela linha.

Neste item do experimento, o resultado deve retornar 9 registros. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice O, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 8 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 8 – Resultado das consultas dos bancos de dados para realizar a RF008



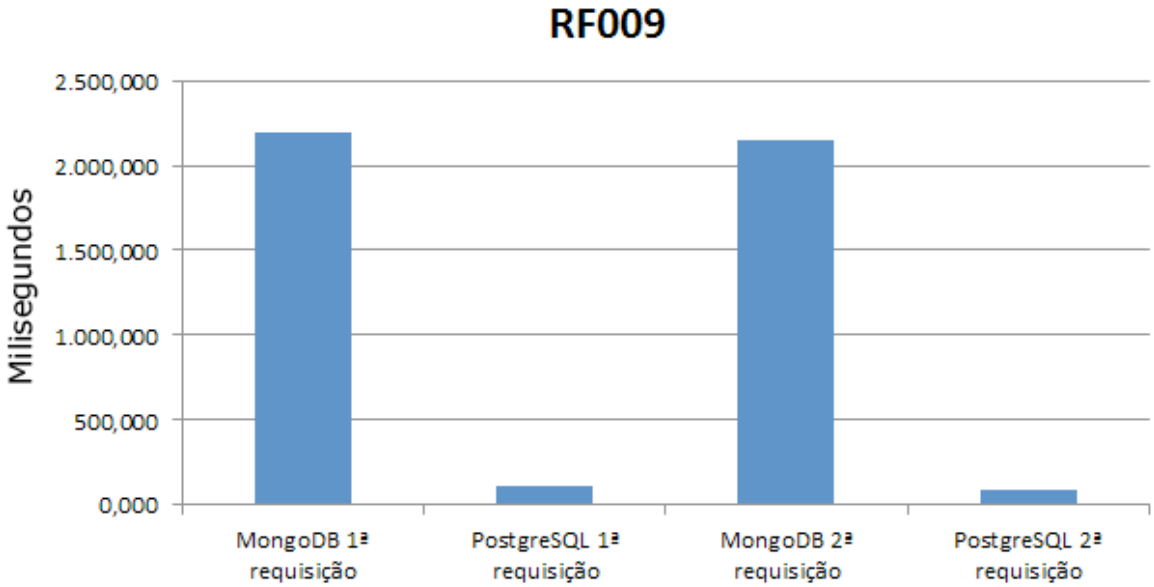
Fonte: O Autor (2015)

Conforme apresentado no gráfico 8, no MongoDB na primeira requisição o tempo de resposta foi de 2932 milissegundos, na segunda requisição foi de 2477 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 438,527 milissegundos, na segunda consulta o tempo de resposta diminuiu para 230,715 milissegundos.

4.4.9 RF009 - A partir de determinado polígono especificado, retornar quais micro regiões contém dentro desse polígono.

Neste item do experimento, o resultado deve retornar 10 registros. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice P, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 9 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 9 – Resultado das consultas dos bancos de dados para realizar a RF009



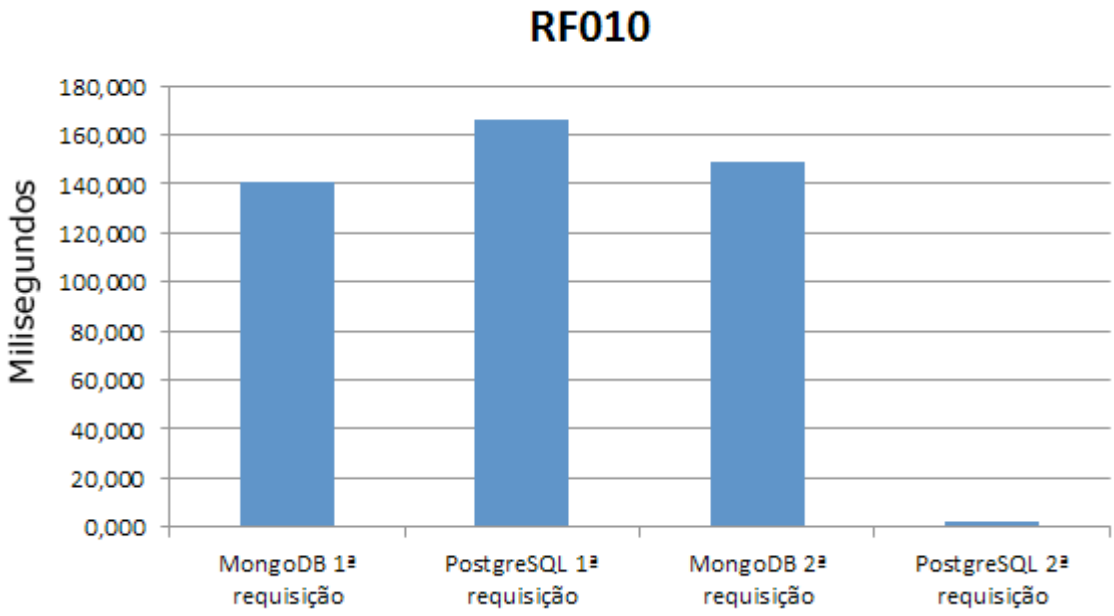
Fonte: O Autor (2015)

Conforme apresentado no gráfico 9, no MongoDB na primeira requisição o tempo de resposta foi de 2197 milissegundos, na segunda requisição foi de 2150 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 106,552 milissegundos, na segunda consulta o tempo de resposta diminuiu para 94,638 milissegundos.

4.4.10 RF010 - A partir de determinado ponto especificado, retornar qual município está naquele ponto.

Neste item do experimento, o resultado deve retornar somente um único registro, no caso das micros regiões, retorna Florianópolis. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice Q, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 10 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 10 – Resultado das consultas dos bancos de dados para realizar a RF010



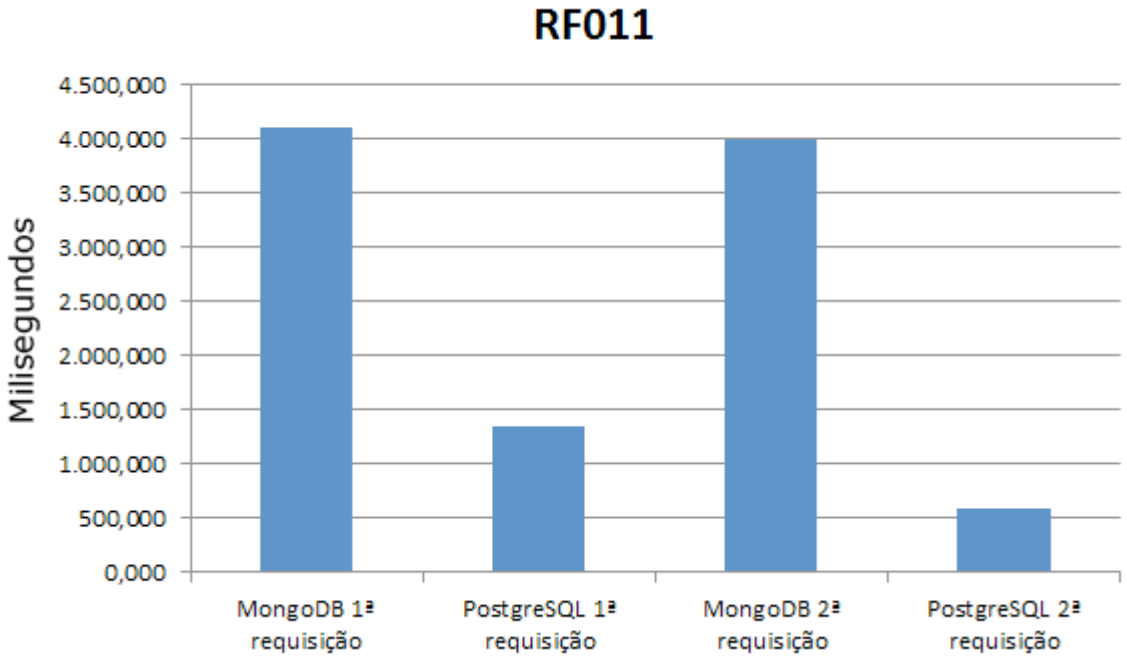
Fonte: O Autor (2015)

Conforme apresentado no gráfico 10, no MongoDB na primeira requisição o tempo de resposta foi de 141 milissegundos, na segunda requisição foi de 149 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 166,31 milissegundos, na segunda consulta o tempo de resposta diminuiu para 2,142 milissegundos.

4.4.11 RF011 - A partir de determinada linha especificada, retornar quais municípios estão naquela linha.

Neste item do experimento, o resultado deve retornar 43 registros. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice R, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 11 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 11 – Resultado das consultas dos bancos de dados para realizar a RF011



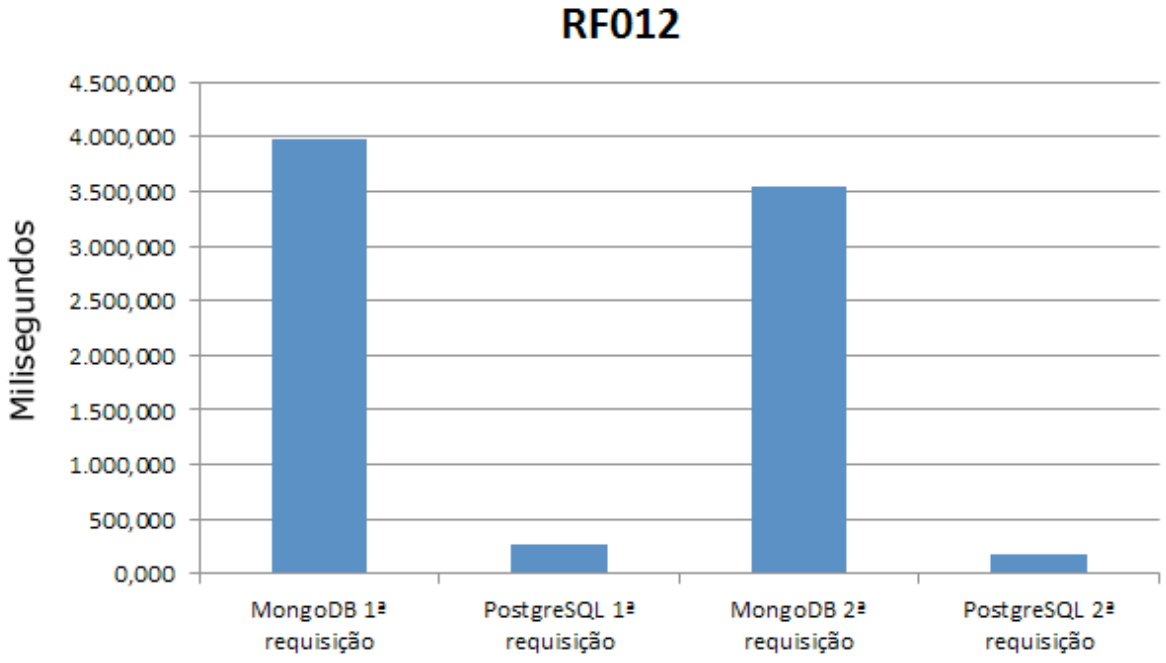
Fonte: O Autor (2015)

Conforme apresentado no gráfico 11, no MongoDB na primeira requisição o tempo de resposta foi de 4101 milissegundos, na segunda requisição foi de 3990 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 1352,57 milissegundos, na segunda consulta o tempo de resposta diminuiu para 594,655 milissegundos.

4.4.12 RF012 - A partir de determinado polígono especificado, retornar quais municípios contém dentro desse polígono.

Neste item do experimento, o resultado deve retornar 92 registros. A consulta realizada para resolver esse requisito funcional tanto para o PostgreSQL quanto MongoDB. No apêndice S, são apresentados os comandos executados para realizar a consulta em ambos os bancos de dados. O gráfico 12 apresenta o tempo de resposta de ambos os bancos de dados.

Gráfico 12 – Resultado das consultas dos bancos de dados para realizar a RF012



Fonte: O Autor (2015)

Conforme apresentado no gráfico 12, no MongoDB na primeira requisição o tempo de resposta foi de 3980 milissegundos, na segunda requisição foi de 3537 milissegundos. No PostgreSQL o tempo de resposta na primeira requisição foi de 268,914 milissegundos, na segunda consulta o tempo de resposta diminuiu para 172,143 milissegundos.

4.5 RESULTADO

Nesta seção, são apresentados os resultados referentes à execução do experimento, conforme tabela 1.

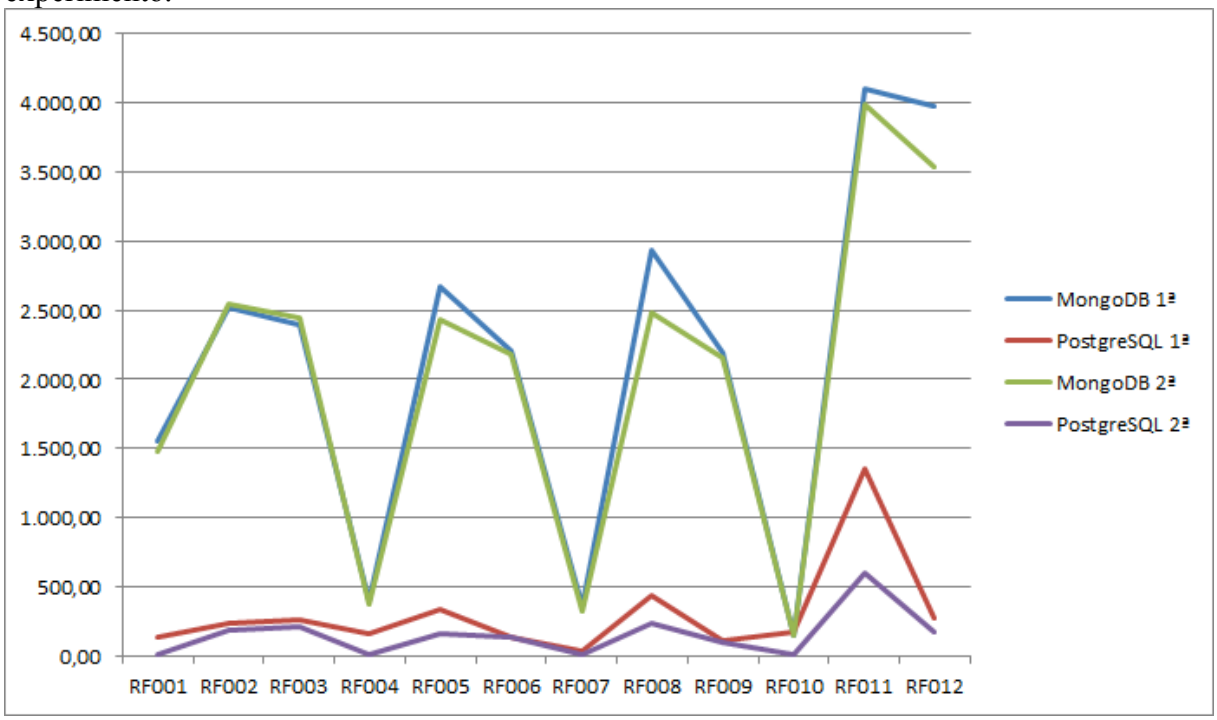
Tabela 1 – Resultados da execução do experimento no MongoDB e PostgreSQL em milissegundos

	MongoDB 1ª	**PostgreSQL 1ª**	**MongoDB 2ª**	**PostgreSQL 2ª**
RF001	1.556,000	137,470	1.478,000	9,036
RF002	2.520,000	228,104	2.543,000	178,188
RF003	2.398,000	256,707	2.443,000	214,577
RF004	403,000	160,155	367,000	4,229
RF005	2.675,000	328,183	2.429,000	160,287
RF006	2.211,000	128,562	2.184,000	132,924
RF007	363,000	26,919	323,000	2,685
RF008	2.932,000	438,527	2.477,000	230,715
RF009	2.197,000	106,552	2.150,000	94,638
RF010	141,000	166,310	149,000	2,142
RF011	4.101,000	1.352,570	3.990,000	594,655
RF012	3.980,000	268,914	3.537,000	172,143

Fonte: O Autor (2015)

A tabela 1, conforme apresentado anteriormente, apresenta os resultados do tempo de resposta em milissegundos do experimento executado. O gráfico 13 apresenta uma comparação entre os resultados.

Gráfico 13 – Comparação dos tempos de resposta de todas as requisições realizadas no experimento.



Fonte: O Autor (2015)

O gráfico 13 apresenta uma comparação dos resultados coletados no experimento, baseado no tempo de resposta em milissegundos para cada requisito executado nas duas requisições executadas para cada banco.

Conforme apresentado na tabela 1 e gráfico 13, dos bancos de dados MongoDB e PostgreSQL, pode-se concluir que:

- O banco de dados PostgreSQL apresenta o tempo de resposta menor na maioria dos experimentos realizados comparado ao MongoDB.
- A maioria dos experimentos que utilizam a linha como seleção, tem o tempo de execução maior para ambo banco de dados.
- Na maioria das segundas requisições do PostgreSQL teve seu tempo de requisição muito menor comparado a primeira requisição, comparado ao MongoDB que apresenta resultado mais “estáveis”.
- Nos requisitos que utiliza ponto como seleção, o banco de dados PostgreSQL apresentou em média ser 58,225 vezes mais rápido comparado ao MongoDB.
- Nos requisitos que utiliza linha como seleção, o banco de dados PostgreSQL apresentou em média ser 9,5 vezes mais rápido comparado ao MongoDB.

- Nos requisitos que utiliza polígono como seleção, o banco de dados PostgreSQL apresentou em média ser 16,5 vezes mais rápido comparado ao MongoDB.
- O banco de dados PostgreSQL no geral, apresentou ser 28,075 vezes mais rápido comparado ao MongoDB.

A partir das conclusões baseadas na tabela 1 descritas anteriormente, a próxima seção deve descrever a conclusão geral desta monografia.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou sobre os temas sistemas de informação geográfica, banco de dados relacional e banco de dados orientado a documentos, tendo como objetivo principal comparar os dois tipos de banco de dados, quais deles tem melhor desempenho focado em dados geoespaciais.

Conforme apresentado neste trabalho, foi apresentado inicialmente sobre sistemas de informação geográficos, como um conjunto de sistema com o objetivo de realizar a manipulação de dados geoespaciais, tendo como objetivo no auxílio a partir desse tipo de dados, nas tomadas de decisões. Posteriormente, foi descrito sobre banco de dados, especialmente sobre os bancos de dados relacionais e banco de dados orientado a documentos, com o foco no modelo de dados.

Com os temas apresentados, logo dá-se início ao desenvolvimento do experimento, conforme descrito nesta monografia para chegar na resposta da pergunta de pesquisa. O experimento realizado nesta monografia utilizou como fonte de dados a malha geográfica do Brasil, sendo disponibilizado pelo IBGE no censo realizado em 2010. Nessa fonte de dados, os dados são divididos em quatro categorias, estados, meso regiões, micro regiões e municípios. A partir desses dados, o experimento foi realizado usando estes dados, e para realizar a comparação foram utilizados os tipos de seleções geográficas ponto, linha e polígono.

Executado o experimento foi realizada uma análise sobre os dados coletados do tempo de resposta de ambo bancos de dados, logo pode-se realizar várias conclusões, como o tempo de resposta do PostgreSQL retornar tempos de respostas mais “estáveis” comparado ao MongoDB, a utilização de linha tem o tempo de resposta maior comparado ao polígono e apresentar o PostgreSQL ser 28,075 mais rápido comparado ao MongoDB.

Entretanto, na preparação das bases para a realização do experimento, o autor desta monografia teve algumas dificuldades em relação a importação dos dados disponibilizado pelo IBGE para o PostgreSQL, entre eles o fato dos arquivos não ter definido o sistema de coordenadas de referências, no qual nesta monografia foi utilizada A WGS84. Outra dificuldade foi devido ao fato da formatação dos arquivos GeoJSON para ser realizado a importação desses dados para o MongoDB. Para contornar esse erro foi necessário criar os GeoJSON via comando SQL apresentados na seção de preparação das bases de dados.

Na seção problemática foi descrito a importância de todas as áreas descritas anteriormente, e baseado na pergunta de pesquisa desta monografia: a utilização de banco de dados orientado a documentos pode ser uma boa alternativa para trabalhar com dados geoespaciais comparados aos bancos de dados relacionais? Com o resultado do experimento pode-se concluir que, o banco de dados orientado a documento pode não ser uma boa alternativa para trabalhar com dados geoespaciais comparados aos bancos de dados relacionais.

Na opinião do autor desta monografia, esse resultado acontece devido aos índices criados de cada banco de dados, cada com seu próprio algoritmo de indexação para dados geoespaciais. Outro fator é o fato que os bancos de dados orientados a documentos foram desenvolvidos com o objetivo de ser utilizados em ambientes distribuídos, ao contrário dos bancos de dados relacionais que tem deficiência em ambientes distribuídos.

Ao analisar mais profundamente esses bancos de dados, também chega-se a algumas hipóteses com o objetivo de explicar porque os bancos de dados relacionais tem desempenho superior comparado ao banco de dados orientado a documentos. Entre os bancos de dados escolhidos nesta monografia, PostgreSQL e MongoDB, o banco de dados mais utilizado com esse objetivo atualmente que seja software livre e código aberto é o PostgreSQL, devido a vários fatores, desde o PostgreSQL ter um time de desenvolvimento que está a mais tempo na função e diversos tipos de interessados como universidades, centros de pesquisas, governos, até sua comunidade de usuários estar a mais tempo e ser mais ativa comparado ao MongoDB.

Entretanto esse experimento levanta algumas questões no comparativo entre banco de dados para manipulação de dados geoespaciais. Entre as questões abertas podemos escrever:

- Como serão o desempenho de ambo bancos de dados em cenários de ambientes distribuídos?
- Em vez de realizar a comparação de bancos de dados orientados a documentos com banco de dados relacionais, como seria o desempenho comparado a outros tipos de bancos de dados? Exemplo pode-se citar banco de dados orientado a grafos.
- Como será o desempenho de ambo bancos de dados ao realizar a manipulação de grande volume de dados (Gibabytes, ou Terabytes)?

Conforme descrito anteriormente, essas questões são também, recomendações de trabalhos futuros do autor desta monografia.

A elaboração deste trabalho proporcionou uma oportunidade única de desenvolvimento técnico e pessoal ao autor desta monografia. Devido aos desafios e dificuldades impostas durante este projeto, a revisão de velhos conceitos assimilados e a busca por novos conhecimentos necessários foi constante durante todo o projeto, principalmente no tema geoprocessamento para o autor desta monografia. Também é valido citar a colaboração de colegas, professores, e colegas de trabalho que atuam diretamente nas áreas descritas nesta monografia.

REFERÊNCIAS

ALMEIDA, Mário de Souza. **Elaboração de projeto, tcc, dissertação e tese**: uma abordagem simples, prática e objetiva. São Paulo: Atlas, 2011.

ALVAREZ, Guilherme Martins. **Criação e utilização de uma base de dados orientada a grafos:** Um estudo de caso sobre rede social. 2013. 104f. Monografia (Graduação em Ciência da Computação) – Universidade do Sul de Santa Catarina, Palhoça, 2013.

ANDRADE, Maria Margarida de. **Introdução à metodologia do trabalho científico:** elaboração de trabalhos na graduação. 5 ed. São Paulo: Atlas, 2001.

BANKER, Kyle. **MongoDB in Action**. Shelter Island: Manning, 2012.

BARASUOL, Érion Ricardo. **MongoDB uma base de dados orientada a documentos que utiliza orientação a objetos**. 2012. 101f. Monografia – Fundação Educacional do Município de Assis, Assis, 2012. Disponível em: <http://fema.edu.br/images/arqTccs/0911270085.pdf>. Acesso em: 19 out. 2014.

BATTY, Petter. **Exploiting Relational Database Technology in GIS.** 1990. Disponível em: <http://www.ebatty.com/Exploiting_Relational_Database_Technology_in_GIS.pdf>. Acesso em: 2 fev. 2015.

BEAL, Vangie. **ETL – Extract, Transform, Load.** Disponível em: <http://www.webopedia.com/TERM/E/ETL.html>. Acesso e: 15 mar. 2015.

BONFIOLI, Guilherme Ferreira. **Banco de dados relacional e objeto-relacional:** Uma comparação usando PostgreSQL. 2006. 62f. Monografia (Graduação em Ciências da Computação) - Universidade Federal de Lavras, Lavras, 2006. Disponível em: <http://www.bcc.ufla.br/wp-content/uploads/2013/2005/Banco_de_dados_relacional_e_objeto_relacional_uma_comparacao_usando_postgresql.pdf>. Acesso em: 14 out. 2014.

CACHO, Nélio Alessandro Azevedo; BENJAMIM, Xiankleber Cavalcante. **Sistemas de Banco de Dados**. Disponível em: <http://www.metropoledigital.ufrn.br/aulas_avancado/web/disciplinas/banco_de_dados/aula_01.html>. Acesso em: 7 out. 2014.

____. **Sistemas de banco de dados**. Disponível em: <http://www.metropoledigital.ufrn.br/aulas_avancado/web/disciplinas/banco_de_dados/aula_03.html>. Acesso em: 13 out. 2014.

CALDEIRA, Carlos Pampulim. **Introdução ao Modelo de Dados Relacional**. Disponível em: <http://host.di.uevora.pt/~ccaldeira/e/sp/imd/docs/BD1.pdf>. Acesso em: 12. out. 2014.

CÂMARA, Gilberto; DAVIS, Clodoveu; MONTEIRO, Antônio Miguel Vieira. **Introdução à Ciência da Geoinformação.** INPE. Disponível em: < http://www.dpi.inpe.br/gilberto/livro/introd/index.html>. Acesso em: 29 ago. 2014.

CAMILO, Márcio da Silva. **Uma breve história dos bancos de dados.** Disponível em: < http://www.sirmacstronger.eti.br/bd/introdbd.php>. Acesso em: 9 nov. 2014.

____. **O que é SQL.** Disponível em: <http://www.sirmacstronger.eti.br/bd/sql0.php>. Acesso em: 9 nov. 2014.

CASANOVA, Marco. Et al. **Banco de Dados Geográficos.** INPE. Disponível em: <http://www.dpi.inpe.br/livros/bdados/index.html>. Acesso em: 15 dez. 2014.

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

ECMA. **The JSON Data Interchange Format.** Disponível em: < http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em: 17 nov. 2014.

ELMASRI, Ramez; NAVATHE, Shamkant B; **Sistemas de banco de dados.** 4. ed. São Paulo: Pearson Addison, 2005.

FOWLER, Martin. **NosqlDefinition.** Disponível em: <http://martinfowler.com/bliki/NosqlDefinition.html>. Acesso em: 1 nov. 2014.

GALLIANO, A. Guilherme. **O Método Científico**: Teoria e Prática. São Paulo: Harbra, 1979.

GARTNER. **Geographic Information System (GIS).** Disponível em: <http://www.gartner.com/it-glossary/geographic-information-systems-gis/>. Acesso em: 1 out. 2014.

GEOJSON. **The GeoJSON Format Specification.** Disponível em: <http://geojson.org/geojson-spec.html>. Acesso em: 17 mar. 2015.

GEOKETTLE. **GeoKettle.** Disponível em: <http://www.spatialytics.org/projects/geokettle/>. Acesso em: 15 mar. 2015.

GEREMIA, Juliana. **Tutorial de Introdução a Banco de Dados.** 2010. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/db/Tut_DB.pdf>. Acesso em: 27 jan. 2015.

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. 6.ed. São Paulo: Atlas, 1999. Disponível em:

HARRISON, Guy. **NoSQL and Document-Oriented Databases,** 2010. Disponível em: <http://www.dbta.com/Columns/Notes-on-NoSQL/NoSQL-and-Document-OrientedDatabases-72035.aspx>. Acesso em: 6 fev. 2015.

Internet Engineering Task Force (IETF). **About the IETF.** Disponível em: < https://www.ietf.org/about/>. Acesso em: 22 mar. 2015.

JÚNIOR, João Ronaldo Tavares de Vasconcellos. **Sistema de informação para controle de dados da Coleção de Culturas de Microrganismos da Bahia**. 2010. 64f. Monografia (Especialização em Residência em Desenvolvimento de Software para Engenharia

Biomédica) – Universidade Estadual de Feira de Santana, Feira de Santana, 2010. Disponível em: <http://www.jvasconcellos.com.br/unijorge/wp-content/uploads/2011/08/Vasconcellos-monografia-digital.pdf>. Acesso em: 19 out. 2014.

KORTH, Henry F.; SILBERSCHATZ, Abraham. **Sistema de Banco de Dados**. 2. ed. São Paulo: McGraw-Hill, 1995.

LARMAN, Craig. **Utilizando UML e padrões**. Porto Alegre: Bookman, 2000.

LOBATO, Mateus Monteiro. Et al. IMPORTÂNCIA DOS SISTEMAS DE INFORMAÇÃO GEOGRÁFICA (SIG'S) PARA A CARTOGRAFIA TRADICIONAL. In: II Simpósio Brasileiro de Ciências Geodésicas e Tecnologias da Geoinformação, 2008, Recife/PE. **Anais eletrônicos...** Disponível em: <http://www.ufpe.br/cgtg/SIMGEOII_CD/Organizado/cart_sig/191.pdf>. Acesso em: 17 ago. 2014.

LONGO, Valéria Aparecida Anti. **A história da cartografia e suas contribuições para a linguagem cartográfica nas séries do ensino fundamental.** 2011. 20f. Monografia (Especialização em Geografia)-Universidade Estadual Paulista, Presidente Prudente, 2011. Disponível em: < http://www.rededosaber.sp.gov.br/portais/Portals/84/docs/tcc/REDEFOR_1ed_TCC_Val%C3%A9ria%20Aparecida%20Anti%20Longo.pdf>. Acesso em: 29 ago. 2014.

LUACES, Miguel Ángel Rodríguez. **A Generic Architecture for Geographic Information Systems.** 2004. 234f. Tese (Doutorado) – Universidade da Coruña, Coruña, 2004. Disponível em: <http://lbd.udc.es/Repository/Thesis/432677332J.pdf>. Acesso em: 12 jan. 2015.

MALHA Geométrica dos Municípios Brasileiros. Disponível em: < http://dados.gov.br/dataset/malha-geometrica-dos-municipios-brasileiros>. Acesso em: 22 mar. 2015.

MALHEIROS, Luiz. **Geoprocessamento, tecnologia e mercado de trabalho**. Disponível <http://www.procenge.com.br/site/geoprocessamento/>. Acesso em: 26 ago. 2014.

MATTEUSSI, Kassiano José. **Protótipo de interface web com php para gerenciamento de banco de dados couchdb.** 2010. 81f. Monografia (Graduação em Ciências da Computação) – Universidade Comunitária da Região de Chapecó, Chapecó, 2010. Disponível em: < https://s3.amazonaws.com/elton/docs/monografia_kassiano.pdf>. Acesso em: 28 jan. 2015.

MEDEIROS, Anderson Maciel Lima de. **O que são dados geográficos? Como são armazenados?** Disponível em: <http://andersonmedeiros.com/conceitos-dados-geograficos/>. Acesso em: 22 jan. 2015.

MITCHELL, Bradley. **HTTP.** Disponível em:< http://compnetworking.about.com/od/networkprotocols/g/bldef_http.htm>. Acesso em: 22 jan. 2015.

MORAIS, Caitlin Dempsey. **Who Coined the Phrase Geographic Information System?** 2012. Disponível em: <http://www.gislounge.com/phrase-geographic-information-systems/>. Acesso em: 14 set. 2014.

____. **GIS Data Explored – Vector and Raster Data.** 2000. Disponível em: < http://www.gislounge.com/geodatabases-explored-vector-and-raster-data/>. Acesso em: 20 jan. 2015.

NANCE, Cory. Et al. 2013. **NoSQL vs RDBMS – Why There is Room for Both.** Disponível em: <http://sais.aisnet.org/2013/Nance.pdf>. Acesso em: 28 jan. 2015.
National Center for Geographic Information and Analysis (NCGIA). **NCGIA Overview**. Disponível em: < http://www.ncgia.ucsb.edu/about/overview.php>. Acesso em: 2 nov. 2014.

Open Geospatial Consortium (OGC). **Frequency Asked Question (FAQ)**. Disponível em: < http://www.opengeospatial.org/ogc/faq>. Acesso em: 30 set. 2014.

Oracle. **Oracle Data Types**. Disponível em: <http://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm#CNCPT513>. Acesso em: 13 jan. 2015.

OZEMOY, Vladimir M. SMITH, Dennis R. SICHERMAN, Alan. **Evaluating computerized geographic information systems using decision analysis.** Interfaces. 1981.

PASSOS, Felipe Garcia. A importância do Sistema de Informação Geográfica - SIG - no ensino de Cartografia. In: Colóquio de Cartografia para Crianças e Escolares. VII, 2011, Vitória/ES. **Anais eletrônicos...** Disponível em: <http://cartografiaescolar2011.files.wordpress.com/2012/03/importanciasistemainformacaogeograficaensinocartografia.pdf>. Acesso em: 17 ago. 2014.

PICHILIANI, Mauro. Modelagem SQL x NoSQL: Veja algumas técnicas de modelagem utilizadas em bancos NoSQL. **Sql Magazine,** Rj, v. 117, n. 10, p.5-9, jan. 2013.

POSTGRESQL. **Feature Matrix**. Disponível em: <http://www.postgresql.org/about/featurematrix/>. Acesso em: 19 out. 2014.

SADALAGE, Pramod J.; FOWLER, Martin. **NoSQL distilled: a brief guide to the emerging world of polyglot persistence**. Pearson Education, 2012.

SANTOS, Guilherme Lages. **Mobilidade em pontos de venda**. 2011. 57f. Monografia (Especialização em Tecnologia Java) – Universidade Tecnológica Federal do Paraná, Curitiba, 2011. Disponível em: < http://www2.dainf.ct.utfpr.edu.br/esp/monografias-de-especializacao-da-turma-vi-2010-2011/CT_JAVA_VI_2010_09.PDF/at_download/file>. Acesso em: 20 out. 2014.

SCUSSEL, Alexande. **Pesquisa revela dados sobre a importância do GIS nas empresas.** 2011. Disponível em: < http://mundogeo.com/blog/2011/03/09/pesquisa-revela-dados-sobre-a-importancia-do-gis-nas-empresas/>. Acesso em: 2 fev. 2015.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianópolis: UFSC, 2005. 138 p. Disponível em: <http://www.convibra.com.br/uploa d/paper/adm/adm_3439.pdf>. Acesso em: 26 de maio 2013.

SIMÕES, André. **Comparando o NoSQL ao modelo relacional.** SQL Magazine, São Paulo, 123. Disponível em:< http://www.devmedia.com.br/comparando-o-nosql-ao-modelo-relacional/30917>. Acesso em: 1 nov. 2014.

TAKAY, Osvaldo Kotaro; ITALIANO, Isabel Cristina; FERREIRA, João Eduardo. **Introdução a Banco de Dados.** Disponível em: < https://www.up.ac.mz/cepe/images/apostila.pdf>. Acesso em: 6 out. 2014.

THE GeoJSON Format: draft-butler-geojson-5. Disponível em: <https://tools.ietf.org/html/draft-butler-geojson-05>. Acesso em: 17 mar. 2015.

TIWARI, Shashank. **Professional NoSQL**. Indianapolis: John Wiley & Sons, 2011.

TOMLINSON, Roger F. **An Introduction to the geo-information system of the Canada land inventory.** 1967. Disponível em: <http://gisandscience.files.wordpress.com/2014/02/3-an-introduction-to-the-geo-information-system-of-the-canada-land-inventory_complete.pdf>. Acesso em: 2 nov. 2014.

UCHOA, Helton Nogueira. Et al. **Arquitetura OpenGIS Baseada em Software Livre para Solução de Geoprocessamento**. 2010. Disponível em: < https://pt.scribd.com/doc/39188873/Arquitetura-OpenGIS-Baseada-em-Software-Livre-para-Solucao-de-Geoprocessamento>. Acesso em: 15 jan. 2015.

UCHOA, Helton Nogueira; FERREIRA, Paulo Roberto. **Geoprocessamento com Software Livre.** 2004. Disponível em: <https://pt.scribd.com/doc/40164871/Geoprocessamento-com-Software-Livre-Versao-1-0>. Acesso em: 15 jan. 2015.

University Consortium for Geographic Information System (UCGIS). **Remembering Dr. Roger Tomlinson, the inventor of computer-based GIS**. Disponível em: <http://ucgis.org/sites/default/files/document-sharing-form-files/209/Remembering%20Roger%20Tomlinson.pdf>. Acesso em: 2 nov. 2014.

ZLATANOVA, Sisi; STOTER, Jantien; **The role of DBMS in the new generation GIS architecture.** Disponível em: <http://3dgeoinfo.bk.tudelft.nl/szlatanova/pdfs/Role_DBMS_in_GIS_architecture.pdf>. Acesso em: 17 dez. 2014.

APÊNDICES

APÊNDICE A – Script criação da base de dados ‘shapefile_brasil’

> create database shapefile_brasil;
> \c shapefile_brasil;
> create extension postgis;

APÊNDICE B – Script criação da base de dados ‘*brasil*’

> create database brasil;
> \c brasil;
> create extension postgis;
> CREATE TABLE "estados" (gid serial NOT NULL, id double precision, cd_geocodu character varying(2), nm_estado character varying(50), nm_regiao character varying(20), geom geometry(MultiPolygon,4326), CONSTRAINT "estados_pkey" PRIMARY KEY (gid)) WITH (OIDS=FALSE); ALTER TABLE "estados" OWNER TO postgres; CREATE INDEX "estados_geom_idx" ON "estados" USING gist (geom);
> CREATE TABLE "micro_regioes" (gid serial NOT NULL, id double precision, nm_micro character varying(100), cd_geocodu character varying(2), geom geometry(MultiPolygon,4326), CONSTRAINT "micro_regioes_pkey" PRIMARY KEY (gid)) WITH (OIDS=FALSE); ALTER TABLE "micro_regioes" OWNER TO postgres; CREATE INDEX "micro_regioes_geom_idx" ON "micro_regioes" USING gist (geom);
> CREATE TABLE "meso_regioes" (gid serial NOT NULL, id double precision, nm_meso character varying(100), cd_geocodu character varying(2), geom geometry(MultiPolygon,4326), CONSTRAINT "meso_regioes_pkey" PRIMARY KEY (gid)) WITH (OIDS=FALSE); ALTER TABLE "meso_regioes" OWNER TO postgres; CREATE INDEX "meso_regioes_geom_idx" ON "meso_regioes" USING gist (geom);
> CREATE TABLE "municipios" (gid serial NOT NULL, id double precision, cd_geocodm character varying(20), nm_municip character varying(60), geom geometry(MultiPolygon,4326), CONSTRAINT "municipios_pkey" PRIMARY KEY (gid)) WITH (OIDS=FALSE); ALTER TABLE "municipios" OWNER TO postgres; CREATE INDEX "municipios_geom_idx" ON "municipios" USING gist (geom);

APÊNDICE C – Comandos SQL para seleção de todos os dados de todas as tabelas na base de dados '*shapefile_brasil*'.

Estados:

```
select
c.gid,c.id,c.cd_geocodu,c.nm_estado,c.nm_regiao,ST_SetSRID(ST_Force_2D(c.geom),4326)
as geom from "11ufe250gc_sir" c
union all
select
d.gid,d.id,d.cd_geocodu,d.nm_estado,d.nm_regiao,ST_SetSRID(ST_Force_2D(d.geom),4326
) as geom from "12ufe250gc_sir" d
union all
select
e.gid,e.id,e.cd_geocodu,e.nm_estado,e.nm_regiao,ST_SetSRID(ST_Force_2D(e.geom),4326)
as geom from "13ufe250gc_sir" e
union all
select
f.gid,f.id,f.cd_geocodu,f.nm_estado,f.nm_regiao,ST_SetSRID(ST_Force_2D(f.geom),4326)
as geom from "14ufe250gc_sir" f
union all
select
g.gid,g.id,g.cd_geocodu,g.nm_estado,g.nm_regiao,ST_SetSRID(ST_Force_2D(g.geom),4326
) as geom from "15ufe250gc_sir" g
union all
select
h.gid,h.id,h.cd_geocodu,h.nm_estado,h.nm_regiao,ST_SetSRID(ST_Force_2D(h.geom),4326
) as geom from "16ufe250gc_sir" h
union all
select
i.gid,i.id,i.cd_geocodu,i.nm_estado,i.nm_regiao,ST_SetSRID(ST_Force_2D(i.geom),4326) as
geom from "17ufe250gc_sir" i
union all
select
m.gid,m.id,m.cd_geocodu,m.nm_estado,m.nm_regiao,ST_SetSRID(ST_Force_2D(m.geom),4
326) as geom from "21ufe250gc_sir" m
union all
select
n.gid,n.id,n.cd_geocodu,n.nm_estado,n.nm_regiao,ST_SetSRID(ST_Force_2D(n.geom),4326
) as geom from "22ufe250gc_sir" n
union all
select
o.gid,o.id,o.cd_geocodu,o.nm_estado,o.nm_regiao,ST_SetSRID(ST_Force_2D(o.geom),4326
) as geom from "23ufe250gc_sir" o
union all
select
p.gid,p.id,p.cd_geocodu,p.nm_estado,p.nm_regiao,ST_SetSRID(ST_Force_2D(p.geom),4326
) as geom from "24ufe250gc_sir" p
union all
```

```
select
q.gid,q.id,q.cd_geocodu,q.nm_estado,q.nm_regiao,ST_SetSRID(ST_Force_2D(q.geom),4326
) as geom from "25ufe250gc_sir" q
union all
select
r.gid,r.id,r.cd_geocodu,r.nm_estado,r.nm_regiao,ST_SetSRID(ST_Force_2D(r.geom),4326)
as geom from "26ufe250gc_sir" r
union all
select
s.gid,s.id,s.cd_geocodu,s.nm_estado,s.nm_regiao,ST_SetSRID(ST_Force_2D(s.geom),4326)
as geom from "27ufe250gc_sir" s
union all
select
t.gid,t.id,t.cd_geocodu,t.nm_estado,t.nm_regiao,ST_SetSRID(ST_Force_2D(t.geom),4326) as
geom from "28ufe250gc_sir" t
union all
select
u.gid,u.id,u.cd_geocodu,u.nm_estado,u.nm_regiao,ST_SetSRID(ST_Force_2D(u.geom),4326
) as geom from "29ufe250gc_sir" u
union all
select
v.gid,v.id,v.cd_geocodu,v.nm_estado,v.nm_regiao,ST_SetSRID(ST_Force_2D(v.geom),4326
) as geom from "31ufe250gc_sir" v
union all
select
w.gid,w.id,w.cd_geocodu,w.nm_estado,w.nm_regiao,ST_SetSRID(ST_Force_2D(w.geom),4
326) as geom from "32ufe250gc_sir" w
union all
select
x.gid,x.id,x.cd_geocodu,x.nm_estado,x.nm_regiao,ST_SetSRID(ST_Force_2D(x.geom),4326
) as geom from "33ufe250gc_sir" x
union all
select
y.gid,y.id,y.cd_geocodu,y.nm_estado,y.nm_regiao,ST_SetSRID(ST_Force_2D(y.geom),4326
) as geom from "35ufe250gc_sir" y
union all
select
aa.gid,aa.id,aa.cd_geocodu,aa.nm_estado,aa.nm_regiao,ST_SetSRID(ST_Force_2D(aa.geom)
,4326) as geom from "41ufe250gc_sir" aa
union all
select
b.gid,b.id,b.cd_geocodu,b.nm_estado,b.nm_regiao,ST_SetSRID(ST_Force_2D(b.geom),4326
) as geom from "42ufe250gc_sir" b
union all
select
a.gid,a.id,a.cd_geocodu,a.nm_estado,a.nm_regiao,ST_SetSRID(ST_Force_2D(a.geom),4326)
as geom from "43ufe250gc_sir" a
union all
```

select
ab.gid,ab.id,ab.cd_geocodu,ab.nm_estado,ab.nm_regiao,ST_SetSRID(ST_Force_2D(ab.geom),4326) **as** geom **from** "50ufe250gc_sir" ab
union all
select
ac.gid,ac.id,ac.cd_geocodu,ac.nm_estado,ac.nm_regiao,ST_SetSRID(ST_Force_2D(ac.geom),4326) **as** geom **from** "51ufe250gc_sir" ac
union all
select
ad.gid,ad.id,ad.cd_geocodu,ad.nm_estado,ad.nm_regiao,ST_SetSRID(ST_Force_2D(ad.geom),4326) **as** geom **from** "52ufe250gc_sir" ad
union all
select
ae.gid,ae.id,ae.cd_geocodu,ae.nm_estado,ae.nm_regiao,ST_SetSRID(ST_Force_2D(ae.geom),4326) **as** geom **from** "53ufe250gc_sir" ae
Meso regiões:
select c.gid,c.id,c.nm_meso,c.cd_geocodu,ST_SetSRID(ST_Force_2D(c.geom),4326) **as** geom **from** "11mee250gc_sir" c
union all
select d.gid,d.id,d.nm_meso,d.cd_geocodu,ST_SetSRID(ST_Force_2D(d.geom),4326) **as** geom **from** "12mee250gc_sir" d
union all
select e.gid,e.id,e.nm_meso,e.cd_geocodu,ST_SetSRID(ST_Force_2D(e.geom),4326) **as** geom **from** "13mee250gc_sir" e
union all
select f.gid,f.id,f.nm_meso,f.cd_geocodu,ST_SetSRID(ST_Force_2D(f.geom),4326) **as** geom **from** "14mee250gc_sir" f
union all
select g.gid,g.id,g.nm_meso,g.cd_geocodu,ST_SetSRID(ST_Force_2D(g.geom),4326) **as** geom **from** "15mee250gc_sir" g
union all
select h.gid,h.id,h.nm_meso,h.cd_geocodu,ST_SetSRID(ST_Force_2D(h.geom),4326) **as** geom **from** "16mee250gc_sir" h
union all
select i.gid,i.id,i.nm_meso,i.cd_geocodu,ST_SetSRID(ST_Force_2D(i.geom),4326) **as** geom **from** "17mee250gc_sir" i
union all
select m.gid,m.id,m.nm_meso,m.cd_geocodu,ST_SetSRID(ST_Force_2D(m.geom),4326) **as** geom **from** "21mee250gc_sir" m
union all
select n.gid,n.id,n.nm_meso,n.cd_geocodu,ST_SetSRID(ST_Force_2D(n.geom),4326) **as** geom **from** "22mee250gc_sir" n
union all
select o.gid,o.id,o.nm_meso,o.cd_geocodu,ST_SetSRID(ST_Force_2D(o.geom),4326) **as** geom **from** "23mee250gc_sir" o
union all
select p.gid,p.id,p.nm_meso,p.cd_geocodu,ST_SetSRID(ST_Force_2D(p.geom),4326) **as** geom **from** "24mee250gc_sir" p
union all

select q.gid,q.id,q.nm_meso,q.cd_geocodu,ST_SetSRID(ST_Force_2D(q.geom),4326) **as** geom **from** "25mee250gc_sir" q
union all
select r.gid,r.id,r.nm_meso,r.cd_geocodu,ST_SetSRID(ST_Force_2D(r.geom),4326) **as** geom **from** "26mee250gc_sir" r
union all
select s.gid,s.id,s.nm_meso,s.cd_geocodu,ST_SetSRID(ST_Force_2D(s.geom),4326) **as** geom **from** "27mee250gc_sir" s
union all
select t.gid,t.id,t.nm_meso,t.cd_geocodu,ST_SetSRID(ST_Force_2D(t.geom),4326) **as** geom **from** "28mee250gc_sir" t
union all
select u.gid,u.id,u.nm_meso,u.cd_geocodu,ST_SetSRID(ST_Force_2D(u.geom),4326) **as** geom **from** "29mee250gc_sir" u
union all
select v.gid,v.id,v.nm_meso,v.cd_geocodu,ST_SetSRID(ST_Force_2D(v.geom),4326) **as** geom **from** "31mee250gc_sir" v
union all
select w.gid,w.id,w.nm_meso,w.cd_geocodu,ST_SetSRID(ST_Force_2D(w.geom),4326) **as** geom **from** "32mee250gc_sir" w
union all
select x.gid,x.id,x.nm_meso,x.cd_geocodu,ST_SetSRID(ST_Force_2D(x.geom),4326) **as** geom **from** "33mee250gc_sir" x
union all
select y.gid,y.id,y.nm_meso,y.cd_geocodu,ST_SetSRID(ST_Force_2D(y.geom),4326) **as** geom **from** "35mee250gc_sir" y
union all
select aa.gid,aa.id,aa.nm_meso,aa.cd_geocodu,ST_SetSRID(ST_Force_2D(aa.geom),4326) **as** geom **from** "41mee250gc_sir" aa
union all
select b.gid,b.id,b.nm_meso,b.cd_geocodu,ST_SetSRID(ST_Force_2D(b.geom),4326) **as** geom **from** "42mee250gc_sir" b
union all
select a.gid,a.id,a.nm_meso,a.cd_geocodu,ST_SetSRID(ST_Force_2D(a.geom),4326) **as** geom **from** "43mee250gc_sir" a
union all
select ab.gid,ab.id,ab.nm_meso,ab.cd_geocodu,ST_SetSRID(ST_Force_2D(ab.geom),4326) **as** geom **from** "50mee250gc_sir" ab
union all
select ac.gid,ac.id,ac.nm_meso,ac.cd_geocodu,ST_SetSRID(ST_Force_2D(ac.geom),4326) **as** geom **from** "51mee250gc_sir" ac
union all
select ad.gid,ad.id,ad.nm_meso,ad.cd_geocodu,ST_SetSRID(ST_Force_2D(ad.geom),4326) **as** geom **from** "52mee250gc_sir" ad
union all
select
ae.gid,ae.id,ae.nm_meso,ae.cd_geocodu,ST_SetSRID(ST_Force_2D(ae.geom),4326) as **geom** from **"53mee250gc_sir" ae**
Micro regiões

```
select c.gid,c.id, c.nm_micro, c.cd_geocodu, ST_SetSRID(ST_Force_2D(c.geom),4326) as
geom from "11mie250gc_sir" c
union all
select d.gid,d.id, d.nm_micro, d.cd_geocodu, ST_SetSRID(ST_Force_2D(d.geom),4326) as
geom from "12mie250gc_sir" d
union all
select e.gid,e.id, e.nm_micro, e.cd_geocodu, ST_SetSRID(ST_Force_2D(e.geom),4326) as
geom from "13mie250gc_sir" e
union all
select f.gid,f.id, f.nm_micro, f.cd_geocodu, ST_SetSRID(ST_Force_2D(f.geom),4326) as
geom from "14mie250gc_sir" f
union all
select g.gid,g.id, g.nm_micro, g.cd_geocodu, ST_SetSRID(ST_Force_2D(g.geom),4326) as
geom from "15mie250gc_sir" g
union all
select h.gid,h.id, h.nm_micro, h.cd_geocodu, ST_SetSRID(ST_Force_2D(h.geom),4326) as
geom from "16mie250gc_sir" h
union all
select i.gid,i.id, i.nm_micro, i.cd_geocodu, ST_SetSRID(ST_Force_2D(i.geom),4326) as
geom from "17mie250gc_sir" i
union all
select m.gid,m.id, m.nm_micro, m.cd_geocodu, ST_SetSRID(ST_Force_2D(m.geom),4326)
as geom from "21mie250gc_sir" m
union all
select n.gid,n.id, n.nm_micro, n.cd_geocodu, ST_SetSRID(ST_Force_2D(n.geom),4326) as
geom from "22mie250gc_sir" n
union all
select o.gid,o.id, o.nm_micro, o.cd_geocodu, ST_SetSRID(ST_Force_2D(o.geom),4326) as
geom from "23mie250gc_sir" o
union all
select p.gid,p.id, p.nm_micro, p.cd_geocodu, ST_SetSRID(ST_Force_2D(p.geom),4326) as
geom from "24mie250gc_sir" p
union all
select q.gid,q.id, q.nm_micro, q.cd_geocodu, ST_SetSRID(ST_Force_2D(q.geom),4326) as
geom from "25mie250gc_sir" q
union all
select r.gid,r.id, r.nm_micro, r.cd_geocodu, ST_SetSRID(ST_Force_2D(r.geom),4326) as
geom from "26mie250gc_sir" r
union all
select s.gid,s.id, s.nm_micro, s.cd_geocodu, ST_SetSRID(ST_Force_2D(s.geom),4326) as
geom from "27mie250gc_sir" s
union all
select t.gid,t.id, t.nm_micro, t.cd_geocodu, ST_SetSRID(ST_Force_2D(t.geom),4326) as
geom from "28mie250gc_sir" t
union all
select u.gid,u.id, u.nm_micro, u.cd_geocodu, ST_SetSRID(ST_Force_2D(u.geom),4326) as
geom from "29mie250gc_sir" u
union all
select v.gid,v.id, v.nm_micro, v.cd_geocodu, ST_SetSRID(ST_Force_2D(v.geom),4326) as
geom from "31mie250gc_sir" v
```

union all
select w.gid,w.id, w.nm_micro, w.cd_geocodu, ST_SetSRID(ST_Force_2D(w.geom),4326) **as** geom **from** "32mie250gc_sir" w
union all
select x.gid,x.id, x.nm_micro, x.cd_geocodu, ST_SetSRID(ST_Force_2D(x.geom),4326) **as** geom **from** "33mie250gc_sir" x
union all
select y.gid,y.id, y.nm_micro, y.cd_geocodu, ST_SetSRID(ST_Force_2D(y.geom),4326) **as** geom **from** "35mie250gc_sir" y
union all
select aa.gid,aa.id, aa.nm_micro, aa.cd_geocodu, ST_SetSRID(ST_Force_2D(aa.geom),4326) **as** geom **from** "41mie250gc_sir" aa
union all
select b.gid,b.id, b.nm_micro, b.cd_geocodu, ST_SetSRID(ST_Force_2D(b.geom),4326) **as** geom **from** "42mie250gc_sir" b
union all
select a.gid,a.id, a.nm_micro, a.cd_geocodu, ST_SetSRID(ST_Force_2D(a.geom),4326) **as** geom **from** "43mie250gc_sir" a
union all
select ab.gid,ab.id, ab.nm_micro, ab.cd_geocodu, ST_SetSRID(ST_Force_2D(ab.geom),4326) **as** geom **from** "50mie250gc_sir" ab
union all
select ac.gid,ac.id, ac.nm_micro, ac.cd_geocodu, ST_SetSRID(ST_Force_2D(ac.geom),4326) **as** geom **from** "51mie250gc_sir" ac
union all
select ad.gid,ad.id, ad.nm_micro, ad.cd_geocodu, ST_SetSRID(ST_Force_2D(ad.geom),4326) **as** geom **from** "52mie250gc_sir" ad
union all
select ae.gid,ae.id, ae.nm_micro, ae.cd_geocodu, ST_SetSRID(ST_Force_2D(ae.geom),4326) as geom from "53mie250gc_sir" ae

Municípios
select c.gid,c.id,c.cd_geocodm,c.nm_municip,ST_SetSRID(ST_Force_2D(c.geom),4326) **as** geom **from** "11mue250gc_sir" c
union all
select d.gid,d.id,d.cd_geocodm,d.nm_municip,ST_SetSRID(ST_Force_2D(d.geom),4326) **as** geom **from** "12mue250gc_sir" d
union all
select e.gid,e.id,e.cd_geocodm,e.nm_municip,ST_SetSRID(ST_Force_2D(e.geom),4326) **as** geom **from** "13mue250gc_sir" e
union all
select f.gid,f.id,f.cd_geocodm,f.nm_municip,ST_SetSRID(ST_Force_2D(f.geom),4326) **as** geom **from** "14mue250gc_sir" f
union all
select g.gid,g.id,g.cd_geocodm,g.nm_municip,ST_SetSRID(ST_Force_2D(g.geom),4326) **as** geom **from** "15mue250gc_sir" g
union all
select h.gid,h.id,h.cd_geocodm,h.nm_municip,ST_SetSRID(ST_Force_2D(h.geom),4326) **as** geom **from** "16mue250gc_sir" h
union all

```
select i.gid,i.id,i.cd_geocodm,i.nm_municip,ST_SetSRID(ST_Force_2D(i.geom),4326) as
geom from "17mue250gc_sir" i
union all
select m.gid,m.id,m.cd_geocodm,m.nm_municip,ST_SetSRID(ST_Force_2D(m.geom),4326)
as geom from "21mue250gc_sir" m
union all
select n.gid,n.id,n.cd_geocodm,n.nm_municip,ST_SetSRID(ST_Force_2D(n.geom),4326) as
geom from "22mue250gc_sir" n
union all
select o.gid,o.id,o.cd_geocodm,o.nm_municip,ST_SetSRID(ST_Force_2D(o.geom),4326) as
geom from "23mue250gc_sir" o
union all
select p.gid,p.id,p.cd_geocodm,p.nm_municip,ST_SetSRID(ST_Force_2D(p.geom),4326) as
geom from "24mue250gc_sir" p
union all
select q.gid,q.id,q.cd_geocodm,q.nm_municip,ST_SetSRID(ST_Force_2D(q.geom),4326) as
geom from "25mue250gc_sir" q
union all
select r.gid,r.id,r.cd_geocodm,r.nm_municip,ST_SetSRID(ST_Force_2D(r.geom),4326) as
geom from "26mue250gc_sir" r
union all
select s.gid,s.id,s.cd_geocodm,s.nm_municip,ST_SetSRID(ST_Force_2D(s.geom),4326) as
geom from "27mue250gc_sir" s
union all
select t.gid,t.id,t.cd_geocodm,t.nm_municip,ST_SetSRID(ST_Force_2D(t.geom),4326) as
geom from "28mue250gc_sir" t
union all
select u.gid,u.id,u.cd_geocodm,u.nm_municip,ST_SetSRID(ST_Force_2D(u.geom),4326) as
geom from "29mue250gc_sir" u
union all
select v.gid,v.id,v.cd_geocodm,v.nm_municip,ST_SetSRID(ST_Force_2D(v.geom),4326) as
geom from "31mue250gc_sir" v
union all
select w.gid,w.id,w.cd_geocodm,w.nm_municip,ST_SetSRID(ST_Force_2D(w.geom),4326)
as geom from "32mue250gc_sir" w
union all
select x.gid,x.id,x.cd_geocodm,x.nm_municip,ST_SetSRID(ST_Force_2D(x.geom),4326) as
geom from "33mue250gc_sir" x
union all
select y.gid,y.id,y.cd_geocodm,y.nm_municip,ST_SetSRID(ST_Force_2D(y.geom),4326) as
geom from "35mue250gc_sir" y
union all
select
aa.gid,aa.id,aa.cd_geocodm,aa.nm_municip,ST_SetSRID(ST_Force_2D(aa.geom),4326) as
geom from "41mue250gc_sir" aa
union all
select b.gid,b.id,b.cd_geocodm,b.nm_municip,ST_SetSRID(ST_Force_2D(b.geom),4326) as
geom from "42mue250gc_sir" b
union all
```

```
select  a.gid,a.id,a.cd_geocodm,a.nm_municip,ST_SetSRID(ST_Force_2D(a.geom),4326)  as
geom from "43mue250gc_sir" a
union all
select
ab.gid,ab.id,ab.cd_geocodm,ab.nm_municip,ST_SetSRID(ST_Force_2D(ab.geom),4326)  as
geom from "50mue250gc_sir" ab
union all
select
ac.gid,ac.id,ac.cd_geocodm,ac.nm_municip,ST_SetSRID(ST_Force_2D(ac.geom),4326)  as
geom from "51mue250gc_sir" ac
union all
select
ad.gid,ad.id,ad.cd_geocodm,ad.nm_municip,ST_SetSRID(ST_Force_2D(ad.geom),4326)  as
geom from "52mue250gc_sir" ad
union all
select
ae.gid,ae.id,ae.cd_geocodm,ae.nm_municip,ST_SetSRID(ST_Force_2D(ae.geom),4326)  as
geom from "53mue250gc_sir" ae
```

APÊNDICE D – Comandos SQL para retornar os dados que serão exportados para GeoJSON.

```
>select * from estados;
>select * from meso_regioes;
>select * from micro_regioes;
>select * from municipios;
```

APÊNDICE E – Remoção de dados inválidos do cenário do experimento

PostgreSQL:

```
>delete from municipios where nm_municip = 'ALFREDO VASCONCELOS';
```

MongoDB:

```
> db.municipios.remove({nm_municip: "ALFREDO VASCONCELOS"});
```

APÊNDICE F – Comandos para realizar a importação dos arquivos GeoJSON para o MongoDB.

```
>mongoimport --db brasil --collection estados --file c:\temp\estados.json
>mongoimport --db brasil --collection meso_regioes --file c:\temp\ meso_regioes.json
>mongoimport --db brasil --collection micro_regioes --file c:\temp\ micro_regioes.json
>mongoimport --db brasil --collection municipios --file c:\temp\ municipios.json
```

APÊNDICE G – Comandos para criar os índices nos campos geográficos no MongoDB.

```
> use brasil;
> db.estados.createIndex({"features.geometry":"2dsphere"});
> db.meso_regioes.createIndex({"features.geometry":"2dsphere"});
> db.micro_regioes.createIndex({"features.geometry":"2dsphere"});
```

> db.municipios.createIndex({"features.geometry":"2dsphere"});

APÊNDICE H – Comandos utilizados para realizar executar a RF001.

PostgreSQL> SELECT nm_estado FROM estados WHERE ST_Intersects(geom,ST_GeomFromText('POINT(-48.56146 -27.54014)', 4326));
MongoDB > db.estados.find({"geometry": { $geoIntersects: { $geometry: { type: "Point" , coordinates:[-48.56146,-27.54014] } } }},{"nm_estado":1});

APÊNDICE I – Comandos utilizados para realizar executar a RF002.

PostgreSQL > SELECT nm_estado FROM estados WHERE ST_Intersects(geom,ST_GeomFromText('LINESTRING(-53.63637 -26.31376, -51.17683 -26.79483, -50.58058 -28.02121, -48.70375 -26.20535)', 4326));
MongoDB > db.estados.find({"geometry": {$geoIntersects: {$geometry: {type: "LineString",coordinates: [[-53.63637,-26.31376],[-51.17683,-26.79483],[-50.58058,-28.02121],[-48.70375,-26.20535],[-48.56146,-27.54014]]}}}},{"nm_estado": 1});

APÊNDICE J – Comandos utilizados para realizar executar a RF003.

PostgreSQL > SELECT nm_estado FROM estados WHERE ST_Intersects(geom,ST_GeomFromText('POLYGON((-51.55627 -26.75418, -48.98832 -26.97099, -49.02220 -28.59713,-49.02220 -28.59713,-51.55627 -26.75418))', 4326));
MongoDB > db.estados.find({"geometry": { $geoIntersects: { $geometry: { type: "Polygon" , coordinates: [[[-51.55627,-26.75418] , [-48.98832,-26.97099] , [-49.02220,-28.59713] , [-49.02220,-28.59713],[-51.55627,-26.75418]]] } } } },{"nm_estado":1});

APÊNDICE K – Comandos utilizados para realizar executar a RF004.

PostgreSQL > SELECT nm_meso FROM meso_regioes WHERE ST_Intersects(geom,ST_GeomFromText('POINT(-48.56146 -27.54014)', 4326));
MongoDB > db.meso_regioes.find({"geometry": { $geoIntersects: { $geometry: { type: "Point" , coordinates:[-48.56146,-27.54014] } } }},{"nm_meso":1});

APÊNDICE L – Comandos utilizados para realizar executar a RF005.

PostgreSQL > SELECT nm_meso FROM meso_regioes WHERE ST_Intersects(geom,ST_GeomFromText('LINESTRING(-53.63637 -26.31376, -51.17683 -26.79483, -50.58058 -28.02121, -48.70375 -26.20535)', 4326));
MongoDB > db.meso_regioes.find({"geometry": { $geoIntersects: { $geometry: { type: "LineString" , coordinates: [[-53.63637,-26.31376],[-51.17683,-26.79483],[-50.58058,-28.02121],[-48.70375,-26.20535],[-48.56146,-27.54014]] } } }},{"nm_meso":1});

APÊNDICE M – Comandos utilizados para realizar executar a RF006.

PostgreSQL > SELECT nm_meso FROM meso_regioes WHERE ST_Intersects(geom,ST_GeomFromText('POLYGON((-51.55627 -26.75418, -48.98832 -26.97099, -49.02220 -28.59713,-49.02220 -28.59713,-51.55627 -26.75418))', 4326));
MongoDB > db.meso_regioes.find({"geometry": { $geoIntersects: { $geometry: { type: "Polygon" , coordinates: [[[-51.55627,-26.75418] , [-48.98832,-26.97099] , [-49.02220,-28.59713] , [-49.02220,-28.59713],[-51.55627,-26.75418]]] } }}},{"nm_meso":1});

APÊNDICE N – Comandos utilizados para realizar executar a RF007.

PostgreSQL > SELECT nm_micro FROM micro_regioes WHERE ST_Intersects(geom,ST_GeomFromText('POINT(-48.56146 -27.54014)', 4326));
MongoDB > db.micro_regioes.find({"geometry": { $geoIntersects: { $geometry: { type: "Point" , coordinates:[-48.56146,-27.54014] } } }},{"nm_micro":1});

APÊNDICE O – Comandos utilizados para realizar executar a RF008.

PostgreSQL > SELECT nm_micro FROM micro_regioes WHERE ST_Intersects(geom,ST_GeomFromText('LINESTRING(-53.63637 -26.31376, -51.17683 -26.79483, -50.58058 -28.02121, -48.70375 -26.20535)', 4326));
MongoDB > db.micro_regioes.find({"geometry": { $geoIntersects: { $geometry: { type: "LineString" , coordinates: [[-53.63637,-26.31376],[-51.17683,-26.79483],[-50.58058,-28.02121],[-48.70375,-26.20535],[-48.56146,-27.54014]] } } }},{"nm_micro":1});

APÊNDICE P – Comandos utilizados para realizar executar a RF009.

PostgreSQL > SELECT nm_micro FROM micro_regioes WHERE ST_Intersects(geom,ST_GeomFromText('POLYGON((-51.55627 -26.75418, -48.98832 -26.97099, -49.02220 -28.59713,-49.02220 -28.59713,-51.55627 -26.75418))', 4326));
MongoDB > db.micro_regioes.find({"geometry": { $geoIntersects: { $geometry: { type: "Polygon" , coordinates: [[[-51.55627,-26.75418] , [-48.98832,-26.97099] , [-49.02220,-28.59713] , [-49.02220,-28.59713],[-51.55627,-26.75418]]] } }}},{"nm_micro":1});

APÊNDICE Q – Comandos utilizados para realizar executar a RF010.

PostgreSQL > SELECT nm_municip FROM municipios WHERE ST_Intersects(geom,ST_GeomFromText('POINT(-48.56146 -27.54014)', 4326));
MongoDB > db.municipios.find({"geometry": { $geoIntersects: { $geometry: { type: "Point" , coordinates:[-48.56146,-27.54014] } } }},{"nm_municip":1});

APÊNDICE R – Comandos utilizados para realizar executar a RF011.

PostgreSQL > SELECT nm_municip FROM municipios WHERE ST_Intersects(geom,ST_GeomFromText('LINESTRING(-53.63637 -26.31376, -51.17683 -26.79483, -50.58058 -28.02121, -48.70375 -26.20535)', 4326));
MongoDB > db.municipios.find({"geometry": { $geoIntersects: { $geometry: { type: "LineString" , coordinates: [[-53.63637,-26.31376],[-51.17683,-26.79483],[-50.58058,-28.02121],[-48.70375,-26.20535],[-48.56146,-27.54014]] } } }},{"nm_municip":1});

APÊNDICE S – Comandos utilizados para realizar executar a RF012.

PostgreSQL > SELECT nm_municip FROM municipios WHERE ST_Intersects(geom,ST_GeomFromText('POLYGON((-51.55627 -26.75418, -48.98832 -26.97099, -49.02220 -28.59713,-49.02220 -28.59713,-51.55627 -26.75418))', 4326));
MongoDB > db.municipios.find({"geometry": { $geoIntersects: { $geometry: { type: "Polygon" , coordinates: [[[-51.55627,-26.75418] , [-48.98832,-26.97099] , [-49.02220,-28.59713] , [-49.02220,-28.59713],[-51.55627,-26.75418]]] } }}},{"nm_municip":1});