Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0201 – Introdução à Ciência da Computação II

Trabalho 3: Base de Dados

Professor: Dr. Rodrigo Fernandes de Mello (`mello@icmc.usp.br`)
Dr. Moacir Antonelli Ponti (`moacir@icmc.usp.br`)
Estagiário PAE: Felipe Simões Lage Gomes Duarte (`fgduarte@icmc.usp.br`)
Gabriel de Barros Paranhos da Costa (`gbpcosta@icmc.usp.br`)
Tiago Santana de Nazaré (`tiagosn@usp.br`)

1 Objetivo do Trabalho

Você deverá implementar um sistema capaz de ler e manipular um arquivo binário de dados cujo formato será descrito por um arquivo de schema.

2 Arquivos

O seu trabalho deverá ser capaz de criar/manipular três diferentes tipos de arquivos: arquivo schema (`.schema`) em formato texto, arquivo de dados (`.data`) em formato binário e arquivo de índice (`.idx`) em formato binário.

2.1 Arquivo Schema (`.schema`)

O arquivo `.schema` contém o formato dos dados que deverão ser processados pelo seu sistema. Ele está salvo em formato textual como o exemplo abaixo.

```
table iris\n
id int order\n
sepal_length double\n
sepal_width double\n
petal_length double\n
petal_width double\n
class char[50]\n
```

A primeira linha do arquivo schema define o nome do arquivo de dados (também denominado tabela), i.e., no exemplo anterior os dados estarão disponíveis no arquivo binário `iris.data`. Observe que todo arquivo de dados terá, portanto, extensão `.data` adicionada ao nome da tabela presente neste arquivo de schema.

As linhas seguintes descrevem os campos (atributos) que estão presentes no arquivo de dados. O número de atributos pode variar de acordo com a aplicação de interesse. O primeiro token contém o nome do campo e o segundo o tipo de dado (deve-se suportar somente `int`, `double` e `char[*]` em que `*` identifica qualquer número inteiro).

Opcionalmente um atributo pode ter o termo especial `order` que determina a criação de um arquivo binário de índice que estará ordenado por esse atributo. O arquivo binário de índice deverá ter um nome padrão no formato `<tabela>-<campo>.idx`, ou seja, no exemplo anterior seu sistema deverá criar um arquivo índice com o nome `iris-id.idx`.

2.2 Arquivo de Dados (`.data`)

O arquivo de dados (`.data`) é um arquivo binário que contém diversos registros. Cada registro segue o formato descrito pelo arquivo textual schema (`.schema`). Assim, seguindo o exemplo da tabela iris, o arquivo `iris.data` contém os registros como no exemplo abaixo:

```
134         6.3         2.8         5.1         1.5  virginica
1           5.1         3.5         1.4         0.2     setosa
80          5.7         2.6         3.5         1.0 versicolor
23          4.6         3.6         1.0         0.2     setosa
```

O arquivo de dados foi exemplificado em formato textual para facilitar a compreensão, mas de fato ele é armazenado de forma binária com cada tipo gravado em sequência, i.e., um após o outro. Note que cada registro terá 86 bytes ao todo $(4 + 8 + 8 + 8 + 8 + 50)$.

2.3 Arquivo Índice (`.idx`)

Este arquivo deverá conter todas as **chaves** (relativas ao atributo daquele índice) mais o *offset*, que é a posição em bytes do registro no arquivo de dados (`.data`). A função `ftell` pode ser utilizada para saber o *offset* atual do arquivo. Entende-se como **chave** todo campo que foi especificado no arquivo schema (`.schema`) como `order`, i.e., no exemplo anterior o campo `id` foi marcado como `order` e, por este motivo, um arquivo de índice deverá ser criado indexando os registros por meio deste campo.

Dado o exemplo anterior, o arquivo `iris-id.idx`, que também deverá ser binário, teria o seguinte conteúdo:

```
1       86
23      258
80      172
134     0
```

Novamente o arquivo foi exemplificado em formato textual para facilitar a compreensão, de fato ele é binário com cada tipo gravado em sequência, i.e., um após o outro.

No exemplo, a primeira linha contém o valor 1 que representa o atributo `id`. O valor 86 é o *offset*, a posição em bytes do início deste registro dentro do arquivo de dados `.data`. De maneira análoga, as demais linhas devem conter a chave que está sendo indexada e a posição (offset) do registro dentro do arquivo de dados. Note que 86 é exatamente a posição do segundo registro no arquivo de dados, visto que o primeiro registro ocupa os bytes de 0 a 85.

Como é possível observar, o arquivo de índice deve ser ordenado pela chave que está sendo usada para indexar os registros. Esta ordenação pode ser feita em memória principal, i.e., o arquivo de índice será salvo no disco já com os registros ordenados.

3 Proposta

Você deverá desenvolver um sistema capaz de efetuar busca e inserção no arquivo binário de dados. Para isto, ao iniciar, seu programa deverá ler o nome do arquivo que contém o schema dos dados. Para facilitar, o nome não terá mais que 30 caracteres e não conterá espaços. Este arquivo textual deverá ser lido pelo seu sistema e interpretado de acordo com a seção 2.1.

Após o processamento do schema, o seu programa deverá ler todo o arquivo binário de dados (`.data`) que tem o nome de acordo com o nome da tabela contida no arquivo schema, i.e., se o arquivo schema descrever a tabela iris, o arquivo binário de dados se chama `iris.data`.

Após a leitura dos dados, os arquivo(s) binário(s) de índice(s) devem ser gerados, 1 para cada atributo com qualificador `order`, i.e., se no schema dois atributos possuírem o qualificador `order`

dois arquivos de índice deverão ser criados (não se esqueça, ambos arquivos de índice devem ser binários). Em particular, em nosso exemplo, o arquivo `iris-id.idx` deverá ser criado indexando os ids dos registros. Os índices devem ser criados tal como explicado na seção 2.3.

Por fim, após leitura e processamento de todos os arquivos, o seu programa deverá executar os comandos abaixo, que serão fornecidos via `stdin`, até o comando de término. Os comandos textuais que seu programa deve obedecer são:

1. `exit`: este comando deverá liberar toda e qualquer memória alocada previamente, fechar possíveis arquivos abertos e por fim finalizar a execução do sistema;
2. `dump_schema`: este comando deverá fazer o dump do arquivo textual schema no `stdout` tal como especificado na seção 4;
3. `dump_data`: este comando deverá fazer o dump do arquivo binário data no `stdout` tal como especificado na seção 5;
4. `dump_index`: este comando deverá fazer o dump do arquivo binário index no `stdout` tal como especificado na seção 6;
5. `insert`: este comando deverá inserir um novo registro no arquivo binário de dados `.data`. Maiores detalhes na seção 7;
6. `select`: este comando deverá efetuar uma busca e exibir o registro encontrado bem como o número de iterações necessárias para encontrá-lo. Maiores detalhes na seção 8.
7. `update_index`: este comando deverá atualizar todos os arquivos de índice tal como especificado na seção 9;

4 Dump Schema

Este comando tem como objetivo garantir que o seu sistema leu e processou corretamente o arquivo textual `*.schema`. Para isso, ele deve imprimir no `stdout` o conteúdo do arquivo schema da seguinte forma:

```
table <nome_tabela>(<Tamanho_registro> bytes)\n
<nome_variavel> <tipo_variavel> <order?>(<tamanho_variavel> bytes)\n
...
<nome_variavel> <tipo_variavel> <order?>(<tamanho_variavel> bytes)\n
```

A primeira linha deve conter o nome da tabela e o tamanho, em bytes, do registro descrito no arquivo. Em sequência seu sistema deve imprimir o nome da variável, o seu tipo, se é `order` ou não e o tamanho em bytes. Cada variável deverá ser impressa em um linha única e a ordem deve ser a mesma do arquivo schema. Em nosso exemplo, se usarmos o arquivo schema fornecido na seção 2.1, o output do comando `dump_schema` deverá ser:

```
table iris(86 bytes)\n
id int order(4 bytes)\n
sepal_length double(8 bytes)\n
sepal_width double(8 bytes)\n
petal_length double(8 bytes)\n
petal_width double(8 bytes)\n
class char[50](50 bytes)\n
```

5 Dump Data

Este comando tem como objetivo garantir que o seu sistema leu e processou corretamente o arquivo binário `.data`. Para isso, ele deve imprimir no `stdout` o conteúdo do arquivo data da seguinte forma:

```
<nome_variavel> = <tipo_variavel>\n
...
<nome_variavel> = <tipo_variavel>\n
```

Para cada registro dentro do arquivo de dados você deverá imprimir o nome da variável (que foi descrita no arquivo schema) e o seu valor. Em nosso exemplo, se usarmos o arquivo dados fornecido na seção 2.2, o output do comando `dump_data` deverá ser:

```
id = 134\n
sepal_length = 6.30\n
sepal_width = 2.80\n
petal_length = 5.10\n
petal_width = 1.50\n
class = virginica\n
id = 1\n
sepal_length = 5.10\n
sepal_width = 3.50\n
petal_length = 1.40\n
petal_width = 0.20\n
class = setosa\n
id = 80\n
sepal_length = 5.70\n
sepal_width = 2.60\n
petal_length = 3.50\n
petal_width = 1.00\n
class = versicolor\n
id = 23\n
sepal_length = 4.60\n
sepal_width = 3.60\n
petal_length = 1.00\n
petal_width = 0.20\n
class = setosa\n
```

Importante ressaltar que os registros devem ser impressos seguindo a mesma ordem que estão salvos no arquivo binário `.data`. Além disso, quando o valor a ser impresso for do tipo `double` ou `float`, o seu programa deverá utilizar 2 casas decimais de precisão.

6 Dump Index

Este comando tem como objetivo garantir que o seu sistema leu e processou corretamente o arquivo binário `.idx`. Para isso, ele deve imprimir no `stdout` o conteúdo do arquivo de índice da seguinte forma:

```
<valor_variavel_order> = <offset>\n
...
<valor_variavel_order> = <offset>\n
```

Em nosso exemplo, se usarmos o arquivo fornecido na seção 2.2, o output do comando `dump_index` deverá ser:

```
1 = 86\n
23 = 258\n
80 = 172\n
134 = 0\n
```

Em um cenário com dois ou mais arquivos de índices, o seu programa deverá imprimir todos eles. A sequência de impressão deve respeitar a mesma ordem das variáveis com o modificador `order` dentro do arquivo schema, i.e., se em nosso exemplo a variável `class` também fosse `order`, o seu programa deveria primeiro fazer o dump do arquivo de índice relacionado à variável `id` e, posteriormente, o dump do arquivo de índice relacionado à variável `class`.

7 Insert

O comando `insert` indica ao sistema que um novo registro deve ser inserido no final do arquivo de dados `.data`. Para isso o seu programa deverá ler todos os campos necessários para criar um novo registro. Lembre-se que os campos do registro e a ordem com que eles devem ser lidos são determinados pelo arquivo schema.

A adição de um novo registro no arquivo de dados torna os arquivos de índices desatualizados. Apesar disso os arquivos de índices **NÃO** deverão ser atualizados a menos que o comando `update_index` seja explicitamente invocado.

Um exemplo de entrada de um caso de teste em que o comando `insert` é utilizado é (observem que os dados para o comando `insert` serão textuais, uma vez que serão encaminhados via stdin):

```
iris.schema
insert
43
6.5
3.0
5.2
2.0
virginica
exit
```

8 Search

O comando `search` deverá efetuar uma busca nos dados por um registro específico. Para isso será informado ao sistema, logo após o comando `search`, por qual campo a busca deverá ser efetuada, qual o termo de busca e qual valor do registro deve ser impresso. Caso o campo de busca informado não seja do tipo `order` e consequentemente não possua índice, o sistema deverá informar a seguinte mensagem:

`index not found`

Ao contrário, caso o índice exista, o sistema deverá efetuar uma busca binária utilizando somente os arquivos de índices (`.idx`) que foram gerados pelo seu sistema (utilize as funções `fseek`, `ftell`, `rewind` entre outras, para manipular o ponteiro de leitura do arquivo). Não será aceito trabalho que carregar todos os dados do arquivo de índice em memória e execute a busca binária em memória, novamente, a busca deve ser feita única e exclusivamente no arquivo. Assim, um possível caso de teste é:

```
iris.schema
select
id
1
petal_length
exit
```

O seu programa deverá imprimir o número de passos necessários para encontrar o registro e o valor do campo de interesse. Por exemplo, suponhamos o arquivo inicial `iris.data` com o caso de entrada anterior, o seu sistema deverá imprimir o seguinte resultado:

```
2
1.4
```

Em que 2 é o número de iterações necessárias para encontrar o registro com `id` igual a 1 e 1.4 é o valor do campo `petal_lenght`.

8.1 Índice desatualizado

Um caso especial acontece quando os arquivos de índice estão desatualizados. Este cenário ocorre quando o sistema insere um novo registro no arquivo de dados (`.data`) e os índices não são imediatamente atualizados (pois não foi solicitada a execução do comando `update_index`). Assim, para efetuar uma busca de um registro você deve seguir os seguintes passos:

1. Efetuar uma busca binária no arquivo de índice. Caso encontre, vá para o passo 3. Caso **não** encontre execute o passo 2.
2. Efetue uma busca sequencial no arquivo de dados (`.data`). Esta busca deve ser realizada somente nos registros que foram adicionados posteriormente e não estão indexados nos arquivos de índices.
3. Exiba o número total de passos (somando a busca binária e sequencial) e o valor do campo requerido.

9 Update Index

O comando `update_index` tem o propósito de atualizar todos os arquivos de índice. A ordenação dos dados pode ser feita em memória, ou seja, o seu programa pode atualizar todos os registros em memória e salvar no arquivo os dados já ordenados.

10 IMPORTANTE

- utilize alocação dinâmica (memória heap)
- não esqueça de liberar toda memória alocada antes de encerrar a execução do seu programa
- Crie quantas funções achar necessário, a modularização do sistema será levado em consideração no processo de correção.
- **PRAZO FINAL: 21/09 - 23:59:59**