

Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladimir Cardoso Brandão

www.wladimirbrandao.com

Fevereiro, 2020



SEÇÃO 17

ARMAZENAMENTO FÍSICO



- ▶ BDs são armazenados *fisicamente* em um **meio de armazenamento** no computador
- ▶ Geralmente como **arquivos de registros** em *discos magnéticos*
- ▶ SGBD *recupera, atualiza e processa* dados de acordo com a necessidade
- ▶ Os meios (mídias) de armazenamento de um computador formam uma **hierarquia de armazenamento**, na qual os dados residem e são transportados
 - ▶ Memória primária → operada pela CPU
 - ▶ Memória secundária → não operada pela CPU
 - ▶ Memória terciária → não operada pela CPU



- ▶ Operação → direto pela CPU
- ▶ Velocidade → alta
- ▶ Capacidade → baixa (KB, MB, GB)
- ▶ Volatilidade → perda de dados na falta de energia
- ▶ Custo → elevado
- ▶ Tipos → *cache* e principal (DRAM)
 - ▶ *Cache* (RAM estática) → extremamente rápida e cara, é usada pela CPU pra agilizar execução de instruções de programa com pré-busca e pipelining
 - ▶ Principal (RAM dinâmica) → de menor velocidade e mais barata que a *cache*, é usada pela CPU pra manter instruções de programa e dados e como área de trabalho



- ▶ Operação → dados devem ser copiados para memória primária para serem operados pela CPU
- ▶ Velocidade → baixa
- ▶ Capacidade → alta (TB)
- ▶ Volatilidade → não volátil
- ▶ Custo → baixo
- ▶ Tipos → discos rígidos magnéticos (HD)



- ▶ Operação → dados devem ser copiados para memória primária para serem operados pela CPU
- ▶ Velocidade → muito baixa
- ▶ Capacidade → muito alta (TB, PB)
- ▶ Volatilidade → não volátil
- ▶ Custo → muito baixo
- ▶ Tipos → fitas magnéticas e dispositivos ópticos
 - ▶ Fitas magnéticas → leitura sequencial, sofrendo interferência eletromagnética. Muito utilizada para *backup*
 - ▶ Dispositivos ópticos → interferência eletromagnética desprezível. Muito utilizada para dados multimídia



- ▶ Forma de armazenamento intermediária entre a memória principal (DRAM) e a secundária (HD)
- ▶ Operação → podem ser operadas diretamente pela CPU
- ▶ Velocidade → alta
- ▶ Capacidade → mediana (GB)
- ▶ Volatilidade → não volátil
- ▶ Custo → médio
- ▶ Tipos → EEPROM
 - ▶ Estão presentes em câmeras, MP3 players, celulares, PDAs e unidades USB e se diferenciam das DRAMs por ser necessário apagar e gravar um bloco inteiro em caso de alteração de dados



- ▶ Programas residem e são executados em memória principal
- ▶ Geralmente grandes BDs residem em armazenamento secundário
- ▶ Partes do BDs são lidos e escritos em *buffers* na memória principal conforme a necessidade



BANCO DE DADOS DE MEMÓRIA PRINCIPAL

- ▶ São BDs que podem ser mantidos inteiros na memória principal (com uma cópia de backup no disco magnético).
- ▶ Úteis em aplicações de tempo real que exigem tempo de resposta rápido.
- ▶ *Por exemplo, em aplicações de comutação de telefone, que armazenam BDs que contêm informações de roteamento e linha.*



MEMÓRIA FLASH

- ▶ Outra forma armazenamento entre a DRAM e o armazenamento em disco magnético;
- ▶ São de alta densidade, alto desempenho, não voláteis e usam a tecnologia EEPROM;
- ▶ A vantagem da memória flash é a velocidade de acesso rápida;
- ▶ A desvantagem é que um bloco inteiro precisa ser apagado e gravado simultaneamente;
- ▶ Estão presentes em câmeras, MP3 players, celulares, PDAs, unidades flash USB.



DISCOS ÓPTICOS

- ▶ Discos de CD-ROM armazenam dados opticamente e são lidos por um laser;
- ▶ Os CD-ROMs contêm dados pré-gravados que não podem ser modificados;
- ▶ O DVD permite 4,5 a 15 GB de armazenamento por disco.



DISCOS ÓPTICOS

- ▶ Os discos WORM são usados para arquivar dados;
- ▶ Permitem que os dados sejam gravados uma vez e lidos qualquer número de vezes;
- ▶ Possuem meio gigabyte de dados por disco e duram mais que os discos magnéticos.



MEMÓRIAS DE JUKEBOX ÓPTICO

- ▶ Utilizam um conjunto de placas de CD-ROM, carregadas em unidades por demanda.
- ▶ Seu tempo de recuperação é mais lento do que dos discos magnéticos.
- ▶ A utilização desse armazenamento diminui devido a diminuição no custo e ao aumento nas capacidades dos discos magnéticos.



FITAS MAGNÉTICAS

- ▶ Usadas para arquivamento e armazenamento de backup dos dados.

JUKEBOXES DE FITA

- ▶ Possuem um banco de fitas que são catalogados e podem ser carregadas automaticamente nas unidades de fita.
- ▶ Populares como **armazenamento terciário**, para manter *terabytes* de dados.



DADOS PERSISTENTES

- ▶ BDs armazenam grande quantidade de dados que persistem por longos períodos de tempo;
- ▶ Dados persistentes são acessados e processados repetidamente durante esse período.

DADOS TRANSIENTES

- ▶ Dados transientes persistem apenas por um tempo limitado durante a execução do programa.



A maioria dos BDs é armazenada de maneira permanente (ou persistente) no armazenamento secundário do disco magnético.

- ▶ BDs são muito grandes para caber inteiramente na memória principal.
- ▶ O custo de armazenamento é **menor** para o armazenamento secundário do que para o primário.
- ▶ No futuro, os bancos de dados poderão residir em diferentes níveis de hierarquia de memória.
- ▶ Contudo, os **discos magnéticos** continuarão a ser a escolha principal para BDs grandes.



- ▶ **Fitas magnéticas** são usadas para o backup de banco de dados.
- ▶ O armazenamento em fita custa menos que o armazenamento em disco.
- ▶ O acesso aos dados na fita é muito lento.
- ▶ Os dados armazenados nas fitas são **off-line**, alguma intervenção para carregar uma fita é necessária antes que os dados se tornem disponíveis.
- ▶ Enquanto os discos são dispositivos **on-line**, podem ser acessados diretamente a qualquer momento.



- ▶ Técnicas para armazenar grande quantidade de dados em disco são importantes para projetistas, DBA e implementadores de um SGBD.
- ▶ Em geral, o SGBD tem várias opções disponíveis para organizar os dados.
- ▶ O processo de **projeto físico de banco de dados** envolve a escolha das técnicas de organização de dados que mais se ajustam aos requisitos.



- ▶ Aplicações típicas precisam apenas de uma pequena parte do BD de cada vez para processamento.
 - ▶ Essa parte é localizada no disco;
 - ▶ Copiada para memória principal para processamento;
 - ▶ E, depois, reescrita para o disco se os dados forem alterados.
- ▶ Dados armazenados no disco são organizados como **arquivos de registros**.
- ▶ Cada registro é uma **coleção de valores de dados**, interpretados como fatos sobre *entidades*, *atributos* e *relacionamentos*.
- ▶ Registros devem ser armazenados em disco de uma maneira que torne possível localizá-los de modo eficiente.



ORGANIZAÇÕES DE ARQUIVO PRIMÁRIO

- ▶ Determinam como os registros de arquivo são *colocados fisicamente* no disco e como os *registros podem ser acessados*.
 - ▶ **Arquivo de heap** (ou **arquivo desordenado**) coloca os registros no disco sem qualquer ordem particular, acrescentando novos registros ao final do arquivo.
 - ▶ **Arquivo classificado** (ou **arquivo sequencial**) mantém os registros ordenados pelo valor de um campo em particular (*campo de classificação*).
 - ▶ **Arquivo em hashing** usa uma função de hash aplicada a um campo em particular (**chave hash**) para determinar o posicionamento de um registro no disco.



ORGANIZAÇÃO SECUNDÁRIA OU ESTRUTURA DE ACESSO AUXILIAR

- ▶ Permite acesso aos registros do arquivo com base em campos alternativos (além dos que foram usados para a organização de arquivo primário).



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ **Discos magnéticos** são usados para armazenar grande quantidade de dados.
- ▶ A unidade de dados básica no disco é um **bit** de informação.
- ▶ Uma área do disco pode representar o valor de bit 0 (zero) ou bit 1 (um).
- ▶ Bits são agrupados em **bytes** (ou **caracteres**) para codificar a informação.
- ▶ Os tamanhos de byte variam de 4 a 8 bits.

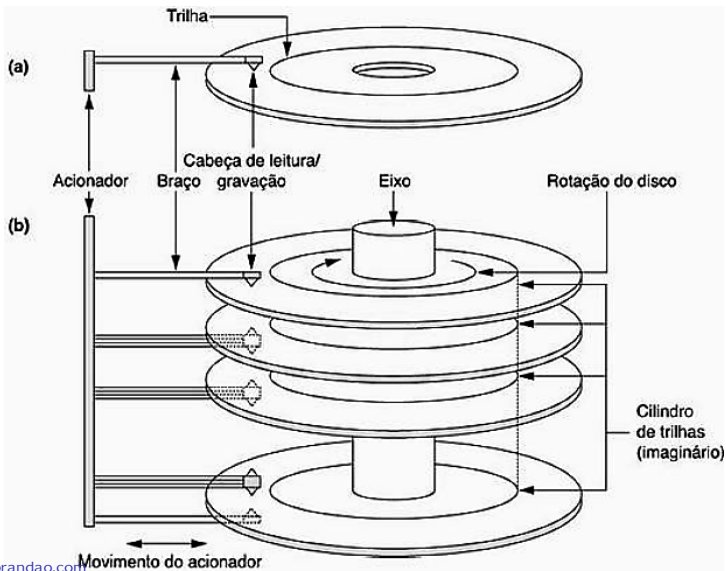


DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ A **capacidade** de um disco é o número de bytes que ele pode armazenar.
- ▶ Discos são feitos de um material magnético modelado como um disco circular fino e protegidos por uma camada de plástico ou acrílico.
- ▶ **Disco de face simples:** Armazena informações apenas em uma de suas superfícies.
- ▶ **Disco de face dupla:** As duas superfícies são usadas para armazenamento.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO





DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Para aumentar a capacidade de armazenamento, discos são montados em um **disk pack**.
- ▶ Um disk pack pode incluir muitos discos (muitas superfícies).
- ▶ Informações são armazenadas na superfície do disco em círculos de *pequena largura* (diferentes diâmetros).
- ▶ Cada círculo é chamado de **trilha**.
- ▶ O número de trilhas em um disco varia de centenas até milhares.
- ▶ A capacidade de cada trilha varia de dezenas de KB até 150 KB.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Há uma melhoria na capacidade de armazenamento, nas taxas de transferência e no custo associadas aos discos.
- ▶ Trilhas com o mesmo diâmetro nas superfícies de disk packs formam um **cilindro**.
- ▶ Dados armazenados em um cilindro podem ser recuperados mais rapidamente do que distribuídos entre diferentes cilindros.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Uma trilha é dividida em **blocos** ou **setores** menores.
- ▶ A divisão é fixada na superfície do disco e não pode ser alterada.
- ▶ Várias organizações de setor são possíveis.
- ▶ No entanto, nem todos os discos têm suas trilhas divididas em setores.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ A divisão de uma trilha em **blocos de disco** (**páginas**) de mesmo tamanho é definida pelo sistema operacional durante a **formatação** (**inicialização**) do disco.
- ▶ O tamanho do bloco é fixado na inicialização e não é trocado dinamicamente.
- ▶ Seu tamanho varia de 512 a 8.192 bytes.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Disco com setores fixos geralmente possuem setores subdivididos em blocos na inicialização.
- ▶ Os blocos são separados por **lacunas** de tamanho fixo.
- ▶ Lacunas entre blocos incluem **informações de controle** codificadas e gravadas durante a inicialização.
- ▶ Tais informações são utilizadas para determinar qual bloco da trilha segue cada lacuna entre blocos.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Um disco é um dispositivo endereçável por **acesso aleatório**.
- ▶ A transferência de dados entre a memória principal e o disco ocorre em *unidades de blocos de disco*.
- ▶ Um bloco de disco pode ser acessado *aleatoriamente* se especificamos seu endereço.
- ▶ O **endereço de hardware** de um bloco é fornecido ao hardware de E/S do disco.
- ▶ Em unidades de disco modernas, um número chamado LBA entre 0 e n é mapeado para o bloco correto pelo controlador de unidade de disco.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ **Buffer** é uma área reservada contígua no armazenamento principal, que mantém um bloco de disco.
- ▶ O *endereço de um buffer* também é fornecido ao hardware de E/S do disco.
- ▶ Vários blocos contíguos podem ser transferidos como uma unidade, denominada **cluster**.



COMANDO DE LEITURA

- ▶ O bloco de disco é copiado para o buffer.

COMANDO DE GRAVAÇÃO

- ▶ O conteúdo do buffer é copiado para o bloco de disco.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Cabeça de leitura/gravação do disco
 - ▶ Mecanismo de hardware que lê ou grava um bloco.
 - ▶ Faz parte de um sistema chamado **unidade de disco** (que inclui um motor que gira os discos).
 - ▶ Uma cabeça de leitura/gravação é conectada a um **braço mecânico**.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ *Disk packs* com superfícies múltiplas são controlados por várias cabeças de leitura/gravação.
- ▶ Braços são conectados a um acionador conectado a outro motor elétrico;
- ▶ Responsável por mover as cabeças de leitura/gravação e as posicionar sobre o cilindro de trilhas especificado em um endereço de bloco.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Unidades de disco rígido giram o disk pack continuamente a uma velocidade constante.
 - ▶ Com a cabeça de leitura/gravação posicionada na trilha correta e o bloco especificado no endereço de bloco;
 - ▶ move-se sob a cabeça de leitura/gravação;
 - ▶ e o componente eletrônico da cabeça de leitura/gravação é ativado para transferir os dados.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

► Discos de cabeça fixa

- Possuem cabeças de leitura/gravação fixas.
- O número de cabeças é correspondente ao de trilhas.
- Não ocorre movimento mecânico.
- Uma trilha ou cilindro é selecionado eletronicamente, passando para a cabeça de leitura/gravação apropriada.
- São mais ágeis, no entanto não são muito utilizados devido seu alto custo.

► Discos de cabeça móvel

- Unidades de disco com um acionador.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

► Controlador de disco

- Embutido na unidade de disco, é o responsável pelo seu controle.
- Interliga a unidade de disco ao sistema de computação.
- Aceita comandos de E/S de alto nível.
- Posiciona o braço e prepara a ação de leitura/gravação.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ **Tempo de busca**
 - ▶ A partir do endereço de um bloco de disco, o controlador de disco posiciona mecanicamente a cabeça de leitura/gravação na trilha correta.
- ▶ **Tempo exigido para a ação de transferência de um bloco de disco**
 - ▶ Tempos de busca típicos em desktops: 5 a 10ms
 - ▶ Tempos de busca típicos em servidores: 3 a 8ms



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ **Atraso rotacional ou latência**
 - ▶ Tempo que o bloco desejado gira até a posição sob a cabeça de leitura/gravação.
 - ▶ Relativo as *rpm* do disco.
- ▶ **Tempo de transferência de bloco**
 - ▶ Tempo adicional para transferir os dados.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Tempo total para localizar e transferir um bloco, a partir de seu endereço
 - ▶ É a soma do tempo de busca, do atraso rotacional e do tempo de transferência de bloco.
 - ▶ O tempo de busca e o atraso rotacional são muito maiores que o tempo de transferência do bloco.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ Para tornar a transferência de múltiplos blocos mais eficiente, transfere-se vários blocos consecutivos na mesma trilha ou cilindro.
- ▶ Eliminando o tempo de busca e o atraso rotacional para todos, menos para o primeiro bloco.
- ▶ **Taxa de transferência em massa**
 - ▶ Calcula-se o tempo exigido para transferir blocos consecutivos.



DESCRIÇÃO DE HARDWARE DOS DISPOSITIVOS DE DISCO

- ▶ O tempo necessário para localizar e transferir um bloco de disco está na ordem de milissegundos.
- ▶ É considerado alto em comparação com o tempo da memória principal.
- ▶ A localização dos dados no disco é um *gargalo principal* nas aplicações de banco de dados.
- ▶ Colocar "informações relacionadas" em blocos contíguos é o objetivo de qualquer organização de armazenamento no disco.



DISPOSITIVOS DE ARMAZENAMENTO DE FITA MAGNÉTICA

- ▶ Fitas magnéticas são dispositivos de acesso sequencial
 - ▶ Para acesso do n -ésimo bloco na fita é necessário varrer os $n-1$ blocos anteriores.
 - ▶ Bytes são armazenados consecutivamente em fitas.
 - ▶ Dados são armazenados em **bobinas de fita** magnética de alta capacidade.
 - ▶ Uma unidade de fita precisa ler os dados ou gravá-los em uma bobina de fita.



DISPOSITIVOS DE ARMAZENAMENTO DE FITA MAGNÉTICA

- ▶ Uma cabeça de leitura/gravação é usada para ler ou gravar dados na fita.
- ▶ Os registros de dados na fita são armazenados em **blocos**.
- ▶ Tais blocos são maiores do que os dos discos, no entanto, as lacunas entre blocos são muito grandes.
- ▶ Assim, agrupa-se muitos registros em um bloco para otimização do espaço.



DISPOSITIVOS DE ARMAZENAMENTO DE FITA MAGNÉTICA

- ▶ Em uma fita acessamos os blocos de dados em **ordem sequencial**.
- ▶ A fita é montada e depois varrida até que o bloco solicitado passe sob a cabeça de leitura/gravação.
- ▶ O acesso da fita pode ser lento, por esse motivo não são usadas para armazenar dados on-line.



DISPOSITIVOS DE ARMAZENAMENTO DE FITA MAGNÉTICA

- ▶ Fitas possuem uma função de **backup** do banco de dados.
- ▶ Com o intuito de manter cópias de arquivos de disco no caso de falhas.
- ▶ Arquivos de disco são copiados periodicamente para a fita.
- ▶ Fitas podem ser utilizadas para armazenar arquivos de banco de dados excessivamente grandes, raramente usados e desatualizados (**arquivados** em fita).



DISPOSITIVOS DE ARMAZENAMENTO DE FITA MAGNÉTICA

- ▶ Para aplicações críticas on-line, são usados **sistemas espelhados** para manter três conjuntos de discos idênticos.
 - ▶ Dois em operação on-line e um como backup.
 - ▶ Discos off-line tornam-se um dispositivo de backup.
 - ▶ Os três são usados de modo que possam ser trocados caso haja uma falha.

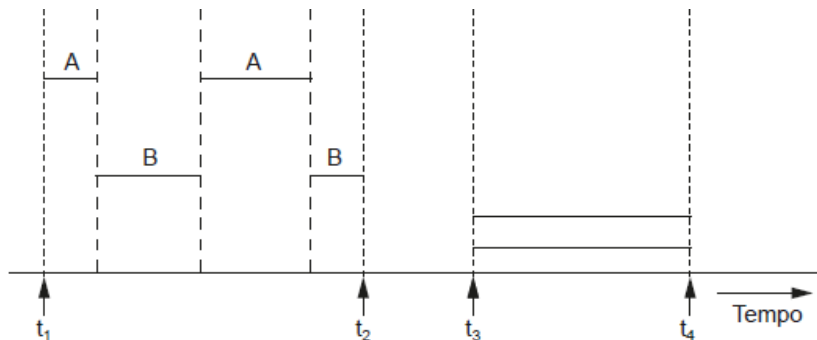


DISPOSITIVOS DE ARMAZENAMENTO DE FITA MAGNÉTICA

- ▶ O backup de bancos de dados corporativos é uma tarefa importante. Bibliotecas de fitas com *slots* centenas de cartuchos são usados como Digital e Superdigital Linear Tapes (DLTs e SDLTs);
- ▶ Com capacidades em gigabytes;
- ▶ Registram dados em trilhas lineares.



- ▶ Buffers podem ser reservados na memória principal para agilizar a transferência de blocos do disco para a memória principal.
- ▶ Existe um **processador (controlador) de E/S** de disco que transfere um bloco de dados entre a memória e o disco independente e em paralelo com o processamento da CPU.
- ▶ Assim, se um buffer está sendo lido ou gravado, a CPU pode processar dados em outro buffer.



Os processos A e B estão rodando **simultaneamente** em um padrão **intervalado**.

Os processos C e D estão rodando **simultaneamente** em um padrão **paralelo**.



- ▶ Se a CPU controla vários processos, a execução paralela não é possível.
- ▶ Contudo, processos podem ser executados simultaneamente de forma intervalada.
- ▶ O buffering é mais útil quando os processos podem ser executados simultaneamente em um padrão paralelo.
- ▶ Seja por um processador de E/S de disco separado ou por vários processadores (CPUs).



BUFFERING DUPLO

- ▶ Uso de dois buffers para leitura do disco.
- ▶ Leitura e processamento prosseguem em paralelo.
- ▶ A CPU pode 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

- ▶ Permite a leitura ou gravação contínua de dados em blocos de disco consecutivos.
- ▶ Elimina o tempo de busca e o atraso rotacional para todas as transferências de bloco, com exceção da primeira.
- ▶ Os dados ficam prontos para processamento, reduzindo o tempo de espera nos programas.



REGISTROS

- ▶ Dados são armazenados na forma de registros.
- ▶ Cada registro contém uma coleção de valores ou itens de dados relacionados, que descrevem entidades e seus atributos.
- ▶ Cada valor é formado por bytes e corresponde a um campo em particular do registro.

TIPOS DE REGISTRO OU FORMATO DE REGISTRO

- ▶ Coleção de nomes de campo e seus tipos de dados correspondentes.



TIPO DE DADO

- ▶ Especifica os tipos de valores que um campo pode assumir, incluem:
 - ▶ **Tipos de dados numéricos:** inteiro, inteiro longo ou ponto flutuante.
 - ▶ **Cadeia de caracteres:** tamanho fixo ou variável.
 - ▶ **Booleanos:** valores 0 e 1, ou TRUE e FALSE.
 - ▶ **Data e tempo,** especialmente codificados.



TIPO DE DADO

- ▶ O número de bytes para cada tipo de dado é fixo para determinado sistemas de computação.
 - ▶ **Inteiro e Número real:** 4 bytes
 - ▶ **Inteiro longo:** 8 bytes
 - ▶ **Booleano:** 1 byte
 - ▶ **Data:** 10 bytes (formato DD-MM-AAAA)
 - ▶ **String de tamanho fixo de k caracteres:** k bytes



OBJETOS BINÁRIOS GRANDES (BLOBs)

- ▶ Grandes objetos não estruturados.
- ▶ Imagens, vídeo digitalizado ou streams de áudio, ou então texto livre.
- ▶ Um item de dados BLOB é armazenado separadamente de seu registro, em um conjunto de blocos de disco.
- ▶ Um ponteiro para o BLOB é incluído no registro.



ARQUIVOS

- ▶ Um **arquivo** é uma *sequência* de registros.
- ▶ Geralmente, todos os registros em um arquivo são do mesmo tipo.

ARQUIVO COMPOSTO DE REGISTROS DE TAMANHO FIXO

- ▶ Cada registro no arquivo tem o mesmo tamanho (em bytes).



ARQUIVO COMPOSTO DE REGISTROS DE TAMANHO VARIÁVEL

- ▶ Diferentes registros no arquivo possuem diversos tamanhos.
 - ▶ Os registros do arquivo são do mesmo tipo, mas possuem **campos de tamanho variável**.
 - ▶ *Por exemplo:* O campo **Nome** de **FUNCIONARIO** possui tamanho variável.



ARQUIVO COMPOSTO DE REGISTROS DE TAMANHO VARIÁVEL

- ▶ Diferentes registros no arquivo possuem diversos tamanhos.
 - ▶ Os registros do arquivo são do mesmo tipo, mas alguns campos possuem múltiplos valores para registros individuais (**campo repetitivo**).
 - ▶ Um grupo de valores para o campo é chamado de **grupo repetitivo**.



ARQUIVO COMPOSTO DE REGISTROS DE TAMANHO VARIÁVEL

- ▶ Diferentes registros no arquivo possuem diversos tamanhos.
 - ▶ Os registros do arquivo são do mesmo tipo, mas possuem **campos opcionais** (podem ter valores para alguns, mas não para todos os registros).

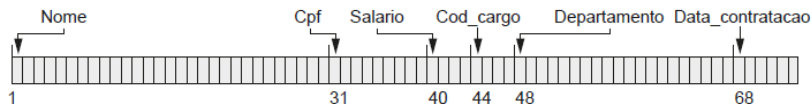


ARQUIVO COMPOSTO DE REGISTROS DE TAMANHO VARIÁVEL

- ▶ Diferentes registros no arquivo possuem diversos tamanhos.
 - ▶ O arquivo contém registros de *tipos de registro diferentes*, portanto de tamanho variável (arquivo misto).
 - ▶ Registros relacionados de diferentes tipos *agrupados* em blocos de disco.
 - ▶ *Exemplo*: Registros de **HISTORICO_ESCOLAR** de determinado aluno podem ser colocados após o registro desse **ALUNO**.



FORMATAÇÃO DE REGISTROS DE UM ARQUIVO DE TAMANHO FIXO



Registro de tamanho fixo com seis campos e tamanho de 71 bytes

- ▶ É possível representar um arquivo que deveria ter registros de tamanho variável como um arquivo de registros de tamanho fixo.
- ▶ Poderíamos incluir em cada registro de arquivo os campos opcionais, mas armazenar o valor NULL se necessário.
- ▶ Para um campo repetitivo, poderíamos alocar espaços em cada registro do número máximo possível de ocorrências do campo.



FORMATAÇÃO DE REGISTROS DE UM ARQUIVO DE TAMANHO VARIÁVEL

Nome	Cpf	Salario	Cod_cargo	Departamento	
Silva, João	12345678966	XXXX	XXXX	Computação	Caracteres separadores
1	12	21	25	29	

Registro com dois campos de tamanho variável e três campos de tamanho fixo

- ▶ Usa-se caracteres **separadores** especiais (? ou % ou \$), para determinar os bytes que representa cada campo;
- ▶ Para terminar os campos de tamanho variável;
- ▶ Ou para armazenar o tamanho do campo em bytes no próprio registro (antes do valor do campo).



FORMATAÇÃO DE REGISTROS DE UM ARQUIVO COM CAMPOS OPCIONAIS

Nome = Silva, João	Cpf = 12345678966	DEPARTAMENTO = Computação	⌘
			Caracteres separadores
			= Separa nome de campo do valor do campo
			█ Separa campos
			⌘ Termina registro

- ▶ Se o número de campos for pequeno, podemos incluir em cada registro <nome-campo, valor-campo>.
- ▶ Usa-se o mesmo caractere separador para o nome do campo do valor e para um campo do campo seguinte.
- ▶ Atribui-se um **tipo de campo** a cada campo, <tipo-campo, valor-campo>.
- ▶ Um campo repetitivo precisa de um caractere separador para afastar valores repetidos e para indicar o término.



FORMATAÇÃO DE UM ARQUIVO COM REGISTROS DE DIFERENTES TIPOS

- ▶ Cada registro é precedido por um indicador de tipo de registro.



BLOCAGEM DE REGISTROS ESPALHADOS *versus* NÃO ESPALHADOS

- ▶ Bloco é a *unidade de transferência de dados* entre o disco e a memória.
- ▶ Registros de um arquivo precisam ser alocados a blocos de disco.



BLOCAGEM DE REGISTROS ESPALHADOS *versus* NÃO ESPALHADOS

- ▶ *Exemplo:*
 - ▶ Bloco com tamanho B bytes
 - ▶ Arquivo de registros de tamanho fixo, R bytes
 - ▶ Sendo $B > R$, estabelecemos $\text{bfr} = \lfloor B/R \rfloor$ registros por bloco
 - ▶ Onde $\lfloor (x) \rfloor$ (função floor) arredonda para baixo o número x para um inteiro
 - ▶ O valor **bfr** é chamado de **fator de blocagem** para o arquivo
 - ▶ Em geral, R pode não dividir B exatamente, de modo que temos algum espaço não usado igual a $B - (\text{bfr} * R)$ bytes



BLOCAGEM DE REGISTROS ESPALHADOS *versus* NÃO ESPALHADOS

► Organização espalhada

- Armazenar parte de um registro em um bloco e o restante em outro.
- Um **ponteiro** no final do primeiro bloco aponta para o bloco que contém o restante do registro.



BLOCAGEM DE REGISTROS ESPALHADOS *versus* NÃO ESPALHADOS

- ▶ Organização não espalhada
 - ▶ Registros que não podem atravessar os limites de bloco.
 - ▶ Usado para registros de tamanho fixo tendo $B > R$.
 - ▶ Para registros de tamanho variável, pode-se usar uma organização espalhada ou não espalhada.
 - ▶ Se o registro médio for grande, é vantajoso usar o espalhamento para reduzir o espaço perdido em cada bloco.

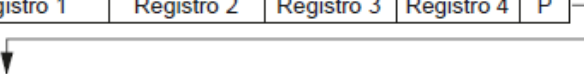


BLOCAGEM DE REGISTROS ESPALHADOS VERSUS NÃO ESPALHADOS

Bloco i	Registro 1	Registro 2	Registro 3	
---------	------------	------------	------------	--

Bloco i + 1	Registro 4	Registro 5	Registro 6	
-------------	------------	------------	------------	--

Bloco i	Registro 1	Registro 2	Registro 3	Registro 4	P
---------	------------	------------	------------	------------	---



Bloco i + 1	Registro 4 (resto)	Registro 5	Registro 6	Registro 7	P
-------------	--------------------	------------	------------	------------	---

A - Organização de registro não espalhada.

B - Organização de registro espalhada.



BLOCAGEM DE REGISTROS ESPALHADOS *versus* NÃO ESPALHADOS

- ▶ Para registros de tamanho variável com organização espalhada, cada bloco pode armazenar um número diferente de registros.
- ▶ O fator de bloco *bfr* representa o número *médio* de registros por bloco para o arquivo.
- ▶ Podemos usar *bfr* para calcular o número de blocos *b* necessários para um arquivo de *r* registros:
 - ▶ $b = \lceil (r/bfr) \rceil$ blocos
 - ▶ $\lceil (x) \rceil$ (função **ceiling**) arredonda para cima o valor de *x* até o próximo inteiro.



ALOCANDO BLOCOS DE ARQUIVO NO DISCO

Técnicas-padrão para alocar os blocos de um arquivo no disco.

- ▶ **Alocação contígua**

- ▶ Blocos de arquivo são alocados a blocos de disco consecutivos.
- ▶ Leitura do arquivo torna-se rápida usando o buffering duplo.
- ▶ A expansão do arquivo é dificultada.



ALOCANDO BLOCOS DE ARQUIVO NO DISCO

Técnicas-padrão para alocar os blocos de um arquivo no disco.

- ▶ **Alocação ligada**

- ▶ Cada bloco de arquivo contém um ponteiro para o próximo bloco.
- ▶ Facilita a expansão do arquivo.
- ▶ Torna a leitura do arquivo mais lenta.



ALOCANDO BLOCOS DE ARQUIVO NO DISCO

Técnicas-padrão para alocar os blocos de um arquivo no disco.

- ▶ **Clusters (segmentos ou extensões de arquivo)**
 - ▶ Uma combinação das duas técnicas aloca clusters de blocos de disco consecutivos, e os clusters são ligados.
- ▶ **Alocação indexada**
 - ▶ Um ou mais blocos de índice contêm ponteiros para blocos de arquivos reais.
 - ▶ Também é comum usar combinações dessas técnicas.



CABEÇALHOS DE ARQUIVO

Cabeçalho de arquivo ou descritor de arquivo

- ▶ Contém informações sobre um arquivo (exigidas pelos programas que acessam os registros do arquivo).
- ▶ O cabeçalho inclui informações para:
 - ▶ Determinar endereços de disco dos blocos de arquivo;
 - ▶ Registrar descrições de formato (tamanhos de campo e a ordem dos campos em um registro);
 - ▶ Para registros não espalhados de tamanho fixo;
 - ▶ E para códigos de tipo de campo, caracteres separadores e códigos de tipo de registro (para registros de tamanho variável).



PESQUISA LINEAR

- ▶ Realizada pelos programas de pesquisa, nos blocos de arquivo, quando o endereço do bloco que contém o registro desejado não for conhecido.
- ▶ Cada bloco de arquivo é copiado para um buffer e pesquisado até que o registro seja localizado utilizando informação do cabeçalho (e todos os blocos pesquisados).
- ▶ Tal processo pode ser demorado para um arquivo grande.
- ▶ Uma boa organização de arquivo localiza um bloco desejado com um número mínimo de transferências de bloco.



ORGANIZAÇÃO DE ARQUIVO

- ▶ Organização dos dados de um arquivo em registros, blocos e estruturas de acesso.
- ▶ Inclui o modo como registros e blocos são colocados no meio de armazenamento e interligados.
- ▶ Uma organização de arquivo bem-sucedida realiza de maneira eficiente as operações que esperamos *aplicar frequentemente* ao arquivo.



MÉTODO DE ACESSO

- ▶ Oferece um grupo de operações que podem ser aplicadas a um arquivo.
- ▶ É possível aplicar vários métodos de acesso a uma organização de arquivo.
- ▶ Alguns métodos de acesso só podem ser aplicados a arquivos organizados de certas maneiras.



Agrupadas em **operações de recuperação** e **operações de atualização**.

OPERAÇÕES DE RECUPERAÇÃO

- ▶ Não altera dados no arquivo.
- ▶ Localiza certos registros de modo que seus valores de campo possam ser examinados e processados.

OPERAÇÕES DE ATUALIZAÇÃO

- ▶ Altera o arquivo pela inserção, exclusão de registros ou modificação dos valores de campo.



CONDIÇÃO DE SELEÇÃO (OU CONDIÇÃO DE FILTRAGEM)

- ▶ Especifica critérios que o registro deve satisfazer.
- ▶ No momento de **selecionar** um ou mais registros para recuperação, exclusão ou modificação.
- ▶ **Condição de seleção simples** envolve uma comparação de igualdade em algum valor de campo. Exemplos:
 - ▶ `Cpf = '1234567899'`
 - ▶ `Salario > 30.000`
- ▶ O caso geral é ter uma expressão booleana nos campos do arquivo como condição de seleção.



OPERAÇÕES DE PESQUISA

- ▶ Geralmente baseadas em condições de seleção simples.
- ▶ Utilizadas para localizar os registros no disco.
- ▶ Condições complexas são decompostas (pelo SGBD ou pelo programador).
- ▶ Cada registro localizado é verificado para determinar se satisfaz a condição de seleção inteira.



OPERAÇÕES DE PESQUISA

- ▶ Pode-se extrair a condição simples
(Departamento = 'Pesquisa')
- ▶ Da condição complexa
((Salario > 30.000) AND (Departamento = 'Pesquisa'));



OPERAÇÕES DE PESQUISA

- ▶ Se vários registros de arquivo satisfazem uma condição de pesquisa, o *primeiro* registro localizado é designado como **registro atual**.
- ▶ Operações de busca subsequentes começam desse registro e localizam o *próximo* registro no arquivo que satisfaz a condição.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

- ▶ Programas de software de SGBD acessam registros utilizando operações representativas.
- ▶ Estas (exceto por *Open* e *Close*) são operações de **um registro por vez**, pois cada uma se aplica a um único registro.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

Open

- ▶ Prepara o arquivo para a leitura ou gravação.
- ▶ Aloca buffers apropriados (pelo menos dois) para manter blocos de arquivo do disco.
- ▶ Recupera o cabeçalho do arquivo.
- ▶ Define o ponteiro de arquivo para o início do arquivo.

Reset

- ▶ Define o ponteiro do arquivo aberto para o início do arquivo.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

Find (ou Locate)

- ▶ Procura o primeiro registro que satisfaça uma condição.
- ▶ Transfere o bloco que contém esse registro para o buffer da memória principal.
- ▶ O ponteiro de arquivo aponta para o registro no buffer (registro atual).
- ▶ Diferentes verbos indicam se o registro localizado deve ser lido ou atualizado.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

Read (ou Get)

- ▶ Copia o registro atual do buffer para uma variável de programa no programa do usuário.
- ▶ Avança o ponteiro do registro atual para o próximo registro no arquivo.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

FindNext

- ▶ Procura o próximo registro no arquivo que satisfaz a condição de pesquisa.
- ▶ Em seguida, transfere o bloco que contém esse registro para um buffer na memória principal (registro atual).
- ▶ Várias formas de FindNext estão disponíveis em SGBDs legados com base nos modelos hierárquicos e de rede.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

Delete

- ▶ Exclui o registro atual e atualiza o arquivo no disco para refletir a exclusão.

Modify

- ▶ Modifica alguns valores de campo para o registro atual e atualiza o arquivo no disco para refletir a modificação.

Close

- ▶ Completa o acesso ao arquivo liberando os buffers e realizando outras operações de limpeza.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

Insert

- ▶ Insere um novo registro no arquivo ao localizar o bloco onde o registro deve ser inserido.
- ▶ Transfere tal bloco para um buffer da memória principal, gravando o registro no buffer e gravando o buffer em disco para refletir a inserção.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

É possível resumir as operações *Find*, *FindNext* e *Read* na operação **Scan**.

Scan

- ▶ Se o arquivo já tiver sido aberto ou reiniciado, Scan retorna o primeiro registro.
- ▶ Caso contrário, ele retorna o próximo registro.
- ▶ Se uma condição for especificada com a operação, o registro retornado é o primeiro ou o próximo registro que satisfaz a condição.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

Operações adicionais de nível mais alto, de **um conjunto de cada vez**, podem ser aplicadas a um arquivo. Exemplos:

- ▶ FindAll
- ▶ Find (ou Locate)
- ▶ FindOrdered
- ▶ Reorganize



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

FindAll

- ▶ Localiza todos os registros no arquivo que satisfazem uma condição de pesquisa.

Find (ou Locate) n

- ▶ Procura o primeiro registro que satisfaz uma condição e depois continua a localizar os próximos $n-1$ registros que satisfazem a mesma condição.
- ▶ Transfere os blocos que contém os n registros para o buffer da memória principal.



CONJUNTO DE OPERAÇÕES REPRESENTATIVAS

FindOrdered

- ▶ Recupera todos os registros no arquivo em alguma ordem especificada.

Reorganize

- ▶ Inicia o processo de reorganização.



ARQUIVOS ESTÁTICOS

- ▶ Operações de atualização raramente são realizadas.

ARQUIVOS DINÂMICOS

- ▶ Podem mudar com frequência, de modo que as operações de atualizações são constantemente aplicadas.



- ▶ O agrupamento de registros em blocos e organização de blocos em cilindros é de acordo com a situação.
- ▶ Em muitos casos, uma única organização não permite que todas as operações necessárias sejam implementadas eficientemente, requer que seja escolhido um meio-termo.



ORGANIZAÇÃO ARQUIVO DE HEAP OU PILHA

- ▶ Tipo de organização mais simples e mais básica.
- ▶ Normalmente, usada com caminhos de acesso adicionais.
- ▶ Também é usada para coletar e armazenar registros de dados para uso futuro.
- ▶ Registros são arquivados na ordem em que são inseridos (novos registros são inseridos ao final do arquivo).



- ▶ Inserção de um novo registro é feita de maneira eficiente.
- ▶ O último bloco de disco do arquivo é copiado para um buffer, o novo registro é acrescentado e o bloco é regravado de volta.
- ▶ O endereço do último bloco de arquivo é mantido no cabeçalho do arquivo.



PESQUISA LINEAR

- ▶ Envolve procurar um registro usando uma condição de pesquisa, pelo bloco de arquivo por bloco.
- ▶ Se apenas um registro satisfazer a condição de pesquisa, um programa lerá a memória e pesquisará metade dos blocos de arquivo antes de encontrar (em média).
- ▶ Para um arquivo de b blocos, isso exige pesquisar $(b/2)$ blocos, na média.
- ▶ Se nenhum registro satisfazer a condição de pesquisa, o programa deve ler e pesquisar todos os b blocos no arquivo.



EXCLUSÃO DE UM REGISTRO

- ▶ Um programa encontra o bloco, o copia para o buffer, exclui o registro do buffer e **regrava** o bloco de volta ao disco.
- ▶ A exclusão de registros resulta em vários espaços desperdiçados.



EXCLUSÃO COM MARCADOR DE EXCLUSÃO

- ▶ Marcador de exclusão é um byte ou bit extra, armazenado em cada registro.
- ▶ Um registro é excluído ao se definir o marcador de exclusão com determinado valor.
- ▶ Um valor diferente para o marcador indica um registro válido (não excluído).
- ▶ Programas de pesquisa consideram apenas registros válidos.



- ▶ As duas técnicas citadas exigem **reorganização** periódica do arquivo para retomar o espaço não usado.
- ▶ Na reorganização, os blocos de arquivo são acessados de maneira consecutiva e os registros são compactados pela remoção de registros excluídos.
- ▶ Após a organização, os blocos são preenchidos até a capacidade.
- ▶ Outra possibilidade é usar o espaço dos registros excluídos ao inserir novos registros.



- ▶ Usamos a organização espalhada ou não espalhada para um arquivo desordenado, com registros de tamanho fixo ou variável.
- ▶ A modificação de um registro de tamanho variável exige a exclusão do registro antigo e a inserção de um registro modificado.
- ▶ Para ler todos os registros na ordem dos valores de algum campo, criamos uma cópia classificada do arquivo.
- ▶ Por ser uma operação de custo alto, técnicas especiais para **classificação externa** são utilizadas.



Para um arquivo de *registros de tamanho fixo* desordenados usando *blocos não espalhados e alocação contígua*, é simples acessar qualquer registro por sua **posição** no arquivo.

ARQUIVO RELATIVO OU DIRETO

- ▶ Registros podem ser facilmente acessados por suas posições relativas.
- ▶ Não auxilia a localizar um registro com base em uma condição de busca.
- ▶ Facilita a construção de caminhos de acesso no arquivo, como os índices.



Os arquivos ordenados estão em blocos armazenados em cilindros contíguos para minimizar o tempo de busca.

Realizada a partir de uma pesquisa binária nos blocos dos arquivos de disco.

- ▶ Acesso de $\log_2(b)$ blocos;
- ▶ Melhoria se comparada a pesquisa linear, onde são acessados:
 - ▶ $(b/2)$ blocos, quando o registro é encontrado;
 - ▶ b blocos, quando o registro não é encontrado.



Tempos de acesso médios para um arquivo de b blocos:

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

- ▶ Um critério de pesquisa envolvendo as condições $>$, $<$, \geq e \leq no campo de ordenação é muito eficiente;
 - ▶ A ordenação física dos registros indica que todos que satisfazem a condição são contíguos no arquivo.
- ▶ Grande dificuldade na prática de **inserção, exclusão e alteração**.



DIFICULDADE NA PRÁTICA DE INSERÇÃO

1. Para inserir, deve-se encontrar sua posição correta no arquivo, com base no campo de ordenação;
2. Criar um espaço no arquivo para inserir registro na posição definida acima;
3. "Deslocar" os registros sucessivos para as posições consecutivas.

Para um arquivo grande, é um processo demorado, uma vez que metade dos blocos do arquivo deve ser lida e regravada após o deslocamento dos registros.



ALTERNATIVAS PARA EFICIÊNCIA DA INSERÇÃO

1. Manter algum espaço não usado em cada bloco para novos registros;
 - ▶ O problema original reaparece ao ser totalmente utilizado.



ALTERNATIVAS PARA EFICIÊNCIA DA INSERÇÃO

2 Criar um arquivo desordenado temporário, chamado de **overflow** ou **transação**.

- ▶ Arquivo ordenado real chamado de **principal** ou **mestre**;
- ▶ Funcionamento:
 - ▶ Os novos registros que deveriam ser inseridos ao mestre, são inseridos no final do overflow;
 - ▶ De tempos em tempos, o arquivo de overflow é classificado e mesclado ao mestre.



ALTERNATIVAS PARA EFICIÊNCIA DA INSERÇÃO

2 Criar um arquivo desordenado temporário, chamado de **overflow** ou **transação**.

- ▶ Maior complexidade no algoritmo de pesquisa;
 - ▶ O overflow precisa ser consultado (pesquisa linear) quando, após uma pesquisa binária, o registro não tiver sido encontrado no mestre.
- ▶ O overflow pode ser ignorado em aplicações que não exigem informações mais atualizadas.



A complexidade da **alteração** de um registro, depende de:

1. Condição de pesquisa para localizar o registro
2. O campo a ser modificado

[FATOR 1] A condição de pesquisa envolve o campo chave de ordenação?

- ▶ Sim: Utiliza-se a pesquisa binária.
- ▶ Não: Utiliza-se a pesquisa linear.



A complexidade da **alteração** de um registro, depende de:

1. Condição de pesquisa para localizar o registro
2. O campo a ser modificado

[FATOR 2] O campo a ser alterado é o campo ordenado do arquivo ?

- ▶ Sim: Dependendo do novo valor, este campo pode ter sua posição física alterada no arquivo.
 - ▶ Requer a exclusão do registro antigo e a inserção do registro modificado.
- ▶ Não: A modificação deste campo é realizada e este é regravado no mesmo local físico do disco.



Aplicações de banco de dados, normalmente, utilizam arquivos ordenados com caminho de acesso adicional (**índice primário**), chamados de:

ARQUIVO SEQUENCIAL-INDEXADO

- ▶ Otimiza o tempo de acesso aleatório ao campo-chave de ordenação;
- ▶ Se o atributo de ordenação não for uma chave, o arquivo é chamado de **arquivo agrupado**.



- ▶ Outro tipo de organização do arquivo principal;
- ▶ Acesso muito eficiente aos registros sob determinadas condições de pesquisa.

ARQUIVO DE HASH

- ▶ A condição de pesquisa precisa ser uma condição de igualdade em um único campo: **campo de hash**
 - ▶ Na maior parte dos casos também é um campo-chave, chamado de **chave hash**.



ARQUIVO DE HASH - FUNCIONAMENTO

- ▶ Oferece uma função de hash h , onde se aplica o valor do campo de hash de um registro e gera o endereço do bloco de disco em que o registro está armazenado;
- ▶ A pesquisa pode ser realizada pelo buffer da memória principal.

Uma boa função de hashing distribui os registros de maneira uniforme pelo espaço de endereços de modo a **minimizar as colisões enquanto não deixam muitos locais não usados**.



HASH INTERNO

- ▶ Implementado através de uma **tabela de hash** por um array de registros;
- ▶ Exemplo de estrutura de dados de hashing interno - array de M posições para uso no hashing:

	Nome	Cpf	Cargo	Salario
0				
1				
2				
3				
	...			
M-2				
M-1				



HASH INTERNO - EXEMPLO

- ▶ Uma função hash comum seria:

$$h(K) = K \bmod M$$

Retorna o resto de um valor de campo de hash inteiro K após a divisão por M : esse valor é usado para o endereço do registro.



HASH INTERNO - COLISÃO

- ▶ Ocorre quando, na inserção, a função de um campo de hash resulta em um endereço já utilizado.
- ▶ É necessário localizar outra posição disponível:
 - ▶ Resolução de colisão
- ▶ Métodos para resolução de colisão:
 1. Endereçamento Aberto
 2. Encadeamento
 3. Hashing múltiplo
 - ▶ Cada método requer os próprios algoritmos para inserção, recuperação e exclusão de registros.



HASH INTERNO - COLISÃO (MÉTODOS DE RESOLUÇÃO)

1. Endereçamento Aberto

- ▶ Verifica as posições subsequentes, a partir da posição ocupada, até que uma posição vazia seja localizada.

2. Encadeamento

- ▶ É mantida uma lista composta por registros de overflow para cada endereço de hash.

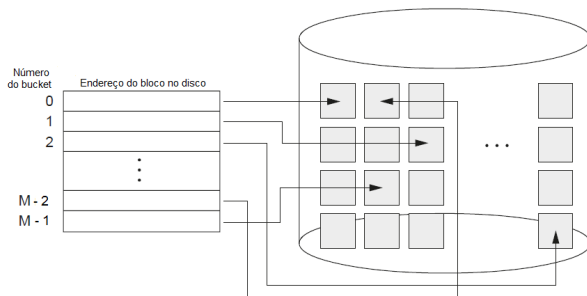
3. Hashing múltiplo

- ▶ Aplica-se uma segunda função hash se a primeira resultar em uma colisão.



HASH EXTERNO

- ▶ Destinado a arquivos de disco;
- ▶ O espaço de endereços de destino é feito em *buckets*, cada um mantendo vários registros.
 - ▶ **Bucket:** bloco de disco ou cluster de blocos de discos contíguos.





HASH EXTERNO - FUNCIONAMENTO

1. A função de hashing mapeia uma chave em um número de bucket;
2. Uma tabela mantida no cabeçalho do arquivo converte o número do bucket para o endereço do bloco de disco correspondente.



HASH EXTERNO - COLISÃO

- ▶ Menos complexa;
 - ▶ Independentemente de quantos registros estejam em um bucket, eles podem ser definidos por hashing ao mesmo bucket sem causar problemas.
- ▶ É previsto o caso em que um bucket está cheio e um novo registro a ser inserido tem um hash para esse bucket.
 - ▶ Utiliza-se a técnica semelhante a de **Encadeamento**:
 - ▶ Os ponteiros na lista devem ser "ponteiros de registro", que incluem um endereço de bloco e uma posição de registro relativa no bloco.



HASH EXTERNO - ESTÁTICO

- ▶ Funções definidas, partindo do princípio que há um número fixo de buckets **M** alocado.
- ▶ Desvantagem para arquivos dinâmicos - *Exemplo*:
 1. Alocamos **M** buckets para o espaço de endereços e deixamos o número máximo **m** de registros que podem caber em um bucket. Ou seja, caberão $m * M$ registros no espaço alocado. Caso o número de registros for muito menor ou muito maior do que $m * M$, é necessário alterar o número de blocos **M** alocados e depois usar uma nova função de hashing para redistribuir os registros.
 - ▶ Reorganização muito demorada para arquivos grandes.



HASH EXTERNO - BUSCA DE REGISTRO

- ▶ Se for buscado um registro em um campo diferente do campo de hash, o comportamento da pesquisa será semelhante a um arquivo desordenado.

HASH EXTERNO - EXCLUSÃO DE REGISTRO

- ▶ É realizada a partir da remoção do próprio registro em seu bucket.
 - ▶ Se o bucket tiver uma cadeia de overflow:
 1. Mover um dos registros de overflow para os excluídos;
 2. Substituir o registro excluído.
 - ▶ Se o registro a ser excluído já tiver no overflow:
 1. Remover o registro da lista.



HASH EXTERNO - ALTERAÇÃO DE REGISTRO

- ▶ Depende de:
 1. Condição de pesquisa para localizar o registro
 2. O campo a ser modificado

[FATOR 1] É uma comparação de igualdade no campo de hash?

- ▶ Sim: Utiliza-se a função de hash.
- ▶ Não: Utiliza-se a pesquisa linear.



HASH EXTERNO - ALTERAÇÃO DE REGISTRO

- ▶ Depende de:
 1. Condição de pesquisa para localizar o registro
 2. O campo a ser modificado

[FATOR 2] O campo a ser alterado é o campo de hash?

- ▶ Sim: Dependendo do novo valor, este campo pode se mover para outro bucket.
 - ▶ Requer a exclusão do registro antigo e a inserção do registro modificado.
- ▶ Não: O campo é modificado e gravado no mesmo bucket.



TÉCNICAS PARA EXPANSÃO DINÂMICA

- ▶ Técnicas de hashing que permitem a expansão dinâmica do arquivo:
 1. Hashing extensível
 2. Hashing dinâmico
 3. Hashing linear
- ▶ O resultado da aplicação de uma função de hashing é **um inteiro não negativo e pode ser representado como um número binário**;
- ▶ Estrutura de acesso feita em "representação binária" do resultado da função de hashing (string de "bits").

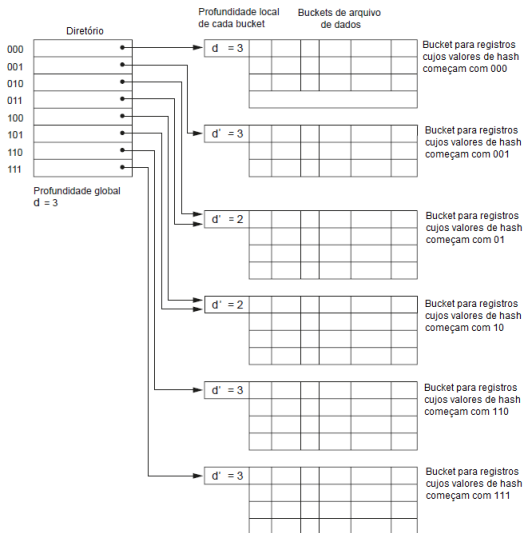


TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING EXTENSÍVEL

- ▶ Um array de 2^d endereços de bucket é mantido, chamado de **diretório**.
 - ▶ d é chamado de profundidade global do diretório.
- ▶ O valor inteiro correspondente aos primeiros d' bits de um valor de hash é utilizado como um índice para o array determinar uma entrada de diretório e o endereço nessa entrada determina o bucket em que os registros correspondentes estão armazenados;
- ▶ Uma profundidade local d' especifica o número de bits em que os conteúdos dos buckets são baseados.



TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING EXTENSÍVEL





TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING EXTENSÍVEL

- ▶ O valor de d pode ser aumentado ou diminuído, dobrando ou reduzindo à metade do número de entradas no array do diretório;
- ▶ A maioria das recuperações de registro exige dois acessos de bloco - um para o diretório e outro para o bucket;
- ▶ Desvantagem:
 - ▶ O diretório precisa ser pesquisado antes do acesso aos próprios buckets, resultando em dois acessos ao bloco.



TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING EXTENSÍVEL

- ▶ Vantagens:
 - ▶ O desempenho da busca não degrada enquanto o arquivo cresce;
 - ▶ Nenhum espaço é alocado no hashing extensível para crescimento futuro
 - ▶ Buckets adicionais podem ser alocados de maneira dinâmica conforme a necessidade;
 - ▶ O tamanho máximo do diretório é 2^k , onde k é o número de bits no valor de hash.



TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING DINÂMICO

- ▶ Armazenamento de registros em buckets semelhante ao hashing extensível.
 - ▶ Se difere na organização do diretório: ao invés de profundidade global, utiliza-se um diretório estruturado em árvore com dois tipos de nós:
 1. **Nós internos com dois ponteiros:** O ponteiro esquerdo representa o bit 0 e o ponteiro direito representa o bit 1.
 2. **Nós folha:** Mantém um ponteiro para o bucket real com registros.

Nó de diretório interno

Nó de diretório de folha

Buckets de arquivo de dados

Bucket para registros cujos valores de hash começam com 000

Bucket para registros cujos valores de hash começam com 001

Bucket para registros cujos valores de hash começam com 01

Bucket para registros cujos valores de hash começam com 10

Bucket para registros cujos valores de hash começam com 110

Bucket para registros cujos valores de hash começam com 111

Diretório:



TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING LINEAR

- ▶ Permite que um arquivo de hash expanda e encolha seu número de buckets dinamicamente sem precisar de um diretório.
- ▶ **Funcionamento:**
 1. O arquivo começa com **M** buckets (0, 1, ..., M-1) e usa a função $h(K) = K \bmod M$ - função de hash inicial (h_i);
 2. Quando a colisão leva a um registro de estouro em qualquer bucket de arquivo, o primeiro bucket (0) é dividido em dois: o bucket original (0) e um novo bucket (M) ao final do arquivo.



TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING LINEAR

► Funcionamento:

- 3 Os registros do bucket 0 são distribuídos entre dois buckets em uma função diferente $h_{i+1}(K) = K \bmod 2^M$.
 - *A medida que mais colisões levem registros a overflow, buckets adicionais são divididos na ordem linear 1, 2, 3...*
- 4 Os registros do overflow são redistribuídos em buckets regulares, usando a função h_{i+1} , por meio de uma *divisão adiada* dos buckets.
 - *Sempre que ocorrer uma divisão, um inteiro n é incrementado por 1, para ao final, determinar a quantidade de buckets divididos.*



TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING LINEAR

- ▶ **Passos** - Recuperar um valor de chave hash K :
 1. Aplicar a função h_i ;
 2. Se $h_i(K) < n$, aplicar a função h_{i+1} (porque o bucket já está dividido).

Quando $n = M$, significa que todos os buckets originais foram divididos e a função hash h_{i+1} já não é mais suficiente. Deste modo, cria-se a $h_{i+2}(K) = K \bmod 4^M$.



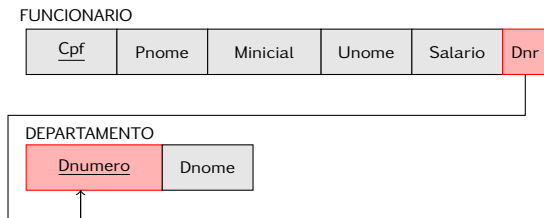
TÉCNICAS PARA EXPANSÃO DINÂMICA - HASHING LINEAR

- ▶ **Passos** - Recuperar um valor de chave hash K:
 - ▶ A divisão pode ser controlada ao monitorar o fator de carga do arquivo (l) ao invés de dividir sempre que ocorrer um overflow.
 - ▶ Definido como $l = r / (bfr * N)$, onde
 - r = número atual de registros do arquivo;
 - bfr = número máximo de registros que cabem no bucket;
 - N = número atual de buckets do arquivo.
 - ▶ Vantagem:
 - Mantém o fator de carga razoavelmente constante enquanto o arquivo aumenta e diminui e não requer um diretório.



ARQUIVOS DE REGISTROS MISTOS

- ▶ Na maioria das aplicações de banco de dados, existem arquivos com registros de diferentes tipos;
- ▶ Os relacionamentos entre registros em vários arquivos podem ser representados por **campos de conexão**;
- ▶ Exemplo:





ARQUIVOS DE REGISTROS MISTOS

- ▶ Se for necessário recuperar valores de campo de dois registros relacionados:
 1. Recuperar um dos registros primeiro;
 2. Utiliza-se o valor do item 1 para recuperar o registro relacionado no outro arquivo;

Os relacionamentos são implementados por referências de campo lógicas entre os registros de arquivos distintos.



ARQUIVOS DE REGISTROS MISTOS

- ▶ Para os SGBDs **orientado a objeto**, **hierárquicos** e de **rede**:
 - ▶ Implementam relacionamentos entre registros como relacionamentos físicos realizados pela continuidade física de registros relacionados ou por ponteiros físicos;
 - ▶ Registros de diferentes tipos podem ser **fisicamente agrupados** no disco.

A implementação física do relacionamento (utilizado com frequência) pode aumentar a eficiência do sistema na recuperação dos registros.



ARQUIVOS DE REGISTROS MISTOS

- ▶ Presença do campo **tipo de registro**:
 - ▶ Diferencia os registros em um arquivo misto;
 - ▶ Posicionado antes do valor do campo;
 - ▶ Utilizado pelo software para determinar o tipo de registro que está prestes a ser processado.

Usando a informação do catálogo, o SGDB, determina os campos desse tipo de registro e seus tamanhos, a modo de interpretar seus valores.



RAID

- ▶ *Redundant Array of Inexpensive Disks;*
- ▶ Nivelar as diferentes taxas de melhoria de desempenho dos discos com as taxas da memória e dos microprocessadores:
 - ▶ As capacidades RAM têm se quadruplicado a cada dois ou três anos;
 - ▶ Os tempos de acesso do disco estão melhorando 10% ao ano;
 - ▶ As taxas de transferência do disco estão melhorando 20% ao ano.

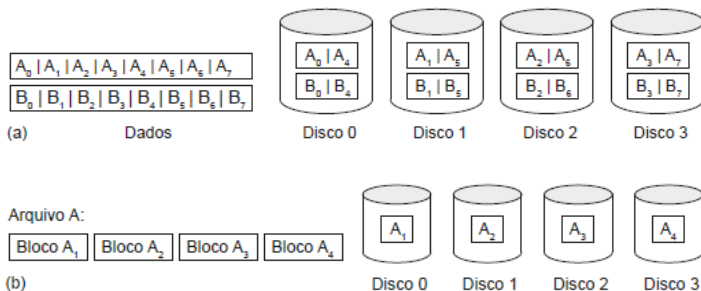


RAID

- ▶ Criação de um grande array de pequenos discos independentes;
- ▶ **Striping de dados**
 - ▶ Emprega o paralelismo (melhora o desempenho do disco);
 - ▶ Distribui os dados transparentemente por vários discos;
 - ▶ Melhora o desempenho de E/S, oferecendo assim altas taxas de transferência gerais;
 - ▶ Balanceia a carga entre os discos.



RAID



Striping de dados em vários discos. (a) Striping em nível de bit em quatro discos. (b) Striping em nível de bloco em quatro discos.



RAID - MELHORANDO A CONFIABILIDADE

- ▶ Para um array de n discos, a probabilidade de falha é de n vezes.

Emprega-se a **redundância de dados**:

- ▶ Desvantagens:
 - ▶ Operações de E/S adicionais para gravação;
 - ▶ Processamento extra para manter a redundância e realizar recuperação de erros.



RAID - MELHORANDO A CONFIABILIDADE

- ▶ Para um array de n discos, a probabilidade de falha é de n vezes.

Emprega-se a **redundância de dados**:

- ▶ **Sombreamento ou Espelhamento**
 - ▶ Técnica para introdução de redundância;
 - ▶ Dobra a taxa em que as solicitações de leitura são tratadas;
 - ▶ A taxa de transferência de cada leitura permanece igual à taxa para um único disco.



RAID - MELHORANDO A CONFIABILIDADE

- ▶ Para um array de n discos, a probabilidade de falha é de n vezes.
- Emprega-se a **redundância de dados**:
 - ▶ **Sombreamento ou Espelhamento**
 1. Os dados são gravados em dois discos físicos idênticos;
 2. Na leitura, buscar a partir de um disco com menor atraso de fila, busca e rotacionais;
 3. Caso algum disco falhe, utilizar outro disco, até que o primeiro seja reparado.



RAID - MELHORANDO A CONFIABILIDADE

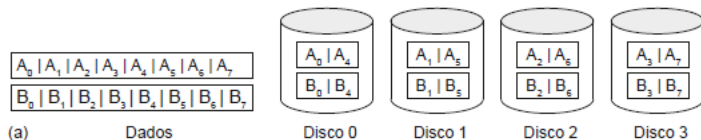
- ▶ Para um array de n discos, a probabilidade de falha é de n vezes.
- Emprega-se a **redundância de dados**:
 - ▶ Necessário selecionar uma técnica para calcular a informação redundante.
 - ▶ Utilizar códigos de correção de erros (bits de paridade).
 - ▶ Necessário selecionar um método de distribuição de informação redundante pelo array de disco.
 - ▶ Armazenar a informação redundante em um pequeno número de discos.



RAID - MELHORANDO O DESEMPENHO

1. Striping de dados em nível de bit

- ▶ Dividir um byte de dados e gravar o bit j no j -ésimo disco;
- ▶ Com bytes de 8 bits, oito discos físicos podem ser considerados um disco lógico, com um aumento de oito vezes na taxa de transferência de dados.
- ▶ Exemplo:

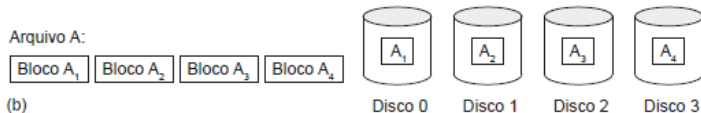




RAID - MELHORANDO O DESEMPENHO

2 Striping de dados em nível de bloco

- ▶ Blocos de um arquivo são espalhados pelos discos;
- ▶ Exemplo:



- ▶ Solicitações independentes que acessam blocos isolados podem ser atendidas em paralelo por discos separados, diminuindo o tempo de enfileiramento das solicitações de E/S.



REDES DE ÁREA DE ARMAZENAMENTO

- ▶ SANs - *Storage Area Networks*;
- ▶ Periféricos de armazenamento online configurados como nós em uma rede de alta velocidade;
- ▶ Alternativas de arquitetura:
 - ▶ Conexões ponto a ponto;
 - ▶ Uso de canal de fibra;
 - ▶ Uso de hubs e switches de canal de fibra.
- ▶ Vantagens:
 - ▶ Conectividade flexível;
 - ▶ Melhores capacidades de isolamento.



ARMAZENAMENTO CONECTADO À REDE

- ▶ NAS - *Network-Attached Storage*;
- ▶ Somente permitem o acréscimo de armazenamento para compartilhamento de arquivos;
- ▶ Substituição dos servidores de arquivos tradicionais;
- ▶ Alto grau de escalabilidade, confiabilidade, flexibilidade e desempenho;
- ▶ Operação confiável e administração fácil;
- ▶ Residem em qualquer local de uma rede local (LAN).



SISTEMAS DE ARMAZENAMENTO iSCSI

- ▶ Os clientes enviam comandos SCSI para dispositivos de armazenamento SCSI em canais remotos;
- ▶ Não exige cabeamento especial;
- ▶ Simplicidade e baixo custo;
- ▶ Funcionamento para o SGBD acessar os dados:
 1. O SO gera comandos SCSI apropriados e, dependendo do caso, encapsula e criptografa;
 2. Um cabeçalho do pacote é acrescentado antes que os pacotes IP resultantes sejam transmitidos por uma conexão Ethernet;



SISTEMAS DE ARMAZENAMENTO iSCSI

- ▶ Os clientes enviam comandos SCSI para dispositivos de armazenamento SCSI em canais remotos;
- ▶ Não exige cabeamento especial;
- ▶ Simplicidade e baixo custo;
- ▶ Funcionamento para o SGBD acessar os dados:
 - 3 Quando um pacote é recebido, ele é descritografado;
 - 4 Os comandos SCSI seguem por meio do controlador SCSI para o dispositivo de armazenamento SCSI.

OBRIGADO

Wladimir Cardoso Brandão

www.wladimirbrandao.com



“Science is more than a body of knowledge. It is a way of thinking.”

Carl Sagan