10 Indices

Revisão: 13/07/2002

Abrangência

Versão 5.07	Versão 5.08	Versão 6.09	Versão 7.10	Versões Anteriores

A utilização de índices para a pesquisa deve ser bem analisada a fim de evitar lentidão ou processamentos redundantes nos relatórios.

Os índices ajudam a otimizar as pesquisas e laços de processamentos, por isto não devem ser subtilizados ou utilizados de forma errônea.

Caso a necessidade do relatório exija uma otimização que não é possível obter com os índices padrões do sistema é possível criar índices temporários através da função Indregua tornando assim os relatórios mais ágeis e bem estruturados.

Criando e Deletando Arquivos de trabalho (Temporários)

Quando criamos um arquivo de trabalho ou um índice de trabalho (utilizando a função Indregua) no final do programa devemos apaga-los.

Para criação de um índice de Trabalho (Temporário) com Indregua:

```
cArqNtx  := CriaTrab( NIL, .F. ) //Criando Arquivo
IndRegua( "SRA", cArqNtx, cIndCond, , cFor, STR0039 )  //Selec.registros..."
```

Para deletar este índice de trabalho no final do processamento:

```
DbSelectArea( “SRA” ) //Selecionando a area
DbSetOrder( 1 ) //Posicionando na ordem de origem
fErase( cArqNtx + OrdBagExt() ) //Deletando arquivo de trabalho
```

Caso o programa que crie um arquivo de trabalho e não o apague no final de seu processamento, este ficará ocupando espaço em disco no ambiente de trabalho. Isto poderá gerar problemas futuros para o cliente. Por isto, é fundamental, que após sua utilização o mesmo seja descartado.

Utilizando Querys no Protheus

Podemos utilizar querys no Protheus quando acessamos bancos de dados via TopConnect.

As querys, quando bem construídas, melhoram enormemente a eficiência (velocidade) das consultas aos dados e reduzem a sobrecarga no servidor de aplicação, TopConnect e Banco de Dados.

Normalmente uma query é utilizada em substituição a um Loop (While) na base de dados de programação convencional. Querys mais complexas utilizando joins poder ser construídas com a mesma função de vários loops.

Dicas Importantes - DBF versus SQL

A princípio não existem diferenças na programação para a versão SQL, já que pelo próprio fato de ser uma linguagem interpretada, o sistema é quem se encarrega de executar os comandos e funções adequadamente no ambiente em que trabalha. Mas é importante manter algumas informações em mente ao programar para o ambiente SQL.

Deve-se lembrar que estamos trabalhando com um banco de dados relacional, que se utiliza de tabelas ao invés de arquivos, e onde o sistema não tem acesso aos dados de forma nativa e sim através do Top Connect. Essa forma de acesso adiciona ao sistema algumas das características e vantagens oferecidas pelo SGBD em uso (por exemplo, o Oracle, MSSQL Server ou o DB2) como por exemplo segurança e integridade referencial, e as imensas facilidades da linguagem SQL, mas por outro lado tem-se também as implicações da conversão dos comandos no padrão xBase para a perfeita compreensão no ambiente SQL.

Imagine a montagem de uma expressão de filtro para um índice condicional. Tome a seguinte expressão como exemplo: "DTOS(E1_VENCTO) >= DTOS(mv_par01)". Em um ambiente padrão xBase, como o NTX ou o ADS, pode-se utilizar variáveis sem qualquer problema em uma expressão de filtro pois a mesma será avaliada registro a registro durante a montagem do índice. Mas no ambiente SQL, o filtro nada mais é do que uma tabela temporária, onde estão selecionados apenas os registros conforme a condição indicada. A seleção de dados em tabelas pelo SQL é mais rápida, mas em conpensação o SGBD não tem como reconhecer a variável informada na expressão. Ela existe apenas no sistema ou, mais especificamente, no seu programa. Por isso, deve-se substituir a expressão anteriormente exemplificada pela seguinte (que também funcionaria perfeitamente em um ambiente xBase): "DTOS(E1_VENCTO) >= ‘"+DTOS(mv_par01)+"’". Esta expressão é melhor que anterior simplesmente porque não se utiliza da variável e sim do conteúdo da mesma, o que pode ser compreendido em qualquer ambiente. Toda essas explicações são válidas, da mesma maneira, a filtros criados através do comando SET FILTER.

Ainda existem outros detalhes a se considerar quando se trabalha com índices em um ambiente SQL. É que na verdade não existem índices condicionais nesse ambiente. O filtro é criado independente do índice. Então, você pode criar um INDREGUA com um filtro e mudar a ordem, mas o filtro permanecerá ativo, em qualquer ordem. Do mesmo modo, não se pode manter dois índices, com filtros diferentes, pois um filtro sobrescreveria o outro.

Outro ponto de atenção deve ser a função xBase chamada DBSETINDEX. Podem ocorrer alguns erros ao tentar-se utilizar essa função para abrir um índice de trabalho criado. Por esses motivos e pelo fato de tornar o processamento mais lento deve-se evitar ao máximo o uso de índices de trabalho no ambiente SQL.

Da mesma maneira que a funcão DBSETINDEX, os comandos COPY TO e APPEND FROM também devem ter uma atenção especial. No ambiente SQL esses comandos são executados entre uma tabela e um arquivo DBF (e vice-versa) ou entre dois arquivos DBF. Por exemplo, o comando COPY TO pode ser usado para copiar os dados da tabela ativa para um DBF local e o comando APPEND FROM pode ser usado para importar os dados de um arquivo local para a tabela ativa. Os dois podem ser usados entre dois arquivos, mas nunca pode-se usar, por exemplo, o comando APPEND FROM para importar os dados de uma tabela para outra.

Grupos Relacionados

Principal / Guias de Referência / Como programar Advpl no ERP

Topo da Página