04 Outros Cuidados

Revisão: 13/07/2002

Abrangência

Versão 5.07	Versão 5.08	Versão 6.09	Versão 7.10	Versões Anteriores

Um dos cuidados que devemos ter quando da criação de relatórios contendo valores é a utilização dos subtotais e totais, a fim de evitar erros que podem ser desastrosos durante uma tomada de decisão errada devido a valores errados.

A utilização de somatórias deve ser bastante criteriosa a fim de não cometermos o erro de misturarmos unidades de medidas diferentes no mesmo cálculo.

Confrontando relatórios e consultas

Quando elaboramos um sistema, existem muitos relatórios que geram dados para outros relatórios e consultas.

Devemos tomar cuidado para que não aconteçam divergências de informações de um para o outro, como por exemplo, no caso de valores.

Um bom exemplo disso, é a rotina de impressão de folha de pagamento. Este relatório exibe informações que são utilizadas em outros relatórios, tais como, valores para o FGTS, guia de recolhimento de impostos.

Uma solução para que não se ocorra uma divergência de valores, seria utilizar uma única função ou rotina de processamento. Isto evitaria que ao se alterar o sistema, por motivo de lei ou outro qualquer, o programador alterasse por exemplo às rotinas de relatório de folha de pagamento e guia de impostos e esquecesse de alterar por exemplo à rotina de relatório de FGTS.

Exemplos como Saldos Bancários, Quantidades de Estoques, Valores de Faturamento, entre outros, devem ser confrontados entre relatórios e consultas para não gerarem informações errôneas ao cliente.

Normalmente estes problemas ocorrem em funções de critérios de filtragens diferenciados entre eles. Para evitar este tipo de problema é fundamental que o analista ao efetuar alguma manutenção em algum relatório ou consulta atente-se ao fato de assegurar que esta alteração não influencie outras situações.

Este é um tipo de não conformidade simples de ser evitada e que pode causar problemas sérios para os usuários além de ser de dificil argumentação quando nos questionado, pois evidencia falta de atenção ou critério na manuteção ou falta de conhecimento sobre o funcionamento do sistema.

Problemas com Looping de Programas

O Protheus utiliza a tecnologia Cliente/Servidor. Isto significa que o aplicativo não é mais executado individualmente em cada máquina, ele será executado no servidor do aplicativo. Até a versão 4.07 um programa travado significava que apenas a estação estava comprometida (o executável estava na memória da estação). Com o Protheus, todo o processamento está no Server e quando o programa

está em looping estaremos gradativamente “usando todo a CPU do Server” e consequentemente parando todo o processamento.

Se ao desenvolvermos uma rotina e a mesma entrar em looping (tiver apenas uma entrada e não tiver uma saída do processamento), este processamento utilizará todos os recursos do servidor comprometendo (reduzindo drasticamente a performance do aplicativo), ou até impedindo, o uso do aplicativo por todos os demais usuários.

Se isso acontecer em uma empresa onde existem apenas 5 usuários, o administrador da rede poderá reiniciar o servidor, porém onde existe um número considerável de usuários poderá haver um prejuízo para a empresa que utiliza nosso sistema.

Exemplo:

```
dbSeek(xFilial(“SE1”)+DTOS(dDtIni))
Do While SE1->(!Eof())
// demais comandos , etc.
<---------- Falta um DbSkip()
Enddo
```

No exemplo acima, a rotina ficará em looping (pois falta um comando de saída da rotina, um DbSkip() seria o mais apropriado), utilizando todos os recursos de processamento do servidor, fazendo com que o mesmo pare de funcionar.

Outro exemplo:

```
aCampos := {}
Do while .T.
   Aadd(aCampos, “Teste”)
Enddo
```

No exemplo acima o caso é ainda mais crítico, pois além utilizar todo o recurso de processamento do servidor, em dado momento haverá uma queda do aplicativo, devido a limitação da variável tipo Array, criada acima. E quando este limite for ultrapassado, o sistema será interrompido abruptamente e todos os demais usuários ficarão impossibilitados de utilizarem o sistema.

Manipulação de Arquivos Externos ao Protheus

A manipulação de arquivos considerados externos ao Protheus deverá ter um tratamento diferenciado. O arquivos a serem manipulados (alterados/consultados) deverão ser copiados do Client para o Server e vice-versa utilizando uma conexão (TPC-IP,IPX,etc). Para copiar os arquivos, foram criadas duas funções que serão executadas via conexão, a CPYS2T() encarregada de copiar do Server para o Client/Terminal e a CPYT2S() encarregada de copiar do Client/Terminal para o Server.

O editor de texto Word da Microsoft, os arquivos de imagens (BMP,JPEG,etc) exigem um lugar físico para abertura dos documentos/imagens, navegando pela Internet por exemplo são copiados via conexão para um diretório temporário no computador para serem visualizados.

O AP5 trabalha da mesma forma, através dessas considerações e utilizando a arquitetura Client/Server via conexão os arquivos serão copiados.

Em alguns Módulos do Protheus são encontradas rotinas de Importação/Exportação de lançamentos, exigindo serem utilizadas as funções CPYT2S() e CPYS2T() para manipulação dos arquivos. Por exemplo, uma importação de lançamentos da Folha de Pagamento poderá ser feita diretamente do Client sem precisar copiar para o Server mas se outro usuário precisar visualizar os lançamentos de origem da importação não terá acesso, agora se for realizado a cópia do Client para o Server todos poderão visualizar (aconselhável). Isso acontece no Módulo de Controle de Documentos, quando todos os arquivos (documentos) são copiados entre o Client e o Server para que todos visualizem e manipulem. Um exemplo que não há necessidade de cópia são os arquivos gerados para contabilização (CPROVA), pois estes são gerados no próprio Server não havendo necessidade de cópia.

Os arquivos que poderão ser copiados deverão estar necessariamente embaixo do RootPath na configuração do Server, isto é, o diretório DOCS do exemplo abaixo deverá ser sub-diretório do RootPath.

Exemplo de cópia do Server para o Client:

```
CPYS2T("\DOCS\EXEMPLO.DOC,"C:\WINDOWS\TEMP",.T.)
```

Onde os parâmetros são:

1o. é o <Nome do Arquivo> a ser copiado para o Client
2o. é o <Nome do Diretório> do Client e/ou local físico onde será copiado o arquivo.
3o. se deseja compactar o arquivo (recomendável)

Exemplo de cópia do Client para o Server:

```
CPYT2S("C:\WINDOWS\TEMP\EXEMPLO.DOC,"\DOCS",.T.)
```

Onde os parâmetros são:

1o. é o <Nome do Arquivo> a ser copiado para o Server
2o. é o <Nome do Diretório> do Server
3o. se deseja compactar o arquivo (recomendável)

As funções possuem um retorno True(.T.) ou False(.F.) indicando se a cópia foi realizada com sucesso ou não.

Desenvolvendo Telas

A aparência e objetividade das telas num sistema é base fundamental da interface Sistema x Usuário.

O AP5 já cria, automaticamente, a grande parte das telas de um módulo, tais como a Browse, a GetDados e Enchoice.

Algumas outras telas necessitam de construção “manual”, ou seja, com a utilização de comandos,

tais como “SAY” , “GET” e “LABEL”, na Dialog.

Procure sempre colocar em tela as informações que mais se objetivam com o assunto abordado.

Sempre que possível, dê preferência aos campos obrigatórios primeiro. Isso facilita a digitação do usuário, que não precisará passar de campo em campo (no caso de estar utilizando a tecla <TAB>) até chegar ao campo desejado. A ordem dos campos também é importante para a fácil localização das informações.

Quando o volume de informações é muito grande, divida os campos em folders, ou seja, pastas, agrupando os campos em assuntos. Isso irá deixar a tela menos poluída e evitará que o usuário navegue por uma tela só. Para fazer essa facilidade, preencha o campo X3_FOLDER, no SX3, com um número, agrupando-os de acordo com a tipo de informação e no SXA, com o ALIAS do arquivo em pauta, a ordem, que equivale ao numero informado no X3_FOLDER e a descrição nos três idiomas. Essa descrição que será a informação contida na pasta do folder. Exemplo: Os campos SZ1_ENDER, SZ1_NUM e SZ1_BAIRRO devem estar com o campo X3_FOLDER preenchido com o conteúdo “1”. No SXA, o XA_ALIAS deverá ser SZ1, o XA_ORDEM = “1” (mesmo valor preenchido no X3_FOLDER), no XA_DESCRIC, “Endereço Residencial” e, nos demais, o mesmo texto em outros idiomas.

O Folder, além de agrupar e facilitar a procura pelos campos, evita a rolagem vertical da tela, facilitando a visualização das informações.

Evite tela com muitos botões. Isso poderá confundir o usuário e induzi-lo ao erro. Utilize telas sequenciais, conhecidas como Wizard (semelhante aos de instalação de um software). Dessa forma, o usuário ficará mais atento aos fatos, dificultando o erro. Mas cuidado: não faça disso uma incansável sequência de telas, pois isso acabará desmotivando o usuário a utilizar o sistema.

Enfim, as telas devem ser limpas e objetivas, de tal forma que impeça o usuário de sair de seu objetivo final. Todo curioso irá apertar todos os botões da tela ou preencher todos os campos com qualquer tipo de informação. Portanto, esteja atento a tamanho dos labels, para que os mesmos não excedam o tamanho da caixa de diálogo definida. Isso, além de não ser estético, prejudica o entendimento da informação.

Salvando Array’s padrões

Quando temos Janelas que necessitem apresentar mais de uma getdados, devemos salvar os elementos, acols, aheader e n, da tela anterior para apresentar uma nova janela.

As principais variáveis são:

```
Acols = Array contendo as linhas usada que serão apresentadas na Getdados
AHeader = Array contendo o cabeção das colunas da Getdados
N = Variável publica que indica a posição do atual no acols
    (a Linha  que está sendo editada na Getdados)
```

Para salva-las podemos:

```
aColsAnt := aClone(Acols)
aHeaderAnt := aClone(aHeader)
nElemAnt := n
```

E para restaura-las:

```
aCols := aClone(aColsAnt)
aHeader := aClone(aHeaderAnt)
n := nElemAnt
```

Grupos Relacionados

Principal / Guias de Referência / Como programar Advpl no ERP

Topo da Página