INF / UFG

Disciplina

Banco de Dados

Conteúdo

Armazenamento, estruturas básicas de arquivo e *hashing*.

(2)

Organização de Registros em Arquivos

Heap – um registro pode ser colocado em qualquer lugar no arquivo onde haja espaço (arquivo não ordenado).

Sequencial – armazene registros em ordem sequencial, com base no valor da chave de busca de cada registro (arquivo ordenado).

Hashing – uma função de *hash* calculada sobre algum atributo de cada registro; o resultado especifica em que bloco do arquivo o registro deve ser colocado.

IMPORTANTE
Registros de cada relação podem ser armazenados em um arquivo separado. Em uma organização de arquivo em *clusters*, os registros de várias relações diferentes podem ser armazenados no mesmo arquivo
Motivação: armazene registros relacionados no mesmo bloco para reduzir a E/S.

Armazenamento do dicionário de dados

Informação sobre relações
- >> nomes de relações
- >> nomes e tipos de atributos de cada relação
- >> nomes e definições de views
- >> restrições de integridade

Informações de usuário, incluindo senhas

Dados estatísticos e descritivos
- >> número de tuplas em cada relação

Informações de organização física do arquivo
- >> como a relação é armazenada (sequencial/hash/...)
- >> localização física da relação
  - **nome de arquivo do sistema operacional, ou**
  - **endereços de disco dos blocos contendo registros da relação**

Informações sobre índices

Armazenamento do dicionário de dados

Estrutura de catálogo:

>> estruturas de dados especializadas, projetadas para acesso eficiente

>> um conjunto de relações, com recursos existentes no sistema usados para garantir o acesso eficiente **(preferida)**

Uma representação de catálogo possível:

Metadados-relação = (nome-relação, número-de-atributos, organização-armazenamento, localização)
Metadados-atributo = (nome-atributo, nome-relação, tipo-domínio, posição, tamanho)
Metadados-usuário = (nome-usuário, senha-criptografada, grupo)
Metadados-índice = (nome-índice, nome-relação, tipo-índice, atributos-índice)
Metadados-view = (nome-view, definição)

Arquivos com registros NÃO ORDENADOS

Também denominados **arquivos de *heap*** ou **de pilha**.

Os registos são colocados no arquivo na ordem em que são inseridos, de forma que novos registos são postos no fim do arquivo.

Esta organização é geralmente usada com caminhos de acesso adicionais, tais como os índices secundários.

Inserção de dados

Inserir um novo registro é muito eficiente. O último bloco do arquivo é copiado para um *buffer*, o novo registro é adicionado, e o bloco é então reescrito volta para o disco.

Arquivos com registros NÃO ORDENADOS

Pesquisa de dados

A busca de um registro usando qualquer condição de pesquisa envolve uma **busca linear** para encontrar o bloco de arquivo que possui o registro: **um procedimento caro**.

Se apenas um registro satisfaz a condição de pesquisa, então, em média, serão pesquisados metade dos blocos de arquivo antes de encontrar o registro.

Arquivos com registros NÃO ORDENADOS

Exclusão de dados

Para excluir um registro, um programa deve primeiro encontrar o seu bloco, copiar o bloco em um *buffer*, excluir o registro do *buffer* e, finalmente, reescrever o bloco de volta para o disco:

>> deixa espaço não utilizado no bloco de disco.

Outra técnica utilizada para a exclusão é ter um byte (ou bit) extra por registro, chamado um marcador de exclusão (exclusão lógica).

Ambas as técnicas requerem **reorganização periódica** do arquivo para recuperar o espaço não utilizado de registros excluídos.
Outra possibilidade é a reutilização do espaço de registros excluídos ao inserir novos registros, mas isso requer o controle de locais vazios.

Arquivos com registros NÃO ORDENADOS

Exclusão de dados (cont.)

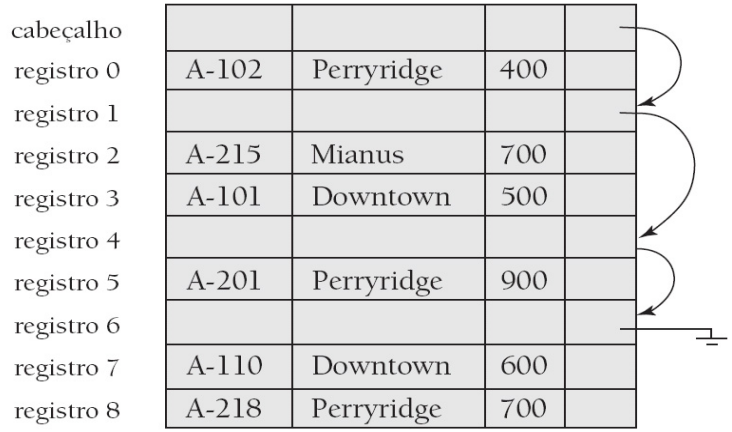
Alternativas para a exclusão do registro i:

>> mova os registros i + 1, . . ., n para i, . . . , n – 1
>> mova o registro n para i
>> não mova registros, mas vincule todos os registros livres em uma lista livre.

Arquivos com registros NÃO ORDENADOS

Exclusão de dados (cont.) - uso de uma lista livre ...

Armazena o endereço do primeiro registro excluído no cabeçalho do arquivo.
Use esse primeiro registro para armazenar o endereço do segundo registro excluído, e assim por diante.
Pense nesses endereços armazenados como ponteiros, pois “apontam” para o local de um registro.
Representação com uso mais eficiente do espaço: reutiliza espaço para atributos normais de registros livres para armazenar ponteiros. (Nenhum ponteiro armazenado nos registros em uso.)



Arquivos com registros NÃO ORDENADOS

Alteração de dados

A alteração ocorre de forma similar à exclusão, mas o novo valor do registro é, idealmente, reescrito no mesmo bloco em disco.

A alteração de um registro de comprimento variável pode exigir a exclusão do registro antigo e inserir um registro modificado, porque o registro modificado pode não caber em seu antigo espaço (bloco).

Arquivos com registros NÃO ORDENADOS

Arquivo com acesso direto (ou acesso relativo)

Ocorre em registros de comprimento fixo, blocos não espalhados e alocação contínua.

Simplifica o acesso, pois qualquer registro é alcançado por sua posição no arquivo.

Se os registos são numerados de ***0, 1, 2, ..., (r - 1)*** e os registos de cada bloco são numerados de ***0, 1, ..., (bfr - 1)***, onde *bfr* é o fator de bloco, o i-ésimo registro do arquivo está localizado no bloco
$\lfloor (i / bfr) \rfloor$, e é o registro na posição ***(i mod bfr)*** do referido bloco.

Essa organização não é útil quando se busca um registro a partir de um predicado, mas pode facilitar a construção de índices.

Arquivos com registros ORDENADOS

Os registros do arquivo são ordenados fisicamente com base nos valores de um de seus campos: **campo de ordenação**. Se o campo de ordenação é também um campo-chave do arquivo (valor único para cada registro), o campo é chamado **chave de ordenação**.

Algumas vantagens:

- a leitura dos registros na ordem dos valores fundamentais de ordenação torna-se extremamente eficiente;

- encontrar o próximo registro ordenado não requer acesso a bloco adicional, porque o próximo registro está no mesmo bloco que o atual (a menos que o registro atual seja o último do bloco);

- permite aplicar a busca binária, o que constitui uma melhoria em relação a pesquisas lineares.

Arquivos com registros ORDENADOS

A busca binária usualmente acessa $\log_2(b)$ blocos:

>> se o registro for encontrado ou não.

Busca linear:

>> (b/2) blocos são acessados quando o registro é encontrado; ou

>> (b) blocos são acessados quando o registro não é encontrado (ou mais de um registro satisfizer o predicado de busca).

Arquivos com registros ORDENADOS

	Nome	Cpf	Data_nascimento	Cargo	Salario	Sexo
Bloco 1	Aaron, Eduardo					
	Abílio, Diana					
	⋮					
	Acosta, Marcos					
Bloco 2	Adams, João					
	Adams, Roberto					
	⋮					
	Akers, Janete					
Bloco 3	Alexandre, Eduardo					
	Alfredo, Roberto					
	⋮					
	Allen, Samuel					
Bloco 4	Allen, Tiago					
	Anderson, Kely					
	⋮					
	Anderson, Joel					
Bloco 5	Anderson, Isaac					
	Angeli, José					
	⋮					
	Anita, Sueli					
Bloco 6	Arnoldo, Marcelo					
	Arnoldo, Estevan					
	⋮					
	Atílio, Timóteo					
	⋮					
Bloco *n*–1	Wanderley, Jaime					
	Wesley, Ronaldo					
	⋮					
	Wong, Manuel					
Bloco *n*	Wong, Pâmela					
	Wuang, Charles					
	⋮					
	Zimmer, André					

Figura 17.7
Alguns blocos de um arquivo ordenado (sequencial) de registros de FUNCIONARIO com Nome como campo-chave de ordenação.

Arquivos com registros ORDENADOS

Algorithm 17.1. Binary Search on an Ordering Key of a Disk File

```
l ← 1; u ← b; (* b is the number of file blocks *)
while (u ≥ l ) do
    begin i ← (l + u) div 2;
    read block i of the file into the buffer;
    if K < (ordering key field value of the first record in block i )
        then u ← i – 1
    else if K > (ordering key field value of the last record in block i )
        then l ← i + 1
    else if the record with ordering key field value = K is in the buffer
        then goto found
    else goto notfound;
    end;
goto notfound;
```


Arquivos com registros ORDENADOS

Inserção e exclusão são operações dispendiosas, porque os registros devem permanecer fisicamente ordenados.

Para **inserir um registro**, temos de encontrar a sua posição correta no arquivo, com base no seu valor de campo de ordenação, e depois abrir espaço no arquivo para inserir o registro nessa posição.

Para um arquivo grande isso pode ser muito demorado, porque, em média, metade dos registros do arquivo deve ser movido para abrir espaço para o novo registro:

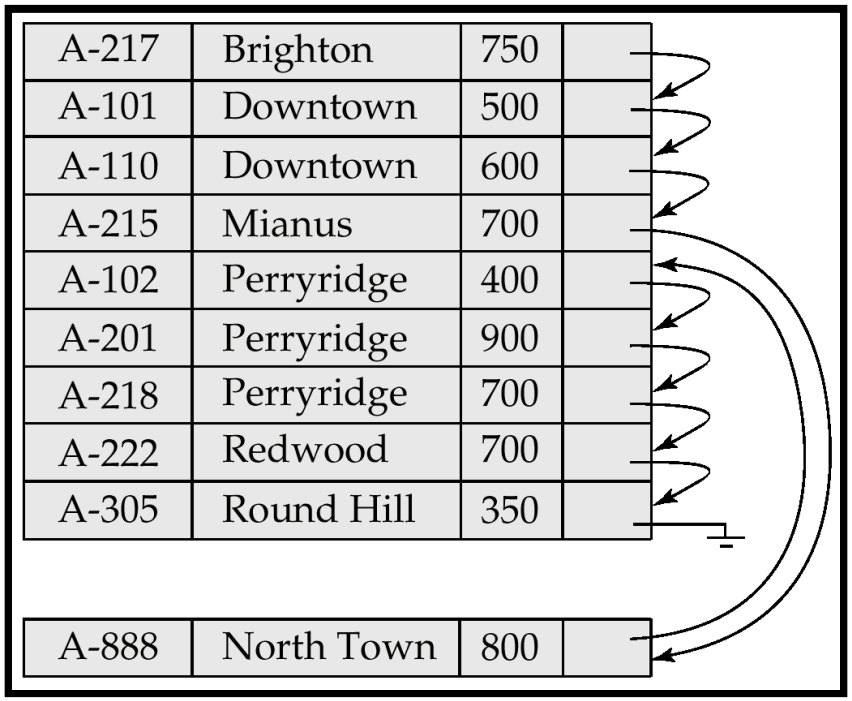
>> se não houver espaço livre no bloco, insira o registro em um bloco de *overflow* (estouro).

Para a **exclusão de registros**, o problema é menos grave se os marcadores de exclusão e a reorganização periódica forem usados.

Arquivos com registros ORDENADOS

Reorganização do arquivo

Os registros no arquivo de *overflow* são ordenados e mesclados com o arquivo principal. Os registros marcados para exclusão são removidos durante a reorganização.



Arquivos com registros ORDENADOS

Acesso por método de busca

Tabela 17.2

Tempos de acesso médios para um arquivo de *b* blocos sob organizações de arquivo básicas.

Tipo de organização	Método de acesso/pesquisa	Média de blocos para acessar um registro específico
Heap (não ordenado)	Varredura sequencial (pesquisa linear)	$b/2$
Ordenado	Varredura sequencial	$b/2$
Ordenado	Pesquisa binária	$\log_2 b$

Arquivos com registros ORDENADOS

A **modificação** de valor de campo de um registro depende de dois fatores:
(i) a condição de pesquisa para localizar o registro, e
(ii) o campo a ser modificado.

Se a condição de pesquisa envolver a chave de ordenação, podemos localizar o registro por busca binária; caso contrário, devemos fazer uma busca linear.

Modificar o valor do campo ordenação significa que o registro pode mudar sua posição no arquivo. Isto requer a exclusão do registro, seguida inserção do registro modificado.