Operações em arquivos sequenciais com arquivo de índices, ordenação externa e lista invertida

Milena Soares Barreira[1], **Thayris Gabriela Ferreira Rodrigues**[1]

[1]

Resumo. *Este documento descreve o a confecção do segundo trabalho realizado pela disciplina de Algoritmos e Estruturas de dados III. Tendo como principal objetivo descrever o processo criação de um arquivo de índices, contendo a posição e o id de uma conta bancária, implementação de uma ordenação exertena do arquivo utilizando intercalação balanceada e a criação de um arquivo contendo uma lista invertida, realizadas em uma conta de banco na linguagem Java.*

1. Introdução

Arquivos sequenciais são uma forma de se ordenar e organizar arquivos, possuindo as com menor complexidade implementações de operações básicas. As quatro operações principais em arquivos são create(inclui novos resgistros no arquivo principal), read(busca de um arquivo), delete(remoção) e uptade (alteração). No código apresentado nesse documento(testes e resultados) serão executadas essas operações com o objetivo de simular os processos realizados em uma conta bancária como criar uma conta, deletar ela, atualizar um registro, ler as informações dessa conta e por último realizar uma transferência para outra conta nesse banco. Nesse trabalho em específico, além de apresentado as operações, também será apresentada um arquivo de índices, a implementação de uma ordenação externa desse arquivo de índices realizado por uma intercalação balanceada de dois caminhos, e, por fim, a criação um arquivo contendo uma lista invertida.

2. Desenvolvimento

Nessa sessão foram trabalhados cada uma dessas funções ”CRUD”: create, read, uptade, a transferência e o delete respectivamente, além da explicação de arquivos de índices, ordenação externa, listas invertidas e o método de busca binária.

2.1. Create

O método create se baseia na criação de uma nova conta no sistema de bancos. Para isso ele necessita de informações do usuário como seu id, seu nome, seu CPF, sua cidade, a quantidade que ele deseja transferir e seu saldo, caso o cliente deseje transferir parte do seu dinheiro para outra conta. O id será a chave utilizada para localizar a conta desse usuário dentro do arquivo.

2.2. Read

O método read se baseia na leitura de uma conta existente no sistema de bancos. Ele funciona por meio do fornecimento do id do usuário que deve ser buscado retornando, assim o programa compara se algum registro possui esse id de um usuário específico, caso o id exista o programa imprime os dados do id informado, caso contrário são lidos novos registros até que se encontre o id especificado ou até o fim desse arquivo.

2.3. Update

O método uptade se baseia na alteração de um registro já criado, ou seja só será realizado caso exista a inserção de pelo menos uma conta. Ele funciona por meio da alteração de dados de um usuário, atualizando-o assim após esse processo de modificaçao.

2.4. Transferência

O método de transferência tem como objetivo realizar uma transmissão de saldos entre duas contas no arquivo. Para isso o usuário deve informar duas contas, uma a debitar e a outra a ser creditada, além do valor que deve ser transferido. Após esse processo as duas contas devem ter seus saldos atualizados, com redução e acréscimo em seu saldo, respectivamente.

2.5. Delete

O método delete se baseia na exclusão de um registro já criado. Para a realização desse método também é necessário o fornecimento do id, com a id o programa deve percorrer o arquivo e colocar uma lápide no registro que deve ser deletado. A lápide é uma marcação que tem como funcionalidade mostrar se o registro é válido ou se é um registro excluído, sinalizando a sua exclusão nesse caso.

2.6. Arquivo de índices

O arquivo de índices é um tipo de arquivo que fornece mecanismos para localizar registros rapidamente, evitando a reorganização do arquivo de dados, além de permitir o acesso a registros via chave sem precissar varrer o arquivo de dados. Esse arquivo é implementado tendo como par a sua chave, id, e a localização do registro, referente a esse id. O arquivo de índices feito nesse trabalho, recebe pela classe Cliente o idCliente, sendo incrementado ou excluído, de acordo com o arquivo principal, o arquivo de dados, além da posição desse registro, calculada pelo tamanho do arquivo(arq.length()). Esse arquivo de índices foi apresentado no setor ”arquivo.Ind.db”.

2.6.1. Busca Binária

A busca binária é um algoritmo eficiente para encontrar um item específico em uma lista ordenado, ele funciona pela divisão da lista pela metade repetidamente até reduzir o valor para o número procurado, para isso ela divide o arquivo da primeira posição até a última pela metade, após essa divisão se o número encontrado não for o procurado, o usuário deve ver se o encontrado é menor ou maior que o procurado, eliminando a maior metade se ele for menor e a menor metade se ele for maior, após isso o tamanho do vetor terá diminuído em 2 vezes, o usuário deverá repetir esse processo até encontrar o número de seu interesse. Nesse trabalho, para se realizar a busca binária, primeiro criamos um método nomeado BuscaBin que é responsavel por ler todo o arquivo de índices por meio de um arrayList de índices do Cliente. Nesse método, também, é chamado o método BuscaBinReneLógica, em que é realizado propriamente a busca binária, na qual se é passado o meio, o fim e o ínicio da lista, buscando, assim, analisar se o id passado como parâmetro está contido na lista e no arquivo de índices. Se esse id estiver contido, ele é retornado para a função BuscBinRene novamente, na qual se pega a posição desse id no

arquivo de dados. Por fim, a busca binária foi utilizado para substituir a busca sequencial nos métodos de read, update e delete, nas quais pela posição do id no arquivo de dados, conseguimos recuperar todos os dados da conta do Cliente, como id, cpf, nome e cidade. Teremos também um método chamado DeleteIndex, que chamara a função BuscBin que verá se o id procurado ainda está presente nesse ArrayList, se estiver ele deleta esse id e retorna a posição deletada.

2.7. Ordenação externa

A ordenação externa desse arquivo será implementada por meio de uma intercalação balanceada de dois caminhos, algoritmo de intercalação que opera com arquivos temporários, baseado na distribuição de blocos escritos entre arquivos temporários, sendo gerados vários segmentos ordenados dispostos nesses arquivos, de forma balanceada, ordenanando, assim, todos os dados do arquivo.

2.8. Lista invertida

Por fim, foi criado um arquivo contendo uma lista invertida, estrutura de dados que mapeia a quantidade de ocorrências de vários termos em um documento, usando a ocorrência desses termos para recuperar o ID das contas bancárias, permitindo buscas rápidas e precisas. Tendo como vantagens o fato de que pode crescer ou diminuir de acrodo com as alterações do arquivo, ocupando somente o espaço necessário, não necessitando ser alterada quando um novo registro é inserido. Para a implementação dessa lista, foram utilizados três métodos principais. O primeiro, foi o método ”addArqLista”, no qual se recebe como parâmetro o id do cliente e uma String, sendo ela sua cidade ou seu nome nesse método se pega todas as palavras da lista e procura se a palavra buscada está entre as já existentes. Para sua realização, o método pega as palavras escritas, nome e cidade, e as coloca num arquivo chamado listaInvertida, seguidos pelo id do arquivo(s) que contém essas strings. Se a palavra já existir se reescreve o arquivo colocando todos os ids que a contém, caso contrário se é criado um registro na lista para essa palavra. O segundo método é o ”buscaLista” responsável por pesquisar na lista esse nome ou cidade fornecido. Para isso, se procura entre todas as palavras do arquivo listaInvertida a string procurada, se ela existir se retorna a conta do cliente em que ela está inclusa, fornecendo o id do cliente, o nome, a cidade, o saldo de sua conta e a quantidade de transferências realizadas, respectivamente. Por fim, foi implementado o método ”deleteIdLista”, ele foi utilizado para mantér a coerência com o arquivo de dados, uma vez que se um registro for deletado, ele deverá também ser deletado do arquivo listaInvertida. Para isso, novamente deverá se ler todos os registros da lista, após isso quando se encontrado o registro que deve ser excluido, ele deve ser apagado permanente de todos os arquivos.

3. Testes e resultados

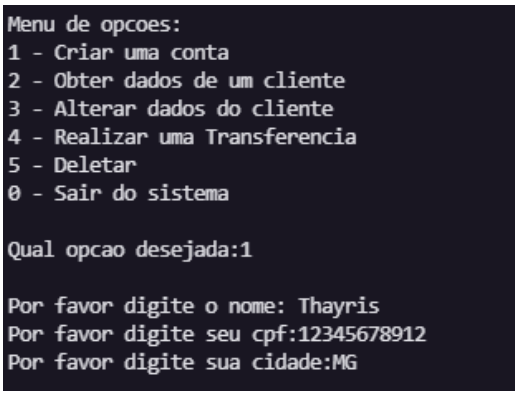

Figure 1. Metodo create

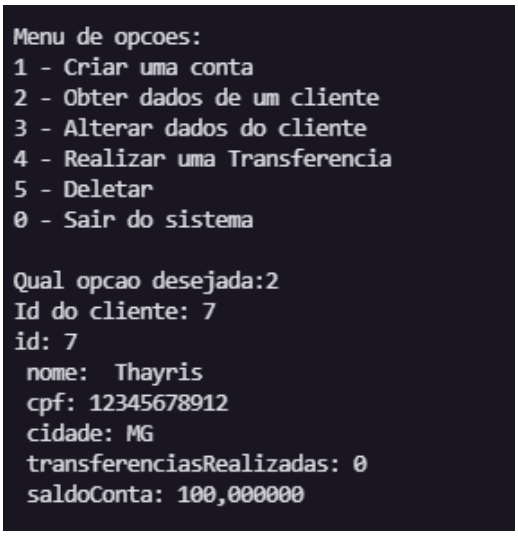

Figure 2. metodo read

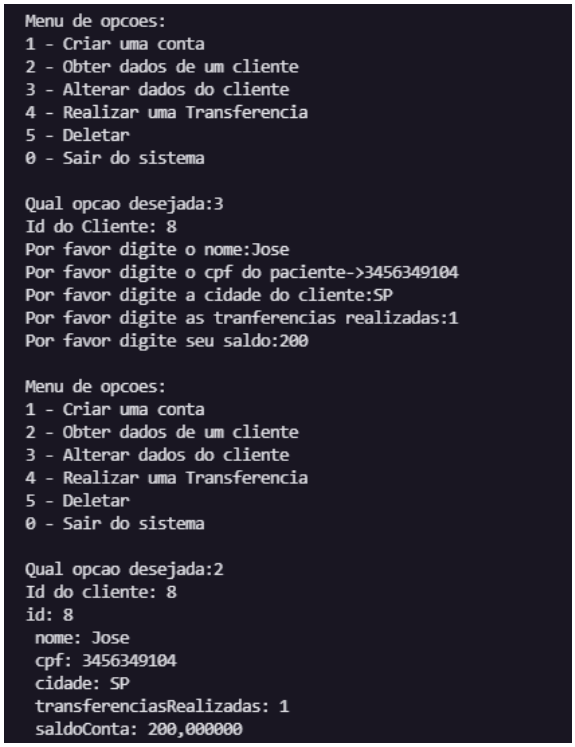

Figure 3. metodo update

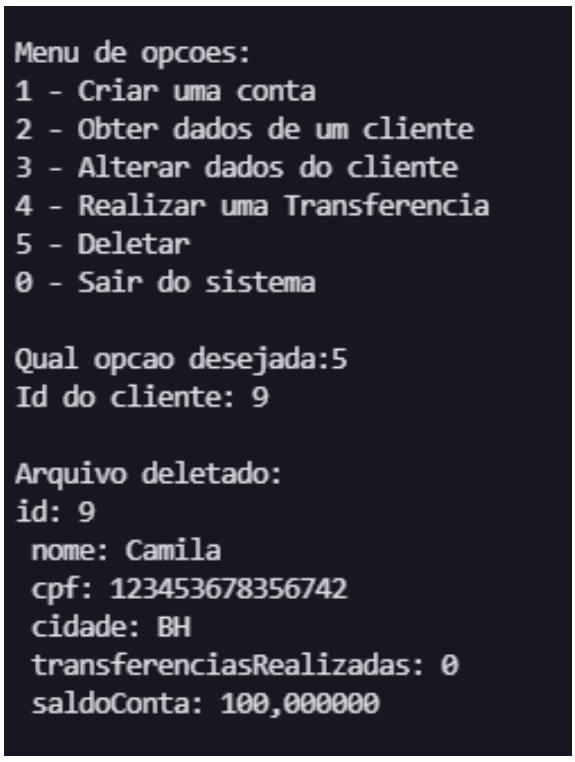

Figure 4. metodo delete

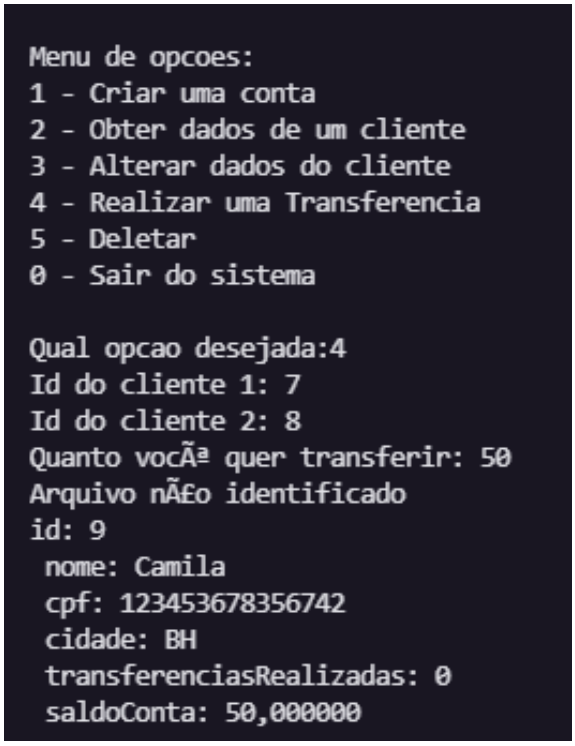

Figure 5. metodo transferencia

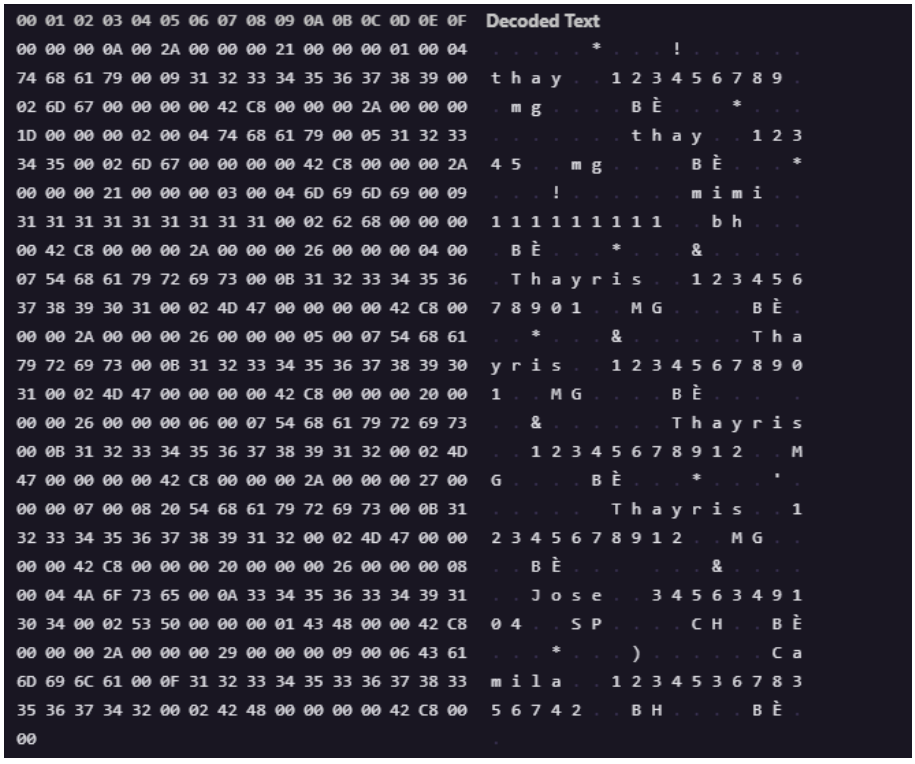

Figure 6. exemplo codigo funcionando

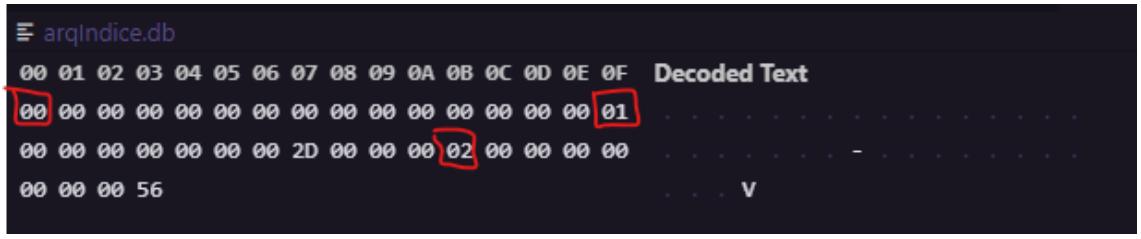

Figure 7. exemplo arquivo de índice

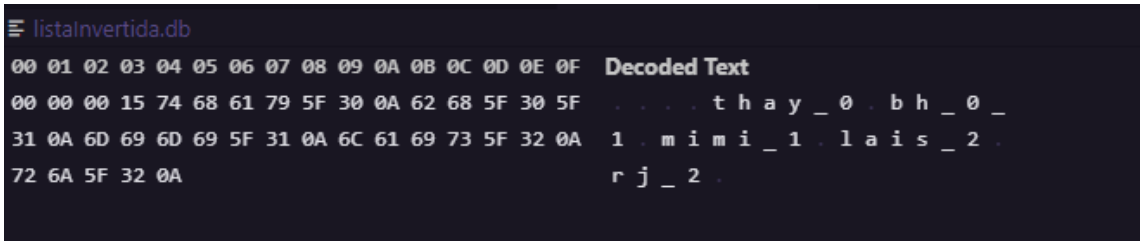

Figure 8. exemplo arquivo de lista invertida

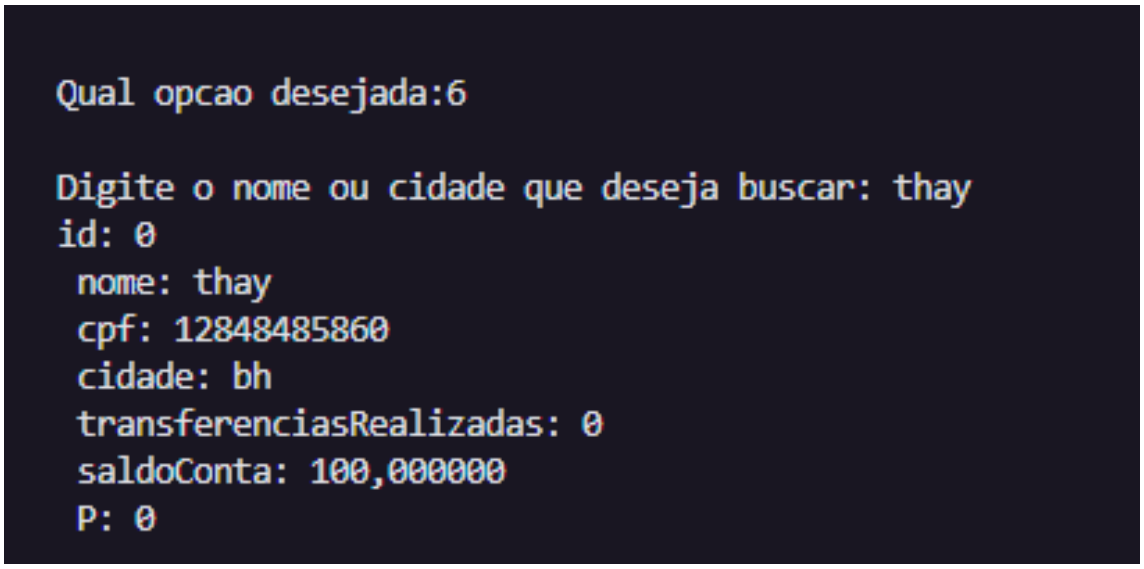

Figure 9. lista funcionando

4. Conclusão

Aprendemos com esse trabalho uma forma de salvar registros em determinado arquivos com o método ”CRUD”. Outro aprendizado foi a lápide, que não exclui um registro e sim altera o valor dele no arquivo sinalizando se foi deletado ou não, informação que tivemos depois de alguns testes. Portando, esse trabalho foi fundamental para aperfeiçoar nossas técnicas sobre arquivos sequencias e nos deixar mais próximas da forma em que as empresas trabalham para salvar dados de clientes.

Na segunda parte deste trabalho, aprendemos mais sobre busca binária, lista invertida e ordenação externa. Já havíamos visto busca binária em AEDS 2, mas aqui implementamos ela em um modelo real, com dados reais. Fizemos ela utilizando um array de

lista (que pega basicamente as informações contidas no nosso arquivo de índices). Já com a lista invertida ainda não havíamos trabalhando, dessa forma foi um processo um pouco mais demorado e complicado, que no final também deu certo e teve um resultado ótimo. E por fim nossa ordenação externa, nosso arquivo já estava ordenado pelo id, que sempre pegava do maior para o menor, então nossa ordenação externa foi um ademais para essa ordenação que já estava funcionando anteriormente.