Trabalho 2 BD1 2022/2

Integrantes: Renato Lousan da Silva

David Pereira Bessa

Prefácio

Este trabalho visa em exercitar práticas de organização e indexação de arquivos através da construção de um repositório de dados, fundamentado em uma entrada massiva de dados unitários. A concretização desses dados ou registros será realizada singularmente, adentrando no arquivo organizado via função hash. Paralelamente à constituição do arquivo de dados, a localização de cada registro será utilizada para a instauração de uma árvore B+ que servirá para indexação primária. A prospecção de dados poderá ser realizada por meio de técnicas de indexação sobre um arquivo de índices primários ou secundários, correspondendo respectivamente a uma árvore B+ de índice primário e uma de índice secundário, ou via função hash. A finalidade deste documento é de apresentar uma descrição sintetizada dos dados, exibir a estrutura do arquivo de dados organizado por hash e dos arquivos de índices, informar a organização do código-fonte, elucidar as funções mais relevantes e explanar a distribuição das tarefas entre os membros da equipe.

Códigos-fonte e compilação do trabalho

Os códigos estão distribuídos entre bibliotecas no diretório 'cabecalhos' e o código-fonte no diretório 'codigosFonte'. Para compilar o trabalho, utilizamos um ambiente GNU/Linux com o compilador do comando g++. As instruções de como a compilação deve ser feita está especificada no arquivo README.md no repositório do trabalho.

Organização dos dados de entrada (entradas nas estruturas)

O arquivo de entrada foi dado pela especificação do trabalho (artigo.csv). Esse arquivo possui várias entradas (registros) separados por quebra de linha (\n) nos quais os campos dos registros são separados por ponto e vírgula (;). Os campos mencionados são: ID, Título, Ano, Autores, citações, atualização e snippet.

Foi encontrado na confecção deste trabalho entradas que não seguem à risca a estrutura supracitada. Por exemplo, alguns campos apresentam um campo nulo (NULL).

Arquivo de dados

O arquivo de dados é subdividido em 51.200 buckets, referenciados pelo hash, nos quais cada bucket contém 2 blocos de 4096 bytes. Cada bloco comporta um vetor de registros de tamanho fixo que são inseridos de maneira contígua. Além disso, cada bloco detém um campo que guarda a quantidade de registros inseridos e outro que armazena a localização de um registro, caso exista, que pertença ao mesmo bucket no arquivo de overflow.

Hash de overflow

Um dos documentos deste trabalho é estruturado por um hash de overflow, portanto, é relevante detalhar a estrutura do arquivo de registros organizados por buckets (referenciado como arquivo de dados) e a estrutura do arquivo que acomodará eventuais registros cujos buckets excederam a quantidade de registros (referenciado como arquivo de overflow). A estrutura que armazenará um registro qualquer engloba apenas os campos necessários para representar os dados expostos na seção 2 e será utilizada constantemente por outras estruturas e funções.

Arquivo de Overflow

O arquivo de overflow é constituído por registros distintos, denominados registros de overflow. Cada registro de overflow é uma estrutura que compreende um único registro com os dados desejados e um campo para armazenar a localização do registro subsequente, resultando em uma cadeia de registros, se houver algum registro a ser encadeado.

Arquivos de índice (primário e secundário)

As estruturas das duas árvores são análogas, no entanto, as chaves diferem, assim são gerados dois arquivos, o arquivo de índice primário para o campo id e o arquivo de índice secundário para o campo título.

Arquivo de índice primário

Os primeiros 4096 bytes são destinados ao cabeçalho que contém a altura da árvore e o ponteiro para a raiz. Os demais blocos representam os nós, externos ou internos, da árvore B+ que são estruturas que possuem um campo que indica a quantidade de registros, um vetor de pares de ponteiro e chave e um último ponteiro, já que o número de ponteiros é o número de chaves somado a um. Segue abaixo uma ilustração aproximada da estrutura do arquivo.

Arquivo de índice secundário

O arquivo de índice secundário segue a mesma estrutura do arquivo de índice primário. A única discrepância reside no valor das chaves, que é o campo título e não o campo id.

Funções Principais

achaEntradaArqDados(int idEntrada, int arqSaida, int arqOverflow)

A função requer apenas o valor do id de um registro fornecido pelo usuário para que ocorra a busca do registro desejado no arquivo de dados ou, eventualmente, nó de overflow. A busca se inicia com a leitura do primeiro bloco do bucket de endereço retornado pela função hash; com o bloco em memória, a busca pelo id será

sequencial. A busca continua para o segundo e último bloco do bucket, caso o registro não seja encontrado no primeiro bloco, sendo então necessário outra carga do bloco para a memória para buscar sequencialmente.

Se o registro não for localizado no segundo bloco, verifica-se o valor do campo que armazena o endereço do próximo registro do bucket no arquivo de overflow. Se o valor for válido, haverá uma busca sequencial no arquivo de overflow na qual serão carregados apenas registros, de forma unitária, e não blocos.

uploadArquivo(string caminhoArquivo)

A função recebe como parâmetro o caminho do arquivo de entrada. A partir disso, são criados arquivos de dados, de overflow, de índice primário e de índice secundário. Cada registro do arquivo de entrada será lido, formatado e acondicionado na estrutura de registro supracitada para que possa ser inserido no arquivo de dados organizado pelo hash.

A função de inserção no hash guardará o registro no arquivo de dados se possível, mas caso um bucket esteja cheio, ocorrerá a gravação deste registro no arquivo de overflow. Toda tentativa de inserção do registro no bloco verificará primeiramente a quantidade de registros no bloco; se o valor for equivalente ao fator de bloco, a função tentará inserir no próximo bloco do bucket até que seja necessária a escrita no arquivo de overflow.

Esta função retornará o endereço do registro e indicará em qual arquivo este registro se encontra, se está no arquivo de dados ou de overflow, tais informações servirão como parâmetro para a criação da B+ de índice primário.

Desenvolvedores do trabalho

- Renato Lousan da Silva:
  - Implementação da B+ Tree
  - Implementação de funções auxiliares
- David Pereira Bessa:
  - Redação e revisão do relatório
  - Implementação da hash e outras funções
  - Script do makefile