

## Documentação para o Trabalho Prático 2 - Bancos de Dados I

Equipe:

David Pereira Bessa 21552409

Ebony Brandão Pereira 21650331

Vinícius Feitosa Monteiro 21753626

### Considerações Iniciais

Configurar Projeto:

Utilize **cmake -B build** para preparar o ambiente de construção, criando um diretório **build** dedicado para os arquivos de compilação.

Compilar Projeto:

Execute **cmake --build build** para compilar o projeto, utilizando as configurações geradas no diretório **build**.

IMPORTANTE : no cabeçalho 'constantes.hpp', é preciso adaptar o tamanho, para a informação de número de blocos lidos" não ficar incorreta. Botamos o comando do terminal como comentário no código.

Enfrentamos dificuldades na implementação das B-trees, o que acarretou em não implementarmos as funcionalidades dos programas 'seek1' e 'seek2', e como consequência nosso upload que não foi totalmente concluído, apesar de estar funcional (mas não faz as árvores B+). Nossos 'finderec' e o hash também estão funcionais (já que o findrec só usa o hash).

Dada a complexidade do trabalho, as funções foram desenvolvidas coletivamente por nossa equipe, então compartilhamos autorias e decisões de projetos, menos das árvores B + que foram trabalhadas por Ebony e Vinicius, mesmo que sem sucesso.

### Arquivo de índice - Hash

A classe **ArquivoHash** gerencia a criação, inserção e busca em um arquivo de índice hash. Ela é projetada para otimizar a recuperação de registros a partir de um identificador único (ID), utilizando a técnica de hashing para mapear IDs a localizações específicas no arquivo.

### **Função: ArquivoHash (Construtor)**

**Descrição:** Inicializa uma instância de ArquivoHash, abrindo o arquivo especificado em modo de leitura ou escrita. Se o modo for escrita, preenche o arquivo com buckets vazios.

#### **Parâmetros:**

***const char \*arquivo:*** Caminho para o arquivo de hash.

***int num\_buckets:*** Número de buckets a serem utilizados no hash.

***Modo modo:*** Modo de abertura do arquivo (leitura ou escrita).

- Retorno: N/A

### **Função: preenche\_buckets**

**Descrição:** Preenche o arquivo de hash com buckets vazios, cada um do tamanho especificado pela constante tamanho\_bloco.

**Parâmetros:** Nenhum.

- Retorno: void.

### **Função: funcao\_hash**

**Descrição:** Calcula e retorna o índice do bucket onde um id deve ser inserido ou buscado, utilizando uma função hash simples de módulo.

#### **Parâmetros:**

***int id:*** Id do artigo a ser hashado.

- Retorno: size\_t: Índice do bucket calculado pela função hash.

### **Função: insere**

**Descrição:** Insere uma nova entrada (id e offset) no bucket apropriado. Se o bucket estiver cheio, exibe uma mensagem de erro e não realiza a inserção.

#### **Parâmetros:**

***int id:*** Id do artigo a ser inserido.

***off\_t offset:*** Offset do artigo no arquivo de dados.

- Retorno: void.

### **Função: busca**

**Descrição:** Busca por um id específico nos buckets. Se encontrado, retorna o offset associado a esse id. Caso contrário, retorna um valor especial indicando que não foi encontrado.

### **Parâmetros:**

*int id:* Id do artigo a ser buscado.

- Retorno: off\_t: Offset do artigo no arquivo de dados ou NAO\_ENCONTRADO se o id não for encontrado.

## **Programas desenvolvidos**

### **Upload**

#### **Estrutura Geral**

O programa de upload é responsável por processar um arquivo CSV de artigos, criando um arquivo de dados binário não espalhado, além de arquivos de índice (Hash, primário e secundário) para facilitar buscas eficientes. Utiliza bibliotecas externas e internas para manipulação de arquivos e dados.

#### **Bibliotecas Utilizadas:**

- arquivo\_hash.hpp, artigo.hpp e constantes.hpp para estruturas de dados e utilitários.
- Padrões da linguagem C++: iostream, string, vector, e filesystem para operações de entrada/saída e manipulação de arquivos e strings.

### **Findrec**

#### **Estrutura Geral**

O programa findrec realiza a busca de um registro no arquivo de dados com base em um ID fornecido como argumento. Utiliza o arquivo de índice Hash para encontrar o offset do registro desejado. Se encontrado, o programa imprime os campos do registro e informa o número de blocos lidos.

### **Bibliotecas Utilizadas:**

- arquivo\_hash.hpp, artigo.hpp, constantes.hpp para manipulação dos registros e índices.
- Padrões da linguagem C++: iostream, string para operações de entrada/saída e manipulação de strings.

## **Artigo**

### **Estrutura Geral**

Programas criados para dar suporte aos dados do arquivo.csv divididos em artigo.hpp e artigo.cpp. O artigo.hpp é composto por duas structs, uma struct data\_hora para representar quando um artigo foi atualizado, e uma struct artigo para representar o artigo científico em si, e a declaração de duas funções para a manipulação da struct artigo, csv\_para\_artigo e imprime\_artigo. Já o artigo.cpp possui a implementação de quatro funções utilizando a struct artigo, duas já mencionadas anteriormente, a csv\_para\_artigo e imprime\_artigo, e duas novas funções copia e pega\_conteudo.

### **Bibliotecas Utilizadas:**

- Padrões da linguagem C++: cstdio, cstdlib e cstring para operações de entrada/saída e manipulação de arquivos e strings.

### **Estrutura data\_hora:**

- char dia: Representando o dia da atualização (1-31).
- char mes: Representando o mês da atualização (1-12).
- short ano: Representando o ano da atualização.
- char seg: Representando o segundo da atualização (0-59).
- char min: Representando o minuto da atualização (0-59).
- char hora: Representando a hora da atualização (0-23).

### **Estrutura artigo:**

- int id: Identificador único do arquivo.
- char titulo[301]: Título do arquivo com espaço para 300 caracteres + '\0'

- int ano: Ano da publicação do artigo.
- char autores[151]: Lista de autores do artigo com espaço para 150 caracteres + '\0'.
- int citacoes: Número de vezes que o artigo foi citado.
- data\_hora atualizacao: Data e hora da última atualização do artigo.
- char resumo[1025]: Resumo do artigo em até 1024 caracteres + '\0'.

### **Função: copia**

**Descrição:** Realiza a cópia controlada de caracteres de uma string de origem para uma string de destino.

#### **Parâmetros:**

char\* dest: Ponteiro para a string de destino onde os caracteres serão copiados.

const char\* src: Ponteiro para a string fonte da qual os caracteres serão copiados.

int tam: O número de caracteres a serem copiados.

- Retorno: void

### **Função: pega\_conteudo**

**Descrição:** Extrai o conteúdo de uma coluna delimitada por ';' de uma string de entrada e o armazena em outra string.

#### **Parâmetros:**

const char \*comeco\_coluna: Ponteiro para o início da coluna no registro CSV.

char\* dest: Ponteiro para a string de destino onde o conteúdo da coluna será copiado.

- Retorno: int: Retorna a posição logo após o fim do conteúdo extraído no registro original, permitindo a continuação da leitura do próximo campo.

### **Função: cvs\_para\_artigo**

**Descrição:** Converte uma linha de CSV em um objeto artigo, preenchendo os

campos da estrutura com os dados extraídos.

#### **Parâmetros:**

const char \*linha: Ponteiro para a string que representa uma linha inteira de um arquivo CSV contendo os dados do artigo.

- Retorno: artigo: Retorna um registro de artigo (artigo), que é uma estrutura contendo todos os campos extraídos e convertidos da linha do arquivo CSV.

#### **Função: imprime\_artigo**

**Descrição:** Imprime os detalhes de um objeto artigo na saída padrão.

#### **Parâmetros:**

artigo& a: Referência para a estrutura artigo que contém os dados do artigo a ser impresso.

- Retorno: void.

### **Constantes**

#### **Estrutura Geral**

O arquivo "constantes.hpp" contém definições de constantes utilizadas em diferentes partes do programa, proporcionando uma fácil manutenção e ajuste de parâmetros.

#### **Constantes:**

- tamanho\_bloco: Define o tamanho do bloco na máquina.
- numero\_buckets: Define o número de buckets para a estrutura de hash.
- PRIMARY\_INDEX\_DUMP: Identificador para a operação de despejo (dump) do índice primário.
- PRIMARY\_ORDER\_MAIN: Ordem principal para a estrutura de índice primário.
- nome\_arquivo\_dados: Caminho para o arquivo binário que armazena os dados.
- nome\_arquivo\_hash: Caminho para o arquivo binário que armazena a estrutura de hash.

O arquivo "constantes.hpp" centraliza essas definições para facilitar futuras modificações e ajustes no programa. Utilizar constantes nomeadas proporciona um

código mais legível e permite uma gestão mais eficiente de parâmetros cruciais.

### **Seek1**

Não funcional

### **Seek2**

Não funcional