Matheus Filipe Sieiro Vargas

Atualização de *gazetteer* ontológico para utilização de recursos de banco de dados geográficos

Belo Horizonte, Minas Gerais
2021

Matheus Filipe Sieiro Vargas

Atualização de *gazetteer* ontológico para utilização de recursos de banco de dados geográficos

Proposta de Pesquisa Tecnológica

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Clodoveu Augusto Davis Júnior

Belo Horizonte, Minas Gerais
2021

Sumário

1 Introdução

Dentre todos os aspectos acerca da globalização, o crescimento do compartilhamento de informações antes limitadas a uma única região é o grande destaque. A internet atuou na potencialização do acesso a essa informação, aumentando a quantidade de dados que podem ser relacionados entre si e permitindo que usuários em todas as partes do globo estejam a poucos cliques de milhares de fontes de informação sobre um mesmo assunto. Esse crescimento incentivou também, a evolução de tecnologias de armazenamento para suprir os mais diversos campos de estudo.

Em particular, o campo da geografia utiliza o armazenamento estruturado para munir os chamados sistemas de informação geográfica (SIG) [1, 2], objetivando compreender e estudar fenômenos apresentados sobre o espaço.

Entretanto, existe ainda um grande volume de dados que podem ser operados de forma não estruturada e, a estes, podemos correlacionar os chamados sistemas de recuperação de informação geográfica, mais conhecidos como *geographical information retrieval* (GIR) [3], que possuem como objetivo suprir as necessidades dos usuários em pesquisas relacionadas a localizações e informações geoespaciais.

Para tanto, sistemas deste tipo devem ser capazes de realizar operações de conversão de consultas do tipo texto em informações não ambíguas sobre dados geográficos. Os processos associados na resolução da consulta e retorno da informação pretendida podem estar intimamente ligados à utilização de um dicionário geográfico comumente chamado *gazetteer* [4].

Os desafios envolvidos na implementação de um *gazetteer*, bem como sua expansão e agregação de dados foram motivações para diversos estudos e produções de artigos que antecederam este projeto [5, 6]. Um dos resultados, a implementação de uma ferramenta chamada *Linked OntoGazetteer (LoG)* [7, 8], será base sobre a qual os processos descritos a seguir serão desenvolvidos.

O presente estudo tem como objetivo explorar as capacidades de atualização da arquitetura envolvida na persistência das informações reunidas no LoG para permitir o suporte a operações geográficas. Deste modo, as seguintes sessões reúnem de forma detalhada as informações e os processos executados em cada uma das etapas do procedimento.

2 Referencial Teórico

Várias soluções de *gazetteers* podem ser facilmente encontradas em buscas na internet: *GeoNames* [1], *Getty Thesaurus of Geographic Names Online* [2] e *Pleiades* [3], são apenas alguns exemplos de dicionários toponímicos. Além destas opções, várias outras estão disponíveis regionalmente ou de maneira privada.

Entretanto, utilizaremos o projeto *Linked OntoGazetteer* [4], desenvolvido com o objetivo de prover um dicionário de nomes geográficos com informações provenientes de diversas fontes, destacadamente *GeoNames* e *DBPedia* [5] [9]. Sua implementação é baseada na combinação de informações oriundas destas fontes em um sistema complexo de persistência em grafos expondo uma interface *WEB* para consultas textuais.

A intenção de aplicar alterações na arquitetura do ambiente de forma a enriquecer os dados persistidos com informações geoespaciais, está diretamente relacionada ao entendimento da estrutura deste sistema como um todo.

2.1 Interface *Web*

O *LoG* utiliza como ponto de entrada uma interface *web* apresentada em uma página com a possibilidade de busca textual por um termo. Os resultados são exibidos em uma lista de combinações das informações presentes no banco de dados com as características apresentadas previamente.

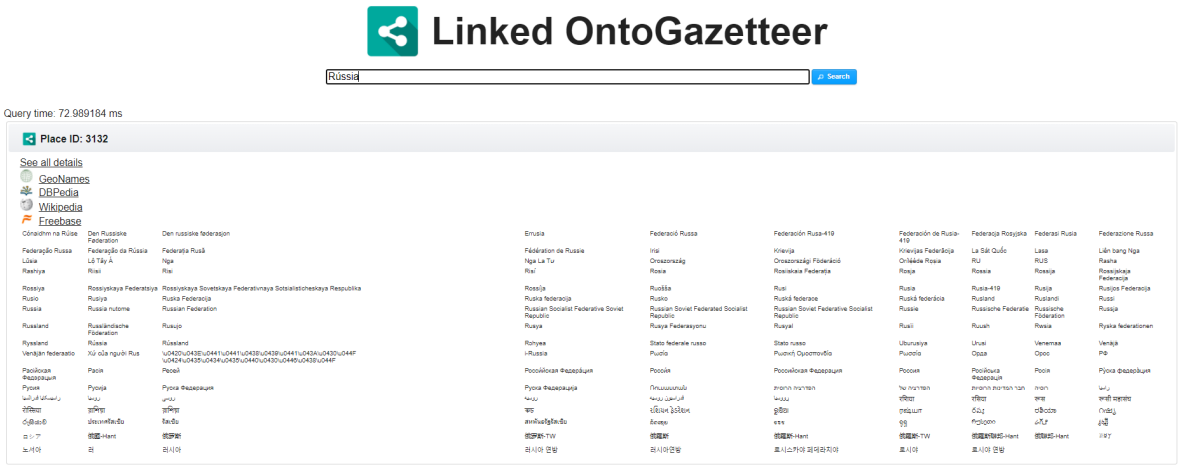

Figura 1 – Resulto de pesquisa simples no LoG

[1] http://geonames.org
[2] http://www.getty.edu/research/tools/vocabularies/tgn/
[3] https://pleiades.stoa.org/
[4] http://aqui.io/log/
[5] https://wiki.dbpedia.org/

2.2 Backend

Os dados do *LoG* são persistidos em um sistema de banco de dados do tipo *Not Only SQL (NoSQL)* [10, 11], em particular uma abordagem orientada a grafos [12]. A formalização é dada por $G(V, E)$ onde os vértices V armazenam os atributos de cada termo e as arestas E representam as relações entre os vértices. O esquema apresentado pela Figura 2 ilustra um vértice atrelado ao termo "Belo Horizonte"e suas relações.

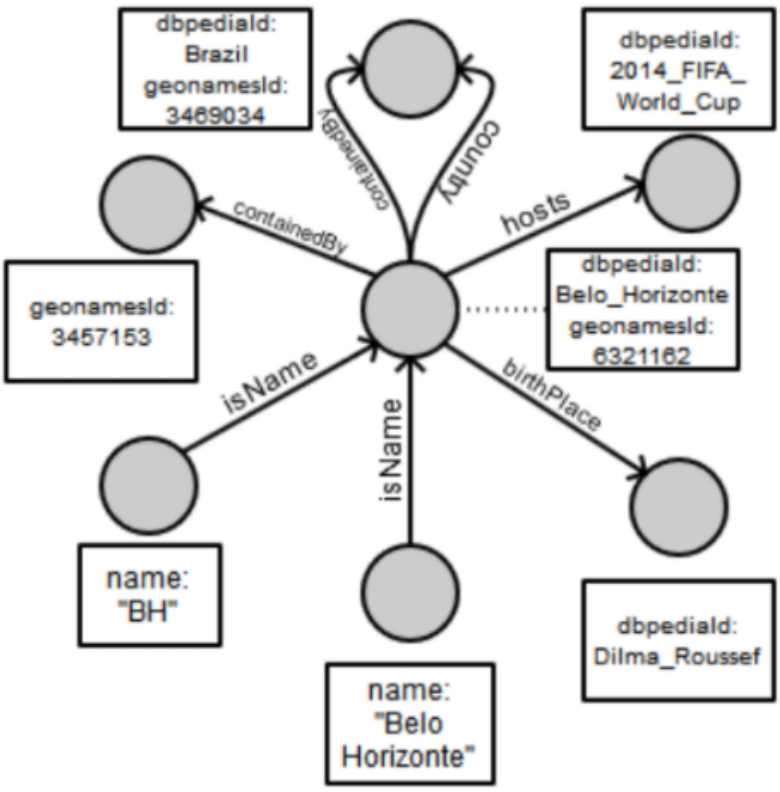

Figura 2 – Representação em grafo do termo Belo Horizonte retirado de [7]

2.3 Pesquisa

As pesquisas são realizadas através de uma travessia pelo grafo buscando por ocorrências do termo como nome de um vértice ou relação entre vértices, a resposta é o resultado da combinação de todas estas ocorrências como apresentado na Figura 1.

3 Metodologia

Uma vez elucidados os processos envolvidos na utilização do sistema, bem como uma explicação em alto nível do formato de persistência, esta sessão apresentará as etapas desenvolvidas durante a realização do projeto

3.1 Arquitetura

3.1.1 Titan

Assim como descrito previamente, o *LoG* utiliza uma ferramenta para o gerenciamento de seu banco de dado em grafos, o *Titan Distribuited Graph Database* [1] [13]. Este sistema é capaz de fornecer uma solução escalável e otimizada para persistência e consultas em grafos contendo centenas de milhares de vértices e arestas, distribuídos em diversos nós de um *cluster*. [2] As motivações para escolha da implementação de um sistema utilizando tal ferramenta se deve a capacidade de escalabilidade, robustez e relação nativa com o *Apache Hadoop*, possibilitando assim uma ampla associação com processos de análise e otimização.

A versão do *Titan* atualmente publicada em produção com *LoG* é a 0.4.4 [3] e possui a matriz de compatibilidade descrita na Figura 3

→ Features → ↓ Editions ↓	Core	Storage Backend Support				External Index Support		Rexster
		BDB JE	Cassandra	HBase	Persistit	Elasticsearch	Lucene	
Titan/All	✓	✓	✓	✓	✓	✓	✓	
Titan/BerkeleyJE	✓	✓						
Titan/Cassandra	✓		✓					
Titan/HBase	✓			✓				
Titan/Persistit	✓				✓			
Titan Server (All + Rexster)	✓	✓	✓	✓	✓	✓	✓	✓

Figura 3 – Matriz de compatibilidade do Titan versão 0.4.4 retirado de [14]

Em particular a versão 0.4.4 utilizada é a *Titan Server*, especificamente utilizando o *Cassandra* como *backend* [4] e o *Rexter* como interface para acesso aos dados.

[1] `http://titan.thinkaurelius.com/`

[2] Cluster: O termo em inglês é utilizado para denominar um conjunto de determinado tipo de objeto. Para o contexto deste artigo, cluster tem o conceito de conjunto de máquinas para processamento e armazenamento de dados

[3] `http://titan.thinkaurelius.com/wikidoc/0.4.4/`

[4] Backend: O termo em inglês faz mensão ao sistema de processamento por trás de uma aplicação. Neste

3.1.2 Apache Cassandra

O *Apache Cassandra*[15] é um banco de dados escalável, provedor de alta disponibilidade, capaz de fornecer tolerância a falhas e *clusterização*[5] nativa. Assim como o *Titan* a utilização deste banco de dados para a aplicação em questão esta diretamente ligada a necessidade de diminuir os gargalos de consultas através de centenas de milhares de dados.

Este é um banco de dados não relacional, isso significa que a organização dos dados não segue o formato tradicional de linhas e colunas e, além disso, os dados podem sofrer serializações e compactações de acordo com a necessidade do arquiteto da solução. Esta característica é fator determinante para as discussões das estratégias adotadas a seguir. Em outras palavras, não existem maneiras triviais de acessar o grafo do *LoG* diretamente através de consultas no *Cassandra*.

3.1.3 Rexster

Por definição, o *Rexster* é um servidor que provê a interface para consultas e operações em grafos através de uma *Application Programming Interface (API)* HTTP ou REST através do *framework doghouse*[6] ou *RexPro* [7]. De forma abstrata, podemos aproximar esta ferramenta como o ponto de interseção entre o *Titan* e o grafo propriamente dito.

3.1.3.1 Gremlin

As atividades descritas como responsabilidade do *Rexster* são desempenhadas através de comandos na linguagem *Groovy* [16] no *Gremlin* console ilustrado pela Figura 4 ou através da *API* disponível para diversas linguagens como Java, Python dentre outras.

3.2 Atualização

Em termos gerais, pode-se observar o sistema do *LoG* como uma interface de gerenciamento de banco de dados em grafos (*Titan*) responsável pelo gerenciamento do *cluster* da solução, recuperação de falhas e escalabilidade, interligado ao sistema de persistência, serialização e compactação de dados (*Cassandra*) através da ponte fornecida pelo *framework Rexster*.

O objetivo primário do projeto foi adaptar o *Titan* para permitir a *API ElasticSearch*[8] desemepenhar operações geográficas assim como a *PostGIS* do banco de dados *PostgreSQL*

contexto o termo se associa especificamente ao processo de persistência de gerenciamento dos dados.

[5] Clusterização: Distribuir a aplicação em clusters

[6] `https://github.com/tinkerpop/rexster/wiki/The-Dog-House`

[7] `https://github.com/tinkerpop/rexster/wiki/RexPro`

[8] `https://www.elastic.co/pt/`

```
gremlin> :help

For information about Groovy, visit:
    http://groovy-lang.org

Available commands:
  :help       (:h  ) Display this help message
  ?           (:?  ) Alias to: :help
  :exit       (:x  ) Exit the shell
  :quit       (:q  ) Alias to: :exit
  import      (:i  ) Import a class into the namespace
  :display    (:d  ) Display the current buffer
  :clear      (:c  ) Clear the buffer and reset the prompt counter
  :show       (:S  ) Show variables, classes or imports
  :inspect    (:n  ) Inspect a variable or the last result with the GUI object b
rowser
  :purge      (:p  ) Purge variables, classes, imports or preferences
  :edit       (:e  ) Edit the current buffer
  :load       (:l  ) Load a file or URL into the buffer
  .           (:.  ) Alias to: :load
  :save       (:s  ) Save the current buffer to a file
  :record     (:r  ) Record the current session to a file
  :history    (:H  ) Display, manage and recall edit-line history
  :alias      (:a  ) Create an alias
  :register   (:rc ) Register a new command with the shell
  :doc        (:D  ) Open a browser window displaying the doc for the argument
  :set        (:=  ) Set (or list) preferences
  :uninstall  (:-  ) Uninstall a Maven library and its dependencies from the Gre
mlin Console
  :install    (:+  ) Install a Maven library and its dependencies into the Greml
in Console
  :plugin     (:pin) Manage plugins for the Console
  :remote     (:rem) Define a remote connection
  :submit     (:>  ) Send a Gremlin script to Gremlin Server

For help on a specific command type:
    :help command

gremlin>
```

Figura 4 – Painel de ajuda de comandos no Gremlin console

[17], tarefa possível de ser executada através da atualização para a versão mais recente publicada 1.0.0[9].

3.2.1 Formato GraphhML

O primeiro obstáculo envolveu a migração das ferramentas *Rexster* e todo seu ecossistema para a *Apache* a partir da versão 3.X. Desta união, as ferramentas sofreram alterações profundas em seu comportamento e passaram a fazer parte da solução *Apache TinkerPop*[10]

[9] `http://s3.thinkaurelius.com/docs/titan/1.0.0/`
[10] `https://tinkerpop.apache.org/`

e sua organização pode ser compreendida através da Figura 5.

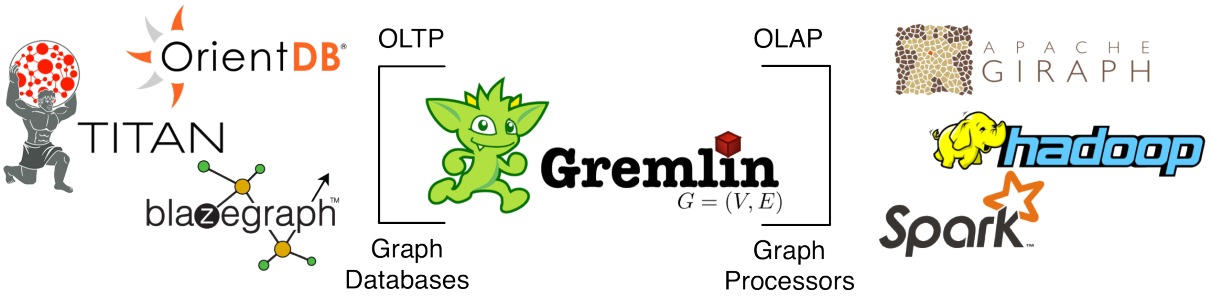

Figura 5 – Apresentação esquemática da arquitetura do TinkerPop retirada de `https://tinkerpop.apache.org/gremlin.html`

Através da leitura detalhada do manual de atualização de versões[11], ficou claro que a simples atualização de cada ferramenta utilizada na estrutura geral do *LoG* não seria suficiente para garantir o comportamento adequado do sistema, bem como a integridade do grafo.

As versões anteriores do *TinkerPop* permitiam um formato de grafos chamada *GraphML* [18] não mais suportado. Portanto, uma migração exige a transformação do grafo em um formato *JSON* como passo intermediário que, em um momento posterior será importado na versão 3.X.

3.2.2 Gremlin e Groovy Language

A exportação do grafo em formato *JSON* pode ser executada trivialmente através de um simples comando na linguagem *Groovy*. Entrentanto o ambiente para executar esse comando deve ser previamente preparado para se conectar ao grafo no *backend*.

Diferentemente do *Gremlin console* presente nas versões atuais, onde é possível adicionar as conexões remotas através de comandos, na versão em produção, a conexão com o banco *Cassandra* deve ser feita através de parâmetros definidos por uma arquivo de configuração. Este arquivo é descrito dentro no diretório de instalação para então ser importado ou definido na execução do *console*.

Essa necessidade de importação/definição de configurações prévias se faz necessária mesmo ao acessar o console através da interface *WEB* fornecida pelo *doghouse* ou pela *API* através do *RexPro*. Vale ressaltar que a conexão via navegador fora do servidor que roda a aplicação exige a liberação das portas 8182 e 8183 além da utilização de um túnel de conexões.

[11] `https://tinkerpop.apache.org/docs/3.4.10/upgrade/#_tinkerpop_2_x`

3.2.3 Apache Thrift

Uma vez vencidos os obstáculos de mapeamento das ferramentas envolvidas na arquitetura do sistema e migrações necessárias, bem como configurações dos ambientes para emissão de comandos e criação correta das propriedades de conexão, o processo de migração deveria estar preparado para desempenhar o processo de exportação, porém, mais um obstáculo se mostrou presente.

A *API Apache Thrift* tem como função criar um denominador comum para diversas linguagens de programação a fim de produzir uma interface para comunicação. No processo interno de exportação do grafo pelo *Gremlin*, a serialização utiliza desta *API* para produção do resultado em seu novo formato. Entretanto, a configuração das propriedades deve ser feita manualmente para atender a demanda do processo por todo o grafo.

Durante a execução deste projeto não foram encontrados os valores para as propriedades desta ferramenta que possibilitassem essa migração com sucesso. Existe ainda, a suposição de que, apesar de todas as ferramentas da arquitetura permitirem o processamento distribuído, atualmente o *LoG* funciona com apenas um nó de execução e os valores das propriedades envolvidas não seriam capazes, mesmo em sua atribuição máxima, de completar o processo em um único nó.

4 Conclusão

O desenvolvimento deste projeto proporcionou o contanto com todas as esferas que compõem um sistema complexo, desde replicação de ambiente local e pequenos ajustes de código para implantação da parte web, a configurações específicas de sistema operacional e de bancos de dados residentes em um servidor externo e comunicável através de *VPN*.

Os conhecimentos adquiridos durante o percurso permitiram a maturação de um olhar crítico para escolha das ferramentas baseados nos *tradeoffs* de custo computacional *vs* benefícios ou possibilidade de integração *vs* dificuldade de configuração.

O opção de escolha por um banco *NoSQL* que em uma solução tal qual a abordada, se justifica pela eliminação simples de redundância de dados, forma simplificada de detecção de grupos de relações além de todos os benefícios relacionados à característica de ser não estruturado, permitiu uma visão aproximada de todos os desafios relacionados a produção de backup e manutenção de consistência.

O aprofundamento em questões muito específicas das ferramentas *Apache* permitiu uma aproximação com a comunidade desenvolvedora do código aberto e proporcionou a capacidade de, em eventuais posteriores, propor melhorias e alterações.

Por fim, para os trabalhos futuros, o presente estudo deixa como legado as formas de contornar os possíveis problemas em todas as etapas de implantação de um sistema que tem grande potencial no campo de processamento de dados geográficos. Além disso destaca a visão de que apesar de performar adequadamente para o sistema em questão, a necessidade de escalar horizontalmente o *cluster* pode ser um fator limitante para a evolução da ferramenta e requer estudos de viabilidade.

Referências

[1] Tor Bernhardsen. *Geographic information systems: an introduction*. John Wiley & Sons, 2002.

[2] Gilberto Câmara, Marco A Casanova, and Geovane C Magalhães. Anatomia de sistemas de informação geográfica. 1996.

[3] Ross S Purves, Paul Clough, Christopher B Jones, Mark H Hall, and Vanessa Murdock. Geographic information retrieval: progress and challenges in spatial search of text. *Foundations and Trends in Information Retrieval*, 12(2-3):164–318, 2018.

[4] Michael F Goodchild and Linda L Hill. Introduction to digital gazetteer research. *International Journal of Geographical Information Science*, 22(10):1039–1044, 2008.

[5] Carsten Keßler, Krzysztof Janowicz, and Mohamed Bishr. An agenda for the next generation gazetteer: Geographic information contribution and retrieval. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in Geographic Information Systems*, pages 91–100, 2009.

[6] Ivre Marjorie Ribeiro Machado. Um gazetteer ontológico para recuperação de informação geográfica. 2011.

[7] Tiago HVM Moura and Clodoveu A Davis Jr. Integration of linked data sources for gazetteer expansion. In *Proceedings of the 8th Workshop on Geographic Information Retrieval*, pages 1–8, 2014.

[8] Tiago HVM Moura, Clodoveu A Davis Jr, and Frederico T Fonseca. Reference data enhancement for geographic information retrieval using linked data. *Transactions in GIS*, 21(4):683–700, 2017.

[9] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Journal of web semantics*, 7(3):154–165, 2009.

[10] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. Nosql databases. *Lecture Notes, Stuttgart Media University*, 20:24, 2011.

[11] Jing Han, Ee Haihong, Guan Le, and Jian Du. Survey on nosql database. In *2011 6th international conference on pervasive computing and applications*, pages 363–366. IEEE, 2011.

[12] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

[13] Chialin Chang, Bongki Moon, Anurag Acharya, Carter Shock, Alan Sussman, and Joel Saltz. Titan: a high-performance remote-sensing database. In *Proceedings 13th International Conference on Data Engineering*, pages 375–384. IEEE, 1997.

[14] Thinkaurelius. thinkaurelius/titan.

[15] Nishant Neeraj. *Mastering Apache Cassandra*. Packt Publishing Ltd, 2013.

[16] Kenneth Barclay and John Savage. *Groovy programming: an introduction for Java developers*. Elsevier, 2010.

[17] Regina Obe and Leo Hsu. Postgis in action. *GEOInformatics*, 14(8):30, 2011.

[18] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. *Graph markup language (GraphML)*. 2013.

[19] Felix Mata. Geographic information retrieval by topological, geographical, and conceptual matching. In *International Conference on GeoSpatial Sematics*, pages 98–113. Springer, 2007.

[20] Christopher B Jones, Harith Alani, and Douglas Tudhope. Geographical information retrieval with ontologies of place. In *International Conference on Spatial Information Theory*, pages 322–335. Springer, 2001.