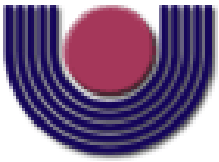

UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Informática

Curso de Bacharelado em Informática

Armazenamento em Disco, Estruturas Básicas de Arquivos e Hashing

Fernando Nunes Bonifácio

Evaristo Wychoski Benfatti

CASCAVEL

2009

Sumário

Capítulo 1

Introdução

As coleções de dados que compõe um banco de dados computadorizado precisa ser de alguma forma armazenada fisicamente em alguma mídia de armazenamento, para que assim os softwares SGBDs possam recuperar, atualizar e processar esses dados conforme necessário.

Este trabalho esta organizado da seguinte forma: no Capítulo 2 falamos sobre os dispositivos de armazenamento, sua hierarquia e como os bancos de dados são tratados neles; no Capítulo 3 descrevemos com mais detalhes alguns dispositivos de armazenamento; no Capítulo 4 falamos sobre a técnica de buffering de dados; no Capítulo 5 abordamos de que forma os registros formam um arquivo e como eles são armazenados no disco; no Capítulo 6 apresentamos as operações básicas que um disco faz nos arquivos; no Capítulo 7 explicamos como funciona o armazenamento em disco de arquivos de registros desordenados; no Capítulo 8 explicamos como funciona o armazenamento de arquivos de registros ordenados; no Capítulo 9 explicamos as técnicas para armazenamento de arquivos hash; no Capítulo 10 apresentamos outras formas de organização primária de arquivos; no Capítulo 11 explicamos uma alternativa de acesso paralelo ao dados conhecida como RAID; no Capítulo 12 discutimos um pouco sobre a crescente utilização das SANs; por fim, no Capítulo 13 apresentamos uma conclusão do presente trabalho.

Capítulo 2

Armazenamento de dados

As mídias de armazenamento em computador formam uma hierarquia de armazenamento que podem ser categorizadas de duas maneiras principais:

- Armazenamento primário: Mídias de armazenamento nesta categoria que podem ser processadas diretamente pela CPU, tais como a memória principal do computador e memória cache. Geralmente são voláteis, fornecem acesso rápido, mas sua capacidade é limitada e o custo por bit é relativamente alto.
- Armazenamento secundário: Dispositivos que geralmente possuem maior capacidade, menor custo, porém proporcionam acesso mais lento aos dados. Os dados contidos nesses dispositivos de armazenamento não podem ser processados diretamente pela CPU, são mídias de armazenamento não voláteis.

Na seção 2.1 abordaremos como os bancos de dados são tratados nesta hierarquia de armazenamento e na seção 2.2 apresentaremos uma visão geral de alguns dispositivos de armazenamento.

2.1 Hierarquia de Memórias e dispositivos de Armazenamento

Em um sistema de computador moderno, os dados residem e são transportados por uma hierarquia de mídias até poderem ser processados pela CPU. As memórias de acesso mais rápido são mais caras e por isso tem menor capacidade, em contrapartida temos as memórias de menor velocidade, que são mais baratas e tem capacidade de armazenamento potencialmente infinita.

No nível de armazenamento primário estão presentes as memórias cache que são memórias RAM (Random Access Memory - Memória de acesso aleatório) estáticas, em seguida temos

as RAMs dinâmicas (DRAM) que popularmente são chamadas de memória principal, essas memórias fornecem a principal área de trabalho para a CPU manter programas e dados. No nível de armazenamento secundário a hierarquia de mídias de armazenamento compreende discos magnéticos, bem como as chamadas mídias de armazenamento em massa, tais como CD-ROM e fitas.

Os programas residem e são executados na RAM. Geralmente os bancos de dados de grande porte usam a memória secundária, parcelas destes quando solicitado são trazidos da memória secundaria para a memória primária em forma de buffers para que possam ser processados. Atualmente como as estações de trabalho estão se tornando cada vez mais rápidas e com maior quantidade de memória principal (da ordem de gigabytes), grande fração desses bancos podem ser trazidos para a memória para que possam ser manipulados, porém após qualquer alteração os mesmos precisam ser devolvidos para a memória secundária para que possam ser armazenados com maior segurança. Em alguns casos bancos de dados inteiros podem ser armazenados em memória principal, geralmente para aplicações de tempo real que exigem tempos de respostas extremamente pequenos, nestes casos o armazenamento secundário funciona como um backup das informações.

Entre o armazenamento na memória principal e discos magnéticos, outra memória que está se tornando comum é a memória flash. Estas memórias são de alta densidade (grande capacidade de armazenamento) e de alto desempenho (acesso rápido). Uma vantagem de seu uso é a velocidade do acesso; sua desvantagem é que um bloco inteiro precisa ser apagado e escrito de cada vez. Este tipo de memória aparece como mídia de armazenamento de dados principalmente em câmeras fotográficas, aparelhos de MP3, acessórios de armazenamento USB, entre outros.

Outros dispositivos de armazenamento secundários são os dispositivos que armazenam os dados de forma óptica e que podem ser lidos com a utilização de laser. Neste tipo de armazenamento entram os CD-ROMs, WORMs e DVDs. Os CD-ROMs mantém os dados pré-gravados e que não podem ser sobrescritos. Os discos WORM (Write-Once-Read-Memory) permitem que os dados sejam escritos uma vez e lidos infinitas vezes, são utilizados principalmente para backup. Os DVDs são considerados um padrão relativamente recente para discos ópticos permitindo armazenamento de grande quantidade de dados, entre 4,5 a 15 gigabytes.

Finalmente temos as fitas magnéticas que são usadas para arquivo e backup de dados. Um conjunto de fitas magnéticas, também chamadas de fitas *jukebox* é chamado de armazenamento terciário e tem capacidade para armazenamento de terabytes de dados.

Para se ter uma idéia geral de como são tratados os bancos de dados na hierarquia de armazenamento basta visualizar a Figura 1. Nela podemos verificar que onde temos dados que precisem ser acessado com maior freqüência, eles estarão armazenados em dispositivos mais da ponta da pirâmide, onde a velocidade de acesso é maior porem a capacidade de armazenamento é pequena, caso os dados não tenham tantos acessos eles estarão armazenados em dispositivos mais da base da pirâmide, onde o acesso aos dados é mais lento, porém tem uma capacidade de armazenamento maior.

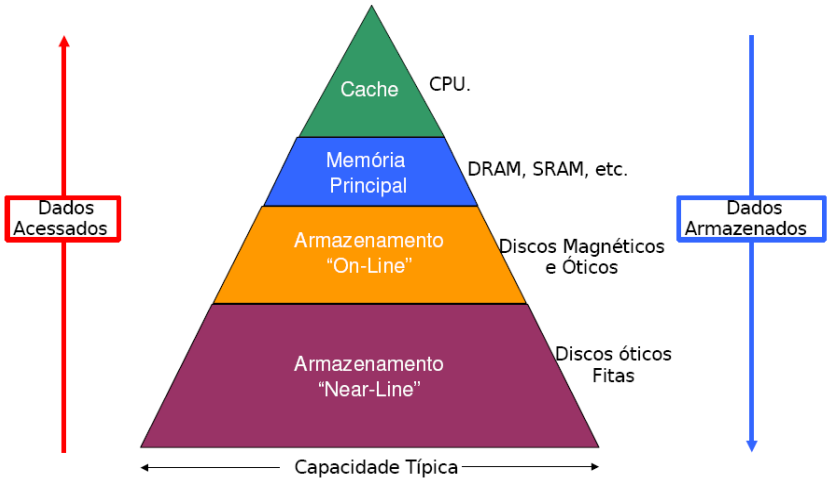

Figura 1. Pirâmide de armazenamento [1].

2.2 Armazenamento de Bando de Dados

Os bancos de dados precisam armazenar grande quantidade de dados que geralmente devem ser mantidos armazenados por longos períodos de tempo e precisam ser processados repetidas vezes durante este período. Como as circunstâncias que causam perdas permanentes aparecem menos freqüentemente em armazenamentos secundários e bancos geralmente são muito grandes para serem armazenados por completo em memórias primária, a maioria deles são armazenados permanentemente em dispositivos de armazenamento secundário.

Algumas das mais novas tecnologias - DVDs e fitas *jukebox* - provavelmente serão opções viáveis para o uso de discos magnéticos, porém, segundo Elmasri [2], pode-se antecipar que os discos magnéticos continuarão a ser a mídia de escolha primária para grandes bancos de

dados por vários anos, por isso daremos atenção especial a este tipo de armazenamento no capítulo seguinte.

As fitas por outro lado são freqüentemente usadas como uma mídia de armazenamento para backup de bancos por causa do seu baixo custo. Pelo fato de serem ainda bastante utilizadas, abordaremos ela de forma mais aprofundada na seção 3.4 do capitulo seguinte. As fitas magnéticas armazenam os dados de maneira off-line, ou seja, para acessar os dados é necessário a intervenção de um operador para carregá-los antes que eles se tornem disponíveis, é uma abordagem em contraste com os discos magnéticos, onde os dados estão disponíveis de maneira on-line, podendo ser acessados diretamente a qualquer momento.

As técnicas utilizadas para armazenar grandes quantidades de dados estruturados em um disco são importantes para os projetistas de bancos de dados, bem como para o DBA e os responsáveis pela implementação de SGBDs. Eles devem conhecer as vantagens e as desvantagens do uso ou do não uso de uma técnica de armazenamento quando projetam, utilizam e operam um banco sobre um SGBD específico. Geralmente os SGBDs possuem diversas configurações quanto à organização dos dados, o processo de projeto físico envolve escolher, entre as opções, as técnicas que melhores se adaptem aos requisitos de uma dada aplicação. Eles devem estudar essas técnicas de forma que possam implementá-las de forma eficiente para proporcionar a maior numero de opções aos DBAs e usuários do SGBD.

Normalmente as aplicações comuns de bancos necessitam de apenas de uma pequena quantidade de dados do banco de dados para o processamento em determinado momento. Quando certa porção dos dados for necessária, ela precisará inicialmente ser localizada no disco e então copiada na memória principal. Esses dados são organizados como arquivos e registros. Cada registro é uma coleção de valores de dados que podem ser interpretados como fatos a respeito das entidades. Esses registros precisam ser armazenados no disco de forma que seja possível localizá-los e alterá-los de forma eficiente quando necessário.

A forma como os registros são armazenados é chamada de organização primária de arquivo e determina como os registros de um arquivo estão dispostos fisicamente no disco. Existem basicamente três tipos de armazenagem: arquivo heap (ou arquivo desordenado), arquivo sorted (ou arquivo ordenado) e arquivo hash. Estes tipos de arquivo serão descritos no Capítulo 9.

Capítulo 3

Dispositivo de armazenamento

Nesta seção descreveremos algumas das características dos dispositivos de armazenamento primário e secundário.

3.1 Memória RAM [3]

A sigla “RAM” significa “Ramdom Access Memory” ou “memória de acesso aleatório”. A principal característica da memória RAM é a capacidade de fornecer dados anteriormente gravados, com um tempo de resposta e uma velocidade de transferência centenas de vezes superior à dos dispositivos de memória de massa, como o disco rígido.

Os chips de memória RAM possuem uma estrutura extremamente simples. Para cada bit 1 ou 0 a ser armazenado, temos um minúsculo capacitor, que quando está carregado eletricamente temos um bit 1 e quando está descarregado temos um bit 0. Para cada capacitor temos um transistor, encarregado de ler o bit armazenado em seu interior e transmiti-lo ao controlador de memória. A memória RAM é volátil justamente devido ao capacitor perder sua carga muito rapidamente. Para ler e gravar dados na memória, assim como controlar todo o trânsito de dados entre a memória e os demais componentes do micro, é usado mais um circuito, chamado controlador de memória, que faz parte do chipset localizado na placa mãe.

Para facilitar o acesso a dados os módulos de memória são dividido em linhas e colunas. Para acessar um determinado transistor (seja para gravar ou ler dados), o controlador de memória primeiro gera o valor RAS (Row Address Strobe), ou o número da linha da qual o transistor faz parte, sendo gerado em seguida o valor CAS (Collum Address Strobe), que corresponde à coluna. Esta abordagem pode ser melhor visualizada na Figura 2.

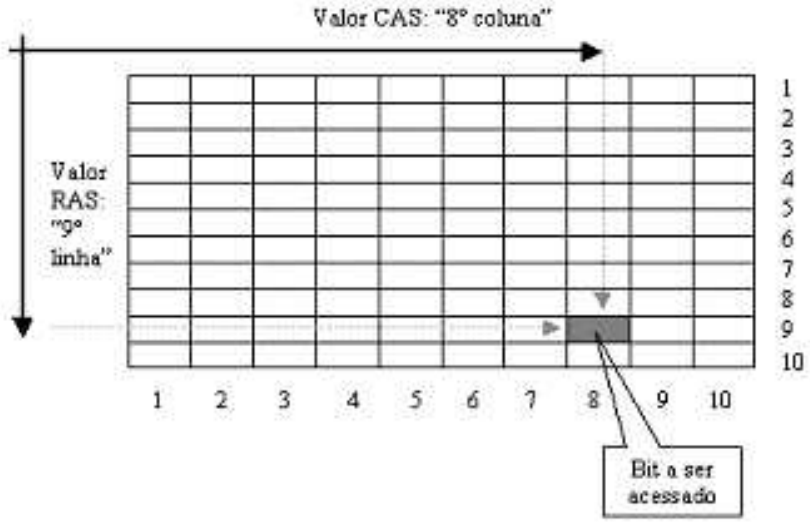

Figura 2. Acesso à memória RAM.

3.2 Memória Cache [4]

A memória cache surgiu quando se percebeu que as memórias não eram mais capazes de acompanhar o processador em velocidade, fazendo com que muitas vezes ele tivesse que ficar "esperando" os dados serem liberados pela memória RAM para poder concluir suas tarefas, perdendo muito em desempenho. Para solucionar este problema, começou a ser usada a memória cache, um tipo ultra-rápido de memória que serve para armazenar os dados mais freqüentemente usados pelo processador, evitando na maioria das vezes que ele tenha que recorrer à comparativamente lenta memória RAM.

São usados dois tipos de cache, chamados de cache primário, ou cache L1 (level 1), e cache secundário, ou cache L2 (level 2). O cache primário é embutido no próprio processador e é rápido o bastante para acompanhá-lo em velocidade. Sempre que um novo processador é desenvolvido, é preciso desenvolver também um tipo mais rápido de memória cache para acompanhá-lo. Como este tipo de memória é extremamente caro (chega a ser algumas centenas de vezes mais cara que a memória RAM convencional) usamos apenas uma pequena quantidade dela. Para complementar, usa-se também.

3.3 Armazenamento em disco

Discos magnéticos são utilizados para armazenamento de grande quantidade de dados. A unidade mais básica de um disco é o bit que representa o valor 0 ou 1, esta representação se

faz através da magnetização de uma área do disco. O conjunto de 8 bits forma o que chamamos de byte, que utilizamos para medir a capacidade de um disco. Atualmente a capacidade dos disco esta na ordem dos terabytes [5].

Independente da capacidade todos os discos rígidos são feitos de uma material magnético moldado como um fino disco circular (Figura 3. a) e protegida por uma cobertura plástica ou acrílica. Têm-se discos de face única e de face dupla que dizem respeito às superfícies do disco que são usadas para armazenamento. Para aumentar a capacidade da unidade de armazenamento, os discos são agrupados em conjuntos de discos (Figura 3. b), que compreendem desta forma várias superfícies.

Para a "ordenação" dos dados no disco rígido faz-se a divisão dele em conceito de cilindros, trilhas e setores. As trilhas são círculos que começam no centro do disco e vão até a sua borda, como se estivesse um dentro do outro. Essas trilhas são numeradas da borda para o centro, isto é, a trilha que fica mais próxima da extremidade do disco é denominada trilha 0, a trilha que vem em seguida é chamada trilha 1, e assim por diante, até chegar à trilha mais próxima do centro. Damos o nome de cilindro ao conjunto das trilhas de mesmo número presentes em todos os discos da unidade. Como uma trilha ainda tem uma quantidade grande de informações ela é dividida em pedaços menores chamados de setores. Existem dois tipos principais da divisão da trilha em setores, na primeira divisão dividem-se as trilhas em partes determinadas por um ângulo fixo a partir do centro (Figura 4. a), na outra divisão o tamanho das partes que a trilha é dividida é determinada por ângulos menores a partir do centro conforme se move para o exterior do disco, desta forma a densidade de gravação do disco é mantida uniforme.

Para que um disco possa realmente receber dados ele precisa passar por um processo chamado formatação. Há dois tipos de formatação: formatação física e formatação lógica. O primeiro tipo é justamente a "divisão" dos discos em trilhas e setores. Esse procedimento é feito na fábrica. A formatação lógica, por sua vez, consiste na aplicação de um sistema de arquivos apropriado a cada sistema operacional. Por exemplo, o Windows é capaz de trabalhar com sistemas de arquivos FAT e NTFS. O Linux pode trabalhar com vários sistemas de arquivos, entre eles, ext3 e ReiserFS. Nesta formatação uma trilha é dividida em blocos de discos, também conhecidas como páginas. O tamanho de cada bloco e disco varia de 512 a 4096 bytes.

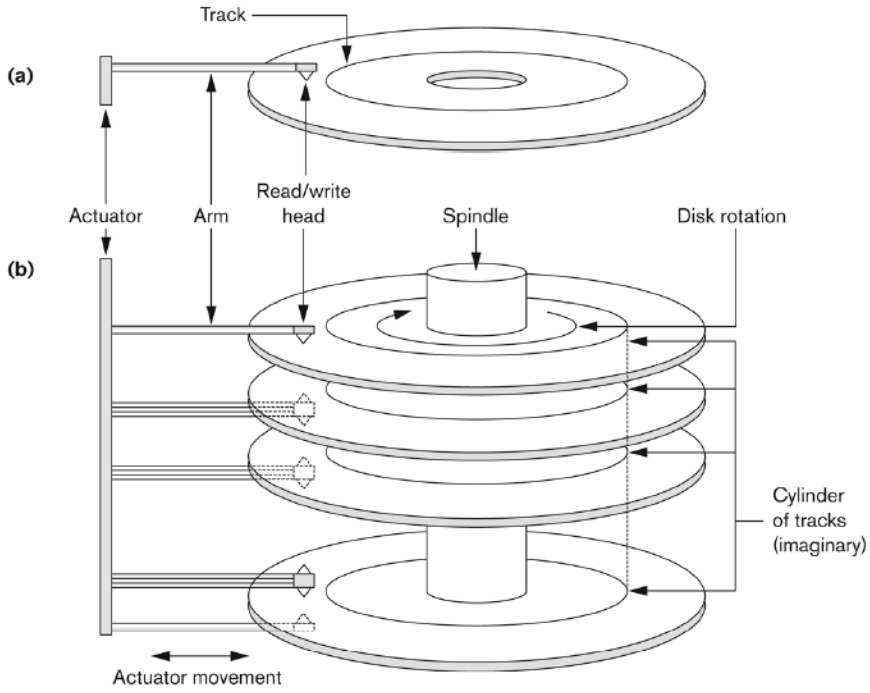

Figura 3. Unidade de disco.

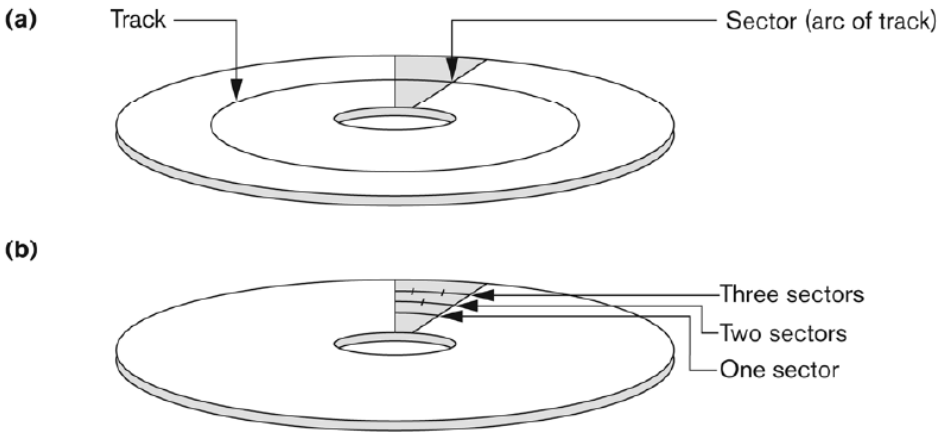

Figura 4. Divisão das trilhas de um disco.

Um disco é um dispositivo de acesso aleatório, isto quer dizer que para acessar certo endereço precisamos apenas informar o endereço desejado. O endereço físico de um disco é determinado pela combinação do número do cilindro, do número da trilha e do número do bloco. Alguns discos mais atuais fazem o acesso baseado em um número único chamado LBA (Logical Block Address - Endereço lógico de bloco), onde se passa para a controladora da unidade do disco o bloco desejado e ela se encarrega de mapear para o endereço físico. A transferência dos dados entre a memória principal e o disco é feito por unidades de blocos. Ao

se fazer um pedido de leitura ou gravação em um disco é reservado na memória principal uma área contígua chamada de buffer, este buffer é utilizado da seguinte forma. Ao se fazer um pedido de leitura ao disco o bloco é copiado do disco para este buffer, já no comando de escrita o bloco é copiado do buffer para o disco. Algumas vezes é possível transferir vários blocos contíguos ao mesmo tempo e neste caso devemos ajustar o tamanho do buffer ao número de bytes deste conjunto, o qual chamamos de cluster.

Uma unidade de disco pode ser determinada como o conjunto dos dispositivos de hardware que permitem que um conjunto de discos possa ser utilizado. Nesta unidade de disco estão presentes: cabeçote de leitura/escrita, braço mecânico, atuador e controladora de disco. O cabeçote é o equipamento de hardware que realmente faz a leitura ou escrita de um bloco no disco, ele inclui um dispositivo eletrônico chamado de braço mecânico. O conjunto de discos com duas superfícies são controlados por diversos cabeçotes de leitura e escrita, um para cada superfície como pode ser verificado na Figura 3.b. Os braços movem os cabeçotes simultaneamente e os posiciona com precisão sobre o endereço solicitado. O atuador esta vinculado a um motor elétrico e todos os braços mecânicos estão ligados a ele.

O cabeçote descrito anteriormente é do tipo que se move para a posição solicitada, porém existe outro tipo de unidade de disco que trabalha com cabeçote fixo. Neste tipo de disco uma trilha ou um cilindro é selecionado por meio por meio de um dispositivo da seleção eletrônica do cabeçote de leitura/escrita apropriado em vez dos movimentos mecânicos reais, conseqüentemente isto é muito mais rápido, porém o custo de cabeçotes de leitura adicionais tem um custo relativamente alto tornando o uso de unidade de disco com cabeçote fixo não tão freqüente.

Uma controladora de disco controla a unidade de disco e realiza a interface entre ela e o sistema do computador, geralmente esta embutida na própria unidade. Ela aceita comando de entrada e saída de alto nível e executa ação apropriada para posicionar o cabeçote no local adequado para a solicitação.

O desempenho de um disco é baseado em três tempos:

- Tempo de pesquisa: tempo que a controladora de disco leva para posicionar mecanicamente o cabeçote de leitura/escrita sobre a trilha correta.
- Tempo de atraso rotacional (latência): tempo decorrido entre a localização da trilha até o cabeçote atingir o inicio do bloco solicitado durante a rotação do disco.

- Tempo de transferência de bloco: transferência propriamente dita dos dados para ou do cabeçote de leitura/escrita.

O tempo de pesquisa e atraso rotacional são geralmente muitos maiores que o tempo de transferência do bloco, por isto para realizar a transferência de vários blocos com uma maior eficiência é comum a transferência de vários blocos consecutivos da mesma trilha ou do mesmo cilindro eliminando desta forma os dois atrasos iniciais (pesquisa e latência) e economizando um tempo substancial de transferência.

Localizar dados no disco é considerado o maior gargalo em aplicações de banco de dados, pois o tempo de transferência dos dados do disco, que é da ordem de milissegundos, é considerado bastante alto se comparado ao tempo necessário para processar dados da memória principal por meio das CPUs atuais. Devido a isto diversas estruturas de arquivo têm sido desenvolvidas tentando minimizar o número de transferência de blocos necessário para a leitura do arquivo.

3.4 Armazenamento em fitas magnéticas

Enquanto os discos são conhecidos como dispositivos de acesso aleatório as fitas magnéticas são conhecidas como dispositivos de armazenamento de acesso seqüencial, porque se quisermos acessar o n-ésimo bloco da fita obrigatoriamente tem-se que percorrer os n-1 blocos anteriores.

Neste tipo de dispositivo os dados são armazenados em cartuchos de fitas magnéticas parecida com as fitas de áudio e vídeo comuns, sendo necessário uma unidade de fita para ler e gravar os dados no cartucho da fita. De forma geral os bits também são armazenado em bytes, com cada byte disposto de forma transversal na fita, conseqüentemente os bytes são armazenados de forma consecutiva no carretel.

Para acessar um bloco no meio de um cartucho, a fita é montada e então percorrida até o que o bloco solicitado esteja sob o cabeçote de leitura/ escrita. Por esta razão o acesso a fita pode ser lento e não poderem ser utilizadas para armazenar dados on-line na maioria das aplicações. Entretanto elas atendem a uma função muito importante: o backup do banco de dados, onde periodicamente copiam-se os dados do disco para fitas magnéticas, evitando-se assim que os dados sejam perdidos devido a um desastre de disco (disco crash).

Outros usos para fitas magnéticas podem ser:

- Armazenar arquivos de bancos de dados demasiadamente grandes;

- Armazenar arquivos de banco de dados que raramente são utilizados ou que estão desatualizados, ou seja para manter registro históricos;
- Utilização em conjunto, nas chamada bibliotecas de fitas, que tem grandes capacidades de armazenamento e transferência. Por exemplo, a biblioteca gigante L550 da Storage Technology, que pode armazenar até 13,2 petabytes e ter uma vazão de 55TB/hora.

Capítulo 4

Buffering de blocos

Quando é necessário transferir um bloco do disco para a memória e todos os endereços são conhecidos, diversos buffers podem ser reservados na memória principal para acelerar a transferência desses dados. Enquanto um buffer estiver sendo lido ou escrito a CPU pode processar os dados em outro. Isso só é possível quando existe uma controladora de entrada e saída independente, que uma vez inicializado pode realizar transferência de um bloco de dados entre a memória e o disco de maneira independente e simultaneamente ao processamento da CPU. O buffering é muito útil para processos executados concorrentemente em paralelo, porque eles podem se dedicar exclusivamente a um processo, enquanto outro resolve os problemas de entrada e saída.

Quando o tempo necessário para processar um bloco do disco na memória for menor do que para ler o próximo bloco e completar o buffer, a CPU pode iniciar o processamento do bloco após sua transferência para a memória principal e, ao mesmo tempo, o processador de entrada e saída do disco pode ler e transferir os próximo dados para um buffer diferente. Essa técnica é chamada de buffering duplo, ela permite que leituras e escritas sejam realizadas de forma continua em blocos consecutivos no disco, eliminando assim o tempo de pesquisa e atraso rotacional para transferência de todos os blocos, exceto o primeiro. Além disso, os dados estão sempre prontos para serem processados, reduzindo assim o tempo de espera dos programas.

Capítulo 5

Disposição de registros de arquivos em disco

Os dados geralmente são armazenados no disco em forma de registros. Cada registro consiste de uma coleção de valores ou itens relacionados na qual cada valor é formando por um ou mais bytes que correspondem a um campo de um registro. Para o caso de banco de dados em particular, registros descrevem entidades e atributos. Um tipo de dado associado a cada campo especifica que tipo de dados o campo pode receber. Coleções de nomes de campos com seus respectivos tipos constituem uma definição de tipo de registro ou formato de registros.

O Tipo de dado de um campo geralmente esta relacionado a tipos de dados padrões de programação, tais como inteiros, reais, lógicos, strings e algumas vezes dados especialmente codificados para representar datas e hora. O numero de bytes necessários para representar esses dados é determinado pelo SGBD.

Atualmente os usuários de bancos de dados estão encontrando necessidade em armazenar itens que consistem grandes objetos desestruturados, tais como imagens, vídeos ou áudio digitalizados. Esses objetos são referenciados como BLOBs (Binary large objects - Grandes objetos binários). Porém como esses dados são geralmente muito grandes, eles são armazenados separadamente de seu registro e ponteiros para essas posições são incluída nos registros.

5.1 Arquivos, Registros de Tamanho Fixo e Registros de Tamanho Variável

Um arquivo é uma seqüência de registros. Quando todos os registros de um arquivo possuem o mesmo tamanho (em bytes) dizemos que o arquivo é formado por registros de tamanho fixo, em contrapartida se os registros em um arquivo possuem tamanhos diferentes

dizemos que o arquivo é composto de registros de tamanho variável. Um arquivo pode ser de tamanho variável quando:

- Os registros contidos em um mesmo arquivo são de mesmo tipo, mas os campos desses registros são de tamanho variável, por exemplo, um vetor;
- Os registros contidos em um mesmo arquivo são de mesmo tipo, porém um ou mais campos podem ter diversos valores para registros individuais;
- Os registros são de mesmo tipo, porém um ou mais campos são opcionais, ou seja, eles podem ter valor ou não;
- O arquivo contém registros de diferentes tipos, este arquivo pode ser definido como arquivo misto. Este caso pode ocorrer quando registros de diferentes tipos relacionados são reunidos em cluster para reduzir o tempo de acesso.

Para um SGBD quando cada registros de um mesmo arquivo possuem os mesmos campos e esses campos são de fixos, o sistema pode identificar a posição inicial do byte inicial de cada campo a partir da posição inicial do primeiro registro. Isso facilita muito a localização dos valores dos campos, pois apenas deslocamentos a partir do registro inicial são necessários para acessar todos os registros. Mas quando registros são de tamanho variável também é possível representar um arquivo que logicamente conteria registros de tamanho variável a partir de um arquivo de registros de tamanho fixo. Por exemplo, para casos de campos de tamanhos variáveis poderíamos armazenar nesses campos o maior valor possível mesmo que ele não fosse totalmente utilizado. Porém em qualquer caso haverá desperdício de espaço se registros não ocuparem esses campos na integralidade.

Algumas técnicas são utilizadas para representar os dados sem desperdiçar espaço em troca de processamento. Para casos de campos de tamanho variável, para determinar os bytes de um registro em particular, podemos usar caracteres separadores especiais para encerrar a entrada desses campos, ou ainda podemos armazenar o tamanho em bytes do campo no registro antes do valor do campo. Casos em que muitos campos são opcionais podemos armazenar tuplas <nome do campo, valor do campo> indicando qual valor recebe que dado, assim campos realmente preenchidos são armazenados na memória. E por último, em campos que podem ser multivalorados, poderíamos utilizar caracteres especiais para representar a separação entre uma entrada e outra e um outro caractere para representar o termino das entradas.

Programas que processam arquivos de tamanho variável são consideravelmente mais complexos que aqueles de tamanho fixo, pois eles precisam tratar todo este indeterminismo descrito anteriormente.

5.2 Divisão de Registros em Blocos e registros Spanned Versus registros Não-Spanned (Unspanned)

Os registros de um arquivo precisam ser alocados em blocos de disco porque um bloco é a unidade de transferência de dados entre o disco e a memória. Raramente um registro é maior que um bloco de disco, sendo assim comumente temos vários registros armazenados em um bloco de disco. Isto nos fornece uma grandeza chamada de fator de blocagem (fator de divisão em blocos) para um arquivo, que é determinada pela fórmula bfr = [B/R] onde B é o número de bytes de um bloco e R é o número de bytes de um registro de um arquivo de registros de tamanho fixo. Fazendo um arredondamento para baixo no calculo de bfr ele nos indica quantos registros daquele arquivo é possível armazenar em um bloco, geralmente esta divisão não é exata e assim devemos ter um espaço de B - (bfr *R) bytes inutilizados.

Para resolver este problema podemos armazenar parte de um registro em um bloco e o restante em outro. Ao final do primeiro bloco teríamos um ponteiro apontando para o bloco que contem o restante do registro caso ele não seja o bloco consecutivo no disco. A esta organização damos o nome de spanned porque os registros podem estar fragmentados por mais de um bloco. A organização descrita anteriormente onde um registro não pode atravessar as fronteiras do bloco, damos o nome de não-spanned.

A organização spanned é utilizada principalmente para arquivos de registros de tamanho fixo, tendo B > R, porque isto faz com que cada registro comece em um localização conhecida do bloco, simplificando o processamento de registros. Para arquivos de registros de tamanho variável podem ser usados tanto a organização spanned quanto a organização não-spanned, porém para casos onde o tamanho dos registros é grande é vantajoso usar spanned para reduzir a perda de espaço em cada bloco.

5.3 Alocação de blocos de arquivos em disco

Existem diversas técnicas para a alocação de blocos de um arquivo em disco. Na alocação consecutiva os blocos de arquivo são alocados consecutivamente no disco. Na alocação encadeada cada bloco de arquivo contém um ponteiro para o próximo bloco de arquivo. Na alocação consecutiva a leitura inteira de um arquivo é muito rápida, pois se pode utilizar o buffering duplo, porém a expansão de um arquivo é feito de forma mais difícil. Na alocação encadeada, a expansão do arquivo é facilitada, porém torna a leitura mais lenta. Uma combinação destas duas técnicas consiste em alocar blocos consecutivos em clusters de disco com o encadeamento deles. Uma outra possibilidade é utilizar a alocação indexada, na qual um ou mais blocos de índice contém ponteiros para os blocos de arquivo de fato.

5.4 Cabeçalhos de arquivo

Um cabeçalho de arquivo, também chamado de descritor de arquivo, contém informações sobre o arquivo que serão necessários ao programas de sistema que irão acessar os registros deste arquivo. Entre as informações disponíveis estão: informações para determinar os endereços de disco dos blocos dos arquivos; descrição do formato, que incluir tamanho e ordem dos campos para um arquivo de tamanho fixo não-spanned; códigos de tipos de campos; caracteres separadores; códigos de tipos de registro, para registros de tamanho variável.

Capítulo 6

Operações em arquivos

As operações em um arquivo se dividem em operações de recuperação e operações de atualização.

As operações de recuperação dizem respeito àquelas operações que não alteram nenhum valor no arquivo, apenas localizam os registros e obtém os seus valores para processamento. As operações de atualização dizem respeito àquelas alterações que fazem alguma alteração no arquivo, através da inclusão, exclusão ou modificação de registros.

Em ambas as operações descritas anteriormente algumas vezes é necessário selecionar mais de um registro no disco tendo como base algum critério de seleção (condição para filtragem) que especificam os critérios que os registros devem satisfazer. A maioria das operações de seleção é do tipo simples do tipo nome = “joao". Quando mais de um registro satisfaz a condição de seleção, o primeiro registro localizado na seqüência física dos registros do arquivo é designado registro atual e as operações de pesquisa seguintes começam a partir desse registro.

Abaixo se apresenta um conjunto representativos de operações que podem ser executados em um arquivo. Estas operações são geralmente de alto nível e são utilizadas principalmente por softwares SGBD:

- Open (Abrir): Prepara o arquivo para leitura ou escrita. Para isto aloca buffers adequados (pelo menos dois), busca o cabeçalho do arquivo e posiciona o ponteiro no inicio do arquivo.
- Reset (Reinicializar): Reposiciona o ponteiro de arquivo, de um arquivo aberto, no seu inicio.
- Find ou Locate (Encontrar ou Localizar): Busca o primeiro registro que satisfaça a condição de pesquisa, transfere aquele bloco que tem a condição de pesquisa para

um buffer de memória principal e faz o ponteiro de arquivo apontar para o registro no buffer, tornando-o o registro atual.

- Read ou Get (Ler ou Obter): Copia o registro atual do buffer para uma variável do programa de usuário.
- FindNext (Encontrar o próximo): procura o próximo registro no arquivo que satisfaça a condição de pesquisa, transferindo o bloco que contém aquele registro para o buffer da memória principal.
- Delete (Excluir): Exclui o registro atual e no final atualiza o arquivo de disco para refletir a exclusão.
- Modify (Modificar): Modifica alguns valores de campos do registro atual e no final atualiza o arquivo no disco para refletir a modificação.
- Insert (Incluir): Acrescente um novo registro no arquivo por meio da localização do bloco no qual o registro deve ser incluído, transferindo aquele bloco para o buffer da memória principal, escrevendo o registro no buffer e no final escrevendo o buffer no disco para refletir a modificação.
- Close (Fechar): Finaliza o acesso ao arquivo por meio da liberação dos buffers e da execução de quaisquer outras operações de limpeza necessárias.

Com exceção da operações Open e Close todas as demais apresentadas acima são chamadas record-at-a-time (um registro por vez), porque cada operação se aplica somente a um único registro. Algumas vezes é possível aplicar um conjunto de operações em uma única operação, por exemplo, as funções Find, FindNext e Read podem ser unificadas em uma operação Scan (Varrer) que funciona da seguinte forma: se o arquivo tiver acabado de ser aberto ou reinicializado, o Scan retorna o primeiro registro, caso contrário retorna o próximo. Em banco de dados, operações podem existir ainda operações record-at-a-time (um conjunto de registro por vez), para serem aplicadas a um arquivo.

Neste ponto é importante diferenciar os termos organização de arquivo e método de acesso. A organização do arquivo se refere à organização dos dados de um arquivo em registros, blocos e estruturas de acesso, ou seja, à forma como os registros e blocos estão posicionados na mídia de armazenamento. O método de acesso se refere ao grupo de operações que podem ser aplicadas a um arquivo, como as operações listadas anteriormente.

Baseado nos tipos de acesso em um banco de dados pode-se ter dois tipos de arquivos: estáticos, significando que operações de atualização são raramente usadas; dinâmicos, onde as

operações de atualização são aplicadas mais freqüentemente. Baseada nestas duas classificações uma organização bem sucedida do arquivo deve realizar da forma mais eficiente possível as operações mais aplicadas a ele. Nos Capítulo 7 e 8 apresentaremos formas de organização de um arquivo em disco e no Capítulo 9 apresentaremos alguma técnicas para criar métodos de acesso tais como classificação e hashing.

Capítulo 7

Arquivos de registros desordenados (Heap Files)

É o tipo de organização de arquivos em disco mais simples e básica, os arquivos são posicionados no arquivo segundo a ordem pela qual foram inseridos, de forma que os novos registros são acrescentados ao final do arquivo.

Para uma operação de inclusão de um novo registro o processo é muito eficiente, basta copiar o ultimo bloco do arquivo no buffer da memória principal, acrescentar o registro desejado e reescrever o bloco no disco.

Uma operação de pesquisa de registro já não é tão eficiente neste tipo de organização, pois independente do critério para seleção do arquivo, deve-se percorrer seqüencialmente bloco a bloco do arquivo. Em média este tipo de operação pesquisa a metade dos blocos do arquivo até encontrar o registro solicitado e na pior das hipóteses, quando o registro não esta no arquivo, pesquisa todos os blocos do arquivo.

A exclusão de um arquivo funciona da seguinte forma: primeiro deve-se encontrar o bloco que contenho o registro solicitado, após isto se deve copiá-lo em um buffer, excluir o registro do buffer e finalmente reescrevê-lo de volta no disco. Outra forma de fazer a exclusão de registro é fazer com que cada registro tenha um campo extra chamado de marcador de exclusão, onde para excluir um arquivo deve-se alterá-lo para certo valor. Em ambas as técnicas ocorrem desperdícios de armazenamento exigindo muitas vezes uma reorganização periódica do arquivo para reaproveitar o espaço sem uso dos registros excluídos.

Neste tipo de organização de arquivo pode-se usar tanto organizações spanned como não spanned e pode ser usando tanto registros de tamanho fixo como registro de tamanhos variáveis. Como localizar um arquivo neste tipo de organização pode ser uma tarefa árdua é comum manter uma cópia ordenada do arquivo e estas organizações geralmente são feitas por técnicas especiais de classificação externa.

Capítulo 8

Arquivos Ordenados (Sorted Files)

Ao armazenar os registros de um arquivo em disco, podemos fazer isto de forma ordenada a partir de algum valor de seus campos. O campo escolhido passara a ser conhecido como campo de classificação. Se o campo de classificação também for um campo chave do arquivo chamaremos o campo de chave de classificação para arquivo.

Arquivos ordenados têm várias vantagens sobre arquivos desordenados. Primeiro, a leitura dos registros na ordem dos valores da chave de classificação se torna muito eficiente, pois nenhuma classificação se faz necessária. Segundo, localizar o próximo registro a partir do atual é muito fácil porque é bem provável que ele esteja no mesmo bloco que o atual. Terceiro, o uso de pesquisa baseado no valor do campo chave de classificação se torna mais rápido quando utilizamos uma técnica de pesquisa binária, a qual constitui uma melhoria em relação à pesquisa linear. Esta ordenação não tem vantagem alguma caso seja feitas consultas com base em valores dos demais campos não ordenados do arquivo. Nesses casos deve-se executar uma pesquisa linear ao arquivo.

Para inserir um registro devemos primeiro encontrar sua posição correta no arquivo e abrir então um espaço nele para inserir o registro naquela posição. Isto leva um tempo consideravelmente grande, principalmente para arquivos grandes, pois, em média, metade dos registros do arquivo deverá ser lida e regravada para abrir espaço ao novo registro. A inclusão pode ser tornar mais eficiente se mantermos alguns espaços não utilizados em cada bloco para novos registros, entretanto ao se adicionar um registro neste espaço o problema ressurge. Outro método freqüentemente utilizado é criar um arquivo desordenado temporário, chamado de overflow ou de transação. Os novos registros a serem adicionados são incluídos ao final do arquivo de overflow em vez de serem incluídos no arquivo principal. Periodicamente o arquivo de overflow deve ser classificado e incorporado ao arquivo mestre durante a reorganização.

Para a exclusão de um registro em arquivos ordenados o problema é menos grave se forem utilizados marcadores de exclusão e se forem feitas reorganizações periódicas.

A modificação de valores de algum campo do registro depende de dois fatores: da condição para localizar o registro e do campo a ser modificado. Se a condição de pesquisa for baseada no campo-chave de classificação poderemos localizar o registro usando a pesquisa binária, caso contrário teremos que utilizar a pesquisa linear. Modificar um campo que não seja o campo chave exige apenas a alteração do registro e de sua regravação na mesma localização no disco, supondo registros de tamanhos fixos, porém se a modificação for para modificar um campo de classificação significará que sua posição poderá ser alterada no arquivo o que exigirá a exclusão do registro antigo seguida de uma inclusão do registro modificado.

Raramente arquivos ordenados são utilizados em aplicações de banco de dados, a menos que um caminho de acesso real, chamado de índice primário seja utilizado, isto resulta em um arquivo seqüencial indexado e melhora ainda mais o tempo de acesso aleatório no campo-chave de classificação.

Capítulo 9

Técnicas de Hashing

É um tipo de organização primária de arquivo baseada em hashing, que fornece um acesso muito rápido aos registros sob certas condições. A condição de pesquisa precisa ser de igualdade em um único campo, neste caso o campo é chamado de campo hash. A idéia do hashing é fornecer uma função h(x), chamada de função hash, que aplicada ao valor do campo de hash de um registro gere o endereço do bloco de disco no qual o retiro está armazenado. A busca pelo registro dentro do bloco pode ser realizada em um buffer da memória principal.

Nas próximas subseções apresentaremos respectivamente o uso de hashing para arquivos internos, o uso de hashing para armazenar arquivos externos e finalmente técnicas para estender o hashing a arquivos com crescimento dinâmico.

9.1 Hashing interno

Para arquivos internos, hashing é normalmente implementada através de uma tabela hash, por meio de um vetor de registros. Por exemplo, suponha que o índice do vetor varie de 0 a M-1, então temos M posições, cujos endereços correspondem aos índices do vetor e escolhemos uma função hash tal que transforme o valor do campo hash em um número inteiro entre 0 e M-1. Uma função típica para isto seria a função h(K) = K mod M, que retorna o valor inteiro do resto após a divisão, este valor será então usado como endereço de registro [2]. Além desta função hash existem outras e também existem outras formas de tratar o campo de hash escolhido, por exemplo, pode-se pegar o terceiro, o quinto e o sétimo dígitos de um campo hash e fazer este número ser o endereço no qual o registro deve ser inserido.

O problema com a maioria das funções hash é que elas não garantem que diferentes valores levarão a diferentes endereços hash porque o número de valores possíveis que um campo hash pode assumir é geralmente muito menor que o número de endereços disponíveis

para os registros. Sendo assim muitas vezes ocorrerão colisões, onde o valor do campo de hash para um registro é a ser incluído é igual ao valor de uma posição onde já tenha algum registro inserido. Desta forma deve-se escolher uma outra posição para inserção do novo registro e esta nova posição pode ser definida de diversas formas, destaca-se abaixo alguns métodos:

- Open Addressing (Endereço aberto): a partir da posição já ocupada pelo endereço hash, o programa prossegue a verificação pela ordem das posições subseqüentes até que seja encontrada uma posição não utilizada.
- Encadeamento (Chaining): são mantidas várias posições de overflow, por meio da extensão do vetor por um número de posições de overflow. Uma colisão é resolvida posicionando se o novo registro em uma localização de overflow não utilizada. Este método utiliza um ponteiro a cada localização de registro, mantendo assim uma lista encadeada de registros de overflow para cada endereço hash.
- Hashing Múltiplo: o programa aplicará uma segunda função hash caso a primeira resulte em colisão. Se ocorrer novamente uma colisão o sistema usará open addressing ou uma terceira função hashing se necessário.

Cada método de resolução de colisão requer seus próprios algoritmos para inclusão, recuperação e exclusão de registros, os mais simples de serem implementados são os de encadeamento.

9.2 Hashing externo para arquivos em disco

Chama-se de hash externo quando se trata de hashing para arquivos em disco. Neste caso considera-se que o espaço de endereçamento alvo é constituído de buckets, que são grupos de blocos de disco consecutivos. A função hash mapeia uma chave a um número de bucket relativo ao invés de um endereço absoluto de bloco para o bucket. Uma tabela, mantida no cabeçalho do arquivo, converte o número do bucket para o endereço de bloco de disco correspondente.

Neste caso o problema de colisão é menos grave, pois várias chaves poderão ter o mesmo número de bucket, bastando ficar atento para a quantidade máxima de registros por buckets. Quando um bucket atingir sua capacidade máxima e a função hash direcionar um novo registro a ele, podemos utilizar uma variação do encadeamento no qual seja mantido, em cada bucket, um ponteiro para uma lista encadeada de registros de overflow de bucket. Os

ponteiros na lista encadeada devem ser ponteiros de registros, que incluem tanto um endereço de bloco quanto a posição relativa do registro no bloco.

Este esquema de hashing descrito é chamado de hashing estático, porque o número de buckets M alocados é fixo. Isto representa um grande problema para arquivos dinâmicos. Suponhamos que tenhamos alocado M buckets para o espaço de endereçamento e que seja m o número de máximo de registros que cabem em um bucket, se tivermos um número bem menor do que m*M registros teremos um desperdício de espaço de armazenamento, se tivermos um número de registros superior a m*M, haverá um número grande de colisões e a recuperação de registros se tornará mais lenta devida às longas listas de registros de overflow. Em ambos os casos talvez seja necessário trocar o número M de blocos alocados e então utilizar uma nova função hash para redistribuição dos registros.

A pesquisa por um registro baseado em um campo diferente do campo chave é tão dispendicioso quanto no caso de um arquivo desordenado. A exclusão de registros pode ser implementada por meio da remoção do registro de seu bucket e se o bucket possuir um encadeamento overflow, trazer um registro do encadeamento para o bucket, se o registro estiver na lista de encadeamento, basta retira-lo da lista.

Modificações em registro dependem de dois fatores: da condição de pesquisa para localizar o registro e do campo a ser modificado. A condição de pesquisa se refere quanto ao campo que será pesquisado, caso seja o campo da chave hash a pesquisa é eficiente, caso seja outro campo a pesquisa passa a ser linear. Quanto ao campo a ser modificado temos a seguinte situação, se o campo for um campo chave é bem provável que teremos que fazer a remoção deste registro do bucket seguida de uma inclusão do registro modificado, caso a modificação seja em um campo diferente do campo chave basta alterar o registro e reescrevê-lo no mesmo bucket.

9.3 Técnicas hashing que permitem a expansão de arquivos dinâmicos

O grande problema com os esquemas hashing apresentados anteriormente, os chamados hashing estáticos, é que o espaço de endereçamento hash é fixo. Deste modo fica difícil aumentar ou diminuir um arquivo dinamicamente. Apresentaremos dois esquemas para reparar esta situação, o primeiro é chamado de hashing extensível e o segundo é chamado de

hashing linear. Ambos tiram a vantagem de que a aplicação de uma função hashing gera um número inteiro não negativo, podendo ser representado como número binário. Sendo assim a estrutura de acesso é construída sobre a representação binária do resultado da função hash, a qual chamamos de valor hash, fazendo com que os registros seja distribuídos entre os buckets tendo como base os valores de primeiros bits de seu valor hash.

Hashing extensível: neste tipo de hashing é mantido um vetor de 2^d endereços de buckets, onde d é chamado de profundidade global, que funciona como um tipo de diretório. O valor inteiro correspondente aos primeiros d bits de um valor hash é utilizado como índice de um vetor para determinar uma entrada no diretório e o endereço naquela entrada determina o bucket no qual os registros correspondentes serão armazenados. Uma profundidade local d', armazenada em cada bucket, especifica o número de bits no qual os conteúdos dos buckets são baseados.

O valor d pode ser aumentado ou diminuído uma unidade por vez, fazendo a capacidade dobrar ou dividir-se ao meio de acordo com as necessidades de armazenamento. A duplicação se faz necessário quando ocorrer um overflow em um bucket cuja profundidade local d' seja igual à profundidade global d. A divisão ao meio deve ocorrer quando ocorrer d > d' para todos os buckets após a ocorrência de algumas exclusões. A maioria das recuperações de registro exige dois acessos a blocos, um para o diretório e outro para o bucket.

As principais vantagens do hashing extensível são:

- o desempenho do arquivo não degrada conforme o arquivo cresça, em oposição ao hashing estático descrito anteriormente;
- nenhum espaço será alocado para o hashing extensível para crescimento futuro, buckets adicionais são podem ser alocados dinamicamente;
- a sobrecarga para a tabela de diretórios é insignificante;
- a divisão de buckets causará menor número de reorganizações, uma vez que apenas os registros de um bucket serão redistribuídos para os dois novos buckets.

Uma desvantagem será que o diretório deverá ser pesquisado antes de acessar o próprio bucket, resultando em dois acessos a blocos em vez de um no caso do hashing estático.

Hashing Linear: a idéia do hashing linear é permitir que um arquivo hash expanda ou diminua seu número de buckets dinamicamente, sem necessitar de um diretório. Seu funcionamento funciona da seguinte forma: supomos que um arquivo comece com M buckets numerados de 0 a M - 1 e use a função hash h(K) = K mod M, chamada de função hash

inicial. O overflow por causa de colisões ainda será necessário e poderá ser tratado por meio da manutenção de cadeias individuais de overflow para cada bucket. Entretanto quando uma colisão levar a um registro de overflow em qualquer bucket do arquivo, o primeiro bucket do arquivo será dividido em dois buckets; o bucket 0 original e o novo bucket M que será adicionado ao final do arquivo. Os registros originalmente posicionados no bucket 0 serão distribuídos entre os dois buckets segundo uma função hash diferente $h_{i+1}(K) = K \bmod 2M$. Uma propriedade importante das duas funções h_i e h_{i+1} é que quaisquer registros que forem levados pela função hash ao bucket 0 pela função h_i serão levados por h_{i+1} ao bucket 0 ou ao bucket M, isto é de fundamental importância para que o hashing funcione adequadamente.

À medida que ocorre overflow, todos os buckets originais do arquivo terão sido divididos, assim o arquivo passa a possuir 2M buckets ao invés de M, e todos usam a função h_{i+1}. Não há diretório neste tipo de hashing, apenas um número n que recebe inicialmente o valor 0 e é incrementado em um unidade quando a divisão ocorre, que determina a quantidade de buckets que foram divididos.

Ao ser incrementado se o valor de n for igual a M, significa que todos os buckets foram divididos e que a função h_{i+1} se aplica a todos os registros do arquivo, neste caso n é inicializado com valor 0 e quaisquer novas colisões levarão ao uso de uma função hash $h_{i+2}(K) = K \bmod 4M$. Em geral as seqüências de funções hash geradas serão do tipo $h_{i+j}(K) = K \bmod (2^j M)$, onde j é incrementado toda vez que n for reinicializado com valor 0.

Capítulo 10

Outras organizações primárias de arquivos

Todas as organizações apresentadas anteriormente supõem que os registros sejam de um mesmo tipo. Nesta sessão apresentaremos algumas organizações primárias de arquivos visando organizações diferentes das apresentadas até agora, iniciaremos apresentando arquivos com registros mistos e depois apresentaremos outras estruturas de organização primaria de arquivos.

10.1 Arquivo de Registro Misto

Na maioria das aplicações de bancos de dados existem várias situações em que vários tipos de entidade estão inter-relacionadas. Esses relacionamentos podem ser representados por campos de conexão. Se desejarmos recuperar o valor de dois registros relacionados, precisaremos primeiramente recuperar um dos registros, então, a partir de seu campo de relacionamento, conseguiremos recuperar o registro relacionado no outro arquivo. Por este motivo, esses relacionamento são implementados como relações lógicas entre registros.

As organizações de SGBSs orientados a objeto, SGBDs hierárquicos ou mesmo SGBDs em rede freqüentemente implementam relacionamentos entre registros por meio de relacionamentos físicos, obtidos por meio de adjacência física dos registros (clustering) ou por ponteiros físicos. Essas organizações de arquivos normalmente atribuem uma área em disco para manter os registros de mais de um tipo, de modo que os registros de tipo diferente possam estar fisicamente agrupados (clustered) no disco. Se for esperada certa freqüência de utilização de certo relacionamento a implementação física poderá aumentar a eficiência na recuperação dos registros relacionados.

Para distinguir os registros em um arquivo misto cada registro possui um campo adicional tipo de registro, que especifica o tipo do registro. Normalmente este é o primeiro campo de

cada registro e é utilizado pelo SGBD para determinar qual tipo de registro ele esta prestes a processar. Usando essa informação e o dicionário de dados ele consegue determinar cada campo deste registro e seus tamanhos a fim de interpretar os valores contidos nesse registro.

10.2 Árvores-B (B-Trees) e Outras Estruturas de dados como Organizações Primárias

Outras estruturas podem ser utilizadas como organizações primárias de arquivos. Por exemplo, se o tamanho do registro quanto ao número de registros em um arquivo for pequeno, alguns SGBDs oferecem a opção de estrutura de dados de árvore B como organização primária de arquivos, esta, visando a otimização das operações de entrada e saída dos dispositivos. Em geral, qualquer estrutura de dados que possam ser adaptada às características dos dispositivos de armazenamento secundário pode ser utilizada como organização primária de arquivo para a disposição de registros.

Capítulo 11

Acesso Paralelo em disco Usando RAID

Com o crescimento exponencial da capacidade e do desempenho de dispositivos semicondutores e memória, microprocessadores cada vez mais rápidos e com cada vez mais memória estão continuamente se tornando mais usuais. Para acompanhar esse crescimento, é natural esperar que a tecnologia de armazenamento secundário deva caminhar neste mesmo ritmo, porém como muitos fabricantes indicam o crescimento do desempenho de acesso aos dispositivos secundários não cresce nessa mesma taxa, para contornar esse fato, a tecnologia RAID foi desenvolvida [2].

Originalmente RAID significava Redundant Arrays of Inexpensive Disks (Vetor Redundantes de discos baratos), ultimamente o 'I' de RAID é dito Independent (Independente). O principal objetivo de RAID é proporcionar melhoria de desempenho de modo a equiparar as taxas, muito diferentes entre os discos e as memórias dos microprocessadores. Como dito anteriormente, enquanto a capacidade das RAMs quadruplica a cada dois ou três anos, os tempos de acesso ao disco aumentam 10% ao ano [1]. A capacidade de fato é aumentada em mais de 50 % entretanto a melhoria da velocidade de acesso é de magnitudes muito inferiores.

Há uma segunda disparidade qualitativa originada pelos microprocessadores especiais que servem a novas aplicações para processamento de vídeo, áudio e imagens com a correspondente falta de acesso rápido a grandes conjuntos compartilhados. A solução natural é usar um grande vetor de pequenos discos independentes atuando como um único disco lógico de mais alto desempenho. Um conceito que usa o paralelismo para melhorar o desempenho de dados é chamado de striping. O striping de dados distribui os dados de maneira transparente por múltiplos discos como se fosse um único disco grande e rápido. O striping de dados melhora o desempenho total de entrada e saída, fornecendo altas taxas de transferência globais, e também pode ser utilizado para balancear as cargas entre os vários

discos. Além disso, por meio do armazenamento redundante de informações, a confiabilidade pode ser melhorada.

Nesta sessão discutiremos a abordagem RAID como melhoria de desempenho para o armazenamento de arquivos.

11.1 Melhoria de Confiabilidade de RAID

Para um vetor de n discos, a probabilidade de falha é n vezes a de um disco. Por isso caso seja suposto que o MTTF (Mean Time to Failure - Tempo Médio entre Falhas) de uma unidade de disco seja 200.000 horas o MTTF de 100 unidades de disco cai para apenas 2.000 horas. Manter apenas uma cópia de dados em tal vetor de discos causará uma significativa perda de confiabilidade. Uma solução adotava na tecnologia RAID a partir do nível 1 é empregar redundância de dados de modo que falhas de discos sejam toleráveis. As desvantagens para isso são muitas: operações adicionais de entrada e saída para a gravação nos vários discos redundantes, processamento adicional para a manutenção da redundância, e para a execução da recuperação de erros [1].

Uma técnica introduzida com a redundância é chamada espelhamento (mirroring ou shadowing). Os dados são escritos de maneira redundante em vários discos diferente como se fosse um único disco lógico. Quando esses dados forem lidos, eles serão recuperados do disco com menor atraso para recuperação. Se um disco desses falhar automaticamente o disco redundante será usado até que o outro disco seja reparado. Suponha que o tempo médio para a reparação seja de 24 horas, então o tempo médio para a perda de dados de um sistema espelhado usando 100 discos, com MTTF 200.000 horas cada, é de $(200.000)^2/ (2*24) = 8.33*10^8$ [2]. O espelhamento de disco também dobra a taxa de tratamento de solicitações de leitura, visto que quando um disco esta ocupado a informação pode ser recuperada em outro não ocupado.

Uma solução para o problema de confiabilidade é armazenar informações adicionais, que não são normalmente necessárias, mas podem ser utilizadas para reconstruir informações perdidas em caso de falha. A incorporação de redundância deve levar em consideração (1) a seleção de uma técnica para a obtenção das informações redundantes e (2) a seleção de um método para a distribuição das informações redundantes pelo vetor de discos. O primeiro pode ser tratado utilizando códigos de correção de erro envolvendo bits de paridade, ou códigos especializados como os códigos Hamming. Para o segundo problema as duas

principais abordagens são armazenar a informação redundante em um pequeno número de discos ou distribuí-la uniformemente por todos os discos. A ultima opção resulta em melhor balanceamento de carga. Os diferentes níveis de RAID optam por uma combinação dessas opções para implementar a redundância e assim aumentar a confiabilidade.

11.2 Melhoria de Desempenho com RAID

O vetor de discos emprega a técnica de striping de dados para obter taxas de transferências mais altas. Os dados podem ser lidos ou escritos em apenas um bloco por vez, assim uma transferência típica contém 512 bytes. O striping de discos pode ser aplicado a uma granularidade mais fina, por meio da quebra de um byte em bits espalhados e distribuí-los em diferentes discos. Assim o striping de dados no nível de bits consiste na divisão de um byte de dados e de gravação do bit j no j-ésimo disco. Com bytes de 8 bits, oito discos físicos podem ser considerados um único disco lógico aumentando assim a taxa de transferência em 8 vezes. Cada disco participa do pedido de entrada e saída, e a quantidade total de dados lidos por pedido é oito vezes maior. O striping de dados no nível de bits pode ser generalizado para um número de discos que seja múltiplo ou divisor de 8.

Com uma granularidade maior os blocos de um arquivo podem ser separados pelo disco surgindo o striping no nível de blocos. Com o striping no nível de blocos, diversos pedidos independentes que acessem os blocos únicos poderão ser atendidos em paralelo por discos separados, diminuindo assim, o tempo de fila de pedidos de entrada e saída. Pedidos que acessam diversos blocos podem ser realizados em paralelo, reduzindo assim seus tempos de resposta. Geralmente, quanto maior o número de discos em um vetor de discos, maior é o ganho de desempenho.

11.3 Organizações e Níveis de RAID

Foram definidas diferentes organizações de RAID com base em diferentes combinações de dois fatores de granularidade de dados e padrão utilizado para calcular a informação redundante. Tais organizações são conhecidas como "níveis de RAID" no total existem 6, as quais são apresentadas abaixo [6]:

- RAID nível 0 - Também é conhecido como "Striping" ou "Fracionamento". Nele, os dados são divididos em pequenos segmentos e distribuídos entre os discos. Este

nível não oferece tolerância a falhas, pois não existe redundância. Isso significa que uma falha em qualquer um dos discos pode ocasionar perda de informações. Por essa razão, o RAID 0 é usado para melhorar a performance do computador, uma vez que a distribuição dos dados entre os discos proporciona grande velocidade na gravação e leitura de informações. Quanto mais discos houver, mais velocidade é obtida. Isso porque, se os dados fossem gravados em um único disco, esse processo seria feito de forma seqüencial, com o RAID os dados cabíveis a cada disco são gravados ao mesmo tempo.

- RAID nível 1 - Também conhecido como "Mirroring" ou "Espelhamento". Funciona adicionando discos paralelos aos discos principais existentes no computador. Os discos que foram adicionados trabalham como uma cópia do original. Assim, se o disco principal recebe dados, o disco adicionado também os recebe. Daí o nome de "espelhamento", pois um disco passa a ser uma cópia praticamente idêntica do outro. Dessa forma, se um dos discos apresentarem falha, o outro imediatamente pode assumir a operação e continuar a disponibilizar as informações. A conseqüência neste caso, é que a gravação de dados é mais lenta, pois é realizada duas vezes. No entanto, a leitura dessas informações é mais rápida, pois se pode acessar duas fontes. Por esta razão, uma aplicação muito comum deste nível de RAID é seu uso em servidores de arquivos.
- RAID nível 2 - este tipo de RAID adapta o mecanismo de detecção de falhas em discos rígidos para funcionar em discos auxiliares. Assim, todos os discos da matriz ficam sendo "monitorados" pelo mecanismo. Atualmente este nível é pouco usado, uma vez que praticamente todos os discos rígidos novos saem de fábrica com mecanismos de detecção de falhas implantados.
- RAID nível 3 - neste nível os dados são divididos entre os discos da matriz, exceto um que armazena informações de paridade. Assim, todos os bytes dos dados tem sua paridade (acréscimo de 1 bit, que permite identificar erros) armazenada em um disco específico. Através da verificação desta informação, é possível assegurar a integridade dos dados, em casos de recuperação. Por isso e por permitir o uso de dados divididos entre vários discos, o RAID 3 consegue oferecer altas taxas de transferência e confiabilidade das informações.

- RAID nível 4 – basicamente, divide os dados entre os discos, sendo que um é exclusivo para paridade. A diferença entre o nível 4 e o nível 3, é que em caso de falha de um dos discos, os dados podem ser reconstruídos em tempo real através da utilização da paridade calculada a partir dos outros discos, sendo que cada um pode ser acessado de forma independente. O RAID 4 é indicado para o armazenamento de arquivos grandes, onde é necessário assegurar a integridade das informações. Isso porque, neste nível, cada operação de gravação requer um novo cálculo de paridade, dando maior confiabilidade ao armazenamento (entretanto isso torna as gravações de dados mais lentas).
- RAID nível 5 - muito semelhante ao nível 4, exceto o fato de que a paridade não fica destinada a um único disco, mas a toda a matriz. Isso faz com que a gravação de dados seja mais rápida, pois não é necessário acessar um disco de paridade em cada gravação. Apesar disso, como a paridade é distribuída entre os discos, o nível 5 tende a ter um pouco menos de performance que o RAID 4.
- RAID nível 6 - O RAID nível 6 ou também conhecido RAID 0 + 1 é uma combinação dos níveis 0 (Striping) e 1 (Mirroring), onde os dados são divididos entre os discos para melhorar o rendimento, mas também utilizam outros discos para duplicar as informações. Assim, é possível utilizar o bom rendimento do nível 0 com a redundância do nível 1. No entanto, é necessário pelo menos 4 discos para montar um RAID desse tipo. Tais características fazem do RAID 0 + 1 o mais rápido e seguro, porém o mais caro de ser implantado.

Um projetista de configuração de RAID para um determinado conjunto de aplicações precisa enfrentar muitas decisões de projeto, tais como o nível de RAID, o numero de discos, a escolha entre os esquemas de paridade e o agrupamento para o striping. Estas decisões são essenciais para um bom desempenho de um sistema empregando essa tecnologia.

Capítulo 12

Área de armazenamento em Rede

Para as atuais organizações direcionadas para a Internet, tornou-se necessário mudar de uma operação com centro de dados (data center) fixo e estático para uma infra-estrutura mais flexível e dinâmica, dado seus requisitos de processamento de informação. O custo total para manutenção de todos os dados está crescendo tão rapidamente que em alguns casos o custo de gerenciamento do servidor conectado ao armazenamento é mais caro que o próprio servidor [2]. Muitos usuários de RAID não podem usar efetivamente sua capacidade porque precisam estar conectados de uma maneira fixa a um ou mais servidores. Por essa razão, grandes organizações estão adotando um conceito chamado Área de Armazenamento em Redes (SAN - Storage Area Networks). Em uma SAN, periféricos de armazenamento on-line são configurados como nós em uma rede de alta velocidade e podem ser conectados e desconectados dos servidores de maneira bastante flexível. Elas permitem que os sistemas de armazenamento sejam posicionados a grandes distancias dos servidores, proporcionando diferentes opções de conectividade e de desempenho.

Algumas das alternativas de arquitetura atuais para SAN incluem:

- conexões 'ponto-a-ponto' entre servidores e sistemas de armazenamento por meio de canal de fibra óptica;
- uso de um switch com canal de fibra para conectar múltiplos sistemas RAID bibliotecas de fitas etc. a servidores;
- uso de hubs e switches com canal de fibra para conectar servidores e sistemas de armazenamento em diferentes configurações.

As SANs têm crescido muito rapidamente, mas ainda enfrentam muitos problemas, tais como a combinação de opções de armazenamento de diferentes fornecedores e negociações envolvendo os padrões de software e hardware de gerenciamento e armazenamento.

Capítulo 13

Conclusões

Considerando o trabalho realizado, verificamos que muita informação referente a esse assunto pode ser encontrado na literatura. Essas informações são de grande importância para inicio de projeto de banco de dados, elas devem ser consideradas por qualquer administrador de banco de dados se o mesmo pretende que sua aplicação tenha desempenho aceitável.

Referências Bibliográficas

[1] CORTEZ, E. Unidade 2 – Organização de Arquivos e Conceitos de Memória Secundária. Disponível em: http://www.dcc.ufam.edu.br/~eccv/bd1/slides/U2-Arquivos-mem_sec.pdf. Ultimo acesso: Setembro de 2009.

[2] ELMASRI, M.; NAVATHE S. B. Sistemas de Banco de Dados. 4 ed. Addison-Wesley, 2003.

[3] Morimoto, C. E. Como funciona a memória RAM. Disponível em: http://www.gdhpress.com.br/hmc/leia/index.php?p=cap3-3. Ultimo acesso: Setembro de 2009.

[4] Morimoto, C. E. Memória Cache. Disponível em: http://www.guiadohardware.net/termos/memoria-cache. Ultimo acesso: Setembro de 2009.

[5] Melanson, D. TDK looks to deliver 2.5TB hard drives in early 2010. Disponível em: http://www.engadget.com/2009/08/07/tdk-looks-to-deliver-2-5tb-hard-drives-in-early-2010/. Ultimo acesso: Setembro de 2009.

[6] Alecrim, E. Tecnologia RAID. Disponível em: http://www.infowester.com/raid. php. Ultimo acesso em: Setembro de 2009.