Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Graduação em Ciência da Computação

Breno Reis do Nascimento

Melhoramento do tempo de resposta em framework de mineração de dados socioeconômicos usando MongoDB, MySQL e Solr

São Luís
2014

Breno Reis do Nascimento

Melhoramento do tempo de resposta em framework de mineração de dados socioeconômicos usando MongoDB, MySQL e Solr

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Geraldo Braz Jr

Doutor em Engenharia de Eletricidade

São Luís

2014

Breno Reis do Nascimento

Melhoramento do tempo de resposta em framework de mineração de dados socioeconômicos usando MongoDB, MySQL e Solr

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 19/12/2014

BANCA EXAMINADORA

Geraldo Braz Jr

Doutor em Engenharia de Eletricidade

Anselmo Cardoso de Paiva

Doutor em Informática

João Dallyson Sousa de Almeida

Doutor em Engenharia de Eletricidade

Nascimento, Breno Reis do.

Melhoramento do tempo de resposta em framework de mineração de dados socioeconômicos usando MongoDB, MySQL e Solr / Breno Reis do Nascimento. – São Luís, 2014.

47 f.

Impresso por computador (fotocópia).

Orientador: Geraldo Braz Jr.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2014.

1. Framework. 2. Mineração. 3. Socioeconômico - Fonte de dados. I. Título.

CDU 004

À família e amigos.

Agradecimentos

À minha família, por todo apoio que me deram, principalmente na decisão de ir estudar em outra cidade.

À minha amiga, namorada e esposa Klyvia, que com todas as dificuldades sempre esteve ao meu lado.

Aos amigos feitos na UFMA ao longo dos anos, que cada um com sua história de vida serviram de exemplo para continuarmos acreditando nos nossos objetivos.

Aos professores do DEINF pelos ensinamentos fornecidos durante o curso.

Ao professor Anselmo e equipe do NCA, por me oferecem a oportunidade de crescer profissionalmente junto deles.

Ao meu orientador Geraldo, pela confiança e apoio ao longo de minha carreira.

À Deus, que a cada dia me ajuda levantar e continuar seguindo em frente.

Resumo

Quando é necessário buscar informações em grandes fontes de dados, as etapas de solicitação, processamento e retorno dos dados devem ser realizadas de forma rápida para o usuário. Sistemas de análise de dados socioeconômicos possuem esse requisito, pois trabalham com grande quantidade e volume de dados, incluindo dados complexos como a informação geográfica de um determinado local. Esta monografia tem como proposta melhorar o tempo de resposta em um framework de mineração de dados socioeconômicos, analisando e implementando as ferramentas que melhor se adaptem para que o processamento das consultas seja realizada rapidamente.

Palavras-chave: Framework, mineração, consulta, socioeconômico, fonte de dados

Abstract

When you need to search for information in large data sources, the request, processing and return the data steps need to be quickly to the user. Socioeconomic data analysis systems have this requirement, because they work with large quantity and volume of data, including complex data such as geographical information of a particular place. This paper is to improve the response time on a socioeconomic data mining framework, analyzing tools to the processing queries be quicker.

Keywords: Framework, data mining, request, socioeconomic, datasources.

Lista de Figuras

Lista de Tabelas

Sumário

1 INTRODUÇÃO

A busca de dados sobre um determinado assunto é de extrema importância para um gestor identificar problemas, encontrar soluções, traçar metas entre outros. Na área governamental temos o exemplo das informações socioeconômicas de um País, Estado ou Município englobam áreas como economia, educação, moradia e saúde. Contudo, devido ao grande número de tipos e quantidades de informação, coletar e analisar uma variedade de conteúdo tão extensa tornam-se demasiado lento para que possam ser realizadas buscas nestas bases de dados.

Atualmente existem sistemas computacionais utilizados por diferentes esferas públicas e privadas com o intuito de representar as informações obtidas de forma simples e acessível. Entretanto, quando a informação buscada é extensa o processo de busca e representação do conteúdo solicitado pelo usuário possuem tempo de resposta que pode não ser aceitável de acordo com a aplicação pretendida.

Este trabalho tem por objetivo implementar recursos visando otimizar a consulta dos dados a serem buscados em um framework de desenvolvimento de sistemas de análise de dados socioeconômicos (LIMA, 2013). Também são objetivos definir um padrão na estrutura da informação e oferecer recursos que facilitem o desenvolvedor a elaborar um sistema de análise socioeconômica. Dessa forma para uma grande quantidade de dados a serem retornados nas consultas, haverá um tempo reduzido na resposta dos mesmos.

O restante deste trabalho está organizado em mais 7 Capítulos. O Capítulo 2 apresenta trabalhos relacionados e suas soluções. No Capítulo 3 são apresentados conhecimentos necessários para entender a proposta deste trabalho. O Capítulo 4 apresenta a proposta descrevendo o que foi feito, o que é necessário para que seu funcionamento ocorra da maneira correta, configurações e testes. Respectivamente, os Capítulos 5 e 6 são apresentados a aplicação criada com a otimização de consultas utilizando o framework e os testes em condições de igualdade, bem como os resultados obtidos. No último Capítulo são informadas as conclusões, limitações e pretensões para futuras melhorias da nova versão do framework com os recursos de consultas otimizadas.

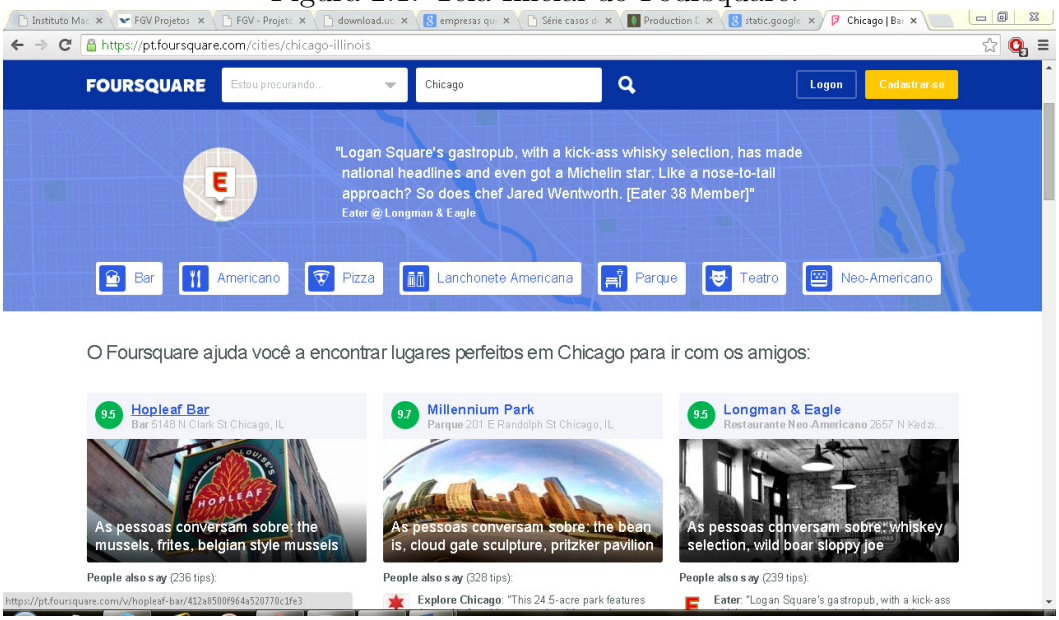
2 TRABALHOS RELACIONADOS

Muitos sistemas apresentam diversas informações dinâmicas de forma rápida e precisa. Sendo assim, é necessário além de uma boa estrutura da fonte de dados, a garantia de recursos para que essa base quando requisitada atue eficientemente.

2.1 Foursquare

O Foursquare (Figura 2.1) é uma rede social que permite aos seus usuários realizarem "checks-in" dos locais visitados. Criada em 2009 teve rápida ascensão e com o aumento dos dados contidos no site houve a necessidade da migração da estrutura de banco de dados relacional para o NoSQL (MongoDB). Em 2011 (início da migração) existiam cerca de 9 milhões de usuários, 3 milhões de check-in diários e 20 milhões de locais cadastrados com informações espaciais, ao todo são 2 bilhões de registros nos seus dados de 2009 à 2011. Com a migração as consultas, atualizações e inserções de informações o tempo de resposta se tornaram mais rápidas antes do serviço ficar lento aos usuários.

Figura 2.1: Tela Inicial do Foursquare.

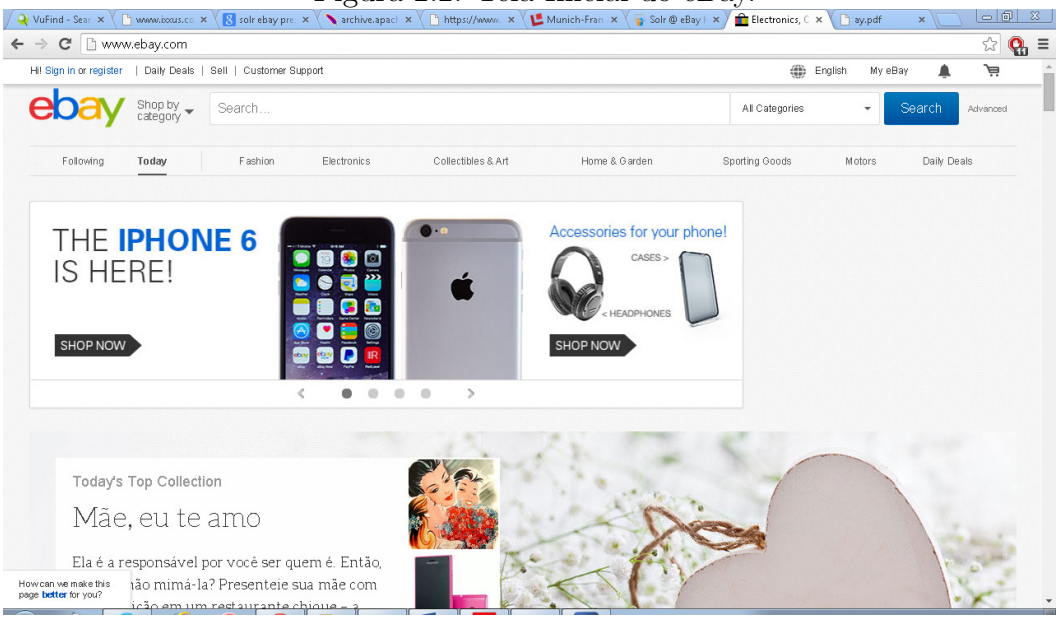


Fonte: (FOURSQUARE, 2014)

2.2 eBay

O eBay (Figura 3.1) é um site do tipo e-commerce, em 2011 haviam 97 milhões de usuários, 200 milhões de itens para compra e venda sendo que por dia eram realizadas 250 milhões de consultas.

Figura 2.2: Tela Inicial do eBay.



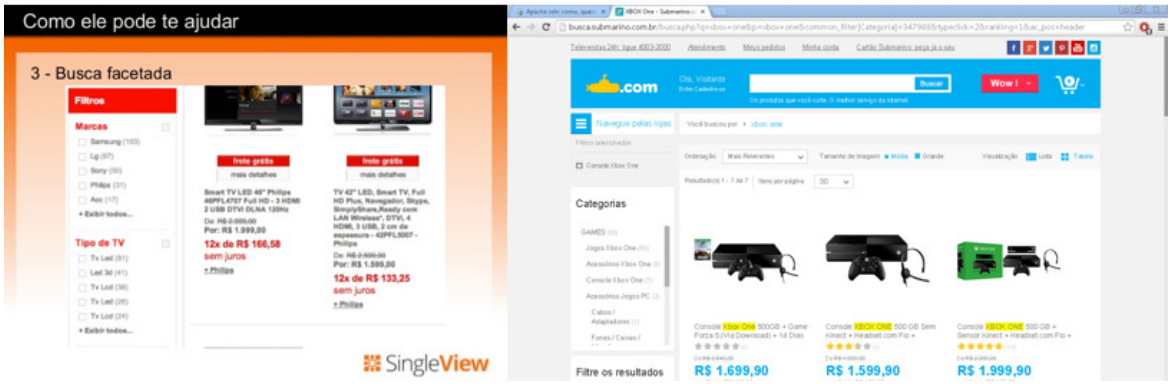
Fonte: (EBAY, 2014)

Baseado nessas grandes quantidades de dados, percebeu-se que esses números iriam aumentar mais e por consequência as consultas segundo a álgebra relacional iriam se tornar mais lentas, até mesmo o recurso de sugestão de itens ao realizar uma busca qualquer levaria mais tempo que o normal.

2.3 Trabalhos Relacionados no Brasil

Como exemplo de e-commerce brasileiro (Figura 2.3) temos os sites americanas e submarino, que utilizam o Solr para a indexação e busca de seus produtos, utilizam da busca facetada que fornece intervalos de valores e classificadores como filtros e caixa de sugestões ao procurar um determinado produto.

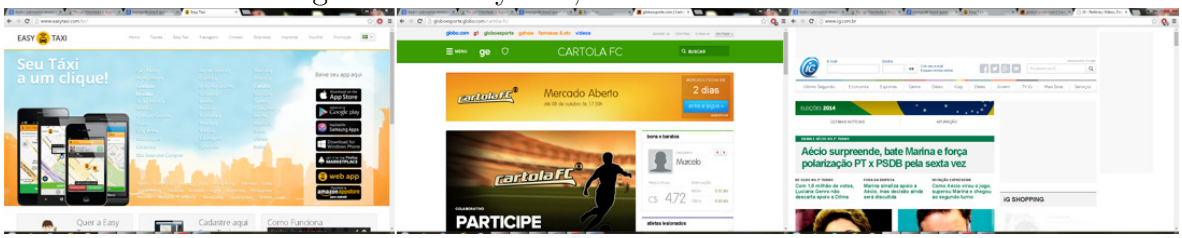
Figura 2.3: Tela Americanas e Submarino.



Fonte: (KLEYNANZEIGEN, 2011)

Fora do setor de e-commerce (Figura 2.4) existem três casos de sucesso no brasil: Easy Taxi (serviço para encontrar táxis baseados na localidade atual do cliente), sistemas como o CartolaFC (fantasy game que em 2013 teve 30 mil acessos simultâneos), e o portal IG (para busca e inserção de conteúdo no site). Todos utilizaram de serviços para que as operações de consultas não ficassem lentas.

Figura 2.4: EasyTaxi, CartolaFC e Portal IG.



Fonte: (KLEYNANZEIGEN, 2011)

3 REFERENCIAL TEÓRICO

A manutenção faz parte do ciclo de vida de um sistema computacional. Dentro deste processo está o aprimoramento de recursos. Sistemas que são projetados sem prever a quantidade de dados a ser mantida e as tecnologias que surgem para melhorar aspectos atuais influenciam diretamente no processo de desenvolvimento de software.

Quando há a necessidade em manter uma grande quantidade de dados o processo de mineração é o recurso que irá explorar a base através de variáveis que possibilitem encontrar novos conjuntos de dados conforme os existentes, retornando assim alguma informação que até então não estavam diretamente implícitas.

O tempo de resposta ao realizar o procedimento de busca em uma base de dados é proporcional a quantidade de informação retida na mesma, ou seja, quando se tem pouca informação retida o tempo de resposta é pequeno, já para uma quantidade maior o tempo aumenta. Sendo assim, a criação de métodos que visam melhorar o tempo de retorno é importante, principalmente para o cliente que deseja uma resposta rápida ao procurar por alguma informação.

3.1 Mineração de Dados

"O processo de extrair" ou "minerar" conhecimento sobre uma grande quantidade de dados dá-se o nome Mineração de Dados"(HAN; KAMBER, 2006). O recurso de mineração de dados vem sendo utilizado cada vez mais por grandes corporações que trabalham com grandes quantidades de dados, devido as vantagens na geração de novos conhecimentos através dos dados já armazenados.

Para realizar a mineração de dados é necessário ter conhecimento em algumas etapas visando garantir o sucesso na busca de um novo conhecimento:

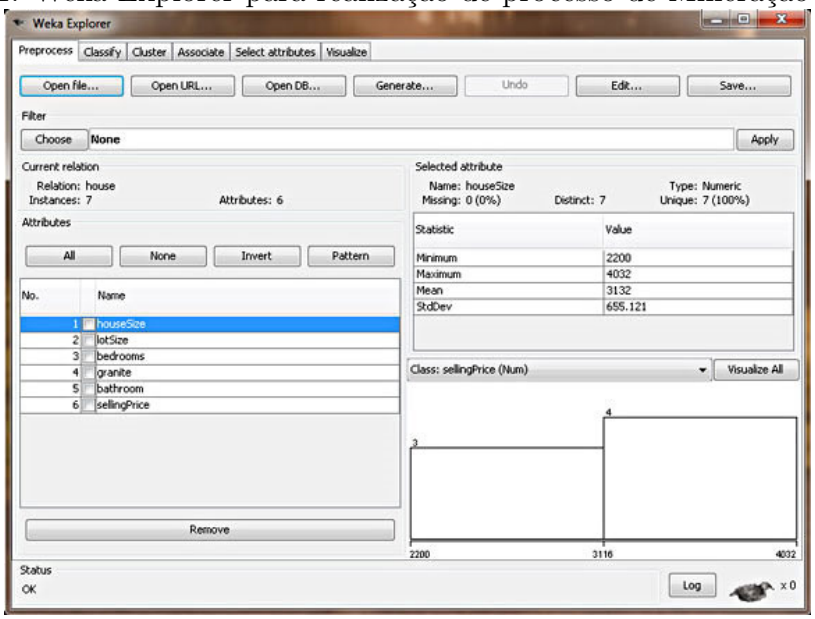
- Definição do problema: O que desejo obter como resposta;
- Aquisição e avaliação dos dados: os dados coletados devem obedecer uma estrutura de dados visando simplificar o entendimento da extração da informação;

- Extração de características e realce: com os dados formatados é verificado quais atributos são de fato interessantes para a resposta que se deseja buscar. Nesse ponto é criado um grupo de dados que já pode ser utilizado ou tratado para melhorar a resposta solicitada;

- Desenvolvimento do modelo e implementação: com a reposta da etapa anterior a resposta já pode ser exibida através de tabelas, gráficos, mapas entre outros.

Para melhorar a interação quando se quer minerar dados, existem ferramentas que auxiliam nesse processo. Um dos mais conhecidos é o Weka - *Waikato Environment for Knowledge Analysis*, desenvolvido na Universidade de Waikato, Nova Zelândia. Com uma variedade de algoritmos, o Weka tornou o processo de mineração mais simplificado, sendo o procedimento realizado via interface gráfica (Figura 3.1).

Figura 3.1: Weka Explorer para realização de processo de Mineração de Dados.



Fonte: (WEKA, 2014)

Um exemplo de mineração seria possuir os dados socioeconômicos do número de pessoa que concluíram o ensino fundamental e médio. Desejando obter a seguinte resposta: quantas pessoas não fizeram faculdade em um determinado local. Para minerar essa informação é necessário ter em conhecer:

- Definição do problema: Quantas pessoas não começaram a faculdade nesse local?;

- Aquisição e avaliação dos dados: somatório de todos da base que não iniciaram a faculdade, somatório de pessoas que fazem o ensino fundamental e médio;
- Extração de características e realce: é separado o grupo de resultados do somatório de acordo com a função mineradora;
- Desenvolvimento do modelo e implementação: o processamento é exibido em algum tipo de relatório (tabular, gráfico ou espacial).

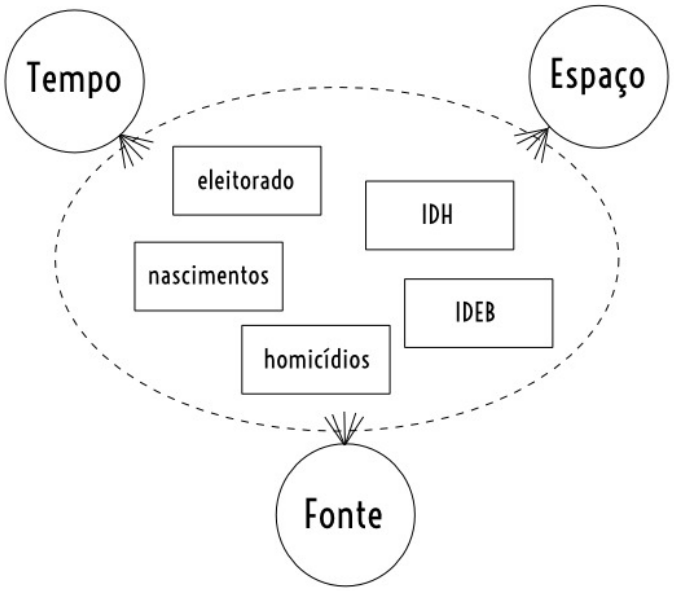
3.2 Framework de Desenvolvimento de Análise de Dados Socioeconômicos

O framework de desenvolvimento de análise de dados socioeconômicos desenvolvido em (LIMA, 2013) padroniza características e funcionalidades para sistemas que tratam de dados socioeconômicos georeferenciados, criando um produto genérico.

A modelagem dos dados utilizada foi baseada em três dimensões: tempo, espaço e fonte (Figura 3.2).

Um exemplo de indicador socioeconômico seria o número de eleitores de uma cidade; exemplo de valores para suas três dimensões principais poderiam ser “São Luís (MA)” para a dimensão espacial, “ano de 2009” para dimensão temporal e “IBGE” para a fonte do dado”(LIMA, 2013).

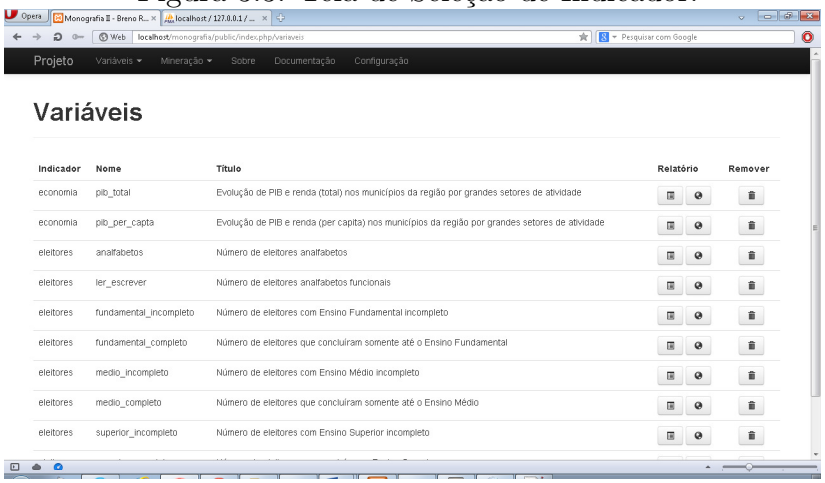
Figura 3.2: Exemplo de indicadores em três dimensões.



Fonte: (LIMA, 2013)

O processo para escolha do relatório é bem simples, dado uma lista de indicadores cadastrados (Figura 3.3) é possível acessar diretamente os relatórios.

Figura 3.3: Tela de Seleção de Indicador.



Fonte: (LIMA, 2013)

Com a modelagem de dados proposta, tem-se a necessidade da geração de relatórios em cima desse dado representado em três dimensões. Dessa forma, existindo o dado e o local é possível gerar três tipos de relatórios: tabular (Figura 3.4) e gráfico (Figura 3.5).

Figura 3.4: Exemplo de Relatório Tabular.

Fonte: (LIMA, 2013)

Neste tipo de relatório quando é escolhido algum indicador que se deseja analisar, ao realizar a consulta é mostrado os dados das três dimensões em forma de tabela ordenada pelo nome do local agrupando seus dados em cada dimensão.

Figura 3.5: Exemplo de Relatório Gráfico.

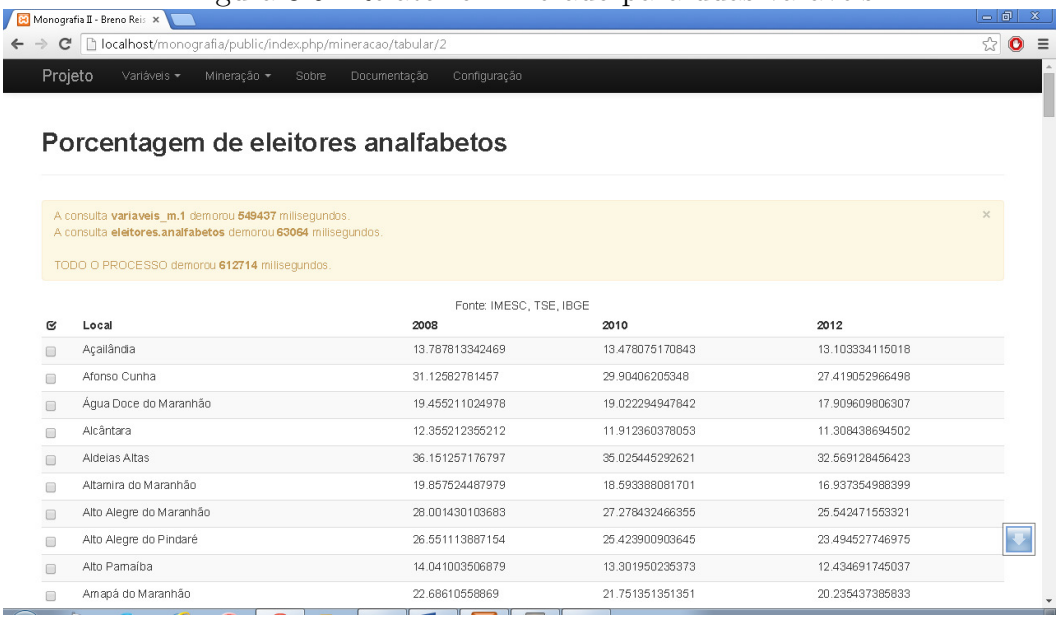
Fonte: (LIMA, 2013)

Quando se tem um relatório gráfico é possível a geração de gráficos comparativos entre os locais retornados ao solicitar o relatório tabular. Esse recurso é interessante quando há necessidade de comparar locais.

Escolhendo os locais que deseja analisar é gerado um relatório gráfico de linhas dos itens selecionados. É importante perceber que uma informação de tempo de processamento da consulta também é calculada e exibida, dessa forma é possível avaliar o desempenho ao processar a requisição para a geração do relatório. Deve-se observar que quanto maior a quantidade de dados a serem retornados, maior é o tempo de resposta.

Os dois relatórios também podem ser obtidos através da mineração de dados (Figura 3.6), ou seja, podemos explorar diversos indicadores analisando-os e gerando novos relatórios.

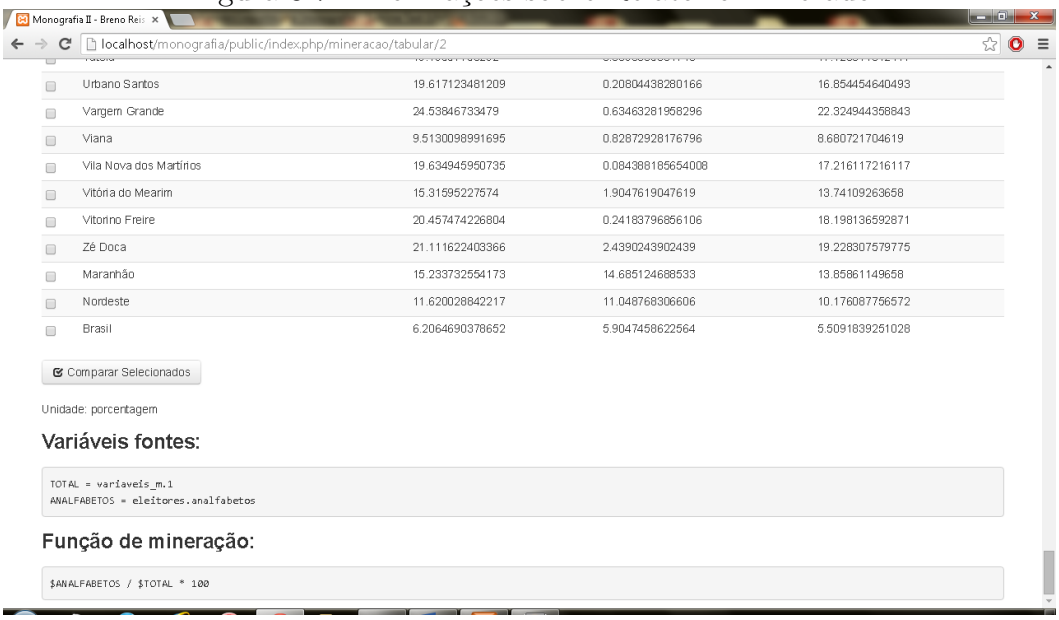
Figura 3.6: Relatório minerado para duas varáveis.



Fonte: (LIMA, 2013)

Para cada relatório minerado é exibida a informação das variáveis fontes do conjunto a ser gerado, bem como a função utilizada para realizar o cálculo de agrupamento dos dados (Figura 3.7).

Figura 3.7: Informações sobre Relatório Minerado.



Fonte: (LIMA, 2013)

O framework criado utiliza a arquitetura MVC (*Model, View e Control*). O componente gerenciador são os controles, estes interagem com as views e os models. Essa arquitetura facilita a manutenção do sistema, dessa forma quando se quer alterar a interface, trabalha-se nas views, na forma como as consultas são realizadas e retornadas temos os modelos.

3.3 Otimização de Consultas

A otimização de consulta é uma alternativa afim de obtermos respostas mais rápidas para as consultas convencionais feitas em banco de dados. Em geral está associada à obtenção da melhor forma de realizar uma consulta demandando o mínimo de recursos necessários.

Pode ser aplicada quando temos lentidão na resposta de consultas ou quando um sistema irá trabalhar com grandes quantidades de dados, com isso as requisições podem ser feitas e retornadas rapidamente para um ou diversos valores.

De forma genérica, quando se fala em otimização de consultas podemos pensar em melhorias na realização da consulta direta, no SQL (*Structered Query Language*), estrutura do SGBD (Sistema Gerenciador de Banco de Dados), tipo de arquitetura entre

outros.

Entretanto, o que será enfatizado é a otimização de consulta não via banco de dados, mas sim através de outros mecanismos que funcionam analogamente a um SGBD convencional, e quando se tem grandes quantidades de dados armazenados estes mecanismos possuem tempo de resposta superior à álgebra relacional, além de oferecer recursos extras que auxiliam o desenvolvedor a criar uma aplicação.

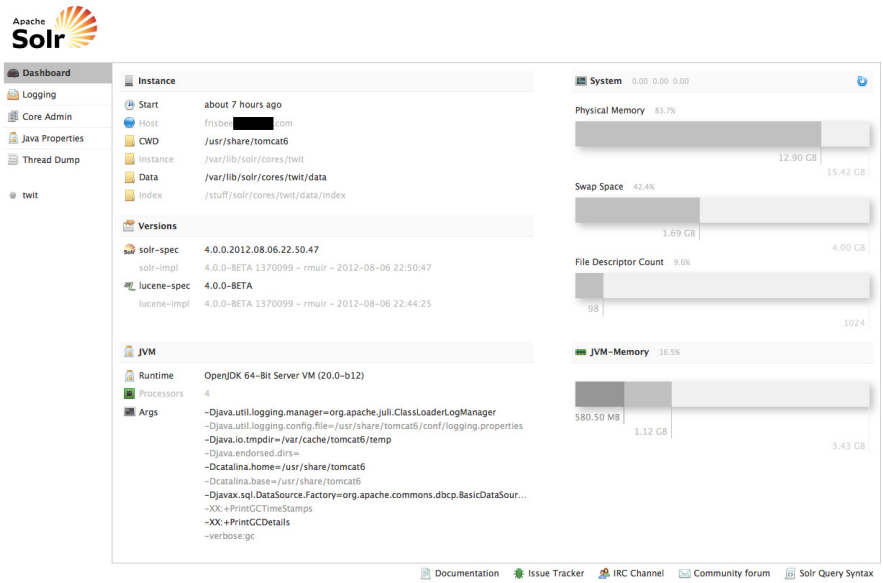
Estes mecanismos serão integrados a nova versão do framework e será facultado ao desenvolvedor utilizá-los.

3.3.1 Solr

O Solr (Figura 3.8) é um servidor de busca Open-Source de alta performance mantido pela Apache Software Fondation, os dados são mantidos pelo núcleo Lucene, uma biblioteca escrita em Java para indexação e busca por textos.

O Solr utiliza o conceito de que cada dado é um documento que possui um identificador interno que é utilizado na indexação do servidor, este documento pode conter os campos que serão indexados junto a cada documento individual.

Figura 3.8: Servidor Solr.

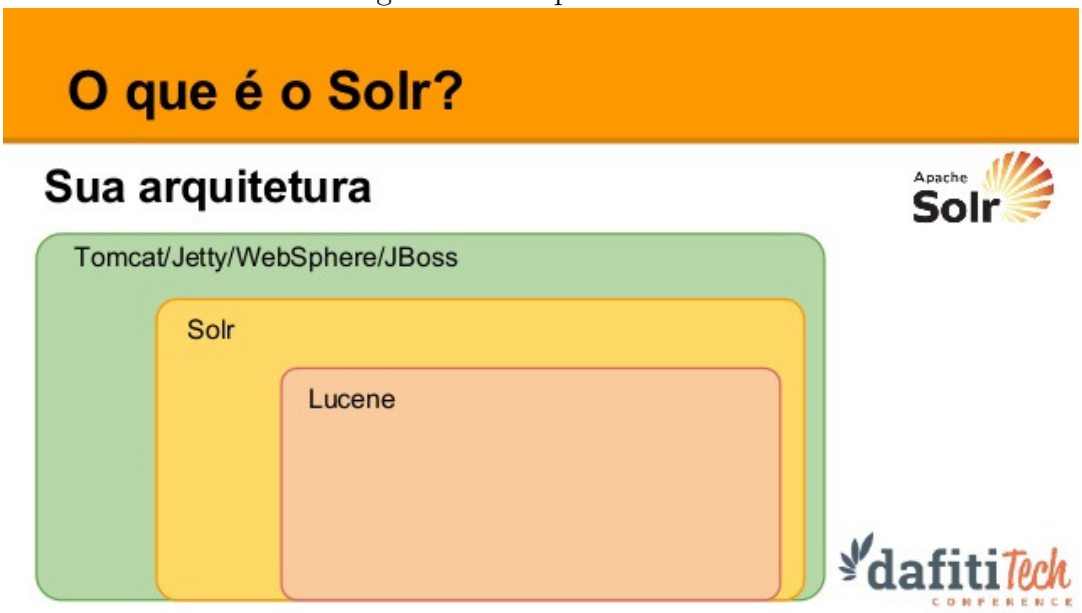


Fonte: (LIMA, 2013)

Sua arquitetura é composta de três camadas (Figura 3.8):

- Jboss: servidor web onde fica a aplicação;
- Solr: servidor de busca;
- Lucene: Framework Open-Source para recuperação de informação.

Figura 3.9: Arquitetura Solr.



Fonte: (KLEYNANZEIGEN, 2011)

Além de oferecer recursos ao desenvolvedor como:

- Busca textual;
- Paginação e ordenação;
- Faceting: Categorização por um atributo comum aos documentos indexados;
- Highlitetext: associada a busca, pode-se destacar conteúdo dentro de textos;
- Auto-suggest: recursos que a medida que é inserida uma informação é ofertada sugestões;
- Faceting: Categorização por um atributo comum aos documentos indexados;
- Spell-checking: corretor ortográfico;

- Geospatial search: quando trabalhado com campos espacial, podem-se executar funções espaciais em cima deste atributo;
- Integração com banco de dados: atualização de índices com suporte a banco.

O Solr pode ser acessado por sistemas que utilizam diversas linguagens de programação, também oferece recursos de retorno de dados em formatos XML, JSON entre outros.

Além disso, segundo pesquisa da Solt TI que mantém um ranking de popularidade entre os usuários de banco de dados, é classificado como a melhor ferramenta de busca. Portanto deve ser considerada quanto há a necessidade de implantar otimização de consultas.

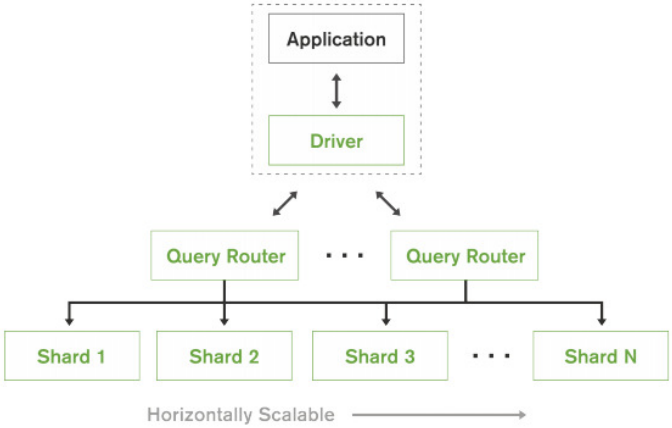
3.3.2 MongoDB

> "MongoDB é um banco de dados orientado a documentos multiplataforma que fornece: alto desempenho, alta disponibilidade e escalável. MongoDB funciona no conceito de coleta de documentos."(ROBOMONGO, 2014)

MongoDB faz parte do movimento NoSQL. Teve início em 2007 pela 10gen e está disponível para os principais sistemas operacionais do mercado, além de possuir drivers que permitem desenvolvimento web e desktop.

Os documentos são distribuídos em Shards (Coleções), o Query Router (Selecionador de rota) baseado no atributo do dado a ser procurado direciona para a coleção que contém as informações do documento (Figura 3.10).

Figura 3.10: Arquitetura MongoDB.



Fonte: (MONGODB, 2014)

Basicamente a estrutura de dados do MongoDB se dá da seguinte forma (Figura 3.11):

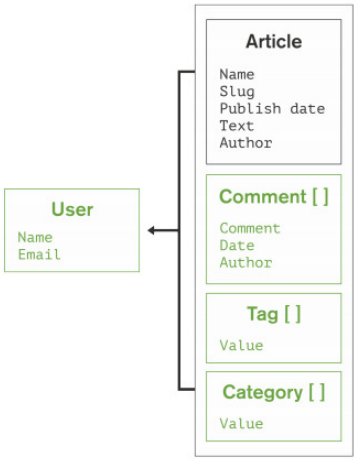
- Um banco de dados armazena um conjunto de coleções;
- Uma coleção armazena um conjunto de documentos;
- Um documento é um conjunto de campos;
- Um campo é um par chave-valor:

  Uma chave é um nome (String);

  Um valor é um (a):

  – Caractere, inteiro, ponto flutuante, timestamp ou binário;
  – Documento;
  – "Array"de valores;

Figura 3.11: Exemplo de um documento MongoDB.

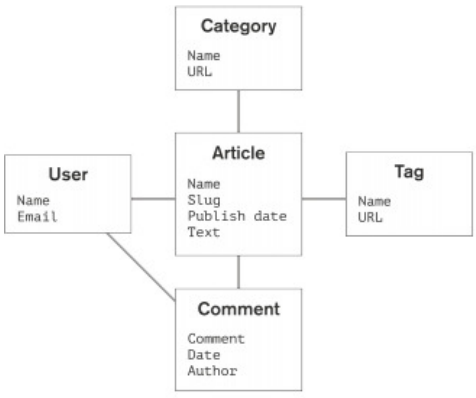


Fonte: (MONGODB, 2014)

Conforme o conceito da orientação a documentos, os dados são guardados em estruturas definidas de acordo com seus atributos, podendo cada documento isolado ter sua estrutura modificada. Entretanto nos bancos de dados relacionais ao adicionarmos uma coluna modificamos a estrutura de toda a tabela, no MongoDB pode-se adicionar atributos em cada documento sem interferir nos demais.

A ideia de coleções de documentos também é utilizada, quando precisamos relacionar documentos entre si, tem-se duas opções criar um atributo documento interno (Figura 3.12) ou quando se há necessidade de relações externas criam–se referências de uma coleção de documentos para outra.

Figura 3.12: Organização de Documentos no MongoDB.



Fonte: (MONGODB, 2014)

4 Proposta

O dado geográfico de um determinado local é considerado um dado complexo, assim é necessário elaborar mecanismos que facilitem o retorno desta informação em uma fonte de dados. Além do georreferenciamento existem as informações socioeconômicas de uma região, que por regra aumenta a complexidade do retorno de todas as informações ao ser realizada uma busca para qualquer que seja a aplicação solicitante.

Visando facilitar tanto a elaboração e o acesso aos dados complexos ou não em um framework de desenvolvimento de sistemas de análise de dados socioeconômicos, um estudo de ferramentas que possam otimizar as consultas de sistemas criados a partir do framework é relevante.

Partindo desse objetivo, este trabalho visa integrar os serviços que realizam as buscas de forma otimizadas junto a versão inicial do framework proposto por LIMA (2013) intitulado Proposta de Framework para Sistemas de Mineração de Dados Socioeconômicos. Através de três fontes de dados: MongoDB, MySQL e Solr aproveitando suas características e de acordo com testes avaliativos selecionar a melhor opção para ser implantada, prevendo assim os casos onde a busca retornará grandes quantidades de resultados complexos não se tornem demasiadas lentas.

4.1 Configuração

Como o framework e o Sistema Gerenciador de Banco de Dados (SGBD) já estão definidos, foram necessárias poucas alterações na estrutura interna do sistema. O framework utiliza-se do padrão MVC *(Model View Control)*. Para não serem realizadas grandes mudanças nas classes utilizadas, foram realizadas alterações nos métodos de busca da informação, assim a camada de modelos foi a que teve as principais inserções de códigos.

4.1.1 Servidores de Otimização de Consultas

Os mecanismos pesquisados que são utilizados para otimizar a busca necessitam de um servidor para que possa ser realizada as requisições pelo cliente. Como o objetivo é a otimização na busca de dados complexos por uma aplicação desenvolvida através do framework, serão realizados testes de retorno dos dados diretamente na API fornecida pelos servidores utilizados para saber como irão se portar com o retorno de dados completos. Posteriormente será utilizada uma aplicação criada que irá realizar os testes.

Dessa forma, junto dos testes iniciais serão feitas solicitações para que sejam retornados todos os dados dos locais, por conterem os dados das geometrias esta dados terão um foco principal no processo de otimização.

A configuração dos servidores MongoDB, MySQL e SOLR será a seguinte:

- Processador: Intel (R) Core (TM) i5-2410M CPU @ 2.30 GHz 2.30GHz;
- Memória: 6 GB Total (4GB alocados para o serviço);
- Sistema Operacional: Windows 7 Professional 64 Bits Service Pack 1;
- Capacidade de Armazenamento: 650 GB.

4.1.2 Estrutura Banco de Dados

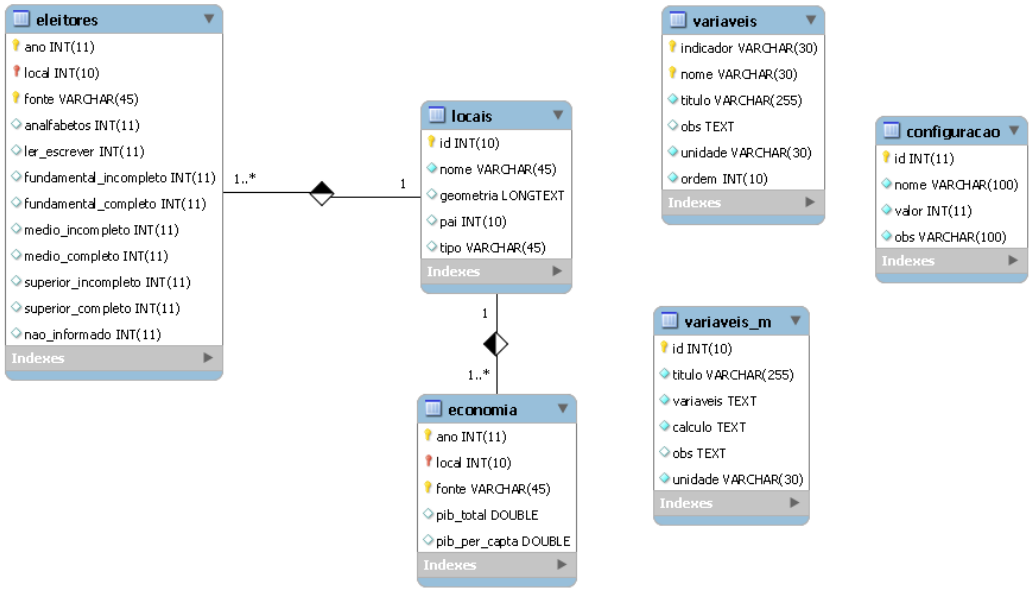
No contexto atual do banco de dados relacional do framework foi utilizada a modelagem multidimensional, onde, tem-se uma tabela centralizadora (fatos) que é ligada através das chaves estrangeiras a tabela de dimensões. Fatos são os dados a serem agrupados e a dimensão estabelece a organização dos dados.

> Como os dados socioeconômicos georreferenciados são organizados em pelo menos três dimensões principais: dimensão temporal, dimensão espacial ou geográfica e a fonte do dado. Nesse caso é considerado um dado multidimensional. Um dado multidimensional é organizado ao redor do tema central ao qual chamamos de fato. (LIMA, 2013)

Dessa forma, temos o Diagrama Entidade Relacionamento (Figura 4.1) que representa os fatos (tabelas de eleitores e economia) associada as dimensões ano, local e fonte. Também temos as tabelas auxiliares variáveis e variáveis_m, onde cada registro é um

relatório que possui dados nas tabelas de fato, além da tabela configuração, utilizada para configurar procedimentos de aplicações criadas com o uso do framework. Por se tratarem de dados monovalorados é possível que os dados referentes a dimensão pertençam a tabela de fatos, a unicidade é garantida pela chave primária composta dos identificadores: ano, local e fonte.

Figura 4.1: D.E.R: Nova versão do framework



Fonte: (AUTORIA PRÓPRIA)

Em uma aplicação criada utilizando o framework será exibida a configuração demonstrada na Figura 4.2.

Figura 4.2: Tela de seleção de fonte de dados ativa

Fonte: (AUTORIA PRÓPRIA)

Por padrão a opção selecionada é a consulta feita diretamente no banco de dados MySQL, assim o desenvolvedor pode optar por qual meio será realizada a busca

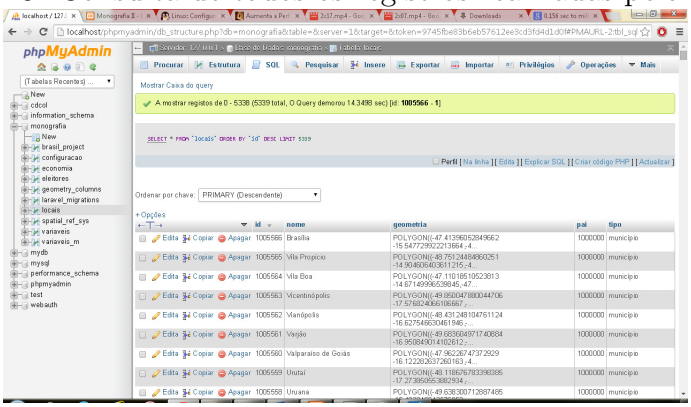
pelos dados que está sendo solicitada.

Além disso, foram adicionadas as informações georreferenciadas de todos os municípios e estados brasileiros bem como algumas regiões que abrangem mais de um local, dessa forma temos um total de 5339 registros na tabela de locais contendo dados complexos de geometrias, totalizando assim 43,7Mb totais armazenados.

4.1.2.1 Servidor MySQL

A instalação padrão do MySQL foi utilizada, contudo, designado memória disponível para os 4 GB citados nos modelos de configuração. A consulta de todos os registros (Figura 4.3) diretamente na API do MySQL demorou 14349,8 milissegundos.

Figura 4.3: Consulta de todos os registros realizadas pelo MySQL.



Fonte: (AUTORIA PRÓPRIA)

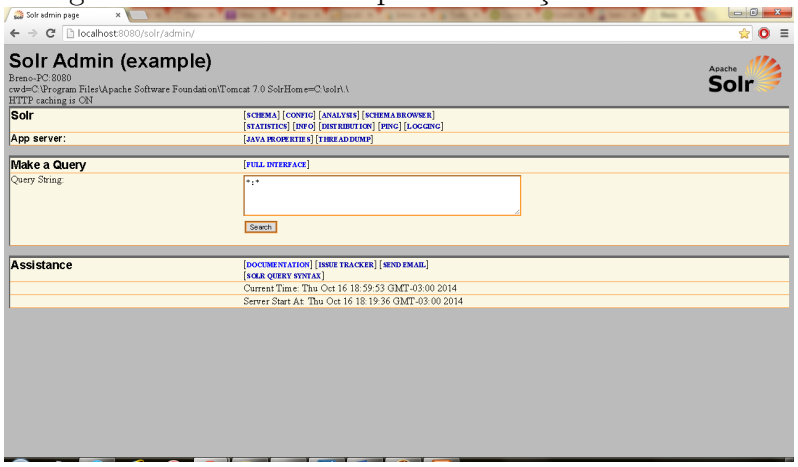
Teve como resposta 5339 linhas com dados simples: id, nome, pai, tipo e complexo: geometria, este que impacta diretamente no desempenho ao processar a consulta e devolver a resposta.

4.1.2.2 Servidor Solr

Após a instalação do SOLR , foi utilizada a consulta de todos os registros foi realizada através de sua API (Figura 4.4).

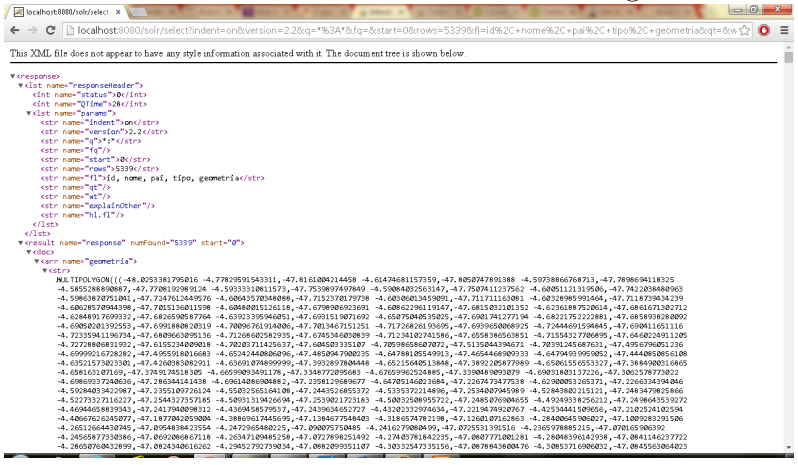
Através da API do Solr, foi realizada a busca visando retornar todos os dados dos locais (id, nome, geometria, pai, tipo).

Figura 4.4: API do Solr para realização de consultas.



Fonte: (AUTORIA PRÓPRIA)

Figura 4.5: Consulta Solr retornando todos os registros de locais.



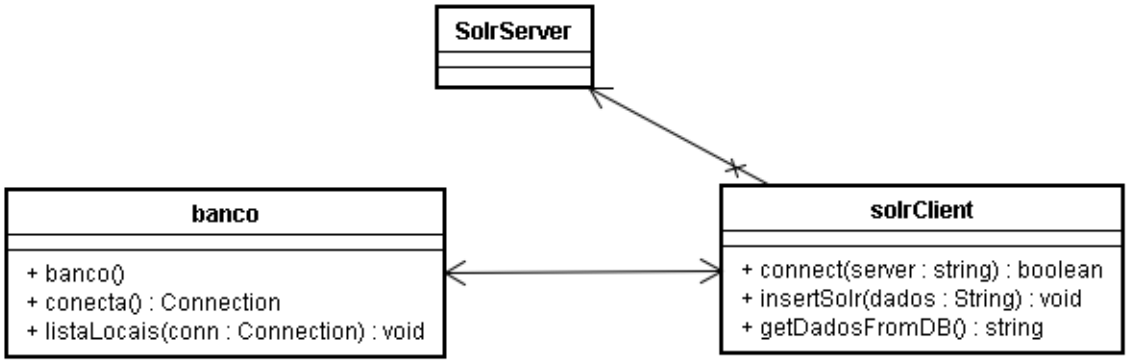
Fonte: (AUTORIA PRÓPRIA)

Como resposta (Figura 4.5) o Solr retornou também no parâmetro QTime que informa a quantidade de tempo do retorno da consulta, para a consulta das informações dos locais demorou 28 milissegundos.

Para cadastrar os dados necessários foi necessário criar uma aplicação cliente que irá alimentar o servidor. Como o Solr fornece bibliotecas em diversas linguagens de programação, foi selecionada a linguagem Java.

O diagrama de classes (Figura 4.6) demostra que existem 3 classes que inserem as informações no servidor Solr.

Figura 4.6: Diagrama de Classe para inserção de dados no Solr.



Fonte: (AUTORIA PRÓPRIA)

De forma manual a aplicação é executada, assim o cliente irá requisitar ao banco de dados as informações para que sejam indexadas no Solr. Entretanto pode-se agendar a execução da mesma, através de agendadores de tarefas que em determinados intervalos de tempo executam o cliente para ter a base Solr sempre atualizada.

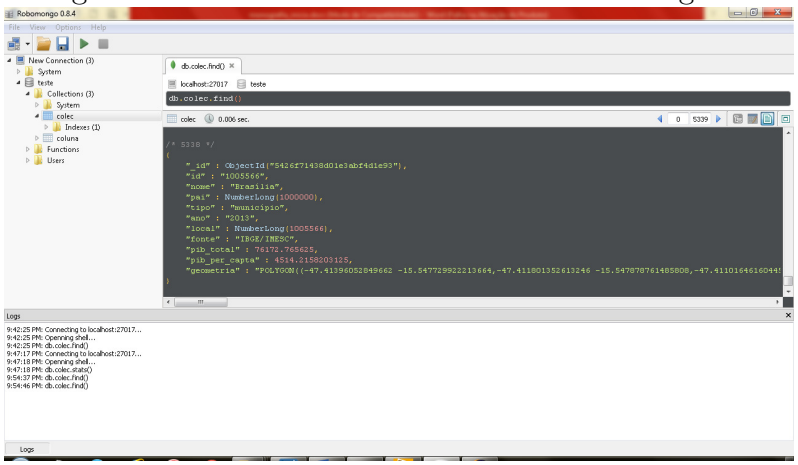
Contudo, quando surgirem novos dados o Solr permite que a inserção seja feita diretamente por uma aplicação cliente, ou seja, não é necessária a existência de um SGBD relacional para atualizar o servidor de busca. Com o servidor Solr criado e configurado, o framework pode realizar buscas através de outras fontes além do MySQL.

4.1.2.3 Servidor MongoDB

Foram criados um banco de dados (DB) e uma coleção (Collection) que permitem a adição das informações retiradas inicialmente do banco de dados MySQL.

Entretanto o MongoDB oferece um terminal para realização de operações, a informação do tempo de resposta não pode ser obtida, dessa forma foi utilizado uma API cliente para conectar ao servidor e realizar as consultas iniciais, através do Robomongo (ROBOMONGO, 2012) foi realizada a busca (Figura 4.7) visando retornar todos os dados dos locais (id, nome, geometria, pai, tipo).

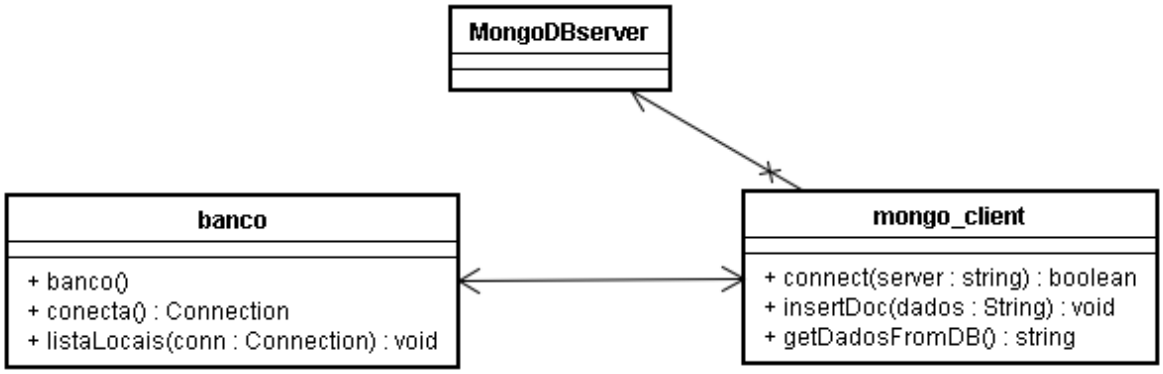
Figura 4.7: Consulta via API no servidor MongoDB.



Fonte: (AUTORIA PRÓPRIA)

Como resposta o MongoDB informou a quantidade de tempo em segundos (0,006) do retorno da consulta, convertendo o resultado, o tempo foi 6 de milissegundos. Similar ao Solr, para realizar o cadastro das informações foi criado um cliente simples que lê as informações de um banco de dados MySQL, e cria os documentos a serem inseridos nas coleções mongo de acordo com o diagrama de classe abaixo (Figura 4.8).

Figura 4.8: Diagrama de Classe para inserção de dados no MongoDB.



Fonte: (AUTORIA PRÓPRIA)

4.1.3 Framework Atual

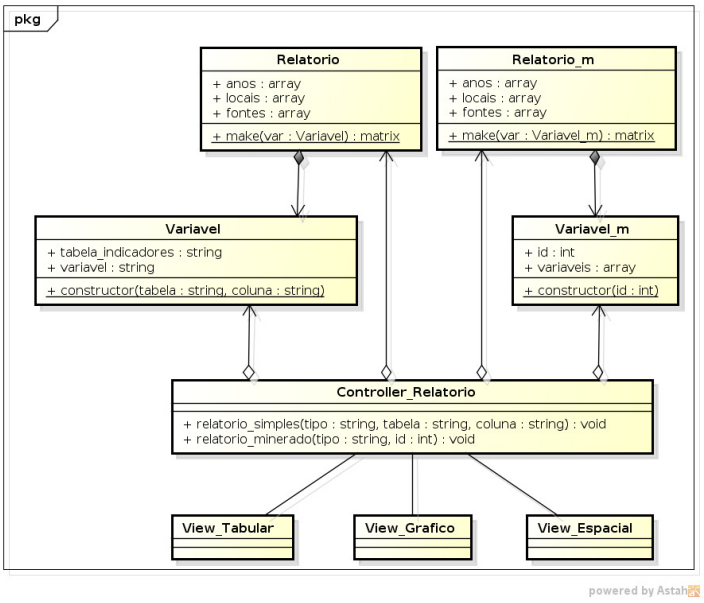
No código fonte do framework foram criadas uma classe controladora (configuração_controller) e um model (configuração) que irão realizar as operações na

tabela configuração no sentido de estar retornando e alterando como serão realizadas as consultas para a geração de relatórios.

Com essa modificação na estrutura do framework, sempre que for realizada a chamada do método make para gerar um relatório, de acordo com o retorno do método DB::table('configuracao')-¿get() a consulta dos dados será realizada em uma determinada fonte de dados disponíveis.

Dessa forma, o diagrama de classes do framework (Figura 4.9) será mantido, ou seja, foram alterados somente alguns métodos já existentes.

Figura 4.9: Diagrama de Classes do Framework.



Fonte: (LIMA, 2013)

5 APLICAÇÃO TESTE

Para a execução dos testes foi desenvolvida uma aplicação a partir do framework modificado. A implementação consiste em fornecer, via web, acesso a consulta de dados de indicadores socioeconômicos de municípios e estados brasileiros em três tipos de relatórios: tabular, gráfico e espacial.

As visões foram construídas com HTML, PHP e o Bootstrap. Para tratar a informação espacial foi utilizado a biblioteca Openlayers. Foi utilizado o framework Laravel 3 (LARAVEL, 2013) que implementa a arquitetura MVC.

Foram aplicadas três fontes de dados: MongoDB, MySQL e Solr com o intuito de aproveitar suas características e realizar testes para avaliar o desempenho de cada um. Após isso será feita a indicação do serviço melhor se adequa para ser implementado junto ao framework.

Para cada fonte de dados foram necessárias a importação de suas bibliotecas para uso do cliente PHP que irá realizar as consultas.

5.1 Visões

A aplicação para realização dos testes possui três principais visões (telas):

- Tabular: Relatório com dados em formato de tabela presente na Proposta de Framework para Sistemas de Mineração de Dados Socioeconômicos (LIMA, 2013);
- Gráfico: Relatório com dados em formato gráfico presente na Proposta de Framework para Sistemas de Mineração de Dados Socioeconômicos (LIMA, 2013);
- Espacial: Relatório com dados em formato de mapa temático.

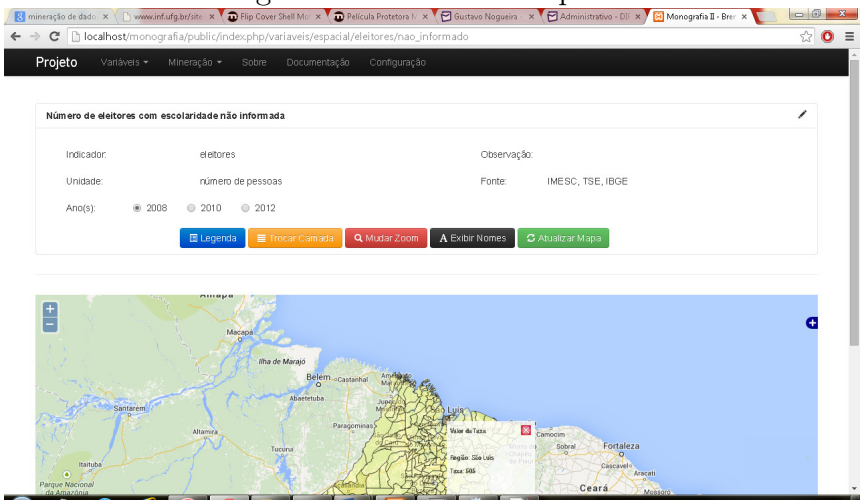
Nesta tela é exibida uma lista de indicadores cadastrados para relatórios com informações mantidas na fonte de dados, também podem ser gerados relatórios minerados oriundos de múltiplas variáveis conforme cadastro contendo especificações de como são utilizadas as variáveis e as métricas de mineração.

5.1.1 Espacial

Este relatório é gerado a partir da tela de listagem de variáveis (Figura 3.3) exibe em um mapa as informações obtidas de acordo com os locais. É feita a média dos valores obtidos e gerada seis faixas de valores, para cada dado de um determinado local que se enquadre em uma destas faixas é atribuída uma coloração corresponde no mapa respectiva a faixa gerada (Figura 5.1). Ainda é possível consultar o valor de cada local clicando na região do mesmo, bem como a troca de camadas que selecionar qual camada de informação deseja visualizar, seja de Municípios, Estados ou Regiões.

Mesmo com a fonte de dados nas três dimensões, é exibido um mapa temático por ano, assim, para cada ano temos um mapa diferente.

Figura 5.1: Relatório Espacial.



Fonte: (AUTORIA PRÓPRIA)

5.1.2 Configuração

A partir da visão configuração (Figura 5.2) é possível selecionar em qual fonte de dados a consulta será realizada. Além disso é possível que o desenvolvedor possa estar adicionando novas configurações para que sejam utilizadas futuramente.

Figura 5.2: Tela de Configuração.

Fonte: (AUTORIA PRÓPRIA)

5.2 Controles

Foram necessários duas classes controladoras, uma para tratar dos relatórios simples e outra para os minerados, existindo dois métodos principais (Figura 5.3): action_index() que controla a exibição da listagem de variáveis e os métodos action_REL() onde REL pode ser tabular, gráfico ou espacial. Este método irá buscar em uma determinada fonte de dados a informação necessária para gerar o relatório escolhido.

Figura 5.3: Métodos do Controlador principal do framework.

```
public function action_index()
{ ... }

public function action_tabular($table, $column)
{ ... }

public function action_espacial($table, $column)
{ ... }

public function action_grafico($table, $column)
{ ... }
```

Fonte: (AUTORIA PRÓPRIA)

5.3 Modelos

Através do método make das classes de modelo é realizada uma consulta para verificar onde será feita a consulta posterior, visando o retorno dos dados para gerar o

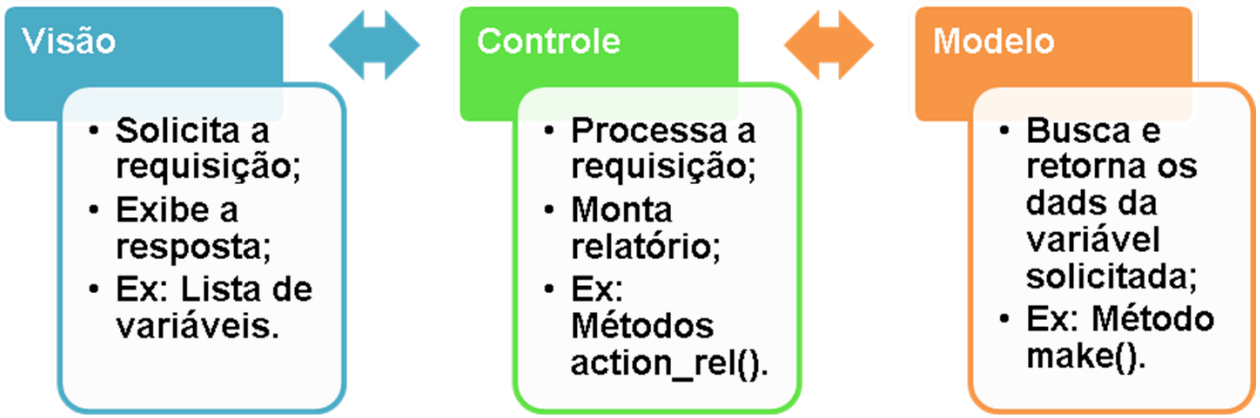
relatório.

Através do método get-¿configuracao() será realizada uma consulta que irá verificar em qual fonte será realizada a consulta dos dados do relatório. Após isso, o próximo passo é a busca dos dados em uma fonte especifica retornada pelo get anterior, os dados vêm em uma estrutura compreensível à aplicação cliente.

Tendo os servidores necessários previamente configurados foram adicionadas as consultas via MongoDB e Solr.

Com a arquitetura definida ao framework, uma aplicação criada a partir dele tem na figura 5.4 o relacionamento demonstrado entre as camadas de visão, controle e modelo utilizados.

Figura 5.4: Fluxograma MVC.



Fonte: (AUTORIA PRÓPRIA)

6 Testes

Para a execução dos testes, é necessário a distinção entre relatórios simples e minerados compreende que: um relatório simples está presente na base de dados de forma explicita (o dado existe em sua forma original), já um relatório minerado é aquele onde se deseja minerar alguma informação e é necessário o cruzamento dos dados existentes na base, ou seja, a geração da informação parte de dados existentes que juntos possam formar a resposta que se deseja. Ao realizar os testes foi desenvolvido uma aplicação visando medir o desempenho das fontes dados MySQL, Solr e MongoDB foram definidos os seguintes critérios para realização dos testes em cada servidor:

1. Devem-se buscar 50, 100, 250, 500, 1000 e o total geral de registros, com o objetivo de elaborar os gráficos de desempenho;
2. Os dados buscados devem conter as informações geográficas dos locais, afim de avaliar o desempenho das consultas em cima de dados complexos;
3. Deve-se medir o tempo da consulta (envio da solicitação até a resposta do servidor) em cada consulta;
4. As consultas serão baseadas em um único ano (2013);
5. O teste realizado através da mineração de dados terá dois indicadores para o cálculo;
6. Serão realizadas 5 consultas para cada valor de registros citados em 1, com o objetivo de calcular a média de tempo necessário para buscar os resultados;
7. A configuração do navegador (Google Chrome) utilizará o modo anônimo, este modo não armazena dados em cache.

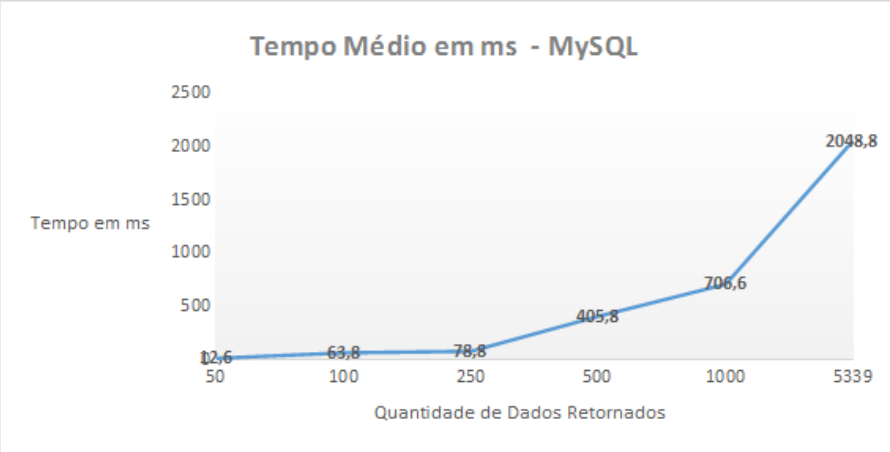
Com esses critérios será definido qual o melhor serviço para implementá-lo na nova versão do framework.

6.1 Resultados obtidos com o uso do MySQL

Tanto para as consultas mineradas (Figura 6.1) quanto por variáveis presentes na base de dados (Figura 6.2) as consultas via servidor MySQL se mostraram proporcional a quantidade de dados retornados, ou seja, quanto maior a quantidade de dados maior é o tempo necessário para solicitar e receber a informação.

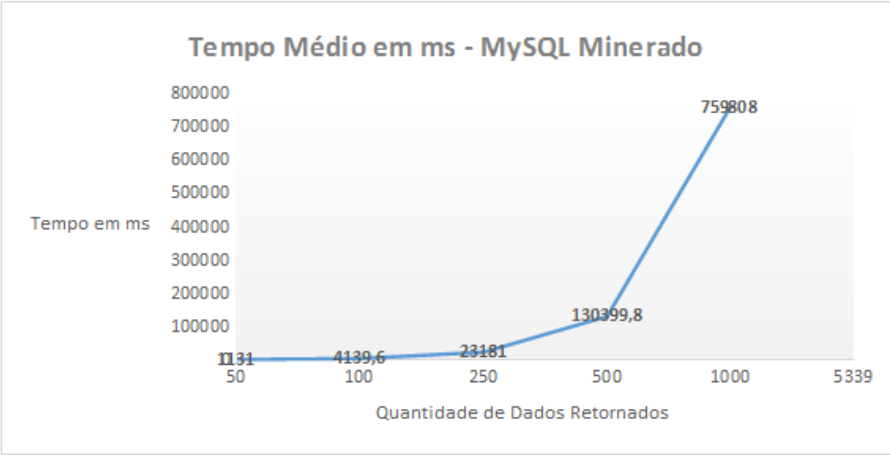
Enquanto as consultas por variáveis levaram no máximo 2 segundos, as consultas mineradas, devidos a grande quantidade de dados demoraram, em alguns casos, mais de 5 minutos, sendo assim não se tornou viável devido à grande espera pela resposta.

Figura 6.1: Tempo médio de resposta via MySQL.



Fonte: (AUTORIA PRÓPRIA)

Figura 6.2: Tempo médio de resposta para consultas mineradas via MySQL.

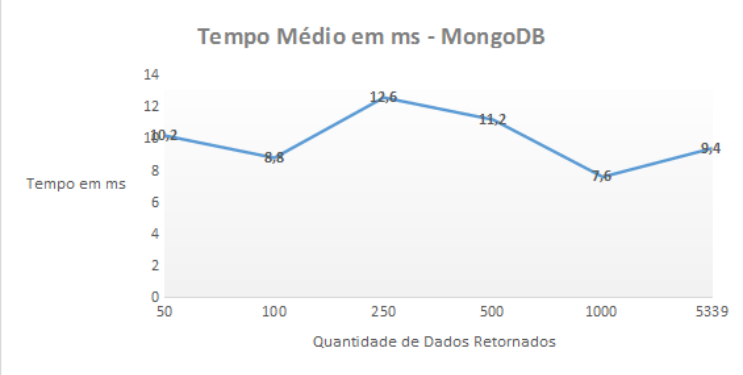


Fonte: (AUTORIA PRÓPRIA)

6.2 Resultados obtidos com o uso do MongoDB

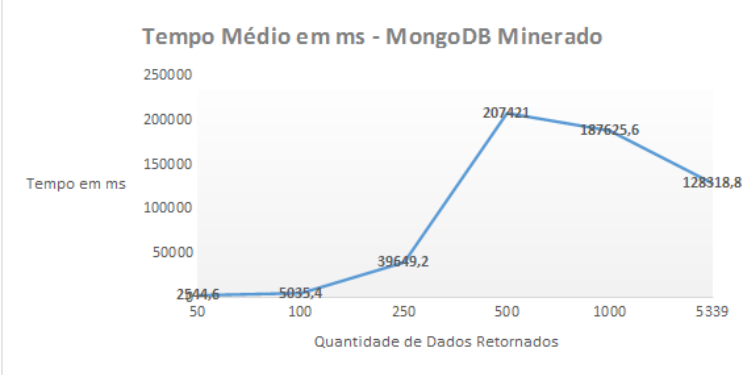
O MongoDB teve variações nos resultados solicitados, entretanto apresentou respostas rápidas, porém para grandes quantidades de dados em alguns testes obteve-se o resultado mais rápido do que nos limites de dados iniciais (Figura 6.3). Para os dados minerados (Figura 6.4) ocorreu um crescente no tempo em relação a quantidade de dados retornados, porém o tempo retrocedeu a partir do limite 500.

Figura 6.3: Tempo médio de resposta via MongoDB.



Fonte: (AUTORIA PRÓPRIA)

Figura 6.4: Tempo médio de resposta para consultas mineradas via MongoDB.



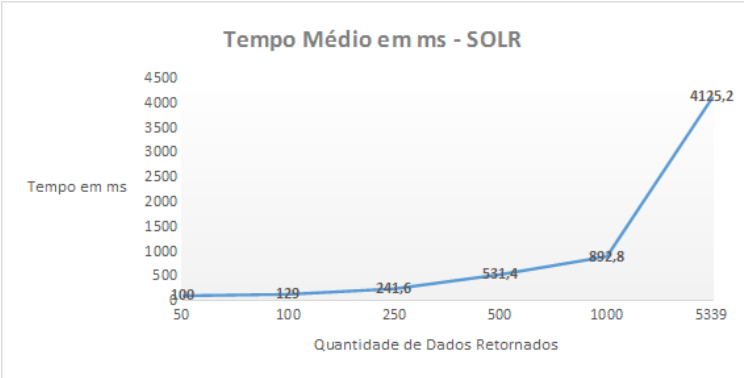
Fonte: (AUTORIA PRÓPRIA)

6.3 Resultados obtidos com o uso do Solr

O Solr apresentou respostas proporcionais a quantidade de dados retornadas (Figura 6.5), conforme aumenta o número de itens a serem retornados o tempo se torna maior.

As consultas mineradas obtiveram um alto tempo de resposta, em alguns casos não foi possível obter o tempo médio devido ao alto tempo para o retorno (Figura 6.6) da requisição HTTP.

Figura 6.5: Tempo médio de resposta via Solr.



Fonte: (AUTORIA PRÓPRIA)

Figura 6.6: Tempo médio de resposta para consultas mineradas via Solr.

Fonte: (AUTORIA PRÓPRIA)

6.4 Resultados Gerais

Com os resultados dos testes foram elaboradas duas tabelas, uma para os relatórios de uma variável (Tabela 6.1) e outra para os relatórios minerados (Tabela 6.2).

Tabela 6.1: Tabela de Desempenho dos Servidores Testados em ms.

QTD de dados	50	100	250	500	1000	5339
MongoDB	10,2	8,8	12,6	11,2	7,6	9,4
MySQL	12,6	63,8	78,8	405	706,6	2048
SOLR	100	129	241,6	531,4	892,8	4125,2

De acordo com os resultados obtidos percebeu-se o MongoDB foi o que melhor se comportou de um modo geral. Obteve tempo de resposta menor que os demais servidores.

Tabela 6.2: Tabela de Desempenho dos Servidores Testados em ms - Relatórios Minerados.

QTD de dados	50	100	250	500	1000	5339
MongoDB	2544,6	5055,4	39649,2	207421	187625,6	128318,8
MySQL	1131	4139,6	23181	130399,8	759808	NDA
SOLR	2925,4	4275,8	24222	119099	762012	899959,8

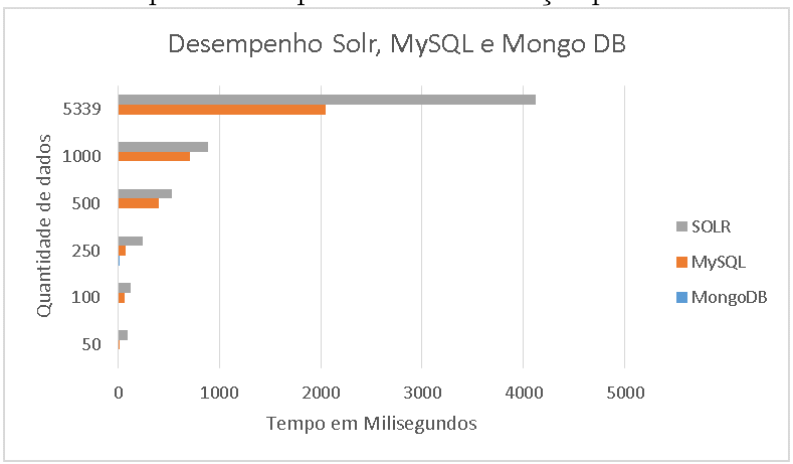
Já nos relatórios minerados o servidor teve melhor desempenho o MySQL, seguido do MongoDB e do Solr.

Além disso a Figura 6.7 demonstra o desempenho de cada serviço lado a lado, enfatizando o desempenho do MongoDB em relação aos demais servidores no que diz respeito aos relatórios simples. O mesmo vale para os relatórios minerados, onde o MySQL teve o melhor desempenho entre os servidores pesquisados.

De um modo geral percebe-se que o MongoDB teve o melhor desempenho nas varáveis e o MySQL foi melhor no quesito mineração. Portanto, o MongoDB é a opção devido aos resultados rápidos para grandes quantidades de dados. Dessa forma, é indicada a integração do mesmo ao framework de geração de relatórios socioeconômicos.

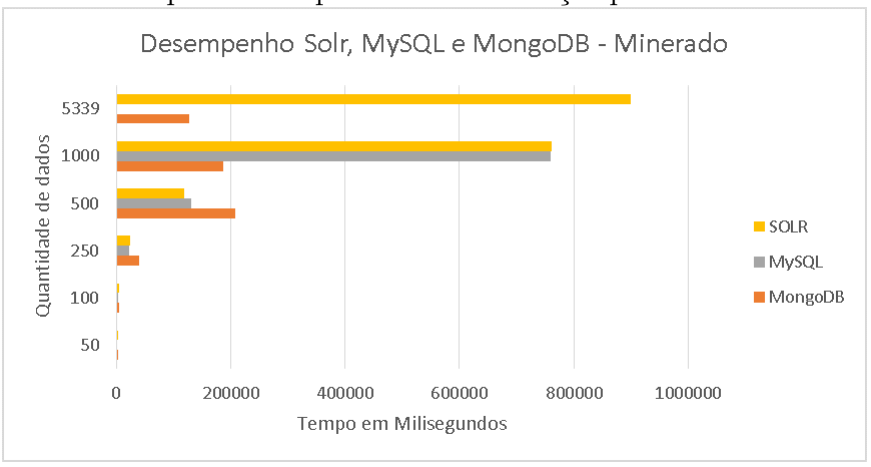
Contudo, cada servidor possui recursos que podem ser utilizados simultaneamente ou até mesmo em paralelo, afim de que o usuário final possa ser beneficiado, dessa forma a utilização destes podem ser consideradas futuramente.

Figura 6.7: Desempenho comparativo dos serviços para relatórios simples.



Fonte: (AUTORIA PRÓPRIA)

Figura 6.8: Desempenho comparativo dos serviços para relatórios minerados.



Fonte: (AUTORIA PRÓPRIA)

7 CONCLUSÕES

O trabalho teve como objetivo melhorar o tempo de resposta de um framework de desenvolvimento de sistemas de dados socioeconômicos. Para isso foram necessários estudos sobre métodos de otimização de consulta para grandes volumes de dados e como aplicar os mesmos no framework e criação de uma aplicação teste utilizando os procedimentos otimizados de consulta que provem a eficácia dos métodos.

Quando foram adicionados todos os municípios e estados brasileiros juntamente da informação de suas geometrias, as consultas obtiveram um tempo maior para se obter as respostas e em alguns casos passaram do limite de 15 minutos estipulado pelo servidor apache. Diante dessa situação e da importância da informação geográfica para a geração dos relatórios e uso dos sistemas gerados a otimização figurou como recurso essencial.

A aplicação desenvolvida a partir do framework já com os recursos de otimização de consultas atendeu os objetivos do que foi proposto neste trabalho, entre os serviços pesquisados, o MongoDB resolveu os problemas de tempo de resposta para as consultas de um grande volume de dados com o retorno da informação geográfica, solucionando assim a pendência do desempenho das consultas na versão anterior do framework.

Futuramente, podem ser associados novos métodos de otimização de consultas ao framework para melhorar o desempenho dos relatórios a serem gerados, implementar os recursos específicos de cada serviço de otimização no framework para que o desenvolvedor possa optar em usá-los, integrar ao framework um servidor que contenha os serviços utilizados do modo que a instalação de servidores físicos seja opcional.

Referências Bibliográficas

[1] CAMILO, C. O.; SILVA, J. C. *Mineração de Dados: Conceitos, Tarefas, Métodos* Monografia (Bacharelado em Ciência da Computação) - Universidade Federal de Goiás, Goiânia, 2009.

[2] HAN; KAMBER. *Data Mining: Concepts and Techniques.* 2. ed: Elsevier, 2006.

[3] HEYMANN, H. *Lead Server Engineer no Foursquare.* 2011. Monografia (Tecnologia em MongoDB) - MongoNYC, New York, NY and San Francisco, CA, 2011.

[4] LIMA, J. *Proposta de Framework para Sistemas de Mineração de Dados Socioeconômicos.* 2013. 53f. Monografia (Bacharelado em Ciência da Computação) - Universidade Federal do Maranhão - UFMA, São Luís, 2013.

[5] OLIVEIRA, H. R.; LÓSCIO, B. F.; PONTES, J. C. S. *NoSQL no desenvolvimento de aplicações Web.* SBSC, Paraty - RJ, 2011. Disponível em: <http://www.addlabs.uff.br/sbsc_site/SBSC2011_NoSQL.pdf> Acesso em junho de 2014

[6] BOOTSTRAP. *Bootstrap.* Disponível em: <http://getbootstrap.com/> Acesso em outubro de 2014

[7] CARTOLAFC. *CartolaFC - uma aplicação python rápida e escalável.* Disponível em: <http://goo.gl/nFkFWe> Acesso em agosto 2014

[8] DBRANKING. *DB Ranking.* Disponível em: <http://db-engines.com/en/ranking¿ Acesso em setembro de 2014

[9] EBAY. *Ebay.* Disponível em: <http://www.ebay.com/> Acesso em agosto de 2014

[10] FOURSQUARE. *Foursquare.* Disponível em: <https://pt.foursquare.com> Acesso em agosto de 2014

[11] LARAVEL. *Laravel.* Disponível em: <http://laravel.com/docs/4.2> Acesso em agosto de 2014

[12] LUCENESOLR. *Lucene Solr.* Disponível em: <http://lucene.apache.org/solr/> Acesso agosto de 2014

[13] MONGODB. *MongoDB.* Disponível em: <http://www.mongodb.org/> Acesso em agosto de 2014

[14] MONGODBGUIDE. *MongoDB Architecture Guide.* Disponível em: <http://info.mongodb.com/rs/mongodb/images/MongoDB_Architecture_Guide.pdf> Acesso em setembro de 2014

[15] OPENLAYERS. *Openlayers.* Disponível em: <http://openlayers.org/en/v3.0.0/apidoc/> Acesso em outubro de 2014

[16] SOLARIUM. *Solarium.* Disponível em: <http://www.solarium-project.org/documentation/> Acesso em outubro de 2014

[17] SOLR. *Solr Evay Kleinanzeigen.* Disponível em: <http://pt.slideshare.net/LucidImagination/solr-ebay-kleinanzeigen> Acesso em setembro de 2014

[18] WEKA. *Weka 3 - Data Mining with Open Source Machine Learning Software in Java.* Disponível em: <http://www.cs.waikato.ac.nz/ml/weka> Acesso em agosto de 2014