Trabalho 2
Árvore-B

SSC0603 - Algoritmo e Estrutura de Dados II

Professora: Elaine Barros Machado de Sousa

Alunos:

Carlos Henrique de Oliveira Franco	9771608
Guilherme Brunassi Nogima	9771629
João Pedro Silva Mambrini Ruiz	9771675
Leonardo Akel Daher	9771682

Sobre a Solução

Nesse trabalho foi cobrado dos alunos a implementação de uma Árvore-B de ordem 5, com a função de indexação de dados. e as operações de criação,inserção e busca.
Os registros de dados foram inseridos da seguinte forma:

1 . ID numérico da música
2 . Título da música
3 .Gênero

Tendo isso,organizamos o projeto em 9 arquivos :

- btree.h
- btree_create.c
- btree_create.h
- btree_insertion.c
- btree_insertion.h
- btree_search.c
- btree_search.h
- t2.c
- t2.h

Da estrutura de índice ficamos com o seguinte esquema:
[RRN raiz][próximo RRN free][folha] [Quantidade de chaves na página] [chave][offset]..[chave][offset] [filho]...

No trabalho,tivemos a utilização das seguintes estruturas de dados utilizadas:

Chave:

```
typedef struct {
    int id;
    offset_t offset;
} chave;
```

Página:

```
typedef struct {
  int rrn ;
    int cntChave ;
    chave *chaves ;
    int *filhos ;
    char folha;
} pagina;
```

Árvore-B:

```
typedef struct {
   int raiz ;
   int nextRrn ;
   int ordem ;    FILE *bTFile ;} bTree;
```

Acerca da estratégia da implementação das operações ,iremos separar em 3 partes para deixar o relatório de forma mais clara.

- Criação

Inicialmente, verificamos a existência de algum arquivo de Árvore B atualizado. Se existir, armazenamos o RRN da página raiz. Se não existir, criamos uma nova Árvore B.

- Inserção

Para o caso da inserção em Árvore-B, devemos estudar 4 casos possíveis de inserção: Árvore vazia, overflow em nó raiz, inserção em nós folha com e sem overflow.

Para árvore vazia, criamos o nó raiz e inserimos as chaves de modo ordenado até o seu overflow.

Para o overflow em nó raiz, particionamos o nó em dois, com chaves divididas igualmente entre eles.

Para a inserção em nós folha sem overflow, buscamos nas páginas para descobrir em qual devemos inserir o elemento em questão. Se houver espaço, o elemento é inserido e ordenado

Para a inserção em nós folha com overflow, buscamos a página a ser inserida, fazemos o particionamento com uma distribuição igual em ambas páginas.

- Busca

Na função de busca, a partir da raiz, verificamos se a chave encontra-se em alguma página, caso não seja encontrada verificamos então em uma subárvore, dessa forma continuamos procurando até achar a chave desejada, caso não é encontrada exibe mensagem de erro.

Complexidade:

Chegando ao fim, temos então de analisar a complexidade do métodos implementados nesse trabalho. Podemos separar em pior e melhor caso sendo o pior com uma ocupação da árvore com 2 elementos por página (no caso de ordem 5), e o melhor a ocupação máxima de 5 elementos por pagina.

Sendo N o número de chaves:

Pior Caso :

Busca = log_3 (N)
Inserção = log_3 (N)

Melhor Caso:

Busca = log_6 (N)
Inserção = log_6 (N)

Acerca do espaço,temos uma complexidade de linear,ou seja O(N).

Conclusão:

De acordo com o vimos em aula, e com o que estudamos com o trabalho, concluímos que a Árvore B é um bom método para realizar a indexação de arquivos, pois é uma estrutura que permite uma complexidade baixa (logarítmica) para todas operações.