Banco de Dados: Trabalho Prático II - Indexação e Organização de Arquivos

Gabriel da Silva Freitas, Gabriel Luciano Nunes, Guilherme Silveira Duarte

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. Gen. Rodrigo Octávio 6200, Coroado – 69080-900 – Manaus – AM – Brasil

{gabriel.freitas,gabriel.luciano,guilherme.silveira}@icomp.ufam

1. Introdução

O objetivo deste trabalho é exercitar os conceitos sobre indexação e organizaçã de arquivos. Para isso, tratamos da criação de programas que realizam a indexação de um arquivo - **artigo.csv** - que contém informações acerca de publicações de artigos. Nesta documentação serão explicitadas a estrutura dos programas, bem como as funções que os compõem. Dentre os programas solicitados na descrição do trabalho, foram implementados somente os dois primeiros: **upload**, **findrec**.

2. Indexação

Para realização do trabalho antes de tudo, foi necessário estruturar o problema com embasamento teórico, sendo assim para a realização dos dois programas, foi necessário conhecer o funcionamento de uma tabela hash para a indexação de arquivos, e como esta abordagem seria adequada ao problema e então implementada. Segundo [Molina et al. 2008]:

> *"Um índice de agrupamento também é um arquivo ordenado com dois campos; o primeiro campo é do mesmo tipo que o campo de agrupamento do arquivo de dados e o segundo campo é um ponteiro de bloco de disco. Há uma entrada no índice de agrupamento para cada valor distinto do campo de agrupamento e contém o valor e um ponteiro para o primeiro bloco no arquivo de dados que possui um registro com esse valor para seu campo de agrupamento."*

Nesta implementação foi utilizado o conceito de indexação por agrupamento ou *clustering index*, em que há um dois campos na indexação: um com o índice do arquivo de dados e o outro com um ponteiro para o disco, como ilustra a figura 1:

3. Estrutura dos Programas

O programa **uppload.cpp** é o programa responsável pela leitura do arquivo, povoamento do banco de dados e armazenamento do arquivo de dados em uma tabela hash, o arquivo de saída é o arquivo binário: **arquivo_dados.bin**, o qual posteriormente será usado pelo

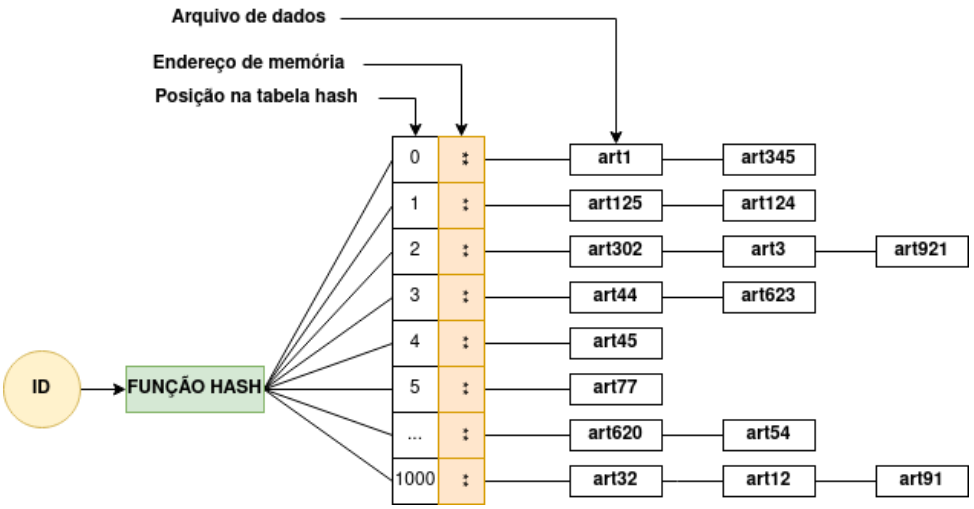

Figura 1. Diagrama representativo da tabela Hash

programa **findrec.cpp**, esse programa é responsável pela busca do ID digitado pelo usuário, essa busca é realizada acessando diretamente a tabela hash armazenada em memória secundária.

4. Biblioteca util.hpp

- Desenvolvido por Gabriel Freitas, Gabriel Luciano e Guilherme Silveira

Esta biblioteca é o cabeçalho dos dois programas, nesta seção vale destacar o uso da biblioteca **regex**, essa biblioteca foi utilizada para a leitura do arquivo de entrada, principalmente para lidar com problemas de formatação dos campos do arquivo de entrada.

Definimos nessa os tipos de dados que foram usados nos programas. A `struct Registro_Arquivo` que será usada para armazenar os dados na tabela hash, e a tabela hash, no qual já é criada e setada para `NULL`.

5. Funções de upload.cpp

- Desenvolvido por Gabriel Freitas e Gabriel Luciano

Este script de upload recebe o nome do arquivo entrada - **artigo.csv** - pela linha de comando e, através de suas funções, separa os campos de cada linha de **artigo.csv** e salva em uma "*instância*"da struct `Registro_Arquivo`. A partir desta etapa, ele insere esse dado na tabela hash.

A saída este programa será o arquivo de dados em formato binário. Esse arquivo será para realizar a busca pelo script **findrec.cpp**.

Para a compilação deste programa usamos o seguinte comando:

```
$ g++ upload.cpp -o upload -O2
```

E para rodar o programa utilizamos:

```
$ ./upload ../diretorio_se_houver/artigo.csv
```

Abaixo segue as funções criadas e uma breve descrição:

- **tabela_hash_t tabela_hash_criar**: Aloca espaço para criação de uma tabela hash;
- **bool posicao_tabela_hash_adicionar**: Insere os itens na posição da tabela Hash, possui como parâmetro um ponteiro para o registro do arquivo e outro para a posição na tabela hash;
- **int tabela_hash_funcao**: Função utiliza o ID para realizar o cálculo da hash, recebe como parâmetro um ponteiro para o registro de arquivo;
- **bool escreve_no_arquivo**: Função que escreve os índices e o endereço do registro no arquivo de dados;
- **int tabela_hash_adicionar**: Adiciona os registros na tabela hash;
- **void Posicao_tabela_hash_escreve_no_final**: Escreve os registros de posição na tabela hash no arquivo de saída;
- **void tabela_hash_escreve_final**:
- **vector<string> aplica_verificacao_campos_nulos**: Essa é uma função que essencialmente trata de valores vazios na leitura do arquivo de entrada, trocando-os por valores nulos;
- **void le_arquivo**: Lê os arquivos de entrada e os armazena em um buffer, utiliza das funções citadas acima para adicionar na tabela hash.

6. Funções de findrec.cpp

- Desenvolvido por Guilherme Silveira

Realiza a busca no arquivo de dados e imprime na tela o valor do resultado da busca.

Para a compilação deste script usamos o seguinte comando:

```
$ g++ findrec.cpp -o findrec -O2
```

E para rodar o programa utilizamos:

```
$ ./upload ../diretorio_se_houver/arquivo_dados.bin
```

Abaixo segue as funções criadas e uma breve descrição:

- **tabela_hash_funcao:** Calcula o índice da tabela hash para o ID solicitado na entrada. A posição é enviada para a variável ID na main para realizar a consulta.
- **main** Recebe o nome do arquivo de dados como parâmetro pela linha de comando. Armazena o nome em uma string e abre o arquivo na variável `entrada_hash`.

Referências

Molina, H., Ullman, J., and Widom, J. (2008). *Database Systems The Complete Book*. Prentice Hall, 2th edition.