Organização de arquivos

Por que ver sistema de arquivos?

O sistema de arquivos é quem realmente se responsabiliza por guardar os dados que um aplicativo ou mesmo o banco de dados gerencia. Não é tarefa de um administrador de banco de dados escolher entre um ou outro esquema de organização, mas o seu entendimento nos ajuda a compreender como os dados efetivamente são guardados. Além disto, linguagens de desenvolvimento ainda bastante utilizadas como o Clipper não usam gerencador de banco de dados, mas acessam diretamente um arquivo guardado no disco.

Veremos também que a escolha de um meio de armazenamento implica em uma organização de arquivos específica devido às suas limitações. Dispositivos como fitas magnéticas permitem apenas o uso seqüencial enquanto discos rígidos por exemplo permitem acesso aleatório (a qualquer parte do arquivo).

Meios de armazenamento

Como sabemos de IPD (Introdução ao Processamento de Dados), para que o processador possa trabalhar com alguma informação, ela precisa estar na memória do computador. Esta memória (RAM) é um dispositivos que depende da alimentação do computador, ou seja, quando o computador é desligado, o conteúdo desta memória é apagado. Para guardar as informações, portanto, devemos recorrer a um meio externo que independente de o computador estar ligado ou não elas permaneçam armazenadas.

Alguns aspectos devem ser considerados na hora de se escolher um dispositivo de armazenamento:

Capacidade: total de dados que pode ser guardado.
Portabilidade: permite o transporte para que os dados sejam guardados em um outro local.
Método de acesso: como os dados serão lidos do dispositivo.
Taxa de transferência: indica o quão rápido o computador troca informações com o dispositivo.
Compartilhamento: capacidade que o dispositivo tem de ser compartilhado. Dispositivos de acesso seqüencial não são indicados para serem compartilhados.

Fitas magnéticas

A fita magnética foi o primeiro meio externo de armazenamento a ser largamente utilizado e hoje ainda é usado principalmente para backup e arquivamento de dados. Embora o preço de uma fita seja menor que o disco, o acesso aos dados é muito mais lento pois as informações são lidas seqüencialmente desde o início. Portanto o uso da fita não é aconselhável para o ambiente de processamento online.

Disco flexível (disquete)

O disco flexível é um dispositivo externo ainda muito utilizado para backup e transferência de arquivos entre computadores. É um dispositivo muito barato que possui um acesso muito lento, também não recomendado para uso em produção. Com o advento de discos removíveis de maior capacidade e taxas de transferência maiores, o disquete está sendo “deixado de lado” sendo que em alguns computadores eles já foram abolidos.

Disco rígido (Winchester ou HD)

O disco rígido é um dispositivo externo que pode ser usado tanto online como offline. Devido à sua grande capacidade, e taxa de transferência alta, ele é usado no ambiente de produção. Para se ter uma idéia da importância deste dispositivo os Sistemas Operacionais utilizam-no para armazenamento e também como uma "extensão" da memória principal do computador. Assim se um programa "grande" precisa ser executado, o SO pega parte da memória e guarda no HD liberando a memória principal para o programa. Este processo de troca entre memória e disco se chama **swap**.

O processador pode se comunicar com o disco rígido através de duas interfaces: IDE ou SCSI. Os discos IDE são os mais comumente utilizados pelos computadores pessoais, pois oferece uma boa taxa de transferência a um preço relativamente baixo. Já a interface SCSI oferece maiores velocidades bem como a capacidade de ligar mais periféricos simultaneamente. Em servidores de banco de dados, onde o trabalho com informações armazenadas em disco é freqüente, recomenda-se o uso de um disco SCSI.

Discos óticos (CD, DVD)

Os discos óticos têm grande capacidade de armazenamento, porém o tempo de acesso é menor que o disco rígido. Eles são utilizados para armazenamento de informações permanentes para o caso de mídias não graváveis (como CD-ROM) ou como dispositivos de backup para mídias regraváveis (como CD-RW). Assim como os discos rígidos elas podem ser ligados tanto à interface IDE como SCSI.

Discos removíveis (Zip drive, Jazz drive)

Também têm uma grande capacidade de armazenamento e tempo de acesso pequeno, porém não é tão rápido como um HD. São ótimos dispositivos para backup e transferência de arquivos entre computadores, sendo candidatos naturais à substituição dos discos flexíveis. A grande desvantagem está no preço que ainda é muito salgado.

Método de acesso

Um arquivo é organizado em uma seqüência de registros os quais são mapeados em blocos no disco. Para acessar estes registros gravados no arquivo, podemos percorrer um a um desde o início até encontrar o registro desejado, podemos acessar diretamente um registro específico do arquivo e podemos também acessar um registro baseado em uma "tabela" auxiliar. Dessa forma podemos dividir os arquivos em:

1. Seqüencial: os registros são todos percorridos desde o início até que se encontre o registro desejado.
2. Direto: um determinado registro em qualquer posição do arquivo pode ser acessado diretamente.
3. Indexado: existe uma tabela auxiliar "indice" que contém as localizações dos registros no arquivo principal.

A escolha de um ou outro método de acesso vai depender muito do tipo de consulta e processamento que queremos ter para os dados lidos do arquivo.

Podemos efetuar consultas a um arquivo de dados de três formas:

- Consulta simples: um valor definido é fornecido para consulta e o arquivo é pesquisado para que este valor seja encontrado. Ex.: *estado='GO'*.

- Consulta por faixa de valores: é fornecido uma faixa de valores para os quais o registro deve ser procurado. Ex.: *data entre '01/01/2003' e '15/01/2003'*
- Consulta booleana: consiste de uma combinação de consultas simples, faixa ou de ambas. Ex.: e*stado='GO' e data entre '01/01/2003' e '15/01/2003'*

A forma de processamento que queremos para os dados guardados em arquivo também influencia na escolha do modo de acesso. Processamentos do tipo online e tempo real requerem um tempo de resposta pequeno, o que implica na necessidade das informações estarem disponíveis o mais rápido possível. Em um sistema de reservas de passagens a informação acerca da disponibilidade de lugares deve ser imediatamente vista e deve refletir a situação naquele exato momento. Um acesso seqüencial é contra-indicado neste tipo de processamento.

Processamentos do tipo batch (em lote), onde os dados são processados em seqüencia e não requerem tempo de resposta imediato, geralmente tem uma liberdade maior para estabelecer uma escolha entre os métodos de acesso. Um relatório de clientes para mala direta é um processamento por natureza seqüencial, pois todos os clientes devem ser processados.

Podemos ter também uma mescla de processamento online e batch. Operações bancárias são um exemplo. Durante o dia, os clientes movimentam sua conta efetuando depósitos, saques ou pagamentos. Estas transações são feitas online, o que requer que elas estejam disponíveis assim que executadas. O banco pode resolver este problema mantendo dois arquivos: um arquivo mestre contendo os lançamentos da conta e um arquivo de transações o qual guarda as transações do dia. O processamento do arquivo de transações é feito online e ao final do dia, as transações são incorporadas ao arquivo mestre num processamento batch.

Arquivo seqüencial

O método de acesso seqüencial é o mais conhecido e mais freqüentemente utilizado. Num arquivo seqüencial, a ordem lógica e física dos registros armazenados é a mesma. Como os registros são armazenados um após o outro em seqüencia, a leitura de um registro n requer que os n-1 registros anteriores também sejam lidos. Historicamente o método seqüencial é associado à fita magnética devido à natureza seqüencial do meio de armazenamento, porém é possível gravar arquivos seqüenciais em meios de acesso direto como discos.

O principal uso dos arquivos seqüenciais é o processamento em série ou seqüencial de registros. Por exemplo um relatório de produtos ou totalização de contas, que requer a leitura de todo o arquivo em seqüencia, pode ser beneficiado com o uso de arquivo seqüencial. Em média metade do arquivo deve ser lida para encontrar um determinado valor.

Um arquivo seqüencial pode ser ordenado por um determinado valor. Num arquivo seqüencial ordenado, cada registro tem um item de dado (campo) chave que serve para manter a ordenação do arquivo. Nos arquivos sem ordenação por uma chave, a ordem é a de gravação, ou seja, eles estão armazenados na ordem em que foram adicionados no arquivo.

Para adicionar registros em um arquivo seqüencial sem chave, basta incluir o registro no final do arquivo. Quando o arquivo é ordenado por uma chave, temos um problema: como manter a ordem do arquivo? A única forma de inserir um registro e manter a ordenação é fazer uma cópia do arquivo até o ponto de inserção, inserir o novo registro e copiar o restante do arquivo. Devido ao grande overhead gerado por esta inserção, geralmente os registros a serem inseridos são agrupados em lotes (ordenado) e depois é feito um processamento no arquivo original para inserir todos os registros de uma só vez. Outra alternativa é inserir os registros ao final do arquivo e depois classificá-los na seqüencia adequada.

Remover um registro do arquivo (ordenado ou não) requer que os registros restantes sejam juntados para ocupar o espaço do registro removido. Para fazer isto basta fazer uma cópia do arquivo sem os registros removidos. Uma alternativa para reduzir o overhead é marcar os registros a serem removidos e posteriormente fazer uma cópia do arquivo removendo os registros marcados.

A atualização de um registro é simples no arquivo sem chave: basta ler o registro, alterá-lo e gravá-lo novamente. Já arquivos ordenados mais uma vez vão apresentar problemas. Se o valor da chave de ordenação for alterado, todos os registros adjacentes devem ser movidos para manter a classificação do arquivo. Assim como na inserção, pode-se agrupar as atualizações em lotes e depois em um único processamento no arquivo original faz-se a atualização de todos os registros alterados.

Arquivo de acesso direto

No método de acesso direto, existe uma relação direta entre a chave de ordenação e o endereço no dispositivo de armazenamento. Os registros são lidos e atualizados através do uso desta relação. Arquivos com acesso direto permite a busca de registros individuais com um tempo de resposta bem melhor pois para um dado registro na posição n não é necessário ler os n-1 registros. Na verdade o ponteiro do arquivo é movido para a posição onde o registro está armazenado e então é feita a leitura do registro.

O endereço usado para localizar um registro pode ser absoluto ou relativo. O endereço absoluto significa o endereço real do registro no arquivo. Se indicarmos o endereço absoluto 4, isto significa que o quarto registro armazenado no arquivo é o que será processado. O endereço relativo, significa o deslocamento a partir da posição atual do arquivo. Se o ponteiro estiver no registro 4 e indicarmos um endereço +4, o ponteiro será movido 4 posições adiante e assim o registro 8 será processado. Se indicarmos um endereço -2, o ponteiro será movido três posições para trás e dessa forma o registro 3 será processado.

Arquivo Indexado

Podemos fazer uma analogia de um arquivo com um livro. Logo no início, existe um índice onde estão relacionados os tópicos e a página onde cada tópico está localizado. A idéia do arquivo indexado é justamente esta: existe um arquivo auxiliar (índice) que contém as chaves e o endereço do registro correspondente no arquivo principal (dados). Num banco de dados, para cada arquivo de dados, podemos ter um ou mais índices, dependendo dos valores de chave para ser utilizados. Um arquivo de clientes poderia ser classificado por ordem alfabética ou pelo número do CPF do cliente. Para satisfazer as duas chaves, seriam criados então dois arquivos de índices: um tendo o nome e o outro tendo o CPF como chave de classificação.

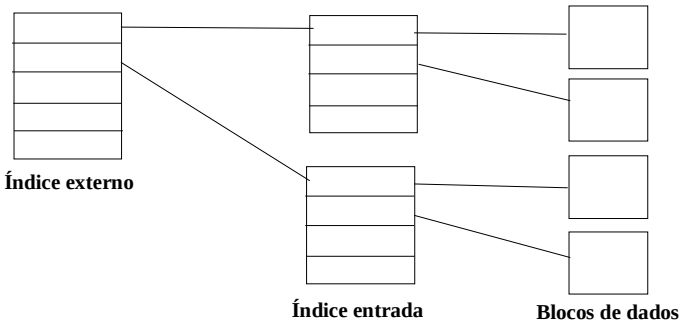
O arquivo de índice pode ser criado de duas formas:

- **Índice denso**: existe um registro no arquivo de índices para cada valor de chave no arquivo principal.
- **Índice esparso**: registros de índice são criados para apenas alguns registros. Outros valores de chave são procurados a partir do registro de índice mais próximo.

O índice denso consegue um tempo de resposta menor para localizar um valor de chave qualquer, porém o índice esparso oferece a vantagem de ocupar menos espaço e precisa de menos manutenção para inclusões e exclusões.

Caso o arquivo de índices seja pequeno o ideal é que o mantenhamos na memória

principal onde o acesso é muito mais rápido que no disco. Assim, caso o arquivo de índices cresca a ponto de não caber na memória, poderiamos tratá-lo como um arquivo seqüencial normal e construiríamos um novo índice esparso no arquivo de índices. Dessa forma para pesquisar um valor de chave, procuramos no índice externo que contém um apontador para um bloco do arquivo de índices, lemos o bloco para encontrar o bloco no arquivo principal que contém a chave de busca.



Indexação esparsa em dois níveis

Arquivo indexado em árvore B+

A organização em arquivo seqüencial indexado, apresenta uma queda de desempenho à medida que o arquivo vai crescendo. É claro que uma reindexação consegue resolver o problema, mas o custo é bastante alto pois é um processo que demanda tempo e o sistema deve estar parado para a reindexação. A estrutura em árvore B+ é a mais utilizada pelos SGBD (Sistemas Gerenciadores de Banco de Dados) pois, apesar de perder eficiência nas inclusões e exclusões, não requer uma reindexação constante do arquivo.

A árvore B+ é uma árvore de pesquisa de caminho múltiplo e de crescimento restrito. Uma árvore B+ de ordem *m* deve satisfazer as seguintes propriedades:

1. Todo nó tem no máximo *m* descendentes.
2. Todo nó, exceto a raiz e os nós terminais tem no mínimo *m/2* (arredondando para mais) descendentes.
3. A raiz tem pelo menos 2 descendentes a menos que seja nó terminal.
4. Todos os nós terminais aprecem no mesmo nível e não tem nenhuma informação.
5. Um nó interno com *k* descendentes contém *k-1* valores de chave.

Aplicando as propriedades acima, temos que uma árvore B+ de ordem 5 deve ter entre 5/2=3 e 5 descendentes contendo 2, 3 ou 4 valores de chave. A raiz pode conter de 1 a 4 valores de chave. Os valores de chave de um nó aparecem ordenados e o número de descendentes é exatamente maior em uma unidade que o número de valores de chave do nó.

A pesquisa se inicia na raiz e uma busca é feita nos seus valores de chave. Se a busca tiver êxito, o valor é localizado, senão existem dois valores de chave no nó entre os quais o valor procurado está. Dessa forma a subárvore entre estes dois valores de chave é pesquisada. O processo é repetido a cada nó e se chegar a um nó terminal sem encontrar o valor de chave procurado, a

pesquisa não teve êxito.

A atualização na árvore Bé relativamente simples: cada nó terminal corresponde a um lugar onde o novo valor de chave pode ser inserido. Se o nó já contiver *m* valores de chave, então ele é fracionado em dois nós e o valor de chave $K_{m/2}$ é inserido no nó pai. Se o nó pai também estiver completo, ele é dividido e assim sucessivamente até a raiz. Caso a raiz tenha que ser dividida, uma nova raiz é criada contendo o valor de chave único $K_{m/2}$. A árvore neste caso cresceu um nível. Assim uma árvore B cresce para cima a partir do topo e não para baixo a partir da base.

A retirada de um valor é mais complicada que a inserção. A retirada de um valor de um nó pode deixá-lo muito vazio (ocorre um *underflow*). Neste caso o irmão é examinado e os valores de chave do irmão é movido até os dois nós apresentarem aproximadamente o mesmo número de valores de chave. Este movimento não ocorre diretamente entre os irmãos, mas o valor de chave anterior do no pai é movido para o nó em underflow e o valor da chave anterior do irmão é movido para o pai.

Embora estas operações de inclusão e exclusão nas árvores B+ sejam complicadas, elas requerem relativamente poucas operações. É possível demonstrar que o número de operações para o pior caso de inclusão ou exclusão é proporcional ao logaritmo do número de chaves de pesquisa. Esta velocidade faz com que esta técnica de indexação seja freqüentemente usada na implementação de banco de dados.

Hashing

O princípio que baseia o hashing é que embora tenhamos um conjunto grande (talvez infinito) de chaves de pesquisa possíveis, o conjunto de chaves armazenadas no banco de dados é muito menor, ou seja, apenas uma parte de todas as chaves possíveis é utilizada. Dessa forma, é implementada uma função de hashing que “transforma” os valores de chave em endereços.

Um determinado endereço não representa um único registro, mas um conjunto de registros nos quais a aplicação da função de hash em suas chaves de pesquisa, resultam em um mesmo endereço. Os registros nesta situação (conflitantes) podem ser organizados em uma lista encadeada a partir do endereço obtido.

A função hash deve ser tal que os conflitos de endereços ocorram de forma balanceada, ou seja, cada endereço tenha aproximadamente a mesma quantidade de registros “pendurados” nele. A pior função de hash mapeia todos os valores de chave para um único endereço. Uma função de hash ideal coloca um endereço para cada valor de chave.

A inserção é bastante simples, basta aplicar a função de hash para encontrar o endereço e se houver conflito, acrescenta o registro ao final da lista. A deleção também é bem simples, bastando remover o registro de onde ele está armazenado.

A função de hash pode ser estática ou dinâmica. A função estática deve ser definida na criação do banco de dados e não pode ser alterada à medida que dados são acrescentados ou removidos. Existem algumas formas de resolver este problema, mas nenhuma delas é uma solução satisfatória. A função de hash dinâmica muda com o tamanho do banco de dados, mas o seu problema é a dificuldade na implementação.

Exercícios

1. Comente sobre a importância do sistema de arquivos para o administrador de banco de dados
2. Na revista info de Fevereiro/2003, seção Tira Teima, tem uma comparação entre um CD-RW externo e um ZIP 750. Comente a reportagem e indique qual a melhor escolha para uma empresa de porte médio que necessita de um sistema robusto, porém simples para efetuar backup, obviamente justificando sua indicação.
3. Além dos dispositivos apresentados na apostila, existem outros interessante, como smart cards, e até dispositivos que se parecem com chaveiros. Faça uma pesquisa (internet, revistas, catálogos, lojas on-line, fabricantes) e cite as características (capacidade, velocidade, comunicação, preço) de alguns destes dispositivos (pelo menos 3).
4. Um problema que ocorre no sistema de arquivo é a fragmentação. Pesquise porque ela acontece e como o sistema operacional gerencia um arquivo fragmentado.
5. Considere um sistema para registro de ponto, onde um funcionário chega no computador e registra suas entradas e saídas diárias. Este sistema suporta um processamento seqüencial? Justifique.
6. Considere agora um sistema para emissão de etiquetas de endereço (mala direta) ordenada por cidade. O processamento seqüencial é indicado? Explique.
7. Num supermercado, existe um depósito onde as mercadorias são armazenadas antes de ir para a prateleira. O estoque das prateleiras é reposto uma vez durante o dia antes do início do expediente e com quantidade suficiente para o dia inteiro de venda. Antes de abastecer porém as quantidades devem ser atualizadas e conferidas. Depois de abastecidas as prateleiras, o supermercado é aberto e as vendas são realizadas durante todo o dia. Comente como seria o processamento (on-line, batch) de cada um destes processos (atualização, conferência, venda).
8. Explique o problema da inclusão de registros em um arquivo de clientes seqüencial em que os registros estão em ordem alfabética.
9. Explique como um índice denso pode melhorar a performance de resposta em pesquisas.
10. Explique por que o índice esparso resulta em economia de espaço.

Para você aprender mais:

Considere um arquivo de produtos em que os registros têm a seguinte estrutura:

```
registro produto
        inteiro: código;
        string: descrição;
        inteiro: quantidade;
        real: preço;
fim registro;
```

Implemente um programa simples em Pascal (que todos viram) para manipular um arquivo seqüencial que deverá estar ordenado pela descrição do produto. Não precisa enfeitar o programa com telas “bonitinhas”. Ele deve apenas ser capaz de incluir, alterar e remover registros do arquivo mantendo a ordenação. A cada modificação do arquivo, o programa deverá mostrar na tela o conteúdo do arquivo (todos os registros).

Bons estudos a todos