ANÁLISE COMPARATIVA DE ESTRUTURAS DE INDEXAÇÃO, ÁRVORE B+ E HASH LINEAR, PARA OPERAÇÕES EM BANCO DE DADOS

Leandro Teixeira Martins[(1)] , Saulo Sander Chaves[(1)]

[(1)]Instituto Federal de Minas Gerais (IFMG) - Campus Bambuí

lleandroltm@hotmail.com, sausancha@gmail.com

RESUMO

O artigo apresentado aborda o desempenho comparativo entre duas estruturas de dados, Árvore B+ e Hash Linear, usualmente utilizadas para organização de arquivos de banco de dados. Tais estruturas são conhecidas pelas características funções de armazenamento e indexação de registros de dados que visam suportar um acesso rápido a dados desejados. O trabalho se iniciou pela implementação das duas estruturas em linguagem de programação muito conhecida, o *Python*. Em seguida, com a ajuda de um arquivo de dados do tipo inteiro, foram executados quatro comandos diferentes em ambas as estruturas. À partir da medida de tempo, a análise do comportamento das duas estruturas se deu pela variação dos campos de cada registro de dados do arquivo, variação do tamanho das páginas (em *bytes*), e, execução das operações de inserção e remoção de dados.

Palavras-chave: Banco de Dados. Árvore B+. Hash Linear. Python.

1 INTRODUÇÃO

Naturalmente os dados básicos em um SGBD, bem como explica Ramakrishnan (2008), é um conjunto de registro, ou um *arquivo*, e cada arquivo consiste de uma ou mais página. O que necessariamente o autor quer transcrever é a precisão com que tais dados devem estar organizados cuidadosamente para suportar um acesso rápido a subconjuntos de registro em dados desejados. O objetivo principal é compreender como os registros estão organizados e, assim, obter um eficaz uso do banco de dados.

Por isso, o presente trabalho busca sedimentar os conhecimentos acerca das implementações de indexação em banco de dados, realizadas por duas estruturas de dados, Hash e Árvore. Tais estruturas, por meio de seus índices, organizam registros de dados em disco no intuito de aperfeiçoar determinados tipos de operações de recuperação.

A indexação baseada em Hash utiliza da técnica *hashing* para primordialmente encontrar rapidamente registros que possuam determinado valor por meio da chamada chave de pesquisa. Esse mecanismo ajuda a acelerar operações de pesquisa não suportadas de maneira eficiente pela organização de arquivo usada para armazenar os registros de dados.

Entre os dois tipos de *Hashing* existentes, onde os dois possuem a função *hash* na chave de pesquisa ao qual irá apontar o *bucket* onde se encontra o registro a ser recuperado, temos o *Hashing*

Extensível que usa um diretório para suportar inserções e exclusões eficientemente sem páginas de overflow. Entretanto, para a execução dos testes de desempenho a serem avaliados neste trabalho foi escolhido arbitrariamente o esquema de *Hashing Linear*, que de acordo com Ramakrishnan, usa uma política inteligente para criar novos *buckets* e suporta inserções e exclusões de forma eficiente sem o uso de um diretório.

Alternativamente ao Hash, a estrutura de dados Árvore, possui as entradas de dados organizadas de maneira ordenada pelo valor da chave de pesquisa. Com uma estrutura de dados de pesquisa hierárquica, essa estrutura nos permite localizar de forma eficiente todas as entradas de dados com valores de chave de pesquisa em um intervalo desejado (RAMAKRISHNAN, 2008).

Conforme citado e estudado pelo presente autor já citado neste trabalho, ao optar pela estrutura dinâmica conhecida como Árvore B+, considera-se que a mesma além de se ajustar a mudanças no arquivo de maneira harmoniosa, também é a estrutura de índice amplamente usada por suportar tanto consultas por igualdade quanto por intervalo.

Outra característica importante a ser abordada em relação à organização de um arquivo é se este é *agrupado* ou *não agrupado*. Sendo o primeiro a opção de implementação deste trabalho, de acordo com Ramakrishnan, quando um arquivo é organizado de forma que a ordenação dos registros de dados seja a mesma ou parecida com a ordenação das entradas de dados em algum índice, dizemos que o índice é grupado, caso contrário, é um índice não agrupado.

Por isso à reafirmar o objetivo do trabalho, com implementação efetuada na linguagem de programação Python, é realizado um comparativo entre o desempenho das estruturas de dados de indexação Hash Linear e Árvore B+, ambas com índices agrupados, executado sobre um mesmo arquivo de dados e representação dos resultados em gráficos gerados através da própria linguagem de programação. Por meio da medida de tempo e de quatro quesitos, o intuito é visualizar o desempenho e o comportamento de cada uma das estruturas, executando as operações de inserção e remoção de dados, variando o tamanho dos campos de cada registro de dados do arquivo, e, variando o tamanho das páginas em *bytes*.

2 METODOLOGIA

2.1 Materiais

2.1.1 Linguagem Python, versão 3;

2.1.2 Editor de texto Sublime;

2.1.3 Ultrabook HP Pavilion 14-b080br para teste;

2.1.3.1 Processador Intel Core i5-3317U de 3ª geração e 1,70 GHz, com tecnologia Turbo Boost de até 2,60 GHz;

2.1.3.2 4 GB de SDRAM DDR3 (1 DIMMs);

2.1.3.3 Gráficos Intel 4000 de alta definição com até 1664 MB de memória de vídeo total;

2.1.3.4 Disco rígido de 500 GB (5400 RPM) com proteção de unidade de disco rígido HP ProtectSmart;

2.1.3.5 Cache de aceleração de disco rígido (cache de unidade de estado sólido de 32 GB) com tecnologia Intel Rapid Start e tecnologia Intel Smart Response.

2.1.4 Linux Mint Cinnamon;

2.1.5 Ferramenta SIOgen.

2.2 Metodologia

Para realizar os testes inicialmente foram implementadas as duas estruturas de índices agrupados, a Hash linear e Árvore B+, proposta durante toda a disciplina de Banco de Dados II. Utilizando a linguagem Python e o editor de texto sublime, foi possível implementar toda a estrutura com auxílio do paradigma de orientação objeto. As classes foram organizadas de forma que um objeto teria o endereço de memória de um outro objeto.

Para isso foi criada a classe "*Pag*" que possui os atributos de uma página, e outra classe "*Hash*" ou "*Bmais*" que possui os atributos e métodos com a lógica de cada índice. Cada estrutura possui sua classe "*Pag*", e foram separadas em pastas diferentes. Para a execução dos algoritmos foi utilizado o terminal do Linux chamando o Python3 e passando o nome do arquivo. A junção das classes para executar todos os testes de uma vez, sem precisar executar a Hash e depois a Árvore, por exemplo, foi criada uma única classe "*main*", que cria uma instância das classes e faz a leitura de um arquivo *csv* com os dados de inserção e remoção. Na figura 1 temos um diagrama de classe simbolico para demostrar somente a estrutura de classes, e como elas se comunicam, omitindo atributos e métodos que não são dependentes.

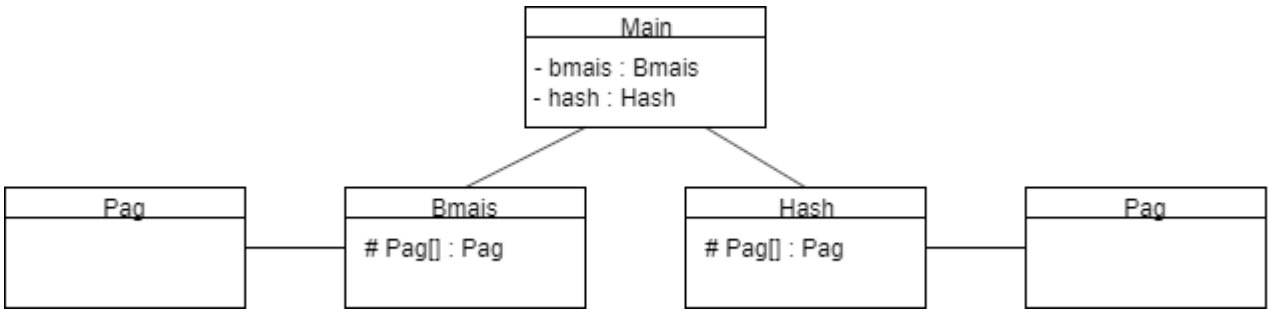

Figura 01 – Diagrama simbólico.
Fonte: Autores (2019).

O arquivo *csv* com os dados é criado por uma ferramenta chamada “SIOgen”, que é um gerador de dados, ele gera dados sintéticos compostos por exclusões e inserções de dados inteiros, onde o primeiro atributo que define se o conjunto de dados daquela linha é uma inserção ou remoção. Usando o caractere “+” e “-“, onde o “mais” serve para inserções e o “menos” para remoções. É possível diferenciar as linhas e usar esse atributo como parâmetro de escolha dentro da implementação (RIBEIRO, 2019).

Após criar os arquivos *csv* com os dados, foi definida a quantidade de testes e valores de cada teste exemplificados na Tabela 1. Os campos da tabela que estão marcados com a cor cinza são os valores padrões utilizados nos testes dos outros campos. Por exemplo, para os testes da quantidade de campos de cada dado na página, foram usados os valores de 4 a 32 campos mostrados na tabela e com tamanho da página fixado em 1536, as inserções em 512000, e as remoções em 256000. As escolhas dos campos padrões seguiram a lógica de campos e tamanho de página definido com um valor intermediário entre os valores de teste, e, para inserções e remoções, se definiram os valores máximo ou próximo do máximo.

Variação	Valores								
Campos	4	8	12	16	20	24	28	32	
Tamanho da Pagina	512	1024	1536	2048	2560	3072	3584	4096	
Inserções	2000	4000	8000	16000	32000	64000	128000	256000	512000
Remoções	2000	4000	8000	16000	32000	64000	128000	256000	512000

Tabela 01 – Valores para teste.
Fonte: Autores (2019).

Com a utilização das bibliotecas CSV e METPLOTLIB os gráficos foram gerados e os dados foram salvos em um arquivo *csv*. Para gerar os gráficos, foi executado cada teste cinco vezes e calculado a media dos valores encontrados.

3 RESULTADOS E DISCUSSÃO

A seguir para a parte de análise e discussões, os quatro gráficos apresentados nesta seção, foram gerados através da própria linguagem de programação e representam os quatro quesitos de comparação de desempenho por medida de tempo.

O primeiro gráfico faz uma variação do número de campos de cada registro contido no arquivo. O segundo gráfico demonstra a variação, no eixo horizontal, a variação do tamanho das

páginas em *bytes*. O terceiro gráfico, de inserção de dados, varia o número de registro de dados assim como o quarto gráfico, que executa a remoção de dados.

Em todos os gráficos o eixo vertical, ou em *y*, possui a medida de tempo com o objetivo de comparar o desempenho das estruturas de dados.

Iniciando a análise de comparação entre as estruturas de dados, Hash Linear e Árvore B+, para organização de arquivos em banco de dados, a variação do número de campos em cada registro de dados contido no arquivo, já nos apresenta, a priori, uma grande diferença entre as estruturas conforme se vê no gráfico 01. Quanto maior o número de campos de um registro de dados maior será o espaço ocupado em cada página, logo, o número de páginas tende a aumentar consideravelmente.

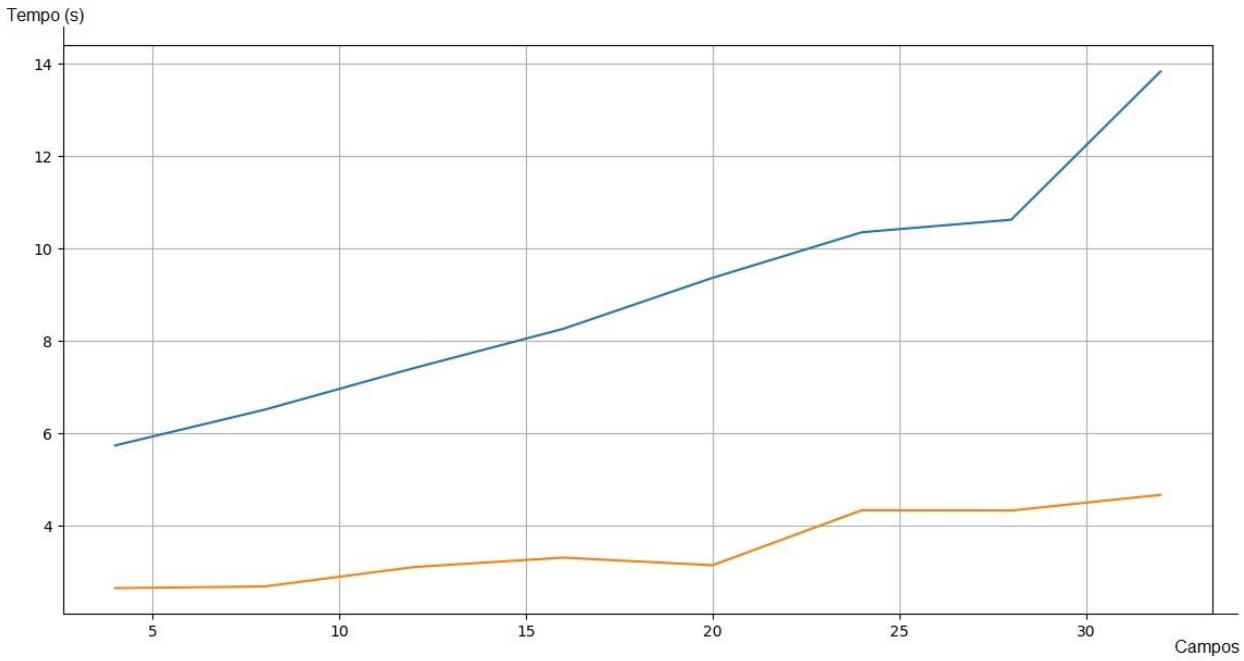

Gráfico 01 - Variação dos campos dos registros de dados.
Fonte: arquivo siogen disponibilizado pelo professor Marcos Riberio.

Como se pode notar, a variação com a qual a estrutura de dados em Árvore B+ inicializa o seu tempo em discrepância em relação à Hash Linear, com um tempo relativamente elevado, aproximados 4 segundos de diferença, e, como a sua inclinação também demonstra um comportamento ainda mais agudo, tendendo a valores ainda mais elevados ao fim da variação do número de campos, a complexidade da estrutura Árvore B+ lhe imputa o desempenho bem inferior em relação ao Hash Linear.

Para o gráfico 02, de comportamento semelhante no que se refere à comparação entre as duas estruturas de dados, a Árvore B+ inicializa em um tempo muito superior em relação ao Hash Linear. Diferença esta que se aproxima dos 12,5 segundos.

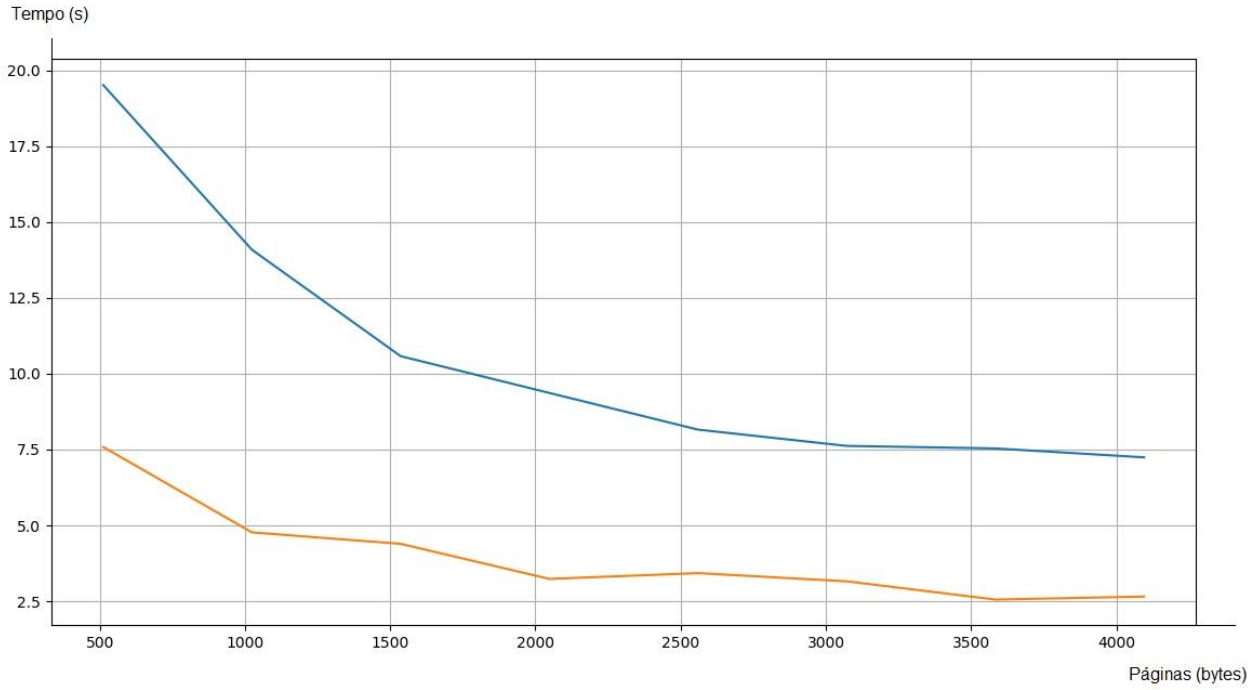

Gráfico 02 - Variação do tamanho (em bytes) das páginas.
Fonte: arquivo siogen disponibilizado pelo professor Marcos Riberio.

Com valor inicial de 500 bytes por página, este teste varia até 4.000 bytes por página e termina a curva de cada estrutura com um comportamento semelhante entre as mesmas. A grande diferença que há entre as duas estruturas na parte inicial do gráfico se ameniza quando o tamanho das páginas aumenta, e assim, o número de registros por página se intensifica. Conclui-se assim o motivo pelo qual o tempo das estruturas se inicializa de maneira elevada em relação à variação tamanho das páginas no eixo horizontal.

Também com uma curva mais acentuada, a Árvore B+ se apresenta com um desempenho inferior para a operação de inserção observada no gráfico 03.

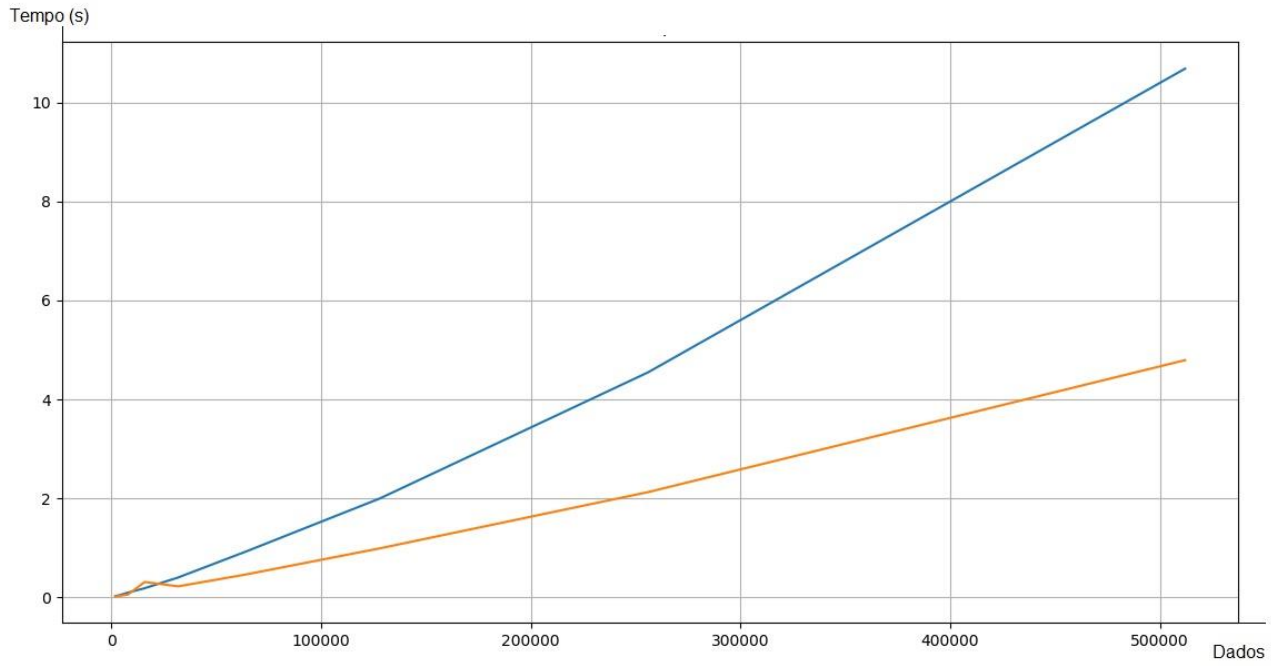

Gráfico 03 - Variação do número de entrada de dados.
Fonte: arquivo siogen disponibilizado pelo professor Marcos Riberio.

Devido à sua complexidade de arranjo, entre criar vários nós e dividi-los inserindo ordenadamente cada um dos registros de dados contido no arquivo, à medida que o número de dados aumenta com a inserção, a inclinação da reta da Árvore B+ se agrava, demonstrando assim, um desempenho inferior ao da Hash Linear. Já a segundo estrutura, apesar de demonstrar naturalmente um aumento no tempo com inserção de um número maior de dados, tem um comportamento mais brando quanto à inclinação da curva.

E por fim, o gráfico 04 é o antagônico aos demais, uma vez que o desempenho da Hash Linear se demonstrou inferior, com um tempo superior, ao da Árvore B+ relativo às remoções.

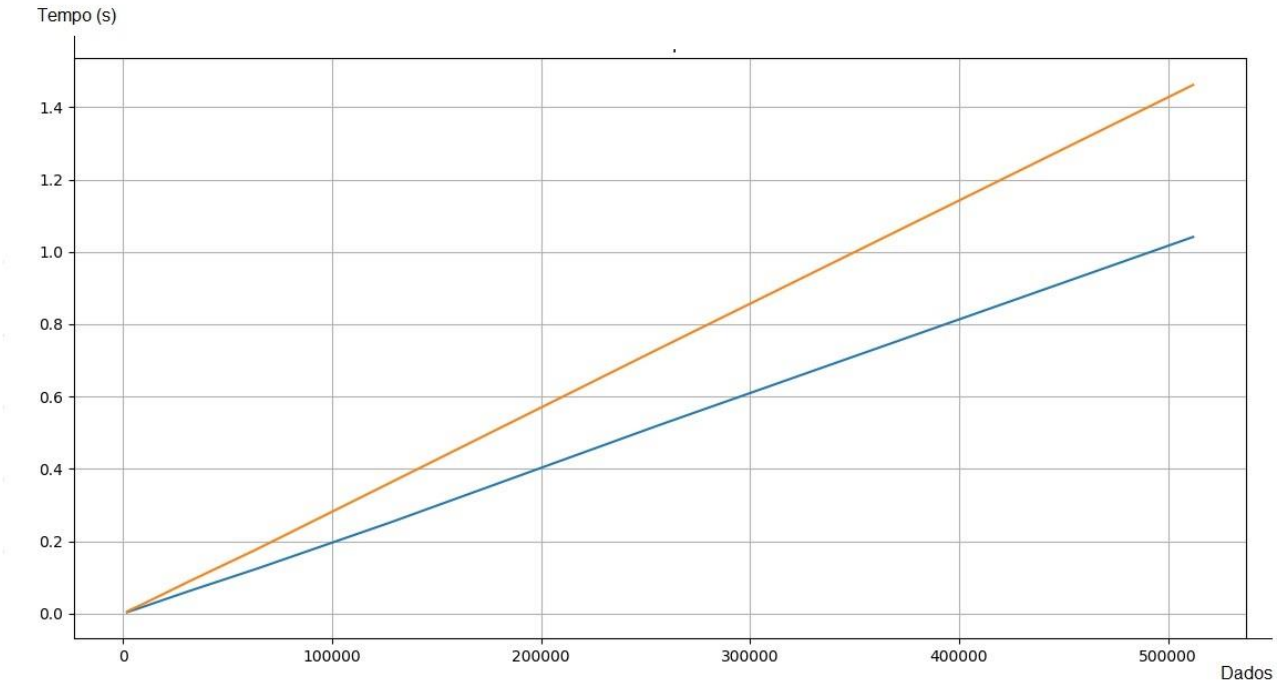

Gráfico 04- Variação do número de remoção de dados.
Fonte: arquivo siogen disponibilizado pelo professor Marcos Riberio.

As duas estruturas de dados apresentam um comportamento semelhante quanto à inclinação da curva, porém, assim como a estrura da Árvore B+ se torna complexa em suas operações com o aumento do número de entrada de dados, também se simplifica à medida que as remoções de dados ocorrem em seu interior.

Diante dos dados apresentados, apesar da Árvore B+ apresentar um desempenho inferior ao da Hash Linear em três das quatro análise, podemos levar em consideração que as duas estruturas possuem um comportamento aproximado quanto a variação da inclinação da reta, ao se variar de forma crescente, os quesitos situados no eixo horizontal de cada gráfico.

4 CONCLUSÃO

A proposta para comparação das estruturas de dados, Hash Linear e Árvora B+, foram implementadas e executadas com sucesso na linguagem de programação escolhida, o *Python*.

Através dos gráficos foi possível elementarmente analisar as diferenças das estruturas para as operações propostas. Diante dos dados obtidos é possível traçar uma diferença de desempenho a favor do Hash Linear quanto à variação do número de páginas, variação do tamanho das páginas e na operação de inserção de dados. Porém, para a operação de remoção a Árvore B+ demonstrou um desempenho superior devido a sua estrutura que tende a se simplificar à medida que o número de dados diminui em seu interior.

Sendo assim, o trabalho proposto cumpre sua função no cerne da diferenciação existente entre as duas estruturas analisadas ao que se refere o desempenho de processamento

REFERÊNCIAS

RAMARKRISHNAN, Raghu. Sistemas de gerenciamento de banco de dados [recurso eletrônico] / Raghu Ramarkrishnan, Johannes Gehrke ; tradução: Célia Taniwake, João Tortello; revisão técnica Elaine Barros Machado de Sousa. – 3. ed. – São Paulo : McGraw-Hill, 2008.

RIBEIRO, Marcos R. SIOgen – Simple Inset Delete Dataset Generator. Maintained by ribeiromarcos. Published with GitHub Pages. Disponível em: <https://ribeiromarcos.github.io/siogen/>. Acesso em: 04 de dezembro de 2019.