S2I+: Armazenamento Eficiente em Índice Espaço-Textual

Tiago F. Athayde-Novaes[1], **Fellipe L. Fonseca**[1], **João B. Rocha-Junior**[1]

[1]Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana – BA – Brasil

tiagoathayde@hotmail.com, fellipefonseca9@gmail.com, joao@uefs.br

Abstract. *With the popularization of Smartphones with GPS, there is a huge amount of data with spatial and textual information being produced and shared. Hybrid indexes that combine spatial and textual information are being proposed to process queries efficiently. One of the most efficient indexes is S2I (Spatial Inverted Index). However, S2I has a major drawback in terms of index size. In this paper, we study new strategies to reduce the size of S2I, maintaining the good performance aspects.*

Resumo. *Com a popularização dos Smartphones com GPS, há uma grande quantidade de dados com informações espaciais e textuais sendo produzidos e compartilhados. Para realizar consultas de forma eficiente sobre esses dados, faz-se necessário utilizar índices híbridos que combinam índices espaciais e textuais. Um dos índices com melhor desempenho é o S2I (Spatial Inverted Index). Infelizmente, o S2I ocupa muito espaço para armazenamento. Este trabalho propõe novas estratégias para reduzir o tamanho do S2I, mantendo os bons aspectos em termos de desempenho.*

1. Introdução

É comum encontrar bases de dados com objetos que possuem informação espacial (latitude e longitude) e textual. O OpenStreetMap[1] é um exemplo de aplicação que coleta e compartilha informações sobre objetos como hotéis e restaurantes. Os objetos possuem informação espacial e textual e são chamados de objetos espaço-textuais.

Estas bases de dados, contendo objetos espaço-textuais, podem ser bastante volumosas. O OpenStreetMap possui mais de 3 bilhões de objetos espaço-textuais[2]. Portanto, extrair informações relevantes destas bases de dados de forma eficiente é um desafio. Isto justifica o interesse por novas consultas e técnicas capazes de selecionar objetos espaço-textuais de grandes bases de dados [Chen et al. 2013, Cong et al. 2009, De Felipe et al. 2008, Li et al. 2011, Rocha-Junior et al. 2011, Zhou et al. 2005].

Uma das principais consultas propostas é a Consulta Espaço-Textual Top-k [Cong et al. 2009, Rocha-Junior et al. 2011]. Esta consulta recebe como parâmetro um local de interesse, um conjunto de palavras-chave e a quantidade de objetos desejados, retornando os k melhores objetos espaço-textuais. Para cada objeto espaço-textual é computado um escore, levando em consideração a distância entre o local da consulta e a localização do objeto; e a relevância textual do objeto para as palavras-chave de busca.

[1]http://www.openstreetmap.org
[2]http://wiki.openstreetmap.org/wiki/Stats

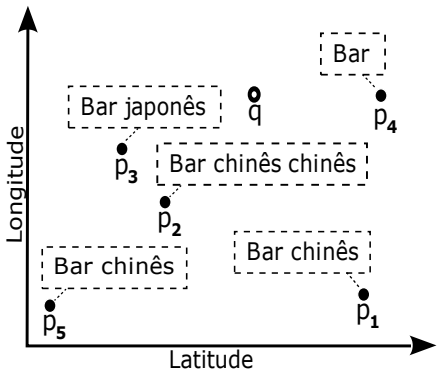

Figura 1. Região espacial com objetos espaço-textuais p.

A Figura 1 apresenta uma área espacial com objetos espaço-textuais p_i e um usuário q. Assumindo que o usuário realize uma consulta por "bar japonês" e $k = 3$, esta consulta retorna p_3, p_4 e p_2. O objeto p_3 é o mais relevante textualmente para a consulta e está próximo de q. Os objetos p_4 e p_2 são relevantes textualmente porque possuem um dos termos da consulta "bar" e estão mais próximos de q que os demais.

Índices híbridos vem sendo utilizados para processar a Consulta Espaço-Textuais Top-k de forma eficiente. Estes índices combinam características de índices espaciais como R-tree [Guttman 1984] e índices textuais como Arquivo Invertido [Zobel and Moffat 2006]. Chen *et al.* [Chen et al. 2013] avaliou 12 (doze) índices propostos para melhorar o desempenho de Consultas Espaço-textuais. Entre estas estruturas, o índice S2I foi um dos que teve melhor desempenho. Entretanto, entre os índices avaliados, o S2I foi o que ocupou mais espaço para armazenamento em disco.

Este trabalho propõe novas formas de armazenar os dados no S2I com o intuito de reduzir o espaço de armazenamento em disco. As principais contribuições são: 1) estudar a distribuição dos dados armazenados no S2I; 2) propor novas técnicas para reduzir o tamanho do índice e 3) avaliar as técnicas propostas utilizando bases de dados reais.

O restante deste artigo é organizado da seguinte forma: Na Seção 2 os trabalhos relacionados são apresentados. Na Seção 3 é apresentado um estudo sobre a distribuição dos dados armazenados no S2I. A Seção 4 contém novas propostas para melhorar o armazenamento de dados no S2I. Na Seção 5, as técnicas propostas são avaliadas utilizando bases de dados reais. A Seção 6 contém as considerações finais.

2. Trabalhos Relacionados

Esta seção está dividida em duas partes: na primeira parte (Seção 2.1), os principais índices para processamento de consultas Espaço-Textuais são apresentados; na segunda parte (Seção 2.2), o índice S2I é apresentado em maiores detalhes.

2.1. Índices para processamento de consultas espaço-textuais

Zhou *et al.* [Zhou et al. 2005] realiza um estudo que avalia o desempenho de índices tradicionais como Arquivo Invertido [Zobel and Moffat 2006] e a R*-tree [Beckmann et al. 1990] para o processamento de consultas espaço-textuais. Zhou *et al.* conclui que os índices híbridos têm vantagem sobre a execução sequencial (um índice

após o outro). Entretanto, o índice híbrido proposto por Zhou *et al.* é restrito a consultas textuais Booleanas, não podendo ser utilizado para processar a Consulta Espaço-Textual Top-k, que requer que o escore textual do objetos seja computado.

Os primeiros índices desenvolvidos para a Consulta Espaço-Textual Top-k foram propostos por Cong *et al.* [Cong et al. 2009] e Li *et al.*[Li et al. 2011]. As duas estruturas tem o mesmo nome IR-Tree e ambas agregam arquivos invertidos aos nós da árvore R-tree com as informações textuais dos objetos contidos nas suas subárvores.

Posteriormente, Rocha-Junior *et al.* [Rocha-Junior et al. 2011] propôs o índice S2I (*Spatial Inverted Index*) que substitui as listas invertidas de um Arquivo Invertido tradicional por índices multidimensionais, permitindo computar a Consulta Espaço-Textual Top-k de forma eficiente. Como objetivo deste trabalho é melhorar a estrutura de armazenamento do S2I, este índice é apresentado em maiores detalhes na próxima seção.

2.2. Spatial Inverted Index

O S2I (*Spatial Inverted Index*) é semelhante a um Arquivo Invertido. Para cada termo da coleção, o S2I mantém o conjunto de objetos espaço-textuais relevantes para o termo. Entretanto, ao invés de armazenar esses objetos em uma lista, o S2I utiliza duas formas de armazenamento: bloco ou índice multidimensional (aR-tree [Papadias et al. 2001]). A escolha da forma de armazenamento dos objetos depende da frequência do termo na coleção. Os objetos de um termo frequente são armazenados em árvores aR-tree, enquanto que os objetos de um termo infrequente são armazenados em blocos.

Cada bloco tem um tamanho fixo, definido no momento da construção do índice, e é capaz de armazenar um número máximo de entradas, delimitado pelo tamanho de uma página do disco. As árvores armazenam os objetos dos termos frequentes e portanto podem armazenar qualquer número de objetos. A árvore utilizada pelo S2I é uma aR-tree [Papadias et al. 2001] que possui a mesma estrutura de uma R-tree tradicional. Entretanto, além do retângulo multidimensional que engloba os objetos na subárvore, aR-Tree armazena, em cada nó, um atributo não espacial que representa os objetos armazenados na subávore. No S2I, este atributo armazena o maior escore textual (impacto do termo) entre os objetos espaço-textuais presentes na subárvore, permitindo evitar o acesso a nós da árvore que não tenham objetos capazes de entrar no Top-k [Rocha-Junior et al. 2011].

O S2I tem bom desempenho para o processamento de consultas Espaço-Textuais Top-k [Chen et al. 2013, Rocha-Junior et al. 2011]. Consultas com apenas uma palavra-chave são processadas, acessando somente uma árvore quando a palavra-chave (termo) for frequente ou um bloco quando o termo for infrequente. A consulta com mais de uma palavra-chave é processada agregando os objetos dos blocos ou das árvores que são recuperados em ordem decrescente de relevância textual.

3. Estudo sobre a distribuição dos termos

O S2I utiliza uma heurística simples para decidir se os objetos devem ser armazenados em blocos ou árvores. Inicialmente, todos os objetos são armazenados em blocos. Cada bloco tem o tamanho de uma página do disco. A medida que o número de objetos aumenta e superam a capacidade máxima de armazenamento do bloco, eles passam a ser armazenados em árvores. Ficando armazenados em blocos apenas os objetos dos termos infrequentes [Rocha-Junior et al. 2011].

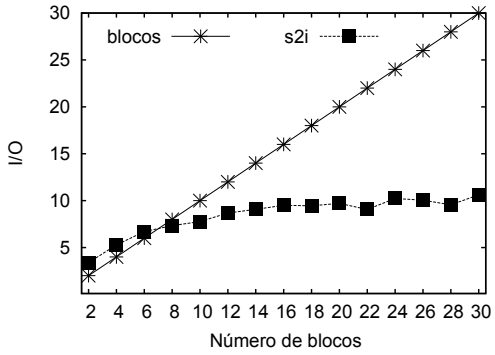

Figura 2. Utilizando múltiplos blocos para armazenar os objetos de um termo

Para identificar o tamanho ideal do número de objetos que devem ficar em blocos ou em árvores, realizou-se um experimento que consiste em aumentar o número de blocos para armazenar os termos infrequentes. Ao invés de armazenar os termos infrequentes em apenas um bloco, permite-se armazenar os termos infrequentes em mais de um bloco quando necessário. O experimento utilizou uma base de dados extraída do OpenStreetMap com 300.891 objetos espaço-textuais. Assumiu-se blocos com 4KB de tamanho, o mesmo tamanho definido para um nó da árvore.

A Figura 2 apresenta o I/O (número de páginas acessadas) para processar uma consulta Espaço-Textual Top-k com apenas um termo, quando os objetos são armazenados em blocos ou árvores. Até 6 blocos, é preferível armazenar os objetos em blocos ao invés de árvores. Entre 7 e 8 blocos, o desempenho das estruturas são similares. A partir de 8 blocos a árvore passa a ter um melhor desempenho.

4. Propostas de Armazenamento para a estrutura de blocos

Nesta seção, duas novas propostas para o armazenamento de objetos em blocos são apresentadas: armazenando objetos de mais de um termo por bloco (Seção 4.1) e o armazenamento sequencial, com blocos de tamanho variável (Seção 4.2).

4.1. Mais de um objeto por bloco

Este método de armazenamento visa permitir que objetos de termos infrequentes ocupem o mesmo bloco, reduzindo espaço vazio. Os objetos espaço-textuais de mais de um termo ficam armazenados em um mesmo bloco para evitar que o índice reserve espaço em disco para o bloco e não utilize.

Para implementar este método é necessário que cada bloco armazene uma estrutura com um cabeçalho, permitindo identificar a qual termo o objeto pertence. O cabeçalho armazena o identificador do termo e a quantidade de objetos associados ao termo. Sendo assim, é possível identificar dentro de um mesmo bloco, quais objetos são referentes a qual termo infrequente.

Além disso, este método requer uma busca sequencial nos objetos armazenados em um mesmo bloco para encontrar os que são relevantes para um termo. Entretanto está busca é rápida porque os dados estão em memória no momento da consulta, visto que o bloco é recuperado com um acesso a disco.

4.2. Armazenamento Sequencial

Este método armazena os objetos de forma sequencial, onde cada bloco tem o tamanho ideal para armazenar os objetos associados a um termo. Assim, os blocos passam a ter tamanho variável, podendo ocupar mais de uma página em disco ou apenas poucos bytes.

Cada termo do vocabulário aponta para uma posição no arquivo onde inicia o bloco que armazena os objetos que contém o termo. O cabeçalho do bloco contém o identificador do termo e a quantidade de objetos armazenados no bloco.

Esta estrutura é excelente em termos de armazenamento, porque preenche todo o espaço vazio. Sendo ideal para bases de dados fixas que não sofrem muita alteração. No entanto, para bases de dados que recebem novos objetos frequentemente, o tempo para reorganizar os blocos dentro do arquivo pode ser muito elevado.

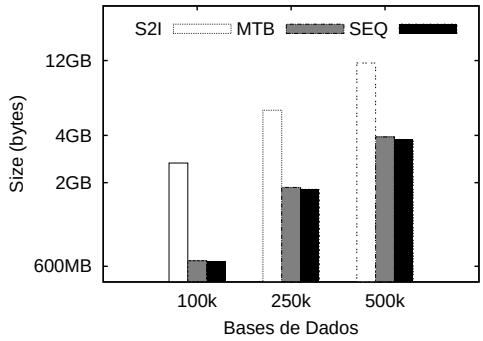

Figura 3. Tamanho dos índices

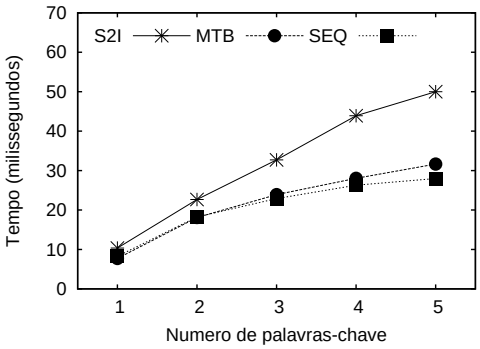

Figura 4. Tempo de resposta

5. Experimentos

Esta seção contém um estudo comparativo das abordagens propostas, avaliando o tamanho do índice e o tempo de resposta das consultas. Os algoritmos foram implementados em Java, utilizado uma máquina com processador Intel Core i5, 3.4GHz e 4GB de memória RAM. Nos experimentos, foram armazenados em árvores os objetos dos termos frequentes que ocuparam mais de 6 blocos (Seção 3).

Para cada experimento foram realizadas 800 consultas com termos extraídos do vocabulário, localização espacial aleatória e valor de $k = 10$. As bases de dados utilizadas foram compostas por textos do Wikipédia e localização espacial extraída de rodovias americanas[3]. Foram utilizadas 3 bases de dados, contendo 100, 250 e 500 mil objetos espaço-textuais (o valor padrão para $\alpha = 0,5$ e 250k para a base [Rocha-Junior et al. 2011]).

5.1. Tamanho e construção do índice

A Figura 3 contém o tamanho dos índices para diferentes bases de dados. Pode-se observar que o S2I tradicional ocupa muito mais espaço em disco que as duas soluções propostas: MTB (Múltiplos Termos por Bloco) e SEQ (Sequencial). O tamanho ocupado por MTB e SEQ são bastantes similares com uma leve vantagem para a solução que armazena os dados sequencialmente. Observe que esta figura possui uma divisão no eixo y para melhor acomodar o tamanho dos índices.

[3]http://www.dis.uniroma1.it/challenge9/download.shtml

Quanto ao tempo de construção do índice, o S2I apresentou o melhor desempenho, seguido pelo SEQ e pelo MTB (mais lento). Para a base de 500k, o tempo de construção do S2I foi de 1h42m, enquanto que o tempo do MTB foi de 4hs. Isso se deve a reorganização do índice MTB para agrupar objetos de termos diferentes em um bloco.

5.2. Desempenho

A Figura 4 contém o tempo de resposta em milissegundos, ao variar o número de palavras-chave da consulta. O tempo de resposta é similar, mas com leve vantagem para o armazenamento SEQ. O que pode explicar o resultado ruim alcançado pelo S2I é o tamanho do índice. Como o índice S2I é muito grande, ele não se beneficia da cache do Sistema Operacional como os demais.

O mesmo experimento foi realizado variando o número de objetos desejados (k), e o resultado foi similar.

6. Conclusão

Este artigo explorou um problema existente no S2I que é o tamanho do índice. Para solucionar este problema, foi realizado um estudo sobre a forma como os dados são armazenados em blocos e árvores no S2I, verificando que vale a pena utilizar árvore apenas acima de um determinado número de blocos (aproximadamente 6).

Após esse estudo, duas novas formas de armazenar os dados foram propostas: armazenar vários objetos em um bloco e armazenar blocos sequencialmente. Os dois métodos propostos apresentaram uma melhora significativa em relação ao tamanho do índice além de obterem melhores tempos de resposta para as consultas realizadas.

Referências

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The r*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*, pages 322–331.

Chen, L., Cong, G., Jensen, C. S., and Wu, D. (2013). Spatial keyword query processing: An experimental evaluation. *Proceedings of the VLDB*, pages 217–228.

Cong, G., Jensen, C. S., and Wu, D. (2009). Efficient retrieval of the top-k most relevant spatial web objects. In *VLDB*, pages 337–348.

De Felipe, I., Hristidis, V., and Rishe, N. (2008). Keyword search on spatial databases. In *ICDE*, pages 656–665.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57.

Li, Z., Lee, K., Zheng, B., Lee, W.-C., Lee, D. L., and Wang, X. (2011). Ir-tree: An efficient index for geographic document search. *Proceedings of TKDE*, pages 585–599.

Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. (2001). Efficient olap operations in spatial data warehouses. In *SSTD*, pages 443–459.

Rocha-Junior, J. a. B., Gkorgkas, O., Jonassen, S., and Nørvåg, K. (2011). Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222.

Zhou, Y., Xie, X., Wang, C., Gong, Y., and Ma, W.-Y. (2005). Hybrid index structures for location-based web search. In *CIKM*, pages 155–162.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *Proceedings of ACM CSUR*, 38(2).