Gabriel Cyrillo dos Santos Cerqueira - no.usp: 9763022
João Pedro Ramos Belmiro - no.usp: 9791198
Juliana de Mello Crivelli - no.usp: 8909303
Matheus Aparecido do Carmo Alves - no.usp: 9791114

Criação e Manipulação de Arquivos Indexados por Chaves Primárias

Universidade de São Paulo - USP
Instituto de Ciências Matemáticas e de Computação - ICMC
Bacharelado em Ciências de Computação

São Carlos
Junho de 2017

Lista de ilustrações

Sumário

1 Introdução

O presente trabalho integra a disciplina "*SCC0215 - Organização de Arquivos*"busca realizar uma aplicação direta do conteúdo apresentado durante todo o oferecimento deste curso, ministrado pela professora Dra. Cristina Aguiar Dutra Cifferi no primeiro semestre de 2017 [(1),(2)]. Seu objetivo principal é explorar a implementação de índices primários para indexação de arquivos de dados, comparando diferentes modos de reaproveitamento de espaço (*first-fit*, *best-fit* e *worst-fit*) e seu desempenho na remoção e inserção de novos dados em cada arquivo existente na estrutura.

Os dados utilizados são de domínio governamental (".gov.br"), disponíveis gratuitamente online (3). Isso, aproxima o trabalho da aplicação real, apresentando e requisitando uma implementação mais robusta que trate todos estes empecilhos comuns no dia a dia. Também é oferecido ao usuário a possibilidade de se manipular com inserções e remoções os registros da maneira desejada por meio de uma interface simples e intuitiva.

2 Organização e compilação do código

A implementação foi feita na Linguagem C respeitando a versão do GCC 4.8.2 ou superior para Linux.

O código está separado em duas pastas, *src* e *include*. Na primeira estão todos os arquivos fontes "**.c*", apresentando, de forma modularizada, cada tipo de organização e apresentação do programa e arquivos vinculados, funções e interface, além do arquivo principal "*main.c*".

Na segunda pasta constam os arquivos de cabeçalho "**.h*", apresentando a biblioteca de cada arquivo fonte, facilitando a leitura e utilização de cada implementação feita separadamente para a composição do trabalho como um todo.

Para compilar o código é possível utilizar as linhas de comando disponíveis no arquivo *Makefile*. Estão disponíveis comandos como: *make/make all* (que realiza a compilação geral do código); *make run* (para executação); e *make debug* (para compilar e executar o código com a apresentação dos traços de memória utilizados - *valgrind*).

3 Organização de campos, registros e índices

A base de dados utilizada para a realização dos testes possui registros com campos de tamanho fixo (*ticket*, *documento*, *dataHoraCadastro*, *dataHoraAtualiza*) e variável (*domínio*, *nome*, *cidade*, *uf*), portanto optou-se por utilizar a organização híbrida de campos e registros, na qual campos de tamanho fixo são apresentados precedendo os campos de tamanho variável no registro. Todos os registros estão separados por delimitadores - delimitador entre registros: carácter "#".

Optou-se também por salvar os campos *documento*, *dataHoraCadastro* e *dataHoraAtualiza* com 19 bytes, pois eles respeitam a máscara "###.###.###/###-##", "dd/mm/aaaa hh:mm:ss" e "dd/mm/aaaa hh:mm:ss", respectivamente, e todos podem ser lidos e armazenados de forma eficiente como um vetor de caracteres (*string* - soma de caracteres = 19). Mesmo sabendo que, em disco, isto pode ser uma desvantagem no acesso, devido ao caráter do trabalho proposto, optou-se por economizar memória ao invés de se realizar um tratamento para garantir melhor acesso a disco. Salvamos o campo *ticket* como um inteiro, o que também diminui o custo de memória na implementação.

Os campos de tamanho variável são salvos utilizando a técnica de recuperação por indicadores de tamanho que é, basicamente, um inteiro que precede o conteúdo do campo informando quantos bytes ele ocupa no documento.

Há 3 pares de arquivos, sendo um de índice e outro de dados para cada par.

O primeiro par adota a estratégia de reaproveitamento de espaço *first-fit*, o segundo *best-fit* e o terceiro, *worst-fit*. Todos os arquivos de dados são indexados por são chaves primárias baseadas no campo *ticket* e no byte *offset* do inicio do registro correspondente.

Os índices estão sempre ordenados (possibilitando uma busca binária no arquivo de índice) e a remoção de um registro implica na remoção lógica dentro arquivo de dados e física do arquivo de índice.

O conteúdo de cada índices é colocado em um vetor em memória primária (RAM) para facilitar a manipulação e diminuir o número de acessos a disco, tornando o sistema mais rápido e eficiente. Ao final da execução o arquivo de índice é reescrito.

4 Funcionalidades

4.1 Inserção

A inserção de um novo registro, o usuário, na realidade, o insere nos 3 arquivos de dados criados, para que possasse realizar uma comparação entre os método de reaproveitamento de espaço distintos implementados em cada arquivo ao final da execução do programa. Os detalhes desta aplicação está exposta na seção 5.

Após a inserção do novo registro nos arquivos de dados, a nova chave e o byte *offset* são inseridos no logicamente nos arquivos de índices correspondente, mantendo a ordenação.

4.2 Remoção

A remoção implementa uma busca binária no arquivo de índice de acordo com a chave especificada pelo usuário. Caso não seja encontrado o registro é mostrada uma mensagem de erro na tela. Se o registro existe ele é removido logicamente dos arquivo de dados e fisicamente dos arquivos de índice.

Na remoção lógica é feita a partir da inserção do carácter "*" no primeiro byte do registro, seguido por dois inteiros que indicam o tamanho total que era ocupado por aquele registro no documento e o byte *offset* do item anterior removido, implementando uma lista invertida para os registros removidos e possibilitando o reaproveitamento deste espaço. No caso do *best-fit* e do *worst-fit* a lista deve ser reordenada de modo que atenda os critérios do método de reaproveitamento. Na remoção física simplesmente são deslocados os registros do índice para cima, a partir do registro que se deseja remover, compactando assim o registro de índice.

4.3 Visualização de Estatísticas de arquivos de índices

Após a realização de diversas remoções e inserções nos arquivos de dados pelo usuário, espera-se que os arquivos de índice apresentem organizações diferentes devido às politicas distintas adotadas em cada implementação. Desta forma, a opção de se visualizar estatísticas que possibilitam a comparação dos desempenhos de cada politica é desejada. São apresentadas as quantidade de entradas em cada índice e os registros ainda existentes neles (um a um).

4.4 Visualização de Estatísticas de arquivos de dados

Após várias remoções e inserções, é esperado que as manipulações gerem fragmentação interna e externa diferentes para cada arquivo de índice, bem como uma diferente ordem dos registros armazenados. As estatísticas dos arquivos de dados mostram quantos registros estão presentes na lista de registros removidos (varia devido ao tratamento de fragmentação interna) e depois o usuário pode visualizar a lista invertida de cada um dos arquivos.

5 Estratégias de reaproveitamento de espaço

Para se garantir a gestão sobre os espaços livres gerados por remoções nos arquivos de dados, existe um cabeçalho contendo o topo de uma lista invertida logicamente implementada. A lista é composta pelos bytes *offset* dos registros removidos. O número "*-1*"no topo da lista é o sentinela para o final da lista.

Ao se reaproveitar o espaço de um registro removido pode-se gerar fragmentação interna, isto é, a quantidade de bytes do novo registro é menor que o espaço total que o registro anterior ocupava, então sobram bytes inúteis no documento. A fragmentação interna (como citado em capítulos anteriores) foi tratada de maneira alternativa em cada índice, mas para todos se alterou o indicador de tamanho do espaço livre do registro removido para a quantidade de bytes que sobraram após a inserção. O conteúdo do novo registro foi colocado de maneira deslocada, de modo que seu último byte coincida com o último byte do registro livre. Assim, o espaço residual é reinserido na lista invertida. Caso esse espaço seja menor que o tamanho mínimo de um registro (Tamanho dos campos fixos, mais indicadores de tamanho dos campos variáveis, mais o delimitador) gera-se fragmentação externa. Como não realizamos compactação do arquivo de dados, esta nunca é tratada. Quando não é possível reaproveitar o espaço de fragmentação interna, sendo pequeno demais para qualquer registro, é posto o carácter "$".

A inexistência de espaços livres por remoção ou de espaços grandes o suficiente para inserção, o novo registro é inserido ao final do arquivo de dados.

5.1 First-fit

Na inserção de dados utilizando a política *first-fit*, o sistema realizará uma consulta à lista remoções que dirá o *byte offset* do último registro removido. Caso o espaço indicado seja menor que o necessário para o novo registro, é examinado o próximo item da lista (*byte offset* do penúltimo registro removido), buscando encontrar um espaço suficiente para esta inserção.

5.2 Best-fit

Na abordagem pela política *best-fit*, busca-se o espaço que seja ao menos tão grande quanto o novo registro a ser inserido, porém tenta-se ocasionar a menor quantidade de bytes residuais possível. Para isso, utiliza-se a lista invertida ordenada de modo crescente de acordo com o tamanho de espaço livre. Este método busca diminuir a fragmentação interna, já que o local de inserção é o mais justo (tamanho do campo a ser inserido $\simeq$ tamanho do campo removido) possível em relação ao novo campo.

5.3 Worst-fit

A abordagem apresentada pelo *worst-fit* busca o espaço que seja ao menos tão grande quanto o novo registro a ser inserido, porém que ocasione na maior quantidade de bytes de sobra. Para isso a lista invertida está ordenada de modo decrescente de acordo com o tamanho de espaço livre. Este método busca diminuir a fragmentação externa, já que os bytes que sobram na inserção possuem maiores chances de reutilização em outra operação (tamanho do campo a ser inserido $<$ tamanho do campo removido), uma vez que o espaço residual é maior.

6 Interface: criação e como usar

A interface do programa foi desenvolvida pensando numa utilização intuitiva, fácil e rápida por parte do usuário. Desta forma, a primeira “tela” mostrada da ao usuário boas vindas e apresenta uma breve explicação sobre o objetivo da aplicação e como utiliza-la. A figura 1 mostra esta implementação. Para seguir no sistema, é necessário pressionar a tecla "*ENTER*".

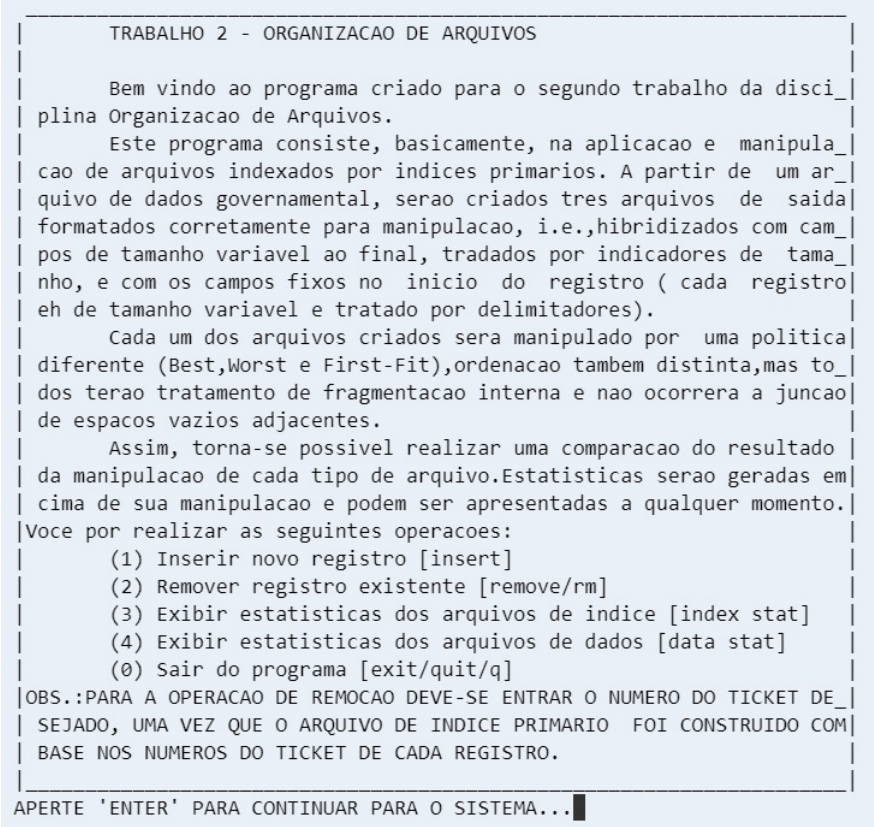

Figura 1 – Apresenta a Tela Inicial do programa desenvolvido.

Após a inicialização do sistema, para se realizar qualquer operação, o usuário deve digitar a código numérico ou a chave referente a cada opção de ação disponível. Exemplo: para se realizar uma remoção o usuário pode digitar "*2*", "*remove*" ou "*rm*". A figura 2 mostra o espaço reservado para entrada de comandos por parte do usuário.

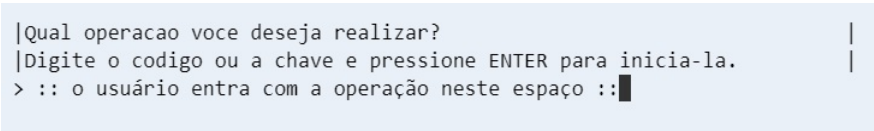

Figura 2 – Apresentação do espaço reservado para o usuário entrar com o código numérico ou chave da operação desejada.

Caso a operação digitada seja válida, o usuário é dirigido a uma tela referente à operação escolhida. Caso contrário uma mensagem de erro é exibida na tela e o sistema pede uma nova entrada. A figura 3 apresenta a situação descrita.

As telas correspondentes a cada operação disponível nesta implementação serão apresentadas no apêndice deste relatório.

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> operação invalida

|Operacao Invalida. As operacoes disponiveis sao:                       |
|       (1) Inserir novo registro [insert]                              |
|       (2) Remover registro existente [remove/rm]                      |
|       (3) Exibir estatisticas dos arquivos de indice [index stat]     |
|       (4) Exibir estatisticas dos arquivos de dados [data stat]       |
|       (0) Sair do programa [exit/quit/q]                              |
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> 
```

Figura 3 – Mensagem de erro apresentada na entrada de um comando inválido.

7 Bateria de testes

7.1 inserção

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> 1
|Digite os campos a serem inseridos no arquivo de dados:
|       Dominio: cprm-te.gov.br
|       Documento: 000.091.652/0005-32
|       Nome: COMPANHIA DE PESQUISA DE RECURSOS MINERAIS
|       UF: Paraná
|       Cidade: Belém
|       DataHoraCadastro: 18/01/2001 11:00:00
|       DataHoraAtualiza:
|       Ticket: 123
| Novo registro de ticket 123 inserido no arquivo 1
| Novo registro de ticket 123 inserido no arquivo 2
| Novo registro de ticket 123 inserido no arquivo 3
```

Figura 4 – Inserindo um novo registro nos arquivos.

7.2 remoção

```
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> 2
Digite o ticket do dominio que deseja remover: 5023
| Registro de chave 5023 removido com sucesso do arquivo 1.
| Registro de chave 5023 removido com sucesso do arquivo 2.
| Registro de chave 5023 removido com sucesso do arquivo 3.
```

Figura 5 – Removendo registro dos arquivos.

7.3 Estatísticas dos índices

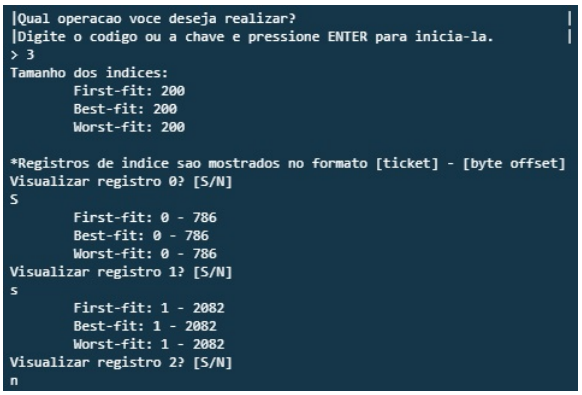

Figura 6 – Visualizando os Índices do arquivo.

7.4 Estatísticas das Remoções

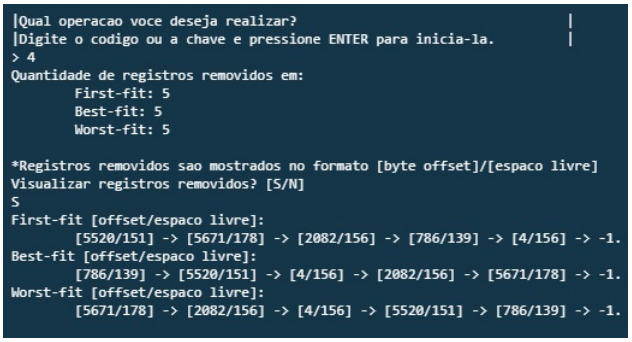

Figura 7 – Visualizando a estrutura da Lista invertida da Remoção.

8 Conclusões

8.1 Resultados

Com o desenvolvimento deste trabalho foi possível se realizar, de forma prática, muitas das teorias apresentadas em sala de aula. A utilização de índices primários para acesso mais rápido de arquivos, os tratamentos existentes para fragmentação interna, utilizando políticas de reaproveitamento como *first-fit*, *best-fit* e *worst-fit* e a manipulação de arquivos de dados são exemplos do conhecimento absorvido.

Ao final de toda a implementação, por meio dos testes e realizando uma analise a partir das funções de visualização, tornou-se possível concluir e observar como a opção por um método de reaproveitamento afeta a disposição dos registros e variam a porcentagem de fragmentação interna e externa gerada após a realização de múltiplas manipulações em arquivos de dados.

Não é possível dizer qual é a melhor estratégia a se adotar em um cenário geral somente observando os resultados deste trabalho. Acredita-se que cada cenário existente requer um método específico que beneficia em um ponto a aplicação desejada, refletindo-se mais como uma decisão de projeto que pode trazer suas vantagens e seus prejuízos de acordo com a finalidade.

8.2 Avaliação de participação

Nome	No. USP	Porcentagem de Participação
Gabriel Cyrillo dos Santos Cerqueira	9763022	25%
João Pedro Ramos Belmiro	9791198	25%
Juliana de Mello Crivelli	8909303	25%
Matheus Aparecido do Carmo Alves	9791114	25%

Tabela 1 – Apresentação da participação parcial de cada integrante do grupo (Grupo3). $P_{total} = 100\%$. Tabela elaborada pelo compilador.

Referências

1 COTEIAWIKI - Organização de Arquivos. 2017. <http://wiki.icmc.usp.br/index.php/SCC0215_012017(cdac)>. Acessado: 2017-06-22. Citado na página 1.

2 FOLK, M.; ZOELLICK, B.; RICCARDI, G. *File Structure an Object Oriented Approach with C++*. [S.l.]: Pearson Education, 2002. Citado na página 1.

3 DOMINIOS Gov.br. 2017. <http://dados.gov.br/dataset/dominios-gov-br>. Acessado: 2017-06-22. Citado na página 1.

A Telas da Interface

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> insert
|Digite os campos a serem inseridos no arquivo de dados:
|       Dominio: dominio-teste.gov.br
|       Documento: 000.000.000/0000-00
|       Nome: Dominio Teste
|       UF: São Paulo
|       Cidade: São Carlos
|       DataHoraCadastro: 00:00:00 01/01/2001
|       DataHoraAtualiza: 00:00:00 02/02/2002
|       Ticket: 5
| Novo registro de ticket 5 inserido no arquivo (1)
| Novo registro de ticket 5 inserido no arquivo (2)
| Novo registro de ticket 5 inserido no arquivo (3)
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
>
```

Figura 8 – Apresentação da interface de inserção implementada (Sucesso na inserção).

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> insert
|Digite os campos a serem inseridos no arquivo de dados:
|       Dominio: dominio-repetido.gov.br
|       Documento: 111.111.111/1111-11
|       Nome: Dominio Repetido
|       UF: São Paulo
|       Cidade: São Carlos
|       DataHoraCadastro: 11/11/2001 11:11:11
|       DataHoraAtualiza: 11/11/2011 12:12:12
|       Ticket: 1
| Inserção Falhou. Chave primaria ja existente no arquivo
| Inserção Falhou. Chave primaria ja existente no arquivo
| Inserção Falhou. Chave primaria ja existente no arquivo
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> 
```

Figura 9 – Apresentação da interface de inserção implementada (Falha na inserção - Chave Primária repetida).

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> remove
Digite o ticket do dominio que deseja remover: -1
Registro invalido.

|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> remove
Digite o ticket do dominio que deseja remover: 1
FF: -1
        Registro de chave 1 removido com sucesso do arquivo 1.
        Registro de chave 1 removido com sucesso do arquivo 2.
        Registro de chave 1 removido com sucesso do arquivo 3.

|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> 
```

Figura 10 – Apresentação da interface de remoção implementada. No primeiro caso, temos a tentativa de remoção de um registro por uma chave inexistente. Logo em seguida temos a remoção de um registro válido.

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> index stat
Tamanho dos indices:
        First-fit: 199
        Best-fit: 199
        Worst-fit: 199

*Registros de indice sao mostrados no formato [ticket] - [byte offset]
Visualizar registro 0? [S/N]
S
        First-fit: 0 - 786
        Best-fit: 0 - 786
        Worst-fit: 0 - 786
Visualizar registro 1? [S/N]
S
        First-fit: 2 - 8301
        Best-fit: 2 - 8301
        Worst-fit: 2 - 8301
Visualizar registro 2? [S/N]
N
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
>
```

Figura 11 – Apresentação da interface de visualização de estatísticas dos arquivos de índices implementada.

```
|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
> data stat
Quantidade de registros removidos em:
        First-fit: 4
        Best-fit: 4
        Worst-fit: 4

*Registros removidos sao mostrados no formato [byte offset]/[espaco livre]
Visualizar registros removidos? [S/N]
S
First-fit [offset/espaco livre]:
        [4/156] -> [10582/137] -> [8301/149] -> [2082/156] -> -1.
Best-fit [offset/espaco livre]:
        [10582/137] -> [8301/149] -> [2082/156] -> [4/156] -> -1.
Worst-fit [offset/espaco livre]:
        [4/156] -> [2082/156] -> [8301/149] -> [10582/137] -> -1.

|Qual operacao voce deseja realizar?                                    |
|Digite o codigo ou a chave e pressione ENTER para inicia-la.           |
>
```

Figura 12 – Apresentação da interface de visualização de estatísticas dos arquivos de dados implementada.