Organização e Recuperação de Dados
Profa. Valéria

6ª Lista de Exercícios

1. Escreva sobre as propriedades e as vantagens do uso de índices sobre o uso de arquivos (de dados) ordenados.

   Propriedades dos índices:
   a. Trabalham por indireção → permitem que uma ordem seja imposta ao arquivo de dados sem rearranjá-lo fisicamente
      Vantagens: Facilita a inserção de registros no arquivo de dados e elimina o problema dos registros "fixos" no caso de ser preciso realizar uma ordenação.
   b. Proporcionam múltiplas "visões" de um mesmo arquivo de dados
      Vantagem: Cada índice impõe uma ordenação diferente aos dados, sem que o arquivo de dados precise ser fisicamente ordenado.
   c. Permitem acesso direto por chave a registros de tamanho fixo e variável
      Vantagem: Maior flexibilidade na escolha do formato dos registros

2. Quando o uso de índices lineares deixa de ser recomendado? Por quê?

   Quando o índice atingir um tamanho que não compense mantê-lo carregado na memória principal. Nesse caso, o índice precisará ser acessado/mantido em disco e a estrutura linear já não será adequada.

3. Os índices secundários podem ser organizados de forma indireta (chave secundária → chave primária) ou de forma direta (chave secundária →*byte-offset*). Discuta as vantagens e desvantagens de ambas as organizações.

   Na forma indireta (*late binding*), as vantagens são: maior segurança devido ao fato de as modificações referentes à byte-offset no arquivo de dados afetarem apenas o índice primário; e menor custo nas alterações/remoções, pois nesses casos apenas o índice primário precisará ser atualizado. A desvantagem é que a busca será mais lenta, pois deverá consultar pelo menos dois índices antes de recuperar o registro.

   Na forma direta (*early binding*), a vantagem é uma busca mais rápida, pois o índice já armazena o byte-offset do registro. As desvantagens são o alto custo das alterações, devido ao fato de uma modificação no byte-offset de um registro demandar ajustes em todos os índices, e o alto custo das remoções, pois nesse caso as entradas referentes ao registro removido deverão ser removidas de todos os índices. Por demandar um número maior de operações para que os índices fiquem consistentes, é considerada uma abordagem menos segura.

4. Quando um registro é alterado em um arquivo indexado, os índices primários e secundários podem ser alterados ou não, dependendo se o arquivo armazena registros de tamanho fixo ou variável e dependendo do tipo de alteração feita. Faça uma lista das diferentes situações de alteração que podem ocorrer em um arquivo de registros e explique como cada uma delas afeta os índices.

   (a) Supondo que os índices estão organizados por *late binding* (índices secundários apontam para o índice primário) e que não é possível alterar a chave primária.
   - Alterações (nos campos do registro):
     Se registro de tamanho fixo, possível alteração de chave nos índices secundários. Não altera o índice primário.
     Se registro de tamanho variável, sem alteração de byte-offset, possível alteração de chave nos índices secundários. Se houver alteração de byte-offset do registro, será necessário alterar o índice primário.
   - Remoções:

Se registro de tamanho fixo ou variável, alteração no índice primário.

(b) Supondo que os índices estão organizados por *early binding* (todos os índices referenciam byte-offsets) e que não é possível alterar a chave primária.

- Alterações (nos campos do registro):
  Se registro de tamanho fixo, possível alteração de chave nos índices secundários. Não altera o índice primário.
  Se registro de tamanho variável, sem alteração de byte-offset, possível alteração de chave nos índices secundários. Se houver alteração de byte-offset do registro, será necessário alterar todos os índices (primário e secundários).
- Remoções:
  Se registro de tamanho fixo ou variável, alteração em todos os índices (primário e secundários).

5. A escolha da abordagem de remoção nos índices (*Delete-all-references* ou *Delete-some-references*) é dependente da forma de organização desses índices. Por quê?

   Se os índices estão organizados de forma indireta (*late binding*), todos os índices secundários referenciam o índice primário, que por sua vez referencia os byte-offsets dos registros. Dessa forma, no caso de uma remoção, pode-se remover apenas a entrada correspondente no índice primário (“*delete-some-references*”). Quando a busca no índice primário retornar falso (não encontrado), sabe-se que o registro foi removido, mesmo que ele esteja referenciado nos índices secundários.

   Se os índices estão organizados de forma direta (*early binding*), todos os índices referenciam os byte-offsets dos registros. Dessa forma, quando ocorre a remoção de um registro, as entradas correspondentes em todos os índices precisam ser removidas (“*delete-all-references*”). Caso contrário, os índices ficariam inconsistentes.

6. Qual a finalidade do uso da lista invertida no contexto de índices secundários?

   Com o uso de uma lista invertida, o índice secundário terá apenas uma ocorrência de cada chave secundária, o que possibilita o uso de busca binária nesse índice. Além disso, novos registros com chaves já presentes no índice serão indexados sem a necessidade de reordenar o índice, pois a inserção ocorrerá apenas na lista invertida.

7. Considerando as chaves mostradas abaixo, juntamente com a ordem temporal de inserção de cada chave, mostre como fica o arquivo de índice secundário e o arquivo de lista invertida.

Ordem Temporal	Chave Secundária	Chave Primária
6	Beethoven	ANG3795
5	Beethoven	DG139201
2	Beethoven	DG18807
3	Beethoven	RCA2626
7	Corea	WAR23699
1	Dvorak	COL31809
4	Prokofiev	LON2312

Índice secundário

Chave	**RRN**
Beethoven	5
Corea	6
Dvorak	0
Prokofiev	3

Lista Invertida

	Chave	**Prox**
0	COL31809	-1
1	DG18807	2
2	RCA2626	-1
3	LON2312	-1
4	DG139201	1
5	ANG3795	4
6	WAR23699	-1

8. O uso de lista invertida gera um problema de perda da “localidade” das chaves. Por quê?

   Porque fisicamente na lista invertida as chaves estão na ordem em que foram inseridas, de modo que referências a uma mesma chave secundária podem estar espalhadas pela lista invertida.

9. Dado o arquivo abaixo, monte dois índices secundários, um por nacionalidade e outro por profissão, usando o mesmo arquivo de lista invertida para os dois índices. Considere a matrícula como chave primária.

	Matrícula	Profissão	Nacionalidade
1	2050	ANALISTA	CHILENA
2	430	PROGRAMADOR	BRASILEIRA
3	980	DIGITADOR	ARGENTINA
4	1010	OPERADOR	BRASILEIRA
5	2000	DIGITADOR	PARAGUAIA
6	1900	ANALISTA	BRASILEIRA
7	1550	DIGITADOR	CHILENA
8	690	OPERADOR	CHILENA
9	730	PROGRAMADOR	BRASILEIRA
10	1100	OPERADOR	ARGENTINA
11	1790	ANALISTA	ARGENTINA
12	1990	ANALISTA	CHILENA
13	2200	OPERADOR	BRASILEIRA
14	1620	PROGRAMADOR	BRASILEIRA
15	790	DIGITADOR	CHILENA
16	1040	DIGITADOR	ARGENTINA

Índice de Profissão

Chave	RRN
ANALISTA	10
DIGITADOR	14
OPERADOR	7
PROGRAMADOR	1

Índice de Nacionalidade

Chave	RRN
ARGENTINA	2
BRASILEIRA	1
CHILENA	7
PARAGUAIA	4

Lista Invertida

	Chave	Prox-Prof	Prox-Nac
0	2050	-1	-1
1	430	8	8
2	980	15	15
3	1010	9	13
4	2000	-1	-1
5	1900	11	12
6	1550	4	11
7	690	3	14
8	730	13	3
9	1100	12	10
10	1790	5	-1
11	1990	0	0
12	2200	-1	-1
13	1620	-1	5
14	790	2	6
15	1040	6	9

10. Considerando que existem os procedimentos *match* e *merge* que fazem, respectivamente, a intersecção e a união de duas listas, e que existem pelo menos dois índices para um determinado

arquivo de registros, escreva um procedimento (pseudocódigo) que processe e responda a buscas com chaves combinadas (AND ou OR de duas chaves).

11. Escreva um procedimento (pseudocódigo) que atualize um índice secundário (organizado como uma lista invertida) referente a um determinado arquivo de dados após a inserção de um novo registro.

(As respostas dos exercícios 10 e 11 não foram incluídas no gabarito por haver várias resoluções possíveis)