Relatório do segundo trabalho de Algoritmo e estrutura de dados II

Nomes (Nº USP):

Leonardo Guarnieri de Bastiani (8910434)

Guilherme José (7150306)

Ricardo Chagas (8957242)

Modelagem

- **Arquivo principal e de apoio (índex primário e secundário)**

Para a organização do banco de dados foram utilizados cinco arquivos: o principal que guarda os registros, um arquivo de índice primário com a ID do usuário ligada ao byte offset do mesmo no arquivo principal; e três arquivos de índice secundário, um que guarda as idades de cada usuários, um que guarda os gêneros que os usuários curtem e o outro que guarda os tipos de usuários do site, sendo que os três ligam a informação aos IDs de cada usuário.

- **Organização dos arquivos**

Os registros, no arquivo principal, possuem tamanho e campo variáveis e o primeiro *byte* indica o tamanho do registro, pois assim, é possível ler toda sua informação num único acesso com exatidão do tamanho do registro.

Os campos são separados pelo caracter '|', já que ele não é válido para os campos em questão. Em especial o campo que guarda os gêneros, cujo tamanho varia como os outros, cada *byte* deste campo corresponde a um gênero que é tabelado.

No arquivo de índice primário, o primeiro *byte* corresponde a uma flag de status para indicar se todos os arquivos de índices estão ordenados ou não, o qual vale 1 se está e 0 se não.

O arquivo de índice primário é organizado de tal forma que os registros gravados nele tenham um tamanho fixo, isso ajuda para a leitura. Esta informação é organizada de tal forma que idealmente a ID do usuário esteja ao ligada ao byte offset do mesmo, organização que se inicia após o primeiro byte do arquivo pela razão já citada.

Os arquivos de índices secundários, assim como o de índice primário guardam informações de tamanho fixo. O que guarda as idades, a idade de cada usuário ao lado de sua ID, o que guarda os gêneros, o código (um int) que representa o gênero musical para que assim ele possa ser tratado como um número ao invés de uma string, deixando o programa mais eficiente e mais fácil de ser implementado, ao lado da ID da pessoa que o curte e o que guarda os tipos de usuários, o código do tipo ao lado da ID do usuário correspondente.

Como tanto a organização do arquivo de índice primário como os de secundários é feita de forma que eles guardam informações fixas, foi utilizada uma struct para escrevê-las em forma binária nos arquivos, sendo que depois é possível recuperar e ler toda sua informação num único acesso, o que é

conveniente já que são apenas arquivos de índice e, por isso, não ocupam muito espaço em RAM e, uma vez na memória, é possível realizar as operações desejadas com a informação de forma prática e eficiente.

Estratégias e algoritmos utilizados

O problema foi resolvido passando-se inicialmente toda informação considerada necessária para a memória RAM e utilizando-a para a resolução dos problemas, sendo que se for necessária posteriormente qualquer informação do arquivo principal, é feito apenas um acesso direto após a obtenção do byte offset do registro através da informação já em memória do arquivo de índice primário.

Foram carregados todos os índices e uma tabela de gêneros com o nome do gênero e um código associado. Os índices em RAM são úteis para um rápido acesso ao *byte offset* ou a algumas informações do registro sem o acesso a disco.

Adotamos uma tabela de gêneros que traduz uma string de gênero para um código. Isso foi útil para armazenarmos os índices secundários dos gêneros e para as comparações entre gêneros que o programa faz.

Obs.: Como o maior número em que se deseja eficiência no problema é de 100.000 registros, pode-se considerar que sempre é possível passar toda a informação indicada anteriormente para a memória RAM.

Explicação da lógica para a solução do problema

- Para todas as funcionalidades - complexidade de tempo e espaço

Considere que n o número de registros e que $\log(n) = \log_2(n)$.

As buscas realizadas nesse programa possuem dois casos: se os arquivos de índices estão ordenados, a busca é binária, portanto [O(log(n))]; mas se os arquivos de índice estão desordenados, a busca é sequencial, portanto [O(n)]. Para ordenar os índices, foi utilizado o quicksort da biblioteca C padrão e sua compelxidade é [O(n * log(n))], ele sempre ordena em relação ao ID do registro. Os índices sempre estam em memória, portanto não há acesso ao disco para ordenar e pesquisar, por isso o programa é muito eficiente.

O quicksort desse programa foi projetado para direcionar os IDs iguais a zero para o final do vetor, assim após cada ordenação, é feita uma compactação dos índices, que melhora a eficiência do programa.

Com relação à complexidade de espaço, os dados guardados em memória são sempre os arquivos de índice, portanto, é proporcial a n, ou seja, O(n).

- Funcionalidade 1

É inserido um novo registro no fim do arquivo principal e uma nova informação nos finais dos arquivos de índices e no final dos índices em memória RAM. Por fim, é atualizado o estado de ordenação para desordenado.

- Funcionalidade 2

A cada remoção, faz-se uma busca pelo ID que se deseja remover e é colocado o número 0 no lugar dos IDs correspondentes em todos os arquivos de índeces e em suas listas correspondentes em memória RAM. Já no arquivo principal, o primeiro caracter é substituido por '*', indicando que o registro foi apagado já que este não é um caractere válido para a situação. Neste caso, o custo computacional é uma busca e acessos a disco de O(1).

- Funcionalidade 3

Na pesquisa por ID, primeiramente verifica-se se o índice primário está ordenado. Caso não esteja, ele é ordenado antes da pesquisa em si. Posteriormente é feita uma busca binária nos dados em RAM.

- Funcionalidade 4
    - A.

O índice será ordenado, pois esta funcionalidade fará muitas buscas. É feita uma varredura de todos os registros. Obtém um vetor com os gêneros mais curtidos dos registros que se enquadram na especificação da busca. É feita uma ordenação que leva os gêneros mais curtidos para as primeiras posições do vetor.

    - B.

Assim como na funcionalidade anterior, o índice será ordenado. É feita uma varredura de todos os registros. Obtém um vetor com todos os registros que se enquadram na especificação da busca. É feita uma ordenação que leva os mais jovens para as primeiras posições do vetor.

- Funcionalidade 5
    - A.

Com uma busca no arquivo de índices secundários de idade, após os dados de índice secundário serem organizados, monta-se o conjunto das pessoas que se encontram na faixa etária desejada através de suas IDs. Então é montado um vetor em que cada posição do mesmo representa o código de um gênero e cada posição tem seu valor incrementado a cada vez em que ele é encontrado entre o gosto das pessoas do vetor anterior, informação que é obtida com uma busca em

RAM nos dados do arquivo de índice secundário que guarda os gêneros das IDs. Assim é montado o vetor com os até 10 gêneros mais curtidos na faixa etária dada, por meio de um algorítimo de ordenação que leva os 10 gêneros mais curtidos para o início do vetor.

- B.

O índice será ordenado e é feita uma varredura para todos os registros. Obtém um vetor com todos os usuários que se enquadram nas especificações da busca. Todos os resgastes de informação do registro já estão em RAM e possuem um acesso por busca binária, por isso o programa é eficiente.

Ilustrações do funcionamento da resolução do problema

Obs: nas ilustrações, o arquivo principal possui gêneros separados com arrobas, o que não acontece no programa de verdade, dessa forma, as ilustrações ficam mais didáticas.

Dados os seguintes arquivos iniciais:

Índice Secundário - Gênero	
ID	Gênero
1	Rock
1	MPB
4	Rock
4	Opera
3	Sertanejo
3	Grunge
2	Funk
2	Sertanejo

Índice Secundário - Idade	
ID	Idade
1	40
4	18
3	20
2	38

Índice Primário	
ID	Byte offset
1	0
4	21
3	48
2	81

Arquivo de registros
211|Ze|40|M|Rock@MPB|1|274|Felipe|18|M|Rock@Opera|1|333|Marina|20|F|Sertanejo@Grunge|1|292Maria|38|F|Funk@Sertanejo|2|

Índice Secundário – Tipo de Usuário	
ID	Tipo de usuário
1	1
4	1
3	1
2	2

Funcionamento da inserção (Insere-se o João -> ID = 5):

Índice Secundário - Gênero	
ID	Gênero
1	Rock
1	MPB
4	Rock
4	Opera
3	Sertanejo
3	Grunge
2	Funk
2	Sertanejo
5	Sertanejo

Índice Secundário - Idade	
ID	Idade
1	40
4	18
3	20
2	38
5	45

Índice Primário	
ID	Byte offset
1	0
4	21
3	48
2	81
5	110

Índice Secundário – Tipo de Usuário	
ID	Tipo de usuário
1	1
4	1
3	1
2	2
5	2

Arquivo de registros
211|Ze|40|M|Rock@MPB|1|274| Felipe|18|M|Rock@Opera|1|333| Marina|20|F|Sertanejo@Grunge|1 |292Maria|38|F|Funk@Sertanejo| 2|245|Joao|45|M|Sertanejo|2|

Funcionamento da Remoção (Remove-se a Maria -> ID = 2):

Índice Secundário - Gênero	
ID	Gênero
1	Rock
1	MPB
4	Rock
4	Opera
3	Sertanejo
3	Grunge
0	Funk
0	Sertanejo
5	Sertanejo

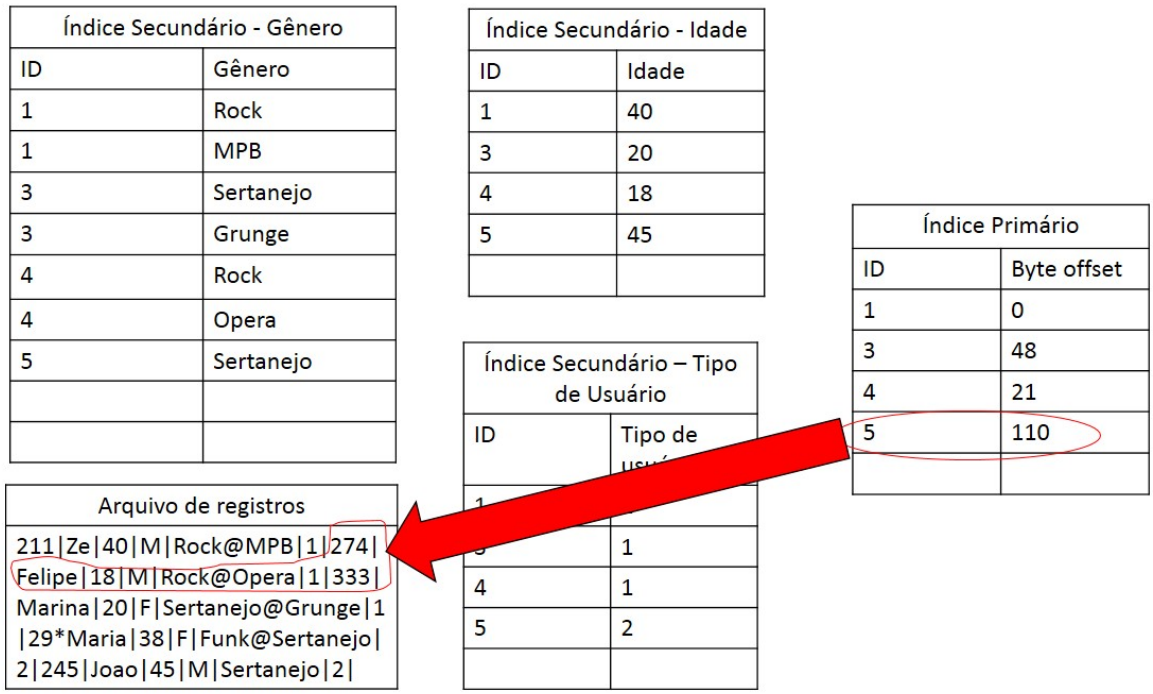
Índice Secundário - Idade	
ID	Idade
1	40
4	18
3	20
0	38
5	45

Índice Primário	
ID	Byte offset
1	0
4	21
3	48
0	81
5	110

Índice Secundário – Tipo de Usuário	
ID	Tipo de usuário
1	1
4	1
3	1
0	2
5	2

Arquivo de registros
211|Ze|40|M|Rock@MPB|1|274| Felipe|18|M|Rock@Opera|1|333| Marina|20|F|Sertanejo@Grunge|1 |29*Maria|38|F|Funk@Sertanejo| 2|245|Joao|45|M|Sertanejo|2|

Acesso direto através do Índice primário (Acessa-se o Felipe pela sua ID = 4):



Índice Secundário - Gênero	
ID	Gênero
1	Rock
1	MPB
3	Sertanejo
3	Grunge
4	Rock
4	Opera
5	Sertanejo

Índice Secundário - Idade	
ID	Idade
1	40
3	20
4	18
5	45

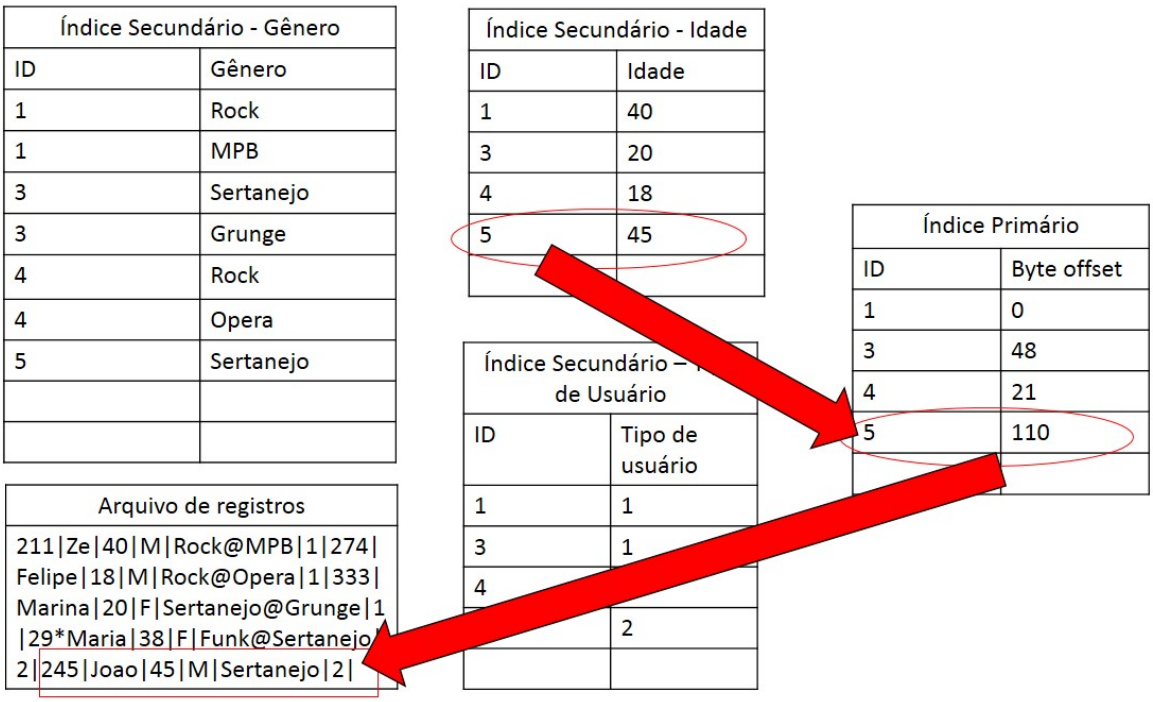
Índice Primário	
ID	Byte offset
1	0
3	48
4	21
5	110

Índice Secundário – Tipo de Usuário	
ID	Tipo de usuário
1	1
3	1
4	1
5	2

Arquivo de registros
211|Ze|40|M|Rock@MPB|1|274| Felipe|18|M|Rock@Opera|1|333| Marina|20|F|Sertanejo@Grunge|1 |29*Maria|38|F|Funk@Sertanejo| 2|245|Joao|45|M|Sertanejo|2|

A busca por um Índice secundário de idade (Busca-se pela idade de 45 anos):

<u>Obs.: O índice secundário de gênero funciona de forma análoga</u>



Índice Secundário - Gênero	
ID	Gênero
1	Rock
1	MPB
3	Sertanejo
3	Grunge
4	Rock
4	Opera
5	Sertanejo

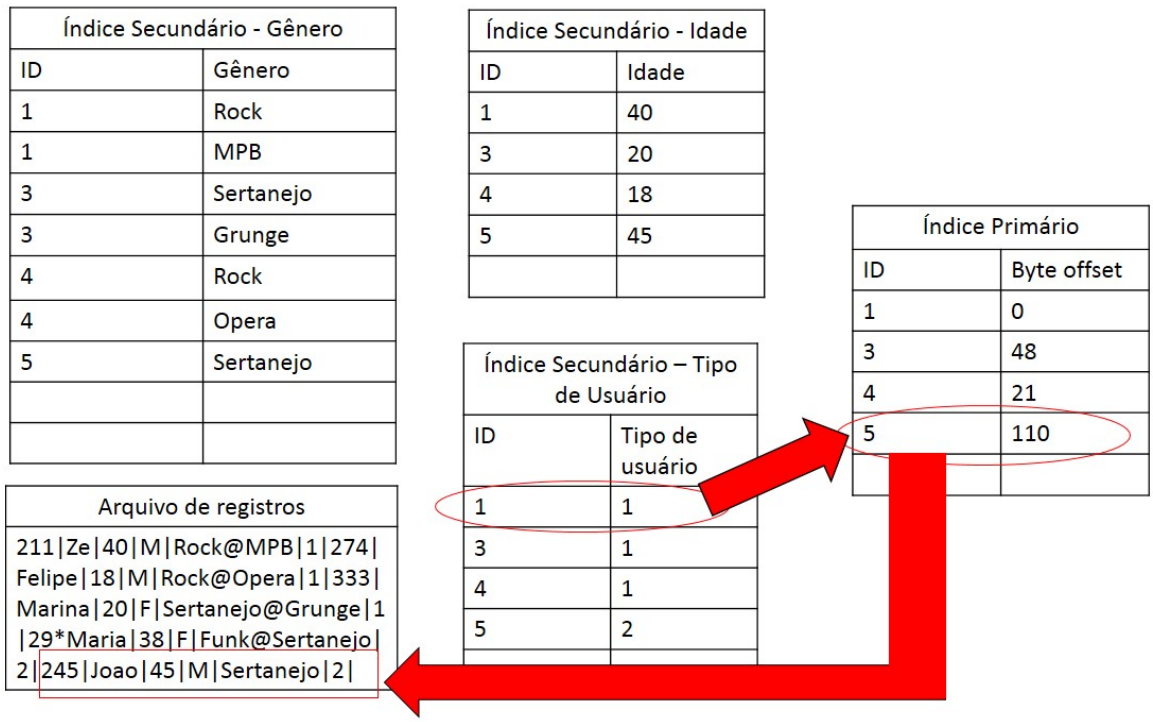
Índice Secundário - Idade	
ID	Idade
1	40
3	20
4	18
5	45

Índice Primário	
ID	Byte offset
1	0
3	48
4	21
5	110

Índice Secundário – Tipo de Usuário	
ID	Tipo de usuário
1	1
3	1
4	
	2

Arquivo de registros
211|Ze|40|M|Rock@MPB|1|274| Felipe|18|M|Rock@Opera|1|333| Marina|20|F|Sertanejo@Grunge|1 |29*Maria|38|F|Funk@Sertanejo| 2|245|Joao|45|M|Sertanejo|2|

Busca por um índice secundário de tipo de usuário pela ID:



Índice Secundário - Gênero	
ID	Gênero
1	Rock
1	MPB
3	Sertanejo
3	Grunge
4	Rock
4	Opera
5	Sertanejo

Índice Secundário - Idade	
ID	Idade
1	40
3	20
4	18
5	45

Índice Primário	
ID	Byte offset
1	0
3	48
4	21
5	110

Índice Secundário – Tipo de Usuário	
ID	Tipo de usuário
1	1
3	1
4	1
5	2

Arquivo de registros
211|Ze|40|M|Rock@MPB|1|274| Felipe|18|M|Rock@Opera|1|333| Marina|20|F|Sertanejo@Grunge|1 |29*Maria|38|F|Funk@Sertanejo| 2|245|Joao|45|M|Sertanejo|2|