

# B+TREE E HASH EXTENSÍVEL: COMPARANDO DESEMPENHO DE ALGORITMOS.

Rodrigo Matheus Rodrigues de Oliveira<sup>1</sup>; Daniel Winter Santos Rocha<sup>1</sup>.

<sup>1</sup>Alunos do Curso de Engenharia da Computação do IFMG - Campus Bambuí

## ABSTRACT

*This article aims to show the comparison made between two algorithms used in Database Management Systems (DBMS), B+Tree and Extensive Hash. Insertions, deletions and searches were performed in both structures, comparing their performance.*

*Keywords: B+ tree, extensive hashing.*

## RESUMO

Este artigo tem por objetivo exibir a comparação feita entre dois algoritmos utilizados em Sistema de Gerenciamento de Banco de Dados (SGBD), B+Tree e Hash Extensível. Foram efetuadas inserções, remoções e pesquisas em ambas estruturas, comparando seu desempenho.

Palavas-chave: B+Tree, Hash Extensível.

## Sumário:

1. Introdução	3
1.1. B+Tree	3
1.2. Hash Extensível	4
2. Comparação	5
2.1. Testes na B+Tree	5
2.2. Testes no Hash Extensível	8
2.3. Comparando os resultados	8
3. Conclusão	8
Referências	8

# 1. Introdução

## 1.1. B+Tree

Estruturas estáticas, como a árvore ISAM, que possui um conjunto de nós iniciais pré definidos, sofrem com o problema de longas cadeias de overflow. Fenômeno este definido pela existência de páginas adjacentes às folhas, criando uma cadeia que diminui a performance da estrutura em importantes operações, como pesquisa. Ao se inserir novos dados na ISAM depois de cheia, esta passa a criar páginas em forma de lista adjacentes às folhas. Este problema motiva o desenvolvedor a flexibilizar a estrutura para algo dinâmico, ajustando-a para inserir e deletar dados com o intuito de minimizar os prejuízos das cadeias de overflow.

A B+Tree (árvore B+) é uma estrutura de busca balanceada. Contém nós internos que tem o intuito de direcionar as pesquisas às folhas, onde se encontram os dados armazenados. Os valores contidos nos nós internos têm o papel de chaves de pesquisa e não armazenam os dados propriamente ditos, diferentemente de estruturas como a BTree, onde tanto os nós internos quanto as folhas armazenam os dados em questão.

Outro problema de árvores como BTree e ISAM é a disposição das páginas folhas. Para acessar uma folha adjacente, se faz necessário retornar a nós internos da árvore para depois efetuar a leitura da folha desejada. Na estrutura B+Tree, as folhas possuem ponteiros ligando umas às outras de forma encadeada. Desta forma, ao se organizar as folhas em listas duplamente encadeadas é possível acessar a sequência de páginas em qualquer direção, sem se fazer necessário a leitura de nós internos.

Ramakrishnan [2003] trás uma abstração da B+Tree, contida na figura 1. A seta no topo da figura indica a raiz da estrutura, local onde se inicia a leitura dos índices para efetuar operações sobre a B+Tree. O triângulo, chamado de “Index entries” contém os índices de chave para pesquisa. Os losangos, chamados de “Data entries”, contém páginas folha de arquivos armazenados. É possível observar setas ligando as páginas folha, buscando abstrair os ponteiros utilizados para criar a lista duplamente encadeada.

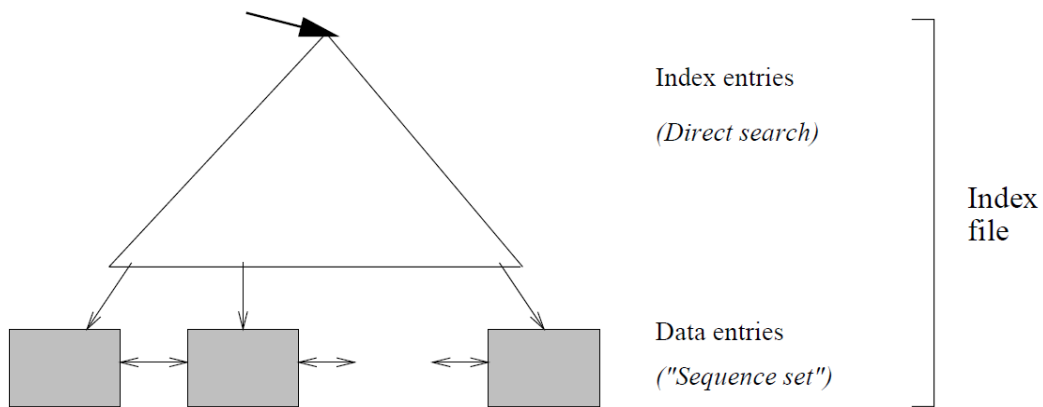


Figura 1: Estrutura de uma B+Tree.

Fonte: Ramakrishnan, Database Management Systems.

É de suma importância destacar algumas características que a estrutura deve possuir para ser denominada de B+Tree. Manter o balanceamento após uma inserção ou remoção é uma delas, também comum em estruturas BTree. Em relação às páginas internas e folha, exceto a raiz, devem respeitar uma ocupação mínima de 50% (cinquenta por cento). Por fim, ao se percorrer da raiz até duas folhas distintas quaisquer, a distância de ambas à raiz deve ser a mesma, característica essa garantida pelo balanceamento da B+Tree.

De acordo com Ramakrishnan [2003], uma estrutura B+Tree contém  $m$  entradas, onde  $d < m < 2d$ . O valor de  $d$  é um parâmetro da B+Tree, chamado de *ordem da árvore*, e mensura a capacidade dos nós. Novamente, a raiz é uma exceção, é necessário que seu  $m$  esteja nos limites de  $1 < m < 2d$ .

## 1.2. Hash Extensível

O hash é uma estrutura que podemos resumir em um conceito parecido com o de um vetor como o de hash linear. É uma estrutura de armazenamento por índices, através de uma função utilizada para encontrar qual bucket do hash o valor será inserido.

O hash apresentado por este artigo, é o hash extensível. Esse hash possui como uma de suas características é o *Overhead Cost* que apresenta em sua essência o rápido acesso à dados, com um custo mínimo de processamento. O hash utiliza a chave gerada pela função hash como uma pseudo-chave, que será quebrada conforme necessidade. A profundidade deve ser tal que, o número de *buckets* contenham todas as chaves, e é de suma importância que a função hash distribua bem as chaves conforme a figura 2.

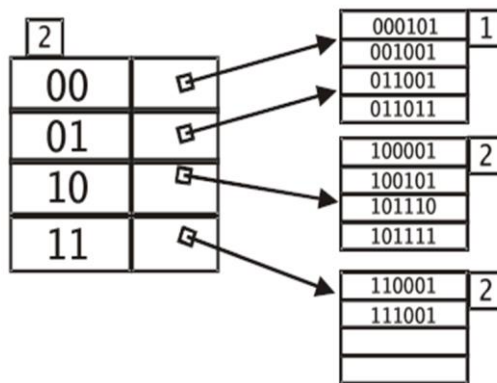


Figura 2: Estrutura de Hash Extensível.

Fonte: UFJF, Professor Jairo Souza.

## 2. Comparação

Este artigo busca, a priori, comparar testes feitos nas estruturas B+Tree e Hash Extensível utilizando entradas geradas na forma de arquivo CSV por um algoritmo disponibilizado pelo professor. Ambas estruturas foram implementadas utilizando a linguagem C++, no ambiente Sublime Text 3 e Code::Blocks e compiladas utilizando o GCC no sistema operacional Windows 7.

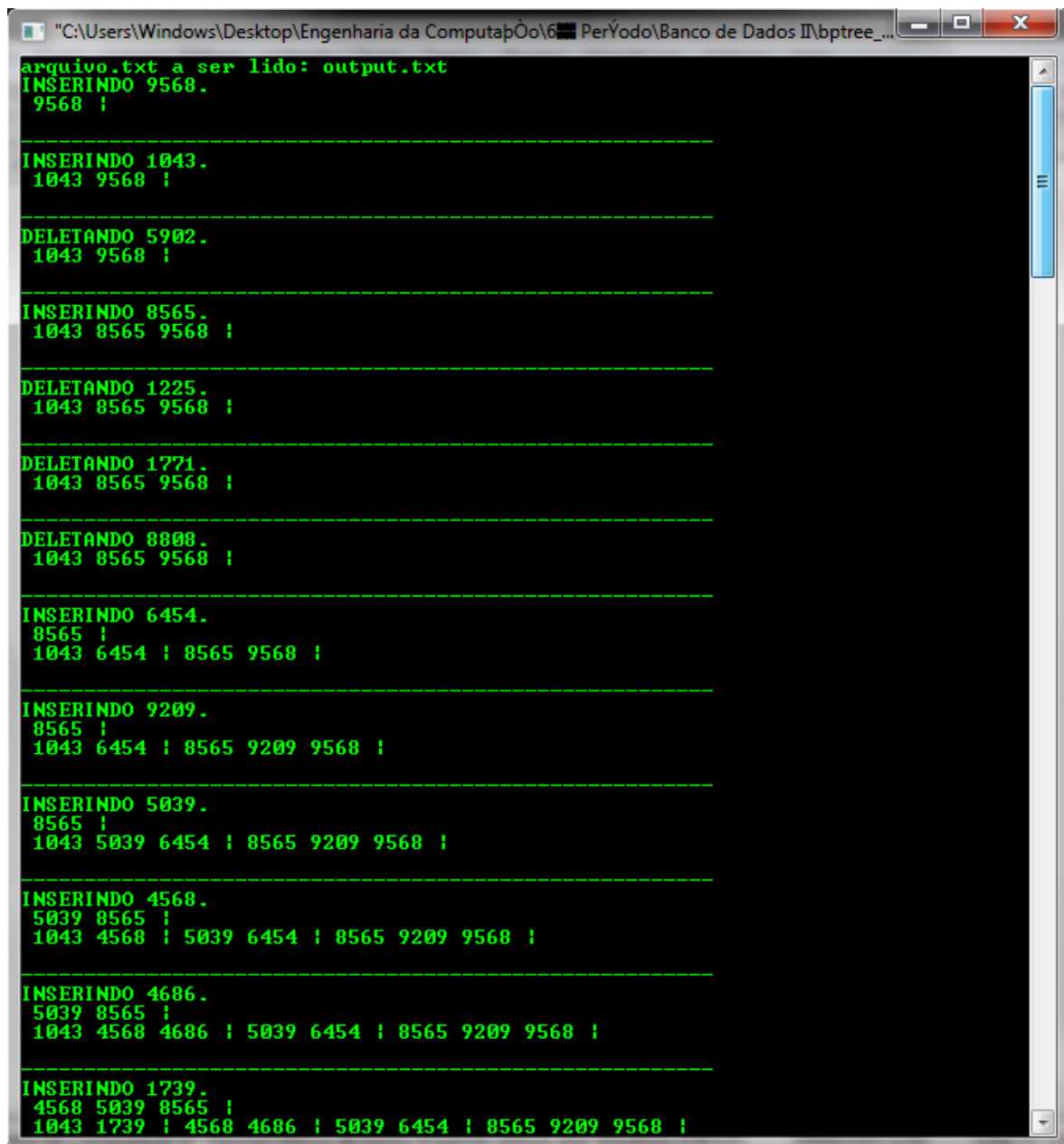
Devido a complexidade para a implementação de um algoritmo para leitura de arquivos CSV, optou-se por converter os arquivos gerados para TXT e, a partir dele, efetuar a leitura.

É importante tomar nota de que este não é um artigo com o intuito de explicar o funcionamento das estruturas de dados, apenas expor uma comparação entre as mesmas. Logo, pseudocódigos e informações detalhadas a respeito da implementação foram omitidas.

### 2.1. Testes na B+Tree

O teste inicial efetuado na B+Tree continha saídas em texto exibindo a estrutura a cada operação efetuada, conforme a figura 2. Ao longo do tempo, os valores exibidos chegavam a ocupar um grande espaço e demandar certo tempo para a escrita na tela. Isto comprometeu em grande escala o desempenho do algoritmo, interferindo na medição final do tempo de execução. Em 483 (quatrocentos e oitenta e três) segundos, equivalente a pouco mais de 8 (oito) minutos, apenas aproximadamente 1.500 (mil e quinhentas) operações das 20.000 (vinte mil) haviam sido efetuadas. Para medir resultados mais concisos foram removidas as linhas de código contendo as exibições. Então o código foi recompilado e

reiniciados os testes. Apenas as chaves foram exibidas a fim de priorizar a organização, os valores contidos nas folhas foram omitidos.



```
"C:\Users\Windows\Desktop\Engenharia da Computação\6º Período\Banco de Dados II\bptree_...
arquivo.txt a ser lido: output.txt
INSERINDO 9568.
9568 !

-----

INSERINDO 1043.
1043 9568 !

-----

DELETANDO 5902.
1043 9568 !

-----

INSERINDO 8565.
1043 8565 9568 !

-----

DELETANDO 1225.
1043 8565 9568 !

-----

DELETANDO 1771.
1043 8565 9568 !

-----

DELETANDO 8808.
1043 8565 9568 !

-----

INSERINDO 6454.
8565 !
1043 6454 ! 8565 9568 !

-----

INSERINDO 9209.
8565 !
1043 6454 ! 8565 9209 9568 !

-----

INSERINDO 5039.
8565 !
1043 5039 6454 ! 8565 9209 9568 !

-----

INSERINDO 4568.
5039 8565 !
1043 4568 ! 5039 6454 ! 8565 9209 9568 !

-----

INSERINDO 4686.
5039 8565 !
1043 4568 4686 ! 5039 6454 ! 8565 9209 9568 !

-----

INSERINDO 1739.
4568 5039 8565 !
1043 1739 ! 4568 4686 ! 5039 6454 ! 8565 9209 9568 !
```

Figura 3: Saída exibindo a B+Tree a cada operação.

Fonte: autores.

Após recompilar o código da B+Tree, a saída contendo o tempo de execução das operações foi coletada, como pode ser visto na figura 3. Foram necessários 364 (trezentos e sessenta e quatro) segundos, o equivalente a pouco mais de 6 (seis) minutos, para efetuar as 20.000 (vinte mil) operações propostas.

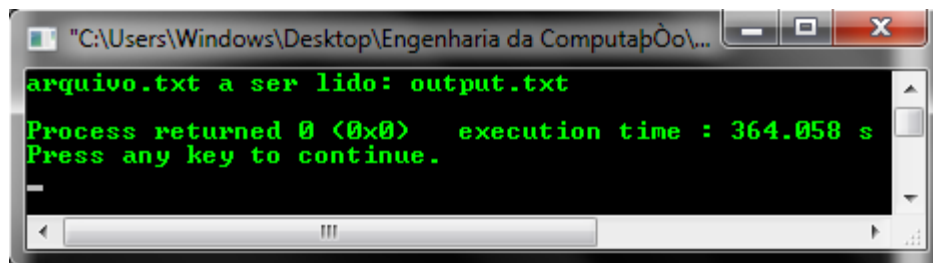


Figura 4: Tempo de execução da B+Tree para vinte mil operações.

Fonte: autores.

Outros testes contendo diferentes quantidades de operações foram efetuados. A tabela 1 contém a informação da quantidade de operações e seu respectivo tempo de execução. Também foi analisado um gráfico (gráfico 1) contendo uma comparação entre os diferentes valores de tempo obtidos.

Nº Operações	Tempo (s)	Tempo (min)
1000	5	0"5'
2000	7	0"7'
5000	23	0"23'
10000	76	1"16'
15000	179	2"59'
20000	364	6"04'

Tabela 1: Custo em tempo para diferentes quantidades de operações na B+Tree.

Fonte: autores.

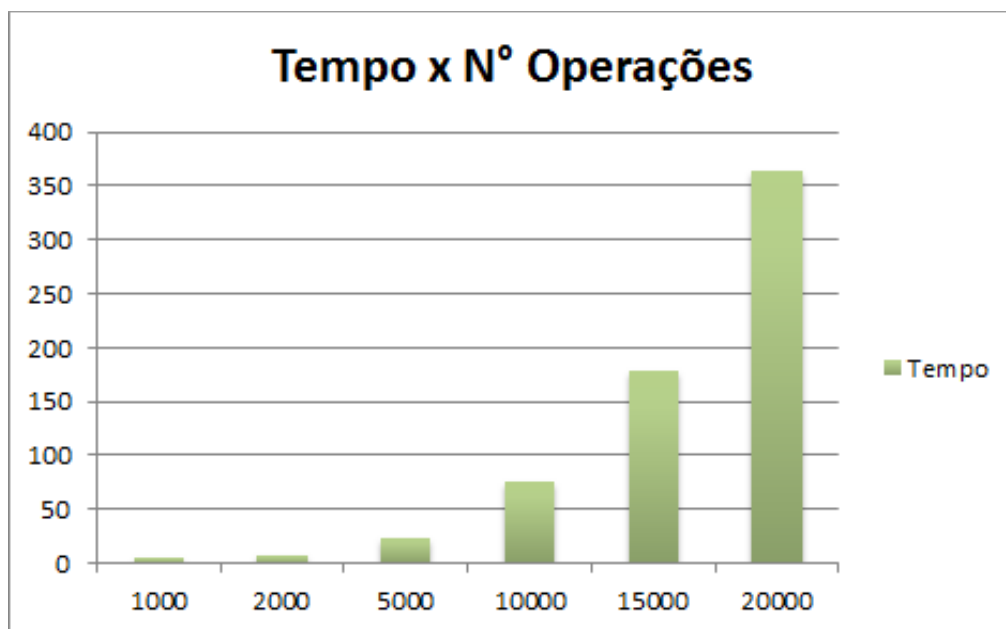


Gráfico 1: Custo em segundos para diferentes quantidades de operações na B+Tree.

Fonte: autores.

## 2.2. Testes no Hash Extensível

O algoritmo para a leitura do arquivo TXT foi implementado no Hash com sucesso. Porém, durante a execução dos testes utilizando os mesmo recursos, como arquivos de entrada e máquina, houve um problema não detectado pelos programadores. Assim sendo, testes não foram efetuados de forma completa como na B+Tree.

## 2.3. Comparando os resultados

Havendo dificuldades para efetuar os testes no Hash, ainda foi possível recolher alguns resultados, mesmo que um pouco pobres se comparados com os recolhidos da B+Tree. Como a diferença de quantidade de dados recolhidos foi discrepante, gráficos comparativos não foram gerados.

## 3. Conclusão

Concluídos os testes, foi possível observar que o Hash possui um melhor desempenho em comparação com a B+Tree, mesmo o número de dados recolhidos não sendo o ideal.

Não obstante, estes resultados se dão a estas implementações em específico, não assegurando o mesmo para implementações contidas em SGBDs comerciais.

Ademais, mantém-se o objetivo de refinar os algoritmos, modificando seu paradigma para orientação a objetos, além de efetuar a implementação do código para leitura de arquivos CSV. O trabalho ainda assim foi concluído com sucesso, como foi apresentado neste artigo.

## Referências

- [1] **Database Management Systems**. CTF. BJE. Library of Congress Cataloging-in-Publication Data. Ramakrishnan, Raghu. Acesso em: 23/11/2018. Disponível em: Biblioteca IFMG - Campus Bambuí, versão em português.
- [2] **Algoritmos/Estruturas de dados/Tabela Hash**. WikiLivros. Acesso em : 23/11/2018. Disponível em:  
<[https://pt.wikibooks.org/wiki/Algoritmos/Estruturas\\_de\\_dados/Tabela\\_Hash](https://pt.wikibooks.org/wiki/Algoritmos/Estruturas_de_dados/Tabela_Hash)>.
- [3] **Tabela HASH**. Departamento de Computação e Matemática - USP. Acesso em : 23/11/2018. Disponível em: <<http://dcm.ffclrp.usp.br/~augusto/teaching/icii/Hash-Tables-Apresentacao.pdf>>.