UNIVEL CENTRO UNIVERSITÁRIO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

FERNANDO BIGUETTI MASCARELLI, GABRIELI APARECIDA KLAUCK e MURILO FERNANDES DOS SANTOS MELO

ÁRVORES B E VARIAÇÕES B* E B+

ESTRUTURA DE DADOS

Cascavel – PR

2023

FERNANDO BIGUETTI MASCARELLI
GABRIELI APARECIDA KLAUCK
MURILO FERNANDES DOS SANTOS MELO

ÁRVORES B E VARIAÇÕES B* E B+

Trabalho de pesquisa científica apresentado ao curso de Análise e Desenvolvimento de Sistemas no Centro Universitário de Cascavel – Univel.

Professor: Ederson Schmeing

Cascavel - Paraná
2023

RESUMO

Esse artigo científico se refere a uma discussão relacionada as árvores B, B* e B+ como estruturas de dados eficazes para o armazenamento e também pesquisas de grandes volumes de informações. Essas árvores balanceadas possuem capacidade de garantir tempos de pesquisa rápidos. São estruturas muito utilizadas em sistemas de arquivos e banco de dados, já que possuem um ótimo desempenho ao lidar com enormes quantidades de dados.

Nos últimos tempos, tem sido evidente o crescimento em relação ao volume de dados gerados e armazenados. Com isso, surge uma nova precisão de estrutura de dados que seja competente para lidar com enormes conjuntos de informações e que possa permitir operações rápidas. Apresentado esse contexto, surgem as árvores B que têm se evidenciado como uma resposta efetiva para sistemas de armazenamento.

As árvores B são um tipo de estrutura de dados muito conhecida e utilizada em ciência da computação e outros cursos semelhantes e também em sistemas de gerenciamento de banco de dados. Elas foram planejadas para realizar a recuperação e armazenar informações de uma maneira eficaz, principalmente em situações em que o tamanho dos dados é de bastante volume, e também que possua operações de inserção, busca e exclusão de forma constante. Nesse artigo, iremos conhecer e explorar aspectos fundamentais das árvores B e suas variações chamadas de árvores B* e B+. Essas árvores foram elaboradas para intensificar ainda mais o funcionamento das árvores B. Ao decorrer deste texto vamos explorar a estrutura básica das árvores B, com foco em como elas são capazes de realizar um balanceamento de dados eficiente.

As árvores B são um dos tipos de estrutura de dados em forma de árvore existentes, elas foram desenvolvidas para organizar e guardar quantidades exorbitantes de dados sem deixar de lado a eficiência. A árvore B possui uma característica bem marcante que são suas propriedades distintas, isso a diferencia de outras estruturas de árvore. As árvores B possuem algumas propriedades essenciais para um bom funcionamento e eficácia.

Ordem: A ordem de uma árvore B, é um parâmetro que designa o número máximo de filhos que um nó pode possuir. Isso quer dizer que um nó em uma árvore B pode abranger no máximo B-1 valores (chaves) e B ponteiros para os filhos. O valor de B é definido ao longo do projeto da árvore, sempre levando em conta os recursos disponíveis e o volume de dados.

Balanceamento: O balanceamento é de suma importância nas árvores B, é uma das características fundamentais, já que tem o papel de garantir que todas as folhas da árvore sejam niveladas, assim ficando no mesmo nível. Isso denota que todas as ramificações da árvore têm a mesma altura. Esse balanceamento é feito através de operações de reequilíbrio, como a reorganização de chave entre os nós e divisão ou fusão de nós quando relevante. Essas operações são feitas no decorrer das inserções e exclusões, certificando que a árvore mantenha um balanceamento correto.

Nó Raiz: As árvores B possuem apenas um único nó denominado nó raiz, que é o ponto inicial que da entrada para a árvore. O nó raiz pode abranger um número variável de chaves e ponteiros para os filhos. Nesse tipo de árvore, os nós não folha (com exceção o nó raiz) possuem um mínimo ceil(B/2) chaves, ceil é a função de teto, inteirando para cima. Isso garante que os nós não fiquem muito vagos e contribui para um melhor funcionamento da árvore B.

Chaves Ordenadas: Uma árvore B guarda seus valores (também chamados de chaves) de forma organizada dentro de cada nó. Isso assente a realização de busca binária eficaz, diminuindo o tempo necessário para achar uma chave específica. O fato das chaves serem ordenadas ajuda a facilitar a busca e demais operações, que podem ser feitar de forma eficiente dispondo da estrutura ordenada da árvore B.

Sabendo dessas propriedades primordiais da árvore B, também devemos conhecer mais detalhes sobre esse tipo de funcionamento estruturado, e uma das maiores vantagens desse tipo de árvore é sua busca eficiente. A busca parte do nó raiz e segue uma busca binária, que compara a chave pretendida com as chaves presentes no atual nó, decidindo assim qual ramo percorrer. Isso ocorre de maneira recursiva até que a chave desejada seja detectada ou seja definido que ela existe na árvore. Atrelado a isso a inserção e exclusão realizada nas árvores B são feitas com o intuito de manter a propriedade de balanceamento. Quando se adiciona uma nova chave na árvore, ela é posta no nó cujo qual é apropriado a ela, de forma que mantenha a ordem das chaves, caso o nó esteja em sua capacidade máxima, ele é dividido em dois e uma chave é promovida para um nível superior. Quando acontece a exclusão de uma chave ocorre um processo parecido, e isso garante que a árvore continua balanceada e com as chaves ordenadas. Como já visto anteriormente nesse artigo, o balanceamento das árvores B é automático, a medida em que as operações são realizadas ocorre também o reequilíbrio da árvore, esse balanceamento automático que as árvores desse tipo possuem ajuda a manter o desempenho consistente e evite que ela se torne desequilibrada. Uma das maiores vantagens de utilizar as árvores B é a eficiência em acesso a disco que é capaz de se atingir com essa estrutura de dados, é notória a redução de acesso a disco que se fazem necessários para o retomar de informações. Como esse tipo de árvore é otimizado para uso em sistemas de armazenamento secundário como SSD ou o disco rígido, elas são arquitetadas para usar o mínimo de operações no disco.

São muitas as vantagens que as árvores B oferecem em comparação as outras estruturas de dados existentes. Sua eficiência em acessos a disco é extremamente

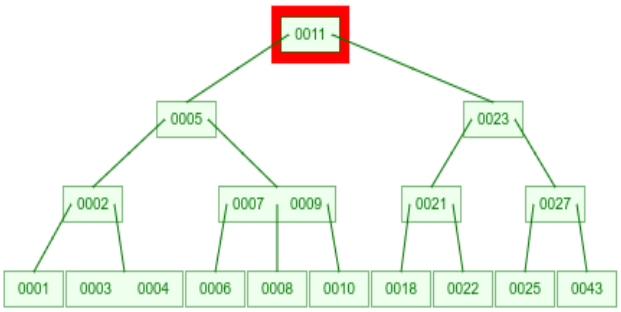
relevante a quem busca esse tipo de solução, já que reduzem exponencialmente o número de operações de leitura e gravação. Além disso, esse tipo de árvore é uma das melhores opções que pode ser encontrada atualmente para quem está tendo problemas ao lidar com um número alto de dados de forma escalável, mantendo um desempenho consistente apesar que o volume de dados aumente.

Nos bancos de dados as árvores B são aplicadas para a indexação e organização dos dados, permitindo consultas velozes e eficazes, otimizando todo o processo de desempenho de consultar grandes tabelas. As árvores B são muito utilizadas em índices primários e secundários, as chaves são mapeadas para posições físicas dos registros de disco, isso permite que as consultas sejam realizadas de forma satisfatória aos usuários, já que é feita de forma rápida e reduz muito no tempo de busca, melhorando o sistema como um todo. Já nos sistemas de arquivos as árvores B são aplicadas na parte de organização eficaz de diretórios e arquivos em estruturas hierárquicas. Cada diretório é simbolizado por um nó da árvore B, que guarda as informações referentes aos arquivos e subdiretórios. É isso que torna realizável a navegação eficiente pelos diretórios, deixando as buscas mais objetivas e o acesso aos arquivos mais fácil. Esse tipo de árvore em sistemas de arquivo é o que garante um rápido acesso aos dados ali contidos, assim como a organização mais escalável e efetiva. Nos caches de memória, onde ficam os dados que são acessados com maior frequência as árvores B são utilizadas para a melhoria no acesso a esse tipo de dado. São utilizadas como estruturas de indexação para guardar informações sobre os dados que são armazenados no cache, ao utilizar essa árvore, é capaz de fazer uma busca eficaz pelos dados que se deseja, reduzindo tempo de acesso. Isso traz uma melhora considerável no desempenho dos sistemas que operam com o uso dos caches de memória. Por fim falando sobre as aplicações das árvores B, é necessário citar os sistemas de armazenamento distribuído onde elas também podem ser aplicadas. Nesses sistemas de armazenamento distribuído os dados são divididos e compartilhados em vários nós ou servidores. As árvores B são usadas para fazer a organização e indexar os dados que foram distribuídos, isso concede a localização rápida e eficaz dos dados em diferentes nós distribuídos pelo sistema, isso é extremamente útil em sistemas que precisam da escalabilidade e tolerância a falhas, já que as árvores B permitem que ocorra a distribuição de dados e que eles possam ser acessados de forma eficiente nesse tipo de ambiente.

Podemos notar que as aplicações das árvores B são inúmeras e incluem diversos sistemas de armazenamento. Sua capacidade de oferecer o rápido acesso

aos dados, uma otimização eficaz de busca e organização e a sua competência de lidar com enormes volumes de dados, torna as árvores B uma escolha comum aos sistemas que precisam de eficiência, bom desempenho e escalabilidade.

Representação gráfica 1 - Árvore B



Fonte: WikiBooks[1].

Árvore B*

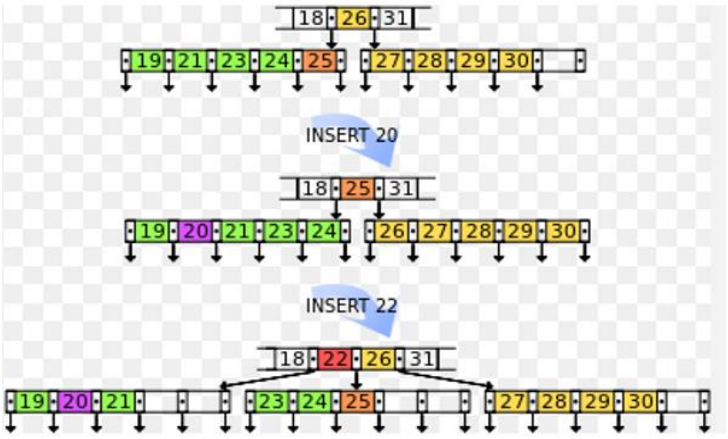
Agora que já fizemos uma ampla análise sobre árvore B em geral é necessário conhecermos uma de suas variações que é conhecida como árvore B*, ela acabou sendo criada pois surgiram algumas limitações referentes a árvore B, uma delas era o alto custo de manutenção em seus ponteiros e também o espaço em disco, uma vez que a estrutura armazenava muitos ponteiros para cada nó da árvore. Além disto a árvore B não era ideal para consultas que envolviam intervalos de valores, já que ela exigia uma busca completa pela árvore para encontrar os dados desejados. Para superar essas limitações Donald E. Knuth sugeriu uma evolução no ano de 1973, por ela ser uma variação apresenta diversas formas de inserção, remoção e busca muito semelhantes com as que são realizadas nas árvores B, mas com a diferença de que a técnica aplicada na redistribuição das chaves também é empregada durante as operações de inserção. Essa técnica é conhecida pelo nome de “divisão de dois para três” ou em inglês “two-to-three split”, que acaba proporcionando propriedades diferentes das árvores B. A grande melhoria está no que a abordagem proporciona, que é por sua vez um aproveitamento melhor do espaço da estrutura, pois no pior dos casos cada nó apresenta no mínimo 2/3 do número máximo de chaves que ele pode

[1] Disponível em: <https://pt.wikibooks.org/wiki/Algoritmos/Estruturas_de_dados/%C3%81rvore_B>. Acesso em: 25 de maio, 2023.

armazenar, enquanto a divisão usual de cada página apresenta, no pior caso, metade do número máximo de chaves.

O funcionamento das árvores B* é definido da seguinte maneira, os nós internos que atuam como pontos de divisão para os dados armazenados na árvore, esses nós internos contêm chaves para que sejam organizadas em ordem crescente, por sua vez, essas chaves dividem a árvore em subárvores menores, assim cada nó interno, com exceção da raiz, contém no mínimo [h-max/2] chaves e no máximo [h-max/1] chave, onde h-max é o número máximo de chaves permitido em cada nó interno. Também encontramos nessa estrutura os nós-folha que contém os dados propriamente ditos, cada um dos nós-folha armazena um intervalo de chaves que ele abrange e contém referências para os dados correspondentes, geralmente armazenados externamente, como em um disco rígido. Os nós-folha estão no mesmo nível da árvore, isso facilita o acesso sequencial aos dados armazenados na árvore. Por sua vez, o balanceamento das árvores B* é feito através da propriedade “h-max”. Essa propriedade especifica o número máximo de chaves em cada nó interno, exceto para a raiz da árvore, que pode conter no máximo h-max/1 chaves. O balanceamento é de extrema importância para garantir que a altura da árvore seja mantida em um nível razoável, o que ajuda a garantir um desempenho eficiente nas operações de busca, inserção e exclusão. As operações de busca nas árvores B*, para realizá-las é necessário começar na raiz e comparar a chave de busca com as chaves de nó. Com base na comparação, você pode determinar qual subárvore deve ser explorada em seguida, o processa de busca continua descendo pela árvore até que a chave desejada seja encontrada em um nó folha. Caso a chave não seja encontrada, a busca retorna um resultado indicando que a chave não existe na árvore. As operações de inserção são realizadas ao inserir uma nova chama em uma árvore B*, você inicia a partir da raiz e encontra a posição adequada para a chave na estrutura, se a subárvore onde a chave deve ser inserida estiver cheia, é necessário dividir o nó interno correspondente. A divisão envolve a redistribuição das chaves e a criação de um novo nó interno. Esse processo continua recursivamente até que a chave seja inserida em um nó folha. Para excluir uma chave de uma árvore B* utilizamos das operações de exclusão, você começa encontrando o nó folha que contém a chave, logo em seguida, você remove a chave desse nó, se a remoção resultar em um número insuficiente de chaves no nó folha, é necessário fazer uma redistribuição ou fusão com outros nós-folha para manter a propriedade “h-max”.

Representação gráfica 2 - Árvore B*



Fonte: Wikipédia[2].

Árvores B+

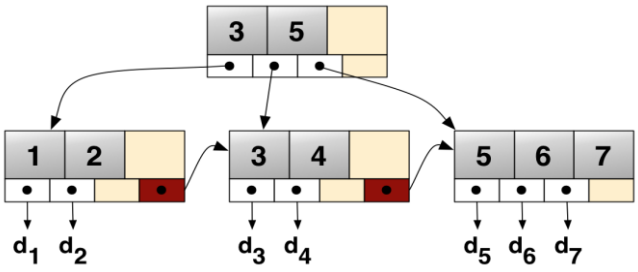
As árvores B+ são estruturas de dados amplamente utilizadas em bancos de dados e sistemas de armazenamento para organizar e indexar grandes volumes de dados de forma eficiente, sendo uma das variações das árvores B, foram introduzidas por Rudolf Bayer e Edward McCreight em 1972, com melhorias adicionais. As árvores B+ são formadas por um, nó raiz, nós internos e nós folha, os nós internos armazenam chaves e ponteiros para outros nós, enquanto os nós folha contêm as chaves e os dados associados. A principal diferença entre as árvores B e B+ é que, nas árvores B+, todas as chaves são armazenadas nos nós folha, enquanto os nós internos contêm apenas chaves de indexação. Essa característica permite um acesso mais veloz aos dados, pois a busca é realizada diretamente nos nós folha.

A principal vantagem das árvores B+ é sua eficiência em operações de leitura sequencial e busca por intervalos. Como todas as chaves estão nos nós folha e são encadeadas, a leitura sequencial é mais rápida do que nas árvores B, que exigem a navegação pelos nós internos. Além disso, a estrutura das árvores B+ facilita a busca por intervalos de chaves, tornando-as ideias para consultas que envolvem intervalos de valores. No entanto, as árvores B+ também apresentam algumas desvantagens, já que elas requerem mais tempo e operações de I/O para inserções e exclusões de chaves, pois envolvem modificações em vários nós folha. Além disso, as árvores B+ podem ocupar mais espaço em disco devido à duplicação das chaves nos nós folha.

[2] Disponível em: <https://pt.wikipedia.org/wiki/%C3%81rvore_B*>. Acesso em: 26 de maio, 2023.

Em relação as funções de funcionamento desse tipo de árvore podemos começar com as inserções que ocorrem percorrendo a árvore da raiz até o nó folha adequado para a chave ser inserida, se o nó folha não estiver cheio a chave é inserida no local correto, mantendo a ordem das chaves, entretanto se o nó folha estiver em sua capacidade máxima é realizada uma divisão, criando um novo nó folha e redistribuindo as chaves entre os nós folha adjacentes. Já a remoção é realizada percorrendo a árvore até encontrar o nó folha que contém a chave a ser removida, se a chave estiver presente nesse nó folha, ela é removida e as chaves restantes são reorganizadas, se necessário. Caso a remoção deixe o nó folha com um número insuficiente de chaves, é efetuado um ajuste que redistribui as chaves entre os nós folha adjacentes. Quando é necessário realizar uma busca em árvore B+ se começa da raiz, comparando a chave desejada com as chaves no nó atual, se essa chave estiver presente no nó atual o valor associado é retornado. Caso contrário, seguimos o ponteiro apropriado para o próximo nó com base na comparação da chave, continuando assim até encontrarmos a chave desejada ou chegarmos a um nó folha. Para efetuar uma impressão nessas árvores as chaves são exibidas em ordem, percorrendo a árvore B+ de forma recursiva, começando pela raiz. É comum utilizar percurso em ordem para imprimir as chaves, o que garante que elas sejam exibidas de forma ordenada. Por fim, não devemos esquecer de falar sobre o balanceamento das árvores B+ que é extremamente necessário para manter a propriedade de balanceamento da árvore, isso é feito durante a inserção e remoção, garantindo que a árvore permaneça equilibrada e otimizada para operações de busca. Durante esses processos, ajustes e redistribuições de chaves podem ocorres nos nós folha e nós internos, mantendo a estrutura balanceada da árvore.

Representação gráfica 3 - Árvore B+



Fonte: Wikipédia[3].

[3] Disponível em: <https://pt.wikipedia.org/wiki/%C3%81rvore_B%2B>. Acesso em: 26 de maio, 2023.

A escolha da estrutura de árvore mais adequadas entre as apresentadas nesse artigo depende das necessidades e requisitos específicos de cada caso em questão. Cada uma dessas estruturas tem suas particularidades e características adequadas a cada demanda.

A árvore B é adequada e usada principalmente em sistemas de arquivos e bancos de dados para oferecer eficácia na busca sequencial e na inserção/remoção de registro, são úteis particularmente para acesso aleatório eficiente a dados armazenados em disco, onde a leitura sequencial pode ser intensa, elas foram projetadas para diminuir o número de acessos a disco. Por sua vez a árvore B* que é uma extensão da árvore B possui melhorias adicionais para um desempenho superior em termos de busca e operações de intervalo. Ela utiliza ponteiros adicionais para fazer uma redução na altura da árvore e, assim, minimizar o número
de acessos a disco inevitáveis. A árvore B* é em sua maioria mais utilizada em sistemas de bancos de dados que precisam de busca eficiente de intervalos, não muito diferentemente da árvore B. Por último a árvore B+ que também é uma ampliação da árvore B, muito usada assim como as demais em sistemas de banco de dados. Ela é elaborada para leituras em sequência e operações de intervalo. Suas características mais marcantes são suas chaves estarem todas presentes apenas nas folhas, havendo um link de ligação entre as folhas e elas serem colocadas em uma lista encadeada de forma organizada, o que facilita a busca sequencial efetiva. Essa árvore é apropriada para eventualidades em que a recuperação eficiente de um intervalo de chaves constantes é uma necessidade contínua.

Em conclusão, a árvore B é a mais adequada para acessos aleatórios em disco, a sua variação árvore B* é a de maior eficiência para buscas de intervalo e a árvore B+ é principalmente indicada para situações de recuperação eficaz de intervalos e leitura sequencial. A escolha de qual estrutura de árvore dependerá de cada uso em particular.

Referências:

Comer, D. (1979). **Árvore B onipresente**. ACM Computing Surveys (CSUR), 11(2), 121-137.

Fischbeck, SE (1987). **A onipresente árvore B: volume II**.

Introdução da árvore B+ (2023) GeeksforGeeks. Disponível em: <https://www.geeksforgeeks.org/introduction-of-b-tree/>. Acesso em: 20 de maio de 2023.

Maelbrancke, R., & Olivié, H. (1995). **Otimizando a implementação de Jan Jannink da exclusão da árvore B+**. ACM Sigmod Record, 24(3), 5-7.

S Giuma, AE (2015). **Estudo comparativo sobre busca de dados em técnicas de lista encadeada e árvore B e árvore B+** (dissertação de doutorado, Universiti Tun Hussein Onn Malaysia).

Árvore B+ Estruturas de Dados, Javatpoint. Disponível em: <https://www.javatpoint.com/b-plus-tree>. Acesso em: 20 de maio de 2023.