

Uso de Expressões Temporais em Busca na Web: Uma análise através das sugestões de consulta

Augusto B. Corrêa¹, Edimar Manica^{1,2}, Renata Galante¹, Carina F. Dorneles³

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

²Campus Avançado Ibirubá – Instituto Federal do Rio Grande do Sul (IFRS)
Rua Nelsi Ribas Fritsch, no 1111, Bairro Esperança – 98200-000 – Ibirubá – RS – Brasil
³Depto de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

Caixa Postal 476 – CEP 88049-900 – Florianópolis – SC – Brasil

{abcorrea, emanica, galante}@inf.ufrgs.br, dorneles@inf.ufsc.br

Abstract. This paper describes a study about Web Query Logs that identifies the way that users express temporal information in Web queries. The Web queries with temporal expressions have been analyzed on the following items: distribution in topics; mean size; most frequent queries; and, most frequent terms. To reach this goal, we have implemented a tool written in Python to get the query log from query suggestions.

Resumo. Este artigo descreve um estudo que analisa logs de consultas a fim de identificar a forma como os usuários expressam informações temporais em consultas na Web. As consultas com expressões temporais foram analisadas sobre os seguintes aspectos: distribuição em tópicos; tamanho médio, consultas mais frequentes e termos mais frequentes. Para atingir esse objetivo, foi implementada uma ferramenta em Python para coletar o log das consultas a partir de sugestões de consulta.

1 Introdução

Os motores de busca mantém arquivos de *log* onde armazenam informações sobre a interação dos usuários. São armazenadas informações sobre as atividades de navegação (*clicks*) e de busca. Esses arquivos de *log* são úteis tanto para entender a estratégia de busca dos usuários, quanto para melhorar as sugestões de consulta [Wen e Zhang 2003] e a qualidade dos resultados retornados [Joachims 2002]. A análise de *logs* com foco na atividade de busca é conhecida como Análise de Logs de Busca (*Search Log Analysis*) [Trevisam et al. 2012].

Páginas web descrevem vários tópicos, tais como conferências, esportes, política e entretenimento. A maioria destes eventos mudam ao longo do tempo. A Escola Regional de Banco de Dados, por exemplo, ocorre todo ano. As olimpíadas ocorrem a cada quatro anos. A comunidade de banco de dados tem dedicado um esforço significativo para permitir a indexação e a consulta a dados temporais nas décadas passadas [Weikum 2011, Li et al. 2010]. Atualmente, o uso de expressões temporais tem emergido em consultas Web uma vez que documentos Web também possuem informações temporais [Manica et al. 2012]. Com isso, saber como o usuário expressa

sua necessidade de informação temporal é essencial para melhorar a qualidade dos resultados deste tipo de busca. Além disso, quando um usuário faz uma busca por uma pessoa famosa, os motores de busca exibem algumas informações dessa pessoa como foto, nome, nascimento, etc. Esse mecanismo não está disponível quando a busca é por uma expressão temporal. Quais informações deveriam ser exibidas neste tipo de busca? Logs de consultas com expressões temporais representam um recurso a ser analisado para responder essa pergunta.

A análise de *logs* tem sido um pouco limitada devido a falta de dados de usuários reais e a existência de questões éticas importantes [Bar-Ilan 2007]. Yoshinaga e Torisawa (2007) afirmam que é difícil para pessoas que não trabalham em empresas que possuem um motor de busca comercial obterem acesso a um grande conjunto de *logs* de consultas reais. Com isso, o presente trabalho pretende formar um *log* de consultas a partir das sugestões que um motor de busca comercial fornece quando submetida uma expressão temporal como consulta. Essa estratégia parte da observação que as sugestões fornecidas pelos motores de busca para uma consulta com uma expressão temporal são as consultas com aquela expressão temporal que mais ocorrem no *log* de consultas do próprio motor de busca. Por exemplo, ao submeter a expressão temporal “17 de fevereiro” como consulta ao motor de busca Bing¹, ele retorna as sugestões “17 de fevereiro signo”, “17 de fevereiro wikipedia” e “17 de fevereiro dia mundial do gato”. Isso significa que essas três sugestões são as consultas com a expressão temporal “17 de fevereiro” que mais ocorrem naquele motor de busca.

Este artigo descreve um estudo que analisa *logs* de consultas a fim de identificar a forma como os usuários expressam informações temporais em consultas na Web. O idioma adotado foi o português e o motor de busca utilizado foi o Bing. As principais análises realizadas sobre consultas com expressões temporais foram: distribuição em tópicos, tamanho médio, consultas mais frequentes e termos mais frequentes. Para atingir esse objetivo, foi implementada uma ferramenta em Python para coletar o *log* das consultas a partir de sugestões de consulta.

Este artigo está organizado como segue. A Seção 2 apresenta os principais conceitos temporais envolvidos no artigo. Na Seção 3 são apresentadas uma visão geral do trabalho desenvolvido e a configuração dos experimentos. Na Seção 4 é apresentada a análise dos dados coletados. A Seção 5 discute os principais trabalhos relacionados. Finalmente, na Seção 6, são apresentadas as considerações finais e as direções futuras.

1 <http://br.bing.com/>

2 Conceitos Básicos

Esta seção descreve os principais conceitos temporais necessários para a compreensão deste trabalho [Alonso et. Al 2007]:

- **entidade temporal** – é a descrição em um nível conceitual de um ponto no tempo, um evento ou um período de tempo;
- **expressão temporal** – é uma sequência de *tokens* que representa uma instância de uma entidade temporal;
- **expressão temporal explícita** – são expressões que descrevem diretamente uma entrada em uma linha de tempo, tal como uma data exata ou ano específico. Por exemplo, a expressão “dezembro de 2013” ou “12 de janeiro de 2014” em um fragmento de texto são expressões temporais explícitas e podem ser mapeadas diretamente para pontos em uma linha de tempo;
- **expressão temporal implícita** – são expressões que precisam de conhecimento predefinido (ontologias de tempo, por exemplo) para serem mapeadas para uma entrada em uma linha de tempo. Nomes de feriados e eventos específicos são típicos exemplos de expressões temporais implícitas. Por exemplo, a expressão “Natal de 2013” precisa ser mapeada para “25 de dezembro de 2013”;
- **expressão temporal relativa** – são expressões temporais que representam entidades temporais que apenas podem ser mapeadas para uma entrada em uma linha de tempo em referência a uma expressão temporal explícita, implícita ou ainda ao momento em que o texto foi escrito. Por exemplo, a expressão “ontem” só pode ser mapeada com base no momento em que o texto foi escrito.

Manica, Dorneles e Galante (2012) classificam as consultas Web com informações temporais em dois tipos.

- **seleção temporal (*temporal selection*)** – consultas nas quais utiliza-se um predicado temporal para filtrar a consulta;
- **saída temporal (*temporal output*)** – consultas onde o usuário está interessado em saber qual o tempo em que um determinado evento ocorreu.

O presente trabalho restringe a análise a consultas Web de seleção temporal contendo expressões temporais explícitas, uma vez que este tipo produz maior número de consultas e logo, maior número de sugestões. Porém, como trabalho futuro se pretende trabalhar com os demais tipos de expressões temporais

3 Ferramenta e Configuração Experimental

Esta seção descreve como o *log* de consultas foi construído a partir das sugestões de consulta fornecidas por um motor de busca e as características do *log* de consultas construído. A Figura 1 apresenta a visão geral da ferramenta desenvolvida em Python utilizada para a construção do *log* de consultas. O usuário deve escolher entre um dos formatos de expressão temporal disponíveis pela ferramenta (por exemplo, “DD de MM de AA”) e definir um valor inicial e um valor final. A ferramenta gera o conjunto de expressões temporais entre o valor inicial e o valor final de acordo com o formato definido. Cada expressão temporal gerada é submetida ao motor de busca Bing como uma consulta. O Bing, através da API (*Application Programming Interface*) Qsonhs², retorna as sugestões para a consulta submetida em um arquivo JSON (*JavaScript Object Notation*). Essas sugestões são extraídas através da biblioteca JSON Decoder³, e enviadas para o *log* de consultas.

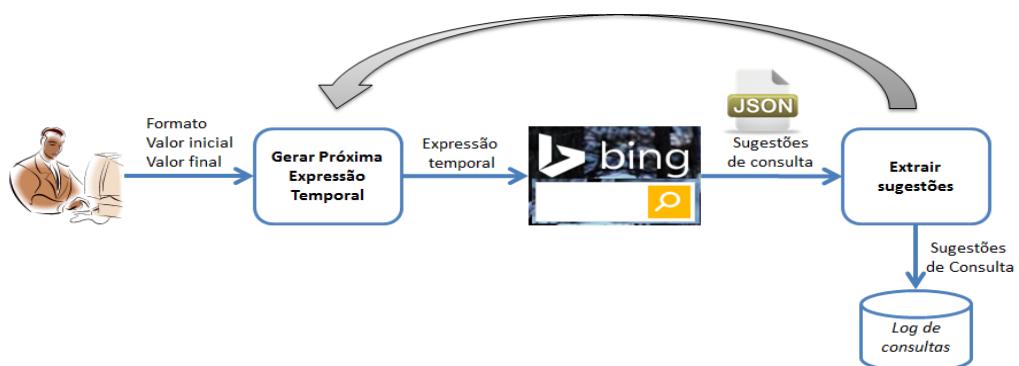


Figura 1: Visão Geral da Construção do *log* de consultas.

A Tabela 1 apresenta os formatos, valores iniciais e valores finais submetidos, sendo DD o dia em algarismos variando de 01 a 31, MM os meses escritos por extenso, AA o ano em algorismos variando de 1500 a 2100, mm o mês representado por algarismos variando de 01 a 12, KK representando a década variando entre 00 e 90 (apenas com números de final 0) e sendo CC o século em algarismos variando de 0 a 21. Neste trabalho foram submetidos apenas formatos de expressões temporais explícitas para consultas de seleção temporal, pois estas forneciam mais consultas e consequentemente um número maior de sugestões para a análise.

² Qsonhs: é uma API do Bing para obter as sugestões para uma dada consulta. Basta executar a URL <http://api.bing.com/qsonhs.aspx?FORM=ASAPIH&q=CONSULTA>, substituindo o valor do parâmetro *q* por uma consulta. As sugestões são retornadas em um arquivo JSON.

³ <http://docs.python.org/2/library/json.html>

Tabela 1: Formatos, valores iniciais e valores finais submetidos.

Formato	Valor Inicial	Valor Final
DD de MM	01 de janeiro	31 de dezembro
DD de MM de AA	01 de janeiro de 1500	31 de dezembro de 2100
DD/mm	01/01	31/12
MM de AA	janeiro de 1500	dezembro de 2100
Século CC	Século 01	Século 21
Década de KK	Década de 10	Década de 90

A Figura 2 apresenta um exemplo de arquivo JSON retornado pela API Qsonhs para a consulta “20 de setembro”. Como pode ser observado, foram retornadas 8 sugestões, sendo a primeira a própria consulta e a última “20 de setembro feriado rs”. Não foi possível estabelecer um limite fixo de pesquisas diárias. Entretanto, notou-se um comportamento inusitado na API: a medida que eram realizadas mais pesquisas, o número de sugestões retornadas pela API eram menores que o número de sugestões fornecidas pelo Bing em consultas manuais. Para a correção de tal equívoco, utilizou-se um *sleep* – tempo em que o sistema ficou ocioso sem a utilização da API – de 120 segundos a cada 3600 pesquisas. Com este dispositivo, anulou-se a margem de distorção entre as pesquisas realizadas manualmente e as pesquisas realizadas através da API.

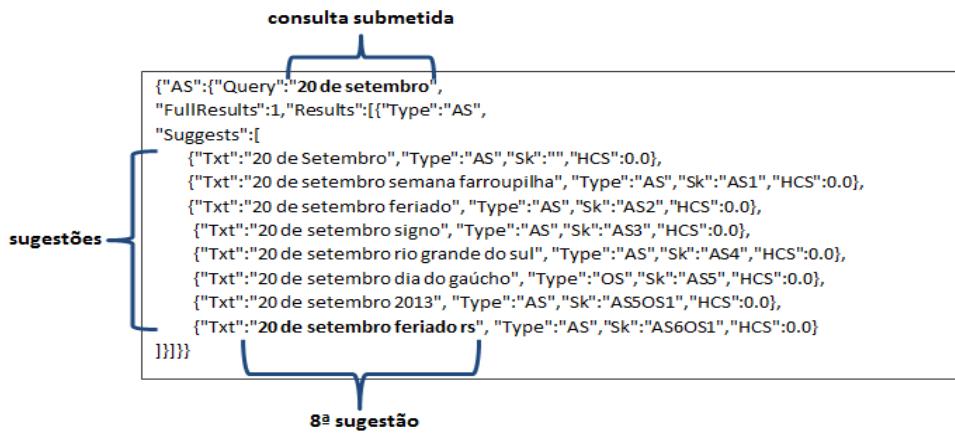


Figura 2: Exemplo de arquivo JSON retornado pela API Qsonhs

O *log* de consultas criado é um arquivo CSV (*comma-separated values*) contendo as seguintes colunas: (i) Pesquisa – expressão temporal que foi submetida na API Qsonhs; (ii) Sugestão – sugestão de consulta que foi retornada na API Qsonhs; (iii) Data – data e horário em que a pesquisa foi realizada; (iv) Formato - formato (Tabela 1) em que a consulta do termo Pesquisa foi realizada.

O *log* de consultas construído possui 529 Kbytes. A Tabela 2 apresenta o número de sugestões obtidas para cada formato.

Tabela 2: número de sugestões obtidas para cada formato.

Formato	Número de Sugestões
DD de MM	2425
DD de MM de AA	1127
DD/mm	2516
MM de AA	68
Século CC	42
Década de KK	62

4 Análise de dados

Esta seção descreve as quatro análises realizadas sobre o *log* de consultas com expressões temporais que foi construído conforme detalhado na seção anterior: *(i)* distribuição das consultas em tópicos; *(ii)* tamanho das consultas; *(iii)* termos mais frequentes; *(iv)* consultas mais frequentes. Essas análises foram realizadas sobre as sugestões de consulta armazenadas no *log*, excluindo os termos da consulta submetida. Por exemplo, para a sugestão “15 de novembro proclamação da república” obtida a partir da consulta “15 de novembro” apenas o fragmento “proclamação da república” é considerado nas análises. Esse fragmento é referenciado nessa seção como consulta.

4.1 Distribuição em Tópicos

O objetivo desta análise é verificar os tópicos que os usuários tem mais necessidade de informação temporal. As consultas foram classificadas em: entretenimento, datas comemorativas, notícias e sociedade, organizações, lugares, pesquisas, esportes e outros.

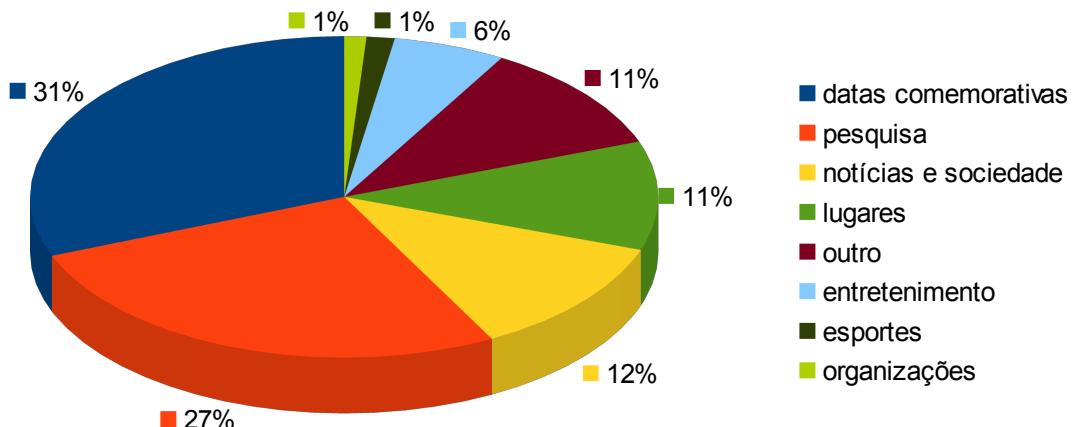


Figura 3: Classificação das consultas em tópicos

A Figura 3 ilustra os resultados obtidos (em ordem decrescente de frequência): datas comemorativas (31%), pesquisa (27%), notícias e sociedade (12%), lugares (11%), outro (11%), entretenimento (6%), esportes (1%) e organizações (1%). Dentro da categoria *outro* foram destacadas algumas subcategorias como, por exemplo: erro de digitação, finanças, compras, URL, computação e veículos, mas que não obtiveram frequência considerável para ser relevante no cenário da pesquisa.

4.2 Tamanho das consultas

O objetivo desta análise é identificar o tamanho das consultas com expressões temporais. Conforme ilustra a Figura 4, é possível observar que quase metade (47,3%) das consultas são compostas por dois termos. Em seguida, com um percentual bastante próximo, seguem as consultas com três termos (22,8%) e com um termo (18,4%). Por fim, houve um percentual bem pequeno de consultas com cinco termos (6,1%) e com quatro termos (4,3%). Nota-se também a presença de consultas de tamanho 6 (maiores consultas encontradas), embora não tenham atingido 1,0%.

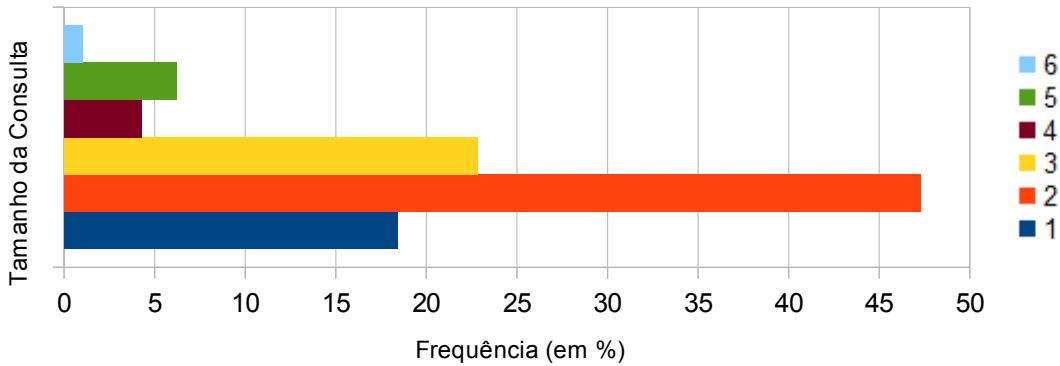


Figura 4: Tamanho da Consulta

4.3 Termos mais frequentes

O objetivo desta análise é identificar os termos que mais ocorrem em consultas com expressões temporais. Como pode ser observado na Tabela 3, os termos mais frequentes no *log* de consultas são: *dia*, *feriado*, *signo*, *nacional* e *brasil*. Nota-se também uma grande frequência de preposições (principalmente *de*, *do*, *e*, *da*), mas que não foram levadas em consideração uma vez que estas são *stopwords*, palavras frequentemente utilizadas mas com pouco significado isoladamente.

Tabela 3: Termos mais frequentes e suas de ocorrências no log de consultas.

Termo	Ocorrências	Termo	Ocorrências
Dia	629	Mundial	37
Feriado	409	Mundo	28
Signo	365	Carter	28
Nacional	45	Fim	25
Brasil	39	Vasco	25

4.4 Consultas mais frequentes

O objetivo desta análise é identificar as consultas que mais ocorrem com expressões temporais. As cinco consultas mais frequentes foram: *signo*, *feriado*, *é feriado*, *dia / dia de / dia do e qual signo*. Essas consultas representam as principais informações sobre uma expressão temporal que os usuários buscam na Web.

Tabela 4: Consultas mais frequentes e seu número de ocorrências

Consulta	Número de ocorrências
Signo	301
Feriado	172
É feriado	105
Dia / Dia de / Dia do	91
Qual signo	31

5 Trabalhos Relacionados

Nunes, Ribeiro e David (2006) analisaram o uso de expressões temporais em motores de busca e concluíram que as mesmas constituem uma fração muito pequena das pesquisas realizadas, porém dada a escalabilidade da Web representam um número considerável de usuários. Nunes, Ribeiro e David também se depararam com a análise de que a maioria das pesquisas que contém expressões temporais referenciam datas atuais ou passadas. Destoando de nossa pesquisa, os tópicos mais encontrados foram carros, esportes, notícias e sociedade, feriados, enquanto no presente trabalho foram datas comemorativas, pesquisa, notícias e sociedade e lugares, sendo que a categoria carros não atingiu 1%. Ressalta-se ainda que a análise realizada por Nunes, Ribeiro e David usou como base pesquisas de língua inglesa no motor de busca AOL.

Kato, Sakai e Tanaka (2013) analisaram as sugestões de consultas em mecanismos de busca Web, analisando três tipos de conjuntos de dados extraídos do

Bing: (1) *log* de consultas, (2) *log* de sugestões, (3) *log* de ações (*clicks*) dos usuários na página. De acordo com suas análises, foi possível elaborar os usos mais frequentes das sugestões de consulta pelo usuário: (1) quando a consulta original não é uma consulta frequente; (2) quando a consulta original contém só 1 termo; (3) quando não há ambiguidade na sugestão de consulta; (4) quando a sugestão é uma generalização ou uma correção da consulta original; (5) após o usuário clicar em diversos *links* na primeira página da consulta. Entretanto, não foram analisadas apenas pesquisas com expressões temporais, mas pesquisas de uma maneira geral. Ao contrário do presente trabalho, Kato, Sakai e Tanaka não efetuaram análises individuais sobre as sugestões de pesquisa – como classificar em tópico e tamanho – e apenas se limitaram a análise das sugestões de pesquisa como um todo.

Beltzel et al. (2007) realizaram uma análise temporal em um *log* de consultas da *American's Online* (AOL) dividido em tópicos – computação, música, carros, filmes, jogos, finanças pessoais, pornografia, saúde, entretenimento, viagens, websites americanos, casa e jardim, governo, esportes, compras e feriado – tentando investigar as mudanças nos padrões de consultas realizadas pelos usuários. Algumas categorias formaram certos padrões em sua frequência – seja ela anual, mensal, semanal ou diária. Por exemplo, as consultas classificadas na categoria *filmes* sofrem um acréscimo entre outubro e fevereiro, enquanto as pesquisas classificadas como *feriado* tendem a diminuir no meio da semana. Com o trabalho de Beltzel et al. torna-se possível analisar o comportamento do usuário de forma mais dinâmica e completa através das consultas dos usuários. Enquanto Beltzel et al. analisam o tempo em que a consulta foi submetida, o presente trabalho analisa a informação temporal que o usuário adicionou na consulta.

6 Conclusões

Com o aumento de dados temporais na Web, aumenta também a necessidade dos usuários de consultar tais dados. A fim de melhorar os resultados para esse tipo de consulta, é necessário saber como os usuários as expressam. Como os dados de *logs* de consultas não são disponibilizados pelos motores de busca, este trabalho criou um *log* de consultas através das sugestões de consultas fornecidas pelos motores de busca usando expressões temporais explícitas como consulta. Analisando este *log*, percebe-se que mais da metade das consultas são relacionadas a datas comemorativas e pesquisas na internet, além de que a maioria delas são curtas, tendo no máximo 3 termos em sua composição. Também, a análise de termos e consultas mais frequentes revela que ao

realizar uma busca por uma expressão temporal, o motor de busca poderia exibir o signo de quem nasce na data representada pela expressão temporal e se ela representa um feriado no contexto do usuário. Como trabalhos futuros se pretende expandir a análise para expressões temporais implícitas e relativas.

7 Referências

- Alonso, O.; Gertz, M.; Baeza-Yates, R. A. (2007). "On the value of temporal information in information retrieval". In: SIGIR Forum, 41(2):35–41
- Bar-Ilan, J. (2007). "Access to query logs - an academic researcher's point of view". In: Query Log Analysis: Social And Technological Challenges Workshop. 16th International World Wide Web Conference (WWW 2007)
- Beitzel, S.; Jensen, E.; Chowdhury, A.; Frieder, O.; Grossman, D. (2007) . In "Journal of the American Society for Information Science and Technology", Volume 58 Issue 2, January 2007. Pages 166-178
- Joachims, T.; (2002), "Optimizing Search Engines Using Clickthrough Data". In: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.
- Kato, M. P.; Sakai, T.; Tanaka, K. (2013). In "Information Retrieval", Volume 16 Issue 6, December 2013. Pages 725-746
- Li, F.; Yi, K.; Le, W.; (2010). In: Top- queries on temporal data. VLDB J., 19(5):715–733.
- Manica, E.; Dorneles, C.; Galante, R.; (2012) "Handling temporal information in web search engines" SIGMOD Record (SIGMOD) 41(3):15-23 (2012)
- Metzler, D.; Jones, R.; Peng, F.; Zhang, R. (2009). In "Improving search relevance for implicitly temporal queries". In *SIGIR '09 Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 700-701
- Nunes, S.; Ribeiro, C.; David, G. (2008). In "Use of temporal Expressions in Web Search". ECIR 2008:580-584
- Oliveira, J. P.; Edelweiss, N. (1994). "Modelagem de Aspectos Temporais de Sistemas de Informação". IX Escola de Computação, Recife.
- Trevisan, M.; Barbu, E.; Barsanti, I.; Dini, L.; Lagos, N.; Segond, F.; Rhulmann, M.; Vald, E.; (2012) "Query log analysis with LangLog". In: EACL 2012:87-91
- Weikum, G.; (2011). "Longitudinal analytics on web archive data: Its about time!". In: 5th Biennial Conference on Innovative Data Systems Research (CIDR2011), 2011. Wen, J; Zhang, H. J.; (2003). "Query Clustering in the Web Context". In: Clustering and Information Retrieval, Kluwer
- Yoshinaga, N.; Torisawa, K. (2007). Open-Domain Attribute-Value Acquisition from Semi- Structured Texts. Workshop on Ontolex-07, [S.l.].

Um Estudo sobre Bancos de Dados em Grafos Nativos

Raqueline R. M. Penteado, Rebeca Schroeder, Diego Hoss,
Jaqueline Nande, Ricardo M. Maeda, Walmir O. Couto, Carmem S. Hara

¹Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

{rrmpenteado, rebecas, jfbn10, djhoss, rmmaeda, wocouto, carmem}@inf.ufpr.br

Abstract. *Graph databases support applications based on complex data models where the interconnectivity of data is an important aspect. The ever-increasing amount of data made available by these systems has been handled by graph databases in order to scale such systems. In spite of the straightforward way to represent complex data, native graph databases provide efficient query processing by avoiding costly joins. This paper presents a comparative analysis among native graph databases by considering some important features for data management system in this context.*

Resumo. *Sistemas de banco de dados em grafos suportam aplicações baseadas em modelos de dados cuja a interconectividade de dados é um aspecto importante. O volume de dados crescente envolvendo tais aplicações vem sendo tratado por uma série de soluções de gerenciamento de dados baseados em grafos visando a escalabilidade destes sistemas. Além de prover uma forma direta para a representação de dados complexos, bancos de dados em grafos classificados como nativos são capazes de executar consultas de forma eficiente por eliminar operações custosas de junções. Este artigo apresenta uma análise comparativa entre alguns sistemas nativos atuais considerando algumas características importantes para sistemas de gerenciamento neste contexto.*

1. Introdução

Durante décadas os Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDRs) dominaram o meio empresarial e acadêmico por utilizarem um modelo de dados intuitivo, uma linguagem padronizada de consulta e manipulação de dados, e garantirem as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) em aplicações diversas. Apesar de todos os benefícios em se utilizar um SGBDR, aplicações baseadas em modelos de dados complexos podem arcar com problemas decorrentes desta adaptação ao modelo relacional, especialmente para bancos de dados que devem dar suporte a um grande volume de dados, de requisições e de usuários concorrentes.

O banco de dados em grafos surgiu como uma alternativa ao banco de dados relacional para dar suporte a sistemas cuja interconectividade de dados é um aspecto importante. Após seu surgimento nos anos 80, os bancos de dados em grafos rapidamente perderam espaço para os bancos de dados semi-estruturados em virtude da ascensão do modelo XML[Angles and Gutierrez 2008]. No entanto, o interesse pelo modelo de banco de dados em grafos ressurgiu juntamente com a web semântica e a disseminação das redes sociais[Liptchinsky et al. 2013]. A evidência recente deste modelo pode ser atestada pelo volume crescente das diversas bases de dados presentes no projeto *Linked Data*¹, assim

¹<http://linkeddata.org/>

como no projeto *DBpedia*² e pelo conjunto de repositórios *Large Triple Stores*³.

Um exemplo clássico do uso de banco de dados em grafos é o *Twitter*⁴, uma rede social que oferece um serviço de microblog. Sua base é modelada diretamente como um grafo, onde os usuários são os vértices e os relacionamentos entre eles são as arestas. Outros exemplos de contextos são: sistemas que recomendam compras em lojas virtuais, sistemas que exploram dados químicos e biológicos para detecção de padrões, e sistemas web como o *PageRank* [Brin and Page 1998] da *Google* que analisa a importância dos sites computando o número de arestas incidentes em cada site.

Os sistemas de banco de dados em grafos modelam seus dados por meio de vértices e arestas facilitando a modelagem de contextos complexos e definindo naturalmente relações existentes entre as entidades de uma base. Nesta categoria os sistemas podem ser classificados como nativos ou não-nativos. Os nativos consideram a estrutura de grafo tanto no armazenamento físico dos dados quanto no processamento de consultas. Listas de adjacências vértice-vértice são usadas no armazenamento físico possibilitando a exploração do grafo de forma simples, direta e eficiente no processamento de consultas. Em contrapartida, os sistemas não-nativos usam outros modelos para o armazenamento e processamento de consultas. Por exemplo, por meio do modelo relacional, as relações de triplas vértice-aresta-vértice em um grafo são armazenadas como tuplas em tabelas. Este tipo de composição é prejudicial para o desempenho de consultas quando diversas junções são necessárias para executar uma consulta complexa envolvendo diversas triplas.

Um projeto que tem recebido destaque no âmbito dos Sistemas Gerenciadores de Bancos de Dados em Grafos (SGBDGs) nativos é o *Tinkerpop*⁵. Este projeto foi criado em 2008 e tem o foco em estruturas de dados, consultas e linguagens de programação para grafos. Nele foi desenvolvida a *Blueprints*, uma interface básica para grafos de propriedades, que define uma série de métodos para interação com um grafo. Além disso, o *Tinkerpop* também oferece ferramentas para interagir com os SGBDGs que dão suporte à *Blueprints*, como o servidor de grafos *Rexster* que pode ser usado por um cliente para acessar uma base remotamente e a linguagem de consultas *Gremlin*.

Atualmente, alguns SGBDGs têm recebido destaque entre pesquisadores e empresas. Uma grande porção destes modelam os dados nativamente como grafos, porém, possuem características diversas que vão desde o tipo de grafo usado na modelagem até a forma de armazenamento de dados em disco. O objetivo deste artigo é apresentar e comparar algumas características de alguns sistemas de processamento nativo. As características usadas na comparação oferecem uma visão geral quanto à possibilidade dos usuários alterarem as funcionalidades dos sistemas, à facilidade do uso das funcionalidades dos sistemas e às variáveis envolvidas no desempenho dos sistemas, sendo elas: tipo de licença, suporte ao *Tinkerpop*, modelagem lógica, linguagem de consulta, modelagem física, armazenamento físico, suporte à transação, tipo de arquitetura e suporte à replicação.

²<http://wiki.dbpedia.org/Datasets>

³<http://www.w3.org/wiki/LargeTripleStores>

⁴twitter.com

⁵www.tinkerpop.com

As contribuições deste artigo envolvem uma descrição do processo de construção e manipulação de uma base de dados em grafos considerando as características utilizadas na comparação objeto deste artigo e um estudo comparativo entre cinco sistemas de bancos de dados em grafos nativos. Os sistemas analisados foram: InfiniteGraph⁶, Neo4j⁷, OrientDB⁸, Titan⁹ e o Trinity [Shao et al. 2013].

O restante deste artigo está organizado da seguinte forma: os principais conceitos de um SGBDG são definidos pela Seção 2; nas seções 3 e 4 é apresentada uma visão geral sobre modelagem em grafos e consultas em grafos; os sistemas selecionados para a análise proposta são introduzidos pela Seção 5 e comparados na Seção 6; a Seção 7 apresenta os trabalhos relacionados deste artigo e; a Seção 8 apresenta as conclusões.

2. Sistema Gerenciador de Banco de Dados em Grafo

Na execução de um projeto de banco de dados, uma base é modelada logicamente e mapeada para uma forma de armazenamento físico para que seus dados possam ser manipulados por um sistema. A modelagem lógica abstrai detalhes do projeto facilitando o seu desenvolvimento. Já o mapeamento e o armazenamento físico envolvem detalhes técnicos e implicam diretamente no desempenho das aplicações.

Naturalmente, a modelagem lógica em um SGBDG é feita por meio do modelo de grafos. Basicamente, a topologia de um grafo G pode ser expressa como $G = (V, E)$, onde V é o conjunto de vértices (nós) e E é o conjunto de arestas. Cada aresta representa uma relação entre dois nós. Um conjunto de vértices conectados por meio de arestas definem um caminho no grafo. Diferentes tipos de grafos podem ser usados na modelagem, como mostra a próxima seção.

O mapeamento lógico-físico classifica os SGBDGs em não-nativos ou nativos. Os não-nativos modelam logicamente seus dados como grafos, porém armazenam-os por meio de outros modelos. Sistemas como RDF-3X [Neumann and Weikum 2010] e SW-Store [Abadi et al. 2009] armazena suas triplas em tabelas relacionais, enquanto o HyperGraphDB [Iordanov 2010] armazena o grafo fisicamente em uma estrutura chave-valor. Já os nativos, em geral, usam listas de adjacência [Aho and Ullman 1995]. Em uma lista de adjacência, cada vértice mantém referências diretas para seus vértices adjacentes formando uma espécie de micro-índice para os vértices próximos. Esta propriedade é conhecida como adjacência livre de índices [Robinson et al. 2013].

Um modelo deve ser implementado para o armazenamento físico dos dados representados por meio de vértices e arestas no modelo lógico. Por exemplo, as listas de adjacência podem ser armazenadas em um repositório por meio do modelo chave-valor onde a chave identifica um vértice e o valor referencia a lista de adjacência. Outra característica importante no armazenamento físico é a forma de armazenamento, em memória ou em disco. Vários sistemas exploram a primeira opção visando melhorar seu desempenho descartando a necessidade de acessos ao disco. Porém, o armazenamento em memória trabalha com dados voláteis e limita a capacidade de armazenamento quando comparado ao disco.

⁶www.objectivity.com/infinitegraph

⁷www.neo4j.org

⁸orientdb.org

⁹thinkaurelius.github.io/titan

Ainda em relação ao armazenamento físico, um SGBD pode ter uma arquitetura centralizada ou distribuída. Na centralizada, os dados ficam em um único servidor responsável por receber/responder todas as requisições dos clientes. Na distribuída, o banco de dados é fragmentado e suas partes componentes são fisicamente armazenadas em diferentes servidores [Date 2004]. Em geral, sistemas que usam a arquitetura distribuída oferecem a funcionalidade de replicação de dados com o objetivo de melhorar a escalabilidade e a disponibilidade de suas aplicações. Os SGBDs distribuídos são a solução ideal para garantir um bom desempenho nas aplicações que demandam um grande volume de requisições e de dados.

3. Modelos de Grafos

Uma base pode ser modelada de diferentes formas dependendo do modelo de grafo escolhido. Um modelo bem simples e limitado é o grafo simples-relacional. Nele todos os vértices denotam o mesmo tipo de objeto e todas as arestas denotam o mesmo tipo de relacionamento. Já o grafo multi-relacional permite um conjunto variado de tipos de objetos e de relacionamentos possibilitando múltiplas relações e um maior poder de modelagem. Ainda há a opção de uma aresta ser direcionada e/ou rotulada e/ou valorada com um peso em um modelo. Adicionalmente, arestas e vértices podem ter propriedades com valores associados [Rodriguez and Shinavier 2008].

Angles[Angles 2012] cita quatro tipos de modelos:

1. Grafos simples: é a definição básica em que um grafo é definido por um conjunto de vértices conectados por arestas par a par;
2. Hipergrafos: uma aresta (hiper-aresta) pode se relacionar com um número arbitrário de vértices;
3. Grafos aninhados: um vértice (hiper-vértice) também pode ser um grafo;
4. Grafos de propriedades: vértices e arestas contêm atributos para descrever suas propriedades.

O modelo mais utilizado entre os SGBDGs atuais é o grafo de propriedades. Rodriguez e Neubauer [Rodriguez and Neubauer 2010] definem este modelo como um grafo multi-relacional com atributos e arestas direcionadas. A Figura 1 exibe um grafo de propriedades no contexto de uma rede social corporativa. Os vértices que representam as pessoas possuem as propriedades “nome” e “idade” e os vértices que representam as empresas possuem as propriedades “nome” e “cidade”. As arestas representam as relações “um empregado trabalha para uma empresa” e “uma pessoa conhece outra pessoa”.

4. Operações sobre Grafos

Quando uma base é armazenada de forma nativa, vários algoritmos podem ser utilizados para consultá-la. Angles[Angles 2012] considera quatro grupos de consultas essenciais em grafos, sendo elas:

- Consultas adjacentes: o princípio básico é a adjacência vértice-aresta. Dois vértices são adjacentes quando possuem uma aresta entre eles. Duas arestas são adjacentes quando compartilham um vértice em comum.
- Consultas de acessibilidade: este tipo de consulta considera um caminho ou a travessia em um grafo. Ele testa se dois vértices estão conectados por um caminho no grafo.

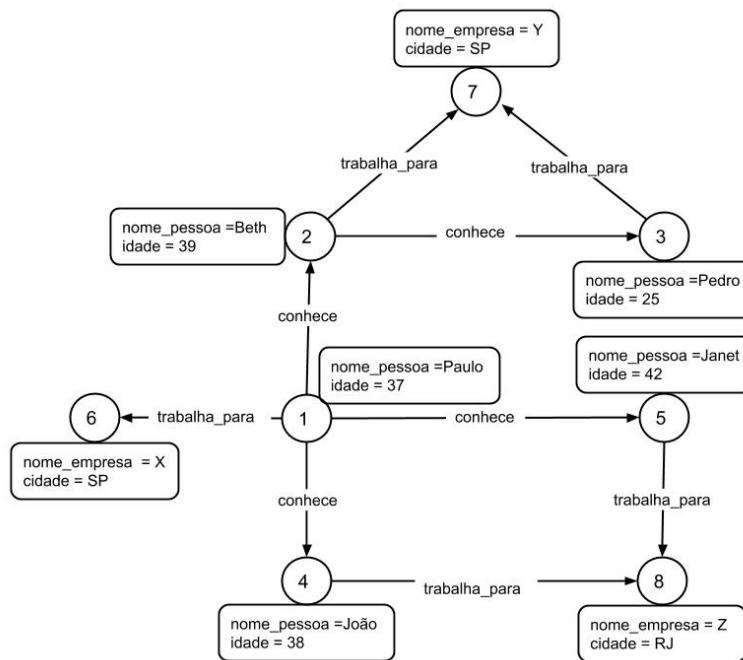


Figura 1. Uma rede social corporativa

- Consultas de combinações de padrões: este tipo de consulta procura todos os subgrafos isomórficos de um grafo padrão de uma determinada consulta.
- Consultas de summarização: este tipo de consulta usa funções especiais que permitem summarizar ou operar nos resultados das consultas, normalmente retornando um valor simples.

Em contrapartida, os SGBDGs não-nativos não conseguem explorar os mesmos algoritmos de consulta que os nativos. Logo, o processamento de suas consultas dependem diretamente da forma de armazenamento escolhida.

Os exemplos da Tabela 1 exibem consultas básicas no grafo da Figura 1 com o *Gremlin*¹⁰, ferramenta desenvolvida pelo *Tinkerpop*. No quesito consulta, *Gremlin* é classificada como uma linguagem de exploração que usa algoritmos de busca em profundidade e em largura. *Gremlin* atende os tipos de consultas listadas anteriormente por meio de diferentes funções de manipulação de dados.

#	Descrição da Consulta	Expressão em <i>Gremlin</i>
1	Nomes dos integrantes da rede social	<code>g.V.nome_pessoa</code>
2	Idade do Paulo	<code>g.V('nome_pessoa', 'Paulo').idade</code>
3	Pessoa que o Paulo conhece	<code>g.V('nome_pessoa', 'Paulo').out('conhece').nome_pessoa</code>
4	Empresa onde trabalham os conhecidos do Paulo	<code>g.V('nome_pessoa', 'Paulo').out('conhece').out('trabalha_para').nome_empresa</code>
5	Nome e idade dos funcionários da empresa X	<code>g.V('nome_empresa', 'X').in('trabalha_para').map()</code>

Tabela 1. Consultas em *Gremlin*

¹⁰<https://github.com/tinkerpop/gremlin/wiki>

Em todos os exemplos “g” referencia o grafo exemplo e “V” é uma variável iteradora para referenciar os vértices do grafo “g”. Cada passo da execução (denotado por um ponto) é uma função que opera nos resultados obtidos no passo anterior. O operador “out” busca as arestas que partem de um vértice e o “in” busca as arestas incidentes em um vértice. A operação “map” captura todas as propriedades do vértice ou aresta em questão.

Além das operações de leitura, uma base também pode receber operações de escrita. Alterações em dados que estão sendo utilizados concorrentemente por diversos usuários exigem um gerenciamento mais complexo por parte dos SGBDs. Para isso alguns SGBDGs dão suporte a transações. Uma transação é uma unidade lógica de trabalho que possui um objetivo específico [Date 2004]. Um SGBD transacional controla a concorrência das operações e também dá suporte à recuperação de dados caso ocorra alguma falha no processamento das transações. Diferentes modelos de consistência em transações podem ser usados nos SGBDs, como o ACID e o BASE [Vogels 2009]. O segundo modelo permite uma consistência dos dados mais relaxada quando comparado com o primeiro.

5. Sistemas Analisados

Todos os sistemas analisados nesse artigo são nativos, ou seja, todos usam listas de adjacências para armazenar grafos e executam algoritmos de exploração de grafos em suas consultas. Apesar deles apresentarem várias características em comum, cada um possui suas particularidades tornando-se adequado ou não para uma determinada aplicação. Nas descrições apresentadas a seguir foram consideradas nove características que podem influenciar na escolha de um sistema, sendo elas:

1. Tipo de licença: há basicamente dois grandes grupos, os sistemas abertos que disponibilizam seus códigos-fonte permitindo alterações, e os proprietários que possuem os códigos fechados permitindo somente o uso dos sistemas.
2. Suporte ao *Tinkerpop*: permite o uso de diversas ferramentas para manipulação de grafos desenvolvidas no projeto *Tinkerpop*.
3. Modelagem lógica: o modelo lógico usado no sistema influencia na facilidade de modelagem de uma aplicação pelo usuário e diz respeito ao modelo de grafo empregado.
4. Linguagem de consulta: define as operações e a forma de manipulação da base por meio de sua sintaxe.
5. Modelagem física: o modelo físico influencia no desempenho e implementação das operações sobre grafos.
6. Armazenamento físico: há dois tipos, em memória e em disco. O armazenamento físico influencia na capacidade de armazenamento e no desempenho das aplicações.
7. Suporte à transação: constitui uma característica essencial em aplicações que precisam garantir recuperação e controle de concorrência em suas requisições.
8. Tipo de arquitetura: há dois tipos, a centralizada e a distribuída. Em geral, a primeira atende às necessidades de aplicações simples e de pequeno porte e, a segunda, às de aplicações que demandam grandes volumes de requisições e de dados.

9. Suporte à replicação: importante para garantir a disponibilidade e a escalabilidade nas aplicações distribuídas.

As características listadas acima oferecem uma visão geral dos sistemas quanto à possibilidade de alteração das suas funcionalidades (característica 1), quanto à facilidade do uso destas funcionalidades (características 2, 3 e 4) e quanto às variáveis envolvidas no desempenho dos sistemas (características 5, 6, 7, 8 e 9). A seguir são apresentados os cinco SGBDGs selecionados para a análise.

5.1. InfiniteGraph

InfiniteGraph é um banco de dados proprietário da Objectivity¹¹. Sua implementação foi feita em Java, com *core* em C++, e sua primeira versão foi lançada em 2010. InfiniteGraph é um banco transacional que oferece a arquitetura centralizada e a distribuída com um mecanismo de replicação com consistência relaxada dos dados.

O modelo lógico suportado pelo InfiniteGraph é o grafo de propriedades. Ele dá suporte ao *Tinkerpop* e também fornece uma API para consultas em Java. O modelo físico usado é orientado a objetos e, sendo assim, duas classes são usadas para armazenar os vértices e os nós de uma base, a *BaseVertex* e a *BaseEdge*, respectivamente. Ele oferece somente o armazenamento físico em disco.

5.2. Neo4j

A primeira versão do Neo4j foi lançada em fevereiro de 2010 pela empresa *Neo Technology*¹². Sua implementação foi feita em Java e ele possui duas versões de licenciamento, uma aberta e outra proprietária. A diferença entre elas é que a proprietária tem código fechado, oferece suporte aos seus usuários e possui mais recursos para garantir o desempenho de grandes aplicações. O Neo4j é transacional e oferece os dois tipos de arquitetura, a centralizada e a distribuída com suporte à replicação.

O Neo4j também utiliza o modelo de grafos de propriedade e oferece suporte ao *Tinkerpop*. Dessa forma, tanto a definição dos esquemas das bases quanto a manipulação dos dados (carregamento e consulta) podem ser feitas por meio da linguagem *Gremlin*. Além disso, o Neo4j possui a linguagem de consulta proprietária *Cypher* e também dá suporte à SPARQL (SQL for linked data). O armazenamento físico pode ser tanto em memória quanto em disco e o modelo físico é baseado em repositórios chave-valor.

5.3. OrientDB

O OrientDB é um banco de dados em grafo aberto lançado comercialmente em 2011 pela empresa OrientTechnologies¹³. Ele foi implementado em Java, é transacional e dá suporte à arquitetura centralizada e distribuída com replicação.

O OrientDB oferece grafos de propriedades para a modelagem lógica e dá suporte ao *Tinkerpop*. A manipulação da base pode ser feita com o *Gremlin*, com o Java ou com uma adaptação do SQL desenvolvida no próprio projeto. O armazenamento físico dos dados pode ser feito em memória e em disco. Como todos os sistemas, ele usa a lista

¹¹www.objectivity.com

¹²[neotechnology.com](http://www.neotechnology.com)

¹³<http://www.orientechnologies.com>

livre de adjacências para viabilizar o processamento nativo das consultas, porém, diferentemente dos outros ele usa recursos de banco de dados de documentos e de orientação a objetos para o armazenamento físico dos vértices para garantir uma maior flexibilidade na estrutura dos dados a serem armazenados na base.

5.4. Titan

Titan é um sistema aberto mantido pela empresa Aurelius¹⁴ desde 2012. Ele foi implementado em Java e pode ser classificado como um framework transacional para banco de dados em grafos. Ele dá suporte tanto à arquitetura centralizada quanto à distribuída.

O projeto do Titan é completamente dependente do *Tinkerpop*. Tanto a definição dos esquemas das bases quanto a manipulação dos dados (carregamento e consultas) são feitos por meio da linguagem *Gremlin*. Logo, a modelagem lógica é feita por meio de grafos de propriedades e o mapeamento por meio de listas de adjacência, que são essenciais ao *Gremlin*. Quanto a forma de armazenamento, Titan oferece a opção de armazenar os dados em memória, desde que a base seja relativamente pequena, e em disco. Os três principais banco de dados utilizados para armazenamento em disco são: Apache Cassandra, Apache HBase e Oracle Berkeley DB. Basicamente, os três utilizam pares chave-valor para armazenar fisicamente os dados. Cada banco tem suas características particulares e a escolha do banco determina a garantia transacional, o suporte à replicação e a escalabilidade de uma aplicação.

5.5. Trinity

O projeto Trinity encontra-se em fase de desenvolvimento pela Microsoft Research. Trinity [Shao et al. 2013] denomina-se um banco de dados distribuído em memória que também constitui uma plataforma computacional para suporte do processamento de consultas sobre grafos.

Na modelagem lógica, Trinity suporta grafos direcionados, não-direcionados e hipergrafos. Diferente dos outros sistemas, ele não dá suporte ao *Tinkerpop*. O processamento bem como a linguagem de consultas é dependente do *engine* de processamento constituído sobre a plataforma Trinity. Trinity.RDF [Zeng et al. 2013] é um exemplo de *engine* constituído sobre esta plataforma para o processamento de consultas SPARQL. O modelo físico adotado por Trinity corresponde ao modelo de um repositório chave-valor onde um par chave-valor representa um nó do grafo RDF e sua lista de adjacência.

6. Análise dos SGBDGs

A Tabela 2 apresenta os sistemas e as características usadas na comparação objeto deste artigo. Alguns sistemas possuem o tipo de licença aberto, mas, infelizmente, eles possuem códigos complexos e não disponibilizam documentação suficiente dos projetos inviabilizando alterações em suas funcionalidades. Quanto ao *Tinkerpop*, pode-se notar a sua importância nos sistemas atuais. Como consequência, predominam o grafo de propriedades como modelo lógico e o *Gremlin* como linguagem de consulta. Esse suporte amplo ao *Tinkerpop* facilita o uso dos sistemas pela comunidade de banco de dados em grafos. Como já citado nas seções anteriores, todos os sistemas fazem o mapeamento lógico-físico por meio de listas livre de adjacências, porém o modelo físico varia entre

¹⁴thinkaurelius.com

eles. Nesse quesito a maioria usa repositórios chave-valor devido à sua simplicidade e desempenho, porém, o OrientDB destaca-se pelo uso do modelo de documentos que dá suporte às aplicações que não têm esquemas de dados pré-definidos. A maioria dá suporte ao armazenamento físico dos dados tanto em memória quanto em disco, lembrando que o uso somente da memória trata de dados voláteis e limita o espaço de armazenamento dos dados, mas, em contrapartida, melhora o desempenho de consultas em grafos. Todos os sistemas são transacionais e dão suporte à distribuição de dados e à replicação com o objetivo de melhorar, respectivamente, a escalabilidade e a disponibilidade de suas aplicações.

SGBDG	Aberto/ Proprietário	<i>Tinkerpop</i>	Modelo lógico	Linguagem de consulta	Modelo físico	Memória/ Disco	Centralizada/ Distribuída	Transação/ Replicação
InfiniteGraph	proprietário	✓	propriedades	Gremlin	objetos	disco	ambas	ambas
Neo4j	ambos	✓	propriedades	Gremlin/Cypher/ SPARQL	chave-valor	ambos	ambas	ambas
OrientDB	aberto	✓	propriedades	Gremlin/SQL	documentos	ambos	ambas	ambas
Titan	aberto	✓	propriedades	Gremlin	chave-valor	ambos	ambas	ambas
Trinity	proprietário		hipergrafos	SPARQL	chave-valor	memória	distribuída	ambas

Tabela 2. SGBDGs e características

7. Trabalhos Relacionados

Alguns trabalhos na literatura comparam bancos de dados em grafos considerando características diversas. Angles apresentou dois estudos comparativos com diversos sistemas da época, um em 2008 [Angles and Gutierrez 2008] e outro em 2012[Angles 2012]. Os estudos apresentam uma análises em nível lógico e comparam, segundo Angles, características de modelos de dados, sendo elas: estruturas de dados, formas de consultas e restrições de integridade. Por outro lado, com foco em desempenho, [Ciglan et al. 2012] apresentou uma análise de cinco sistemas por meio de um *benchmark* e, [McColl et al. 2013] publicou um breve relatório técnico considerando diversos sistemas abertos. O primeiro considerou somente o processamento de consultas e, o segundo, consultas e atualizações. Nesse artigo o critério de seleção dos sistemas foi o processamento de consultas nativo sobre grafos e o objetivo da comparação apresentada é mostrar uma visão geral quanto à possibilidade dos usuários alterarem as funcionalidades dos sistemas, quanto à facilidade do uso das funcionalidades dos sistemas e quanto às variáveis envolvidas no desempenho dos sistemas.

8. Conclusão

Em geral, os sistemas nativos oferecem um melhor desempenho no processamento de consultas quando comparado com os não-nativos, porém, várias características diferenciam esses sistemas. Na comparação objeto desse artigo pode-se notar que o suporte ao *Tinkerpop* é uma tendência entre eles devido à facilidade de manipulação de dados que as ferramentas do projeto oferecem aos usuários. Desta forma, o modelo de grafos de propriedades e a linguagem *Gremlin* estão presentes na maioria dos sistemas analisados, apesar de ainda não existir um padrão de linguagem de consulta para banco de dados em grafos. Outro destaque é o suporte à distribuição de dados. Todos procuram atender às necessidades das aplicações atuais que demandam grandes volumes de requisições e de dados. O suporte à distribuição e à replicação oferece uma maior escalabilidade e disponibilidade às aplicações.

Agradecimentos Este trabalho foi parcialmente financiado pela Capes, Fundação Araucária e *Microsoft Azure*.

Referências

- Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2009). Sw-store: A vertically partitioned dbms for semantic web data management. *The VLDB Journal*, 18(2):385–406.
- Aho, A. and Ullman, J. (1995). *Foundations of Computer Science: C Edition*. Principles of Computer Science Series. W. H. Freeman.
- Angles, R. (2012). A comparison of current graph database models. In *ICDE Workshops*, pages 171–177.
- Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39.
- Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.
- Ciglan, M., Averbuch, A., and Hluchý, L. (2012). Benchmarking traversal operations over graph databases. In Kementsietsidis, A. and Salles, M. A. V., editors, *ICDE Workshops*, pages 186–189. IEEE Computer Society.
- Date, C. (2004). *Introdução a Sistemas de Bancos de Dados*. Campus.
- Iordanov, B. (2010). Hypergraphdb: A generalized graph database. In Shen, H., Pei, J., Özsü, M., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., and Shao, J., editors, *Web-Age Information Management*, volume 6185 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin Heidelberg.
- Liptchinsky, V., Satzger, B., Zabolotnyi, R., and Dustdar, S. (2013). Expressive Languages for Selecting Groups from Graph-structured Data. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 761–770.
- McColl, R., Ediger, D., Poovey, J., Campbell, D., and Bader, D. A. (2013). A brief study of open source graph databases. *CoRR*, abs/1309.2675.
- Neumann, T. and Weikum, G. (2010). The rdf-3x engine for scalable management of rdf data. *The VLDB Journal*, 19(1):91–113.
- Robinson, I., Webber, J., and Eifrem, E. (2013). *Graph Databases*. O'Reilly Media.
- Rodriguez, M. A. and Neubauer, P. (2010). The graph traversal pattern. *CoRR*, abs/1004.1001.
- Rodriguez, M. A. and Shinavier, J. (2008). Exposing multi-relational networks to single-relational network analysis algorithms. *CoRR*, abs/0806.2274.
- Shao, B., Wang, H., and Li, Y. (2013). The Trinity Graph Engine. In *SIGMOD international conference on Management of data*.
- Vogels, W. (2009). Eventually consistent. *Commun. ACM*, 52(1):40–44.
- Zeng, K., Yang, J., Wang, H., Shao, B., and Wang, Z. (2013). A Distributed Graph Engine for Web Scale RDF Data. *VLDB Endowment*.

Comparação de Performance entre PostgreSQL e MongoDB

Cristiano Politowski¹, Vinícius Maran¹

¹ DCEEng – Departamento de Ciências Exatas e Engenharias - Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ) – RS 344 s/n, Santa Rosa - RS

crispolitowski@gmail.com, vinicius.maran@unijui.edu.br

Abstract. From traditional DBMS solutions involving the storage of documents, graphs and even key-value structures, models of databases are created with the passage of time, and made the term NoSQL arise. However, doubts permeate this scenario, for example, the performance of these new technologies with respect to relational databases. The main aim of this work is perform a comparison between the database MongoDB document-oriented and relational database PostgreSQL, based on the runtime of read and write operations. As a result we realized a big difference in performance between the tested data bases, especially in search operations.

Resumo. Dos SGBDs tradicionais à soluções envolvendo o armazenamento de documentos, grafos e até estruturas chave-valor, modelos de bancos de dados são criados com o passar do tempo, e fizeram surgir o termo NoSQL. No entanto, algumas dúvidas permeiam este cenário, como, por exemplo, a performance de novas tecnologias em relação aos bancos de dados relacionais. O principal objetivo deste trabalho é realizar uma comparação entre o banco de dados orientado a documentos MongoDB e o banco de dados relacional PostgreSQL, baseado no tempo de execução de operações de leitura e escrita. Como resultado percebeu-se uma grande diferença de performance entre os bancos testados, principalmente nas operações de busca.

1. Introdução

O mundo nunca trabalhou com volumes de dados tão grandes [Diana and Gerosa 2010], e as grandes aplicações Web são as grandes responsáveis. O alto tráfego de dados exige que empresas procurem soluções para poder suprir a demanda cada vez maior de performance e disponibilidade que fazem com que outros requisitos até então indiscutíveis, como consistência de dados, sejam revistos [Vogels 2009, Pritchett 2008].

Bancos de dados Relacionais são os mais utilizados para armazenamento de dados a décadas nos permitindo gravar grandes porções de dados no disco e provendo uma maneira simples de coletar estes dados através de *queries* SQL. Esta, uma linguagem padrão (na maioria dos Sistemas de Gerenciamento de Banco de Dados (SGBDs)) de interação com o banco de dados que mantém todos familiarizados com sua sintaxe, evitando problemas relacionados ao *language cacophony problem*¹.

¹Cacofonia de Linguagem: consiste na preocupação de que se aprender uma linguagem é difícil, ao usar várias se tornaria ainda mais [Fowler 2010]

Outro fator importante do sucesso dos Banco de Dados Relacionais (BDRs) é a integração de aplicações usando a mesma base de dados. Isso assegura a consistência dos dados e os mantém sempre atualizados. Esta característica só é possível devido ao controle de concorrência provido pelos SGBDs, como, por exemplo, o uso de transações. No entanto, tecnologias emergentes, recentemente chamadas de movimento NoSQL (*Not Only SQL*), surgiram como uma nova classe de SGBDs, que utilizam outros modelos, diferentes do relacional e que vem evoluindo trazendo um paradigma diferente de armazenamento de dados, como, por exemplo, armazenamento de documentos sem esquema definido.

Por se tratar de um conjunto de tecnologias e modelos relativamente novos, os bancos de dados NoSQL são vistos com desconfiança por parte da comunidade e descrença por parte de outros. Apesar de grandes corporações como Google e Twitter terem adotado a solução, usá-la em produção é sempre um risco, visto a sua prematuridade em relação aos BDRs.

Ainda existem dúvidas em relação ao potencial da aplicação da tecnologia NoSQL em algumas áreas. Portanto o principal objetivo deste artigo é realizar uma comparação de desempenho de consultas entre dois bancos de dados gratuitos e de grande adoção por parte dos usuários. Para realizar este comparativo, foram escolhidas as ferramentas PostgreSQL [PostgreSQL 2014a] e MongoDB [MongoDB 2014a]. Através da implementação de um ambiente de testes, foi possível realizar a comparação, analisar os resultados e tirar conclusões sobre o desempenho das duas ferramentas no cenário utilizado.

O artigo está estruturado como descrito a seguir. Na Seção 2 são apresentados os trabalhos relacionados e seus resultados. Na Seção 3 é apresentado um estudo sobre a área de banco de dados NoSQL. Na Seção 4, são apresentadas as características dos bancos de dados testados. A Seção 5 apresenta a etapa de preparação do ambiente para a execução dos testes. Na Seção 6 são apresentados os resultados dos testes. Por fim, a Seção 7 apresenta as conclusões finais deste trabalho.

2. Trabalhos Relacionados

Um dos aspectos que motivaram a realização deste trabalho é a pequena quantidade de trabalhos relacionados. Boicea [Boicea et al. 2012] faz uma comparação de performance entre MongoDB e Oracle [Oracle 2014], detalhando suas diferenças e fazendo testes de inserção, remoção e atualização de dados. Como resultado houve uma grande diferença de desempenho, em favor do banco de dados MongoDB, principalmente quando o número de registros no banco de dados é grande.

Li [Li and Manoharan 2013], por sua vez, faz testes de leitura, escrita, remoção e instanciação de vários bancos NoSQL e também do banco relacional MS SQL Server [Microsoft 2014]. Como resultado, foi possível afirmar que os bancos NoSQL obtiveram melhores resultados, porém, variando muito de acordo com a ferramenta utilizada.

Parker [Parker et al. 2013] também compara MongoDB com MS SQL Server. Os resultados são semelhantes para uma estrutura de base de dados modesta. MongoDB obteve resultados um pouco melhores em todas as comparações exceto quando foram utilizadas funções agregadas.

Há trabalhos relacionados ao uso dos bancos de dados MongoDB e Postgresql [Carniel et al. 2012b, Carniel et al. 2012a, Litt et al.], porém, estes trabalhos não fazem uma comparação direta entre os bancos de dados, que é o objetivo deste trabalho.

3. NoSQL - Not Only SQL

Bancos de Dados Relacionais foram projetados para serem executados em uma máquina apenas, onde para escalar, é necessário comprar uma máquina melhor (ou fazer um *upgrade*) com mais recursos, necessários para lidar com o alto tráfego de dados na web (*big data*). Esta técnica é chamada de escalabilidade vertical. Porém percebe-se que essa técnica, além de economicamente inviável, pois o custo de se melhorar um hardware é muito alto, é também limitada, pois chegará um momento em que o hardware não poderá ser melhorado. Em contrapartida, quando a escalabilidade vertical não é mais possível, técnicas de escalabilidade horizontal, através do uso de clusters de máquinas, podem ser úteis.

O ascensão dos *Web Services* proveram uma alternativa efetiva aos bancos de dados compartilhados para a integração de aplicativos, tornando fácil para diferentes aplicações escolherem seu próprio meio armazenamento de dados [Fowler 2012]. Pode-se dizer então que os precursores do movimento NoSQL foram empresas que, dada esta necessidade, buscaram novas soluções como o Bigtable [Bigtable 2014] e o Dynamo [Dynamo 2014].

NoSQL não possui um significado padrão, segundo Fowler [Fowler 2012] NoSQL pode ser traduzido em um conjunto de características, apresentadas a seguir:

- Não usa modelo de dados relacional e portanto não usa a linguagem SQL;
- Costuma ser projetado para ser executado em um *cluster*;
- Costuma ser *Open-Source*;
- Não possui esquema fixo (*SCHEMA-LESS*), permitindo gravar qualquer dado em qualquer estrutura.

4. Bancos de Dados Utilizados no Comparativo

4.1. MongoDB

MongoDB (de “*humongous*”) é um banco de dados *schema-less*, orientado a documentos, *open-source* escrito em C++ [MongoDB 2014a].

MongoDB persiste documentos no formato BSON (*Binary JSON*), que são objetos JSON binários. BSON, por sua vez, suporta estruturas como *arrays* e *embedded objects* assim como JSON. MongoDB permite usuários realizem modificações de apenas um atributo em um documento sem a necessidade de interação com o restante da estrutura.

Documentos podem ser armazenados em coleções (*collections*), onde serão efetuadas operações de busca (*queries*) e indexação (*indexing*). *Queries* são expressadas na sintaxe JSON e enviadas ao MongoDB como objetos BSON pelo *driver* de conexão ao banco.

Para persistência, MongoDB usa arquivos mapeados em memória, deixando o gerenciador de memória virtual do sistema operacional decidir quais partes vão para o

disco e quais ficam na memória. Isto faz com que o MongoDB não tenha controle sobre o momento onde os dados são escritos no disco [Matthes and Orend 2010].

MongoDB não possui controle de concorrência e gerenciamento de transações. Então, se um usuário lê um documento, escreve uma modificação e devolve ao banco de dados, pode acontecer de outro usuário escrever uma nova versão do documento entre o processo de leitura e escrita do primeiro.

4.2. PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (SGB-DOR)² baseado no POSTGRES Versão 4.2, desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. É totalmente compatível com ACID³, tem suporte completo a chaves estrangeiras, junções, visões e gatilhos. Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos.

Possui controle de concorrência multiversionado (MVCC - *Multi-Version Concurrency Control*), recuperação em um ponto no tempo (PITR - *Point in Time Recovery*), tablespaces, replicação assíncrona, transações agrupadas (*savepoints*), cópias de segurança (*online/hot backup*), um planejador de consultas e registrador de transações sequencial para tolerância a falhas.

5. Ambiente de testes

Os testes foram feitos em um computador com processador Intel Core i7-3630QM CPU de 2.4 GHz com 8 GB de RAM e SSD de 128 GB com o sistema operacional Linux **Ubuntu** [Ubuntu 2014] 13.04 de 64 bits. A versão do PostgreSQL usada foi a 9.1 a do MongoDB foi a 2.4.9.

Para implementar os testes, a linguagem **Python** [Python 2014] foi utilizada juntamente com os respectivos *drivers* necessários para a conexão com o banco. Para o PostgreSQL, foi utilizado o *driver psycopg* [Psycopg 2014] e para o MongoDB foi utilizado o *driver pymongo* [Pymongo 2014]. Para medir o tempo de execução, foi usado o módulo da linguagem Python **timeit** [Timeit 2014].

A estrutura do banco foi estabelecida através de um diagrama de classe, apresentado na Figura 1. Podemos entender esse diagrama como a modelagem de um *blog*. É difícil fazer uma comparação entre dois bancos de dados que armazenam estruturas de dados diferentes, por isto a mesma estrutura foi utilizada nas duas ferramentas durante os testes.

Os testes foram divididos em três categorias: **inserção**, **busca simples** e **busca complexa**. Cada teste foi executado em um *loop* de 1, 10, 100, 1.000, 10.000 e 100.000 vezes, repetidos três vezes sendo considerada a média aritmética dos mesmos.

²Um banco de dados objeto-relacional (ORD), ou sistema de gerenciamento de banco de dados objeto-relacional (ORDBMS ou SGBDOR), é um sistema de gerenciamento de banco de dados relacional que permite aos desenvolvedores integrar ao banco de dados seus próprios tipos de dado e métodos.

³Atomicidade, Consistência, Isolamento e Durabilidade

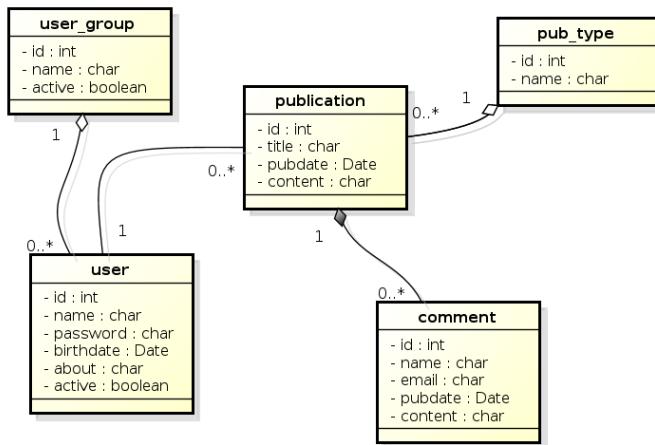


Figura 1. Diagrama de classe representando a estrutura de dados utilizada nos testes.

A **inserção** foi feita englobando todas as tabelas no PostgreSQL e um documento completo no MongoDB com todos os dados preenchidos. A **busca simples** consiste na busca de todas as publicações com o título Teste publicacao. Já na **busca complexa** o resultado deve trazer todas as publicações entre as datas 2013-1-1 e 2015-1-1, que sejam do tipo Novidade, de usuários nascidos a partir de 1980-10-27, que estejam ativos, pertencentes ao grupo de nome Administracao e que possua, pelo menos, 1 comentário. Tudo isso ordenado pela data de publicação. Todas retornaram 333.333 (trezentos e trinta e três mil, trezentos e trinta e três) registros, que foi a quantidade de inserções em ambos os bancos.

5.1. Operações com PostgreSQL

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

```
conn = psycopg2.connect("dbname=testedb user=teste")
cur = conn.cursor()
# <QUERIES AQUI>
conn.commit()
cur.close()
conn.close()
```

Na primeira linha a conexão é feita através do *driver* `psycopg2`. Na linha dois um cursor é criado para a manipulação do banco. Em `<QUERIES AQUI>` são colocadas as consultas na linguagem SQL pura, sem *frameworks*. Todas as consultas são executadas e persistidas (`commit`) apenas uma vez dentro do *loop* de repetições. No final da execução, tanto o cursor quanto a conexão são fechados.

A **inserção** é feita com cinco inserções nas respectivas tabelas, usando a interpolação de strings do Python (`%s`) para passar os valores. Essa técnica é recomendada pela documentação oficial da linguagem por questões de performance. O trecho `datetime.datetime.utcnow()` é uma função Python que retorna a data atual do sistema. O *driver* faz a conversão automática de tipos. O trecho de código a seguir apresenta um exemplo de realização de um conjunto de inserções realizadas nos testes:

```
cur.execute("INSERT INTO testesc.usuario_grupo (nome, ativo) VALUES (%s, %s)",
```

```

        ("Administracao", True))
cur.execute("INSERT INTO testesc.usuario (nome, senha, datanascimento, sobre, ativo,
    id_grupo) VALUES (%s, %s, %s, %s, %s)", ("Claudio Francisco da Silva", "admin123",
    "1987-10-27", "Pesquisador e programador nas horas vagas.", "true", 0))
cur.execute("INSERT INTO testesc.pub_tipo (nome) VALUES (%s)", ("Novidade",))
cur.execute("INSERT INTO testesc.publicacao (titulo, datapub, conteudo, id_usuario,
    id_pub_tipo) VALUES (%s, %s, %s, %s)", ("Publicacao teste",
    datetime.datetime.utcnow(), "Bla bla bla...", 0, 0))
cur.execute("INSERT INTO testesc.comentario (nome, email, datapub, conteudo,
    id_publicacao) VALUES (%s, %s, %s, %s)", ("Fulano de Tal", "fulano@gmail.com",
    datetime.datetime.utcnow(), "First!", 0))

```

O trecho do código para a operação de **busca simples** usa apenas uma cláusula WHERE para achar a o título 'Publicacao teste' através do LIKE. O trecho de código a seguir apresenta a realização desta busca nos testes:

```
cur.execute( "SELECT * FROM testesc.publicacao p WHERE p.titulo LIKE 'Publicacao teste'" )
```

O trecho do código para a operação de **busca complexa** (apresentado abaixo) usa várias junções entre tabelas e inúmeras funções AND além de uma sub-consulta para retornar o número de comentários. Novamente foi usado interpolação de strings.

```

cur.execute( "SELECT * FROM testesc.publicacao p, testesc.usuario u,
    testesc.usuario_grupo ug, testesc.pub_tipo pt, testesc.comentario c
    WHERE p.id_usuario = u.id_usuario AND u.id_grupo = ug.id_usuario_grupo
    AND p.id_pub_tipo = pt.id_pub_tipo AND c.id_publicacao = p.id_publicacao
    AND p.datapub > %s AND p.datapub < %s AND pt.nome LIKE %s
    AND u.datanascimento > %s AND u.ativo = %s
    AND ug.nome = %s AND (SELECT count(*) FROM testesc.publicacao p,
        testesc.comentario c WHERE c.id_publicacao = p.id_publicacao) >= 1
    ORDER BY p.datapub", (datetime.datetime(2013, 11, 12, 12),
    datetime.datetime(2015, 11, 12, 12), "Novidade", "1980-1-1", "True",
    "Administracao"))

```

5.2. Operações com MongoDB

Para as operações no banco de dados MongoDB, o código de conexão com o banco de dados foi estruturado da seguinte maneira:

```

client = MongoClient('localhost', 27017)
db = client.testedb
# <QUERIES AQUI>
client.close()

```

Na primeira linha, uma instância do MongoDB é criado. Após, é criado/recuperado o banco de dados de nome testedb. Assim como na estrutura do banco PostgreSQL, em <QUERIES AQUI> vão as respectivas consultas das operações. Ao final, a instância do MongoDB é fechada.

Para a **inserção**, um documento contendo todos os dados estruturados em formato de JSON válido foi criado e inserido através da função `insert()` dentro de uma coleção de documentos (`collection`) de nome 'publicacoes'. O trecho de código referente a operação de inserção nos testes é apresentado abaixo:

```

publicacao = { "publicacao": { "titulo": "Publicacao teste",
    "datapub": datetime.datetime.utcnow(), "conteudo": "Bla bla bla...",
    "tipo": "Novidade", "usuario": { "nome": "Claudio Francisco da Silva",
    "senha": "admin123", "datanascimento": "1987-10-27",
    "sobre": "Pesquisador e programador nas horas vagas.", "ativo": "True",
    "grupo": { "nome": "Administracao", "ativo": "True" } },
    "comentario": [ { "nome": "Fulano de Tal", "email": "fulano@gmail.com",
    "datapub": datetime.datetime.utcnow(), "conteudo": "First!" } ] }
publicacoes = db.publicacoes
publicacoes.insert(publicacao)

```

A **busca simples** (código apresentado abaixo) é feita através da função `find()` da coleção, passando um dicionário com as respectivas chaves e valores. Este dicionário de dados também deve ser um JSON válido.

```
publicacoes = db.publicacoes
publicacoes.find({"publicacao.titulo": "Publicacao teste"})
```

Para a **busca complexa** (código apresentado abaixo) utilizamos vários operadores de comparação da linguagem de consulta do MongoDB, como `$gt`, `$lt` e `$size`. Este último usado para o cálculo do número de comentários.

```
publicacoes = db.publicacoes
publicacoes.find({ "publicacao.datapub": {"$gt": datetime.datetime(2013, 11, 12, 12)},
    "publicacao.datapub": {"$lt": datetime.datetime(2015, 11, 12, 12)},
    "publicacao.tipo": "Novidade", "publicacao.usuario.datanascimento": {"$gte": "1980-1-1"},
    "publicacao.usuario.ativo": "True", "publicacao.usuario.grupo.nome": "Administracao",
    "publicacao.comentario": {"$size": 1 }}).sort("publicacao.datapub")
```

6. Resultados

A Tabela 1 apresenta os resultados dos testes feitos no banco PostgreSQL onde os valores estão representados em segundos. Nas operações de inserção o banco relacional obteve bons resultados, onde em um centésimo de segundo foi realizada a inserção de apenas um registro em todas as tabelas e em cerca de 18 minutos foi realizada a mesma operação, mas com trezentos mil registros. Para uma busca simples os problemas começam a aparecer, chegando a um total de 11 horas e meia para buscar todos os registros cem mil vezes. No entanto, a busca complexa, que possui *queries* mais elaboradas, teve um crescimento de 600% no tempo para recuperar apenas um registro e mais de 77 horas, em média, para fazer essa operação cem mil vezes.

Tabela 1. Testes com o banco PostgreSQL.

Operações	Número de repetições					
	1	10	100	1000	10000	100000
Inserção	0,0096	0,1108	1,0792	11,0729	105,1221	1106,3959
B. simples	0,3902	3,9198	39,5233	451,3150	4286,2453	40892,9120
B. complexa	2,3452	24,4739	250,1055	2568,9904	26929,7949	278438,6800

A Tabela 2 apresenta os resultados dos testes feitos no banco MongoDB onde os valores também estão representados em segundos. Ao contrário do banco anterior, a tarefa que mais despende esforço, nesse caso, é a operação de inserção. Isto acontece por ser uma ação bloqueadora, diferente das buscas. Essas, por sua vez, mostram nos resultados obtidos o motivo do banco de dados MongoDB ser bem aceito na comunidade⁴. A primeira constatação é de que a diferença entre as duas buscas é inexpressiva. O outro detalhe, é a velocidade para a busca de grandes quantidades de dados, onde foi possível recuperar todos os dados, cem mil vezes, em pouco mais de 50 segundos.

Para mostrar que a comparação entre os valores médios é significativa, foi utilizado o Teste T de Student que usa conceitos estatísticos para rejeitar ou não uma hipótese nula. Usando uma significância de 5%, achamos um **valor-p** sempre abaixo desse parâmetro, com exceção da operação de inserção de 1 registro / documento, mostrando a relevância na diferença dos tempos coletados. O resultado deste teste é apresentado na Tabela 3.

⁴Quinto lugar no ranking do site DB-Engines [DB-Engines 2014].

Tabela 2. Testes com o banco MongoDB.

Operações	Número de repetições					
	1	10	100	1000	10000	100000
Inserção	0,0046	0,0072	0,0781	0,7294	7,0441	77,7849
B. simples	0,0007	0,0051	0,0528	0,5159	5,0282	49,7979
B. complexa	0,0008	0,0085	0,0529	0,5068	5,1852	51,7124

Tabela 3. Teste T Student

Operações	Número de repetições					
	1	10	100	1000	10000	100000
Inserção	0,30366106	0,00043316	0,00000004	0,00016811	0,00038688	0,00000283
B. simples	0,00001081	0,00010017	0,00019375	0,00543608	0,00000190	0,00000259
B. complexa	0,00000193	0,00030572	0,00006304	0,00031423	0,00021793	0,00036125

Para melhor comparar os resultados, gráficos foram criados e os resultados dos testes foram separados por operação. Na Figura 2, os resultados da operação de inserção são comparados. Considerando que os resultados para o banco PostgreSQL na operação de inserção foram melhores em relação as outras operações, e para o banco MongoDB ocorre justamente o oposto, ainda assim, o banco sem esquema fixo é largamente superior.

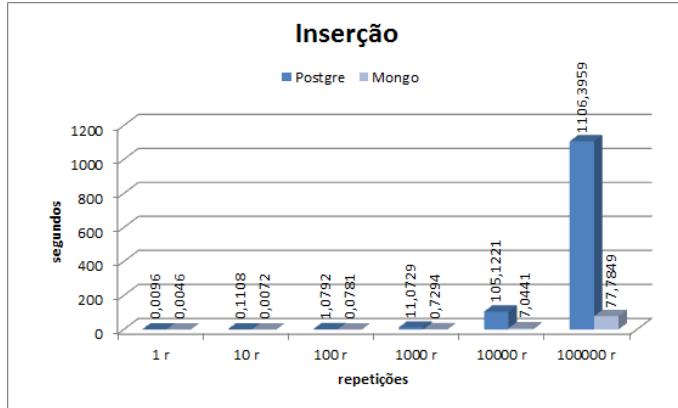


Figura 2. Comparação da Inserção: PostgreSQL e MongoDB

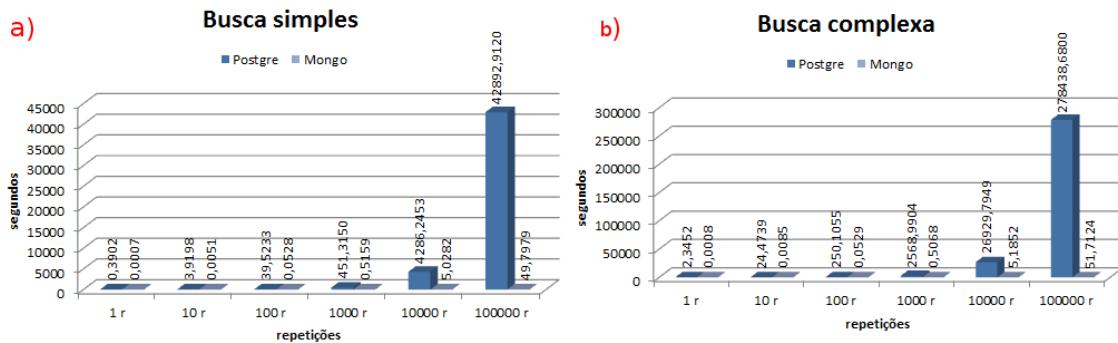


Figura 3. Comparação da Busca Simples (a) e Busca Complexa (b)

Na Figura 3 (buscas simples e complexas), a diferença é ainda maior. O banco MongoDB mostrou melhores resultados na velocidade das consultas de recuperação de

documentos. Já o PostgreSQL, apesar de ser possível otimizar as consultas utilizando um código SQL diferente, demonstrou resultados piores.

7. Conclusão

Utilizando os dois bancos de dados na configuração padrão, sem ajustes de otimização⁵ e baseado no ambiente de testes proposto, podemos concluir que o MongoDB obteve melhores resultados. Isso se dá pelo fato de que os bancos NoSQL terem nascido para suprir a demanda por performance, deixando outros detalhes, como atomicidade, por exemplo, em segundo plano.

Bancos de dados NoSQL e Relacionais utilizam paradigmas diferentes e, por sua vez, possuem finalidades diferentes, mas com o mesmo propósito: persistir dados. Segundo os testes de performance, para uma aplicação com alta carga de consultas à base de dados, como serviços web, por exemplo, o banco MongoDB é uma ótima alternativa. Entretanto, se a aplicação necessita de uma camada de segurança mais robusta, com controle de acessos simultâneos à base de dados, o banco PostgreSQL é a melhor alternativa.

Para trabalhos futuros abre-se um grande leque de possibilidades que vem de encontro com este tema, como: abordar outros bancos Relacionais e NoSQL como MySQL, FireBird e CouchDB, incrementar as buscas utilizando mais critérios e tabelas, fazer testes de backup, replicação e concorrência.

Referências

- [Bigtable 2014] Bigtable (2014). Bigtable web site. <http://research.google.com/archive/bigtable.html>.
- [Boicea et al. 2012] Boicea, A., Radulescu, F., and Agapin, L. I. (2012). MongoDB vs Oracle – Database Comparison. *MongoDB vs Oracle - database comparison*, pages 330–335.
- [Carniel et al. 2012a] Carniel, A., de Aguiar Sa, A., Brisighello, V., Ribeiro, M., Bueno, R., Ciferri, R., and de Aguiar Ciferri, C. (2012a). Análise Experimental de Bases de Dados Relacionais e NoSQL no Processamento de Consultas sobre Data Warehouse. d:113–120.
- [Carniel et al. 2012b] Carniel, A., de Aguiar Sa, A., Brisighello, V., Ribeiro, M., Bueno, R., Ciferri, R., and de Aguiar Ciferri, C. (2012b). Query processing over data warehouse using relational databases and nosql. In *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, pages 1–9.
- [DB-Engines 2014] DB-Engines (2014). Db-engines web site. <http://db-engines.com/en/ranking>.

⁵Há várias maneiras de otimizar consultas aos bancos. O MongoDB tem uma área exclusiva [MongoDB 2014b] onde são indicados métodos como o uso de *index* e *capped collections* entre outros recursos para a realização da otimização. Já o PostgreSQL possui um recurso chamado *Generic Query Optimizer* [PostgreSQL 2014b], que possibilita melhorar a performance de *queries* que possuem muitos *joins*. Entretanto, o objetivo do teste é realizar a comparação dos bancos de dados em seu estado inicial, sem modificações de configuração.

- [Diana and Gerosa 2010] Diana, M. D. and Gerosa, M. A. (2010). NOSQL na Web 2.0: Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0.
- [Dynamo 2014] Dynamo (2014). Dynamo web site. <http://aws.amazon.com/pt/dynamodb/>.
- [Fowler 2012] Fowler (2012). Martin Fowler's talk from the GOTO Aarhus Conference 2012. http://www.youtube.com/watch?v=qI_g07C_Q5I.
- [Fowler 2010] Fowler, M. (2010). *Domain-Specific Languages*.
- [Li and Manoharan 2013] Li, Y. and Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. pages 15–19.
- [Litt et al.] Litt, G., Thompson, S., and Whittaker, J. Improving performance of schemaless document storage in PostgreSQL using BSON.
- [Matthes and Orend 2010] Matthes, F. and Orend, K. (2010). Analysis and Classification of NoSQL Databases and Evaluation of their Ability to Replace and Object-relational Persistence Layer.
- [Microsoft 2014] Microsoft (2014). Microsoft sql server web site. <https://www.microsoft.com/en-us/sqlserver/default.aspx>.
- [MongoDB 2014a] MongoDB (2014a). Mongodb web site. <http://mongodb.org>.
- [MongoDB 2014b] MongoDB, D. (2014b). Mongodb documentation web site. <http://docs.mongodb.org/manual/administration/optimization/>.
- [Oracle 2014] Oracle (2014). Oracle web site. <http://www.oracle.com/technetwork/indexes/downloads/index.html>.
- [Parker et al. 2013] Parker, Z., Poe, S., and Vrbsky, S. V. (2013). Comparing NoSQL MongoDB to an SQL DB. *Proceedings of the 51st ACM Southeast Conference on - ACMSE '13*, page 1.
- [PostgreSQL 2014a] PostgreSQL (2014a). Postgresql web site. <http://postgresql.org>.
- [PostgreSQL 2014b] PostgreSQL, D. (2014b). Postgresql documentation web site. <http://www.postgresql.org/docs/9.1/static/geqo.html>.
- [Pritchett 2008] Pritchett, D. (2008). Base: An acid alternative. *Queue*, 6(3):48–55.
- [Psycopg 2014] Psycopg (2014). Psycopg web site. <http://initd.org/psycopg/docs/index.html>.
- [Pymongo 2014] Pymongo (2014). Pymongo web site. <http://api.mongodb.org/python/current/index.html>.
- [Python 2014] Python (2014). Python web site. <http://python.org>.
- [Timeit 2014] Timeit (2014). Timeit web site. <http://docs.python.org/2/library/timeit.html>.
- [Ubuntu 2014] Ubuntu (2014). Ubuntu web site. <http://www.ubuntu.com/>.
- [Vogels 2009] Vogels, W. (2009). Eventually consistent. *Commun. ACM*, 52(1):40–44.

Uma Metodologia Agile ROLAP para Implantação de Ambientes de Inteligência de Negócios

Elielson B. de Souza¹, André L. Andrade Menolli¹, Ricardo G. Coelho¹

¹Centro de Ciências Tecnológicas – Universidade Estadual do Norte do Paraná (UENP)
Caixa Postal 261 – 86.360-000 – Bandeirantes – PR – Brasil

{elielson,menolli,rgcoelho}@uenp.edu.br

Abstract. Deploying business intelligence environments allow organizations to make decisions that help their development. These environments are constantly changing due to trends in the market, which makes traditional technologies having difficulties in meeting the needs of business. This way, it was developed a methodology that enables the implementation of such environment in a quickly way, allowing to apply adjustments that comes from the market.

Resumo. A implantação de ambientes de inteligência de negócio permitem que as organizações tomem decisões que auxiliem no seu desenvolvimento. Estes ambientes sofrem constantes mudanças devido as evoluções sofridas pelo mercado, o que faz com que tecnologias tradicionais tenham dificuldades em atender as necessidades das empresas. Assim, foi desenvolvida uma metodologia que propicia a implantação de tal ambiente de forma rápida, permitindo que adaptações oriundas do mercado sejam aplicadas.

1. Introdução

Inteligência de Negócio ou *Business Intelligence* (BI) é o conjunto de tecnologias orientadas a disponibilizar informação e conhecimento para as empresas, permitindo uma melhor visualização do que está ocorrendo, o que contribui para tomadas de decisões mais assertivas [Machado 2010]. As informações disponibilizadas por estas tecnologias são oriundas dos dados que as próprias empresas geram na realização de tarefas corriqueiras, sendo que esses dados são organizados permitindo a análise mais eficiente da informação gerada.

Os principais objetivos do BI é permitir que os dados possam ser acessados de forma interativa, proporcionando assim a sua manipulação e fornecendo aos gerentes e analistas de negócio a capacidade de realizar a análise adequada das informações geradas pela própria organização [Turban et al. 2009]. Dentre as ferramentas e tecnologias utilizadas pelos sistemas de BI destacam-se os *Data Warehouse* (DW) e as ferramentas *On-Line Analytical Processing* (OLAP), que surgiram no começo dos anos 90, como novas ferramentas que foram a base dos sistemas de BI [Shim et al. 2002].

Segundo [Kimball 2004] DW é o processo de transformação dos dados obtidos de sistemas legados e de bancos de dados transacionais que ficam organizados sob um formato compreensível ao usuário, auxiliando na tomada de decisão. Um DW é considerado um armazém de dados que tem como objetivo auxiliar a tomada de decisão dentro de uma empresa. Esse armazém de dados possui algumas características que o diferencia das outras formas de análise dos dados, eles são: orientados por assunto,

variantes no tempo, integrados e não voláteis [Machado 2010].

Os dados que são armazenados no DW seguem a modelagem dimensional, pois ela possibilita a criação de um modelo mais simples sendo entendido facilmente pelos analistas de negócio. O modelo dimensional possui em sua estrutura dois tipos de tabelas, uma representa as dimensões e a outra representa os fatos, os fatos se relacionam com as dimensões e assim permitem obter informações que auxiliam na tomada de decisão.

A concepção de um ambiente de BI ocorre por meio de execução de algumas etapas, mas dependendo da arquitetura utilizada pode-se eliminar algumas etapas que deixam de serem necessárias. Os dados provenientes ao ambiente de BI são obtidos de fontes externas como de bases de dados *On-line Transaction Processing* (OLTP), esses dados são então armazenados em uma nova base de dados, necessitando a execução do processo de *Extract, Transform and Load* (ETL). Segundo [Kimball 2004] um sistema de ETL devidamente projetado extraí os dados dos sistemas de origem, reforça a qualidade dos dados e padrões de consistência, conforma os dados de forma que fontes distintas possam ser usadas juntas e, finalmente, oferece dados em um formato de apresentação pronto para que os desenvolvedores de aplicativos possam construir aplicações e possibilita que os usuários finais tomem decisões.

A execução do processo de ETL manipula os dados de origem armazenando-os no DW, em um formato que permite a análise dos dados pelos analistas de negócio por meio de ferramentas OLAP. OLAP é uma forma de se analisar grandes volumes de dados sobre múltiplas perspectivas, sendo amplamente utilizado em ambientes de BI por possibilitar a análise rápida dos dados gerados pela implantação de tal ambiente. De acordo com [Machado 2010] ferramentas OLAP são aplicações às quais os usuários finais têm acesso para extrair os dados de suas bases e construir os relatórios capazes de responder às suas questões gerenciais.

No entanto, a implantação de um ambiente de BI, por meio da construção de um DW é um processo que demanda muito tempo e recurso para sua elaboração. As empresas desejam obter respostas rápidas aos seus investimentos, por isso, novas metodologias, tais como *Agile BI* foram propostas para tentar facilitar esta implantação. Entretanto, essas metodologias possuem diversas restrições, tais como utilizar ferramentas OLAP específicas. Assim, visando facilitar a implantação do BI, e ao mesmo tempo permitir que ferramentas de BI já existentes sejam utilizadas, é proposta uma nova metodologia, que visa agilizar o processo e reduzir os custos.

O restante deste artigo está organizado como segue: na Seção 2 são abordadas as diferentes arquiteturas disponíveis para a implantação de um ambiente de BI, bem como suas vantagens e desvantagens; na Seção 3 é apresentada a metodologia proposta para implantação de ambientes de BI; na Seção 4 são descritos os resultados obtidos com a nova proposta; na Seção 5 as conclusões do trabalho são apresentadas.

2. Trabalhos Relacionados

É possível projetar um ambiente de BI seguindo diferentes arquiteturas, existem vários fatores que determinam qual a melhor tecnologia a escolher, como o volume de dados, o capital a ser investido, o tempo de implantação, entre outros.

Um DW global é uma arquitetura que visa atender todas as necessidades da empresa, sua capacidade de suporte a tomada de decisão fica disponível para todos os

setores da empresa. Essa arquitetura é composta por *Data Marts* (DMs) que podem ser acessados por toda a empresa, eles podem ficar unidos em uma única instalação física ou serem distribuídos. A arquitetura global está dividida em 5 fases, como mostrado na Figura 1, e apresenta a vantagem de permitir o acesso à visão corporativa dos dados, porém sua construção tem alto custo de implementação e um longo tempo de desenvolvimento [Inmon 1997].



Figura 1: Fases da Arquitetura de Data Warehouse Global

A arquitetura de um DW com *Data Marts* Independentes tem características diversas da arquitetura Global. Esta não possui foco corporativo, pois os DMs são construídos sem apresentarem relação uns com os outros, atendendo a cada setor específico da organização, como ilustrado na Figura 2. Essa arquitetura implica em *Data Marts stand alone* controlados por um grupo específico de usuários atendendo somente às necessidades específicas e departamentais, é a preferida entre os fornecedores de software para consulta de informações por ser isolada [Machado 2010]. Os DMs independentes requerem as mesmas técnicas para implementação de uma arquitetura global, sua implementação é mais rápida, porém não possui muita integração corporativa o que acaba por não permitir uma visão global do negócio [Machado 2010].

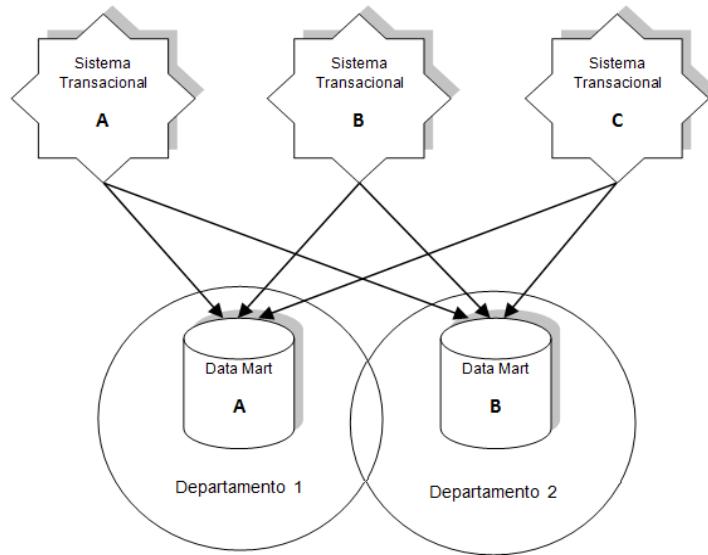


Figura 2: Arquitetura de Data Marts Independentes

Os DMs também podem ser construídos de forma incremental, neste tipo de arquitetura tem-se as características de um DW global, onde sua construção é feita de modo incremental. Segundo [Clemes 2001], nesta abordagem, os requisitos são levantados de forma global, os DMs que serão construídos são identificados, e é definida a maneira como serão integrados. A partir deste momento, cada DM é

implementado completamente até que todos os DM tenham sido implementados, constituindo o DW global da organização. Suas principais vantagens são a apresentação dos resultados de forma rápida, os mecanismos de extração são projetados uma única vez e sua arquitetura permite uma visão global dos dados, porém sua implementação exige a criação de políticas que determinam a sequência de implementação dos DMs e um maior controle no nível de granularidade dos dados [Sell 2001]. A Figura 3 mostra o processo de desenvolvimento dessa abordagem que se constitui de vários ciclos até o completo desenvolvimento do ambiente de BI.

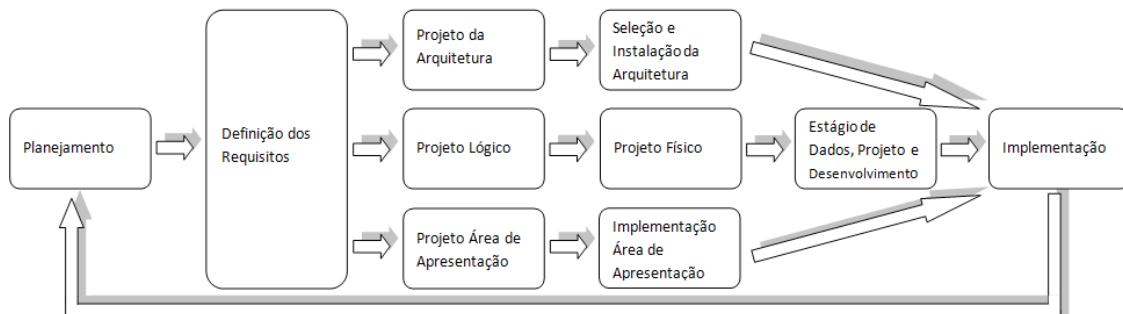


Figura 3: Fases da Arquitetura de Incremental

Um DW construído seguindo a arquitetura de DMs integrados e incrementais permite que seja adicionado um novo DM ao ambiente de BI, ou até mesmo que um DM já existente seja modificado para atender a necessidade da organização. Segundo [Menolli 2006], a principal característica desta arquitetura é a substituição de fontes de vários formatos para uma base de dados padronizada, facilitando a integração de dados. Essa arquitetura está dividida em cinco camadas, desse modo é possível ter independência entre as camadas, ocultando detalhes de implementação entre uma camada e outra. As cinco camadas definidas pela arquitetura são [Menolli 2006]:

- Camada de Fonte de Dados
- Camada de ETL
- Camada de Área de *Staging*
- Camada de *Data Warehouse*
- Camada de Área de Análise

Além das arquiteturas convencionais de desenvolvimento de projetos de BI, recentemente surgiu a arquitetura ágil, que procura oferecer um método mais simples para a elaboração de ambientes de BI. Essa arquitetura ganhou o nome de *Agile BI* e como o próprio nome diz, ela se trata de uma metodologia rápida, de baixo custo e flexível, que visa reduzir o tempo de construção de um ambiente de BI tradicional [Logix, 2012].

As ferramentas *Agile BI* permitem que os dados sejam consultados diretamente das bases transacionais da empresa (Figura 4), eliminando grande parte do processo de implementação efetuado por um ambiente convencional.

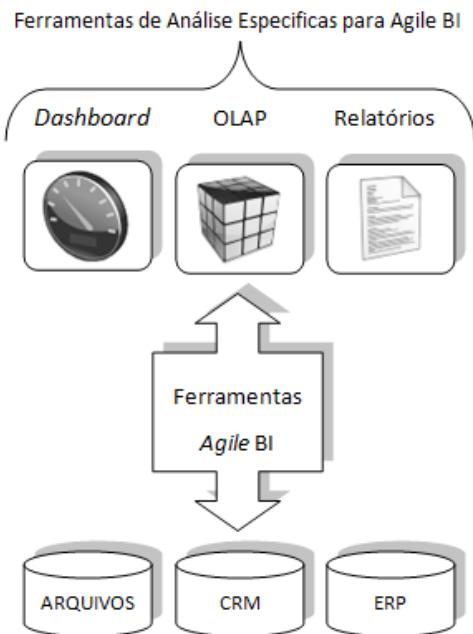


Figura 4: Arquitetura Agile BI
Adaptado de Sandhill (2014)

Essa abordagem defende, que como principais características o projeto de DW tradicional é lento, caro e inflexível e com a implementação do *Agile BI*, as organizações passam a ser capazes de responder às rápidas mudanças do mercado, obtendo informações em um prazo mais curto [Yellowfin 2010]. No entanto, o grande problema desta metodologia é que as ferramentas de análise de dados precisam ser construídas junto ao projeto, ou seja, não é possível utilizar as ferramentas de análise de dados para ambientes de DW tradicionais, pois o formato físico dos dados nesta arquitetura difere do formato dimensional.

3. Metodologia Proposta

Neste trabalho é proposta uma nova metodologia que visa amenizar alguns dos problemas encontrados na tecnologia *Agile BI*, principalmente no que se refere a utilização de ferramentas OLAP construídas para ambientes de BI tradicionais. Estas ferramentas dispõem de inúmeros recursos que auxiliam na análise dos dados e que poderiam contribuir para um desempenho ainda melhor da tecnologia ágil.

O método denotado de *Agile ROLAP* tem por objetivo permitir a implantação de forma ágil de ambientes de BI que utilizem bancos de dados relacionais, e ao mesmo tempo permitam a utilização de ferramentas OLAP projetadas para ambientes tradicionais por meio de um servidor ROLAP.

A metodologia *Agile ROLAP* visa permitir que empresas de pequeno porte possam utilizar dessa tecnologia para auxiliar na tomada de decisões pertinentes ao seu negócio, essas empresas devem possuir seus dados integrados em um único banco de dados, ou seja, os dados gerados pelos sistemas operacionais da organização devem concentrar-se em um único repositório.

Essa metodologia dispensa o processo de ETL realizado nas metodologias supracitadas, pois não é necessária a construção de uma nova base de dados. Os dados

são consultados diretamente das bases transacionais da organização. Estima-se que mais de 1/3 do custo e do tempo são gastos com o processo de ETL, gerados por problemas na extração, transformação ou na limpeza dos dados e que podem consumir até 80% do tempo gasto no desenvolvimento de um projeto de BI [Menolli 2006]. A possibilidade de se utilizar ferramentas OLAP convencionais permite que a organização tenha a possibilidade de escolher ferramentas que se adequem as suas necessidades, não sendo necessária sua implementação, o que agiliza o processo de implantação da tecnologia.

3.1. Arquitetura

A arquitetura do *Agile ROLAP* procura utilizar em sua implementação os recursos já disponíveis no mercado, procurando reduzir os custos e principalmente o tempo necessário para que seja possível sua utilização pelos analistas de negócios. A Figura 5 ilustra a arquitetura, que possui alguns elementos que são essenciais para o seu funcionamento, como as bases de dados OLTP, o servidor Mondrian e as ferramentas de análise dos dados.

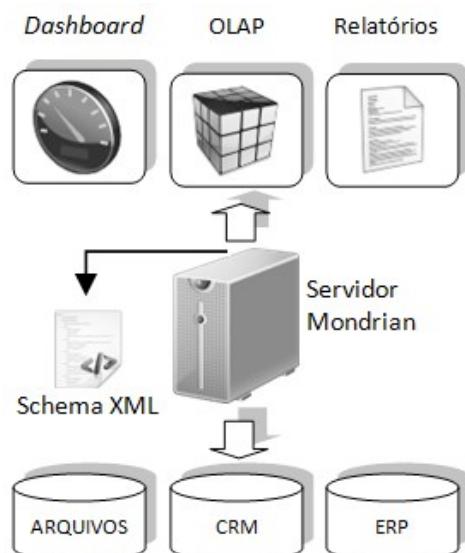


Figura 5: Arquitetura Agile ROLAP

Os dados utilizados pelo ambiente de BI são obtidos das fontes de dados da própria empresa, essas fontes permitirão obter os dados necessários para prover aos analistas de negócios as informações que auxiliarão na tomada de decisão. Os dados são acessados pelo servidor Mondrian que realizará as consultas necessárias para responder as questões levantadas pelos analistas de negócios, feitas por meio das ferramentas de análise dos dados.

O Mondrian é um servidor OLAP escrito em JAVA que recebe consultas na linguagem *multi-dimensional expressions* (MDX) e converte essas consultas para a linguagem SQL, fazendo assim a obtenção dos dados através de bases de dados que são gerenciadas por SGBDs relacionais, por fim o servidor retorna a consulta solicitada em um formato multidimensional permitindo a análise dos dados sobre vários ângulos [Hyde 2006].

Para que o servidor consiga converter as consultas em MDX para SQL é necessário que o mesmo leia um arquivo conhecido como *schema XML*, esse arquivo

realiza o mapeamento dos dados que estão armazenados na forma relacional para os dados que devem ser mostrados na forma dimensional.

O *schema xml* trabalha com a representação de dois modelos, o lógico que trata da representação multidimensional utilizada pelas consultas em MDX como os cubos, dimensões, hierarquias, níveis e membros e o modelo físico que trata da representação dos dados presentes no banco de dados, realizando assim o relacionamento entre o modelo físico e o lógico [Hyde 2011].

3.2. Ambiente Agile ROLAP

O processo de implementação de um ambiente de BI seguindo a metodologia *Agile ROLAP* se diferencia das arquiteturas mencionadas anteriormente, pois neste caso os dados são acessados diretamente da base OLTP da organização por intermédio de um servidor ROLAP.

Os dados que serão utilizados não passam pelo processo de limpeza, eles permanecem da forma com que os sistemas operacionais da organização os gerou, o ambiente de BI somente usufruirá desses dados, da forma com que estão armazenados e organizados, desse modo para que seja possível realizar a análises sob uma perspectiva temporal é necessário que a base de dados mantenha dados históricos gerados pela organização.

O processo de implantação do ambiente *Agile ROLAP* é constituído de 6 fases ilustradas na Figura 6, que permitirão a utilização dos dados da organização por meio das ferramentas de análise dos dados utilizados por ambientes tradicionais.



Figura 6: Fases da arquitetura Agile ROLAP

O primeiro passo a ser realizado é o levantamento dos dados e dos requisitos, assim como nas outras arquiteturas é necessário conhecer o que a organização pretende com a utilização da tecnologia, quais as áreas que ela deseja estudar, além de outras informações, assim é possível implementar um ambiente que atenda às necessidades da empresa.

Conhecendo as necessidades da empresa é necessário obter o acesso as fontes de dados que serão utilizadas pelo *Agile ROLAP*, isso permitirá uma análise mais detalhada dos dados que serão utilizados, sendo possível identificar quais os requisitos solicitados poderão ser atendidos.

Assim como no processo de DW, é necessário criar uma ou mais dimensões tempo, que conterá todas as informações necessárias para permitir a análise temporal. A dimensão de tempo possui uma importância acentuada em todo o modelo de dados de um ambiente de BI, estando presente em qualquer que seja ele, pois quando se deseja analisar uma fato, não se interessa guardar informações sobre cada transação, mas sobre as transações em um espaço de tempo definido [Machado 2010]. Nessa dimensão é necessário que se faça a criação de uma nova tabela na base OLTP da organização, essa

nova tabela armazenará informações temporais que consigam permitir a análise temporal sub uma perspectiva diária.

O próximo passo desse processo é a criação das outras dimensões que serão utilizadas na análise dos dados, elas fornecerão informações descritivas sobre as atividades da organização. Diferente das dimensões de tempo, essas dimensões são organizadas seguindo a estrutura da base de dados de modo que se consiga obter as informações necessárias a dimensão.

Nesta fase é necessário definir qual será a tabela principal que a dimensão utilizará e quais informações ela dispõe. Caso não seja possível extrair os dados necessários de uma única tabela, são realizadas junções (*joins*) entre a tabela principal e outras tabelas que relacionam com esta. Este relacionamento é feito de forma lógica por meio do mondrian *schema*, mantendo a estrutura física original, mas sendo visualizada na ferramenta de BI de forma desnormalizada.

Os fatos, assim como a dimensão de tempo, precisam de uma estrutura própria para que armazenem as atividades da organização. Porém os fatos como são originados das tabelas das bases transacionais não necessitam a criação de uma nova tabela, neste caso se utiliza a criação de *views*. Cada *view* criada corresponde a um fato, lá são armazenados os relacionamentos com as dimensões e as medidas referentes ao fato, sua criação ocorre por meio da execução de uma consulta constituída de *joins* que acessam as tabelas da base OLTP e recuperam as informações necessárias que permitirão o acesso as dimensões e para a criação das medidas.

A criação dos fatos por meio de *views* permitem que o tempo de execução de uma consulta seja reduzido, pois os fatos são originados de diversas tabelas o que acaba exigindo um trabalho extremo do SGBD, pois já são realizados *joins* na constituição das dimensões.

O último passo desse processo é a criação do *schema XML* que será utilizado pelo servidor Mondrian para fazer o mapeamento do modelo lógico para o físico, são lá que a estrutura das dimensões e fatos ficam definidas bastando apenas serem interpretadas pelo servidor.

Finalizado esse processo, o ambiente de BI já pode ser utilizado pelos analistas de negócio por meio de ferramentas de análise dos dados disponíveis no mercado que a organização identificou como sendo a melhor para atender suas necessidades.

4. Resultados

A utilização da metodologia *Agile ROLAP* proporciona os benefícios já consagrados pelas tecnologias ágeis, como o *Agile BI*, mas além disso proporcionam a possibilidade da utilização de ferramentas tradicionais, o que aumenta consideravelmente a variedade de ferramentas disponíveis no mercado.

Um estudo realizado em uma base de dados relativamente simples mostrou que a implantação de um ambiente utilizando a metodologia *Agile ROLAP* é atrativa para empresas de pequeno porte, que não possuem uma base de dados volumosa e que deseja utilizar dessa tecnologia para auxiliar no desenvolvimento da organização.

A base de dados utilizada, chamada de Pagila, retrata os dados provenientes de um sistema OLTP de locação de filmes, ela está disponível sob a licença *Berkely Software Distribution* (BSD). O estudo de caso desenvolvido procurou analisar as

locações de filmes registradas na base, procurando responder a questões como o número total de locações nos fins de semana, os gêneros dos filmes preferidos pelos clientes, entre outros.

Essa base foi desenvolvida por Mike Hillyer membro da equipe de desenvolvimento do MySQL com o objetivo de fornecer um esquema padrão para ser usados como exemplo em livros, tutoriais, artigos, entre outros, sua base contém mais de 45.000 registros e é composta por 21 tabelas que procuram armazenar dados como os autores dos filmes, as locações realizadas, os pagamentos, etc [PgFoundry 2008].

Foram criadas seis dimensões convencionais, duas dimensões de tempo e um fato como na Figura 7, o tempo de estudo da base não foi levado em consideração, pois é o mesmo para qualquer arquitetura utilizada.

Enquanto que na criação do ambiente de BI tradicional provida de um DW foi utilizado a ferramenta kettle, sendo possível realizar o processo de ETL construindo-se uma nova base de dados, no ambiente *Agile ROLAP* a definição das dimensões foi feita direto no *schema XML* e a criação do fato no próprio SGBD.

Neste estudo pode-se utilizar a mesma ferramenta de análise em ambos os ambientes, cumprindo o objetivo da tecnologia. Além disso, a metodologia permite que por exemplo, seja possível utilizar conceitos como *role play dimensions* que é a capacidade de se utilizar uma dimensão várias vezes em um fato, de modo a permitir que por exemplo sejam necessário a existência de somente uma dimensão de tempo em um fato que necessite representar os dados sob duas perspectivas temporais.

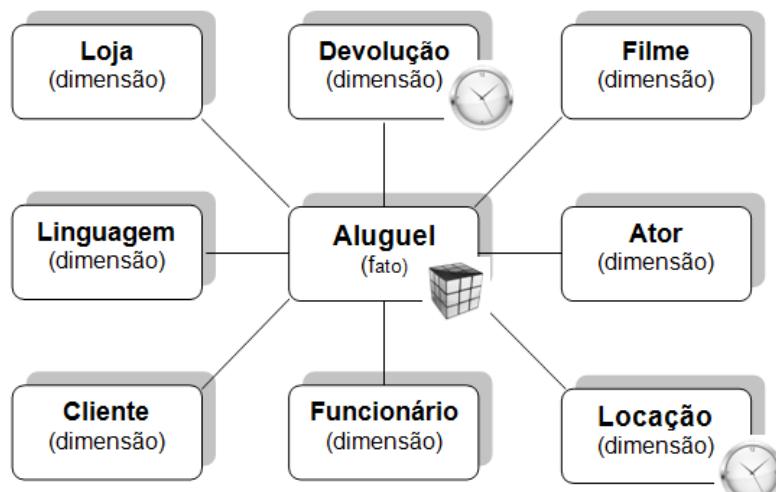


Figura 7: Dimensões e fato criados no estudo realizado

5. Conclusões

As constantes mudanças que ocorrem no mercado fazem com que ambientes de BI tenham que se adaptar para atender as necessidades da organização. Os ambientes tradicionais possuem dificuldade em se adaptarem, pois são complexos, o que não acontece com a metodologia *Agile ROLAP*, que permite que o processo de implantação de tal ambiente seja mais simples, rápido e adaptável.

Uma nova alternativa para implantação de ambientes de BI dentro das organizações permite que um maior número de empresas possam utilizar desta tecnologia a fim de auxiliar na tomada de decisões, além disso pretendesse construir um componente para a ferramenta Kettle com o intuito de automatizar o processo do *Agile* ROLAP na implantação de ambientes de BI.

Referências

- Clemes, M. (2001) “Data warehouse como suporte ao sistema de informações gerenciais em uma instituição de ensino superior: estudo de caso na UFSC.”, Dissertação de Mestrado – Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção, Florianópolis.
- Hyde, J. (2006) “Mondrian Documentation: Arquiteture”, Pentaho, Disponível em: <<http://mondrian.pentaho.com/documentation/architecture.php>>.
- Hyde, J. (2011) “Mondrian Documentation: MDX Specification”, Pentaho, Disponível em: <<http://mondrian.pentaho.com/documentation/mdx.php>>.
- Inmon, W. H. (1997) “Como Construir o Data Warehouse”, Rio de Janeiro.
- Kimball, R., Caseta, J. (2004) “The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data”, John Wiley & Sons.
- Logix. (2012) “What is Agile BI”, Disponível em <<http://www.nathean.com/index.php/what-is-agile-bi/>>.
- Machado, F. N. R. (2010) “Tecnologia e Projeto de Data Warehouse: uma visão multidimensional”, 5 ed., Érica, São Paulo.
- Menolli, A. L. A. (2006) “A Data Warehouse Architeture in Layers for Science and Technology”, Proceedings of the Eighteenth International Conference on Software Engineering Knowledge Engineering (SEKE'2006), San Francisco, CA, USA.
- PgFoundry. (2008) “A collection of sample databases for PostgreSQL”, Disponível em: <<http://pgfoundry.org/projects/dbsamples/>>
- Sandhill. (2014) “BI Ready Data Warehouse Automation”, Disponível em: <<http://www.sandhillconsultants.com/BIReadyProduct.asp>>
- Sell, D. (2001) “Uma arquitetura para distribuição de componentes tecnológicos de sistemas de informações baseado em data warehouse”, Dissertação de Mestrado – Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção, Florianópolis.
- Shim, J. P., Warkentin, M., Courtney, J., Power, D. J., Sharda, R. e Carlsson, C. (2002) “Past, Present, And Future Of Decision Support Technology”, Decision Support System, V. 33, N. 2, P. 111-126.
- Turban, E., Sharda, R., Aronson, J. E. and King, D. (2009) “Business Intelligence: um enfoque gerencial para a inteligência do negócio”, Grupo A.
- Yellowfin. (2010) “Making Business Intelligence Easy: Write Paper Agile Business Inteligence”, Disponível em: <<http://www.yellowfinbi.com/Document.i4?DocumentId=97458>>.

Abordagem de suporte a transação através de consulta *HiveQL*

**Juliano Gomes da Silveira, Tobias Stifft, Daiane Hemerich, Duncan Dubugras
Alcoba Ruiz**

Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

Faculdade de Informática (FACIN)

Caixa Postal 14.19 – 90.619-900 – Porto Alegre – RS – Brazil

juliano.pro@gmail.com, stifft@gmail.com, daiane.hemerich@pucrs.br,
duncan@pucrs.br

Abstract. *We propose in this paper an approach that simulates support for transactions on a Hadoop-Hive environment through a HiveQL query mechanism of temporal series. Such proposal has as premise to get round the limitation of Hadoop-Hive environment regarding UPDATE and DELETE operations, given that data persisted on Hadoop Distributed File System (HDFS) can be provided by a relational database.*

Resumo. *Propomos neste trabalho uma abordagem que simula o suporte a transações em um ambiente Hadoop-Hive através de um mecanismo de consulta HiveQL de série temporal. Tal proposta tem como premissa contornar a limitação do ambiente Hadoop-Hive quanto às operações de UPDATE e DELETE, uma vez que os dados persistidos no Hadoop Distributed File System (HDFS) podem ser providos por banco de dados relacional.*

1. Introdução

A plataforma *Apache Hadoop* [1] consolidou-se como uma sofisticada ferramenta para trabalhar com *Big Data*. Seu sistema de armazenamento distribuído (*Hadoop Distributed File System - HDFS*) [2][3] viabiliza performance no manejo de grandes conjuntos de dados em hardware de baixo custo. Além disso, o HDFS conta com recursos que o torna tolerante a falhas. O modelo de programação de *Hadoop* é inspirado em *MapReduce* [4], sendo este orientado para o processamento de grandes volumes de dados (estruturados ou não) sob uma arquitetura distribuída. Tal modelo divide o trabalho em pequenas tarefas independentes que são processadas em paralelo. Comumente, fazem uso de *Hadoop* sistemas de análise de logs, armazenamento e recuperação de dados oriundos de sistemas de sensores ou de identificação por radiofrequência (RFID), entre outras aplicações.

Dentre os diversos subprojetos associados à plataforma *Hadoop*, encontra-se o *Hive*, que foi inicialmente desenvolvido pelo *Facebook* e posteriormente admitido como um projeto de código aberto da *Apache*. O *Hive* provê um ambiente de abstração de

armazenamento e acesso a dados, focado em aplicações de *Data Warehouse*. Esta ferramenta é operada através de uma linguagem de alto nível, baseada em SQL ANSI (chamada *HiveQL*) e mantém abstrações equivalentes aos bancos de dados relacionais, tais como tabelas, atributos, registros, etc. *Hive* trabalha sobre os dados persistidos no HDFS e seus comandos *HiveQL* são convertidos em operações *MapReduce*, destacando-se as operações de sumarização, consulta e análise dos dados. Embora seja baseado em SQL, algumas extensões não são compatíveis com o *HiveQL*, como o suporte a transações e subconsultas.

Comumente, sistemas de banco de dados relacionais fornecem dados para o *Hadoop*, e, ainda que não seja um cenário típico de *Big Data*, existem aplicações que exigem suporte a transações, mesmo que seja apenas para exclusão e atualização de registros. Nesse ponto *Hadoop-Hive* é limitado, pois de forma nativa não admite operações de *DELETE* e *UPDATE* de registros [6], operações essas tomadas como triviais em ambientes de bancos de dados relacionais. Examinando a documentação, encontramos alguns subterfúgios para contornar esta limitação, como o uso de partições. Nesse caso, no momento do projeto, a tabela deve prever um particionamento de dados em seu nível elementar e, quando necessário, por uma operação de *INSERT OVERWRITE*, a partição é sobrescrita. Embora a atualização de partição funcione, esse recurso se torna paliativo, uma vez que o particionamento elementar gera fragmentação desnecessária. Para lidar com essa situação, neste trabalho propomos uma abordagem de suporte a transações para o *Hadoop-Hive*. Tal abordagem baseia-se em um mecanismo simplificado de armazenamento e consulta *HiveQL* que retorna os registros vigentes, ou seja, suprimindo aqueles que no SGDB de origem dos dados foram sobrescritos através de uma operação de *UPDATE* ou excluídos por uma operação de *DELETE*, sem a necessidade da utilização de artifícios adicionais, como o particionamento de tabela. Basicamente, a abordagem aqui demonstrada busca contornar a limitação de suporte transacional do *Hive* através de uma série temporal. Para tanto, primeiramente definimos algumas condições que os dados devem sustentar para que o mecanismo de suporte a transação possa operar. Após isso, detalhamos as operações da consulta *HiveQL*. Por fim, demonstramos alguns exemplos e experimentos. Ao longo do texto, para facilitar a compreensão, usaremos o acrônimo ASTC (abordagem para suporte a transação por consulta).

2. Abordagem de suporte a transação por consulta (ASTC)

Geralmente, *Hadoop* é integrado com SGDBs relacionais, tanto como receptor quanto fornecedor de dados. Para tanto, *Hive* conta com interfaces que viabilizam esse trabalho, como, por exemplo, os componentes de JDBC e ODBC. Porém, em virtude de seu foco, *Hadoop* possui algumas características nativas que contrastam com os SGDBs tradicionais [6]. Uma dessas é o suporte a transações, não admitindo atualizações e exclusões de registros.

Embora saibamos que este não é um cenário comum de aplicações de *Big Data*, a limitação da plataforma em processar atualizações e exclusões pode gerar alguns inconvenientes. Para ilustrar, consideremos o seguinte exemplo: uma tabela de SGDB que mantém registros de vendas de comércio eletrônico, onde cada registro representa uma compra, sendo que tal transação possui diversos status ('aguardando pagamento', 'despachado', 'entregue'). Supondo que esses dados sejam integrados com *Hadoop* e nessa integração esteja contemplado o código de venda (chave), valor, status e data de última atualização de status. Nesse caso, admitimos que para cada estímulo de *UPDATE* no SGDB o registro atualizado é encaminhado para o *Hadoop* ou posto em uma fila para posterior processamento em lote. Dada a forma em que esses dados foram concebidos e integrados, uma simples consulta para obter a quantidade de vendas por status pode ser inviável, mesmo usando uma ferramenta de consulta de alto nível como o *Hive*, tendo em vista que, no *Hadoop*, cada atualização incluirá uma nova entrada de dados, mantendo também as antigas entradas de registros de venda (com situação não vigente). Neste exemplo, se os dados fornecessem mais detalhes, como a data de pagamento, despacho e entrega, tornar-se-ia prático construir uma consulta para se analisar o panorama de vendas por status, uma vez que essas datas podem ser operadas pelo *HiveQL*. Porém, nem sempre se tem todos os dados necessários para o trabalho.

A ASTC pode ser usada como um artifício para lidar com a situação descrita acima. Esta abordagem pode ser resumida em três etapas: definição dos dados (onde são delineados alguns critérios aos quais os dados devem atender); redutos de integração (meio por onde os dados devem ser integrados do SGDB para o HDFS); e consulta (mecanismo de suporte a transação). As próximas subseções detalham as etapas.

2.1. Propriedade dos dados

Na ASTC, faz-se necessária a definição de duas propriedades que devem ser aplicadas sobre o SGDB, mais especificamente, pela tabela do SGDB que se deseja integrar com o *Hadoop*, conforme descrito a seguir:

- i. A tabela do SGDB a ser integrada deve possuir uma chave candidata ou algum atributo que garanta a unicidade do registro;
- ii. A tabela do SGDB a ser integrada deve possuir um atributo de série temporal contendo o apontamento da última modificação do registro. No caso de registros que nunca foram atualizados, tal atributo deve conter o apontamento do momento da inserção.

Na sequência deste trabalho, identificamos a chave candidata como atributo KEY e o atributo de série temporal como DAT.

2.2. Redutos de integração

A ASTC define o conceito de integração por dois redutos (reduto de *INSERT/UPDATE* e *DELETE*), sendo esses parte do processo de ETC (extração, transformação e carga de

dados). Esses redutos são o meio de transporte dos dados, podendo ser arquivos de texto, tabelas persistidas no próprio SGDB a ser integrado, ou qualquer outro tipo compatível com *Hadoop*. No caso de arquivo texto, uma opção é utilizar um mecanismo de gatilho externo, para que a cada estímulo de *INSERT*, *UPDATE* e *DELETE* executado no SGDB, uma nova entrada seja inserida no arquivo, sendo na sequência esses arquivos integrados ao HDFS. Se os redutos forem concebidos no próprio SGDB através de tabelas, uma opção é utilizar um gatilho interno, onde cada estímulo gera uma nova entrada para a respectiva tabela e na sequência os dados são integrados ao HDFS via driver ODBC/JDBC, disponível no *Hive*. Neste trabalho usamos a opção de integração via tabela, ou seja, cada reduto é uma tabela persistida no SGDB com o mesmo layout da tabela a ser integrada, onde cada estímulo de *INSERT*, *UPDATE* e *DELETE* é capturado por um gatilho e o registro é inserido no respectivo reduto. Necessita-se, portanto, construir três gatilhos, um para cada comando. O ponto determinante aqui é que a cada operação o registro seja encaminhado para o respectivo reduto, ou seja, uma operação de *INSERT* ou *UPDATE* sobre a tabela a ser integrada deve gerar uma entrada no reduto de *INSERT/UPDATE*. O mesmo vale para as operações de *DELETE*, onde as entradas devem ser escritas sobre o reduto de *DELETE*. A fim de facilitar a compreensão, vamos nomear os redutos como: *RED_INSERT_UPDATE* e *RED_DELETE*. Ao final, os redutos são movidos para HDFS. Neste trabalho, para transporte dos dados utilizamos o componente de ODBC disponível no *Hive*. A Figura 1 ilustra o mecanismo de redutos adotado neste trabalho.

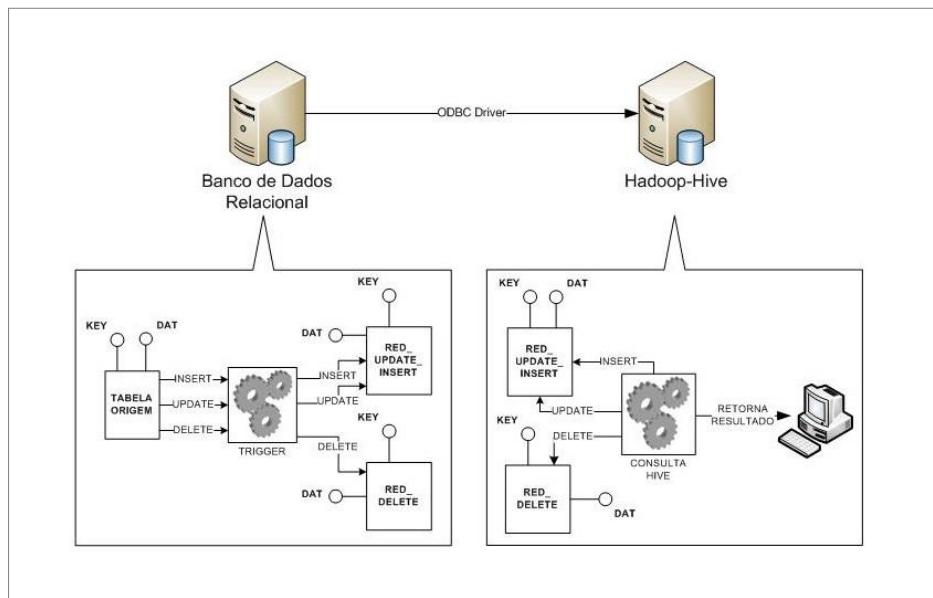


Figura 1. Mecanismo de redutos de integração utilizado neste trabalho.

2.3. Consulta *HiveQL*

A consulta foi elaborada com objetivo de abstrair os registros não vigentes, ou seja, aqueles que foram excluídos ou que foram sobreescritos por atualização. Tal consulta foi construída sobre o *Hive* utilizando apenas operações comuns, como junções, uniões e

operadores lógicos sobre o par key/dat. No Quadro 1 é demonstrada a consulta *HiveQL*. Cada fragmento numerado do código é detalhado na sequência do texto.

```

select red_insert_update.*
  from red_insert_update
  join (select mx_ins_upd.key, mx_ins_upd.dat
         from (select red_insert_update.key, max(red_insert_update.dat) dat
               from red_insert_update
              group by red_insert_update.key) mx_ins_upd) 2.3.1
        left outer join (select red_delete.key, max(red_delete.dat) dat
                          from red_delete
                         group by red_delete.key) mx_del 2.3.2
                           on (mx_ins_upd.key = mx_del.key) 2.3.3
        where mx_del.key is null
          or mx_ins_upd.dat > mx_del.dat) mx 2.3.4      2.3.5
      on (mx.key = red_insert_update.key)
      where mx.dat = red_insert_update.dat
    
```

Quadro 1. Consulta Hive-QL.

2.3.1. Neste trecho RED_INSERT_UPDATE é consultada abstraindo aqueles registros que foram sobrescritos através da operação de UPDATE. Para tanto, se utiliza a função MAX sobre série temporal (DAT), agrupando pela chave (KEY). Com isso, tem-se a situação mais atual de cada chave;

2.3.2. Neste ponto são consultados os registros persistidos em RED_DELETE, ou seja, aqueles que foram excluídos. Porém, há situações em que, segundo a chave, um mesmo registro é excluído mais de uma vez, portanto nesse trecho aplica-se a função MAX sobre a série temporal (DAT), agrupando pela chave (KEY). Desta forma tem-se como resultado o par de atributos (DAT/KEY) da última exclusão;

2.3.3. Esta junção estabelece a relação entre o trecho 2.3.1 (mx_ins_upd) e o trecho 2.3.2 (de mx_del) através da chave (KEY). Neste caso, utiliza-se a junção *left outer join*, pois tal consulta deve retornar todos os registros de mx_ins_upd independentemente se há um respectivo representante em mx_del;

2.3.4. Neste trecho aplica-se a cláusula WHERE sobre os atributos mx_del.key, mx_del.dat e mx_ins_upd.dat. Aqui se iguala a nulo o atributo mx_del.key, para eliminar os registros que foram excluídos. Há situações em que um registro excluído pode ser posteriormente reinserido. A cláusula seguinte (mx_ins_upd.dat > mx_del.dat) evita que esses registros sejam desconsiderados;

2.3.5. Este bloco resulta em pares de valores (KEY, DAT) dos registros vigentes.

Na próxima seção, cada um destes trechos de código é executado e comentado sobre uma amostra de dados de exemplo.

3. Exemplo

Para melhor compreender a ação do mecanismo de suporte a transações por consulta, nessa seção vamos usar um exemplo prático, no qual simulamos a ação da consulta *HiveQL* (seção 2.2) sobre as estruturas RED_INSERT_UPDATE e RED_DELETE. Abaixo cada trecho da consulta é executado, comentado e exposto o resultado.

As tabelas abaixo representam os redutos RED_INSERT_UPDATE e RED_DELETE, respectivamente.

Tabela 1. Reduto RED_INSERT_UPDATE

KEY	DAT	AUTHOR	BOOK
1	01/08/2013	KNUTH; DONALD E.	ART OF COMPUTER PROGRAMMING V.1
2	01/08/2013	TANENBAUM; ANDREW S.	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION
3	01/08/2013	STEWART; JAMES	ESSENTIAL CALCULUS
4	01/08/2013	HAN; JIAWEI	DATA MINING CONCEPTS AND TECHNIQUES
1	02/08/2013	KNUTH; DONALD ERVIN	ART OF COMPUTER PROGRAMMING V.1
2	03/08/2013	TANENBAUM; ANDREW S.	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION
2	04/08/2013	TANENBAUM; ANDREW STUART	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION
2	06/08/2013	TANENBAUM; ANDREW STUART	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION

Tabela 2. Reduto RED_DELETE

KEY	DAT	AUTHOR	BOOK
2	02/08/2013	TANENBAUM; ANDREW S.	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION
4	02/08/2013	HAN; JIAWEI	DATA MINING CONCEPTS AND TECHNIQUES
2	05/08/2013	TANENBAUM; ANDREW STUART	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION

Observando os dados de cada reduto, podemos desdobrar a série temporal das transações que ocorreram na tabela do SGDB, conforme abaixo:

- i. Em 01/08/2013 são inseridos os registros identificados como KEY(1, 2, 3, 4);
- ii. Em 02/08/2013 é atualizado o atributo *AUTHOR* do registro KEY(1);
- iii. Em 02/08/2013 os registros KEY(2, 4) são excluídos;
- iv. Em 03/08/2013 volta a ser inserido o registro identificado como KEY(2);
- v. Em 04/08/2013 o registro KEY(2) sofre uma atualização no atributo *AUTHOR*.
- vi. Em 05/08/2013 o registro KEY(2) é excluído novamente;
- vii. Em 05/08/2013 volta a ser inserido o registro identificado como KEY(2).

Quando executado o trecho 2.2.1 da consulta, temos o resultado de pares (KEY/DAT) listado na Tabela 3, onde constam os registros que estão persistidos em RED_INSERT_UPDATE desconsiderando aqueles desatualizados, ou seja, aqueles que segundo o par KEY/DAT não estão vigentes.

Tabela 3. Resultado parcial - consulta 2.2.1

KEY	DAT
3	01/08/2013
4	01/08/2013
1	02/08/2013
2	06/08/2013

Ao executar o trecho 2.2.2 da consulta, tem-se o resultado de pares (KEY/DAT) listado na Tabela 4. Da mesma forma que a consulta 2.2.1, também busca o registro mais atual da série temporal para cada chave, através da função MAX no atributo DAT.

Tabela 4. Resultado parcial - consulta 2.2.2

KEY	DAT
4	02/08/2013
2	05/08/2013

O trecho 2.2.5 da consulta utiliza-se do resultado do trecho 2.2.1 e 2.2.2 para encontrar os pares (KEY/DAT) vigentes. Neste caso, através da junção 2.3.3 e da cláusula 2.2.4 além de buscar os registros mais atuais, faz a diferenciação de exclusões que foram novamente inseridas. A Tabela 5 exibe o resultado do par (KEY/DAT).

Tabela 5. Resultado parcial - consulta 2.2.5

KEY	DAT
3	01/08/2013
1	02/08/2013
2	06/08/2013

Por fim, a consulta completa retorna todos os dados, restringindo os valores através dos pares de valores resultantes do trecho 2.3.5, conforme demonstrado abaixo.

Tabela 6. Resultado da consulta completa

KEY	DAT	AUTHOR	BOOK
3	01/08/2013	STEWART; JAMES	ESSENTIAL CALCULUS
1	02/08/2013	KNUTH; DONALD ERVIN	ART OF COMPUTER PROGRAMMING V.1
2	06/08/2013	TANENBAUM; ANDREW STUART	OPERATING SYSTEMS DESIGN AND IMPLEMENTATION

Na tabela acima verificamos que o registro representado por KEY(4) não foi exibido, pois, segundo RED_DELETE, este registro foi excluído na sequência e não foi inserido novamente. No caso de KEY(2), que também foi excluído, a consulta o preservou no resultado, pois segundo a série temporal houve uma nova inserção após a exclusão. O registro representado por KEY(1) foi retornado pela consulta, porém se observa que haviam dois registros para essa chave, e neste caso a consulta considerou apenas o registro mais atual (DAT = 02/08/2013). O registro identificado por KEY(3) também permaneceu, pois neste não houve atualizações ou exclusões.

4. Testes

A fim de testar o desempenho da abordagem aqui proposta foram realizados alguns experimentos, nos quais foi avaliado o tempo de resposta da ASTC. Para tanto, foram processados 10 conjuntos de dados de diferentes tamanhos, sendo o menor com 1.650.000 registros, crescendo linearmente até o maior, de 16.500.000. Os conjuntos continham registros de inserção, atualização, exclusão e portanto mantiveram-se segregados nos redutos RED_INSERT_UPDATE e RED_DELETE. O reduto RED_INSERT_UPDATE continha todos os registros que foram em algum momento inseridos ou atualizados na tabela de banco de dados origem, ou seja, registros capturados pelos gatilhos de *INSERT/UPDATE*. Da mesma forma, o reduto

`RED_DELETE` continha todos os registros que foram em algum momento excluídos da tabela de banco de dados origem, ou seja, registros capturados pelo gatilho de `DELETE`.

Os experimentos aqui demonstrados foram executados sobre 4 nós virtuais. O sistema operacional utilizado foi *Linux Red Hat 5.5*, sendo cada um dos nós virtuais de 2GB de memória RAM, compartilhando o processador Intel i5-2450M de 4 núcleos de 2.5 GHz. As versões utilizadas de *Hadoop* e *Hive* são 1.1.1 e 0.9.0, respectivamente.

4.1. Conjunto de dados

Todos os conjuntos de dados testados possuíam inserções, atualizações, exclusões e inserções pós-exclusão. Chama-se de registros de inserção pós-exclusão aqueles que segundo o par KEY/DAT foi inserido, excluído e na sequência voltou a ser inserido. O gráfico abaixo demonstra a composição de registros dos 10 conjuntos utilizados.

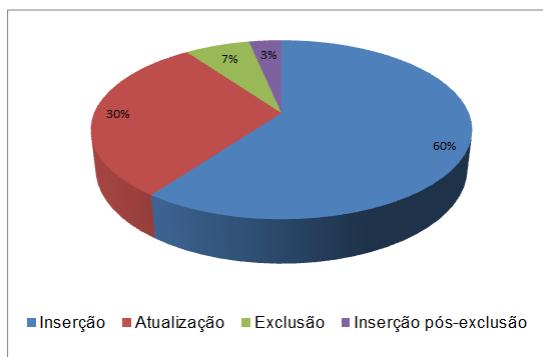


Figura 2. Composição dos registros de teste.

4.2. Consultas de teste

Para cada um dos 10 conjuntos, a consulta utilizando ASTC foi executada, incluindo-se em seu escopo a função `COUNT(*)`. Adicionalmente, para fins de comparação, executamos consultas de contagem sobre os mesmos conjuntos sem o uso da ASTC. A intenção dos testes foi verificar o desempenho da abordagem ASTC para uma consulta de contagem dos registros vigentes frente a execução de uma consulta simples, que considera todos os registros (vigentes e não vigentes). Nos quadros seguintes são demonstradas ambas as consultas.

```

select count(*)
from red_insert_update
join (select mx_ins_upd.key, mx_ins_upd.dat
      from (select red_insert_update.key, max(red_insert_update.dat) dat
            from red_insert_update
            group by red_insert_update.key) mx_ins_upd
     left outer join (select red_delete.key, max(red_delete.dat) dat
                      from red_delete
                      group by red_delete.key) mx_del
      on (mx_ins_upd.key = mx_del.key)
     where mx_del.key is null
       or mx_ins_upd.dat > mx_del.dat) mx
   on (mx.key = red_insert_update.key)
  where mx.dat = red_insert_update.dat

```

Quadro 3. Consulta ASTC.

```
select count(*)
from red_insert_update
```

Quadro 4. Consulta simples.

4.3. Análise dos resultados

Os tempos de consulta de ambas as consultas foram plotados em um gráfico de linhas, para facilitar a análise dos resultados.

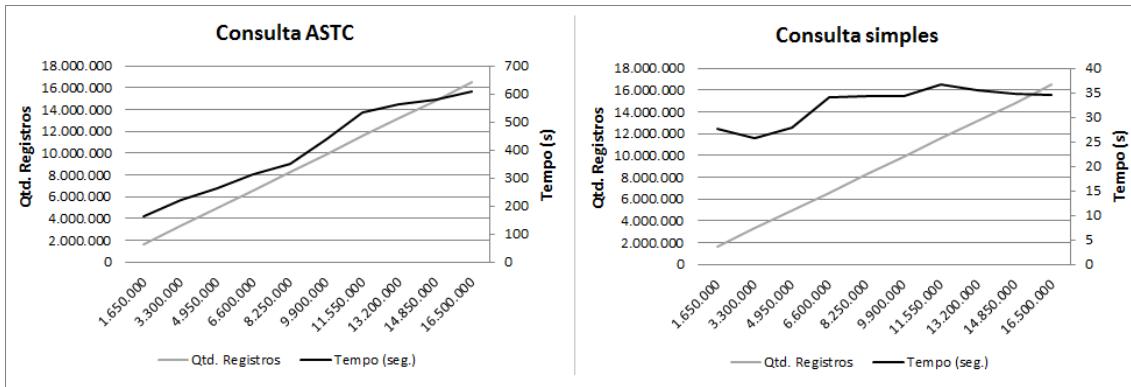


Figura 3. Comparação dos tempos de consulta.

Os gráficos acima possuem a seguinte composição: as escalas do eixo x e do eixo y (esquerdo) representam a quantidade total de registros processados; o eixo y direito representa a escala de tempo de execução; a linha cinza exibe o crescimento da quantidade de registros processados; a linha preta demonstra a curva de tempo das consultas. Ao observar os indicadores exibidos na Figura 3, constatamos que o tempo de resposta da consulta que usa ASTC foi significativamente maior em relação ao tempo de resposta da consulta que não usou ASTC. Entendemos que esse comportamento deu-se em função das sincronizações realizadas pelas relações de junção e a aplicação da função *MAX* sobre o atributo DAT. O indicador de tempo da consulta ASTC permaneceu associado ao crescimento dos registros, com uma leve queda à medida em que a quantidade de registros se eleva. Porém, ao observar o mesmo indicador sem o uso de ASTC, identificamos que a relação entre o crescimento dos dados e o tempo de execução é menos aparente. Logo, considerando o ambiente utilizado neste experimento e os dados processados nos testes, podemos concluir que o tempo de resposta da consulta da ASTC tem uma forte relação com o volume de dados processados.

5. Considerações Finais

A abordagem proposta neste trabalho mostrou-se efetiva na tarefa de simular o comportamento de suporte a transações. Pode ser utilizada em um ambiente *Hadoop-Hive*, onde se deseja obter uma visão das transações vigentes, desconsiderando aquelas transações que foram invalidadas por operações de *UPDATE* ou *DELETE* na origem.

Ao observar os resultados apresentados na subseção 4.3, percebemos a clara queda de desempenho (tempo de resposta) quando empregada a consulta ASTC frente à consulta de contagem simples. Tal queda já era prevista, pois a complexidade da consulta ASTC é significativamente maior em função das suas junções e o uso da operação *MAX*. Porém o tempo de resposta de ASTC permaneceu associado ao crescimento dos dados, sendo que no mesmo indicador da consulta simples essa associação se manteve fraca, ou seja, a complexidade da consulta ASTC se manteve linear ao crescimento dos dados, diferentemente da consulta simples onde a linha nos indica uma escala mais logarítmica do que linear (Figura 3). Esse resultado nos convida a progredir essa pesquisa, investigando as causas deste crescimento linear.

Referências

- [1] The Apache Software Foundation, Apache-Hadoop, 2013. Disponível em: <http://hadoop.apache.org>. Acesso em: agosto de 2013.
- [2] Shvachko, K., Kuang, H., Radia, S., Chansler, R., The Hadoop Distributed File System. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-10.
- [3] Ghemawat, S., Gobioff, H., Leung, S., The Google File System. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003, pp. 20-43.
- [4] Dean, J., Ghemawat, S., MapReduce: Simplified Data Processing on Large Clusters. In: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation (OSDI), 2004, pp. 1-10.
- [5] The Apache Software Foundation, Hadoop-Hive, 2013. Disponível em: <http://hive.apache.org/>. Acessado em: Agosto de 2013.
- [6] The Apache Software Foundation, Apache Wiki (Hive Tutorial). Disponível em: <https://cwiki.apache.org/confluence/display/Hive/Tutorial>. Acessado em: Agosto de 2013.
- [7] Jia, B., WiktorWlodarczyk, T., Rong, C., Performance Considerations of Data Acquisition in Hadoop System. In: 2nd IEEE International Conference on Cloud Computing Technology and Science, 2010, pp. 545-549.
- [8] Goldman, A., Kon, F., Junior, F. P., Polato, I., Pereira, R. P., Apache Hadoop: Conceitos teóricos e práticos, evolução e novas possibilidades. In: XXXI Jornadas de atualizações em informática, 2012, pp. 88-136.

Conectando Dados de Movimento Textualmente Anotados a Dados Ligados

Cleto May¹, Renato Fileto¹

¹Departamento de Informática e Estatística,
Universidade Federal de Santa Catarina (UFSC), Florianópolis, SC, Brasil

cleto.may@inf.ufsc.br, r.fileto@ufsc.br

Abstract. *The recent progress in movement analysis mostly considers spatio-temporal data (e.g. trajectories). However, other information (e.g., tags and comments associated to the spatiotemporal data) could also help to better understand movements (e.g., identify classes of frequented places and events). This article introduces a method for semantic enriching textually annotated movement data (e.g., trajectories, user's trails in social media) by connecting them to Linked Data. The current version of our method uses spatial and textual similarity to find and rank possible connections. Real data from Flickr and LinkedGeoData have been used to test the method implementation.*

Resumo. *O progresso recente na análise de movimento considera principalmente dados espaço-temporais (e.g. trajetórias). Contudo, outras informações (e.g., tags e comentários associados a dados espaço-temporais) também poderiam ajudar a explicar movimentos (e.g., identificar classes de locais e eventos frequentados). Este artigo introduz um método para enriquecer semanticamente dados de movimento textualmente anotados (e.g., trajetórias, trilhas de usuários em mídias sociais) conectando-os a dados ligados. A versão atual do nosso método usa similaridade espacial e textual para encontrar e ranquear possíveis conexões. Dados reais do Flickr e LinkedGeoData foram utilizados para testar a implementação do método.*

1. Introdução

A última década contou com grande popularização de dispositivos móveis (e.g., smartphones) e outros equipamentos dotados de sensores (e.g., GPS, GSM, RFID, câmeras) capazes de registrar movimento. Diversas aplicações podem ser suportadas pela enorme quantidade de dados coletados pelo uso de tais tecnologias. Todavia, isso requer técnicas apropriadas para extrair informação desses dados.

Dados de movimento são sequências temporalmente ordenadas de posições ocupadas por objetos que se movem. Cada amostra de posição de um objeto móvel inclui coordenadas geográficas, o momento em que o objeto ocupa tal posição e possivelmente anotações associadas aos dados espaço-temporais (e.g., tags, comentários). Este trabalho utiliza o termo *dados de movimento* como uma generalização para trajetórias [Parent et al. 2013] e trilhas de usuários de mídias sociais (e.g., Twitter, Facebook, FourSquare). As primeiras usualmente são colhidas por sensores e aplicativos específicos para tal finalidade, enquanto as últimas são sequências de postagens de um usuário em uma mídia social. Devido a própria forma de coleta de dados, trajetórias costumam ter boa

precisão espaço-temporal, ao passo que trilhas de redes sociais costumam ser esparsas, devido a característica assíncrona das postagens dos usuários em redes sociais. Por outro lado, trilhas de redes sociais costumam ser ricas em informações textuais, enquanto trajetórias raramente possuem anotações.

O desenvolvimento de sistemas que capturam diversos aspectos semânticos do movimento (e.g., locais visitados, razões de movimentos) ainda é um desafio, mesmo quando a entrada de tais sistemas inclui dados anotados textualmente. Este trabalho visa contribuir para preencher tal lacuna, propondo técnicas para conectar dados de movimento anotados textualmente com recursos de coleções de dados ligados disponíveis na Web Semântica, para auxiliar a análise do movimento segundo diversos aspectos semânticos.

[Fileto et al. 2013] mostra os benefícios da utilização de grandes coleções de dados ligados atualmente disponíveis, aderentes a padrões e com semântica bem definida, na análise de dados de movimento. Porém, não resolve adequadamente o problema da conexão entre os dados de movimento e os dados ligados.

As principais contribuições deste artigo são: (i) proposição de um método para conectar dados de movimento a recursos de dados ligados, utilizando informações espaciais e textuais; (ii) ordenação (*ranking*); e (iii) teste da implementação do método proposto em experimentos com dados reais.

As próximas seções deste artigo estão organizadas da seguinte forma. A Seção 2 apresenta alguns fundamentos. A Seção 3 descreve o método proposto. A Seção 4 discute os experimentos realizados. A Seção 5 estuda o estado da arte de problemas relacionados à conexão entre dados de movimento e dados ligados. Finalmente, a Seção 6 sumariza os resultados obtidos até aqui e enumera trabalhos futuros.

2. Fundamentos

Esta seção apresenta os fundamentos necessários ao entendimento do problema tratado e do método de solução proposto.

2.1. Dados Ligados

Dados ligados abertos, do inglês *Linked Open Data*¹ (LOD), surgem da necessidade de interligar com semântica bem definida e tornar acessível dados disponíveis na Web. Coleções de LOD são estruturadas como triplas RDF² da forma recurso-propriedade-valor [Antoniou and Harmelen 2008]. Um recurso é identificado unicamente por uma URI a partir da qual podem ser acessadas suas propriedades. Um recurso pode ter diversas propriedades (e.g., tipo, nome, rótulo, descrição). As possíveis propriedades de um recurso variam de acordo com o tipo (e.g., uma pessoa pode ter data e local de nascimento, um local pode ter coordenadas geográficas). O valor de uma propriedade pode ser outro recurso (e.g., o valor da propriedade local de nascimento é a URI do respectivo local) ou literal (conjunto de caracteres ou número).

2.2. Dados de movimento

Uma sequência bruta de dados de movimento, em inglês *Raw Movement Data*, é uma sequência de amostras de posições de um objeto móvel, cada qual coletada em um

¹<http://linkeddata.org/>

²<http://www.w3.org/TR/rdf-primer/>

instante no tempo. A definição 1 formaliza tal conceito.

Definição 1. Sequência bruta de dados de movimento $RawMD$ é uma sequência temporalmente ordenada de amostras de posições p_1, \dots, p_n de objetos móveis. Cada posição p_i tem a forma $p_i((x_i, y_i), t_i, S_i)$ onde:

- (x_i, y_i) são coordenadas geográficas;
- t_i é um instante de tempo; e
- $S_i = \{s_1, \dots, s_l\}$ é uma coleção de valores textuais de atributos associados à amostra de posição espaço-temporal (e.g., palavras-chave, *tags*).

A Figura 1 ilustra uma sequência bruta de dados de movimento. Cada amostra de posição está representada por ponto preto.

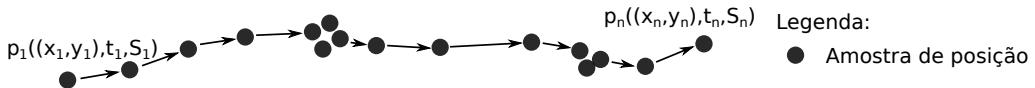


Figura 1. Sequência bruta de dados de movimento.

Uma sequência estruturada de dados de movimento, em inglês *Structured Movement Data*, é uma sequência temporalmente ordenada de episódios. Consideramos cada episódio como um segmento maximal não aninhado de amostras do movimento que cumprem um predicado (e.g., não se mover mais que uma dada distância durante um dado período de tempo) [Buchin et al. 2010]. Os predicados usados para identificar um episódio variam de acordo com a classe de episódio e a aplicação. A Definição 2 formaliza o conceito de sequência estruturada de dado de movimento.

Definição 2. Sequência estruturada de dados de movimento é uma sequência temporalmente ordenada de episódios E_1, \dots, E_m . Cada episódio E_i tem a forma $E_i(c_i, RawMD_i)$ onde:

- c_i é a classe do episódio (e.g., *stop*, *move* [Alvares et al. 2007]); e
- $RawMD_i$ é um segmento maximal de uma sequência bruta de dados de movimento que cumprem um predicado que determina o episódio (e.g., limite de velocidade, intervalo de tempo).

A Figura 2 ilustra uma sequência estruturada de dados de movimento. Episódios das classes *AltaVelocidade* e *BaixaVelocidade* estão representados por balões vermelhos e azuis, respectivamente.

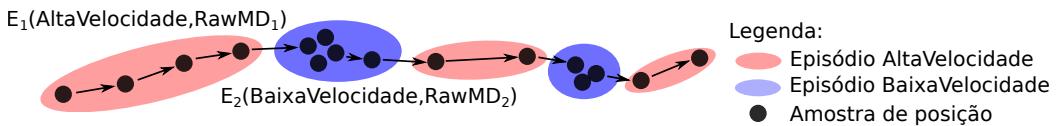


Figura 2. Sequência estruturada de dados de movimento.

Existem casos em que sequências de dados de movimento não são suficientes para responder consultas tais como: “Quais objetos móveis estiveram em locais turísticos?”. Para responder tal tipo de consulta é necessário que cada episódio precisamente esteja anotado semanticamente com as classes dos locais visitados. Preferencialmente, os

dados de movimento precisam ser conectados a itens de informação (e.g., um hotel, um restaurante, um local de interesse turístico) com definições precisas e com semântica bem definida, tais como recursos presentes em coleções de dados ligados [Fileto et al. 2013]. A Definição 3 formaliza o conceito de sequência semântica de dados de movimento.

Definição 3. Sequência semântica de dados de movimento é uma sequência temporalmente ordenada de episódios semanticamente anotados (i.e., conectados a recursos específicos presentes em coleções de dados ligados) SE_1, \dots, SE_m . Cada episódio semanticamente anotado SE_i tem a forma $SE_i(E_i, A_i)$, onde:

- E_i é um episódio da forma descrita na Definição 2; e
- A_i é uma coleção de anotações semânticas da forma $a(p, v)$, onde p é a propriedade que conecta o episódio E_i ao recurso v de uma coleção de LOD.

A Figura 3 ilustra uma sequência de episódios semanticamente anotados. As anotações estão representadas por balões verdes.

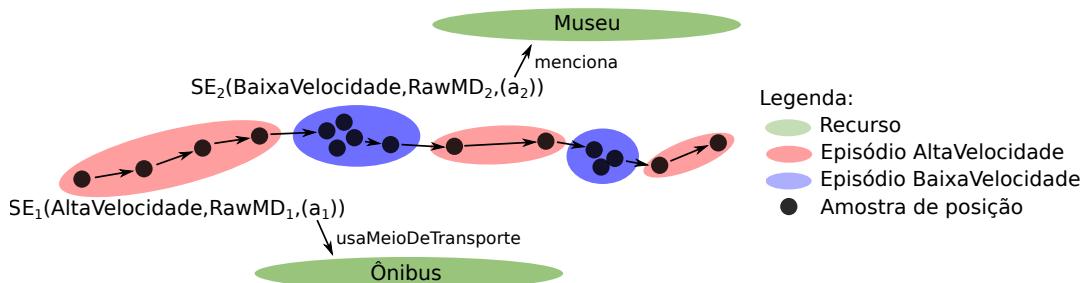


Figura 3. Sequência semântica de dados de movimento.

2.3. Problema Abordado

A Figura 4 exemplifica o problema de conexão entre dados de movimento e LOD tratado neste artigo. O canto inferior direito mostra uma imagem de satélite da região e o restante da figura uma foto daquela região, ambas obtidas do Google Maps³. O balão sobre a imagem de satélite representa a posição de uma postagem no Flickr⁴ que foi associada às palavras-chave apresentadas acima do balão amarelo (wheel, violet, toy, rod, etc.).

Nota-se que a posição do objeto móvel encontra-se em uma área densa da cidade. Além disso, a precisão da posição não é suficiente para inferir o local visitado pelo objeto móvel. A descrição textual auxilia tal tarefa. Considerando os rótulos e as descrições dos locais dentro de um certo raio de distância do ponto da postagem, pode-se concluir que o estabelecimento denominado Bike Dream é um forte candidato. O título e a descrição do recurso denominado Bike Dream estão léxica e semanticamente mais relacionados com as palavras-chave associadas a tal postagem, que aqueles de outros estabelecimentos ao redor e descritos na mesma coleção de LOD.

O objetivo deste trabalho é realizar a conexão entre dados de movimento e LOD de forma automática. Pode-se utilizar para isso informações espaço-temporais e textuais, assim como similaridade semântica e informações de contexto (e.g. características do objeto móvel), para encontrar conexões de menções a entidades nomeadas associadas aos dados de movimento com recursos de LOD (e.g., estabelecimentos, eventos).

³<https://maps.google.com/>

⁴<https://www.flickr.com/>



Figura 4. Conexão em área densa na cidade de Florianópolis

3. Método Proposto

O método proposto neste artigo almeja o enriquecimento semântico de dados de movimento textualmente anotados através da conexão a recursos de LOD. Sua versão atual utiliza proximidade espacial e textual, e é formalmente descrita pelo Algoritmo 1. As entradas para o método proposto são a sequência bruta de dados de movimento, uma coleção de dados ligados referente à mesma região e período de tempo que os dados de movimento e os limiares de proximidade espacial e similaridade léxica para efetuar as ligações. O resultado é a sequência semântica de dados de movimento, anotada com recursos da coleção de LOD fornecida como entrada.

Algoritmo 1: Função conecta

Entrada: *RawMD* // Sequência bruta de dados de movimento
LOD // Coleção de LOD
 τ_s // Limiar espacial
 τ_t // Limiar textual
Saída: *SemMD* // Sequência semântica de dados de movimento

```

1 início
2   StrMD  $\leftarrow$  estruturaDadosDeMovimento(RawMD)
3   para  $i = 0$  até  $|StrMD| - 1$  faça
4      $e \leftarrow StrMD[i]$ 
5     rProximos  $\leftarrow$  filtraEspacialmente(e, LOD,  $\tau_s$ )
6     eliminaCaracteresEspeciais(e)
7     eliminaCaracteresEspeciais(rProximos)
8     rankE  $\leftarrow$  filtraLexicamente(e, rProximos,  $\tau_t$ )
9     A  $\leftarrow \emptyset$ 
10    para cada r  $\in$  rankE faça
11      |   A  $\leftarrow A \cup$  menciona(r.recurso, r.proximidade)
12    fim
13    SemMD[i]  $\leftarrow$  SE(e, A)
14  fim
15  retorna SemMD
16 fim

```

Inicialmente, estruturamos a sequência bruta de dados de movimento e submete-

mos os episódios resultantes ao enriquecimento semântico. A estruturação é flexível, de acordo com a aplicação. Por exemplo, aplicações que desejam identificar pontos turísticos visitados por turistas podem utilizar a velocidade baixa para detectar episódios relevantes (e.g., admirar uma construção).

A função **filtraEspacialmente**, linha 5, seleciona todos os recursos da coleção LOD fornecida que estejam localizados a uma distância igual ou inferior a τ_s do episódio e . Utilizamos o centroíde do episódio para medir a distância a recursos e simplificar o processamento, mas pode-se também utilizar outros critérios (e.g., centro de massa). A Figura 5 ilustra um exemplo de aplicação do filtro espacial em que os recursos r_1 e r_2 são selecionados por estarem a uma distância inferior a τ_s do centroíde c_2 do episódio e_2 . Para simplificar, abstraimos as anotações textuais na imagem. Uma extensão para o filtro espacial seria a inclusão da compatibilidade temporal (e.g., momento do episódio compatível com o horário de funcionamento de um local ou evento), tornando-o então um filtro espaço-temporal.

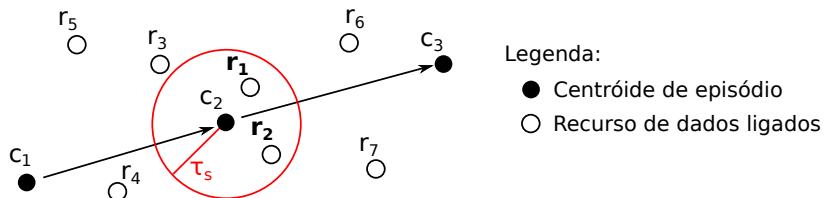


Figura 5. Filtro espacial

Nas linhas 6 e 7, a função **eliminaCaracteresEspeciais**, elimina das anotações textuais caracteres como asteriscos, parênteses, chave e barras, pois estes podem prejudicar a aplicação de funções de similaridade textual.

A função **filtraLexicamente**, presente na linha 8, analisa o conjunto de termos de cada par episódio-recurso para encontrar os recursos que são similares textualmente utilizando a função SoftTFIDF [Cohen et al. 2003]. Ela é adequada para o método pois combina funções de similaridade textual entre conjuntos de palavras (e.g., modelo vetorial) e palavras individuais (e.g., Jaro-Winkler). O recursos que possuírem similaridade textual superior ao limiar τ_t são selecionados. A Figura 6 ilustra a aplicação da função de similaridade textual entre um episódio (e_1) e dois recursos (r_1 e r_2) de coleções de dados ligados. A função é aplicada às anotações textuais de cada par episódio-recurso. Pode-se perceber que no exemplo o recurso r_1 possui a anotação textual “Bike” muito semelhante a anotação “bike” presente no episódio e_1 . Então r_1 é selecionado. Não podemos dizer o mesmo do recurso r_2 . Isso depende do limiar textual utilizado.

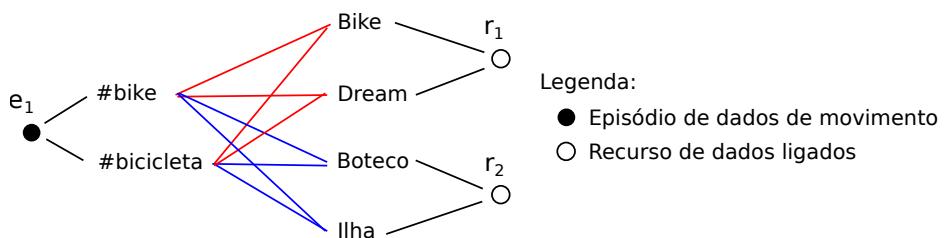


Figura 6. Investigação de correspondências léxicas

Finalmente, após encontrar os recursos próximos espacial e textualmente, na linha 11 são instanciadas todas as anotações semânticas com o respectivo recurso encontrado. Em seguida, na linha 13, o novo episódio conectado é acrescido a sequência semântica de dados de movimento. Novamente, uma extensão do método proposto seria a identificação da melhor propriedade para a conexão realizada, diferente do atual que adiciona a propriedade *menciona* a todas as conexões.

4. Experimentos

As sequências brutas de dados de movimento utilizadas na experimentação foram extraídas do CoPhIR⁵. Tais dados referem-se a fotos e dados associados (posição, momento, tags, etc.) publicadas na mídia social Flickr⁶ contidas no Brasil nos anos de 2005 a 2007. Utilizaremos tags associadas a cada ponto amostrado do movimento como sua descrição textual. Consideramos cada episódio descrito por todas as tags associadas a ponto de amostragem espaço-temporal que dele façam parte.

Para enriquecer semanticamente os dados de movimento oriundos do Flickr utilizamos os dados ligados do LinkedGeoData⁷ os quais foram triplificados do OpenStreetMap⁸ (ferramenta da coleta colaborativa de dados geográficos). Utilizamos os rótulos dos recursos para investigar as conexões léxicas com tags associadas aos dados de movimento.

Os experimentos práticos foram realizados em uma máquina com processador Intel(R) Core(TM) 2 Quad 2.40GHz, com 4Gb de memória RAM e um disco rígido de 500 Gb 7200 RPM. O algoritmo para efetuar a conexão semântica foi desenvolvido utilizando a linguagem Java além dos dados espaço-temporais coletados e armazenados em uma base de dados PostgreSQL, utilizando a extensão geográfica PostGIS.

4.1. Resultados

Submetemos 36.476 pegadas de usuários do Flickr ao algoritmo. O método foi capaz de realizar 9.598 conexões utilizando os parâmetros $\tau_s = 1000$ metros e $\tau_t = 0,2$. Parâmetros estes que foram escolhidos após um estudo simples da base dados utilizada. Em trabalhos futuros pretendemos utilizar diferentes parâmetros e comparar os resultados, além de avaliar a qualidade das conexões realizadas. A Tabela 1 ilustra algumas conexões realizadas. O tempo de execução médio foi de 1,3 segundos para cada episódio analisado. O ganho de performance é acrescido quando utiliza-se uma base de dados ligados local, pois cada requisição a uma coleção de dados ligados é custosa.

4.2. Discussão

Após analisar manualmente as conexões realizadas, percebemos relevantes conexões realizadas. Por exemplo, a Figura 7 ilustra uma amostra de dado de movimento (balão amarelo) e a sua conexão com um recurso (balão azul). Nota-se que o recurso conectado não precisa necessariamente estar próximo ao dado de movimento. No entanto, a sua conexão ainda é verdadeira, inclusive comum na base de dados utilizada, pois como se trata de fotos o objeto móvel pode estar localizado a uma certa distância do alvo fotografado. No exemplo, a fotografia foi tirada próxima ao morro do Corcovado, mas o objeto

⁵<http://cophir.isti.cnr.it>

⁶<http://www.flickr.com>

⁷<http://www.linkedgodata.org>

⁸<http://www.openstreetmap.org>

Tabela 1. Experimentos - Visão textual

Identificador do episódio	Palavras-chave do dado de movimento	URI do recurso	Rótulos
50344933	riodejaneiro, corcovado, brazil, brasil	lgd:/triplify/node1551149888	Corcovado
28567054	lovelyphotos, fortecopacabana, forte, copacabana	lgd:/triplify/node2308298808	Forte de Copacabana
107131876	rio, jardim, janeiro, garden, de, coutinho, by, bothanical, cotanico	lgd:/triplify/node1613993266	Jardim Botânico
40774808	viradacultural, virada, teatromunicipaldesãopaulo, teatromunicipal, teatro, sãopaulo, são, paulo, municipal, cultural	lgd:/triplify/way46927931	Teatro Municipal de São Paulo
72964391	pezinho, foot, copacabana, brazil, beach	lgd:/triplify/way179496798	Praia de Copacabana
37508938	voador, rio, janeiro, dish, deep, de, circo	lgd:/triplify/node331382172	Circo Voador

móvel não estava no local. A conexão ilustrada pela Figura 4 também foi realizada com um recurso relacionado com as tags associadas a amostra de dado de movimento, mesmo estando tal recurso em uma região densa de pontos de interesse.

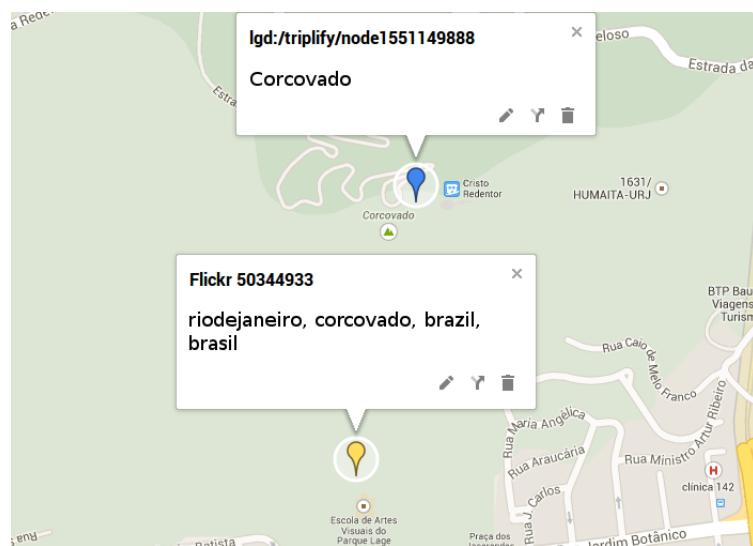


Figura 7. Experimentos - Visão geográfica

As conexões realizadas sugerem que informações espaciais e textuais são relevantes. No entanto, somente essas informações não são suficientes em todos os casos. Outras técnicas (e.g., aprendizado de máquina) e outras informações (e.g., informações temporais e de contexto) poderiam ser utilizadas para potencializar a qualidade das conexões.

5. Trabalhos Relacionados

O problema de conectar dados de movimento a dados ligados foi inicialmente definido em [Fileto et al. 2013]. Tal trabalho apontou os benefícios desta abordagem para o enriquecimento semântico de dados de movimento e a necessidade de desenvolver técnicas eficazes e eficientes para efetuar tal conexão.

Uma linha de pesquisa recente que ajuda na definição formal e solução eficiente do problema aqui tratado propõe maneiras de fazer a junção por similaridade entre bases de dados espaciais textualmente anotadas. [Ballesteros et al. 2011] propõe o cálculo para junção espaço-textual como sendo a razão entre similaridade textual (coeficiente de Jaccard) e espacial (distância Ortodromia). [Bourros et al. 2012] e [Liu et al. 2012] calculam as similaridades independentemente e consideram similares elementos que possuem valores de similaridade espacial e textual superiores aos seus respectivos limiares.

Outra linha de pesquisa, conhecida por *Entity Linking*, consiste em detectar em um texto menções a entidades de uma base de conhecimento. Um recente trabalho que define o problema e propõe sua solução é [Ceccarelli et al. 2013]. No entanto, em tal trabalho não são consideradas as informações espaço-temporais.

O diferencial do trabalho submetido quando comparado a trabalhos relacionados está na utilização de conhecimentos de junção por similaridade de bases de dados espaciais textualmente anotadas na análise de dados de movimento, identificando menções a entidades presentes em coleções de dados ligados.

6. Conclusões e Trabalhos Futuros

O enriquecimento semântico de dados de movimento é uma questão bastante discutida atualmente na literatura [Parent et al. 2013][Yan et al. 2013]. O potencial do enriquecimento semântico utilizando conexões a coleções de dados ligados merece atenção devido a estruturação com semântica bem definida que tais dados possuem. Este artigo propõe um método para realizar o enriquecimento semântico de dados de movimento textualmente anotados através da conexão a recursos provenientes de coleções de dados ligados. Suas principais contribuições são: (i) critérios para filtragem e ranqueamento de recursos de dados ligados com dados de movimento, baseados na distância espacial e similaridade léxica entre anotações do movimento e atributos textuais de dados ligados usando a medida SoftTFIDF; (ii) implementação e teste do método proposto com dados de movimento reais obtidos de postagens em mídia social e dados ligados. Os resultados obtidos mostram que a proposta é viável e promissora, embora muitas pesquisas ainda sejam necessárias para aprimorar e validar métodos de enriquecimento semântico de dados de movimento através da conexão com recursos de coleções de dados ligados.

Em trabalhos futuros planeja-se: (i) estender o método proposto com o uso de informações de contexto, similaridade semântica e técnicas como aprendizado de máquina, para ranquear as conexões candidatas identificadas pelo uso de similaridade espaço-temporal e textual; (ii) analisar a qualidade dos resultados gerados pelo método proposto com diferentes coleções de dados de movimento e dados ligados; (iii) estender o método proposto para determinar propriedades de conexões realizadas (e.g., *meio de transporte, local, evento*); e (iv) aprimorar o método proposto para execução mais eficiente sem comprometer a qualidade dos resultados.

Agradecimentos. Este trabalho contou com o apoio do projeto European Union's IRSES-SEEK (concessão 295179), do CNPq (concessão 478634/2011-0) e da FEESC.

Referências

- Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., and Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. In *Proceedings of the 15th annual ACM Intl. Symp. on Advances in geographic information systems*, GIS '07, pages 22:1–22:8, New York, NY, USA. ACM.
- Antoniou, G. and Harmelen, F. v. (2008). *A Semantic Web Primer, 2Nd Edition (Cooperative Information Systems)*. The MIT Press, 2 edition.
- Ballesteros, J., Cary, A., and Rishe, N. (2011). Spsjoin: Parallel spatial similarity joins. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 481–484, New York, NY, USA. ACM.
- Bouros, P., Ge, S., and Mamoulis, N. (2012). Spatio-textual similarity joins. *Proc. VLDB Endow.*, 6(1):1–12.
- Buchin, M., Driemel, A., van Kreveld, M., and Sacristán, V. (2010). An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 202–211, New York, NY, USA. ACM.
- Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., and Trani, S. (2013). Learning relatedness measures for entity linking. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 139–148, New York, NY, USA. ACM.
- Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A comparison of string metrics for matching names and records. In *Proceedings of the KDD-2003 Workshop on Data*, pages 13–18, Washington, DC.
- Fileto, R., Kruger, M., Pelekis, N., Theodoridis, Y., and Renso, C. (2013). Baquara: A holistic ontological framework for movement analysis using linked data. In Ng, W., Storey, V., and Trujillo, J., editors, *Conceptual Modeling*, volume 8217 of *Lecture Notes in Computer Science*, pages 342–355. Springer Berlin Heidelberg.
- Liu, S., Li, G., and Feng, J. (2012). Star-join: Spatio-textual similarity join. In *Proceedings of the 21st ACM Intl. Conf. on Information and Knowledge Management*, CIKM '12, pages 2194–2198, New York, NY, USA. ACM.
- Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., and Yan, Z. (2013). Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4):42:1–42:32.
- Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., and Aberer, K. (2013). Semantic trajectories: Mobility data computation and annotation. *ACM Trans. Intell. Syst. Technol.*, 4(3):49:1–49:38.

Análise de Abordagens para Recuperação de Informação em Tabelas na Web

Filipe Roberto Silva¹, Ronaldo dos Santos Mello¹

¹Departamento de Informática e Estatística– Universidade Federal de Santa Catarina
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil

{filipesilva.sc, ronaldo}@inf.ufsc.br

Abstract. *The web has been turning into a rich data source. Recent researches try to create even more ways to use these data and/or to facilitate their access. There are a lot of tables in the Web that hold useful data for human consumption. They are characterized by the <table> tag. This paper details information retrieval approaches related to Web tables, presenting a brief description of them, as well as a comparison of their main features. Besides, some open issues in this research area are highlighted.*

Resumo. *A web tem se tornado uma rica fonte de dados. Pesquisas recentes tentam criar cada vez mais meios de utilizar e/ou facilitar o acesso a esses dados. Existem muitas tabelas na Web que possuem dados úteis para o consumo humano. Elas são caracterizadas pela tag <table>. Este trabalho detalha abordagens de recuperação de informação relacionadas a tabelas na web, apresentando uma breve descrição delas e mostrando um comparativo entre suas principais características. Além disso, são destacadas algumas questões em aberto nessa área de pesquisa.*

1. Introdução

Segundo Lai (2013) existe uma grande quantidade de tabelas na web e essas tabelas possuem dados que poderiam ser de grande proveito para a obtenção de informação útil para consumo humano. Essa informação poderia ser melhor aproveitada se processada por um computador e indexada, visto que este é um trabalho bastante árduo para ser feito de forma manual. Porém, essas fontes de dados não possuem um padrão bem definido. As tabelas na web são criadas para serem lidas por pessoas. Por isso podem estar em várias disposições diferentes e tratar sobre diversos assuntos. Enfim, pode ser complexo para um computador entendê-las e coletar seu conteúdo.

As tabelas na web, assim como as tabelas relacionais, basicamente são compostas por rótulos que caracterizam os atributos e valores que caracterizam as tuplas. Porém a forma como essas estruturas aparecem nas tabelas pode variar bastante. Por isso, muitas abordagens tratam de identificar essas estruturas nas tabelas. Outras abordagens observadas tratam de recuperar informação semântica presente nas tabelas. Isso poderia ser útil para uma indexação mais significativa.

Este artigo tem como objetivo descrever e comparar abordagens de recuperação de informação em tabelas na web, mostrando suas características, pontos fortes e fracos e por fim, apresentando sugestões de tópicos a serem abordados nessa área de pesquisa.

A Seção 2 apresenta uma breve descrição de cada abordagem pesquisada. Ela está dividida em três subseções: Identificação da estrutura, Extração de dados e Recuperação com semântica, de acordo com a intenção dos trabalhos analisados. A Seção 3 mostra um comparativo das abordagens apresentadas e a Seção 4 apresenta a conclusão obtida da pesquisa e algumas propostas de pesquisa na área de recuperação de informação em tabelas na web.

2. Abordagens

Nesta subseção são apresentadas as abordagens estudadas. Os trabalhos analisados foram subdivididos em três classificações de acordo com o conteúdo: identificação de estrutura, extração de dados e recuperação com semântica. Os trabalhos sobre identificação de estrutura visam descobrir como as tabelas estão estruturadas. Trabalhos sobre extração de dados possuem menor foco na estrutura das tabelas e mais na extração dos dados em si. Por fim, os trabalhos sobre recuperação com semântica identificam e anotam informação semântica nas tabelas na web.

2.1. Identificação da estrutura

Vários métodos de classificação de tabelas foram propostos no passado [Wang and Hu 2002, Cafarella and Wu 2008], porém grande parte das pesquisas considera esse um problema binário, ou seja, com duas possibilidades de classificação (genuína x não-genuína, relacional x não-relacional, *layout* x dados). Diferente destes trabalhos anteriores, Crestan e Pantel (2010) propõe uma nova taxonomia feita empiricamente para as tabelas na web.

Crestan e Pantel (2010) explica que a maioria das tabelas na web são utilizadas para definir a estrutura de apresentação de dados na Web, estas são chamadas de tabelas *layout*. Também existem as tabelas que contêm dados relevantes para serem extraídos. Essas tabelas de dados são classificadas pelo trabalho como tabelas relacionais. Assim, são formados dois grandes grupos descritos pelo trabalho. Dentro dessas tabelas relacionais e *layout*, foram feitas classificações mais específicas. As tabelas relacionais são subdivididas entre os seguintes tipos:

- **Vertical:** as tuplas estão dispostas na direção vertical;
- **Horizontal:** as tuplas estão dispostas na direção horizontal;
- **Atributo/Valor:** caso específico de tabela Vertical/Horizontal que não apresenta o assunto na própria tabela, visto que este pode ser obtido do contexto da tabela. Muito utilizadas em especificações técnicas de produtos;
- **Matriz:** este tipo de tabela possui cabeçalhos na vertical e horizontal e no cruzamento entre os dois encontra-se o valor. São utilizados para cruzar dois atributos, por exemplo, número de acidentes por mês para cada estado;
- **Calendário:** tipo especial de tabela Matriz, sendo que um dos atributos é uma data;
- **Enumeração:** este tipo de tabela lista uma série de objetos relacionados;
- **Formulário:** este tipo de tabela é composto por campos de formulário;
- **Outros:** tabelas que não se enquadram nos tipos anteriores.

As tabelas *layout*, por sua vez, podem ser classificadas em dois tipos:

- **Navegação:** tabelas utilizadas para navegação pelo site, por exemplo, categorias de produtos disponíveis;
- **Formatação:** tabelas utilizadas para organizar visualmente os elementos da página.

Crestan e Pantel (2010) propõe, além dessa taxonomia para as tabelas na web, um sistema de classificação supervisionado para tabelas na web. Porém, esse sistema não utiliza toda a taxonomia proposta. Ele classifica as tabelas somente em atributo/valor, layout ou outras.

Lautert et al. (2013) também apresenta uma taxonomia semelhante a Crestan e Pantel (2010). Ele leva em conta alguns tipos de tabelas que não foram citados neste trabalho anterior. Assim, além das já citadas tabelas Verticais, Horizontais e Matriciais, são apresentadas as seguintes classificações:

- **Concisa:** tabela que possui células mescladas;
- **Aninhada:** tabelas dentro de tabelas;
- **Dividida:** tabelas que, por questão de espaço, são divididas horizontal ou verticalmente e suas partes são posicionados lado a lado ou uma sobre a outra;
- **Multivvalorada Simples:** tabelas com múltiplos valores de um mesmo domínio em uma célula;
- **Multivvalorada Composta:** tabelas com múltiplos valores de diversos domínios em uma célula.

Lautert et al. (2013) além de ser um pouco mais abrangente em sua classificação de tabelas, também apresenta um sistema de classificação supervisionado para as tabelas que, diferente de Crestan e Pantel (2010), utiliza toda a taxonomia proposta.

Son e Park (2013) propõe um método de classificação de tabelas entre relacionais e *layout*. O trabalho explica que as tabelas possuem informações de conteúdo e estruturais. Porém, nem sempre é fácil definir as características estruturais das tabelas. Por isso ele utiliza um algoritmo de análise de padrões denominado *Convolution Kernel*.

Segundo Son e Park (2013), existem dois tipos de informação estrutural. Uma delas consiste nas *tags* que constituem as tabelas e suas relações. O outro tipo consiste no contexto onde a tabela está inserida, ou seja, relações entre as *tags* internas e externas à tabela. Essas características, juntamente com as informações de conteúdo, são processadas separadamente em algoritmos de análise de padrões. Por fim, os padrões são utilizados para treinar máquinas de vetores de suporte (SVM) que verificam quais características melhor definem uma tabela de dados ou de layout. A partir de um modelo treinado é possível utilizar uma SVM para classificar novas tabelas.

Lai (2013) propõe um método de extrair a estrutura das tabelas na web e reorganizá-las para melhorar a acessibilidade aos usuários com deficiência visual. O modo mais comum para essas pessoas acessarem a web é através de softwares que traduzem texto em fala. Mas esses sistemas simplesmente falam o que está na tela de forma linear, o que dificulta o entendimento de tabelas na web. Por isso, o trabalho foca na extração da estrutura dessas tabelas para recuperar suas informações e melhor apresentá-las aos deficientes visuais.

Primeiramente é necessário classificar as tabelas em tabelas de layout e de dados. Para isso são verificadas similaridades das células horizontais e verticais das tabelas, o que

é chamado no trabalho de *Hparallel* e *Vparallel*. Essa similaridade é verificada através de características visuais (como dados CSS) e de texto utilizando funções de similaridade.

Assim são verificadas as células similares horizontalmente e verticalmente, e comparadas com o total de colunas ou linhas. Dependendo de um valor de corte as células são tidas como *Vparallel* ou *Hparallel*. A seguir, a partir da quantidade dessas células similares, compara-se com a quantidade de células totais e dependendo de outro valor de corte a tabela é classificada como *layout* ou de dados. Os valores de corte são obtidos a partir de tabelas de treinamento. O sistema também busca por linhas ou colunas que possuam menor similaridade com o restante da tabela para identificar cabeçalhos ou rodapés.

Assim, identificadas as estruturas das tabelas, é possível transformá-las em estruturas mais fáceis de serem interpretadas por sistemas de leitura para deficientes visuais. Lai (2013) explica que os sistemas de leitura interpretam essas tabelas lendo linha por linha ou coluna por coluna dependendo da disposição da tabela, e associa o cabeçalho à célula que está sendo lida.

2.2. Extração de dados

Embley et al. (2011) mostra uma forma de manipular tabelas na web visando indexar seus valores relacionando-os com os cabeçalhos. Este trabalho trata em específico tabelas que segundo Crestan e Pantel (2010) possuem a classificação de Matriz. O trabalho introduz o conceito de *Header Path*, que organiza de forma hierárquica os cabeçalhos de uma tabela, dos níveis superiores até os inferiores. A Tabela 1, por exemplo, apresenta os cabeçalhos 'Temperature' e 'Day' e suas especializações que são 'Min', 'Max', 'Monday', 'Tuesday' e 'Wednesday'. Portanto o *Header Path* para esse caso seria:

- Temperature
 - Min
 - Max
- Day
 - Monday
 - Tuesday
 - Wednesday

Tabela 1. Tabela com cabeçalhos aninhados

		Temperature	
		Min	Max
Day	Monday	11C	22C
	Tuesday	9C	19C
	Wednesday	10C	21C

Para criar esses *Header Paths*, são processados arquivos CSV com as tabelas já extraídas da web. Os arquivos CSV são processados por rotinas escritas em Python de forma a encontrar os cabeçalhos e criar a estrutura dos *Header Paths*. Esses *Header Paths* podem ser criados para cabeçalhos na vertical, horizontal ou ambos. A partir disso, o trabalho apresenta uma linguagem de consulta baseada nos operadores lógicos de união e intersecção.

Por exemplo, novamente na Tabela 1, pode-se aplicar uma consulta na forma $(Temperature * Min) + (Temperature * Max)$, que seria uma união das colunas Min e Max, resultando no conjunto de valores completo de valores exibido na tabela. Porém, se aplicarmos a consulta da forma $(Temperature * Min) * (Day * Monday)$ seleciona-se somente a célula da intersecção entre Day = Monday e Temperature = Min, no caso: 11C.

Por fim, a partir desses *Header Paths* e dessa linguagem de consulta, o trabalho demonstra como é possível gerar uma nova estrutura relacional, de forma a permitir consultas SQL sobre os dados extraídos da tabela web.

Nagy et al. (2011) é uma continuação do trabalho apresentado por Embley et al. (2011). Ele mostra um tratamento dado a tabelas mais complexas. Um exemplo dado pelo trabalho é a Tabela 2 que possui a célula no canto superior esquerdo com valor 'A', onde não se sabe se é um cabeçalho de linha ou de coluna. O trabalho explica que na grande maioria dos casos observados, essa célula de canto é um cabeçalho de linha. Por isso, nesses casos, esta célula é aceita como cabeçalho de linha.

Tabela 2. Tabela com canto indefinido [Nagy et al. 2011]

A	B1	B2
C1	D11	D12
C2	D21	D22

Também em casos onde a tabela não está na forma de matriz, o trabalho propõe utilizar os valores como índices e construir os *Header Paths* sobre eles. Porém, em casos como da Tabela 3 onde, por exemplo, os valores de 'State' se repetem, seria necessário utilizar mais de uma coluna de valores como índice. No caso da Tabela 3 seriam utilizadas as três primeiras colunas para construir o *Header Path* vertical.

Tabela 3. Índice com múltiplas colunas [Nagy et al. 2011]

State	Company Name	Plant I.D.	Plant Name	County	Biomass / Coal Cofiring Capacity	Total Plant Capacity
AL	DTE Energy Services	50407	Mobile Energy Services LLC	Mobile	91	91
AL	Georgia-Pacific Corp	10699	Georgia Pacific Naheola Mill	Choctaw	31	78
AL	International Paper Co	52140	International Paper Prattville Mill	Autauga	49	90
AZ	Tucson Electric Power Co	126	H Wilson Sundt Generating Station	Pima	173	559
ROWS OMITTED						
MI	S D Warren Co	50438	S D Warren Muskegon	Muskegon	51	51
MI	TES Filer City Station LP	50835	TES Filer City Station	Manistee	70	70
MN	Minnesota Power Inc	10686	Rapids Energy Center	Itasca	27	28
MN	Minnesota Power Inc	1897	M L Hubbard St	Louis	73	123
ROWS OMITTED						

Para solucionar esses problemas, o trabalho mostra uma solução supervisionada para correção dos *Header Paths*. A geração dos *Header Paths* em si, é feita do mesmo modo como é mostrado em Embley et al. (2011). Porém, é acrescentada a verificação pelo usuário que precisa informar ao software se a detecção de cabeçalhos e dados está correta.

Ainda na linha de extração de dados, Cafarella et al. (2009) descreve o sistema *WebTables*, desenvolvido para extrair dados estruturados apresentados na forma de tabelas na web. O sistema *WebTables* utiliza uma combinação de classificadores para recuperar as tabelas relacionais de um conjunto de tabelas na web. Após essa classificação é obtido um grande conjunto de dados relacionais.

São apresentados dois passos para a obtenção das bases de dados relacionais a partir de tabelas HTML cruas. Primeiramente, uma amostra de tabelas é classificada em relacional e *layout*. Para isso são utilizadas heurísticas escritas manualmente para filtrar tabelas com características mais específicas. Por exemplo, as que possuem somente uma linha ou somente uma coluna, tabelas utilizadas para mostrar calendários e tabelas com formulários.

O segundo passo é rotular as tabelas restantes como relacionais e *layout* usando classificadores treinados. Esses classificadores se baseiam em aspectos pré definidos que caracterizam cada tipo de tabela, como número de linhas, colunas, células vazias e etc.

Depois desse filtro o sistema tenta recuperar os metadados de cada relação. A principal fonte de metadados apresentada são os cabeçalhos das tabelas. Para recuperá-los foi utilizado outro classificador treinado que compara a primeira linha de cada coluna com o corpo da tabela para detectar se existe cabeçalho ou não.

Sardi Mergen et al. (2010) apresenta um sistema de busca por dados em tabelas na web que utiliza uma linguagem de consulta simples e que possibilita uma seleção mais precisa de dados obtidos de tabelas na web.

Primeiramente o sistema indexa tabelas na web a partir de seus atributos. Não está claro no trabalho como são obtidos esses atributos, mas eles são referentes aos dados contidos nas tabelas, como por exemplo, título do filme, ano de lançamento e etc. Esses índices são criados na forma *atributo* → *valores* → *tabelas* e *atributo* → *tabelas* → *valores*. Também são identificados os tipos de valores presentes nas tabelas. A partir disso, o usuário pode criar consultas selecionando atributos e especificando condições de consulta.

2.3. Recuperação com Semântica

Venetis et al. (2011) descreve um sistema que busca recuperar a semântica das tabelas na web acrescentando nelas anotações. O objetivo principal desse sistema é contribuir com as buscas na web.

O trabalho explica que os motores de busca da web tratam as tabelas como documentos de texto comuns. Porém, as tabelas poderiam ser melhor aproveitadas se fossem tratadas de forma diferente, observando a semântica contida nelas. Por exemplo, muitas vezes as tabelas não possuem cabeçalhos explícitos demonstrando o assunto tratado na mesma. Com a recuperação da semântica dessas tabelas seria possível utilizá-las como resultado de buscas mesmo elas não contendo dados explicitamente relacionados à palavra chave. O conhecimento da semântica das tabelas também permitiria a aplicação de operações de combinação de tabelas, como *join* e *union*.

Assim, para recuperar a semântica das tabelas, o trabalho propõe a criação de duas bases de dados obtidas automaticamente a partir de textos da web. A base *isA* com pares na forma (classe,instância) é obtida utilizando basicamente padrões linguísticos. A outra possui relações em triplas na forma (argumento1,predicado,argumento2). Ela é obtida utilizando o TextRunner [Banko and Etzioni 2008] como ferramenta de extração de informação a partir de textos.

Segundo o trabalho, uma coluna *A* é rotulada com uma classe *C* da base de dados *isA*, se uma fração substancial das células na coluna *A* são rotuladas com a classe *C* na

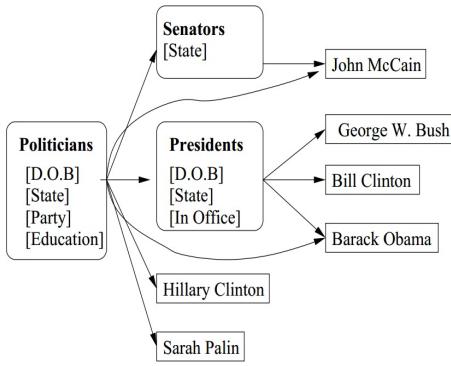


Figura 1. Um fragmento do Probbase [Wang et al. 2012]

base de dados *isA*. De forma semelhante, a relação entre duas colunas *A* e *B* é rotulada com *R* se uma fração substancial de pares de valores de *A* e *B* ocorre nas extrações na forma (a, R, b) na base de dados de relações.

Wang et al. (2012) também propõe uma forma de recuperar a semântica das tabelas na web. Ele cita que a chave para entender as tabelas é saber qual conceito melhor descreve as entidades e atributos contidos nas tabelas. Para encontrar esses conceitos, o trabalho utiliza uma base de conhecimento chamada Probbase [Wu et al. 2012]. Esta base contém conceitos, atributos e entidades. A Figura 1 mostra um exemplo disso. Ela possui conceitos, como 'Politicians', 'Presidents' e 'Senator', atributos como 'State', 'Party', 'D.O.B' e entidades como 'John McCain' e 'Bill Clinton'. As linhas e colunas das tabelas são comparadas a essa base e é acrescentada semântica às tabelas.

Feita essa detecção da semântica das tabelas, o trabalho apresenta um sistema de busca semântica em tabelas. Esse sistema, ao invés de recuperar páginas inteiras a partir de palavras chave, retorna tuplas e atributos específicos contidos em tabelas, semelhante a consultas SQL.

Fan et al. (2013) apresenta um sistema semi-supervisionado para encontrar conceitos em tabelas. O objetivo do trabalho é descobrir esses conceitos e utilizá-los em comparações de tabelas na web. Esse processo seria o mesmo que recuperar o esquema das tabelas e realizar um *schema matching*.

Para encontrar os conceitos de um grupo de tabelas, Fan et al. (2013) utiliza uma base de conhecimento, buscando adicionar conceitos a cada coluna dessas tabelas. O sistema verifica o grau de dificuldade em se descobrir os conceitos de cada coluna. Para isso, é utilizada uma função de similaridade. Essa função tem como entradas os valores de uma coluna *A* e um conceito *C* e retorna a probabilidade de *A* e *C* serem relacionados. Com isso, cada coluna recebe pesos associados a conceitos. Quanto mais idênticos são os pesos, maior a dificuldade em classificar a coluna. Com isso é determinado um grau de dificuldade para cada coluna. Para os casos considerados difíceis, o usuário deve realizar a classificação manualmente.

A seguir Fan et al. (2013) explica que saber os conceitos de algumas colunas pode ajudar a descobrir os conceitos de outras colunas. Por isso ele calcula o grau de influência de uma coluna sobre as outras. Descobertos os conceitos de cada tabela, é possível compará-las.

3. Comparativo das Abordagens

Esta seção apresenta um comparativo entre as abordagens, destacando suas similaridades e diferenças. A Tabela 4 resume este comparativo.

Tabela 4. Comparativo das Abordagens

Trabalho	Abordagem	Tipo de Tabelas Processadas	Processamento	Objetivo	Estratégia
[Crestan and Pantel 2010]	Estrutura	Dados e Layout	Supervisionado	Classificação	Árvore de decisão
[Lautert et al. 2013]	Estrutura	Todos os tipos dentro da taxonomia	Supervisionado	Classificação	Rede neural
[Son and Park 2013]	Estrutura	Dados e Layout	Semi-supervisionado	Classificação	<i>Convolution kernel</i> (análise de padrões)
[Lai 2013]	Estrutura	Horizontais, Verticais, Matriciais	Automático	Identificação da Estrutura	Funções de similaridade
[Embley et al. 2011]	Extração de dados	Matriciais	Automático	Extração de dados	Header Paths, fatoração algébrica
[Nagy et al. 2011]	Extração de dados	Horizontais, Verticais, Matriciais	Supervisionado	Extração de dados	Header Paths, fatoração algébrica
[Cafarella et al. 2009]	Extração de dados	Horizontais	Semi-supervisionado	Extração de dados	Classificador treinado
[Sardi Mergen et al. 2010]	Extração de dados	Horizontais	Semi-supervisionado	Extração de dados	Índices
[Wang et al. 2012]	Recuperação com semântica	Horizontais	Automático	Busca semântica	Base de Conhecimento (Probbase)
[Venetis et al. 2011]	Recuperação com semântica	Horizontais	Automático	Busca semântica	Base de Conhecimento (própria)
[Fan et al. 2013]	Recuperação com semântica	Horizontais	Semi-supervisionado	<i>Schema Matching</i>	Base de Conhecimento e funções de similaridade

As abordagens de identificação de estrutura são úteis no sentido de descobrir como a tabela está disposta ou se ela possui dados ou não. Isso pode facilitar a descoberta das informações contidas ali. Dentro dessas abordagens, Lautert et al. (2013) é a que mais se destaca por sua classificação mais completa. Vale lembrar que Crestan e Pantel (2010) também apresenta uma taxonomia bastante abrangente, porém seu classificador não utiliza a taxonomia completa em seu sistema de classificação supervisionado, ao contrário de Lautert et al. (2013). Son e Park (2013) se destaca por seu algoritmo de detecção de padrões, tornando menos necessária a interação humana com entradas de dados. Porém, esse trabalho classifica somente as tabelas em dados e layout e dentro das tabelas de dados existe uma grande variedade de disposições de tabelas. Lai (2013) possui uma abordagem relativamente simples de encontrar linhas, colunas e cabeçalhos nas tabelas, não utilizando inteligência artificial para isso.

Quanto às abordagens de extração de dados, todas exceto Sardi Mergen et al. (2010) buscam extrair os dados e inseri-los em bases de dados relacionais. Embley et al. (2011) e Nagy et al. (2011) com sua linguagem de consulta e seu *Header Path*, poderiam ser utilizados no acesso aos dados diretamente das tabelas na web, porém esses trabalhos utilizam diretamente dados já extraídos para arquivos CSV. Cafarella et al. (2009) por sua vez, extraí os dados diretamente das páginas web, utilizando um classificador treinado. Sardi Mergen et al. (2010) se destaca por obter informações da web em tempo real, sem a necessidade de guardar as tabelas em um repositório. Isso torna os dados sempre atualizados de acordo com as páginas web.

Os trabalhos de recuperação com semântica são bastante semelhantes entre si. Todos utilizam uma base de conhecimento e buscam acrescentar anotações ou conceitos às tabelas. Wang et al. (2012) e Venetis et al. (2011) utilizam as anotações semânticas para aprimorar as buscas por tabelas na web. Em termos de precisão, Wang et al. (2012) apresentou melhores resultados nos experimentos de busca, porém os dois trabalhos possuem abordagens de busca um pouco diferentes. Enquanto Venetis et al. (2011) retorna tabelas inteiras, Wang et al. (2012) retorna tuplas retiradas das tabelas. Fan et al. (2013) por sua vez, possui seu foco em descobrir os esquemas das tabelas e após isso, tenta descobrir outras tabelas com mesmo esquema. Vale lembrar que dos trabalhos de recuperação de semântica estudados, Fan et al. (2013) é o único que não possui um sistema automático.

Notou-se na grande maioria dos trabalhos, o uso de ferramentas de inteligência artificial. Dentre essas ferramentas, as mais utilizadas foram bases de conhecimento e sistemas treinados. Também notou-se o uso explícito de funções de similaridade em pelo menos dois trabalhos. Porém, em nenhum trabalho foram encontradas verificações de sinônimos.

Em praticamente todos os trabalhos estudados, apesar de alguns não darem tanto foco nisso, notou-se a necessidade de separar tabelas de dados e de layout, porém, cada trabalho apresenta uma forma um pouco diferente de realizar esse processo. Também em muitos trabalhos notou-se a necessidade de detecção dos cabeçalhos contidos nas tabelas, sendo que em alguns casos, eles não estão presentes diretamente nas tabelas, e sim, implícitos no contexto em que as tabelas estão inseridas.

4. Conclusão

Este trabalho apresenta abordagens na literatura relacionadas à recuperação de informação contida em tabelas na web. Uma breve descrição de cada trabalho é mostrada, vantagens e desvantagens são apontadas e um comparativo foi produzido. Os trabalhos possuem abordagens que diferem umas das outras, porém cada um contribui de certa forma para recuperar informações contidas nas tabelas.

Alguns temas, como a detecção de estrutura das tabelas e classificação entre tabelas de layout e de dados são de grande necessidade na manipulação de tabelas na web. Também existe uma grande tendência no uso de ferramentas de inteligência artificial e funções de similaridade e as abordagens de extração de dados e anotação semânticas, em sua essência, são bastante semelhantes entre si.

Porém, alguns temas poderiam ser melhor explorados, como a detecção de temas das tabelas a partir da exploração do contexto onde estão inseridas, extratores baseados nas classificações propostas, além do uso mais efetivo de dicionários nas anotações semânticas. Outras abordagens promissoras seriam a análise das tabelas na web sem a necessidade de extraí-las para bases de dados relacionais, execução de junções entre tabelas e a descoberta de tabelas similares para fins de consulta unificada.

Referências

- Banko, M. and Etzioni, O. (2008). The tradeoffs between open and traditional relation extraction. In *ACL*, pages 28–36.
- Cafarella, M. J., Madhavan, J., and Halevy, A. (2009). Web-scale extraction of structured data. *SIGMOD Rec.*, 37(4):55–61.

- Cafarella, M. J. and Wu, E. (2008). Uncovering the relational web. In *In under review*.
- Crestan, E. and Pantel, P. (2010). A fine-grained taxonomy of tables on the web. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1405–1408, New York, NY, USA. ACM.
- Embley, D. W., Krishnamoorthy, M., Nagy, G., and Seth, S. (2011). Factoring web tables. In *Proceedings of the 24th international conference on Industrial engineering and other applications of applied intelligent systems*, IEA/AIE'11, pages 253–263, Berlin, Heidelberg. Springer-Verlag.
- Fan, J., Lu, M., Ooi, B. C., Tan, W.-C., and Zhang, M. (2013). A hybrid machine-crowdsourcing system for matching web tables. Technical report, Technical Report.
- Lai, P. P. Y. (2013). Adapting data table to improve web accessibility. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 33:1–33:4, New York, NY, USA. ACM.
- Lautert, L. R., Scheidt, M., and Dorneles, C. F. (2013). Web table taxonomiy and formalization. *SIGMOD*.
- Nagy, G., Seth, S. C., Jin, D., Embley, D. W., Machado, S., and Krishnamoorthy, M. S. (2011). Data extraction from web tables: The devil is in the details. In *ICDAR*, pages 242–246. IEEE.
- Sardi Mergen, S. L., Freire, J., and Heuser, C. A. (2010). Indexing relations on the web. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 430–440, New York, NY, USA. ACM.
- Son, J.-W. and Park, S.-B. (2013). Web table discrimination with composition of rich structural and content information. *Appl. Soft Comput.*, 13(1):47–57.
- Venetis, P., Halevy, A., Madhavan, J., Paşa, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4(9):528–538.
- Wang, J., Wang, H., Wang, Z., and Zhu, K. Q. (2012). Understanding tables on the web. In *Proceedings of the 31st international conference on Conceptual Modeling*, ER'12, pages 141–155, Berlin, Heidelberg. Springer-Verlag.
- Wang, Y. and Hu, J. (2002). A machine learning based approach for table detection on the web. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 242–250, New York, NY, USA. ACM.
- Wu, W., Li, H., Wang, H., and Zhu, K. Q. (2012). Probbase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 481–492, New York, NY, USA. ACM.

Um Estudo das Abordagens para Extração de Esquemas XML

Geomar A. Schreiner¹, Denio Duarte¹

¹Universidade Federal da Fronteira Sul - UFFS
Campus Chapecó

geomarschreiner@gmail.com, duarte@uffs.edu.br

Abstract. *XML (Extensible Markup Language) is a mark-up language that models semistructured data. XML lets the user define his own customized markup language for different document classes. Users can create XML documents freely. This flexibility has a drawback: processing XML documents can be computational onerous. In this context, schemas play an important role. This work presents a survey of three XML schema extraction approaches. We use a running example to show how the approaches work.*

Resumo. *Documentos XML são caracterizados por possuírem a estrutura armazenada junto com os dados tornando a sua aplicação bastante flexível. Porém, o efeito colateral dessa flexibilidade é que o processamento dos documentos se torna custoso computacionalmente. Isso pode ser resolvido associando um esquema ao documento permitindo que a aplicação conheça ‘a priori’ a estrutura do documento a ser processado. Este trabalho tem como objetivo estudar algumas destas ferramentas apresentando as abordagens utilizadas por elas e uma breve comparação entre as mesmas.*

1. Introdução

Documentos XML são caracterizados por não possuirem tipos associados as suas estruturas, ou seja, as aplicações e/ou usuários podem criar e alterar a estrutura dos dados e os dados livremente. Essa característica é propria da proposta da XML: modelar dados semi-estruturados. Tais dados não possuem tipos associados, são auto-descritivos e irregulares, sem distinção entre suas estruturas e os dados propriamente ditos.

Ao *tipar* instâncias XML (documentos XML), as aplicações e/ou usuários não podem mais livremente criar e modificar tais instâncias. A atualização dessas instâncias deve respeitar as restrições impostas pelo tipo associado (esquema). Documentos XML associados a esquemas são chamados válidos quando respeitam as regras impostas pelos esquemas. A validade não é característica obrigatória para os documentos XML. Assim, um documento XML associado a um esquema e que não respeita o mesmo continua sendo um documento bem formado sendo tratado pelos processadores de documentos XML sem problemas.

Se um esquema restringe a liberdade de criação e atualização de documentos XML por que, então, utilizá-los? Apesar da flexibilidade dos dados semi-estruturados, esquemas são úteis pois: (*i*) descrevem os dados e auxiliam a consulta sobre os mesmos, (*ii*) permitem, também, otimizar tais consultas, (*iii*) são base para abordagens eficientes para armazenamento dos dados, (*iv*) permitem que projetistas das aplicações descrevam a estrutura dos documentos, criando classes de documentos, e (*v*) facilitam o processo de transformação dos documentos para diferentes formatos (*i.e.*, dados relacionais).

A maioria dos documentos encontrados na WEB não possuem um esquema associado a eles ou o documento não respeita seu esquema [Barbosa et al. 2005]. O que torna muito pertinente que através de uma determinada coleção de documentos XML (mesmo que unitária) seja possível a extração de um esquema que descreva a estrutura da coleção. Segundo [Garofalakis et al. 2000], seres humanos fazem a melhor inferência possível de um esquema para determinada coleção de documentos XML. Evidentemente esta abordagem torna-se impraticável para documentos extensos ou coleções com muitos documentos. Sendo assim, é necessária uma ferramenta que realize a extração de forma automática ou semi-automática.

Diversas abordagens foram propostas para solucionar este problema: XTRACT [Garofalakis et al. 2000], Inferência Gramatical [Chidlovskii 2001], Min&Chung [Min et al. 2003], XStruct [Hegewald et al. 2006] e XTLSM [Amiel et al. 2008]. Este trabalho apresenta três das abordagens citadas anteriormente: XTRACT e Inferencial Gramatical por terem o maior número de citações segundo ScholarGoogle[©] e XTLSM por ser o mais recente.

A metodologia de apresentação segue estes passos: o método é brevemente descrito com o seu pseudo-código e, em seguida, um exemplo do funcionamento é apresentado. Todos os métodos estudados utilizam uma coleção unitária de documentos XML como entrada para fins de simplificação.

2. Abordagens Estudadas para Extração de Esquemas

O problema de extração de esquema pode ser definido como: dada um coleção de documentos XML (possivelmente unitária) com alguma semelhança estrutural, inferir um esquema que descreva a coleção de entrada. Esta seção apresentará algumas abordagens para extração de esquemas XML a partir de documentos XML e utiliza como exemplo de estudo de caso o documento XML *X* apresentado na Figura 1.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <motos> <moto cilindradas = "250">
3   <marca>Yamaha</marca> <modelo>Fazer</modelo>
4   <opcional>Freio Disco</opcional>
5   <revisao> <km>15000</km><valor>350,00</valor>
6     <km>35000</km><valor>600,00</valor> </revisao>
7   </moto>
8   <moto>
9     <marca>Honda</marca> <modelo>CBX</modelo>
10    <opcional>Abs</opcional> <opcional>Alerta</opcional>
11    <revisao>
12      <km>1000</km><valor>17,00</valor>
13      <km>5000</km><valor>35,00</valor>
14      <km>10000</km><valor>320,00</valor> </revisao>
15    </moto>
16    <moto cilindradas="125">
17      <modelo>Biz</modelo> <ano_modelo>2013</ano_modelo>
18    </moto>
19 </motos>
```

Figura 1. Exemplo de documento XML

2.1. Inferência Gramatical

O método descrito em [Chidlovskii 2001] é baseado em inferência gramatical, ou seja, dado um conjunto de entrada é inferida uma gramática que deve descrever todo o conjunto de dados. Este método consiste na execução de 3 passos: (i) os documentos são

representados em forma de uma árvore, (ii) é feita a indução de uma gramática com base na árvore gerada no primeiro passo e (iii) a gramática é transformada em um esquema na linguagem XML-Schema (XSD). O documento X (Figura 1) é utilizado para ilustrar o funcionamento do Algoritmo 1 que apresenta o funcionamento do método.

Algoritmo 1: Inferência Gramatical

```

1 Entrada : documento XML  $X$  ;
2 arvoreDerivacao ← criarArvore( $X$ ) ;
3 eCFG ← infereGramaticaInicial(arvoreDerivacao) ;
4 juncaoNaoTerminaisPorContexto(eCFG) ;
5 juncaoNaoTerminaisPorConteudo(eCFG) ;
6 foreach regra in eCFG do
7   | foreach elemento in regra.producao do
8     |   | if elemento.valor == "Any" then
9     |   |   | extraiTipoPrimitivo(elemento, Entrada) ;
10    |   | end
11   | end
12 end
13 esquema = converteEcfgParaXMLSchema(eCFG) ;

```

No primeiro passo é feita a conversão de X para uma representação em árvore (linha 2). A árvore gerada será uma árvore de derivação de uma gramática livre de contexto sem os seus símbolos não terminais. A árvore t_1 da Figura 2 representa a árvore gerada a partir de X . Para maior clareza algumas subárvore de t_1 foram omitidas. Após a conversão, o segundo passo do método é dividido em quatro etapas:

(i) Na primeira etapa é gerada uma gramática livre de contexto extendida (*eCFG - extended Context Free Grammar* [Brüggemann-Klein and Wood 1998]) (linha 3). Nesta etapa, t_1 é utilizada para criar as regras da gramática. Inicialmente, uma regra denominada *Start* que representa o elemento raiz da árvore é criada. Para cada um dos elementos presentes em t_1 são criadas regras. O nome da regra será da forma A_n , sendo n o identificador do número da regra. As produções contidas na regra representam os filhos que o elemento pode possuir na árvore. Assim, elementos complexos (possuem filhos) tem seus valores representados por rótulos de uma nova regra e elementos simples por *Any*. Após executado o método, a variável *eCFG* (linha 3) receberá o conjunto das regras contidas na Figura 3(a). Por exemplo, o elemento *revisao* de t_1 possui filhos então é gerada a regra A_5 .

(ii) Nesta etapa são feitas as junções das regras que possuem o mesmo contexto, afim de eliminar ambiguidades (linha 4). O método irá comparar as regras e a forma que estas estão sendo utilizadas e fará a junção. Levando em consideração a gramática contida na variável *eCFG* (Figura 3(a)), as regras A_1 , A_2 e A_3 estão sendo usadas no mesmo contexto (regra *Start*), sendo assim as regras A_2 e A_3 serão fusionadas à regra A_1 formando uma única regra.

(iii) Em seguida, são realizadas as junções das regras verificando se possuem produções semelhantes para tentar uní-las em uma única regra. As regras selecionadas são as regras A_4 e A_5 . A produção da regra A_5 é incorporada à produção A_4 concatenando-as utilizando o operador "*ou*". Ao final desta etapa a variável *eCFG* possui apenas as regras *Start* (regra inicial), A_1 (fundida com as regras A_1 , A_2 e A_3) e A_4 (adicionada com a regra A_5).

(iv) Na quarta etapa é realizada a extração de tipos básicos dos elementos simples (linhas 6 a 12). Para realizar esta etapa as produções de todas as regras são varridas, sendo selecionados os elementos que possuem como valor o terminal *Any* (elementos simples) e passados para a função *extraiTipoPrimitivo* (linha 9) que analisa o documento de entrada e infere através de tentativa e erro um tipo para o elemento. Ao fim desta parte a variável *eCFG* possui o conteúdo apresentado na Figura 3(b)

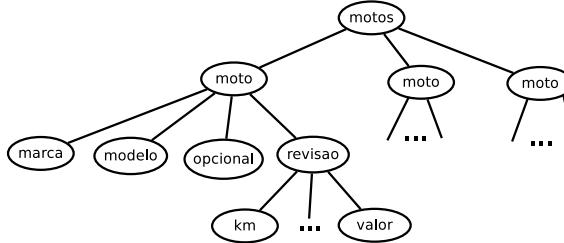


Figura 2. Árvore de derivação gerada a partir de *X*

O último passo do método faz, através de uma série de regras, a conversão da *eCFG* em uma XSD. A Figura 4 apresenta o esquema gerado a partir da *eCFG* apresentada na Figura 3(b).

(a) Inferência inicial das regras

```

Start ← moto : A1 moto : A2 moto : A3
A1 ← marca : Any modelo : Any opcional : Any revisao : A4
A2 ← marca : Any modelo : Any opcional : Any opcional : Any revisao : A5
A3 ← modelo : Any ano_modelo : Any
A4 ← km : Any valor : Any km : Any valor : Any
A5 ← km : Any valor : Any km : Any valor : Any km : Any valor : Any
  
```

(b) Após a execução do segundo passo do algoritmo

```

Start ← moto : A1 moto : A1 moto : A1
A1 ← marca : String modelo : String opcional : String revisao : A4 |
marca : String modelo : String opcional : String opcional : String revisao : A4 | modelo :
String ano_modelo : Int
A4 ← km : Int valor : Float | km : Int valor : Float km : Int valor : Float
  
```

Figura 3. Estruturas intermediárias para a construção do esquema.

2.2. XTRACT

O XTRACT [Garofalakis et al. 2000] é um método de extração que constrói uma DTD que representa a estrutura do documento XML dado como entrada. O Algoritmo 2 apresenta o seu pseudo-código e é utilizado para explicar o funcionamento do método tendo *X* com entrada.

O método pode ser resumido em três etapas principais: (i) o documento XML é transformado em um conjunto de sequências de símbolos: para cada elemento são geradas sequências que representam a ocorrência de seus sub-elementos, a partir dessas sequências, são construídas expressões regulares candidatas (*ERc*), (ii) *ERc* são otimizadas e simplificadas através do processo de fatoração, e (iii) do conjunto gerado é escolhida uma das candidatas para compor a DTD final, essa escolha é feita baseada no princípio do *Minimum Description Length* (MDL) que seleciona a candidata que melhor descreve estruturalmente a sequência no menor tamanho possível (em *bits*) [Vitanyi and Li 2000].

```

1 <?xml version="1.0"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="motos">
4     <xs:complexType>
5       <xs:element name="moto" minOccurs="1" maxOccurs="3" >
6         <xs:complexType>
7           <xs:element name="marca" type="xs:string"/>
8           <xs:element name="modelo" type="xs:string"/>
9           <xs:element name="opcional" type="xs:string" maxOccurs="2"/>
10          <xs:element name="revisao">
11            <xs:complexType>
12              <xs:sequence>
13                <xs:element name="km" type="xs:int"/>
14                <xs:element name="valor" type="xs:float"/>
15              </xs:sequence>
16            </xs:complexType>
17          </xs:element>
18        <xs:element name="ano_modelo" type="xs:int"/>
19      </xs:complexType>
20    </xs:element>
21  </xs:complexType>
22 </xs:element>
23 </xs:schema>

```

Figura 4. XSD criada utilizando o método de Inferência Gramatical

Inicialmente, é feita a busca dos elementos de X (linha 2). Cada tag de X é considerada um elemento. Sendo assim, a variável *elementos* receberá todos os elementos de X (i.e. *motos*, *moto*, *marca*, *modelo*, *opcional*, *revisao*, *km*, *valor*, *ano_modelo*). Para cada um dos elementos de *elementos* (linha 3) é executada a extração de sequências (linha 4). A extração das sequências, basicamente, é o processo de retirada dos elementos contidos em um determinado elemento. Por exemplo para o elemento *motos* a sequência correspondente é $\{moto\ moto\ moto\}$, pois a tag *motos* contém 3 elementos *moto*, e aparece uma vez. Assim, a variável *seqElementos* recebe $\{moto\ moto\ moto\}$ (linha 4).

Para cada sequência de elementos de *seqElementos* serão construídas ERc a elementos da DTD (linhas 5 a 11). Essa etapa do método é chamada de generalização. Inicialmente, a sequência é adicionada ao vetor de candidatas (linha 6), então o método *geraCandidataE* é invocado 3 vezes variando o parâmetro r em 2, 3 e 4 que representa o número de vezes que um símbolo pode se repetir. *geraCandidataE* varre a sequência com o intuito de criar expressões regulares que correspondam a estrutura do elemento candidato. Assim, a primeira candidata c_1 a ser gerada ($r = 2$) será $moto^*$. c_1 é passada para o método *geraCandidatasOu* que tentará gerar outras candidatas baseadas em c_1 , utilizando o operador *ou*. Todas as candidatas geradas são adicionadas ao vetor *candidatas*. Passando c_1 como parâmetro para o método, será obtida como resposta a candidata a própria c_1 pois não é possível obter mais candidatas utilizando o operador *ou*. Outra candidata gerada pelo método *geraCandidataE* (com $r = 3$) é $moto\ moto^*$, que ao passar pelo método *geraCandidataOu* não sofreria alterações. $r = 4$ não gera nenhuma nova candidata. Sendo assim, ao final da etapa de generalização para a sequência $\{moto\ moto\ moto\}$ seriam geradas as candidatas: $\{moto\ moto^*,\ moto^*\}$. Para a maior parte das candidatas geradas a partir dos elementos de X , não serão feitas alterações ao passar pelo método *geraCandidataOu* exceto a candidata $(km\ valor)^*$, onde o método retornará as candidatas $(km\ valor)^*$ e $(km\ | valor)^*$.

A próxima etapa a ser executada é a fatoração onde as candidatas geradas são submetidas a processos de simplificação (linha 12). Basicamente, o sistema de fatoração

Algoritmo 2: XTRACT

```
1 Entrada : documento XML  $X$  ;
2  $elementos \leftarrow extraiElementos(X)$  ;
3 foreach elemento in elementos do
4     seqElementos  $\leftarrow extraiSeqElementos(elemento)$  ;
5     foreach sequencia in seqElementos do
6         adiciona sequencia em candidatas;
7         for r in (2,3,4) do
8             geraCandidataE(sequencia, r, candidata) ;
9             geraCandidatasOu(candidata,candidatas) ;
10        end
11    end
12    fatora(candidatas) ;
13    menor  $\leftarrow MDL(candidatas[0])$  ;
14    DTD  $\leftarrow candidatas[0]$ ; i  $\leftarrow 1$ ;
15    while  $i++ < tamanho(candidatas)$  do
16        if  $MDL(candidatas[i]) < menor$  then
17            menor  $\leftarrow MDL(candidatas[i])$  ; DTD  $\leftarrow candidatas[i]$ ;
18        end
19    end
20    adiciona DTD em DTDFinal;
21 end
22 return DTDFinal
```

varre todas as candidatas e simplificá-as, podendo até criar, unindo ou construindo, novas candidatas. Passando para o módulo de fatoração as candidatas obtidas anteriormente, *moto moto** será eliminada pois após fatorada ficará igual a candidata *moto**. Ao fim desta etapa, a variável *candidatas* terá somente a candidata *moto**. Para a maior parte dos conjuntos de candidatas que serão geradas ao longo da execução do exemplo, a etapa de fatoração não fará grandes mudanças, exceto quando esta receber as candidatas { *marca modelo*, *modelo ano.modelo* }, que resultará a candidata (*marca | modelo | ano.modelo*).

A última etapa a ser executada é a eleição da candidata que participará dos elementos que constituem a DTD final. Esta eleição é baseada no princípio do MDL. Como a variável *candidatas* contém apenas uma candidata então está será a eleita para fazer parte da DTD final. Sendo assim, será calculado o MDL para a primeira candidata que estiver no vetor de candidatas (linha 13 - *i.e.* *moto**, que é 32 (pois *moto** ocupa 5 bytes para ser armazenada mais 27 bytes dele convertido em elemento da DTD). Como não existem mais candidatas, a candidata é descrita em termos de DTD e é adicionada a DTD final (linhas 14 e 20 respectivamente). O elemento resultante deste passo é apresentado na Figura 5 (linha 2). Para a maior parte dos elementos de X apenas uma candidata será passada para o módulo do MDL. Exceto para as candidatas (*km valor*)*, (*km | valor*)* para o elemento *revisao*. A eleita entre essas será (*km valor*)* pois possui um custo de MDL menor que sua concorrente.

O processo é repetido para todos os elementos que foram extraídos do documento de entrada. Após a execução de todas as etapas para todos os elementos, a DTD obtida será a apresentada pela Figura 5. Percebe-se que o método faz a extração de um esquema que descreve a estrutura do documento XML utilizado como entrada, porém não se preocupa com a extração de atributos e de tipos básicos dos elementos.

```

1 <!DOCTYPE motos [
2 <!ELEMENT motos (moto*)>
3 <!ELEMENT moto ((marca | modelo | opcional | revisao | ano_modelo)*)>
4 <!ELEMENT marca (#PCDATA)>
5 <!ELEMENT modelo (#PCDATA)>
6 <!ELEMENT opcional (#PCDATA)>
7 <!ELEMENT revisao (km, valor)>
8 <!ELEMENT km (#PCDATA)>
9 <!ELEMENT valor (#PCDATA)>
10 <!ELEMENT ano_modelo (#PCDATA)> ]>

```

Figura 5. DTD criada utilizando o Xtract

2.3. XTLSM

O método apresentado em [Amiel et al. 2008] é baseada em um novo tipo de máquina de estado a TLSM (*two layer state machine*) usada para validar gramáticas de árvore. Cada estado da TLSM contém uma máquina de estados regular que descreve os filhos de um elemento da coleção de documentos XML. O método propõe 4 etapas para a extração do esquema: (i) é feita a criação de um reconhecedor de árvores de prefixos (PTA - *prefix tree acceptor*) a partir de um documento XML, (ii) a PTA é convertida em uma máquina de estados combinando árvores semelhantes, (iii) através da máquina de estados criada na etapa anterior é criada uma TLSM, e (iv) a TLSM é convertida para uma XSD. O Algoritmo 3 apresenta um pseudocódigo do método e é utilizado na aplicação do estudo de caso utilizando X como entrada.

Algoritmo 3: XTLSM

```

1 Entrada : documento XML  $X$ ;
2 PTA  $\leftarrow$  criaPTA( $X$ ,  $X.root$ );
3 fsm  $\leftarrow$  generaliza(PTA,  $X$ );
4 foreach estado in fsm do
5   estadoInterno  $\leftarrow$  novo estado inicial;
6   foreach nodo in PTA do
7     filho  $\leftarrow$  filhoDireita(nodo);
8     while filho do
9       if (estadoInterno, filho) não mapeado then
10         if  $\exists$  filho transição na TLSM then
11           addTLSMtransição (estadoInterno, filho);
12         else
13           criaEstadoInterno (e, filho);
14           estadoInterno  $\leftarrow$  e;
15         end
16         adicionaMapeado (estadoInterno, filho);
17       end
18       filho  $\leftarrow$  irmãoEsquerda(filho);
19     end
20   end
21 end
22 xsd  $\leftarrow$  converteTLSMParaXSD (tlsm);

```

Inicialmente, o algoritmo transforma X em uma PTA (Figura 6). Para tal, X é visto como uma árvore (linha 2). O método *criaPTA* recebe dois parâmetros: X é a raiz de X (*i.e.*, *motos*). Ao fim desse passo a variável *PTA* armazena a PTA gerada (Figura 6).

Onde as transições são rotuladas pelo nome do elemento, os círculos representam estados e os círculos duplos representam estados de reconhecimento.

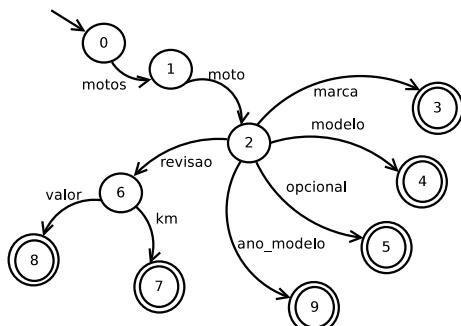


Figura 6. PTA gerada a partir do documento X

O algoritmo então, transforma a PTA em uma máquina de estados (linha 3). Essa transformação é feita realizando a junção de subárvores contidas na PTA levando em consideração suas semelhanças e contexto. Como a PTA gerada pelo exemplo não possui subárvores semelhantes, ela não sofrerá mudanças sendo somente transportada da variável *PTA* para *fsm* (linha 3). Sendo assim, ao fim do processo a máquina gerada será a mesma apresentada pela Figura 6.

O próximo passo a ser executado cria a TLSM (linhas 4 a 21). Durante este passo, a máquina de estados criada será transformada em uma TLSM. A TLSM se diferencia de uma máquina de estados normal por possuir estados com máquinas de estados internas. O XTLSM faz uso das máquinas de estado internas para realizar a validação dos filhos de um elemento presente no documento.

Para criar a TLSM, é feita uma análise de todos os estados presentes na máquina de estados existente (*fsm* - linha 4) em conjunto com os nodos presentes no documento de entrada (*i.e.*, *X*). Sendo assim, em cada estado da máquina é verificado se o nodo da transição possui filhos, caso existam, é criada uma nova camada, ou seja, uma máquina de estados interna e são adicionadas as transições que validam a ordem em que os filhos deverão aparecer. Seguindo a execução do exemplo, o estado selecionado será o estado inicial com o primeiro nodo (*motos*). Na linha 7, a variável *filho* recebe o filho mais a direita do elemento *motos* (*i.e.*, *moto*). Após verificado que o *estadoInterno* (agora um estado inicial) com o *filho* (*moto*) ainda não foi mapeado - linha 9 - e que não existe um estado para o filho na TLSM (linha 10), é criado um novo estado que é alvo de uma transição do estado *estadoInterno* pelo elemento *filho*, e o novo valor do *estadoInterno* é o estado criado (linhas 13, 14 e 16). Então, é selecionado o irmão a esquerda do *filho*, que é novamente *moto*. É criada uma transição para o estado *estadoInterno* por *moto* (linha 11). As camadas da TLSM geradas por esses passos estão representadas na Figura 7. As três máquinas geradas (Figura 7 (1), (2) e (3)) são utilizadas para criar as regras do esquema que representa *X*.

A última etapa do método consiste em transformar a TLSM em uma XSD (linha 22). Para isso, são propostos alguns algoritmos de redução da TLSM que simplificam e reduzem a TLSM gerada, então através de força bruta a TLSM é transformada em uma XSD. A XSD gerada é muito semelhante a XSD gerada pelo método de Inferência Gramatical (Figura 4). A única diferença entre as XSDs será nas linhas 5 e 9 (Figura 4)

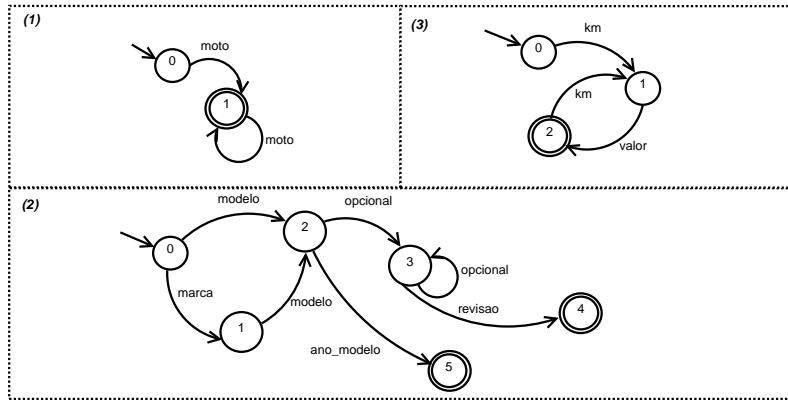


Figura 7. Camadas internas criadas a partir da TLSM criada no estudo de caso.

onde o atributo *maxOccurs* que apresenta os valores 3 e 2 respectivamente, serão alterados para *unbounded*, já que o XTLSM não limita o número de repetições máximas de um elemento.

2.4. Considerações Finais

As abordagens apresentadas realizam a extração da estrutura de um documento XML, ou seja, um esquema que descreve a estrutura de uma coleção de documentos XML. A Tabela 1 apresenta algumas das características dos métodos apresentados. Duas das abordagens (*Inferência Gramatical* e *XTRACT*) criam uma gramática que representa a estrutura do documento e a converte para uma XSD ou para DTD (coluna *Rep. Intermediária*). Já o XTLSM cria uma máquina de estados multinível que contém a estrutura do documento, então, através de força bruta converte essa máquina em uma XSD.

Apesar das abordagens de *Inferência Gramatical* e *XTRACT* possuírem uma semelhança (ambas extraem um gramática que armazena a estrutura do documento) a forma que realizam a extração é distinta. O *XTRACT* extrai para cada elemento presente no documento de entrada uma série de sequências, para cada sequência é criada uma regra. O conjunto de regras é submetido a uma série de processos que irão simplificá-las e então uma destas é escolhida para fazer parte de gramática que representa todo o documento. Em contrapartida, o método de *Inferência Gramatical* interpreta o documento em forma de árvore e infere para a árvore uma eCFG. A eCFG gerada é submetida a um processo de simplificação que leva em consideração o contexto e o conteúdo das regras para gerar uma gramática mínima.

O XTLSM propõe uma abordagem muito diferente das demais apresentadas. Apesar das demais representarem a estrutura do documento de entrada através de algum tipo de gramática, o XTLSM usa uma máquina de estados multinível para esta tarefa. Assim, o XTLSM inicialmente realiza a extração de uma PTA que é transformada em uma máquina de estados. A máquina de estados gerada apresenta apenas a hierarquia do documento. Então ela é transformada em uma TLSM onde são adicionados estados internos que validam a ordem em que os elementos devem aparecer no documento.

Pela coluna *Esquemas*, percebe-se que todas as abordagens transformam as estruturas intermediárias em linguagens de esquemas XML (duas delas em XSD). Percebe-se também que nenhuma das abordagens trata atributos e apenas o *XTRACT* não extrai os tipos, transformando todos os elementos simples em textuais (*string*).

Método	Rep. Intermediária	Esquema	Atributos	Tipos
XTRACT	ER	DTD	-	-
Inferência Gramatical	eCFG	XSD	-	X
XTLSM	Máquina de Estados	XSD	-	X

Tabela 1. Características métodos de extração de esquemas.

3. Conclusão

Este artigo apresentou um estudo de algumas abordagens para extração de esquemas XML a partir de um coleção de documentos. A extração de esquemas é importante pois conhecendo a estrutura de uma classe de documentos XML é possível otimizar consultas e armazenamento dos dados, criar índices, entre outros. Atualmente, na Web, existem várias coleções de documentos XML que não estão associadas a esquemas, por isso a importância de abordagens eficientes e corretas para a extração. Percebe-se que a maioria dos métodos apresentados utiliza conceitos de linguagens formais para a extração de esquema. Isso é esperado já que um documento XML pode ser representado por uma árvore e uma árvore pode ser gerada por uma gramática de árvore. Os leitores encontrarão em [Schreiner 2014] uma discussão mais detalhada sobre os métodos de extração de esquemas XML. Não é do conhecimento dos autores outros trabalhos que façam comparações entre métodos de extração de esquemas de documentos XML.

Espera-se que com este estudo seja possível propor novos métodos de extração de esquemas bem como aplicar os métodos aqui vistos em outros domínios similares: documentos JSON e RDF.

Referências

- Amiel, S., Harrusi, S., and Averbuch, A. (2008). XML Schema Inference from Positive Example: The TLSM Approach. Technical report, Tel Aviv University.
- Barbosa, D., Mignet, L., and Veltri, P. (2005). Studying the XML Web: gathering statistics from an XML sample. *World Wide Web*, pages 1–32.
- Brüggemann-Klein, A. and Wood, D. (1998). Regular tree languages over non-ranked alphabets.
- Chidlovskii, B. (2001). Schema Extraction from XML Data: A Grammatical Inference Approach. *KRDB'01 Workshop (Knowledge Representation and Databases)*.
- Garofalakis, M., Gionis, A., and Rastogi, R. (2000). XTRACT: a system for extracting document type descriptors from XML documents. *ACM SIGMOD*, pages 165–176.
- Hegewald, J., Naumann, F., and Weis, M. (2006). Xstruct: Efficient schema extraction from multiple and large xml documents. In *Proceedings of the ICDEW '06*. IEEE Computer Society.
- Min, J.-K., Ahn, J.-Y., and Chung, C.-W. (2003). Efficient extraction of schemas for XML documents. *Information Processing Letters*, 85(1):7–12.
- Schreiner, G. A. (2014). Extração de esquemas de documentos XML: Uma abordagem probabilística. TCC, Universidade Federal da Fronteira Sul.
- Vitanyi, P. and Li, M. (2000). Minimum description length induction, bayesianism, and kolmogorov complexity. *Information Theory, IEEE Transactions on*, 46(2):446–464.

Anotação de Trajetórias via Fusão com Trilhas de Mídias Sociais

Ricardo Gil Belther Nabo, Renato Fileto
Cleto May, Lucas André de Alencar

¹Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis, Santa Catarina, Brazil

Resumo. *O aumento da utilização de dispositivos móveis tem ocasionado um grande crescimento na geração de trajetórias de objetos móveis. Entretanto somente as trajetórias muitas vezes não são suficientes para permitir a análise semântica dos movimentos. Junto ao crescimento da utilização de dispositivos móveis também aumentou a utilização de mídias sociais remotamente, onde postagens dos usuários podem ser vistas como rastros esparsos e anotados de seu movimento. Este trabalho propõe um método para fusão de trajetórias com dados provenientes de mídias sociais. Os resultados são coleções de trajetórias anotadas com texto inserido em mídias sociais. O método é implementado e avaliado em experimentos com trajetórias reais de postagens de usuários do Twitter, efetuadas na mesma região geográfica onde ocorreram as trajetórias.*

Abstract. *The increased use of mobile devices has led to a large increase in the moving objects trajectories generation. However, only the trajectories are often not sufficient to allow the movements semantic analysis. The increasing use of mobile devices has also increased the remotely social media use, where users' posts can be viewed as sparse and annotated traces of their movement. This paper proposes a method for fusing trajectories with data from social media. The results are collections of trajectories annotated with texts inserted in social media. The method is implemented and evaluated in experiments with real trajectories and postings of Twitter users, conducted in the same geographic region where the trajectories occurred.*

1. Introdução

A utilização de dispositivos móveis que permitem a coleta de coordenadas espaciais (eg., GPS, *smartphones*, *tablets*) tem tido um crescimento considerável nos últimos anos. Este crescimento tem acarretado a geração de grandes volumes de dados de trajetórias brutas.

Muitos trabalhos relacionados a mineração de padrões espaço-temporais vem sendo desenvolvidos na literatura. Dentre eles, estruturação de trajetórias [Spaccapietra et al. 2008, Xiu-li and Wei-xiang 2009] e anotação de trajetórias [Alvares et al. 2007, Yan et al. 2012]. A anotação de trajetórias é importante, pois somente os dados espaço-temporais das trajetórias geralmente não são suficientes para o enriquecimento semântico. As trajetórias brutas coletadas por dispositivos móveis (e.g., *smartphones*) quase sempre carecem de dados textuais. Por outro lado, dados que às vezes são coletados pelos mesmos dispositivos móveis e inseridos em mídias sociais (e.g.,

tweets, posts no Facebook) possuem informações textuais (e.g., comentários, *hashtags*) que podem ajudar a descrever e analisar semanticamente as trajetórias.

Este trabalho propõe um método para realizar a fusão de trajetórias brutas com *posts* de usuários em mídias sociais, utilizando como critério da fusão as coordenadas espaciais e os instantes de coleta de pontos de trajetórias. O método é validado utilizando uma base de dados de trajetórias brutas e dados da mídia social Twitter coletados no mesmo local.

O restante deste trabalho está organizado da seguinte maneira. A seção 2 define alguns conceitos fundamentais para a compreensão do trabalho. A seção 3 apresenta o método proposto. A seção 4 descreve e discute experimentos para validar o método. A seção 5 discute e compara este trabalho com outros trabalhos relacionados. Finalmente, a seção 6 apresenta as conclusões e trabalhos futuros.

2. Fundamentação

Esta seção apresenta alguns fundamentos e definições utilizados na descrição formal do problema abordado e do método de solução proposto neste artigo.

2.1. Trajetórias

Trajetórias brutas são sequências temporalmente ordenadas de coordenadas espaço-temporais. Cada coordenada pode ser definida como um ponto.

Definição 1. (Ponto espaço-temporal) Coordenada espaço-temporal representada pela quádrupla: $\mathbf{P}(Pid, x, y, t)$, onde:

- Pid é o identificador do ponto;
- (x, y) é um par de coordenadas geográficas; e
- t é um instante de tempo.

Um dispositivo móvel que coleta amostras de localizações na forma de pontos espaço-temporais dentro de um determinado intervalo de tempo gera uma trajetória bruta.

Definição 2. (Trajetória Bruta - TB). Sequência temporalmente ordenada de pontos, $(Pid, x, y, t) (p_1, p_2, \dots, p_n)$ visitados por um objeto móvel, onde cada elemento desta sequência é representado pela tripla: **RawTraj**($MOid, Tid, Pj$), onde:

- $MOid$ é o identificador do objeto móvel;
- Tid é o identificador da trajetória; e
- P é a referência para um ponto espaço-temporal (Definição 1).

Visando melhorar o desempenho e os resultados produzidos pelo processamento de TBs, seus pontos são agrupados de acordo com alguma característica em comum entre eles. Os grupos de pontos resultantes são denominados episódios.

Definição 3. (Episódio). Subsequência maximal de pontos de uma trajetória que satisfazem um determinado predicado $(P_{inicial} \dots P_{final}) : \Rightarrow \{\text{true}, \text{false}\}$. Um episódio é representado pela quádrupla: **Episódio** ($Tid, Eid, ETtype, P_{inicial} \dots P_{final}$ ($1 \leq inicial \leq final \leq n$))), onde:

- Tid é o identificador da trajetória a quem o episódio pertence;
- Eid é o identificador do episódio;

- $Etype$ é o tipo do episódio (e.g. "stop", "move"); e
- $P_{initial} \dots P_{final}$ é a subsequência de pontos que constituem o episódio.

Como a lista de pontos pertencentes a um episódio é uma subsequência maximal, somente referências para o primeiro e o último ponto de cada episódio precisam ser armazenadas em sua estrutura. Os demais pontos podem ser recuperados diretamente da TB, desse modo não duplicando informações.

Os episódios, quando temporalmente ordenados, geram outro tipo de trajetória, uma trajetória estruturada. Cada elemento da trajetória estruturada é um episódio.

Definição 4. (Trajetória Estruturada - TE). Sequência temporalmente ordenada de episódios não aninhados. Cada elemento da sequência é representado pelo par: **StrTraj**($STid, Ei$), onde:

- $STid$ é o identificador da trajetória estruturada; e
- Ei é um episódio.

2.2. Dados de Movimentos colhidos em Mídias Sociais

Uma pegada de mídia social é o registro de uma interação entre um usuário e uma mídia social (e.g., Twitter, Facebook, Foursquare). Quando o usuário posta algo a informação associada (e.g., posição espaço-temporal, foto) fica gravada na respectiva mídia (e.g., Twitter, Facebook) e acessível via APIs específicas de cada mídia. Uma sequência temporalmente ordenada de pegadas constitui uma trilha.

Definição 5. (Pegada de Mídia Social). Registro em um sistema de mídia social de interação efetuada por um usuário, representado pela quíntupla: **SMF**($MOid, SMFid, SMid, P, c$), onde:

- $MOid$ é o identificador do objeto móvel;
- $SMFid$ é o identificador da pegada;
- $SMid$ é o identificador da mídia social da pegada (e.g., Twitter, Facebook);
- P é um ponto espaço-temporal como descrito na Definição 1;
- c são os conteúdos das pegadas (e.g., tags, imagens, textos);

Definição 6. (Trilha de Mídia Social - TMS). Sequência temporalmente ordenada de pegadas de mídia social, do mesmo usuário. Cada elemento desta sequência é representado pela dupla: **SMT**($SMTid, SMF$), onde:

- $SMTid$ é o identificado da trilha de mídia social; e
- SMF é a referência a uma pegada de mídia social.

Embora análogas em termos de estruturas de dados, TBs e TMSs são diferentes. Trajetórias usualmente têm melhor precisão espaço-temporal que trilhas. A amostragem de pontos de TBs usualmente é realizada em intervalos fixos e curtos (e.g., 10 segundos, 10 metros). Por outro lado, as postagens em mídias sociais são assíncronas (o usuário decide quando postar) e usualmente esparsas, além das TMSs possuírem anotações de usuários.

2.3. Descrição do Problema

Considere um conjunto de TBs (TB_DB) e um conjunto de TMSs (TMS_DB), como o ilustrado na Figura 1. Se considerarmos que alguns objetos moveis que geraram TBs (Definição 2) podem ser os mesmos (ou ao menos ter movimentos e intenções afins) àqueles que geraram TMSs (Definição 6), é possível fundir algumas trajetórias com trilhas. Entretanto para realizar essa fusão, é necessária a utilização de uma medida de correlação entre uma TB e uma TMS, tais como correlação espaço-temporal.

O desafio deste trabalho é desenvolver um método para fundir trajetórias provenientes de dispositivos GPS com TMSs. Em outras palavras, o problema é determinar pares trajetória-trilha que são espacial e temporalmente correlacionados. Por exemplo, na Figura 6, as trajetórias horizontais e verticais (representadas por linhas contínuas), podem corresponder às respectivas trilhas horizontais e verticais (representadas por linhas pontilhadas). Tal correlação pode indicar: (i) trajetória e trilha de cada par geradas pelo mesmo dispositivo; (ii) geradas pelo mesmo objeto móvel (e.g. pessoa) portando dispositivos distintos; (iii) pares trajetória-trilha com movimentos análogos (e.g., uma pessoa usando um smartphone para acessar mídia social de dentro de um veículo equipado com GPS).

O método proposto neste trabalho utiliza proximidade espaço-temporal entre episódios e pegadas, das respectivas trajetórias e trilhas, para medir tal correlação e calcular os coeficientes de correspondência de pares trajetória-trilha. Considera-se que a segmentação adequada das trajetórias foi previamente realizada para suportar a investigação das correspondências. A determinação de quais objetos móveis (referentes a trajetórias) correspondem a quais usuários de redes sociais (que geram trilhas) também está fora do escopo deste trabalho, sendo deixada para trabalhos futuros.

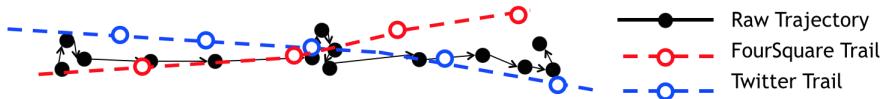


Figura 1. Ilustração do Problema

3. Método de Fusão Proposto

Esta seção apresenta um método para anotação de trajetórias mediante sua fusão com dados de mídias sociais. Inicialmente é apresentado o processo geral proposto, depois alguns detalhes de tarefas específicas deste processo.

3.1. Um Processo para Fusão de Trajetórias com Trilhas de Mídias Sociais

O método proposto neste trabalho utiliza as definições descritas na seção 2. A figura 2 apresenta o processo proposto, o qual é composto de três fases: pré-processamento, compressão de trajetórias e fusão de trajetórias com trilhas. Todas as fases são flexíveis quanto aos métodos utilizados para sua implementação.

A fase de pré processamento das TBs consiste da limpeza e estruturação dessas trajetórias. A fase de compressão comprime os dados de trajetórias em uma representação que possa ser analisada de forma menos custosa que os dados brutos ou a mera agregação em episódios. A fase de fusão, foco principal deste trabalho, consiste na computação das probabilidades de correspondências entre pares trajetória-trilha. Os coeficientes de

correspondência local (CCL) de pares (episódio-trilha) espaço-temporais próximos são usados para computar o coeficiente de correspondência global (CCG) da respectiva trajetória com a trilha. Ao final, os pares (trajetória-trilha) com mais alto CCG são selecionados, fundidos e ordenados de acordo com seu respectivo CCG.

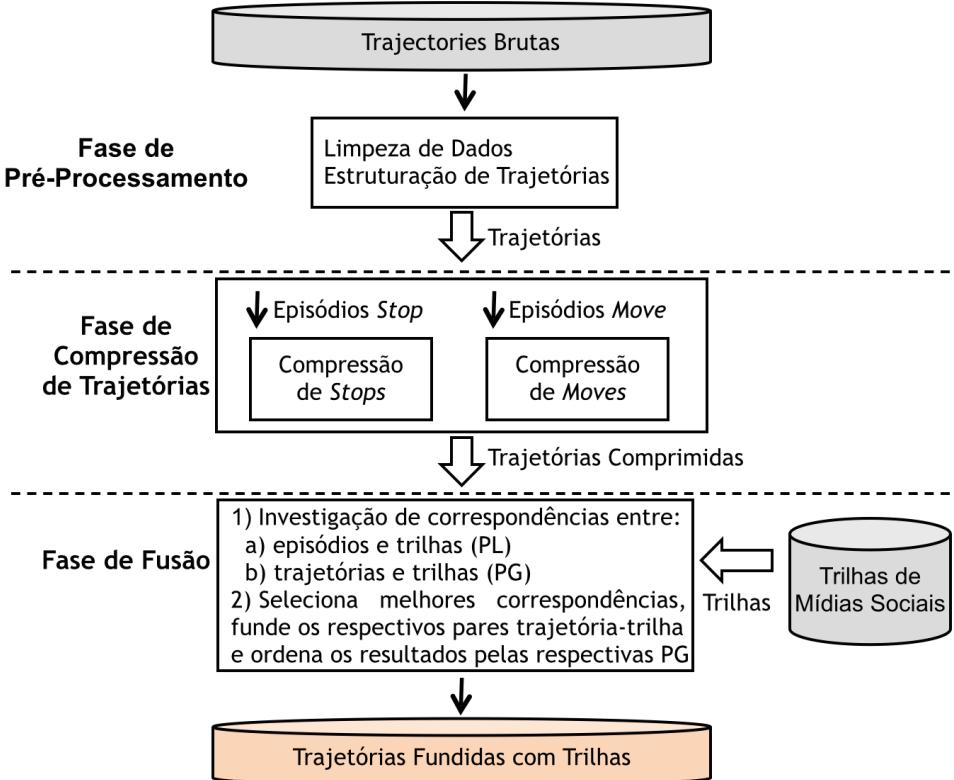


Figura 2. Fases do método proposto

3.1.1. Investigando Correspondências entre Episódios e Pegadas

O Coeficiente de Correspondência Local (CCL) é calculado para cada episódio de uma TE, verificando se há pegadas de mídias sociais dentro de um determinado *buffer* espaço-temporal ao redor de um episódio. Se há pegadas de uma trilha dentro deste *buffer* a chance da respectiva trilha e trajetória estarem correlacionada aumenta. A equação 1 apresenta o cálculo do CCL entre um episódio e as pegadas de uma mesma trilha que estão dentro do *buffer* espaço-temporal em torno do respectivo episódio.

$$CCL(Ep_i, SMT_j) = \frac{|SMF|}{|ALLFP|} \quad (1)$$

Ep_i é o episódio, SMT_j é a trilha em que se está calculando o coeficiente, SMF é o conjunto de pegadas da SMT_j que estão dentro do *buffer* espaço-temporal em torno de Ep_i e ALLFP é o conjunto de pegadas de mídias sociais de toda a base de trajetórias que estão dentro do *buffer* espaço-temporal para este episódio.

A Figura 3 exemplifica o cálculo da CCL para duas TMSs (SMT_1 e SMT_2) e uma TE ($StrTraj_1$). As duas TMSs possuem pegadas dentro do *buffer* espaço-temporal

em torno de Ep_1 . Portanto são candidatas a serem fundidas. Para calcular o CCL entre primeiro episódio (Ep_1) e SMT_2 inicialmente verificamos a quantidade de pegadas que estão dentro do *buffer* do Ep_1 e pertencem a SMT_1 , ou seja $|SMF| = 1$. O próximo passo é verificar quantas pegadas estão dentro do *buffer* em torno de Ep_1 e pertencem a todas as trilhas candidatas, ou seja $|ALLFP| = 2$. Substituindo estes valores na formula temos: $P(Ep_1, SMT_1) = \frac{1}{2} = 0,5 = 50\%$. Repetindo os mesmos passos para SMT_2 , temos: $P(Ep_1, SMT_2) = \frac{1}{2} = 0,5 = 50\%$. Portanto, as duas trilhas tem a mesma chance de representar a $StrTraj_1$, levando em consideração somente o Ep_1 .

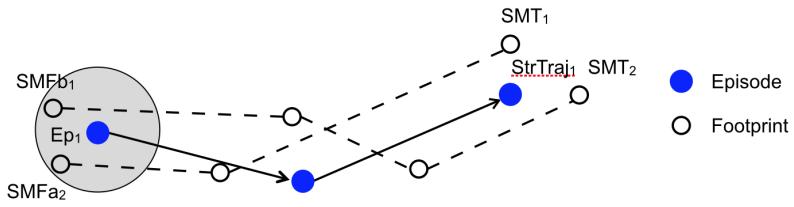


Figura 3. Representação do cálculo do Coeficiente de Correspondência Local

3.1.2. Investigando Correspondências entre Trajetórias e Trilhas

Após calcular o CCL para as base de dados, precisamos compor o Coeficiente de Correspondência Global (CCG) que os usuários que geraram uma trajetória e uma TMS serem a mesma pessoa. O CCG é composto do somatório de todos os CCL que uma trilha possui para uma determinada trajetória dividido pelo número total de episódios da trajetória, a formula utilizada para o cálculo da CCG pode ser observada na Equação 2.

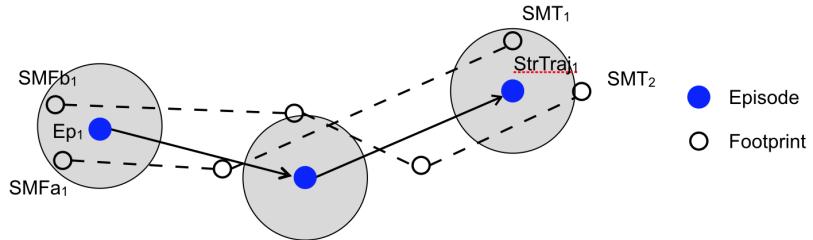


Figura 4. Representação do cálculo da Coeficiente de Correspondência Global

$$CCG(StrTraj_i, SMT_j) = \frac{\sum_{k=1}^n CCL(Ep_k, SMT_j)}{n} \quad (2)$$

$StrTraj_i$ é a TE que se esta calculando o CCG, SMT_j é a trilha em que se está calculado o CCG, $P(Ep_k, SMT_j)$ é o CCL entre Ep_k e SMT_j e n é total de episódios da $StrTraj_i$.

A Figura 4 exemplifica o cálculo do CCG para duas TMSs (SMT_1 e SMT_2) e uma TE ($StrTraj_1$). Para o cálculo de $CCG(StrTraj_1, SMT_1)$, é necessário o cálculo dos CCLs referentes a relação $StrTraj_1 SMT_1$, como visto na subseção anterior. Desse modo os CCLs para os episódios da $StrTraj_1$ são: $CCL(Ep_1, SMT_1) = \frac{1}{2}$,

$CCL(Ep_1, SMT_1) = 0$ e $CCL(Ep_1, SMT_1) = 0$. O número de episódios de $StrTraj_1$ (n) é 3. Portanto $CCG(StrTraj_1, SMT_1) = \frac{\sum_{k=1}^3 P(Ep_k, SMT_1)}{3} = \frac{\frac{1}{2}+0+1}{3} = \frac{\frac{3}{2}}{3} = \frac{1}{2} = 0.5 = 50\%$. A $StrTraj_1$ tem 50% de chance de ser representada pela SMT_1 .

Calculando do mesmo modo para a relação $StrTraj_1SMT_2$ temos:
 $CCG(StrTraj_1, SMT_2) = \frac{\sum_{k=1}^3 P(Ep_k, SMT_1)}{3} = \frac{\frac{1}{2}+0+0}{3} = \frac{\frac{1}{2}}{3} = \frac{1}{2} = \frac{3}{2} \approx 0.33 \approx 33\%$.

4. Experimento

O método apresentado na seção anterior foi parcialmente implementado, com o intuito de validar a ideia de fusão de TBs e TMSs proposta neste trabalho. Para realizar a validação do método foram realizadas algumas simplificações referentes ao método, devido a sua alta complexidade de implementação. Dentro deste contexto o experimento descrito nesta seção utiliza somente *stops* para fundir trajetórias com TMSs.

Este experimento tem como objetivo comparar de forma qualitativa os tipos de anotações produzidas pelo método proposto neste trabalho e outros trabalhos da literatura, tais como [Yan et al. 2012] e [Alvares et al. 2007]. As entradas de dados descritas pelo método são duas bases de dados, uma de TBs e outra de TMSs coletadas na mesma região geográfica durante o mesmo período de tempo. Entretanto a construção de bases de dados de TBs e TMSs não é uma tarefa trivial, desse modo visando validar a fusão em si iremos utilizar bases de dados coletadas na mesma região, mas em um período de tempo diferente.

A base de dados de TBs foi selecionada de um grande conjunto de dados de táxis coletados na região de Fortaleza, durante o período de 20/07/2012 à 20/10/2012. Para este experimento foram selecionados quatro motoristas de táxis e foi considerada que cada trajetória teria a duração de 24 horas, gerando o total de 357 trajetórias, uma para cada dia de trabalho dos quatro taxistas. As TBs utilizadas neste experimento só consideram o caminho percorrido pelos taxistas, quando há um passageiro dentro do táxi. A base de dados de TBs é ilustrada na Figura 5.

A base de dados de TMSs consiste na aquisição de trilhas da mídia social Twitter provenientes de 227 usuários, na região de Fortaleza durante o período de (31/12/2013 à 02/01/2014). As trilhas possuem combinadas 1436 pegadas de mídia social, e não são segmentadas. Portanto o tamanho máximo de uma TMS neste experimento é de três dias. A base de dados de mídias sociais pode ser observada na Figura 6, os pontos azuis representam as pegadas da mídia social Twitter e as linhas lilases representam a ligação entre as pegadas geradas pelos mesmos usuários.

É responsabilidade da fase de pré-processamento garantir que as bases de dados estejam limpas e prontas para a aplicação do resto do método, também é nesta fase em que é realizada a estruturação de TBs. Neste experimento é utilizado o algoritmo o algoritmo *Cluster Based Stops and Moves on Trajectories* (CB-SMoT) [Xiu-li and Wei-xiang 2009], ele é responsável por encontrar episódios do tipo *stop* e *move*, considerando *clusters* de variação de velocidade e a densidade dos pontos.

Com a estruturação das TBs completas a fase de pré-processamento é encerrada, e é dado inicio a fase de compressão de trajetórias. Como já dito os episódios do tipo *Move* não foram considerados na fase de fusão e portanto não foram comprimidos durante a fase de compressão de trajetórias. Por outro lado os episódios do tipo *stop* precisam

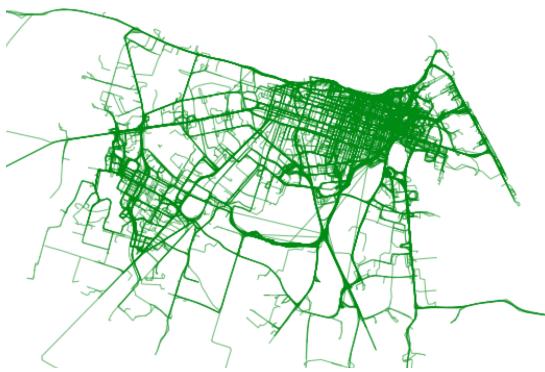


Figura 5. Base de dados de trajetórias brutas



Figura 6. Base de dados de trilhas de mídias sociais

ser comprimidos, para tal neste trabalho iremos calcular o centroide de cada episódio do tipo *stop*. Desse modo cada *stop* se transforma em um ponto. Na Figura 7 está exposto a base de dados de TEs somente com os *stops* das trajetórias, já representados pelos seus centroides. Os pontos amarelos representam os centroides dos episódios do tipo *stop* e as linha verdes representam a ligação entre os *stops* dos mesmos usuários.

Durante a fase de fusão é realizado o cálculo dos coeficientes de correlação locais e globais para todo o par trajetória-trilha das bases de dados, como descrito nas seções 3.1.1 e 3.1.2, entretanto como as bases de dados não são do mesmo período de tempo foi necessário realizar uma simplificação temporal para a fusão. Desse modo não foram utilizados parâmetros temporais para a realização dos cálculos das probabilidades locais e globais. A Figura 8 ilustra o cálculo das probabilidades locais e globais das bases de dados de TBs e de TMSs, a Figura também ilustra os *buffers* utilizados para o cálculo das probabilidades locais.

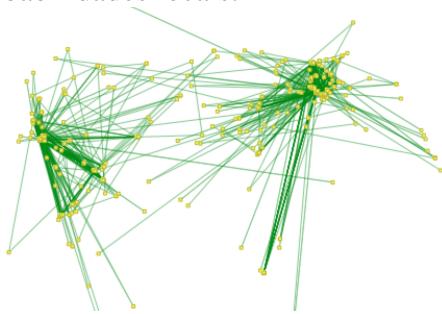


Figura 7. Visualização de trajetórias estruturadas, utilizando os centroides de episódios *Stop*

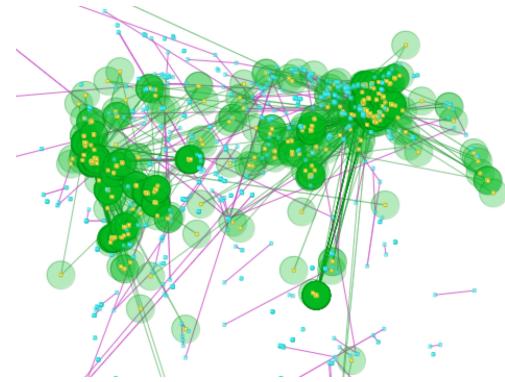


Figura 8. Trajetórias estruturadas com o *buffer* construído e trilhas de mídias sociais

Após o cálculo dos coeficientes de correlação foi selecionada a TMS com a maior probabilidade global para todos os pares trajetória-trilha, e foi realizada a fusão destes pares, adicionando os dados inseridos pelo usuário no Twitter aos episódios do tipo *stop*. Esta fusão pode ser encarada como uma anotação de trajetórias.

4.1. Resultados

A Figura 14 apresenta um exemplo de TE e anotada produzida pelo método de fusão proposto, a partir dos dados usados nos experimentos (trajetórias de táxis e *tweets* em

Fortaleza). A trajetória, representada pela linha contínua foi segmentada em 3 episódios: Ep_1 , Ep_2 e Ep_3 . O método proposto detectou a correlação desta trajetória com a trilha composta pelas pegadas SMF_1 , SMF_2 , SMF_3 , SMF_4 . Isso permitiu a associação das anotações a tais pegadas (listadas na tabela à direita) aos episódios espaço-temporamente próximos das respectivas pegadas.

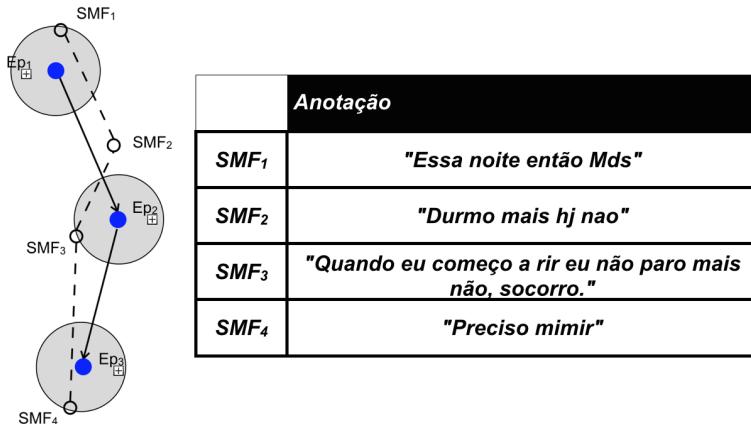


Figura 9. Exemplo de fusão

Os experimentos iniciais, relatados neste trabalho, sugerem a viabilidade de fundir trajetórias com TMSs, para obter anotações para as primeiras. Entretanto, é necessário a realização de experimentos com bases de dados do mesmo período de tempo para realizar tais afirmações.

5. Trabalhos Relacionados

O enriquecimento semântico de dados de trajetórias tem sido amplamente investigado na literatura por diversos projetos internacionais como por exemplo o MODAP (Mobility, Data Mining, and Privacy)¹ e SEEK (SEmantic Enrichment of trajectory Knowledge discovery)². Diversos trabalhos dentro destes projetos buscam aumentar a quantidade de informações que se pode extrair dos mesmos mediante enriquecimento semântico [Alvares et al. 2007, Yan et al. 2012].

[Alvares et al. 2007] busca enriquecer as trajetórias utilizando o processamento de trajetórias (e.g., inferência do veículo que esta sendo utilizado pelo portador do objeto móvel baseado em sua velocidade), além de utilizar bases de dados de informações externas as trajetórias (e.g., pontos turísticos da região).

[Yan et al. 2012] propõe uma plataforma de anotação de trajetórias, esta plataforma consiste na estruturação de trajetórias em diferentes camadas, de algoritmos de processamento de trajetórias como citado em [Alvares et al. 2007] e utilização de bases de dados de pontos de interesse provenientes de *linked data*.

As anotações geradas por [Alvares et al. 2007] e [Yan et al. 2012] são fruto do processamento de padrões nas trajetórias e interpretação desses padrões, utilizando conhecimento externo as trajetórias, portanto as anotações não requerem somente uma base de dados para sua execução, mas também conhecimento especialista para sua anotação.

¹<http://www.modap.org/>

²<http://www.seek-project.eu/>

O método aqui proposto, por outro lado, permite anotar trajetórias automaticamente, não sendo necessário conhecimento especialista. Além disso, até onde vai nosso conhecimento, esta abordagem baseada em fusão de trajetórias com TMSs é inédita na literatura sobre trajetórias de objetos móveis.

6. Conclusão e Trabalhos Futuros

Este trabalho propôs um método para fundir trajetórias com TMSs para agregar informações textuais a TEs. Esta fusão é realizada em três fases: pré-processamento, compressão de trajetórias e fusão de trajetórias com trilhas. A principal contribuição deste trabalho é a definição de um novo método para realização de fusão de trajetórias, entretanto para medir seu desempenho quantitativo comparado com outros métodos é necessário a realização de mais experimentos.

Tal método pode ser expandido para realização de fusões espaço-temporais entre bases de dados de TBS e trilhas de redes sociais, desde que as bases de dados pertençam ao mesmo período de tempo. As trajetórias anotadas geradas podem ser utilizadas para trabalhos futuros, como por exemplo o enriquecimento semântico de trajetórias com dados ligados [Fileto et al. 2013].

Como trabalhos futuros se espera a (i) realização de experimentos com base de dados pertencentes ao período de tempo; (ii) melhora da complexidade do algoritmo de fusão; (iii) enriquecer semanticamente trajetórias anotadas.

7. Agradecimentos

Este trabalho foi apoiado pelo projeto da União Europeia IRSES-SEEK, CNPq e CAPES.

Referências

- Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., and Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. In *Proc. of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '07, pages 22:1–22:8, New York, NY, USA. ACM.
- Fileto, R., Krüger, M., Pelekis, N., Theodoridis, Y., and Renso, C. (2013). Baquara: A holistic ontological framework for movement analysis using linked data. In Ng, W., Storey, V., and Trujillo, J., editors, *Conceptual Modeling*, volume 8217 of *Lecture Notes in Computer Science*, pages 342–355. Springer Berlin Heidelberg.
- Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. (2008). A conceptual view on trajectories. volume 65, pages 126–146, Amsterdam, The Netherlands, The Netherlands. Elsevier Science Publishers B. V.
- Xiu-li, Z. and Wei-xiang, X. (2009). A clustering-based approach for discovering interesting places in a single trajectory. In *Intelligent Computation Technology and Automation, 2009. ICICTA '09. 2nd Int. Conf. on*, volume 3, pages 429–432.
- Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., and Aberer, K. (2012). Semantic Trajectories: Mobility Data Computation and Annotation. volume 9, pages 39:1–39:34, New York. ACM Transactions on Intelligent Systems and Technology.

Análise de Inconsistências Espaçotemporais entre Trajetórias e Tarefas Planejadas ou Relatadas

Felipe Pinto da Silva, Renato Fileto

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
(UFSC) Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil

fpsilva98@gmail.com, fileto@ufsc.br

Abstract. Several studies try to infer what happened with a moving object over a trajectory, but generally without using user' reports or data relating to planned tasks. This work proposes to analyze spatiotemporal inconsistencies between trajectories of moving objects and planned or reported tasks using three data sources: the trajectories of moving objects (automatically collected by GPS); User reports (spatiotemporal points that identify start and end of task execution) and data planned tasks (duration and place of execution). We are currently implementing the proposal and preparing experiments with real data provided by a company specialized in services related to water supply.

Resumo. Vários trabalhos tentam inferir o que ocorreu com um objeto móvel em uma trajetória, mas geralmente sem utilizar relatos do usuário ou dados de tarefas planejadas. Este trabalho propõe analisar inconsistências spaçotemporais entre trajetórias de objetos móveis e tarefas planejadas ou relatadas, utilizando três fontes de dados: as trajetórias dos objetos móveis (coletada por GPS); Relatos dos usuários (pontos spaçotemporais identificam início e fim de execução da tarefa) e dados de tarefas planejadas (tempo de duração e local de execução). Atualmente, estamos implementando a proposta e preparando experimentos com dados reais fornecidos por uma empresa especializada em serviços relacionados com suprimento de água.

1. Introdução

Trajetórias de objetos móveis têm sido exploradas em estudos e experimentos sobre objetos móveis (pessoas, veículos, animais, etc.), principalmente após a popularização de dispositivos móveis como os *smartphones*. Estes dispositivos, frequentemente munidos de equipamentos de localização geográfica (GPS, GSM), capturam sua localização geográfica a cada momento. As trajetórias brutas coletadas por tais dispositivos, geralmente, consistem de pontos spaçotemporais (amostras de sua posição) ordenados temporalmente [Parent et al. 2012].

A análise de trajetórias visa obter informações relevantes, como locais de interesse e episódios (e.g., STOPS e MOVES) [Alvares (2007)]. Um episódio é uma subsequência maximal de pontos de uma trajetória que satisfaz um predicado [Mountain and Raper 2001]. Um *Stop* é um tipo de episódio definido por um predicado espacial (e.g., pontos da subtrajetória dentro de um local ou com distância máxima entre eles dentro de certos limites) e/ou temporal (tempo transcorrido entre o primeiro e o último ponto da subtrajetória menor que um valor limite dado).

A análise de trajetórias pode também identificar a realização de tarefas do agente que porta o dispositivo móvel. Segundo Huang, Li e Yue (2010), uma tarefa é composta pela localização da sua execução, instante inicial, duração e propósito. Desta forma, os autores identificam uma atividade e a definem de acordo com o tipo do local onde a atividade é realizada. Já Clark e Doherty (2008) identificam as tarefas e focam no escalonamento realizado pelo objeto móvel. Em ambos os casos não há utilização de dados relatados pelo agente executor da tarefa.

Dispositivos móveis, além de fornecerem histórico de localização, permitem que aplicativos coletem relatos dos usuários que podem conter informações como posição geográfica e instante de início e fim de execução de tarefas, etc. Estabelecendo alguns critérios, como sobreposição espacotemporal, é possível correlacionar os relatos de um usuário com a tarefa planejada, permitindo a identificação e a análise de inconsistências entre o que foi relatado pelo usuário e o que foi planejado.

Este artigo propõe um método computacional para detectar e classificar inconsistências espacotemporais entre trajetórias e tarefas planejadas e/ou relatadas. O método faz uso de três fontes de dados: trajetórias brutas dos objetos móveis, dados relacionados às tarefas e os relatos dos usuários. O método proposto confronta os dados espacotemporais das tarefas planejadas com a trajetória e os relatos do objeto móvel. Para auxiliar na inferência, os casos são classificados de acordo com uma proposta também apresentada neste trabalho.

A proposta está em fase de implementação e sua validação será realizada em um banco de dados cedido por uma empresa que fornece software para acompanhamento de serviços de manutenção de redes de água e leitura manual de consumo mensal. As equipes se movimentam no espaço geográfico usando dispositivos móveis que coletam suas trajetórias e relatos da realização de tarefas previamente planejadas.

O restante deste artigo está organizado como descrito a seguir. A seção 2 apresenta conceitos básicos necessários à compreensão do problema tratado e da solução proposta no artigo. A seção 3 descreve a abordagem proposta. A seção 4 descreve um estudo de caso e experimentos preliminares. A seção 5 discute alguns trabalhos relacionados. Finalmente, a seção 6 conclui o artigo e revela trabalhos futuros.

2. Fundamentos

Uma das fontes de dados utilizada pelo método proposto é a trajetória bruta do usuário. A partir desta será criada a trajetória estruturada, contendo os episódios *stops*, conforme as definições apresentadas a seguir, adaptadas de [Yan et al. 2013].

Definição 1: Uma **trajetória bruta** é uma sequência temporalmente ordenada $T_b = p_1, p_2, \dots, p_n$ de amostras de pontos espacotemporais coletados de um objeto em movimento. Cada ponto P_i tem a forma (x_i, y_i, t_i) , onde x_i, y_i são coordenadas geográficas e t_i é um instante de tempo.

Definição 2: Um **episódio** é uma subsequência de trajetória bruta (subtrajetória) $E = p_{inicial}, \dots, p_{final}$ ($1 \leq inicial \leq final \leq n$) que satisfaz um dado predicado e é maximal em termos do seu número de pontos.

Definição 3: Uma **trajetória estruturada** é uma sequência temporalmente ordenada de episódios $T_e = E_1, E_2, \dots, E_m$ não aninhados.

Uma tarefa é composta de sua localização, instante inicial, duração e propósito [Huang, Li e Yue 2010]. Entretanto, o método proposto não se limita a um cronograma temporalmente rígido, i.e., não interessa o instante em que uma tarefa deve começar e sim o tempo mínimo e máximo de duração. O propósito de execução de uma tarefa fica definido em uma estrutura de chave-valor auxiliar denominado tipo de tarefa.

Além disso, cada tarefa é executada por um agente, que porta um dispositivo móvel (e.g., indivíduo, grupo, veículo) durante certo período de tempo. O agente se locomove em uma determinada região geográfica a fim de executar uma série de tarefas previamente planejadas.

Definição 4: Uma **tarefa planejada** é um tupla $\tau = (\text{id_tarefa}, x, y, \text{qt_minutos_mínimo}, \text{qt_minutos_máximo}, \text{id_tipo_tarefa}, \text{id_agente})$, onde id_tarefa é a chave primária da entidade, x e y são coordenadas geográficas, qt_minutos_mínimo e qt_minutos_máximo representam o tempo mínimo e máximo planejado para a execução da tarefa, id_tipo_tarefa representa o tipo de tarefa e id_agente o identificador do agente que deve executar a tarefa. As tarefas planejadas são mantidas no conjunto $C = \{\tau_1, \tau_2, \dots, \tau_n\}$.

Um agente pode, então, relatar o início e fim da execução de uma tarefa a qualquer momento e fornecer informações expressivas para a análise de inconsistências.

Definição 5: Um relato de tarefa R é uma tupla: $(\text{id_relato}, \text{dt_instante_início}, \text{dt_instante_fim}, x, y, \text{id_agente}, \text{id_tarefa})$, onde id_relato é a chave primária da entidade, $\text{dt_instante_início}$ e dt_instante_fim representam o instante do início e fim da execução da tarefa, segundo o agente, x e y são coordenadas do local onde o relato foi realizado, id_agente é o identificador do agente e id_tarefa representa o identificador da tarefa planejada. Os relatos do usuário são armazenados no conjunto $O = \{R_1, R_2, \dots, R_n\}$.

3. Proposta

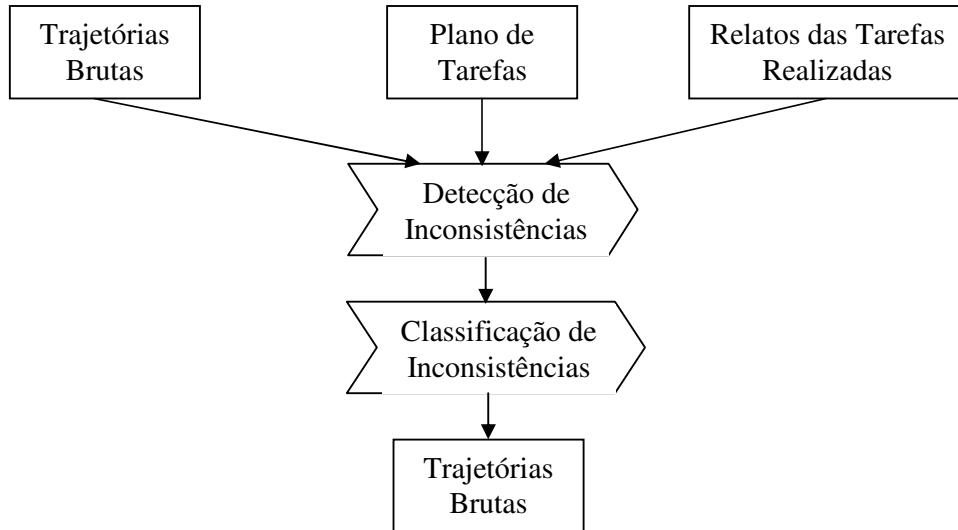
Esta seção descreve o método proposto para analisar inconsistências espacotemporais entre trajetórias e tarefas planejadas ou relatadas. A seção 3.1 descreve a modelagem do método e as seções 3.2 e 3.3 detalham os processos de detecção e classificação.

3.1. Visão Geral do Processo Proposto

A Figura 1 apresenta a visão geral do processo para *Análise de Inconsistências* entre trajetórias e tarefas planejadas e/ou relatadas. As entradas deste processo são: as *Trajetórias Estruturadas* (Definição 3), o *Conjunto de Tarefas Planejadas* (Definição 4) e o *Conjunto de Relatos de todos os agentes* (Definição 5). As saídas são as *Inconsistências Classificadas*. O processo é dividido em duas etapas, sendo: (i) *Detecção de Inconsistências*, consiste em determinar quais são inconsistências espaciais e temporais das trajetórias em relação ao que foi planejado e relatado e (ii) *Classificação das Inconsistências*, classifica as inconsistências retornadas pela etapa anterior de

acordo com o tipo de desvio no tempo e no espaço. Essas duas etapas do método proposto são descritas em detalhes nas seções 3.2 e 3.3, respectivamente.

Figura 1. Processo proposto



3.2. Detecção de Inconsistências

A primeira etapa do método proposto consiste em relacionar cada tarefa τ a um ou mais episódios do tipo *stop* e ao seu respectivo relato \mathbf{R} . Como definido na seção 2, um relato é composto pelo identificador da tarefa a qual se refere. Já para encontrar *stops* de uma tarefa o método utiliza a distância geodésica entre as coordenadas geográficas de τ com cada episódio \mathbf{S} da trajetória estruturada. Quando a distância for inferior ao *threshold* definido em metros pelo usuário, o método correlaciona o *stop* aquela tarefa e posteriormente classificados. Os *stops* que não foram relacionados a nenhuma tarefa também são classificados. A seguir o pseudocódigo utilizado pelo método para a detecção de *stops* e relatos de cada tarefa.

Algoritmo 1: Detecção de Inconsistências

Entradas: Trajetórias Estruturadas, Relatos dos Usuários, Conjunto de Tarefas Planejadas e Threshold

Saídas: Casos para serem classificados

```

Para cada tarefa  $\tau \in \mathbf{C}$  {
    Para cada stop  $\mathbf{S}$  da trajetória estruturada  $\mathbf{T}_e$  {
        Se (distancia ( $\tau$ ,  $\mathbf{S}$ ) < threshold) {
             $\tau$ ->listaStop.adiciona( $\mathbf{S}$ );
             $\mathbf{S}$ ->listaTarefas.adiciona( $\tau$ );
        }
    }
     $\mathbf{R}'$  = nulo;
    Para cada relato  $\mathbf{R}$  dos relatos de agentes {
        Se ( $\mathbf{R}$ -> $\tau$  ==  $\tau$ ) {
             $\mathbf{R}'$  =  $\mathbf{R}$ ;
        }
    }
    classificar( $\tau$ ,  $\mathbf{R}'$ ,  $\tau$ ->listaStop);
}
  
```

```

Para cada stop S da trajetória estruturada Te {
    Se (ehVazia(S->listaTarefas)) {
        classificar(nulo, nulo, S);
    }
}

```

Todas as tarefas são classificadas, mesmo que não possua relato ou *stop*, pois o método supõe que seja uma tarefa não realizada e assim será classificada. O método também classifica os *stops* que não foram relacionados a uma tarefa do plano de tarefas, neste caso, o método supõe que seja um período de ociosidade do agente.

3.3. Classificação das Inconsistências

A classificação das inconsistências detectadas leva em consideração os valores espaciais e temporais de cada fonte de dados (Tarefa Planejada, *Stop* e Relato). Os dados spaçotemporais de cada tarefa planejada são usados como valores de referência para a classificação das inconsistências. Entretanto, há situações onde os dados de duas fontes estão ausentes (e.g. tarefa prevista, mas não relatada e não realizada). A Tabela 2 sugere a classificação quando só há dados sobre uma tarefa de apenas uma fonte.

Tabela 2. Classificação com apenas uma fonte de dado

Caso	Único dado presente	Classificação
1	Tarefa planejada	Tarefa não realizada
2	Episódio <i>Stop</i> na trajetória	Fuga do trabalho/ociosidade
3	Relato da tarefa	Fraude ou erro

O primeiro caso refere-se a uma tarefa planejada que não possui relato de execução e *stop*. O segundo caso se trata de um *stop* que, durante o processo de detecção, não foi relacionado a nenhuma tarefa planejada. Já o terceiro caso ilustra uma situação impossível, já que, segundo o que foi previamente definido, um relato deve pertencer a uma tarefa, garantido por uma associação de composição.

Há casos onde uma tarefa não teve relato, somente um *stop*. A tabela 3 apresenta a classificação para um cenário com as fontes de dados: Tarefa planejada e *Stop*.

Tabela 3. Classificação com duas fontes de dados (planejado e stops)

Caso	Local do Stop	Duração do Stop	Classificação
4	Próximo ao local planejado	Entre o intervalo mínimo e máximo	Esquecimento de relato
5	Próximo ao local planejado	Superior ao máximo planejado	Esquecimento de relato com tempo de parada superior ao planejado
6	Próximo ao local planejado	Inferior ao mínimo planejado	Esquecimento de relato com tempo de parada inferior ao planejado

Em todos os casos da Tabela 3, o agente não relatou a execução da tarefa, entretanto executou um *stop* próximo ao local planejado. No primeiro caso da tabela, o agente realizou o *stop* por um tempo entre o mínimo e o máximo planejado. Já no quinto e sexto caso o tempo de permanência foi superior e inferior, assim respectivamente. A tabela não contém casos onde o *stop* foi realizado em local diferente do planejado porque a detecção só relaciona as entidades quando o *stop* foi realizado próximo ao local planejado.

Tabela 4. Classificação com duas fontes de dados (planejado e relatado)

Caso	Local Relato	Duração Relatada	Classificação
7	Próximo ao local planejado	Entre o intervalo mínimo e máximo	Fraude
8	Próximo ao local planejado	Superior ao máximo	Fraude + Suspeita de valorização de tempo
9	Próximo ao local planejado	Inferior ao mínimo	Fraude
10	Divergente do local planejado	Entre o intervalo mínimo e máximo	Fraude
11	Divergente do local planejado	Superior ao máximo	Fraude + Suspeita de valorização de tempo
12	Divergente do local planejado	Inferior ao mínimo	Fraude

Nenhum caso da Tabela 4 há *stop* do agente próximo ao local planejado. O sétimo caso se trata de um relato com uma duração e local planejado. Já os dois casos seguintes divergem temporalmente, para mais e para menos, assim respectivamente. Relatos em local divergente estão apresentados nos três casos seguintes. Sendo o décimo com o tempo relatado dentro do intervalo esperado e os dois subsequentes superior ao máximo e inferior ao mínimo, respectivamente.

A Tabela 5 apresenta a classificação dos casos de acordo com as possíveis variações espacotemporais das três fontes de dados. Lembrando que um *stop* é relacionado a uma tarefa quando o mesmo se encontrar próximo ao local planejado, assim, é impossível ter uma tarefa com um *stop* realizado em local divergente ao planejado no cenário de classificação.

Tabela 5. Classificação com três fontes de dados

Caso	Local Stop	Duração Stop	Local Relato	Duração Relatada	Classificação
13	Próximo ao local planejado	Entre o intervalo planejado	Próximo ao local planejado	Entre o intervalo planejado	Em conformidade
14	Próximo ao local planejado	Entre o intervalo planejado	Próximo ao local planejado	Superior ao máximo	Supervalorização do tempo
15	Próximo ao local planejado	Entre o intervalo planejado	Próximo ao local planejado	Inferior ao mínimo	Erro no relato ou melhor caso
16	Próximo ao local planejado	Entre o intervalo planejado	Divergente do local planejado	Entre o intervalo planejado	Relato Posterior
17	Próximo ao local planejado	Entre o intervalo planejado	Divergente do local planejado	Superior ao máximo	Supervalorização do tempo
18	Próximo ao local planejado	Entre o intervalo planejado	Divergente do local planejado	Inferior ao mínimo	Relato posterior + (erro no relato ou melhor caso)
19	Próximo ao local planejado	Superior ao máximo	Próximo ao local planejado	Entre o intervalo planejado	Pausa (não relatada)

20	Próximo ao local planejado	Inferior ao mínimo	Próximo ao local planejado	Entre o intervalo planejado	Falha na estimativa de tempo e supervvalorização do tempo
21	Próximo ao local planejado	Superior ao máximo	Próximo ao local planejado	Superior ao máximo	Falha de estimativa ou pior caso
22	Próximo ao local planejado	Superior ao máximo	Próximo ao local planejado	Inferior ao mínimo	Pausa não relatada + melhor caso
23	Próximo ao local planejado	Inferior ao mínimo	Próximo ao local planejado	Superior ao máximo	Supervalorização do tempo
24	Próximo ao local planejado	Inferior ao mínimo	Próximo ao local planejado	Inferior ao mínimo	Falha de estimativa ou melhor caso
25	Próximo ao local planejado	Superior ao máximo	Divergente do local planejado	Entre o intervalo planejado	Pausa/ociosidade (não relatada)
26	Próximo ao local planejado	Inferior ao mínimo	Divergente do local planejado	Entre o intervalo planejado	Supervalorização do tempo
27	Próximo ao local planejado	Superior ao máximo	Divergente do local planejado	Superior ao máximo	Falha de estimativa ou pior caso
28	Próximo ao local planejado	Superior ao máximo	Divergente do local planejado	Inferior ao mínimo	Pausa no local da tarefa
29	Próximo ao local planejado	Inferior ao mínimo	Divergente do local planejado	Superior ao máximo	Supervalorização do tempo
30	Próximo ao local planejado	Inferior ao mínimo	Divergente do local planejado	Inferior ao mínimo	Falha de estimativa ou melhor caso

Do décimo terceiro caso até o décimo oitavo, o agente executou sempre *stop* no local planejado e com o tempo de duração esperado. O décimo terceiro ilustra a situação onde o relato do usuário coincide espacial e temporalmente com o que foi planejado. Os dois casos seguintes têm divergência temporal para mais e para menos, respectivamente. O décimo sexto possui somente divergência espacial, enquanto seus dois subsequentes possuem divergência espacial e temporal.

Casos onde há apenas divergência temporal do *stop* estão ilustrados no décimo nono e vigésimo caso. Enquanto os casos a partir do vigésimo primeiro até o vigésimo quarto apresentam as combinações de divergência temporal de *stop* e relato em relação ao planejado. O vigésimo quinto e vigésimo sexto descrevem casos onde o relato foi feito em local divergente do planejado e com *stop* realizado em tempo superior e inferior, respectivamente, ao tempo planejado para a tarefa.

A Tabela 5 apresenta situações com relato realizado em local divergente do planejado a partir do vigésimo sétimo caso. Todos eles possuem divergência temporal no relato e no *stop*. Estes casos descrevem as combinações de divergência temporal superior ao máximo e inferior ao mínimo do relato e do *stop*.

4. Estudo de caso

Avaliamos o método de detecção e classificação proposta em experimentos iniciais sobre um banco de dados de uma empresa prestadora de serviço de saneamento, que possui equipes de manutenção para realizar diversos tipos de serviços em uma determinada região geográfica. Cada agente (equipe de manutenção) porta um dispositivo móvel com uma aplicação que lista as tarefas que devem ser executadas e permite que o agente relate o instante de início e término da execução. A aplicação ainda coleta a localização geográfica do dispositivo a cada minuto. Os relatos e as localizações do agente são enviados para um banco de dados na sede da empresa. O banco de dados ainda possui dados de endereço de cada local de execução da tarefa, bem como o tempo padrão para a execução do serviço. O número de registros de pontos espacotemporais de equipes de manutenção somam pouco mais 900 mil.

4.1. Pré-processamento

Antes de iniciar os experimentos foi necessária a realização do *Geocoding* dos endereços dos locais de execução de tarefas e a transformação das trajetórias brutas de cada equipe em trajetórias estruturadas, com o intuito de obter os *stops* realizados pela equipe. Definiu-se um *stop* como uma subsequência de pelo menos três pontos espacotemporais, onde todos os pares de pontos têm distância igual ou inferior a 15 metros e diferença temporal de no máximo 6 horas em relação.

4.2. Resultados de Experimentos Preliminares

Os experimentos conduzidos até aqui revelam diversas situações. A Tabela 6 apresenta algumas inconsistências encontradas para a equipe 201 durante o mês de Junho de 2013.

Tabela 6. Situações equipe 201 em Junho/2013.

Exemplo	ID Tarefa	Dia	Tempo Padrão (minutos)	Distância Stop (*)	Duração Stop (minutos)	Distância Relato (*)	Duração Relatada (minutos)
1	438094	05	20 a 30	Não houve	Não houve	428	46
2	439621	13	30 a 60	1,23	102	7,25	14
3	441832	25	20 a 30	6,79	16	6,785	16
4	442639	26	40 a 60	4,9	41	4,821	1
5	-	29	-	-	70	-	-

(*) entre o dispositivo e o local planejado, em metros.

A Tabela 6 contém o identificador da tarefa e o dia do mês em que a tarefa foi executada. A coluna “Tempo Padrão” representa o tempo mínimo e máximo planejado para a execução da tarefa em minutos. A quinta coluna apresenta a distância entre o local do *stop* e o local planejado em metros, já a sexta coluna o tempo de duração do *stop*. A distância entre o local do relato e o local planejado está apresentada na sétima coluna enquanto a coluna seguinte contém a duração da execução da tarefa informada pelo agente. Os exemplos citados foram então classificados de acordo com o método proposto, conforme apresentado na Tabela 7.

Tabela 7. Classificação das situações.

Exemplo	Classificação	Caso Classificação
1	Fraude + Supervalorização do tempo	11
2	Pausa não relatada + melhor caso	22
3	Falha de estimativa ou melhor caso	24
4	Erro no relato ou melhor caso	15
5	Fuga do trabalho/Ociosidade	2

A classificação proposta tenta inferir o que ocorreu em cada caso. O primeiro caso descreve uma tarefa sem *stop* próximo ao local planejado, mas relatada com um tempo superior ao planejado a mais de 400 metros do local planejado, justificando a suspeita de fraude e supervalorização. Já o quinto exemplo caracteriza um caso de ociosidade uma vez que um *stop* de 70 minutos foi realizado sem nenhum relato.

5. Trabalhos Relacionados

Parent *et al.* (2013) é uma resenha ampla e atual de conceitos e técnicas relacionados a detecção de episódios relevantes e entendimento do comportamento de objetos móveis a partir de suas trajetórias. Somente alguns trabalhos mais específicos propõem soluções para analisar o comportamento dos objetos móveis na realização de atividades.

O processo proposto em Clark e Doherty (2008) compara trajetórias com um cronograma de tarefas planejadas, com o intuito de identificar e entender os motivos dos re-escalonamentos das tarefas. Tal processo detecta relocação de tarefas, usando instantes de início e fim previstos para cada uma. Ao final do processo, uma entrevista confirma as relocações das tarefas realizadas.

O método introduzido em Huang, Li e Yue (2010) pressupõe que as atividades realizadas ao longo de uma trajetória ocorrem nos pontos de interesse (POI) onde o objeto móvel permanece estacionado e as classificam de acordo com categorias de POIs previamente cadastrados. Logo, a quantidade e a qualidade dos POIs usados é decisiva na classificação. Entretanto o método pode ser falho, já que um mesmo POI pode ter funções diferentes para pessoas distintas (e.g., lar ou local de execução de um serviço).

Similarmente, o algoritmo apresentado em Furletti *et al.* (2013) identifica tarefas realizadas ao longo de trajetórias, usando POIs extraídos das API's do Google Places e OpenStreetMap. O algoritmo proposto elege o provável POI associado a cada episódio e então confronta esta informação com diários de viagem preenchidos pelos executores das tarefas. Entretanto, tal algoritmo não usa conhecimento prévio de tarefas planejadas.

Nenhum dos trabalhos citados acima detecta e classifica inconsistências das trajetórias com planejamento e relato de tarefas. O método proposto neste artigo identifica tais inconsistências de acordo com um conhecimento prévio das tarefas que devem ser executadas e considera anotações do agente executor da tarefa para detectar e classificar inconsistências espacotemporais entre a trajetória do objeto móvel e: (i) o que foi planejado para ser executado; e/ou (ii) o que é relatado pelo agente executor.

6. Conclusão e Trabalhos Futuros

A análise de trajetórias de objetos móveis vem sendo muito estudada na literatura. No entanto, uma possibilidade ainda pouco explorada é o uso de outras fontes de dados,

como diários e tarefas planejadas para descoberta de informações. Este artigo propõe um método para detectar e classificar inconsistências espacotemporais entre trajetórias de um objeto móvel e tarefas planejadas e/ou relatadas. Suas principais contribuições são um algoritmo para detecção de inconsistências e uma classificação das mesmas. O método proposto enumera e qualifica diversos tipos de inconsistência, funcionando mesmo quando há alguns tipos de ausência de dados (e.g., tarefas não relatadas).

Atualmente, estamos concluindo o desenvolvimento de um protótipo do método proposto e iniciando testes com grandes volumes de dados reais. Os trabalhos futuros incluem: (i) estender o método para avaliar os locais onde há comportamentos suspeitos usando uma base de dados geográfica, tal como o *LinkedGeoData*; (ii) analisar semanticamente as anotações em forma de texto efetuadas por executores de tarefas, visando obter informações adicionais, tais como eventos, ações e intenções.

7. Agradecimento

Este trabalho foi apoiado pelo CNPq.

Referências

- Alvares, L.O., Bogorny, V., Kuijpers, B., Macedo, J.A.F., Moelans, B. and Vaisman, A. (2007). *A model for enriching trajectories with semantic geographical information*. Proc. ACM-GIS, pp. 162–169, New York, NY, USA. ACM Press.
- Bogorny, V., Avancini, H., de Paula, B.L., Kuplisch, C.R., And Alvares, L.O. (2011). *Weka-STPM: a Software Architecture and Prototype for Semantic Trajectory Data Mining*. Transactions in GIS, 15:(2) 227-248
- Clark, A. F., Doherty, S. T. (2008) *Use of GPS to automatically track activity rescheduling decisions*. 8th International Conference on Survey Methods in Transport.
- Furletti, B., Cintia, P., Renso, C., Spinsanti, L. (2013). *Inferred human activities from GPS tracks*. UrbComp '13 Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing.
- Huang, L., Li, Q., Yue, Y. (2010). *Activity identification from GPS trajectories using spatial temporal POI's attractiveness*. 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks.
- Mountain, D. and Raper, J. (2001). *Modelling human spatio-temporal behaviour: a challenge for location based services*. International Conference on Geocomputation, pages 24–26, Brisbane, Australia
- Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., and Yan, Z. (2013). *Semantic trajectories modeling and analysis*. ACM Computing Surveys, 45(4).
- Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., Aberer, K. (2013) *Semantic trajectories: Mobility data computation and annotation*. ACM TIST 4(3): 49.

LinkMapia: Uma Abordagem para Converter Dados Geográficos Livremente Anotados em Dados Ligados

Juarez A. P. Sacenti¹, Renato Fileto¹

¹Dep. de Informática e Estatística – Universidade Federal do Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brazil

juarezsacenti@inf.ufsc.br, r.fileto@ufsc.br

Abstract. *Collecting geographic data by using satellites, global positioning system (GPS) and even crowdsourcing with free annotations often gathers data with little semantic, with ambiguities and little interoperability. This paper proposes a process to convert textual annotated geographic data to a linked geographic data collection. This proposal is based in a process that filters and cleans data, and then applies several techniques to align this data with existing linked data collection. Experiments applying the proposed process to Wikimapia and Linked Geo Data generate linked data that support several useful SPARQL queries.*

Resumo. *A coleta de dados geográficos, utilizando tecnologias de sensoriamento (e.g., GPS, câmeras) e mesmo sistemas de anotação colaborativa, frequentemente geram dados com semântica insuficiente, com ambiguidades e pouca interoperabilidade. Este trabalho propõe um processo para converter dados geográficos anotados textualmente em dados ligados. A proposta baseia-se em um processo que filtra e limpa dados, e então aplica diversas técnicas para alinhar estes dados a coleções existentes de dados ligados. A aplicação do processo proposto aos dados do Wikimapia e Linked Geo Data em experimentos gerou dados ligados que suportam diversas consultas SPARQL úteis.*

1. Introdução

Dados associados a características espaciais, que referenciam uma localização na superfície da Terra, são chamados de dados geográficos [Casanova et al. 2005]. O espaço, ocupado por elementos (*e. g.*, construções, rios e estradas) e fenômenos (*e. g.*, massas de ar, dissimilação de doenças e temperatura), é representado computacionalmente por formas vetoriais (*e. g.*, pontos, retas, polígonos) ou matriciais (*e. g.*, *raster*). Estas representações são os atributos espaciais dos dados geográficos.

Porém, dados geográficos possuem informações alfanuméricas, tão importantes quanto atributos espaciais: os atributos descritivos [Rigaux et al. 2000]. Descrever dados geográficos depende da interpretação humana do elemento ou fenômeno representado. A coleta de dados geográficos utilizando tecnologias de sensoriamento (*e.g.*, GPS, câmeras) é insuficiente para levantar estas informações.

Sistemas de coleta colaborativa de dados geográficos (*e. g.*, Wikimapia¹, *Open Street Map*² - OSM) são providos de recursos para permitir principalmente a coleta de

¹Disponível em: <http://wikimapia.org/about>. Acesso em mar. 2014.

²Disponível em: <http://www.openstreetmap.org/about>. Acesso em mar. 2014.

atributos descritivos, geralmente de forma voluntária e via *web*. Entretanto, dados anotados desta forma, *i. e.*, *volunteer geographic information* (VGI) [Goodchild 2007], geralmente não apresentam semântica suficientemente formal para ser utilizada computacionalmente e suficientemente detalhada para suprir as necessidades das aplicações. As anotações livres, *i. e.*, anotações que não apresentam nenhum tipo de estruturação explícita de seu conteúdo (*e. g.*, vocabulário comum, glossário), herdam os problemas da manipulação de textos livres: sinônimos, ambiguidades e erros ortográficos.

Nestes sistemas sociais de anotação (*Social Tagging Systems* - STS), algumas anotações livres são utilizadas para posteriormente recuperar os objetos anotados. A estrutura conceitual destas anotações, emergente de STSs, é chamada de *folksonomia* [García-castro and García 2011].

A adoção e manutenção de convenções de anotação, *e. g.*, enciclopédias (*thesaurus*), glossários, dicionários geográficos (*gazetters*), em sistemas com grande número de voluntários são custosas. As soluções da *web* semântica, como as anotações semânticas [Berners-Lee et al. 2001], ontologias [Guarino 1998] e dados ligados (*linked data*) [Heath and Bizer 2011], podem contribuir para resolver os problemas de ambiguidade e interoperabilidade das anotações colaborativas.

Para obter dados ligados de um sistema colaborativo, é necessário converter a *folksonomia* em ontologia. Este trabalho propõe um processo para conversão de dados geográficos textualmente anotados em dados ligados, e aplica-o a dados do projeto Wikimapia, um sistema colaborativo ainda não conectado a *web* de dados.

O restante deste artigo é disposto em diferentes seções. A seção 2 descreve o processo proposto. A seção 3 apresenta a implementação de cada passo deste processo. A seção 4 vislumbra a implicação do enriquecimento semântico na recuperação dos dados geográficos. A seção 5 relata alguns trabalhos correlacionados, que discorrem sobre VGI, dados geográficos ligados e alinhamento ontológico. Finalmente, a seção 6 apresenta conclusão e trabalhos futuros, acrescido de agradecimentos.

2. Proposta

Este trabalho define um processo de conversão de dados geográficos livremente anotados em dados ligados, composto por cinco etapas ilustradas na figura 1. Este processo não apenas triplifica os dados geográficos como também cria uma ontologia a partir dos termos usados na categorização destes dados, correlacionando suas categorias com entidades de ontologias como Linked Geo Data³ (LGD) e GeoNames⁴.

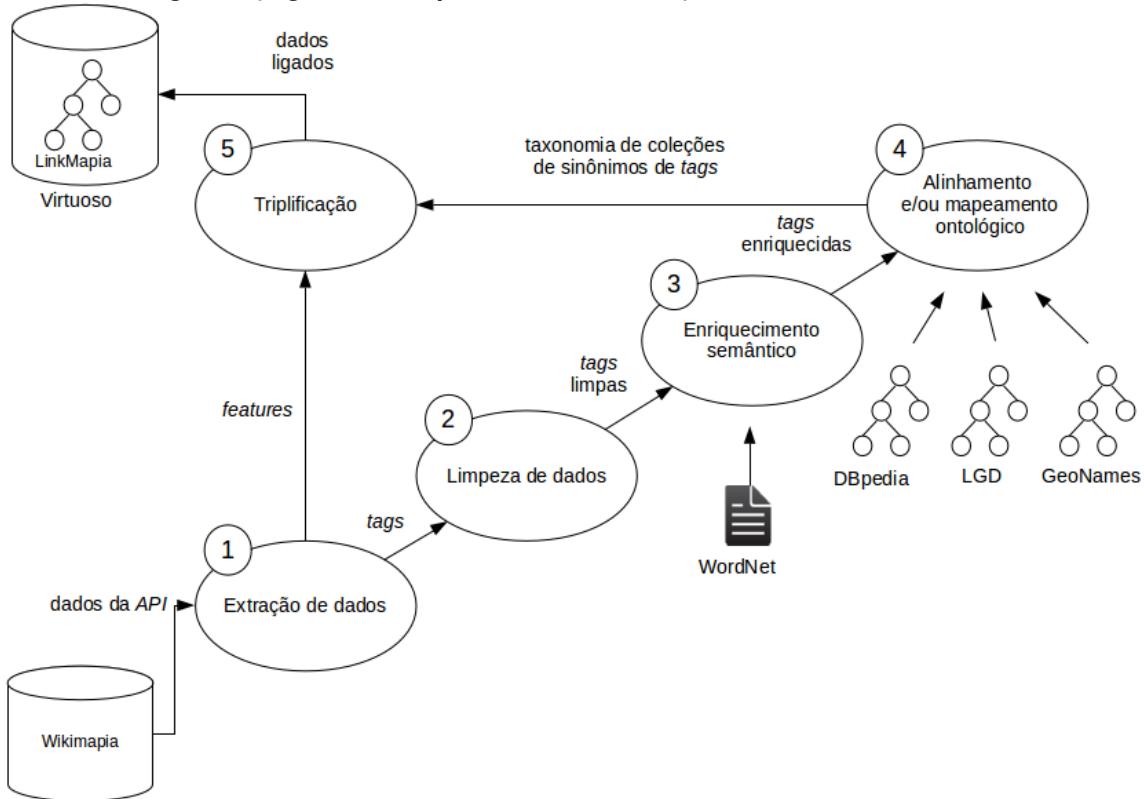
A primeira etapa deste processo realiza a extração (ou coleta) de objetos geográficos (*features*) de sistemas de anotação livre de dados geográficos, como os sistemas colaborativos OSM e Wikimapia. As anotações destes sistemas são utilizadas para categorizar e recuperar objetos geográficos. A extração de categorias é necessária para elaborar uma ontologia dos objetos extraídos.

Na segunda etapa, os nomes de categorias são tratados lexicamente. Nomes compostos são separados em termos (*tokenização*). A remoção de *stopwords* exclui termos

³Disponível em: <http://linkededgeodata.org/About>. Acesso em nov. 2013.

⁴Disponível em: <http://www.geonames.org/>. Acesso em nov. 2013.

Figure 1. Processo proposto - conversão de dados geográficos anotados em dados ligados (Figura criada pelo autor, em 2013)



semanticamente irrelevantes (*e.g.*, artigos, preposições). O passo de *stemming* reduz lexicalmente termos restantes a radicais. O tratamento léxico é necessário para obter melhores resultados em comparações sintáticas.

Na terceira etapa, os termos são comparados sintaticamente a termos de dicionários léxicos, como o WordNet⁵. O enriquecimento semântico explicitiza o conceito da categoria, antes implícito nos diferentes significados das palavras anotadas. A categoria passa a ser representada por lista de conceitos (conjunto de sinônimos). Por exemplo, o termo *service* é expandido em seu conjunto de sinônimos:

[*service, work, assist, assistance, help, activity, care, maintenance, upkeep*]

A quarta etapa, o alinhamento ou mapeamento ontológico, relaciona as categorias com classes de uma dada ontologia geográfica (*e.g.*, GeoNames, LGD). As categorias (lista de conceitos) são comparadas com os rótulos (*label*) de classes da ontologia. Esta comparação é inspirada do processo de alinhamento ontológico LOM [Li 2004].

LOM é uma técnica de *matching* de ontologias que considera apenas a representação léxica (rótulos) de classes, desconsiderando a estrutura relacional da ontologia. Deste modo, é possível adaptar LOM para o alinhamento de termos da *folksonomia* para conceitos de ontologias. Este processo identifica conceitos equivalentes, que facilitam a conversão da *folksonomia* em uma nova ontologia e integra-a a ontologias existentes.

⁵<http://wordnet.princeton.edu/>. Acesso em nov. 2013.

LOM compara os conjuntos dos rótulos dos conceitos das duas ontologias que se pretende alinhar. Tendo em mãos os dois conjuntos, são aplicados quatro processos de pareamento de conceitos: pareamento de termo completo, pareamento de termos (*tokens*) que constituem os termos completos, pareamento de conjunto de sinônimos correspondentes ao termos completos e pareamento de tipos (não explorado nesse trabalho).

Figure 2. Fluxo de pareamento entre *folksonomia* e ontologia (Figura criada pelos autores, em 2013)

LOM Adaptado - Etapas de comparação

1. Matching de termo completo (tags).

BANK = bank

2. Matching das palavras (*tokens*) que compõem os termos.

BANKS -> 'banks'	x	'bank' <- bank	= 0 %
BANKS -> 'banks'	x	'saving' + 'bank' <- saving bank	= 0 %

3. Matching das stemms (radicais dos tokens) que compõem os termos.

BANKS -> 'bank'	x	'bank' <- bank	= 100 %
BANKS -> 'bank'	x	'save' + 'bank' <- saving bank	= 50 %

4. Matching dos conjuntos de sinônimos das palavras.

STREAM BANK -> 'stream' + 'bank'	x	'bank' <- bank	
synset:stream + synset:bank	x	synset: bank	= 50 %

Nesta proposta, o pareamento de categoria e conceitos é realizado por diferentes níveis: termo completo, *tokens*, *stemming tokens* e conjunto de sinônimos (figura 2). O pareamento de termo completo envolve a comparação do texto sem tratamento. Em caso de igualdade o alinhamento é positivo e os termos envolvidos removidos das listas de candidatos a alinhamento.

Os rótulos de cada classe devem seguir os passos da etapa de limpeza para realizar os pareamentos seguintes. O pareamento de *stemming tokens* é adicionado a adaptação do LOM em virtude de radicais possuirem maior grau de generalização, facilitando a comparação léxica. Ambas as etapas utilizam uma métrica para avaliar a similaridade entre categorias da *folksonomia* (A) e conceitos de ontologia (B), tal qual:

$$sim(A, B) = \frac{\text{quantidade_termos_iguais}}{\text{tamanho_maior_conjunto}(A, B)}$$

A comparação léxica pode utilizar métricas de similaridade léxica (e.g., soft TFIDF [Cohen et al. 2003]). A categoria mais similar (categoria alinhada) a uma classe é associada a esta classe por meio da URI da classe na ontologia. As categorias menos similares àquela classe, que respeitem um limite inferior de similaridade, são relacionadas à classe alinhada como conceitos próximos.

Na quinta e última etapa, triplificação, tanto os objetos geográficos quanto as categorias são convertidas para o formato *Resource Description Framework*⁶ (RDF). Primeiro, as categorias são convertidas em classes gerando uma ontologia rasa (*i.e.*, uma árvore de altitude igual a 1, cuja raiz é a classe *Place*). Segundo, classes de categorias

⁶<http://www.w3.org/RDF/>. Acesso em mar. 2014.

alinhas a classes externas são ligadas a estas classes. Terceiro, conceitos próximos encontrados pelo alinhamento são ligados as classes. Finalmente, os objetos geográficos são convertidos em RDF e associados aos conceitos (pelos URIs) da ontologia formada.

Os dados podem então ser publicados em *end-points* SPARQL⁷ (e.g., Strabon [Kyzirakos et al. 2012], Parliament⁸ e Virtuoso⁹), possibilitando consultas integradas na coleção de dados ligados gerada e nas ontologias externas.

3. Implementação

Nesta seção é ilustrado o desenvolvimento do protótipo do processo proposto para a conversão de dados geográficos livremente anotados em dados ligados. O sistema de coleta colaborativa de dados geográficos deste caso de estudo é o Wikimapia, um projeto cujo objetivo é criar e manter um mapa atualizado, completo e multilíngue de todo o mundo, com potencial para enriquecer o conteúdo da *web* de dados geográficos (LGD, GeoNames, GeoLinkedData¹⁰, dentre outros). Os objetos geográficos deste caso de estudo são restritos aos limites da cidade de Milão, Itália, para análises espaço-temporais de trabalhos futuros. Contudo, o protótipo permite a conversão de dados de qualquer localidade.

Extração de dados. A etapa de extração de dados pode ser dividida na extração dos objetos geográficos e das categorias utilizadas na classificação destes objetos. Este conjunto de categorias forma a *folksonomia* do Wikimapia.

Objetos Geográficos - O projeto Wikimapia disponibiliza seus dados a partir de sua *Application Programming Interface* (API)¹¹. Para realizar a extração, é necessário informar o menor retângulo que contém a cidade de Milão (*minimal bounding rectangle*) é definido pelos pontos (45.388039, 9.043907) e (45.536266, 9.278963), de latitude e longitude respectiva, segundo o polígono dos limites políticos da cidade na base de dados *Database of Global Administrative Areas*¹² (GADM).

A extração de objetos geográficos (*features*) utiliza as funções *box* e *object* da API da Wikimapia. A função *box* resultou em uma lista 1276 ids de objetos, separados por páginas, sendo a variável *count* o limite de resultados por página, em formato XML¹³ ou JSON¹⁴. A função *object* obteve a descrição detalhada de cada objeto, dado seu *id*. Os dados extraídos apresentaram 243 categorias e 1513 relações de anotação. O Sistema de Gerenciamento de Banco de Dados (SGBD) utilizado para armazenar dados extraídos foi o PostgreSQL¹⁵ 9.2.4 com a extensão geográfica PostGIS¹⁶ 2.0.3-1.

Folksonomia do Wikimapia - A função *Category.GetAll* retorna todas as categorias do Wikimapia, compondo a *folksonomia* do sistema colaborativo. A extração obteve 8615 categorias: *ids*, nomes e frequência de uso (tabela 1).

⁷<http://www.w3.org/TR/rdf-sparql-query/>. Acesso em mar. 2014.

⁸Disponível em: <<http://parliament.semwebcentral.org/>>. Acesso em nov. 2013.

⁹Disponível em: <<http://virtuoso.openlinksw.com/>>. Acesso em nov. 2013.

¹⁰Disponível em: <http://geo.linkeddata.es/web/guest/home>. Acesso em nov. 2013.

¹¹Disponível em: <http://wikimapia.org/api>. Acesso em mar. 2014.

¹²Disponível em: <http://www.gadm.org/>. Acesso em mar. 2014.

¹³Disponível em: <http://www.w3.org/XML>. Acesso em mar. 2014.

¹⁴Disponível em: <http://www.json.org/xml.html>. Acesso em mar. 2014.

¹⁵Disponível em: <http://www.postgresql.org/about/>. Acesso em mar. 2014.

¹⁶Disponível em: <http://postgis.net/>. Acesso em mar. 2014.

ELEMENTO	DESCRIÇÃO	TIPO	RESTRIÇÃO	CLASSE
ID	IDENTIFICADOR	NUM.	NÃO NULO	SIMPLES
AMOUNT	NÚMERO DE LUGARES CLASSIFICADOS POR ESTA CATEGORIA.	NUM.	NÃO NULO	SIMPLES
ICON	ENDEREÇO DA IMAGEM DA CATEGORIA.	URL	-	SIMPLES
NAME	NOME DA CATEGORIA.	TEXTO	NÃO NULO	SIMPLES

Table 1. Dicionário de dados - Elemento retornado pela função Categoria: Categoria utilizada para classificar objetos criados por usuários do Wikimapia (tabela criada pelo autor, em 2013)

A *folksonomia* geralmente é representada por nuvem de palavras, onde a variação de tamanho de um termo é proporcional a sua frequência de uso em anotações do sistema. A figura 3 foi gerada utilizando a ferramenta Wordle¹⁷, desconsiderando as 5 categorias mais frequentes (*place without photos*, *place without description*, *place without category*, *building without address*, *place without polygon*).

Figure 3. Nuvem de anotações da folksonomia (figura criada pelo autor, em 2013)



Limpeza de dados. A etapa de limpeza de dados foi realizada aplicando a ferramenta de indexação Lucene da fundação Apache¹⁸. Essa ferramenta não só dispõe de um módulo indexador, mas fornece recursos de análise e tratamento de palavras comumente necessários em processos de limpeza de dados. Esta ferramenta foi elegida devido ao reaproveitamento de trabalhos bem sucedidos do grupo de pesquisa. Contudo, existem outras ferramentas para implementar esta etapa, como o TSearch2¹⁹.

Três processos foram aplicados aos termos da *folksonomia* e das ontologias comparadas: *tokenização*, remoção de *stopwords* e *stemming*.

Enriquecimento semântico. A expansão de *tokens* em conjunto de sinônimos foi restrita a termos da língua inglesa e utilizou a base léxica WordNet. Para melhores resul-

¹⁷<http://www.wordle.net>. Acesso em out. 2013.

¹⁸<https://lucene.apache.org/>. Acesso em nov. 2013.

¹⁹http://sai.msu.su/~megera/postgres/gist/tsearch/V2/docs/Tsearch_V2_Readme.html. Acesso em mar. 2014.

tados nas comparações entre termos da ontologia e da *folksonomia*, os *tokens* resultantes do processo de limpeza são submetidos como parâmetros de consulta à base do WordNet e recuperados os conjuntos de sinônimos de cada termo. A comunicação com a API do WordNet é realizada pelo conversor.

Alinhamento e/ou Mapeamento Ontológico. O alinhamento das categorias do Wikimapia com conceitos de ontologias (*e.g.*, DBpedia, GeoNames e LGD) apresentada neste trabalho é uma adaptação das idéias apresentadas no artigo sobre LOM [Li 2004]. Foram selecionadas 101 categorias sob o critério de maior número de objetos anotados. Os conceitos de ontologias selecionados foram 143 conceitos do GeoNames²⁰, já mapeados para conceitos do DBpedia e LGD. A tabela 2 mostra o número de alinhamentos obtidos nos diferentes níveis de pareamento.

RESULTADOS	
NÚMERO DE CATEGORIAS UTILIZADAS DO WIKIMAPIA	101
NÚMERO DE CONCEITOS DE ONTOLOGIAS EXTERNAS	143
NÚMERO DE ALINHAMENTOS POR TEXTO ORIGINAL	15
NÚMERO DE ALINHAMENTO POR TOKENS	7
NÚMERO DE ALINHAMENTO POR STEMMING TOKENS	2
NÚMERO DE ALINHAMENTO POR SYMSET	0
TOTAL DE ALINHAMENTOS	24

Table 2. Tabela de resultados do alinhamento (tabela criada pelo autor, em 2013)

A similaridade entre rótulos é um número real entre 0 e 1. As relações de equivalência e proximidade de conceitos são obtidas classificando os resultados, baseando-se no valor de similaridade dos rótulos (*e.g.*, resultados com valor de similaridade dentro do intervalo (1; 0,9) são considerados idênticos, no intervalo [0,9; 0,5] são considerados similares, e resultados menores que 0,5 não são considerados similares).

Triplificação. O processo de transformação dos dados do Wikimapia em uma coleção RDF, chamada LinkMapia, iniciou-se com a conversão de sua *folksonomia* em uma ontologia rasa, *i.e.* uma árvore de altura 1. Foi escolhida a classe *Place* como derivada direta da classe *Thing*. Todas as categorias da *folksonomia* foram inicialmente classificadas como subclasses de *Place*. O esquema *Web Ontology Language*²¹ (OWL) foi baseado em um protótipo gerado automaticamente pela ferramenta Protege²², versão 4.3.0.

A coleção de dados resultante ainda não está ligada a outras fontes. A ontologia precisa ser correlacionada com ontologias externas como LGD e GeoNames. O mapeamento da ontologia gerada para as ontologias externas é realizado convertendo as relações de equivalência de rótulos, obtidas na etapa de alinhamento e/ou mapeamento ontológico, em propriedades *rdfs:equivalentClass*, enquanto que relações de proximidade de conceito são convertidas em propriedades temporárias *nearConcept* para, posteriormente, facilitar a estruturação da ontologia pela comunidade do sistema colaborativo, como o caso da identificação da correlação entre *Hotel* e *Motel*, ilustrado na figura 4. A figura 5 ilustra o RDF de exemplo gerado pelo Protege, utilizado como esquema para a conversão.

²⁰Disponível em: http://www.geonames.org/ontology/mappings_v3.01.rdf. Acesso em nov. 2013.

²¹Disponível em: <http://www.w3.org/2001/sw/wiki/OWL>. Acesso em mar. 2014.

²²Disponível em: <http://protege.stanford.edu/>. Acesso em mar. 2014.

Figure 4. Triplificação de categoria (figura criada pelo autor, em 2013)

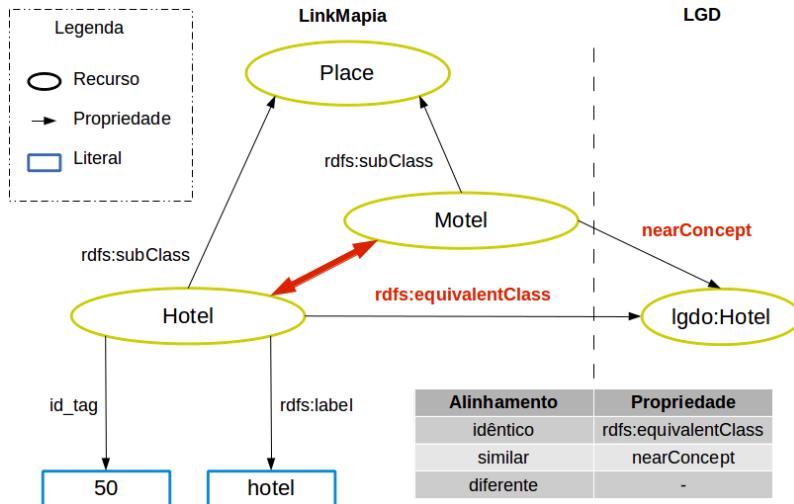
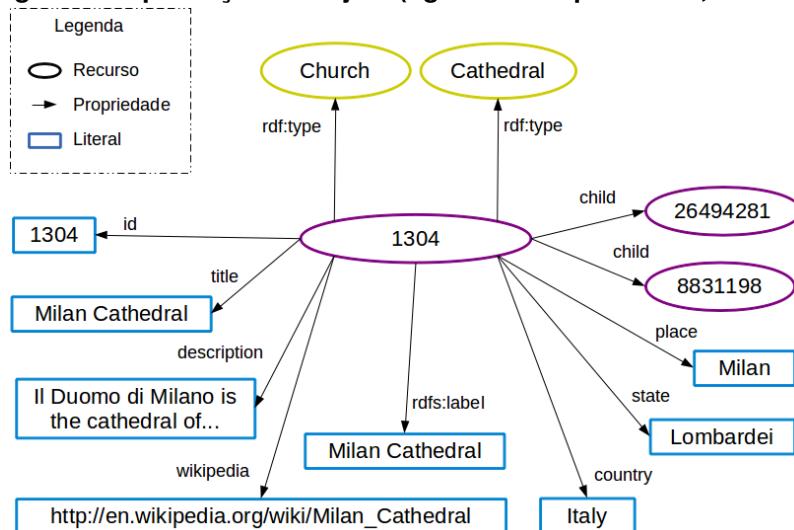


Figure 5. Triplificação de objeto (figura criada pelo autor, em 2013)



4. Resultados esperados

Com a nova coleção de dados LinkMapia, é possível responder perguntas como: **Quais são as opções de locais para dormir (como hotel, pensão, albergue) a até quinhentos metros do centro tecnológico da UFSC?** Dependendo de como a ontologia foi estruturada pela comunidade, é possível utilizar termos mais abrangentes (locais para dormir) para considerar na consulta uma coleção de conceitos (hotel, pensão, albergue), possibilitando diferentes granularidades para a categoria do objeto espacial.

O endpoint SPARQL permite consultas como: **Qual o restaurante conhecido como "Universitário" a até 200m do Centro de Eventos?** (figura 6). Esta consulta considera não apenas objetos geográficos do LGD classificados como *lgdo:Restaurant* como também da Wikimapia, anotados como *restaurant* e conceitos mais específicos (*seafood restaurant*, *drive-in restaurant*, dentre outros).

Figure 6. Consulta SPARQL - Qual o restaurante conhecido como "Universitário" a até 200m do Centro de Eventos?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?local ?nomeLocal {
?local
  a lgdo:Restaurant;
  rdfs:label ?nomeLocal ;
  geom:geometry [ ogc:asWKT ?geo ] .
FILTER( regex(?nomeLocal, "Universitário") &&
       bif:st_intersects ( ?geo,
                           bif:st_point (-48.52015,-27.60225), 0.2) ) .
}
```

5. Trabalhos Correlatos

Os principais trabalhos relacionados são o Projeto SEEK²³, o Linked Geo Data [Stadler et al. 2012], GeoLinkedData²⁴ (Espanha), o Linked Wikimapia²⁵, o OurMap [Gonzalez et al. 2013], o FolksOntology [Damme et al. 2007] e o LOM [Li 2004].

A iniciativa deste trabalho foi a obtenção de dados geográficos anotados para realizar análises semânticas de trajetórias de objetos móveis. Pretende-se com este trabalho apoiar trabalhos futuros, como o projeto *Semantic Enrichment of trajectory Knowledge discovery* (SEEK) e Baquara [Fileto et al. 2013].

O enriquecimento semântico de VGI já foi realizado em dados do projeto OSM [Stadler et al. 2012]. Outra coleção de dados geográficos ligados é o GeoLinkedData, uma iniciativa do *Ontology Engineering Group* (OEG) destinada ao enriquecimento da Web de dados com dados geoespaciais da Espanha. Há outro projeto de publicação de dados do Wikimapia, o Linked Wikimapia. Entretanto, são publicados dados lidados de objetos geográficos, sem ontologia para representar a *folksonomia*. Os dados são ligados com o DBpedia a partir da conversão dos *links* já existentes para páginas do Wikipedia.

Uma diferente abordagem é seguida no sistema de coleta colaborativa *OurMap* [Gonzalez et al. 2013], onde o sistema de coleta colaborativa utiliza anotações semânticas e o usuário tem a opção de gerenciar a ontologia.

O processo de derivação de uma ontologia a partir de uma *folksonomia* é discutido por [Damme et al. 2007]. Este trabalho inspirou o processo proposto. Outro trabalho que ajudou na definição e implementação do processo foi o de [Li 2004].

6. Conclusões e Trabalhos Futuros

Este trabalho apresenta uma proposta de conversão de dados geográficos coletados voluntariamente em dados ligados, descreve uma abordagem de implementação e explora os resultados esperados pela conversão. Trabalhos futuros incluem análises de outros sistemas

²³Disponível em: <http://www.seek-project.eu/>. Acesso em nov. 2013.

²⁴Disponível em: <http://geo.linkeddata.es/web/guest/home>. Acesso em nov. 2013.

²⁵Disponível em: <http://openeanwrap.appspot.com/>. Acesso em nov. 2013.

colaborativos, elaboração e implementação de novas propostas de processo e análises de novos casos de estudo para determinar a melhor maneira de converter *folksonomias* em ontologias. A elaboração do processo proposto, sua implementação e resultados preliminares demonstram que a abordagem proposta é promissora.

Agradecimentos. Este trabalho contou com o apoio do projeto European Union's IRSES-SEEK (concessão 295179), do CNPq (concessão 478634/2011-0), da CAPES, e da FEESC. Agradecimentos também a comunidade Wikimapia, e a Willian Ventura Koerich e André Salvaro Furtado pelo apoio técnico.

References

- Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web: Scientific American. *Scientific American*, 284(5).
- Casanova, M., Câmara, G., Davis, C., Vinhas, L., and de Queiroz, G. R., editors (2005). *Bancos de Dados Geográficos*. MundoGEO.
- Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78.
- Damme, C. V., Hepp, M., and Siorpaes, K. (2007). Folksontology: An integrated approach for turning folksonomies into ontologies. In *Proceedings of the ESWC Workshop Bridging the Gap between Semantic Web and Web 2.0*. Springer.
- Fileto, R., Krüger, M., Pelekis, N., Theodoridis, Y., and Renso, C. (2013). Baquara: A holistic ontological framework for movement analysis using linked data. In *ER*, volume 8217 of *Lecture Notes in Computer Science*, pages 342–355. Springer.
- García-castro, L. J. and García, E. (2011). Folksonomies behind the scenes.
- Gonzalez, A., Izidoro, D., Willrich, R., and Santos, C. (2013). Representação Aberta e Semântica de Anotações de Incidentes em Mapas Web. In *Simpósio Brasileiro de Sistemas Multimídia e Web*, pages 1–12.
- Goodchild, M. F. (2007). Citizens as voluntary sensors: spatial data infrastructure in the world of web 2.0. *International Journal of Spatial Data Infrastructures Research*, 2:24–32.
- Guarino, N. (1998). Formal ontology and information systems. pages 3–15. IOS Press.
- Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition.
- Kyzirakos, K., Karpathiotakis, M., and Koubarakis, M. (2012). Strabon: A semantic geospatial dbms. In *International Semantic Web Conference (1)*, volume 7649 of *Lecture Notes in Computer Science*, pages 295–311. Springer.
- Li, J. (2004). Lom: A lexicon-based ontology mapping tool. In *Proceedings of the Performance Metrics for Intelligent Systems (PerMIS)*, page 2004.
- Rigaux, P., Scholl, M., and Voisard, A. (2000). *Introduction to Spatial Databases: Applications to GIS*. Morgan Kaufmann.
- Stadler, C., Lehmann, J., Höffner, K., and Auer, S. (2012). Linkedgeodata: A core for a web of spatial open data. *Semantic Web Journal*, 3(4):333–354.

Uma Aplicação baseada em SIG para Análise de Infrações Cometidas por Menores Infratores: Estudo de caso no Município de Vitória no estado do Espírito Santo

Maria Luiza G. Silva¹, Edvaldo C. Mantovanelli¹, Jefferson O. Andrade¹, Karin S. Komati¹

¹Instituto Federal do Espírito Santo (Ifes – Campus Serra)
Rodovia ES-010 – Km 6,5 – Manguinhos 29.173-087 – Serra – ES
mlguimaraess@gmail.com, ecmantovanelli@gmail.com, joandrade@ifes.edu.br, kkomati@ifes.edu.br

Abstract. This paper describes an application based on GIS (Geographic Information System), which describes the methodology for the creation of thematic maps of violations of under age that occurred in the year 2012, located in the state of Espírito Santo. Thematic maps will be created aiming the analysis with respect to time: for different times of day, days of the week and months of the year. This solution uses the open source tool Quantum GIS.

Resumo. Este trabalho descreve uma aplicação baseada em SIG (Sistema de Informação Geográfica), onde se descreve a metodologia de criação de mapas temáticos de infrações de menores ocorridos no ano de 2012, localizado no Estado do Espírito Santo. Os mapas temáticos serão criados visando a análise com relação ao tempo: para os diversos períodos do dia, os dias da semana e os meses do ano. Esta solução usa a ferramenta livre Quantum GIS.

1. Introdução

A informação contida nas bases de dados criminais é de fundamental importância para a realização da análise criminal, isto é, quando e onde acontecem os crimes, e consequentemente, para permitir que as autoridades de segurança pública tenham mais mecanismos para realizar o planejamento estratégico e do efetivo controle operacional das suas áreas. A interpretação e visualização dos grandes volumes de dados em informação que sirvam de apoio ao planejamento de ações estratégicas exigem um esforço considerável. Nesse contexto, a utilização de um SIG (Sistema de Informação Geográfica) traz uma nova forma de visualização e interpretação dos dados, sendo uma ferramenta capaz de auxiliar o planejamento estratégico, ligando o crime ao local do acontecimento [Ribeiro et. al, 2007; Máximo e Loch, 2002].

Este trabalho tem como propósito mapear geograficamente as infrações cometidas por menores que ocorrem no município de Vitória no estado do Espírito Santo durante o ano de 2012, enfatizando o processo desde a aquisição dos dados até a criação dos mapas temáticos usando uma ferramenta SIG de código aberto. Os mapas temáticos serão criados focados para a análise com relação ao tempo: para os diversos períodos do dia, os dias da semana e os meses do ano. Este trabalho surgiu por demanda da própria polícia militar do ES que precisa analisar as informações, mas cujos dados encontram-se dispersos em diferentes setores, e em diferentes formatos. Isto é, antes da inserção dos dados no SIG, houve um esforço para integração de todas as informações de infrações.

2. Metodologia e Desenvolvimento dos Mapas Temáticos

A metodologia usada para este trabalho obedeceu a um esquema sequencial, composto pelas seguintes etapas: o levantamento de dados pré-existentes, a inserção dos dados no SIG, a análise em SIG e finalmente a análise dos resultados, de modo semelhante a outros trabalhos, tal como o de Paixão e Komati (2013).

Os dados das infrações de menores foram cedidos pela GEAC (Gerência de Estatística e Análise Criminal) sitiada na SESP (Secretaria de Estado da Segurança Pública e Defesa Social) e pela PMES (Polícia Militar do Espírito Santo). Os dados que compõem o layout dos mapas foram fornecidos pelo GEOBASES (Sistema Integrado de Bases Geoespaciais do Estado do Espírito Santo). Em todos os casos, enviou-se documento solicitando os dados com a explicação do propósito do trabalho para o responsável de cada órgão. Com isso, fechando o primeiro passo de levantamento de dados pré-existentes.

Para a criação dos mapas temáticos foi utilizado o software livre Quantum GIS (QGIS) [QGIS PSC, 2012], um software de Sistema de Informação Geográfica (SIG). No QGIS foi habilitado um *plugin* específico para a manipulação dos dados, onde estes posteriormente são apresentados como um mapa de calor, este *plugin* é denominado HeatMap.

O Heatmap permite criar um mapa de calor através de um mapa de pontos, em nosso estudo, cada ponto é composto das informações da infração e de uma coordenada geográfica. As informações relativas às infrações são endereço, dia da semana, horário da ocorrência, bairro e município. Esse mapa de calor apresenta a densidade ou a magnitude das informações relacionadas dos pontos, onde “áreas quentes” podem ser identificadas facilmente. A Figura 1 mostra à esquerda uma imagem de pontos (imagem de entrada do mapa de calor) e à sua direita, o mapa de calor resultante sobreposto à imagem de entrada.

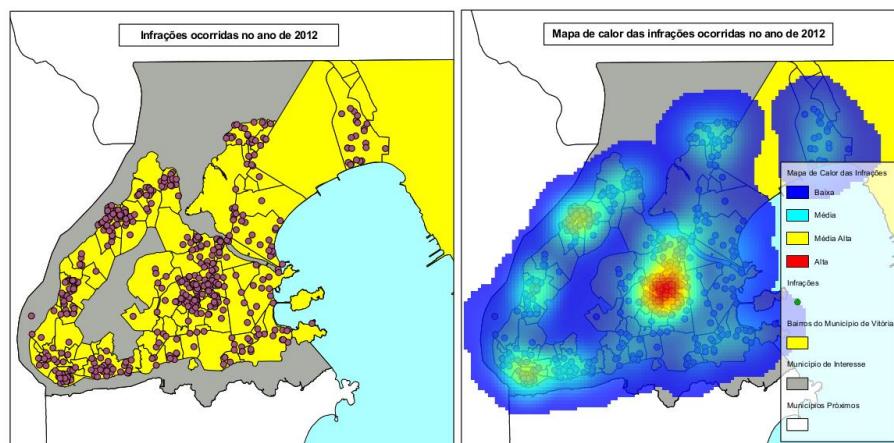


Figura 1. Imagem de pontos à esquerda e mapa de calor associado à direita

3. Resultados e Discussão

Foram criados 18 mapas temáticos, 2 mapas associados aos dias da semana (dias de semana e finais de semana) – Figura 2; 4 mapas associados aos períodos do dia (de 0h às 6h; das 6h às 12h; das 12h às 18h e das 18h às 24h) – Figura 3 e 12 mapas associados aos meses do ano (não apresentados no artigo por restrição de páginas).

Como pode ser observado nos mapas da Figura 2, na região noroeste há uma diminuição na quantidade de crimes ocorridos nos dias de semana para o final de semana. O contrário é visto na região sudoeste em que existe um aumento dos crimes. Na região central há uma pequena variação, contudo é a região com o maior índice de infrações.

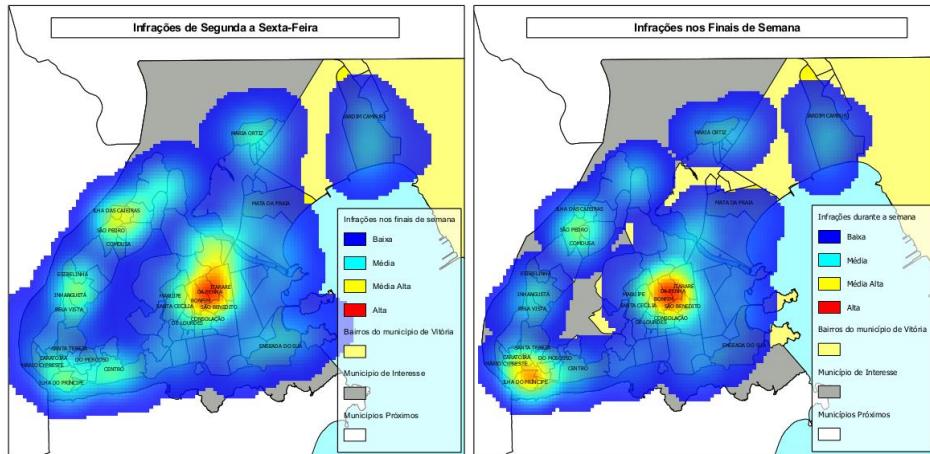


Figura 2. Infrações avaliadas quanto aos dias da semana

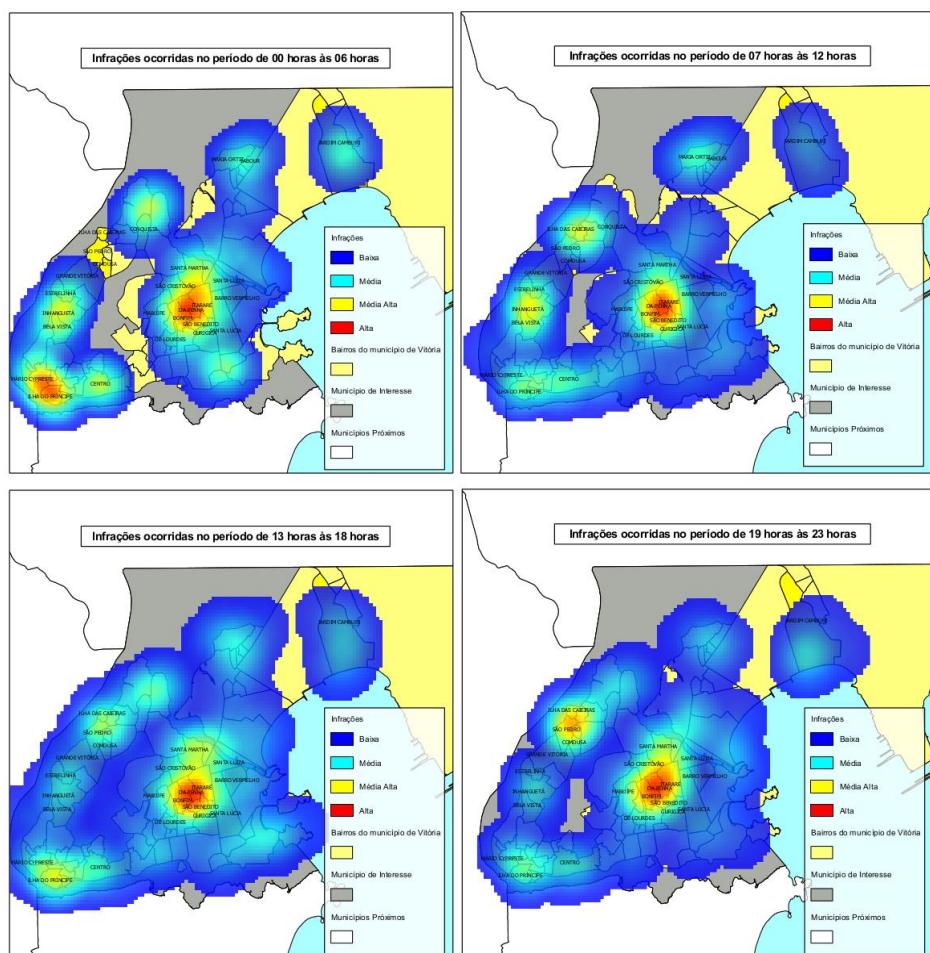


Figura 3. Infrações avaliadas quanto aos períodos do dia

Na Figura 3 é mostrada uma comparação entre períodos do dia. A região sudoeste do mapa ocorre um número maior de infrações no período de meia noite as seis da manhã comparado ao período das seis até o meio dia. Assim como um deslocamento para o norte da capital das infrações no período da madrugada em comparação com a manha. Enquanto que na região central mesmo com uma expansão da região as infrações possuem um mesmo local de foco e continua sendo destaque de ocorrências em ambos os períodos. A parte inferior da Figura 3, mostra os mapas na segunda metade do dia. Na região sudoeste há uma maior concentração de infrações no período das 12h as 18 horas. Contudo no período das 18h as 24h as infrações são predominantes nas regiões oeste e central do mapa. Novamente a região central se destaca novamente com a predominância das infrações em ambos os períodos do mapa.

Com relação aos meses do ano, os meses de janeiro a março mostram picos na região sudoeste e nem tanto na região central. Nos outros meses do ano, a região central mostra elevada concentração de ocorrências.

4. Considerações Finais

O propósito deste trabalho foi o de mapear geograficamente a ocorrência das infrações por menores que ocorrem no município de Vitória/ES durante o ano de 2012. Os mapas temáticos criados focaram na variação da informação com o tempo: para os diversos períodos do dia, os dias da semana e os meses do ano.

Infelizmente, os dados mostraram que há uma região central, que engloba os bairros de Itararé, Bairro da Penha, Bonfim e São Cristóvão, que sempre apresenta elevada concentração de ocorrências, independentemente de variações de dia, horário ou mês. Há outros três pontos focais: região do centro da cidade, em torno no bairro Mario Cypreste e em torno do bairro Ilha das Caieiras. Nota-se que há bairros em que não há nenhuma ocorrência, como em Mata da Praia e Praia do Canto, considerados bairros nobres do município.

Há muitos trabalhos futuros possíveis, desde a análise de outros anos (além do ano de 2012), além da criação de outros tipos de mapas temáticos, por exemplo, separando por tipo de infração (posse de entorpecentes, furto, entre outros) ou por idade do menor. Futuramente, construir uma correlação do local de sua residência com relação aos locais dos atos infracionais podem definir uma área de atuação dos menores.

Referências

- Máximo, A. A., Loch, C. (2002). “A utilização do Sistema de Informação Geográfica (SIG) e do Sistema de Posicionamento Global (GPS) no combate da criminalidade pelos serviços de segurança pública”. Florianópolis (SC): COBRAC 2002.
- Paixão, W. R., Komati, K. S. (2013). “Uma Aplicação baseada em SIG para Análise de Acidentes de Trânsito: Estudo de caso na Rodovia BR-101/ES”. Em: Anais do IX Escola Regional de Banco de Dados 2013 (IX ERBD). Camburiú/SC.
- QGIS. (2012) “Quantum GIS”, <http://www.qgis.org>, Agosto.
- Ribeiro, T. V. B., Moreira, P. D. O., Silva Filho, L. A., De Souza, C. R. B., Betini, R. C. (2007). “Arquitetura de um SmallSIG para apoio ao Planejamento Estratégico na Área de Segurança Pública”. Em: Anais da Conferência Latino-Americana de Informática, 2007, San José.

OMT-G Design: uma ferramenta para modelagem de dados espaciais

Álvaro Osvaldo Teixeira Martínez, Angelo Augusto Frozza

Instituto Federal Catarinense (IFC) - Câmpus Camboriú
Rua Joaquim Garcia, S/N - 88.340-960 - Camboriú - SC

alvaro@lorenagroup.net, frozza@ifc-camboriu.edu.br

Resumo. Geralmente, a tarefa de modelar um Banco de Dados (BD) fica limitada a uma representação que pode não ser fiel aos propósitos do BD caso a notação da modelagem seja insuficiente ou carente de recursos que permitam abstrair a realidade. O uso da abordagem Entidade-Relacionamento, por exemplo, para modelar Bancos de Dados Geográficos (BDG) pode causar perda de parte da semântica do BDG. Em função disso, foram propostos modelos específicos para BDG, como o modelo OMT-G (*Object Modeling Technique for Geographic Applications*). Este artigo apresenta uma ferramenta desenvolvida para a modelagem de BDG, denominada OMT-G Design.

1. Introdução

Em geral, os bancos de dados geográficos (BDG) têm cumprido a função de armazenar, ordenar e realizar consultas sobre os dados geográficos. Contudo, em seu projeto, é necessário especificar um modelo de representação. Geralmente, esse modelo é utilizado em uma etapa denominada modelagem de dados. Essa é uma atividade complexa, que envolve transformar o espaço em uma representação discreta adequada ao fenômeno que se deseja trabalhar (QUEIROZ e FERREIRA, 2006). Os modelos de dados tradicionais apresentam limitações para aplicações geográficas, pois não possuem primitivas apropriadas para representar corretamente a semântica dos dados geográficos (QUEIROZ e FERREIRA, 2006).

Diante dessa dificuldade, Borges (1997) desenvolveu um modelo para dados geográficos, denominado OMT-G (*Object Modeling Technique for Geographic Applications*), que supre essa carência de primitivas e busca ser mais fiel à realidade a ser modelada, utilizando um conjunto menor de objetos gráficos do que seria utilizado em outros modelos para dados geográficos. No entanto, esse modelo ainda carece de ferramentas para apoiar a construção de seus diagramas (FROZZA, 2007). Este artigo apresenta a ferramenta OMT-G Design e suas funcionalidades que suportam o modelo OMT-G proposto por Borges (1997).

O presente artigo introduz o modelo OMT-G na seção 2 e apresenta a ferramenta OMT-G Design na seção 3, finalizando com as considerações a respeito da ferramenta e os meios para contribuir com o projeto.

2. O modelo OMT-G

Borges (1997) descreve que o paradigma da orientação a objetos é mais propício para um SIG (Sistema de Informações Geográficas). Entre as principais possibilidades, está a representação do mundo real diretamente no modelo conceitual e com mecanismos de abstração capazes de representar estruturas complexas, como os objetos geométricos, os quais podem ser alterados em um período de tempo.

Uma das propostas de modelagem com esse paradigma é o modelo denominado OMT-G (BORGES, DAVIS JUNIOR e LAENDER, 2001), que foi proposto assumindo algumas vantagens em relação a outros padrões de modelagem (p.ex. GeoOOA e UML-GeoFrame). Entre as vantagens, destacam-se a representação de primitivas geográficas, suas formas de relacionamentos e restrições de integridade.

As primitivas geográficas são organizadas em dois grupos: Geo Objetos - para representar objetos discretos e estruturas em rede (topologia); e, Geo Campos - para representar dados distribuídos continuamente no espaço (ex. minerais ou sementes sob o solo). A Figura 01 exibe a representação das classes do modelo OMT-G (usando o padrão de desenho do OMT-G Design).

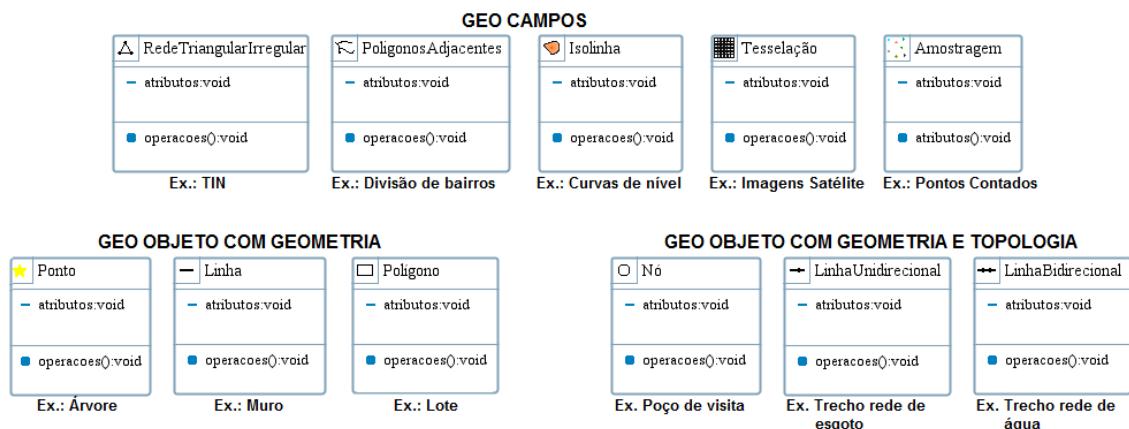


Figura 01 - Representação das classes OMT-G

(Fonte: adaptado de BORGES, 1997)

Borges (1997), ainda, defende que a maioria dos modelos possui uma carência em definir as relações existentes nos fenômenos do mundo real. Para isso ela acrescenta ao modelo OMT-G conceitos provenientes da modelagem de objetos, tais como: cardinalidade; relacionamentos simples, para representar as relações estruturais entre classes; relacionamentos espaciais, para relações topológicas, métricas, ordinais e *fuzzy* entre os objetos; e, agregações e generalizações, ambas podendo ser espaciais ou não (Figura 02). Com isso, aumenta-se a fidelidade entre a representação do universo físico modelado e sua representação computacional.

Diante das virtudes citadas, considera-se o OMT-G adequado para a tarefa de modelagem de bancos de dados geográficos e, entre seus principais usuários têm-se: PRODABEL - Empresa de Informática e Informação do Município de Belo Horizonte; CONCAR - Comissão Nacional de Cartografia; e, ANA - Agência Nacional de Águas (FROZZA, 2007). Porém, ainda existem poucas ferramentas que suportem o modelo OMT-G, a maioria contendo poucos recursos e limitações de plataforma. As ferramentas conhecidas são:

- a) uma customização para o *Enterprise Arquitecture* usada na PRODABEL;
- b) um *stencil* para uso com o *Microsoft Visio* (DAVIS, 2010);
- c) um *plug-in* para o *StarUML* (DAVIS, 2010). A partir deste *plug-in* é possível fazer conversão de um diagrama OMT-G para *scripts SQL* através de outro programa ainda não disponível publicamente (SCHALY, 2009).

O OMT-G *Design* é uma nova opção, sendo a única que é ao mesmo tempo livre e aberta (*open source*), multiplataforma, e permite exportar de forma integrada os modelos criados para outros formatos, além de *scripts SQL*.

3. A ferramenta OMT-G *Design*

Para aplicações geográficas, os modelos de dados são classificados de acordo com o nível de abstração empregado: (1) mundo real; (2) representação conceitual; (3) apresentação; e (4) implementação (BORGES, DAVIS JUNIOR e LAENDER, 2001).

O OMT-G *Design* é uma ferramenta para modelagem de BDG, seguindo o modelo OMT-G proposto por Borges (1997) que atende os níveis de abstração 2 e 3 acima. Além disso, foram acrescidos mecanismos de tradução dos diagramas desenvolvidos pelo usuário para *scripts SQL* (*Structured Query Language*) e XML (*eXtensible Markup Language*), permitindo que a ferramenta também atenda ao quarto nível de abstração. Ainda, a ferramenta é flexível o suficiente para permitir a criação de novos tradutores e módulos, sem que isso signifique uma codificação extensa ou um conhecimento profundo da arquitetura da plataforma.

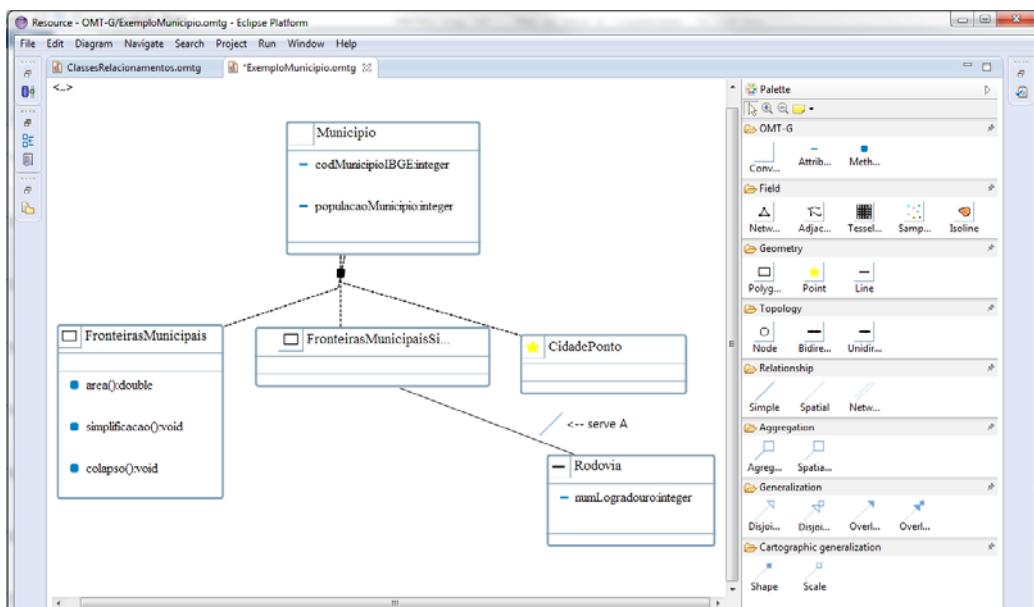


Figura 02 – Interface do OMT-G *Design* e palette de recursos

O seu desenvolvimento partiu da análise de 12 (doze) requisitos pré-definidos em Martínez (2013), entre eles: documentação; licença; portabilidade; recursos de exportação/importação; versão da UML utilizada; capacidade para estender a ferramenta; modularidade; custo; código aberto; e, usabilidade. Como resultado, escolheu-se o *Eclipse Graphical Modeling Framework* como plataforma base. Uma das vantagens desse *framework* é o suporte à arquitetura MDA (*Model Driven*

Architecture), que permitiu desenvolver a ferramenta praticamente sem escrever código fonte. Quanto aos mecanismos de tradução, foi adotado o *framework XPand* sob o *Modeling Workflow Engine*, conforme descrito em Martínez (2013). Essa escolha permite escrever novos tradutores do modelo OMT-G para outras linguagens além do SQL, sem a necessidade de um conhecimento específico da arquitetura da ferramenta. A Figura 02 apresenta a interface principal do OMT-G *Design* (*Eclipse*) com um fragmento de diagrama e os recursos do modelo suportados (*Palette*).

4. Considerações finais

Apesar do OMT-G *Design* permitir seu uso em ambiente de produção, ainda restam complementar alguns requisitos da OMT-G, tal como a implementação das restrições espaciais do modelo. Contudo, seu uso é encorajado, até mesmo para que se inicie a avaliação e os novos ciclos de atualizações do *software*. As principais vantagens da ferramenta em relação às opções disponíveis já citadas são: ser multiplataforma, permitir a exportação do modelo de banco criado para *scripts* SQL e XML; ter o código fonte aberto em repositório público; permitir a criação de tradutores para outros formatos. Acredita-se que o fato de se utilizar MDA pode fomentar o interesse da comunidade acadêmica em explorar as facilidades dessa arquitetura para outros projetos, além da expansão do OMT-G *Design*. A ferramenta (binário e código fonte) pode ser encontrada no *link*: <http://code.google.com/p/omt-g-design/>.

Referências bibliográficas

- BORGES, K. A. de V. **Modelagem de dados geográficos**: uma extensão do modelo OMT para aplicações geográficas. 1997. 128f. Dissertação (Mestrado em Administração Pública) - Fundação João Pinheiro, Belo Horizonte.
- BORGES, K. A. de V.; DAVIS JUNIOR, C. A.; LAENDER, A. H. F. OMT-G: An Object-Oriented Data Model for Geographic Applications. **Geoinformatica**, Dordrecht, Holanda, v. 5, n. 3, p. 221-260, 2001.
- DAVIS JUNIOR, C. A. **Object Modeling Technique for Geographic Applications - OMT-G**. DocuWiki. [S.I.], 11 jun. 2010. Disponível em: <<http://homepages.dcc.ufmg.br/~clodoveu/DocuWiki/doku.php?id=omtg>>
- FROZZA, A. A. **Um método para determinar a equivalência semântica entre esquemas GML**. 2007. 139f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina – UFSC, Florianópolis.
- MARTÍNEZ, Á. O. T. **OMT-G Design**: ferramenta para projeto de banco de dados geográficos suportando o modelo OMT-G. 2013. 88f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Instituto Federal de Educação, Ciência e Tecnologia - IFC, Camboriú.
- QUEIROZ, G. R.; FERREIRA, K. R. (Comp.). Tutorial sobre Bancos de Dados Geográficos. GeoBrasil, 2006. **Anais...** São José dos Campos: INPE, 2006. 104 p.
- SCHALY, K. W. **Ferramenta para Criação de Bancos de Dados Geográficos a partir de Diagramas OMT-G**. 2009. 79f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade do Planalto Catarinense - UNIPLAC, Lages - SC.

Armazenamento de trajetórias de objetos móveis para a Plataforma *UrbanMob*

Gustavo Costa Meireles, Angelo Augusto Fozza

Instituto Federal Catarinense (IFC) - Câmpus Camboriú
Rua Joaquim Garcia, S/N - 88.340-960 - Camboriú - SC

{gustavo, fozza}@ifc-camboriu.edu.br

Resumo. A plataforma *UrbanMob* visa disponibilizar um banco de dados (BD) de trajetórias urbanas de cidades brasileiras, sobre o qual podem ser aplicadas técnicas de mineração de dados com a finalidade de gerar informações para tomada de decisão na área de Gestão Urbana. Neste artigo se apresenta parte desse projeto, que é o web service para o armazenamento dos dados. Para tanto, foram definidas três metas: criar os modelos de arquivos JSON para integração entre o web service e os dispositivos móveis; desenvolver um web service para receber os arquivos JSON e salvar as informações no BD; criar um modelo de armazenamento de dados consistente.

1. Introdução

Com o advento da globalização, a necessidade de controle sobre o comportamento de objetos (p.ex., pessoas e veículos) que se movem sobre o globo terrestre vem se elevando rapidamente, despertando cada vez mais interesse de empresas e órgãos governamentais (TORRES, 2009). A tomada de decisão em Gestão Urbana permite, entre outras coisas, propor alterações no trânsito para reduzir pontos de congestionamento, identificar pontos de aglomeração de indivíduos ou que representam pontos de interesse público, identificar relacionamentos entre esses pontos, além de identificar redes sociais criadas ao redor desses pontos de interesse.

Telefones e outros dispositivos móveis têm se tornado popular e permitem capturar dados do usuário (vestígios digitais), como sua identificação e localização em dado momento (TORRES, 2009). Capturar esse tipo de dado é importante, contudo a coleta de trajetórias gera dados brutos que, isoladamente, não trazem informações relevantes para a análise do comportamento humano. É necessário, portanto, aplicar algoritmos computacionais que extraiam informações desses dados, como locais em que o objeto parou ou diminuiu consideravelmente sua velocidade, para que possam ser reconhecidos pontos importantes, como a residência, o local de trabalho, os restaurantes mais visitados, pontos de congestionamentos etc. (TORRES, 2009).

Pensando nisso, foi proposto o desenvolvimento da Plataforma *UrbanMob*, a qual visa disponibilizar uma grande base de dados de trajetórias urbanas em cidades brasileiras e sobre ela aplicar técnicas de mineração de dados (*datamining*) com a finalidade de gerar informações para tomada de decisão na área de Gestão Urbana. A motivação para este projeto está relacionada à pouca disponibilidade de trabalhos focados no armazenamento de trajetórias, uma vez que a maior parte da literatura relata

algoritmos de mineração de dados. Nessa linha, ainda, os trabalhos correlatos utilizam bases de testes que não têm dados reais ou cujos dados não refletem realidade brasileira.

Nesse artigo apresenta-se o *web service* criado para armazenamento de trajetórias urbanas, coletadas de usuários voluntários por meio de um aplicativo para dispositivos móveis. Além desta seção introdutória, o artigo está organizado em mais três seções. A Seção 2 apresenta conceitos básicos de trajetórias de objetos móveis. A Seção 3 apresenta o *web service* que foi desenvolvido e os modelos de banco de dados e dos arquivos de interação. A Seção 4 apresenta as considerações finais do trabalho.

2. Trajetórias de objetos móveis

As trajetórias estão presentes em quase toda parte, do movimento das pessoas à circulação de animais ou veículos. Frente a esse contexto, a disponibilidade de dados de trajetórias abriu novas perspectivas para um grande número de aplicações, construídas sobre o conhecimento dos movimentos dos objetos e voltadas para áreas que vão desde transporte e logística, passando pela ecologia e a antropologia (SEEK, 2012; SPACCAPIETRA *et al.*, 2008).

A seguir são apresentados alguns conceitos básicos (BOGORNY *et al.*, 2014):

- *Ponto*: é um registro (x, y, t) , no qual x e y são as coordenadas geográficas e t é a identificação de tempo em que a coordenada foi coletada;
- *Trajetória*: é uma lista ordenada de n pontos $(p^1, p^2, p^3, \dots, p_n)$. Uma trajetória pode conter diversas características, p.ex., velocidade, informações climáticas, entre outras, que podem ser coletadas automaticamente ou fornecidas pelo usuário;
- *Subtrajetória*: é uma trajetória que faz parte de outra trajetória maior;
- *Dados semânticos*: são dados que buscam complementar as informações referentes às trajetórias, agregando valor (p.ex.: meio de transporte, velocidade, informações climáticas, entre outras);
- *Objeto*: é o objeto em movimento e que transporta um dispositivo móvel. Pode ser uma pessoa, um carro, um animal, um robô etc.

3. Web service para armazenamento de trajetórias de objetos móveis

O modelo de dados proposto neste artigo considera que uma trajetória semântica é constituída de diferentes subtrajetórias semânticas, devendo possuir o maior número de dados semânticos quanto possível. As subtrajetórias são capturadas pelos usuários e devem ser incrementadas com dados semânticos. Esse processo está inserido em um modelo abrangente que relaciona vários tipos de informações contextuais que podem ser usados para enriquecer semanticamente as trajetórias.

O trabalho baseou-se na proposta de Bogorný *et al.* (2014), que desenvolveram um modelo de dados para trajetórias semânticas que também prevê sua aplicação em tarefas de mineração sobre trajetórias. O modelo foi adaptado para o domínio do Turismo, foco inicial do trabalho (MEIRELES, 2013). A Figura 1 apresenta o modelo, sendo que as classes *Trajetoria*, *SubTrajetoria* e *Ponto*, representam os principais dados necessários para a definição de uma trajetória. As classes *Usuario* e *Dispositivo* são usadas para armazenar os dados dos usuários que alimentam o banco de dados de trajetórias. As demais classes são usadas para atribuir semântica às trajetórias.

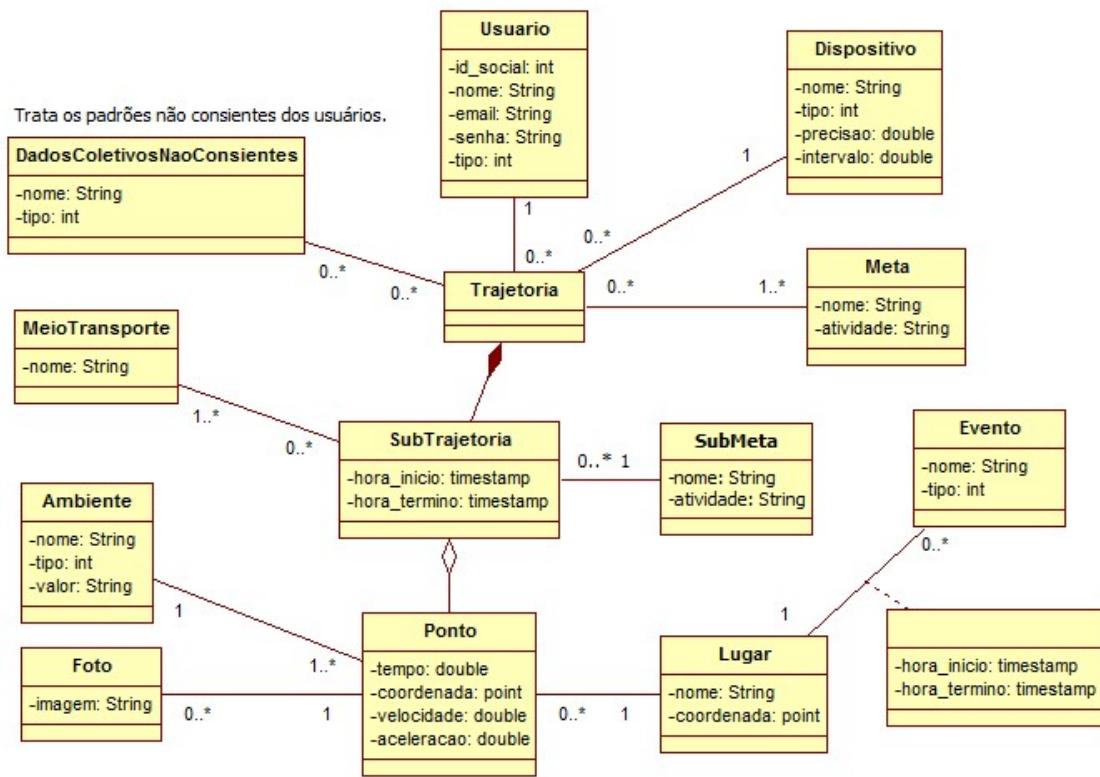


Figura 1 - Modelo de dados para trajetórias móveis
(Fonte: adaptado de BOGORNY et al., 2014)

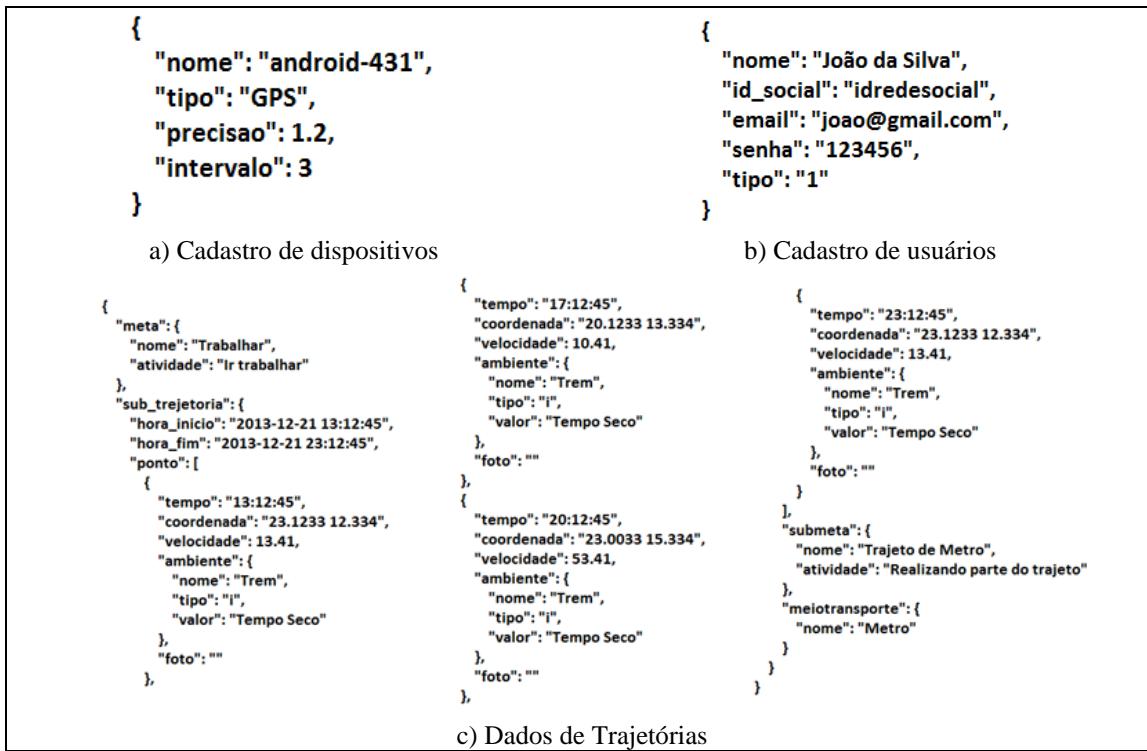


Figura 2 – Modelos dos arquivos JSON

Web services são serviços que visam facilitar o processamento distribuído em sistemas heterogêneos. Estes serviços são baseados em um conjunto de padrões da Internet definidos pelo W3C (RECKZIEGEL, 2013). O *web service* criado utiliza, basicamente, as seguintes tecnologias: PHP, JSON - como formato de interação entre o dispositivo móvel e o serviço *web*; *PostgreSQL* e *PostGIS* - para manipular dados geográficos. Além dessas tecnologias, são utilizadas outras técnicas e conceitos relacionados ao assunto, entre elas: SQL e POO.

Como citado, usou-se JSON para fazer a integração entre o dispositivo móvel e o *web service*. Para tanto, tem-se três tipos de arquivos com dados JSON que são enviados pelo dispositivo: *Cadastro de usuários*, *Cadastro de dispositivos* e *Trajetórias*. A Figura 2 apresenta fragmentos desses arquivos. Como o aplicativo móvel do projeto ainda não está concluído, os testes de acesso ao *web service* foram realizados através de um formulário HTML. Neste formulário pode-se escolher qual dos arquivos JSON se deseja testar, preenchendo os dados necessários. Então é gerado um arquivo JSON que é enviado ao *web service*, simulando o envio que o aplicativo móvel faz.

4. Considerações finais

Web service é uma tecnologia bastante útil, permitindo a integração entre aplicações de diferentes tecnologias (como é o caso dos dispositivos móveis). Neste projeto propõe-se uma base de dados e um *web service* que se comunica com os dispositivos móveis para coletar e armazenar as informações de trajetórias. Para tanto, criou-se o modelo de dados para o armazenamento das trajetórias capturadas por dispositivos móveis. Após essa fase, criaram-se os três modelos de arquivos JSON utilizados para a integração com o dispositivo móvel: Cadastro do usuário; Dados do aplicativo; Dados de trajetórias. Por fim, o *web service* foi desenvolvido e testado, viabilizando que a base de dados *PostgreSQL* receba as informações do dispositivo móvel que foram enviadas via JSON. Um aplicativo móvel próprio para o projeto está em fase de desenvolvimento. As próximas etapas envolvem a coleta de trajetórias reais e o desenvolvimento de algoritmos para mineração sobre essas trajetórias, no domínio do Turismo, além da disponibilização dos códigos fonte em repositórios públicos.

Referências bibliográficas

- BOGORNY, V. *et al.* CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. *Transactions in GIS*. v.18. n.1. fev. 2014. p. 66-68.
- MEIRELES, G. C. **Armazenamento de trajetórias de objetos móveis no domínio do Turismo**. 2013. 50f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) IF Catarinense – Câmpus Camboriú, Camboriú.
- RECKZIEGEL, M. Entendendo os WebServices. **iMasters**. jun. 2006.
- SEEK. **Semantic Enrichment of trajectory Knowledge Discovery**. Página oficial do projeto. Disponível em: <<http://www.seek-project.eu>>. Acessado em: 03 mai. 2012.
- SPACCAPIETRA, S. *et al.* A conceptual view on trajectories. **Data & Knowledge Engineering**, 2008. 65(1), 126–146.
- TORRES, G. M. **Análise comportamental de objetos móveis baseada em dados de trajetórias**. 2009. 55f. Trabalho de Graduação (Bacharelado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre.

Uma Extensão MDA para Geração Automática de Codificação SFS para Banco de Dados Geográficos

João Victor Guinelli¹, André de Souza Rosa¹, Carlos Eduardo Pantoja¹

¹CEFET/RJ - Campus Nova Friburgo

Av. Gov. Roberto da Silveira, 1900 – Prado – 22.635-000 – Nova Friburgo – RJ – Brasil

jvguinelli@gmail.com, andre_souza.rosa@hotmail.com, pantoja@cefet-rj.br

Abstract. This paper proposes a geographic extension for a relational database modeling tool using the Model-Driven Architecture (MDA), which should be able to generate automatic code for Special Features Specification (SFS) of Open Geospatial Consortium (OGC). The extension was defined by the creation of a new cartridge to generate artifacts based on specification Model-To-Text and by revision of the generic meta-model, to make it adaptable to conceptual constructions of geographical database.

Resumo. Este artigo propõe uma extensão geográfica para uma ferramenta de modelagem de bancos de dados relacionais utilizando o Model-Driven Architecture (MDA), para ser capaz de gerar codificação automática para a Special Features Specification (SFS) da Open Geospatial Consortium (OGC). A extensão foi definida através da criação de um novo cartucho de geração de artefatos de texto baseado na especificação Model-To-Text e na revisão do meta-modelo genérico para ser adaptável às construções conceituais de banco de dados geográficos.

1. Introdução

A modelagem conceitual para banco de dados é uma abstração de um determinada realidade transcrita em um modelo de conceito apoiada por modelos gráficos que incluem detalhes do projeto de banco de dados em um nível independente de plataforma [ELMASRI and NAVATHE 2005]. Atualmente, a área de banco de dados está utilizando dados não convencionais, e.g. dados espaciais e espaço-temporais, para aplicação de banco de dados geográficos [LAENDER et al. 2005].

Existem diversos modelos para modelagem de banco de dados geográficos, como o OMT-G e o UML-GeoFrame, que utilizam estereótipos específicos para modelagem geográfica e diferem da modelagem relacional tradicional. Existem também algumas especificações, que propõem um esquema padrão para o armazenamento, leitura, consulta e atualização desses dados geográficos através do SQL, como o *Special Feature Specification (SFS) da Open Geospatial Consortium (OGC)*.

A *Model-Driven Architecture* (MDA) é uma abordagem de desenvolvimento dirigida por modelos, que os utiliza em diversos níveis de abstração, partindo de um modelo independente de plataforma, que será transformado em um modelo específico de plataforma, dado uma tecnologia específica, até sua efetiva implementação [MELLOR et al. 2005]. Algumas ferramentas utilizam a MDA para a modelagem de banco de dados geográficos como o ArgoCASEGEO + TerraLib e o OMT-G Design.

Porém, tais ferramentas estão atreladas a modelos geográficos específicos e não se utilizam de um meta-modelo genérico para banco de dados.

Existe também uma outra ferramenta MDA [ROSA et al. 2013] que utiliza um meta-modelo genérico para as linguagens de modelagens relacionais que gera codificação automática de DDL para os padrões ANSI/SQL 93/99/03 [ROSA and PANTOJA 2013]. Contudo, tal ferramenta não gera codificação automática para especificações geográficas nem utiliza nenhum modelo geográfico para modelagem.

Portanto, o objetivo deste artigo é propor uma ferramenta MDA, que seja capaz de gerar codificação automática na especificação SFS/SQL da OGC, a partir de modelos geográficos, através da extensão: i) nas regras de transformação; ii) do meta-modelo para garantir as estruturas geográficas; e iii) propor a adaptação da metodologia para modelos conceituais geográficos. O artigo está estruturado da seguinte forma: na seção 2 será apresentada a extensão da ferramenta; na seção 3 será apresentado um rápido exemplo; e por fim, na seção 4 a conclusão.

2. A Extensão da Ferramenta MDA

A extensão da ferramenta consiste em um conjunto de regras *Model-To-Text* para geração de codificação automática SFS/SQL a partir da inserção de novos tipos geográficos no meta-modelo genérico da metodologia. A ferramenta foi desenvolvida para o ambiente Eclipse e utiliza a MDA para permitir tanto a modelagem conceitual de banco de dados relacionais quanto a geográfica.

A M2T [OMG 2008] é uma especificação de geração de artefatos de textos a partir de instâncias de modelos através de regras de transformações. A metodologia [ROSA et al. 2013] é composta de um meta-modelo genérico que permite a adição de novos cartuchos de geração de texto observando-se os conceitos existentes no metamodelo e um conjunto de regras de transformação para geração automática de código ANSI SQL 92/99/03. Dessa forma é proposto a adição de um conjunto novo de regras para a geração de atributos de tipos geográficos. Para se implementar essa alteração, foi necessária a criação de uma classe *Type* associada a classe *Field*, além de um *enumeration* com todos os tipos geográficos encontrados na especificação SFS. A metodologia utilizada pela ferramenta pode ser vista na figura 1.

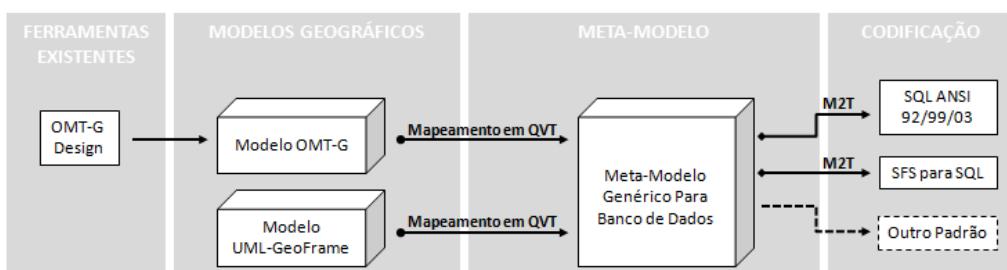


Figure 1. A Metodologia Estendida.

A metodologia também permite a adição de ferramentas já existentes ao seu meta-modelo a partir de um conjunto de transformações usando a linguagem de transformação *Query-View-Transformation*. A linguagem permite um mapeamento de conceitos entre

o modelo específico geográfico e o meta-modelo genérico. Uma vez que a ferramenta é construída baseada na estrutura de um modelo e esses conceitos sejam aderentes aos conceitos do meta-modelo adotado pela metodologia, a codificação poderá ser realizada independentemente de atrelamento modelagem-especificação.

3. Resultados Obtidos

Nesta seção são apresentados a execução de um simples exemplo na ferramenta proposta. A modelagem conceitual geográfica (figura 2) utilizada como exemplo é parte de uma base de dados de municípios e logradouros. A modelagem foi realizada utilizando o modelo OMT-G.

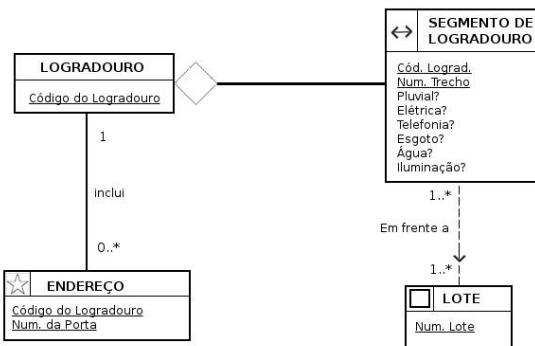


Figure 2. Exemplo de modelagem conceitual geográfica.

O primeiro passo da ferramenta é instanciar o modelo conceitual observando cada estrutura pertencente ao meta-modelo. O Ecore é um aplicativo pertencente ao *Eclipse Modeling Framework (EMF)* [STEINBERG et al. 2008] e responsável pela criação de meta-modelos. A instância do modelo pode ser vista na figura 3.



Figure 3. A instância do modelo..

Em seguida, é necessário executar as transformações *Model-To-Text* para que seja possível gerar a codificação SFS/SQL. As transformações são executadas em cascata e o esquema do banco de dados gerado pode ser visto na figura 4.

4. Conclusão

O artigo apresentou uma extensão geográfica de uma ferramenta MDA de modelagem conceitual de banco de dados relacionais para a geração de codificação automática

```

CREATE DATABASE BDGeo;
CREATE TABLE BDGeo.Logradouro (
    id_logradouro int(0) ,
    CONSTRAINT id_logradouro PRIMARY KEY (id_logradouro)
);
CREATE TABLE BDGeo.Lote (
    id_lote int(0) ,
    campo_espacial POLYGON,
    CONSTRAINT id_lote PRIMARY KEY (id_lote)
);
CREATE TABLE BDGeo.Endereco (
    id_endereco int(0) ,
    campo_espacial POINT,
    id_logradouro int(0) ,
    CONSTRAINT id_endereco PRIMARY KEY (id_endereco),
    CONSTRAINT FK_id_logradouro FOREIGN KEY (id_logradouro)
        REFERENCES Logradouro (id_logradouro)
);
CREATE TABLE BDGeo.SegmentoDeLogradouro (
    id_segmento_de_logradouro int,
    campo_espacial LINESTRING,
    id_logradouro int(0) ,
    CONSTRAINT id_segmento_de_logradouro PRIMARY KEY (id_logradouro),
    CONSTRAINT FK_id_logradouro2 FOREIGN KEY (id_logradouro)
        REFERENCES Logradouro (id_logradouro)
);
CREATE TABLE BDGeo.Lot_SDL (
    id_lot_sdl int(0) ,
    id_lote int(0) ,
    id_segmento_de_logradouro int(0) ,
    CONSTRAINT id_lot_sdl PRIMARY KEY (id_lot_sdl),
    CONSTRAINT FK_id_lote FOREIGN KEY (id_lote) REFERENCES Lote (id_lote),
    CONSTRAINT FK_id_segmento_de_logradouro FOREIGN KEY (id_segmento_de_logradouro)
        REFERENCES SegmentoDeLogradouro (id_segmento_de_logradouro)
);

```

Figure 4. O SQL gerado.

baseada na especificação SFS/SQL. O artigo também apresentou uma extensão para o meta-modelo genérico inicial e propôs um conjunto de mapeamentos para integração com ferramentas já existentes que utilizam um determinado modelo conceitual geográfico.

A utilização da MDA permite a adição de novos cartuchos de geração de código sem a necessidade de se modificar seus modelos anteriores gerando uma flexibilização da ferramenta por não gerar um atrelamento modelo-código e permitindo ao projetista de banco de dados a escolha entre diversas linguagens de modelagens. A ferramenta utiliza um meta-modelo genérico que serve tanto para modelagem conceitual quanto para modelagem geográfica. A MDA garante também a adição de novas ferramentas através do mapeamento entre o modelo específico de plataforma e o modelo independente de plataforma, sem o descarte ou a necessidade de se refazer as regras para a geração da codificação.

References

- ELMASRI, R. and NAVATHE, S. (2005). *Sistemas de banco de dados*. Pearson Addison Wesley.
- LAENDER, A., DAVIS, C., BRAUNER, D., CÂMARA, G., QUEIROZ, G., BORGES, K., FERREIRA, K., LIGIANE, V. L., and CARVALHO, M. (2005). *Bancos de Dados Geográficos*. MundoGEO.
- MELLOR, S. J., SCOTT, K., UHL, A., and WEISE, D. (2005). *MDA Destilada: Princípios de Arquitetura Orientada por Modelos*. Ciência Moderna.
- OMG (2008). Mofmodel to text transformation language (mofm2t). In <http://www.omg.org/spec/MOFM2T/1.0>, number 1.
- ROSA, A., GONÇALVES, I., and PANTOJA, C. E. (2013). A mda approach for database modeling. In *Lecture Notes on Software Engineering*, volume 1, pages 26–30.
- ROSA, A. and PANTOJA, C. E. (2013). Uma ferramenta mda para modelagem de banco de dados relacionais. In *IX Escola Regional de Banco de Dados*.
- STEINBERG, D., BUDINSKY, F., MERKS, E., and PATERNOSTRO, M. (2008). *Emf: Eclipse Modeling Framework*. Pearson Education.