VIII Simpósio de Iniciação Científica 2025

"Do conhecimento acadêmico à transformação sustentável: inovação com validação científica"

ARMAZENAMENTO MASSIVO DE DADOS SOBRE ÍNDICES MULTIDIMENSIONAIS

Gabriel Barbosa Fernandes[1] (IC), Luiz Olmes Carvalho (PQ)[1]

[1]*Universidade Federal de Itajubá.*

Palavras-chave: Armazenamento. Dados espaciais. Indexação. R-Tree.

Introdução

Os Sistemas de Gerenciamento de Bases de Dados Relacionais (SBGDR) são bem equipados para o armazenamento e manipulação de dados escalares: números, datas e pequenas cadeias de caracteres. Entretanto, de acordo com a evolução das aplicações que fazem uso de SGBDR, outros tipos de dados também necessitam de suporte. Entre eles, destacam-se os dados *espaciais*, ou *multidimensionais* (Fevgas et al., 2020).

Os dados espaciais têm se tornado cada vez mais relevantes em aplicações modernas, como sistemas de informação geográfica (SIG), geolocalização, logística e análise (Manolopoulos et al., 2005). Diferentemente de dados escalares, eles possuem dimensões espaciais, o que exige técnicas e estruturas específicas para seu armazenamento e processamento. O desafio central no tratamento de dados espaciais está na eficiência de consultas, além de como o dado é processado e armazenado. Estruturas tradicionais de indexação, como a B^+-Tree (Comer, 1979), não são adequadas para lidar com dados multidimensionais, resultando em operações custosas em termos de tempo e processamento.

Para enfrentar esses desafios, a família de estruturas da R-Tree (Manolopoulos et al., 2005) contém várias estruturas de índices balanceadas e projetadas para armazenar dados espaciais. De maneira similar à B^+-Tree, uma R-Tree organiza os dados espaciais em nós índices e folhas. Os objetos armazenados na R-Tree são inseridos sempre no nível das folhas e sua geometria é representada na forma de seu Retângulo Delimitador Mínimo (MBR). Por sua vez, os nós índices consistem de MBRs que agrupam os objetos, de forma hierárquica, como apresentado na Figura 1 (Broilo et al., 2010). Ainda na Figura 1(a), os retângulos em vermelho representam os MBR dos objetos espaciais (que são salvos em folhas), que são hierarquicamente agrupados pelos retângulos azuis e pretos (em índices). Na Figura 1(b), pode-se observar a representação equivalente da R-Tree.

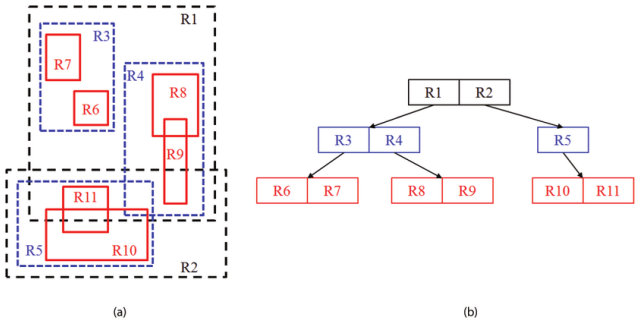

Figura 1: Representação espacial dos dados e estrutura hierárquica da R-Tree.

Neste contexto, este trabalho de Iniciação Científica tem por objetivo realizar a implementação de uma variante da estrutura de indexação R-Tree e avaliar o seu desempenho em relação às operações de inserção e consulta de dados.

Os experimentos realizados consideraram um conjunto de dados reais, formado por dados de latitude e longitude das cidades brasileiras. Os resultados obtidos mostram que a estrutura se comporta melhor com tamanhos de blocos de discos menores (2 KB), e tempo médio de 0,0000622 segundos para inserção e consulta de dados.

Metodologia

Após a realização de um estudo da literatura sobre dados espaciais e da R-Tree, a implementação da estrutura de indexação espacial foi realizada tendo como base o *framework Object Injection* (Carvalho et al., 2013). O *framework* é dividido em vários módulos, onde cada um dos módulos é responsável por um aspecto distinto da implementação de estruturas de dados. Os principais módulos utilizados, bem como e as definições realizadas sobre eles, são:

- *Metaclass*: onde são definidos os objetos que devem ser persistidos. No caso do presente trabalho, foram definidos pontos *n*-dimensionais e (hiper)retângulos (MBR). Os (hiper)retângulos

são definidos a partir de uma coordenada de origem (ponto *n*-dimensional) e a extensão ao longo de cada eixo Cartesiano.
- *Block*: módulo onde são definidos páginas e blocos de disco. A definição dos nós folha e índices da estrutura de indexação espacial foram realizadas neste módulo.
- *Storage*: módulo onde as estruturas são implementadas. Onde é realizada a definição da estrutura espacial, suas políticas de divisão de nós (*split*) e promoção de objetos (*promote*), e as implementações dos métodos básicos de inserção e consultas. O índice espacial desenvolvido, semelhante à R-Tree, diferencia-se por conter ligações (ponteiros) duplas para varredura sequencial entre todos os seus nós folhas e todos os nós de índices de um mesmo nível. Neste caso, a árvore torna-se cônica, ao invés de planar, com implicações que serão alvo de futura pesquisa.
- *Queries*: módulo responsável pelo cômputo de estatísticas associadas ao uso das estruturas de dados. No caso, os tempos médios de inserção e consulta são obtidos a partir deste módulo. Outras estatísticas, como a quantidade de verificações de sobreposições de MBR no índice espacial também são coletadas.

Os principais destaques do módulo Block são as modelagens dos nós folha e índice da estrutura de dados. Todo nó é representado na forma de um *array* de *bytes* de tamanho *N*, sendo o valor de *N* uma potência de 2. Um nó folha contém as seguintes informações:

- *Tipo do nó*: valor inteiro usado para identificar univocamente o nó folha entre nós de demais estruturas de dados.
- *Ligações para outros nós*: três inteiros longos que representam o identificador do bloco de disco do nó "pai" ao nó atual (no nível superior), o nó antecessor a ele no mesmo nível hierárquico e o nó sucessor a ele no mesmo nível hierárquico.
- *Número de chaves*: valor inteiro *K*, que indica quantos objetos estão armazenados no nó folha.
- *K* identificadores únicos universais (UUID): um UUID para cada objeto armazenado na folha. Estes UUID servem de *chave* para os objetos espaciais armazenados na estrutura.
- *Área de espaço livre no nó*: parte ainda não usada da folha. Se houver espaço suficiente, novos objetos podem ser inseridos naquela folha.
- *K chaves*: contém o dado espacial, isto é, a serialização do objeto inserido no índice.

Já os nós do tipo índice armazenam as seguintes informações:

- *Tipo do nó*, *Ligações para outros nós* e *Número de chaves*: todos esses campos são idênticos aos presentes nos nós folha.
- *K ligações para as sub-árvores*: inteiros longos que representam o identificador do bloco de disco do nó que está no nível imediatamente inferior ao nó atual.
- *Área de espaço livre no nó*: como na folha. O espaço pode ser usado para inserir novos objetos conforme divisões de folhas (*splits*) ocorram.
- *K Retângulos Delimitadores Mínimos*: agrupam os MBR dos níveis inferiores.

Em relação aos algoritmos de inserção e consulta, a inserção considera a mesma estratégia da R-Tree original: menor aumento de MBR, seguindo versão quadrática da política de divisão de nós. Embora vários tipos de consultas possam ser realizados sobre dados espaciais, a implementação realizada ficou na consulta pontual, isto é, na recuperação exata do objeto de consulta.

Resultados e discussão

Esta seção experimental tem como objetivo avaliar a implementação realizada em relação às seguintes métricas: *tempo de execução*, *quantidade de verificações de sobreposição* e *quantidade de acessos a blocos de disco*, tanto para a operação de inserção de objetos quanto para consulta pontual.

A estrutura de índice espacial foi implementada sobre o *framework Object Injection*, em linguagem Java. Os experimentos foram realizados em um computador com sistema operacional Windows 11, 8 GB RAM, 256 GB SSD e processador Intel Core i5.

O conjunto de dados utilizado para testes é o GNS[1], que compreende pontos de interesse, com valores de latitude e longitude de todo o globo terrestre, sendo escolhidos, aleatoriamente, um total de 456000 pontos.

[1] GNS: https://geonames.nga.mil/geonames/GNSData/

A metodologia de execução dos experimentos consistiu em inserir todos os pontos na estrutura de índice espacial e, após a sua criação, consultar cada um dos pontos inseridos, armazenando os resultados aferidos.

A fim de avaliar a escalabilidade do índice implementado, a inserção de dados (seguida da consulta) foi realizada para 4 valores distintos do número de elementos: iniciou-se com 40% (182400) do total de pontos do *dataset* e as seguintes foram incrementadas em passos de 20%, até atingir o total (100% = 456000 pontos). Cada uma dessas configurações foi executada para 5 tamanhos distintos de blocos de disco: 1 KB, 2 KB, 4 KB, 8 KB e 16 KB.

A Figura 2 apresenta o tempo médio de inserção de objetos no índice. Pode-se notar que conforme o tamanho do bloco de disco aumenta, mais elementos podem ser inseridos em um mesmo nó. Isso faz com que mais elementos (MBR) de um nó se qualifiquem para receber o novo objeto. Dessa forma, o número de cálculos de sobreposição aumenta, refletindo no aumento do tempo. A Figura 3, que mostra o número médio de verificações de sobreposição, confirma essa afirmação. Ainda na Figura 2, quanto maior o número de elementos na estrutura, maior é o tempo de inserção, devido, também, às operações de divisão de nós da estrutura.

A Figura 4 apresenta a quantidade média de acessos a blocos de disco para a operação de inserção. Pode-se notar que blocos maiores resultam em uma estrutura com menos blocos de disco no total, visto que mais elementos são serializados dentro dos blocos. Isso faz com que a quantidade de acessos diminua, conforme o tamanho do bloco aumenta.

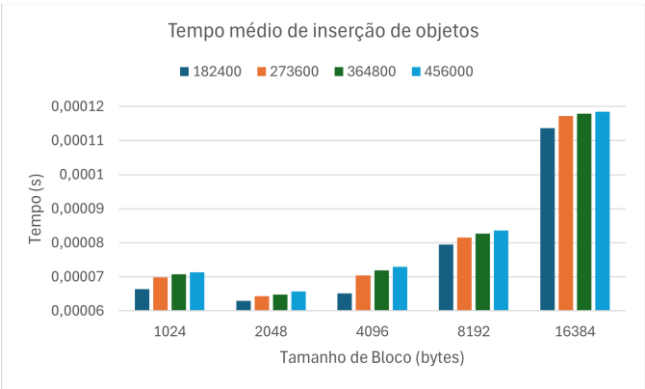

Figura 2: Tempo de inserção.

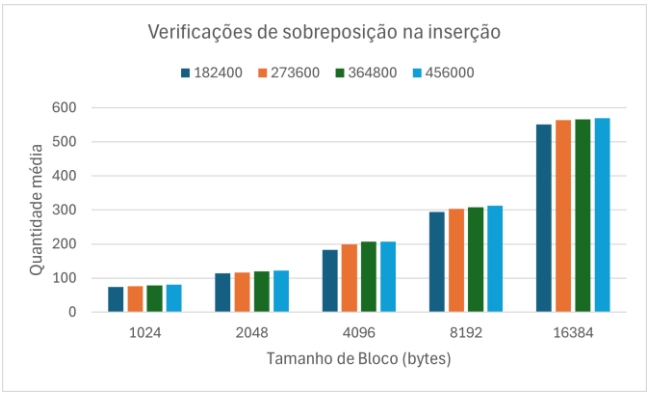

Figura 3: Número de verificações na inserção.

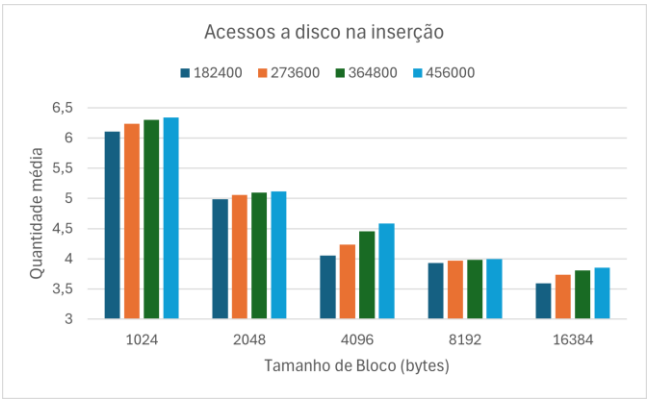

Figura 4: Número de acessos a disco na inserção.

A Figura 5 apresenta o tempo médio de consulta pontual de cada objeto anteriormente inserido na estrutura de dados. Pode-se observar um comportamento mais uniforme para blocos de disco de 1 KB e 2 KB. Já para os demais tamanhos, a quantidade de sobreposição de MBRs no espaço torna-se maior, fazendo com que mais entradas de um nó índice se qualifiquem para serem processadas, visto que o bloco armazena uma quantidade maior de elementos.

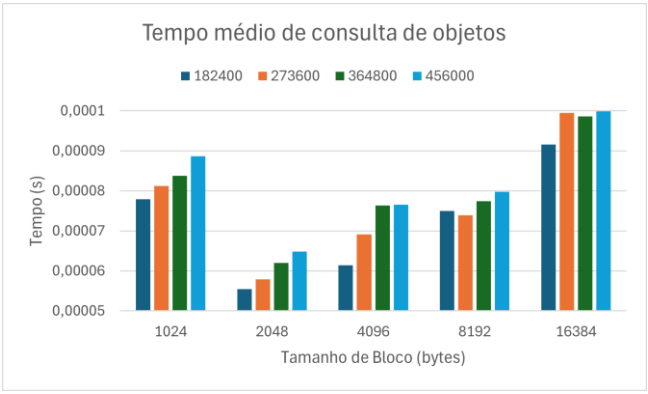

Figura 5: Tempo de consulta pontual.

Este mesmo padrão de comportamento pode ser observado na Figura 6, que mostra o valor médio de verificações de sobreposição de MBR. A sobreposição de MBR no espaço se deve à ordem de inserção dos dados e também à política de divisão de nós (*split*) empregada. Em todos os experimentos, a ordem de disposição dos elementos é aleatória e a política de *split* não foi alterada.

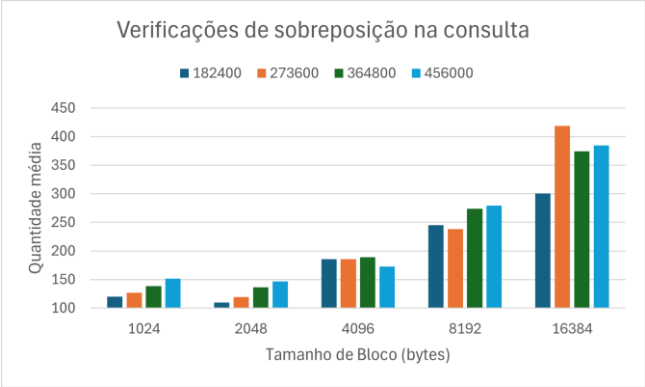

Figura 6: Número de verificações na consulta pontual.

Por fim, a Figura 7 mostra a quantidade de acessos a páginas de disco na operação de consulta pontual. Observa-se que esse valor decai conforme o tamanho do bloco aumenta. Porém, ao contrário do comportamento visto na operação de inserção de dados (Figura 4), a consulta não se mantém crescente ou decrescente, visto que as sobreposições de regiões do espaço de busca fazem com que o número de acessos não possa ser modelado de maneira uniforme.

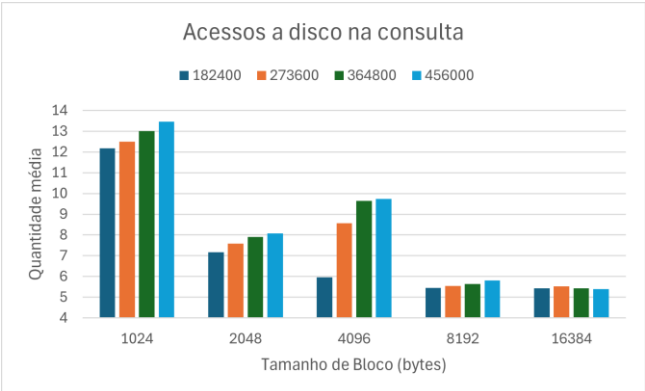

Figura 7: Número de acessos a disco na consulta pontual.

Conclusões

Este trabalho apresentou a implementação de uma estrutura de dados para indexação de dados espaciais. A estrutura tem sua base no índice R-Tree. A implementação foi realizada utilizando o *framework Object Injection* como base.

A modelagem dos nós da árvore compreende o armazenamento das informações de controle da estrutura (referências para os demais níveis) e uma área para a serialização dos objetos armazenados. A implementação dos métodos de inserção de dados na estrutura e consulta pontual seguem os algoritmos da R-Tree original.

Os experimentos realizados utilizando um conjunto de dados real mostraram que a estrutura se comporta de maneira mais uniforme quando os blocos de disco possuem tamanho 2 KB, para ambas as operações de inserção e consulta.

Trabalhos futuros envolvem a investigação das propriedades das ligações dentro do mesmo nível, que são um diferencial da estrutura implementada. Estas referências permitem obter ganho de desempenho durante operações de remoção de objetos e, possivelmente, na divisão dos nós.

Agradecimentos

Os autores agradecem à Universidade Federal de Itajubá por apoiar este trabalho de Iniciação Científica.

Referências

BROILO, M.; PIOTTO, N.; BOATO, G.; CONCI, N.; DE NATALE, F.G.B. (2010). Object Trajectory Analysis in Video Indexing and Retrieval Applications. Studies in Computational Intelligence, vol 287. Springer.

CARVALHO, L. O.; SERAPHIM, T. F. P.; TRAINA JÚNIOR, C.; SERAPHIM, E. (2013). ObInject: a NoODMG Persistence and Indexing Framework for Object Injection. Journal of Information and Data Management, 4(3), p. 220.

COMER, D. (1979). Ubiquitous B-Tree. ACM Computing Surveys, 11(2), pp. 121–137.

FEVGAS, A.; AKRITIDIS, L.; BOZANIS, P.; MANOLOPOULOS, Y. (2020). Indexing in flash storage devices: a survey on challenges, current approaches, and future trends. The VLDB Journal, 29, pp. 273–311.

MANOLOPOULOS, Y.; NANOPOULOS, A.; PAPADOPOULOS, A. N.; THEODORIDIS, Y. (2005). R-Trees: Theory and Applications. Springer.