Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Disciplina de Organização de Arquivos
Profa. Dra. Cristina Dutra de Aguiar Ciferri
PAE Anderson Chaves Carniel (Turma A)
PAE João Pedro de Carvalho Castro (Turma B)

Segunda Parte do Trabalho Prático (Parte II)
Valor: 60%

Este trabalho tem como objetivo realizar o armazenamento de registros em arquivos e realizar a recuperação eficiente desses registros usando um índice árvore-B virtual, ou seja, um índice árvore-B estendido com uma técnica de gerenciamento de *buffer-pool*.

O trabalho deve ser feito em grupo de 4 alunos, que deve ser o mesmo grupo da primeira parte do trabalho prático. Qualquer alteração de grupo deve ser decidida pela docente da disciplina. A solução deve ser proposta exclusivamente pelo grupo com base nos conhecimentos adquiridos ao longo das aulas. Consulte as notas de aula e o livro texto quando necessário.

Descrição do índice

O índice árvore-B com ordem *m* é definido formalmente como descrito a seguir.

1. Cada página (ou nó) do índice árvore-B deve ser, pelo menos, da seguinte forma:

   $<P_1, <C_1,P_{R1}>, P_2, <C_2,P_{R2}>, ..., P_{q-1}, <C_{q-1},P_{Rq-1}>, P_q>$, onde $(q \leq m)$ e

   - Cada P_j $(1 \leq j \leq q)$ é um ponteiro para uma subárvore ou assume o valor nulo caso não exista subárvore (ou seja, caso seja um nó folha).
   - Cada C_i $(1 \leq i \leq q - 1)$ é uma chave de busca.
   - Cada P_{Ri} $(1 \leq i \leq q - 1)$ é um campo de referência para o registro no arquivo de dados que contém o registro de dados correspondente a C_i.

2. Dentro de cada página (ou seja, as chaves de busca são ordenadas)

   - $C_1 < C_2 < ... < C_{q-1}$.

3. Para todos os valores X da chave na subárvore apontada por P_i:
   - $C_{i-1} < X < C_i$ para $1 < i < q$
   - $X < K_i$ para $i = 1$
   - $K_{i-1} < X$ para i = q.
4. Cada página possui um máximo de m descendentes.
5. Cada página, exceto a raiz e as folhas, possui no mínimo $\lceil m/2 \rceil$ descendentes (*taxa de ocupação*).
6. A raiz possui pelo menos 2 descendentes, a menos que seja um nó folha.
7. Todas as folhas aparecem no mesmo nível.
8. Uma página não folha com k descendentes possui k-1 chaves.
9. Uma página folha possui no mínimo $\lceil m/2 \rceil - 1$ chaves e no máximo $m - 1$ chaves (*taxa de ocupação*).

Organização do registro de cabeçalho do índice. O registro de cabeçalho do arquivo do índice deve conter os seguintes campos:

- status: indica a consistência do arquivo de índice, pois devido à queda de energia, travamento do programa, etc. as atualizações armazenadas no *buffer* podem ser perdidas. O campos status pode assumir os valores 0 (falso) ou 1 (verdadeiro). O valor 0 indica que o arquivo de índice está inconsistente e o valor 1 indica que o arquivo de índice está consistente – tamanho: 1 *byte*
- noRaiz: armazena o RRN do nó (página) raiz do índice árvore-B – tamanho: 4 *bytes*
- altura: armazena a altura do índice árvore-B. A altura do índice é determinada pela altura de seu nó raiz. Nota: nós folha possuem altura 0 – tamanho: 4 *bytes*
- ultimoRRN: armazena o último RRN alocado pelo índice árvore-B. Ou seja, refere-se ao identificador do último nó (página) do índice árvore-B – tamanho: 4 *bytes*

Representação gráfica do registro de cabeçalho do índice. O tamanho do registro de cabeçalho deve ser de 13 bytes, representado da seguinte forma:
Tamanho do registro de tamanho fixo: 13 bytes.

1 byte	4 bytes	4 bytes	4 bytes
status	noRaiz	altura	ultimoRRN

Organização do arquivo de índice. Deve ser considerada a seguinte organização: campos de tamanho fixo e registros de tamanho fixo. A ordem da árvore-B é 10, ou seja, $m = 10$. Portanto, um nó (página) terá 9 chaves e 10 ponteiros. A chave de busca da turma A é o campo *codEscola*, enquanto que a chave de busca da turma B é o campo *codINEP*.

Representação gráfica de um nó (página) do índice. Cada nó terá 116 bytes. *n* é o número de chaves do nó, armazenado por um inteiro. Lembre-se: a quantidade de ponteiros válido de um nó é determinado pela quantidade de chaves de busca + 1, ou seja, $n + 1$.

4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	...	4 bytes	4 bytes	4 bytes
n	P_1	C_1	P_{R1}	P_2	C_2	P_{R2}	...	C_9	P_{R9}	P_{10}
0	1	...							114	115

Especificação do buffer-pool. A capacidade total do *buffer-pool* mantido em memória principal deve ser suficiente para armazenar 5 nós (páginas) do índice. O nó raiz deve obrigatoriamente ser armazenado no *buffer-pool*. O restante do espaço (ou seja, os 4 nós restantes) deve ser gerenciado por uma política de substituição própria. Cada grupo de cada turma deve implementar uma política de substituição de páginas a ser definida por sorteio em sala de aula. A política de substituição definida para cada grupo estará disponibilizada na página da disciplina.

Toda vez que o programa terminar de executar alguma funcionalidade especificada neste documento, o número de *page fault* e o número de *page hit* devem ser armazenados em um arquivo do tipo texto chamado "*buffer-info.text*". Esse arquivo sempre deve ser aberto com a opção *append* ("a"), ou seja, sempre para se

escrever no final dele. Cada linha desse arquivo corresponde ao número de *page fault* e *page hit* após a execução de uma funcionalidade e deve possuir o seguinte formato:
`Page fault: X; Page hit: Y.`
Onde *X* e *Y* são inteiros.

Note que, após o processamento de uma funcionalidade, o *buffer-pool* deve ficar vazio. Ou seja, o *buffer-pool* ficará sempre vazio quando o programa terminar sua execução.

Importante. Quando um nó (página) do índice tiver chaves de busca que não forem preenchidas, pode-se armazenar lixo de memória ou uma sequencia de zeros. O valor -1 pode ser usado para denotar que um ponteiro *P* de um nó é nulo. Não é necessário armazenar o tipo do nó (folha ou interno) pois isso pode ser verificado pela altura da árvore durante a execução dos algoritmos de busca, inserção e remoção.

Programa

Descrição Geral. Implemente um programa em C que estenda as funcionalidades oferecidas pela primeira parte do programa para incorporar um índice árvore-B virtual, ou seja, um índice árvore-B estendido com uma técnica de gerenciamento de buffer-pool. O programa deve oferecer uma interface, via linha de comando, por meio da qual o usuário possa realizar *inserção*, *remoção* e *atualização* de dados baseado na *abordagem dinâmica* de reaproveitamento de espaços de registros logicamente removidos. Deve-se levar em consideração a descrição e a organização do arquivo de dados especificados na primeira parte do trabalho prático, bem como a descrição e a organização do arquivo de índice especificados nessa segunda parte do trabalho prático. A definição da sintaxe de cada comando deve seguir estritamente as especificações definidas em cada funcionalidade.

Importante. A definição da sintaxe de cada comando bem como sua saída na saída padrão devem seguir estritamente as especificações definidas em cada funcionalidade. Para especificar a sintaxe de execução, considere que o programa seja chamado de

"programaTrab2" (entretanto, o grupo pode definir qualquer outro nome para o programa). Essas orientações devem ser seguidas uma vez que a correção do funcionamento do programa se dará de forma automática por meio de *scripts* de execução. De forma geral, o primeiro parâmetro a ser passado para o programa por meio da linha de comando é sempre o identificador de suas funcionalidades (ou seja, um inteiro de 10 a 9), conforme especificado a seguir.

Descrição específica. Estenda o programa da Parte I (ou seja, primeira parte do trabalho prático) para que ele também ofereça as funcionalidades descritas a seguir.

[10] Toda a vez que um registro for inserido no arquivo de dados, a sua entrada correspondente deve ser inserida no arquivo de índice árvore-B. Ou seja, ocorre a inserção registro a registro no arquivo de dados, e elemento a elemento no arquivo de índice. Essa funcionalidade deve ser acoplada à funcionalidade [1] da Parte I (ou seja, carga do arquivo .csv). A mesma sintaxe e o mesmo formato de saída da funcionalidade [1] da Parte I devem ser usados na funcionalidade [10].

[11] Toda a vez que um registro for inserido no arquivo de dados, a sua entrada correspondente deve ser inserida no arquivo de índice árvore-B. Ou seja, ocorre a inserção registro a registro no arquivo de dados, e elemento a elemento no arquivo de índice. Essa funcionalidade deve ser acoplada à funcionalidade [6] da Parte I (ou seja, inserção de um único registro). A mesma sintaxe e o mesmo formato de saída da funcionalidade [6] da Parte I devem ser usados na funcionalidade [11].

[12] Permita a recuperação de um registro do arquivo de dados com base em um valor do campo que é chave primária por meio do uso do índice árvore-B. A chave primária da turma A é o campo *codEscola* enquanto que a chave primária para a turma B é o campo *codINEP*.

Sintaxe do comando para a funcionalidade [12]:

```
./programaTrab2 12 chaveDeBusca
```

Saída caso o programa seja executado com sucesso:

O registro deve ser mostrado em uma única linha e os seus campos devem ser mostrados de forma sequencial, separados por espaço em branco. Para os campos com tamanho variável, mostre também seu tamanho em bytes.

Mensagem de saída caso não seja encontrado o registro que contém o valor do campo ou o campo pertence a um registro que esteja removido:

```
Registro inexistente.
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha no processamento do arquivo.
```

Exemplo de execução para a Turma A:

```
./programaTrab2 12 35000024
35000024 01/02/2012 21/12/2012 25 GAVIAO PEIXOTO BRIGADEIRO 9 SAO PAULO 11 RUA MOGEIRO
```

Exemplo de execução para a Turma B:

```
./programaTrab2 12 31031984
31031984 22/03/2011 MG 14 EE JOSE MANOEL 7 ARAUJOS  4 CTBC
```

[13] Permita a remoção de um registro do arquivo de dados com base em um valor do campo que é chave primária por meio do uso do índice árvore-B. A chave primária da turma A é o campo *codEscola* enquanto que a chave primária para a turma B é o campo *codINEP*. Mais especificamente, o índice deve ser usado para buscar o elemento a ser removido, o qual é então removido logicamente do arquivo de dados como feito pela funcionalidade [5] da Parte I e então removido fisicamente do arquivo de índice.

Sintaxe do comando para a funcionalidade [13]:

```
./programaTrab2 13 chaveDeBusca
```

Saída caso o programa seja executado com sucesso:

```
Registro removido com sucesso.
```

Mensagem de saída caso não seja encontrado o registro que contém o valor do campo ou o campo pertence a um registro que esteja removido:

```
Registro inexistente.
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha no processamento do arquivo.
```

Exemplo de execução:

```
./programaTrab2 13 35000024
Registro removido com sucesso.
```

[14] Permita a atualização de todos os campos de um registro do arquivo de dados com base em um valor do campo que é chave primária por meio do uso do índice árvore-B. A chave primária da turma A é o campo *codEscola* enquanto que a chave primária para a turma B é o campo *codINEP*. Mais especificamente, o índice deve ser usado para buscar o elemento a ser atualizado. Esse elemento deve ser atualizado primeiramente no arquivo de dados como feito pela funcionalidade [7] da Parte I. Na sequencia, deve ser verificada a necessidade de se alterar a chave de busca no índice árvore-B. Se houver necessidade de atualização da chave de busca, isso deve ser tratado como uma sequencia de remoção e inserção no arquivo de índice.

Sintaxe do comando para a funcionalidade [14]:

```
./programaTrab2 14 chaveDeBusca valorCampo1 valorCampo2 valorCampo3 valorCampo4 valorCampo5 valorCampo6
```

Observações: caso o valor de um campo seja uma *string*, ela deve estar entre aspas simples (por exemplo, 'SAO PAULO'). Caso o valor seja *null*, ele deve ser identificado como 0 para os campos de tamanho fixo e como '' para os campos de tamanho variável. Não é necessário realizar o tratamento de truncamento de dados. Portanto, a soma dos tamanhos para os campos de tamanho variável nunca deve ultrapassar o tamanho do registro de tamanho fixo.

Mensagem de saída caso o programa seja executado com sucesso:

```
Registro alterado com sucesso.
```

Mensagem de saída caso não seja encontrado o registro no arquivo de índice:

```
Registro inexistente.
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha no processamento do arquivo.
```

Exemplo de execução para a Turma A:

```
./programaTrab2 14 35000012 3500456012 0 0 'EE DISCIPLINA' 'SAO CARLOS' ''
Registro alterado com sucesso.
```

Exemplo de execução para a Turma B:

```
./programaTrab2 14 31031917 3103378917 18/01/2018 SP 'EE DISCIPLINA' 'RUA INPE' ''
Registro alterado com sucesso.
```

Restrições

As seguintes restrições têm que ser garantidas no desenvolvimento do trabalho.

[1] O arquivo de índice deve ser gravado em disco no **modo binário**. O modo texto não deve ser usado.

[2] O índice árvore-B virtual deve ser definido sobre o campo que é chave primária do arquivo de dados.

[3] Cada página (ou nó) do índice árvore-B virtual deve ser, pelo menos, da forma descrita no item **Representação gráfica de um nó (página) do índice.**

[4] O modo texto para gravar arquivos só é permitido para o armazenamento das informações relacionadas ao *buffer-pool*.

[5] Os integrantes do grupo devem constar como comentário no início do código (i.e. NUSP e nome de cada integrante do grupo). Não será atribuída nota ao aluno cujos dados não constarem no código fonte.

[6] Todo código fonte deve ser documentado. A **documentação interna** inclui, dentre outros, a documentação de procedimentos, de funções, de variáveis, de partes do código fonte que realizam tarefas específicas. Ou seja, o código fonte deve ser documentado tanto em nível de rotinas quanto em nível de variáveis e blocos funcionais.

[7] A interface deve obrigatoriamente ser via linha de comando, sendo que a sintaxe de cada comando deve seguir estritamente as especificações definidas em cada funcionalidade.

[8] A implementação deve ser realizada usando a linguagem de programação C. As funções das bibliotecas <stdio.h> devem ser utilizadas para operações relacionadas à escrita e leitura dos arquivos. Não deve ser usada bibliotecas de estrutura de dados, tais como aquelas do C++. A implementação não deve ser feita em qualquer outra linguagem de programação. O programa deverá compilar no GCC versão 4.8.2 ou superior.

[9] O programa deve ser acompanhado de uma **documentação externa** de, no máximo, 10 páginas. A documentação externa deve conter uma descrição em alto nível de cada algoritmo implementado para cada uma das funcionalidades [10] a [13]. Em detalhes, a documentação externa deve possuir:

- CAPA, com as seguintes informações: o nome da instituição, o nome do curso, o nome da disciplina, o nome do professor responsável, o nome do trabalho prático, o nome dos participantes e os respectivos números USP, e a data de entrega do trabalho prático.
- ÍNDICE, listando os nomes das seções que compõem o trabalho prático e as suas respectivas páginas de início.
- SEÇÃO 1: Deve-se explicar como o *buffer-pool* foi implementado. Ou seja, qual é o algoritmo de troca de páginas e quais estruturas de dados foram usadas e implementadas.
- SEÇÕES 2 a 5: Cada uma dessas seções deve conter um algoritmo descrito em alto nível que mostra o passo a passo realizado para implementar a funcionalidade relacionada. Devem ser incluídas quaisquer decisões de projeto. Ou seja, a documentação referente a essas seções deve conter a descrição dos principais conceitos usados no trabalho prático, incluindo desenhos que facilitem a compreensão das estruturas de dados, as decisões de projeto e as suas justificativas, assim como qualquer outra consideração adicional assumida no desenvolvimento do trabalho prático. Quaisquer decisões de projeto. Também devem ser especificados nessas seções os sistemas operacionais que foram usados e como o programa deve ser compilado e executado.
- REFERÊNCIAS BIBLIOGRÁFICAS, caso necessário.

Fundamentação Teórica

Conceitos e características dos diversos métodos para representar os conceitos de campo e de registro em um arquivo de dados podem ser encontrados nas transparências de sala de aula e também nas páginas 96 a 107 do livro *File Structures* (*second edition*), de Michael J. Folk e Bill Zoellick.

Conceitos e características do índice árvore-B virtual podem ser encontrados nas transparências de sala de aula e também nas páginas 333 a 404 do livro *File Structures* (*second edition*), de Michael J. Folk e Bill Zoellick.

Material para Entregar

Arquivo compactado. Deve ser preparado um arquivo .zip contendo:

- Código fonte do programa devidamente documentado.
- Makefile para a compilação do programa.
- Bibliotecas necessárias para a execução do programa.
- Documentação externa em formato .pdf.

Instruções de entrega. Enviar o arquivo compactado da seguinte forma:

- e-mail: labbdciferri@gmail.com
- assunto: [Organização de Arquivos] Trabalho Prático 2018; Turma X; Parte Y
- corpo da mensagem: deve constar no corpo da mensagem o NUSP e nome de cada integrante do grupo. Não será atribuída nota ao aluno cujos dados não constarem no corpo da mensagem.
- documento anexado: arquivo compactado no formato .zip

Critério de Correção

Critério de avaliação do trabalho. Na correção do trabalho, serão ponderados os seguintes aspectos.

- Corretude da execução do programa.
- Atendimento às especificações do registro de cabeçalho e dos registros de dados.
- Atendimento às especificações da sintaxe dos comandos de cada funcionalidade e do formato de saída da execução de cada funcionalidade.
- Qualidade da documentação (interna e externa) entregue.

Restrições adicionais sobre o critério de correção.

- A não execução de um programa devido a erros de compilação implica que a nota final da parte do trabalho será igual a zero (0).
- O não atendimento às especificações do registro de cabeçalho e dos registros de dados implica que haverá uma diminuição expressiva na nota do trabalho.
- O não atendimento às especificações de sintaxe dos comandos de cada funcionalidade e do formato de saída da execução de cada funcionalidade implica que haverá uma diminuição expressiva na nota do trabalho.
- A ausência da documentação interna implica que haverá uma diminuição expressiva na nota do trabalho.
- A ausência da documentação externa implica que haverá uma diminuição expressiva na nota do trabalho.
- A inserção de palavras ofensivas nos arquivos e em qualquer outro material entregue implica que a nota final da parte do trabalho será igual a zero (0).
- Em caso de cola, as notas dos trabalhos envolvidos será igual a zero (0).

Critério de avaliação dos integrantes. Podem ser incluídas uma ou mais perguntas a respeito do trabalho na prova.

Data de Entrega do Trabalho

Na data especificada na página da disciplina.

Bom Trabalho !