INF / UFG

Disciplina

Banco de Dados

Conteúdo

Armazenamento, estruturas básicas de arquivo e *hashing*.

Preâmbulo

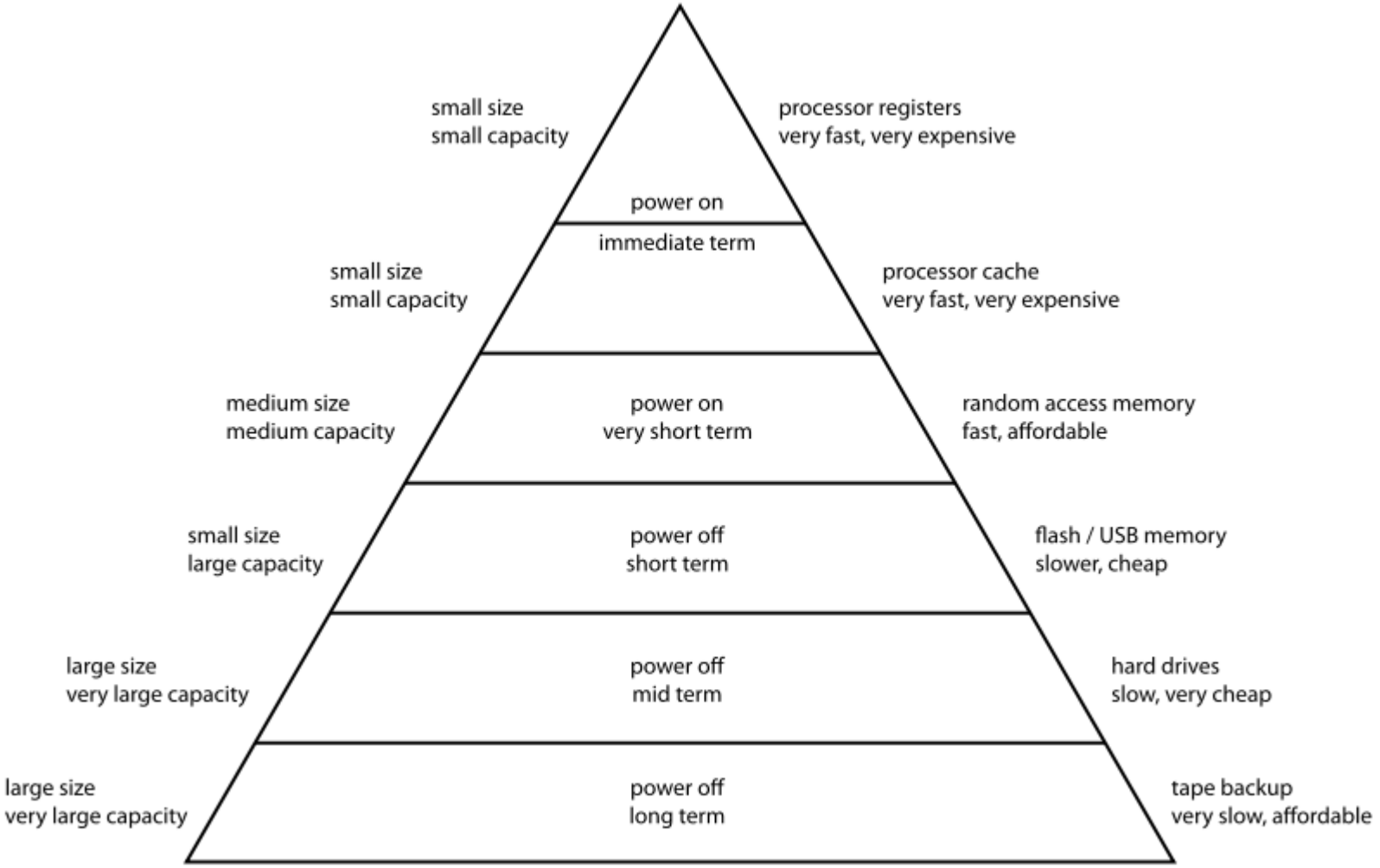
A coleção de dados que compõem um banco de dados computadorizado deve ser armazenada fisicamente em algum meio de armazenamento no computador.

Armazenamento primário
- pode ser operado diretamente pela CPU, como a memória principal do computador e memórias cache menores, porém mais rápidas.

Armazenamento secundário
- discos magnéticos, discos ópticos.

Os dados residem e são transportados por uma hierarquia de meios de armazenamento. A memória de velocidade mais alta é a mais cara e, portanto, está disponível com a menor capacidade.

Preâmbulo


small size
small capacity
processor registers
very fast, very expensive
power on
immediate term
small size
small capacity
processor cache
very fast, very expensive
medium size
medium capacity
power on
very short term
random access memory
fast, affordable
small size
large capacity
power off
short term
flash / USB memory
slower, cheap
large size
very large capacity
power off
mid term
hard drives
slow, very cheap
large size
very large capacity
power off
long term
tape backup
very slow, affordable

Preâmbulo

Os bancos de dados costumam armazenar grande quantidade de dados que precisam persistir por longos períodos de tempo e, portanto, eles costumam ser considerados **dados persistentes**.

A maioria dos bancos de dados é armazenada de maneira permanente (ou persistentemente) no armazenamento secundário do disco, pelos seguintes motivos:

- em geral, os bancos de dados são muito grandes para caber inteiramente na memória principal;

- as circunstâncias que causam perda permanente de dados armazenados surgem com menos frequência para o armazenamento secundário (não volátil);

- o custo de armazenamento está na ordem de grandeza menor para armazenamento de disco secundário do que para o armazenamento primário.

Buffering de blocos

As transferências (cópias) de dados entre a memória primária e secundária é realizada por meio de **blocos**. Tamanhos típicos de blocos variam de 512 a 8192 bytes.

***Buffer* (retentor)**
- é uma região de memória temporária utilizada para escrita e leitura de dados (*wikipedia*).

Buffering de blocos

Leitura e o processamento podem prosseguir em paralelo: quando o tempo exigido para processar um bloco de disco na memória é menor que o tempo exigido para ler o próximo bloco e preencher um *buffer*.

Enquanto um *buffer* está sendo lido ou gravado, a CPU pode processar dados em outro *buffer*, pois existe um processador (controlador) de E/S de disco separado que, uma vez iniciado, pode prosseguir para transferir um bloco de dados entre a memória e o disco independente e em paralelo com o processamento da CPU.

Buffering de blocos

O *buffering* é mais útil quando processos podem ser executados simultaneamente e em um padrão paralelo:

>> existe um processador de E/S , e/ou
>> há vários processadores (CPUs).

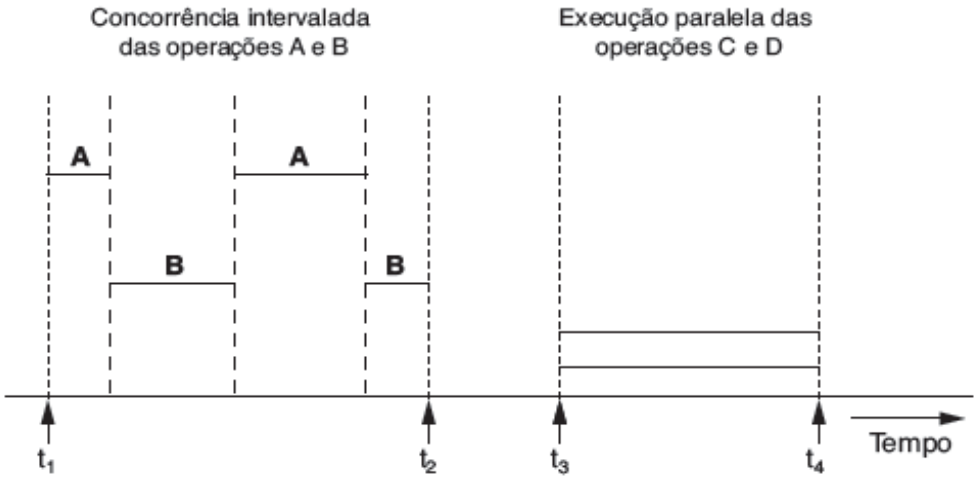


Figura 17.3
Concorrência intervalada *versus* execução paralela.

Buffering de blocos

No exemplo, os processos A e B estão rodando "simultaneamente".

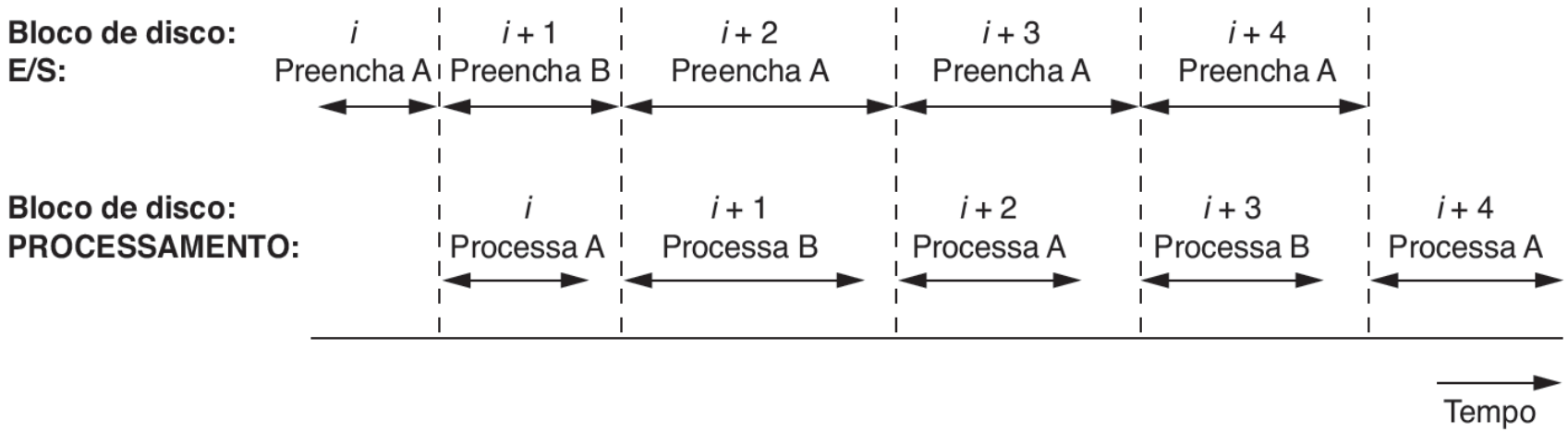


Figura 17.4

Uso de dois buffers, A e B, para a leitura do disco.

Buffering de blocos

A CPU pode começar a processar um bloco quando sua transferência para a memória principal termina; ao mesmo tempo, o processador de E/S de disco pode estar lendo e transferindo o próximo bloco para um *buffer* diferente (*buffering* duplo).

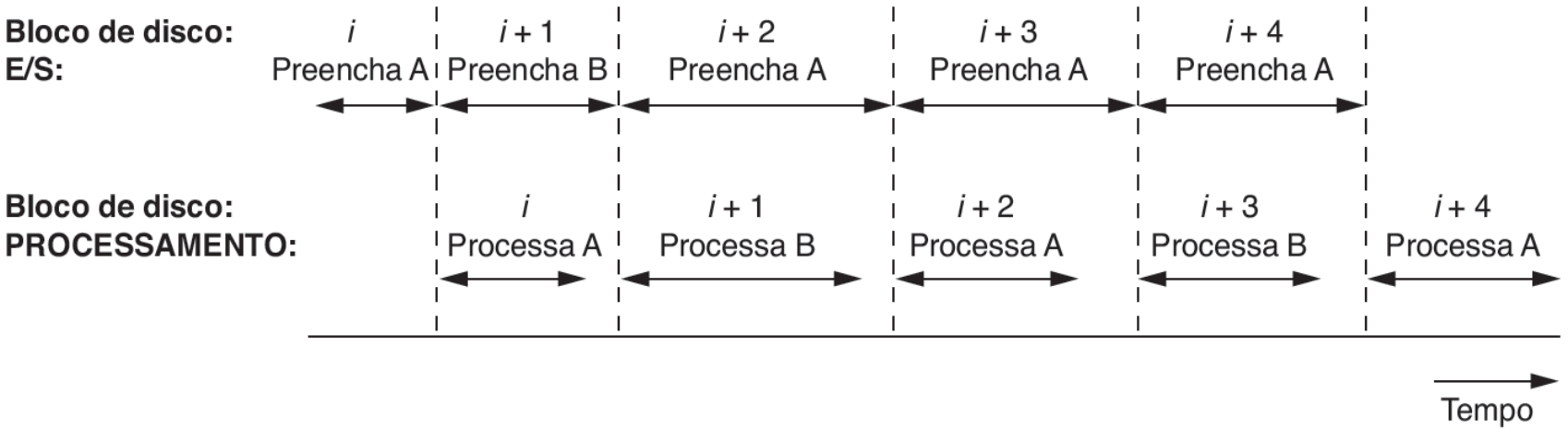


Figura 17.4

Uso de dois buffers, A e B, para a leitura do disco.

Registros de Dados

Os dados costumam ser armazenados na forma de **registros**.

Cada registro contém uma coleção de valores ou itens de dados relacionados, no qual cada valor é formado por um ou mais bytes e corresponde a um campo em particular do registro.

Os registros normalmente descrevem entidades e seus atributos:

>> Por exemplo, um registro de FUNCIONARIO representa uma entidade de funcionário, e o valor de cada campo no registro especifica algum atributo desse funcionário, como Nome, Data_nascimento, Salario ou Supervisor.

```
struct employee{
    char name[30];
    char ssn[9];
    int salary;
    int job_code;
    char department[20];
} ;
```


Registros de Dados

Um arquivo é uma sequência de registros.

Um **tipo de registro** é uma coleção de:
>> nomes de campos, e
>> seus tipos de dados correspondentes.

Em muitos casos, todos os registros em um arquivo são do mesmo tipo de registro.

Se cada registro no arquivo tem exatamente o mesmo tamanho (em bytes), o arquivo é considerado composto de **registros de tamanho fixo**.

Registros de Dados

Um tipo de dado indica se o espaço ocupado pelo dados é de tamanho fixo ou variável:

>> *strings* podem ser de tamanho fixo ou variável; se for variável, o espaço ocupado depende do valor do dado.

Em dados associados a tipos de tamanho variável, geralmente é usado um caractere especial que denota o **final do dado**.

Caracteres especiais também são utilizados para representar o **final de registro**.

Registros de Tamanho Variável

Registros de tamanho variável surgem nos sistemas de banco de dados de várias maneiras:

>> armazenamento de vários tipos de registro em um arquivo;
>> tipos de registro que permitem tamanhos variáveis para um ou mais campos;
>> tipos de registro que permitem campos repetitivos (usados em alguns modelos de dados mais antigos).

Problemas:

>> dificuldade com exclusão
>> dificuldade com crescimento

Registros de Dados

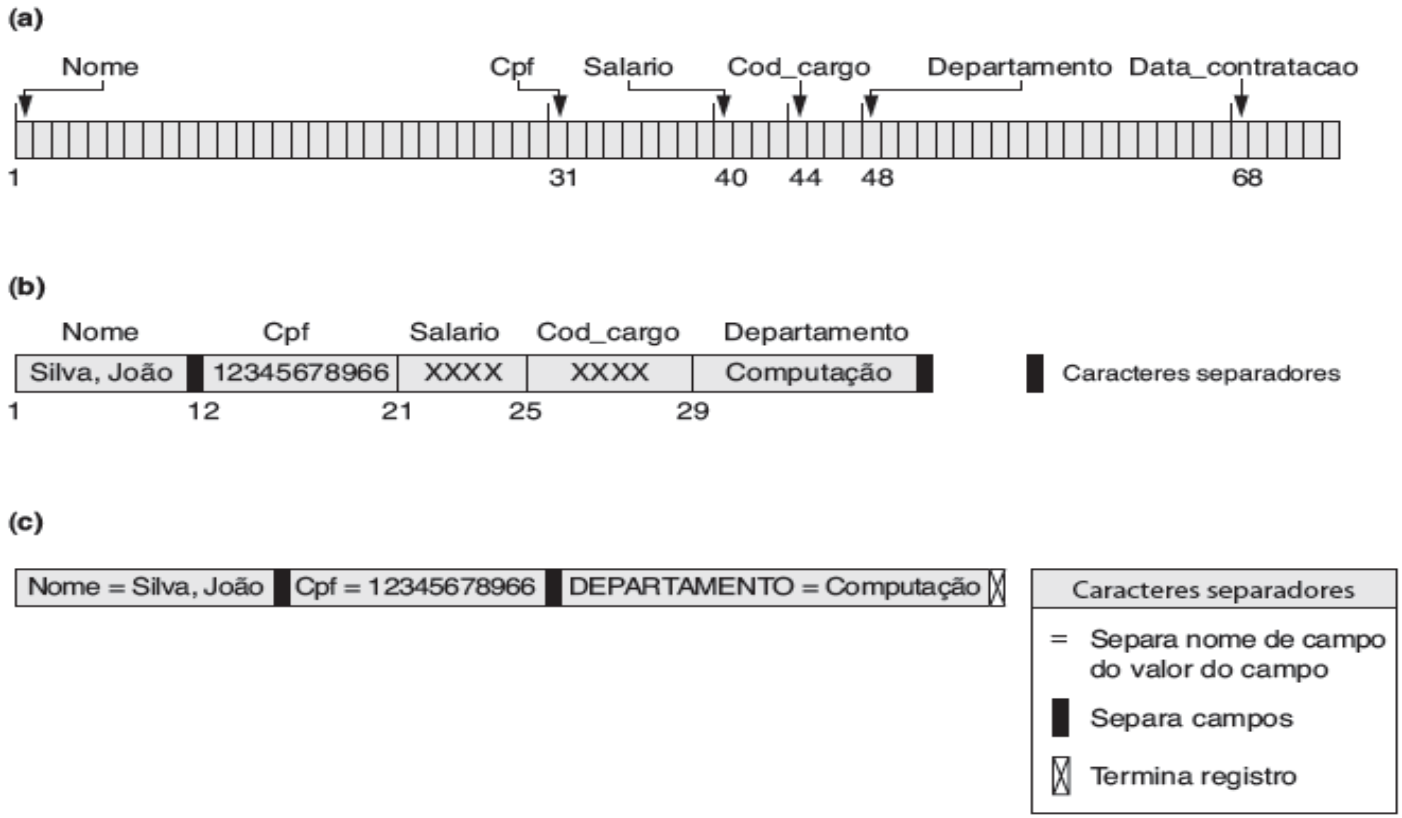


Figura 17.5

Três formatos de armazenamento de registro. (a) Um registro de tamanho fixo com seis campos e tamanho de 71 bytes. (b) Um registro com dois campos de tamanho variável e três campos de tamanho fixo. (c) Um registro de campo variável com três tipos de caracteres separadores.

Blocagem de Registros

Um arquivo de banco de dados é particionado em unidades de armazenamento de tamanho fixo, chamadas blocos. Os blocos são unidades de alocação de armazenamento e transferência de dados.

O sistema de banco de dados busca minimizar o número de transferências de bloco entre o disco e a memória. Podemos reduzir o número de acessos ao disco mantendo o máximo de blocos possível na memória principal.

Buffer – parte da memória principal disponível para armazenar cópias de blocos de disco.

Gerenciador de *buffer* – subsistema responsável por alocar espaço em *buffer* na memória principal.

Gerenciamento de *Buffer* (resumo)

Quando se precisa de um bloco do disco:

(1) se o bloco já estiver no *buffer*, o programa solicitante recebe o endereço do bloco na memória principal;

(2) se o bloco não estiver no *buffer*, o gerenciador de *buffer* aloca espaço no *buffer* para o bloco, substituindo (descartando) algum outro bloco, se for preciso, para criar espaço para o novo bloco:

>> o bloco que é descartado é escrito de volta ao disco somente se foi modificado desde o momento mais recente em que foi escrito/lido do disco.

- >> quando o espaço é alocado no *buffer*, o gerenciador de buffer lê o bloco do disco para o *buffer* e passa o endereço do bloco na memória principal para o solicitante.

Blocagem de registros e registros espalhados versus não espalhados

Os registros de um arquivo precisam ser alocados a blocos de disco, porque um bloco é a unidade de transferência de dados entre o disco e a memória.

Quando o tamanho do bloco é maior que o tamanho do registro, cada bloco terá diversos registros, embora alguns arquivos possam ter registros excepcionalmente grandes que não cabem em um bloco.

Registros de tamanho variável: estrutura de página em *slots*

Cabeçalho da página em *slots* contém:

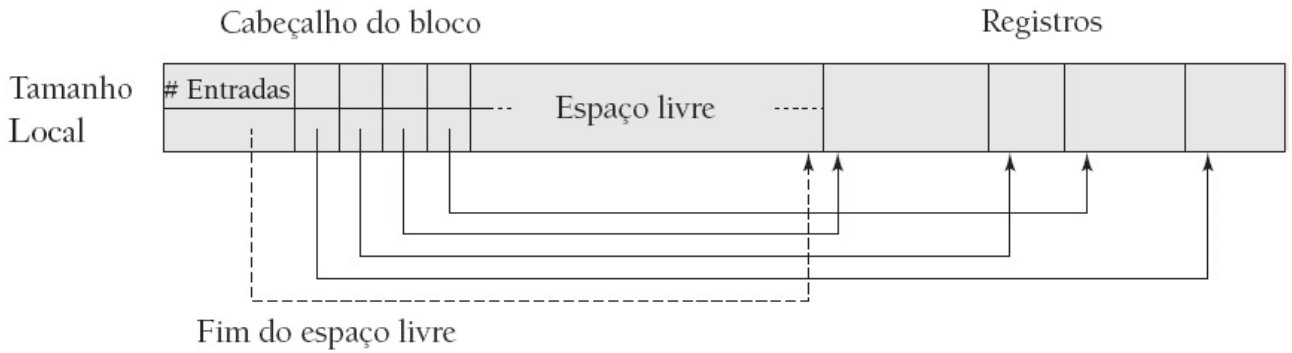
>> número de entradas de registro

>> fim do espaço livre no bloco

>> local e tamanho de cada registro

Os registros podem ser movimentados dentro de uma página para mantê-los contíguos, sem espaço entre eles; a entrada no cabeçalho precisa ser atualizada.

Os ponteiros não devem apontar diretamente para o registro - devem apontar para a entrada para o registro no cabeçalho.



Blocagem de registros e registros espalhados versus não espalhados

Os registros de um arquivo precisam ser alocados a blocos de disco, porque um bloco é a unidade de transferência de dados entre o disco e a memória.

Quando o tamanho do bloco é maior que o tamanho do registro, cada bloco terá diversos registros, embora alguns arquivos possam ter registros excepcionalmente grandes que não cabem em um bloco.

Para registros de tamanho fixo, com B ≥ R, podemos encaixar **bfr = ⌊ B / R ⌋** registros por bloco (**bfr** é chamado de fator de bloco). para o arquivo:

- algum espaço não usado por bloco é igual a **B - (bfr * R)** bytes.

Blocagem de registros e registros espalhados versus não espalhados

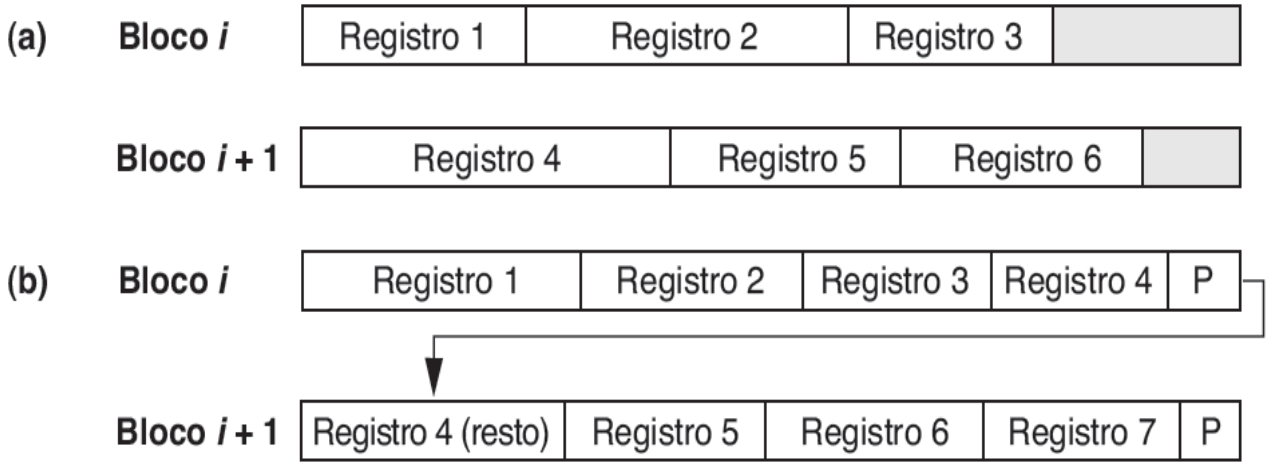


Figura 17.6

Tipos de organização de registro. (a) Não espalhada. (b) Espalhada.

A atribuição de blocos de arquivos no disco

Existem várias técnicas convencionais para a alocação dos blocos de um arquivo no disco.

Na **alocação contígua**, os blocos são alocados para blocos de disco consecutivos. Isso agiliza a leitura do arquivo inteiro usando o *buffer* duplo, mas dificulta a expansão o arquivo.

Na **alocação ligada** (*linked*), cada bloco de arquivo contém um ponteiro para o próximo bloco. Isto torna mais fácil a expansão do arquivo, mas torna mais lenta a leitura do arquivo inteiro.

Uma combinação das alternativas acima aloca *clusters* (conjuntos) de blocos , e os *clusters* (conjuntos de blocos) estão ligados.

Cabeçalho (*header*) de arquivo

Um **cabeçalho** de arquivo (ou **descritor** de arquivo) contém informações necessárias para acessar os registros do arquivo.

O cabeçalho inclui informações sobre:

- os endereços de disco dos blocos de arquivos;
- descrições de formato, tais como: tamanho de campo e a ordem de campos em um registro (registros não espalhados comprimento fixo); e códigos de tipo de campo e caracteres separadores (registros de comprimento variável).

Pesquisa por um Registro

Para procurar um registro no disco:

>> um ou mais blocos são copiados para os *buffers* da memória;
>> os programas procuram o registro (ou registros) dentro dos *buffers*, usando a informação do cabeçalho do arquivo;

(1) se o endereço do bloco que possui o registro for desconhecido, é preciso fazer uma **busca linear** pelos blocos do arquivo;
– (2) cada bloco do arquivo é copiado para um *buffer* e pesquisado até o registro seja localizado, e todos os blocos do arquivo tenha sido pesquisados com sucesso.

O objetivo de uma boa organização de arquivo é localizar o bloco que contém um registro desejado com um número mínimo de transferências de bloco.

Operações em Arquivos

<< apenas para exemplificar, não é necessário aprofundar ... >>

Open. Prepares the file for reading or writing. Allocates appropriate buffers (typically at least two) to hold file blocks from disk, and retrieves the file header. Sets the file pointer to the beginning of the file.
Reset. Sets the file pointer of an open file to the beginning of the file.
Find (or Locate). Searches for the first record that satisfies a search condition. Transfers the block containing that record into a main memory buffer (if it is not already there). The file pointer points to the record in the buffer and it becomes the current record. Sometimes, different verbs are used to indicate whether the located record is to be retrieved or updated.
Read (or Get). Copies the current record from the buffer to a program variable in the user program. This command may also advance the current record pointer to the next record in the file, which may necessitate reading the next file block from disk.
FindNext. Searches for the next record in the file that satisfies the search condition. Transfers the block containing that record into a main memory buffer (if it is not already there). The record is located in the buffer and becomes the current record. Various forms of FindNext (for example, Find
Next record within a current parent record, Find Next record of a given type, or Find Next record where a complex condition is met) are available in legacy DBMSs based on the hierarchical and network models.
Delete. Deletes the current record and (eventually) updates the file on disk to reflect the deletion.
Modify. Modifies some field values for the current record and (eventually) updates the file on disk to reflect the modification.
Insert. Inserts a new record in the file by locating the block where the record is to be inserted, transferring that block into a main memory buffer (if it is not already there), writing the record into the buffer, and (eventually) writing the buffer to disk to reflect the insertion.
Close. Completes the file access by releasing the buffers and performing any other needed cleanup operations.

Operações em Arquivos

Arquivos estáticos:

>> as operações de atualização são raramente realizadas.

Arquivos dinâmicos:

>> podem mudar com frequência, pois as operações de atualização são constantemente aplicadas a eles.

A organização escolhida para um arquivo deve ser tal que traga eficiência às operações aplicadas ao arquivo.

Exemplos de operações – arquivo FUNCIONÁRIOS:

>> inserir registros (quando os funcionários são contratados);

>> excluir registros (quando empregados deixam a empresa);

>> modificar registros (por exemplo, alterar o salário de um funcionário);

>> excluir ou modificar um registro requer uma condição de seleção para identificar um registro específico ou um conjunto de registros;

>> recuperar um ou mais registros também requer uma condição de seleção.

Organização de Arquivo X Método de Acesso

Organização de arquivo:

>> refere-se à organização dos dados de um arquivo em registros, blocos e estruturas de acesso; isso inclui o modo como registros e blocos são colocados no meio de armazenamento e interligados.

Método de Acesso:

>> oferece um grupo de operações (*open*, *delete*, *modify*, *insert*, …) que podem ser aplicadas a um arquivo.

Em geral, pode-se aplicar vários métodos de acesso a uma organização de arquivo.

Alguns métodos só podem ser aplicados a certas organizações de arquivo:

>> o acesso indexado não pode ser aplicado a arquivo sem índice.

Organização de Arquivo X Método de Acesso

Se um grupo de usuários espera, principalmente, aplicar uma condição de pesquisa com base no CPF, o projetista deve escolher uma organização de arquivos que facilita a busca quando for dado o seu valor de CPF:

>> por exemplo, ordenar fisicamente os registros do arquivo pelo valor do CPF, ou criar um índice cuja chave seja o CPF.

Se um outro grupo de usuários necessita gerar contracheques dos funcionários e exige que salários estejam agrupadas por departamento:

>> por exemplo, ordenar os dados por departamento.

O projetista pode privilegiar um grupo de usuários, com base no “valor” das operações de cada grupo.

Se ambas as aplicações são importantes, o projetista deve escolher uma organização de arquivo que permite que ambas as operações sejam atendidas de forma eficiente.

Infelizmente, em muitos casos, isso não é possível ...

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.