Quando você utiliza o Visual FoxPro para criar e executar aplicativos, deseja obter o melhor desempenho do seu sistema operacional, do Visual FoxPro e do seu aplicativo.

Para obter informações sobre como otimizar o seu computador e sistema operacional, consulte o capítulo 4, Otimizando o sistema, no *Guia de Instalação e Índice Principal*.

Este capítulo aborda os seguintes tópicos:



# Otimizando tabelas e índices

Você pode acelerar o acesso a dados em tabelas utilizando índices e buffer com eficiência. Além disso, você pode utilizar a tecnologia Rushmore para otimizar suas consultas.

## Utilizando índices

Para acelerar o acesso a dados em uma tabela, utilize índices. Adicionar um índice a uma tabela acelera a procura, principalmente se você puder utilizar a tecnologia Rushmore para otimizar a sua procura. A indexação também permite o trabalho com dados em uma ordem especificada, como visualizar uma tabela de clientes em ordem pelo sobrenome.

Se os registros em uma tabela tiverem chaves únicas, crie um índice primário ou candidato no campo. Estes tipos de índices permitem que o Visual FoxPro torne válida a chave em um nível inferior, resultando em um melhor desempenho.

Além de indexar campos utilizados para procura e classificação, você também deve indexar qualquer campo envolvido em uma associação. Se você associar duas tabelas em campos não indexados, a operação de associação poderá durar muito mais tempo.

Um recurso importante do Visual FoxPro é a possibilidade de criar um índice em qualquer expressão. (Em alguns produtos de banco de dados, somente é possível a indexação em campos). Esta capacidade permite a utilização de índices para otimizar a procura, classificação ou associação em combinações de campos ou em expressões derivadas de campos. Por exemplo, você pode indexar um campo de nomes com base em uma expressão que utiliza a função SOUNDEX( ). Desta forma, o seu aplicativo pode fornecer acesso extremamente rápido a nomes que parecem semelhantes.

Ao adicionar índices a tabelas, você deve comparar os benefícios obtidos em momentos de recuperação com a perda de desempenho ao atualizar a tabela. Ao adicionar mais índices à sua tabela, as atualizações e inserções na tabela tornam-se mais lentas, pois o Visual FoxPro precisa atualizar cada índice.

Por último, evite utilizar índices em campos que contêm apenas alguns valores distintos, como um campo lógico. Nestes casos, o índice contém apenas um pequeno número de entradas e a despesa na manutenção do índice provavelmente supera os benefícios fornecidos durante a procura.

Para obter informações detalhadas sobre como indexar com eficiência ao utilizar a tecnologia Rushmore, consulte Utilizando a tecnologia Rushmore para acelerar o acesso a dados posteriormente neste capítulo.

## Otimizando associações

Ao criar associações utilizando SELECT - SQL, as seguintes situações podem reduzir o desempenho e causar resultados inesperados:

- Associar tabelas em dados que não são uma chave primária ou única em uma das tabelas.
- Associar tabelas que contêm campos vazios.

Para evitar essas situações, crie associações com base no relacionamento entre as chaves primárias em uma tabela e as chaves estrangeiras em outra. Se você criar uma associação com base em dados não únicos, o resultado final poderá ser o produto de duas tabelas. Por exemplo, a instrução SELECT - SQL a seguir cria uma associação que pode causar um resultado muito grande:

```
SELECT *;
 FROM  tastrade!customer INNER JOIN tastrade!orders ;
   ON  Customer.postal_code = Orders.postal_code
```

No exemplo, o código postal identifica com exclusividade uma localização em uma cidade, mas possui pouco valor se a sua intenção for corresponder as linhas de clientes com as suas linhas de pedidos. Nem sempre o código postal identifica com exclusividade um cliente ou um pedido. Em vez disso, crie uma associação utilizando uma instrução como a seguinte:

```
SELECT *;
 FROM  tastrade!customer INNER JOIN tastrade!orders ;
   ON  Customer.customer_id = Orders.customer_id
```

Neste exemplo, o campo `customer_id` identifica com exclusividade um cliente e os pedidos pertencentes a esse cliente e logo cria um conjunto de resultados que combina a linha de cliente com cada linha de pedido.

Além disso, cuidado ao associar tabelas a campos vazios pois o Visual FoxPro irá corresponder campos vazios. No entanto, o Visual FoxPro não corresponde campos que contêm valores nulos. Ao criar uma associação, qualifique as expressões de campo na condição de associação testando para uma seqüência vazia.

Por exemplo, se você acha que o campo de identificação de cliente na tabela Pedidos pode estar vazio, utilize uma instrução como a seguinte para filtrar os registros de pedido sem nenhum número de cliente.

```
SELECT *;
 FROM  tastrade!customer INNER JOIN tastrade!orders ;
   ON  Customer.customer_id = Orders.customer_id;
   WHERE tastrade!orders <> ""
```

**Dica** Você também pode testar para uma seqüência vazia utilizando a função EMPTY( ), mas incluir uma chamada de função na expressão de filtro não é tão rápido quanto comparar com um valor constante.

## Utilizando o Gerenciador de projetos

Ao utilizar o **Gerenciador de projetos**, você pode combinar um número ilimitado de programas e procedimentos em um arquivo .APP ou .EXE único. Isso pode aumentar muito a velocidade de execução do programa por vários motivos.

Em primeiro lugar, o Visual FoxPro abre um arquivo de programa e o deixa aberto. Quando, mais tarde, você utilizar um comando DO em um programa que está contido no arquivo, o Visual FoxPro não precisará buscar ou abrir um arquivo adicional.

Em segundo lugar, um aplicativo com apenas um ou dois arquivos reduz o número de arquivos necessários no diretório de trabalho. A velocidade de todas as operações com arquivos aumenta, já que o sistema operacional possui menos entradas de diretórios para examinar ao abrir, renomear ou excluir arquivos.

Para obter informações sobre a utilização do **Gerenciador de projetos** para criar aplicativos, consulte o capítulo 13, Compilando um aplicativo.

## Sugestões gerais de otimização de tabela e índice

Para criar as tabelas e os índices mais rápidos possíveis, siga as recomendações listadas abaixo.

- Se o buffer de tabela ou registro não estiver ativado, utilize INSERT - SQL em vez de APPEND BLANK seguido de REPLACE, especialmente com uma tabela indexada em um ambiente multiusuário, pois os índices só precisam ser atualizados uma vez.
- Se você precisar incluir um número grande de registros em uma tabela indexada, talvez seja mais rápido remover o índice, incluir os registros e, então, recriar o índice.
- Em instruções SQL, evite chamadas de função, se possível; principalmente em instruções que retornarão mais de um registro, pois a instrução deve ser reavaliada (e as funções chamadas novamente) para cada registro. Se você estiver criando uma instrução SQL com dados variáveis, utilize expressões de nome ou substituição de macro a favor da função EVALUATE( ). A melhor estratégia, entretanto, é construir a instrução inteira dinamicamente, não apenas sentenças individuais. Para obter maiores informações, consulte Utilizando macros e Criando expressões de nome.
- Se você normalmente utilizar uma determinada ordem de índice, poderá aprimorar o desempenho classificando a tabela nessa ordem periodicamente.
- Utilize arquivos .CDX em vez de .IDX em ambientes multiusuário pois é mais rápido atualizar um arquivo .CDX do que atualizar vários arquivos .IDX.

# Utilizando a tecnologia Rushmore para acelerar o acesso a dados

Para ajudá-lo a otimizar o desempenho dos seus aplicativos, o Visual FoxPro inclui a tecnologia de acesso a dados Rushmore. Utilizando a tecnologia Rushmore, você pode executar algumas operações com tabelas bastante complexas de cem a mil vezes mais rápido do que antes.

## Conhecendo a tecnologia Rushmore

A tecnologia Rushmore é uma técnica de acesso a dados que utiliza índices padronizados do Visual FoxPro para otimizar o acesso a dados. Você pode utilizar essa tecnologia com qualquer índice do Visual FoxPro, inclusive índices do FoxPro 1.*x* (.IDX), índices compactados (.IDX) ou índices compostos (.CDX).

Tanto índices .CDX quando índices compactados .IDX utilizam uma técnica de compressão que produz índices cujos tamanhos correspondem a 1/6 dos índices do formato antigo não-compactados. O Visual FoxPro pode processar índices compactos com maior rapidez porque requer menos acessos a disco para processar um índice e retém porções maiores de um índice em seus buffers de memória. Assim como outras técnicas de acesso a arquivos, a tecnologia Rushmore opera melhor com índices compactados devido ao seu tamanho reduzido, além de funcionar muito bem com índices em formatos antigos.

Quando o Visual FoxPro processa tabelas muito grandes em computadores com somente o mínimo de memória RAM, talvez não haja memória suficiente para a tecnologia Rushmore operar. Neste caso, uma mensagem de aviso é exibida (“Memória insuficiente para otimização”). Embora seu programa funcione corretamente e sem perder dados, a consulta não utilizará a tecnologia de otimização Rushmore.

Em sua forma mais simples, Rushmore agiliza o desempenho de comandos de uma tabela única utilizando cláusulas FOR que especificam conjuntos de registros de acordo com os índices existentes. Além disso, ela pode agilizar a operação de determinados comandos, como LOCATE e INDEX. Para obter uma lista completa dos comandos otimizáveis, consulte a próxima seção “Utilizando a Tecnologia Rushmore com Tabelas”.

Os comandos SQL do Visual FoxPro utilizam a tecnologia Rushmore como uma ferramenta básica na otimização de consultas de tabelas múltiplas, utilizando índices existentes e até mesmo criando novos índices ad hoc para agilizar as consultas.

# Utilizando a tecnologia Rushmore com tabelas

Utilize a tecnologia Rushmore para otimizar o acesso a dados de acordo com o número de tabelas envolvidas. Quando você acessa tabelas únicas, pode tirar proveito da tecnologia Rushmore onde quer que apareça uma cláusula FOR. Quando você acessa múltiplas tabelas, consultas SELECT - SQL substituem todas as otimizações com a tecnologia Rushmore. Em um comando SQL, o Visual FoxPro decide o que é necessário para otimizar uma consulta e faz o trabalho para você. Você não precisa abrir tabelas nem índices. Se a SQL decidir que precisa de índices, ela criará índices temporários para uso próprio.

**Para utilizar a tecnologia Rushmore**

Escolha uma das seguintes opções:

- Para acessar dados de uma tabela única, utilize a cláusula FOR em um comando como AVERAGE, BROWSE ou LOCATE ou utilize comandos SQL para atualizar tabelas. Para obter uma lista completa dos comandos que usam cláusulas FOR, consulte a tabela a seguir.

  – Ou –

- Para acessar dados de mais de uma tabela, utilize os comandos SELECT - SQL, DELETE - SQL e UPDATE - SQL.

A tabela a seguir lista os comandos que utilizam cláusulas FOR. A tecnologia Rushmore é estruturada para que sua velocidade seja proporcional ao número de registros recuperados.

**Comandos potencialmente otimizáveis com cláusulas FOR**

| | |
|---|---|
| AVERAGE | BLANK |
| BROWSE | CALCULATE |
| CHANGE | COPY TO |
| COPY TO ARRAY | COUNT |
| DELETE | DISPLAY |
| EDIT | EXPORT TO |
| INDEX | JOIN WITH |
| LABEL | LIST |
| LOCATE | RECALL |
| REPLACE | REPLACE FROM ARRAY |
| REPORT | SCAN |
| SET DELETED | SET FILTER |
| SORT TO | SUM |
| TOTAL TO | |

Se você utilizar uma cláusula de escopo além de uma expressão otimizável com a cláusula FOR, é preciso que o escopo seja ALL ou REST para que a tecnologia Rushmore possa ser utilizada. Cláusulas de escopo NEXT ou RECORD desativam esta tecnologia. Já que o escopo padrão é ALL para a maioria dos comandos, Rushmore funciona quando você omite a cláusula de escopo.

A tecnologia Rushmore pode utilizar qualquer índice aberto, exceto índices filtrados e UNIQUE.

**Observação** Para um melhor desempenho, não defina a ordem da tabela.

A criação de índices ou marcas define a ordem automaticamente. Se você quiser tirar o máximo proveito da tecnologia Rushmore, com um grande conjunto de dados que precisa estar em uma determinada ordem, emita SET ORDER TO para desativar o controle de índice e, em seguida, utilize o comando SORT.

## Indexando com eficiência para a tecnologia Rushmore

A tecnologia Rushmore não pode tirar proveito de todos os índices. Se você utilizar uma cláusula

FOR no comando INDEX, a tecnologia Rushmore não pode utilizar o índice para otimização. Por exemplo, pelo fato de conter uma cláusula FOR, a instrução a seguir não pode ser otimizada:

```
INDEX ON ORDNUM FOR DISCOUNT > 10 TAG ORDDISC
```

De forma semelhante, Rushmore não pode utilizar um índice criado com uma *condição* NOT. Por exemplo, a expressão a seguir pode ser otimizada:

```
INDEX ON DELETED() TAG DEL
```

Mas esta não pode:

```
INDEX ON NOT DELETED() TAG NOTDEL
```

No caso especial em que você deseja excluir registros deletados de uma consulta, utilizar um índice, como o primeiro exemplo acima, acelerará as operação quando você tiver definido SET DELETED como ativado (ON).

### Operando sem a tecnologia Rushmore

As operações de recuperação de dados prosseguem sem a otimização através da tecnologia Rushmore nas seguintes situações:

- Quando a tecnologia Rushmore não puder otimizar expressões com a cláusula FOR em um comando potencialmente otimizável. Para obter maiores informações sobre a criação expressões FOR otimizáveis, consulte a seção Combinando expressões básicas otimizáveis.
- Quando um comando que poderia utilizar a tecnologia Rushmore tiver uma cláusula WHILE.
- Quando houver pouca memória. A recuperação de dados continua, mas não é otimizada.

## Desativando a tecnologia Rushmore

Você pode desativar a tecnologia Rushmore, apesar de isso ser pouco desejável. Quando você emite um comando que utiliza esta tecnologia, o Visual FoxPro imediatamente determina os registros que correspondem à expressão com cláusula FOR. Esses registros são, então, manipulados pelo comando.

Se um comando potencialmente otimizável modificar a chave de índice na cláusula FOR, o conjunto de registros, no qual a tecnologia Rushmore está operando, irá se tornar desatualizado. Neste caso, você pode desativar a tecnologia para garantir que possui as informações mais atuais da tabela.

**Para desativar a tecnologia Rushmore para um único comando**

- Utilize a cláusula NOOPTIMIZE.

Este comando LOCATE, por exemplo, não está otimizado:

```
LOCATE FOR DueDate < {01/01/95} NOOPTIMIZE
```

Você pode ativar ou desativar a tecnologia Rushmore globalmente para todos os comandos que a utilizam, com o comando SET OPTIMIZE.

**Para desativar a tecnologia Rushmore globalmente**

- Utilize o seguinte código:

```
SET OPTIMIZE OFF
```

**Para ativar a tecnologia Rushmore globalmente**

- Utilize o seguinte código:

```
SET OPTIMIZE ON
```

A definição padrão da otimização através da tecnologia Rushmore é ativada (ON).

## Otimizando expressões com a tecnologia Rushmore

A tecnologia Rushmore depende da presença de uma *expressão básica otimizável* em uma cláusula FOR ou SQL WHERE. Uma expressão básica otimizável pode formar uma expressão inteira ou pode aparecer como parte de uma expressão. Você também pode combinar expressões básicas

para formar uma expressão complexa otimizável.

## Criando expressões básicas otimizáveis

Uma expressão básica otimizável pode assumir uma das seguintes formas:

*eÍndice relOp eExpr*
– Ou –
*eExpr relOp eÍndice*
Uma expressão básica otimizável possui as seguintes características:

- *eÍndice* corresponde exatamente à expressão na qual um índice é construído.
- *eExpr* é qualquer expressão e pode incluir variáveis e campos de outras tabelas não relacionadas.
- *relOp* é um dos seguintes operadores relacionais: <, >, =, < =, > =, < >, #, == ou !=. Você também pode utilizar as funções ISNULL( ), BETWEEN( ) ou INLIST( ) (ou seus equivalentes SQL como IS NULL, etc.).

Você pode utilizar BETWEEN( ) ou INLIST( ) de uma das seguintes formas:

*eÍndice* BETWEEN(*eÍndice, eExpr, eExpr*)
– Ou –
*eExpr* INLIST(*eÍndice, eExpr*)
**Observação** ISBLANK( ) e EMPTY( ) não são otimizáveis na tecnologia Rushmore.

Se você criar os índices firstname, custno, UPPER(lastname) e hiredate, cada uma das seguintes expressões será otimizável:

```
firstname = "Fred"
custno >= 1000
UPPER(lastname) = "SMITH"
hiredate < {12/30/90}
```

Uma expressão otimizável pode conter variáveis e funções que resultam em um valor específico. Utilizando o índice addr, por exemplo, se você emitir o comando STORE 'WASHINGTON AVENUE' TO cVar, as seguintes instruções também serão expressões básicas otimizáveis:

```
ADDR = cVar
ADDR = SUBSTR(cVar,8,3)
```

## Compreendendo quando as consultas são otimizadas

É importante compreender quando as consultas serão otimizadas e quando não serão. O Visual FoxPro otimiza condições de procura, procurando uma correspondência exata entre o lado esquerdo de uma expressão de filtro e uma expressão de chave de índice. Por isso, a tecnologia Rushmore só poderá otimizar uma expressão se você procurar em relação à expressão exata usada em um índice.

Por exemplo, suponha que você tenha acabado de criar uma tabela e esteja adicionando o primeiro índice utilizando um comando como o seguinte:

```
USE CUSTOMERS
INDEX ON UPPER(cu_name) TAG name
```

O comando a seguir não é otimizável, pois a condição de procura baseia-se apenas no campo cu_name, não em uma expressão que é indexada:

```
SELECT * FROM customers WHERE cu_name ="ACME"
```

Em vez disso, você deve criar uma expressão otimizável utilizando um comando como o seguinte, no qual a expressão que você está procurando corresponde exatamente a uma expressão indexada:

```
SELECT * FROM customers WHERE UPPER(cu_name) = "ACME"
```

**Dica** Para determinar o nível de otimização com a tecnologia Rushmore sendo utilizada, chame SYS(3054).

## Combinando expressões básicas otimizáveis

Você poderá combinar expressões simples ou complexas com base na cláusula FOR ou WHERE para aumentar a velocidade de recuperação de dados, se as expressões FOR tiverem as características de expressões básicas otimizáveis.

As expressões básicas podem ser otimizáveis. Você pode combinar expressões básicas utilizando os operadores lógicos AND, OR e NOT para formar uma expressão com a cláusula FOR complexa que também pode ser otimizável. Uma expressão criada com uma combinação de expressões básicas otimizáveis é totalmente otimizável. Se uma ou mais expressões básicas não forem otimizáveis, talvez a expressão complexa seja parcialmente otimizável ou não seja de forma alguma otimizável.

Um conjunto de regras determina se uma expressão composta por expressões básicas otimizáveis ou não otimizáveis é total ou parcialmente otimizável ou não é otimizável. A tabela a seguir resume as regras de otimização de consultas com a tecnologia Rushmore.

**Combinando expressões básicas**

| Expressão básica | Operador | Expressão básica | Resultado da consulta |
|---|---|---|---|
| Otimizável | AND | Otimizável | Totalmente Otimizável |
| Otimizável | OR | Otimizável | Totalmente otimizável |
| Otimizável | AND | Não otimizável | Parcialmente otimizável |
| Otimizável | OR | Não otimizável | Não otimizável |
| Não otimizável | AND | Não otimizável | Não otimizável |
| Não otimizável | OR | Não otimizável | Não otimizável |
| — | NOT | Otimizável | Totalmente otimizável |
| — | NOT | Não otimizável | Não otimizável |

Você pode utilizar o operador AND para combinar duas expressões otimizáveis em uma expressão totalmente otimizável:

```
FIRSTNAME = "FRED" AND HIREDATE < {12/30/89}        && Otimizável
```

Neste exemplo, o operador OR combina uma expressão básica otimizável com uma expressão não otimizável para criar uma expressão que não é otimizável:

```
FIRSTNAME = "FRED" OR "S" $ LASTNAME          && Não otimizável
```

A utilização do operador NOT em uma expressão otimizável cria uma expressão totalmente otimizável:

```
NOT FIRSTNAME = "FRED"     && Totalmente otimizável
```

Você também pode utilizar parênteses para agrupar combinações de expressões básicas.

## Combinando expressões complexas

Da mesma forma que combina expressões básicas, você pode combinar expressões complexas para criar uma expressão mais complexa que seja total ou parcialmente otimizável ou não otimizável. É possível, em seguida, combinar essas expressões mais complexas para criar expressões que, novamente, sejam total ou parcialmente otimizáveis ou não otimizáveis. A tabela a seguir descreve os resultados da combinação dessas expressões complexas. Estas regras também se aplicam às expressões agrupadas entre parênteses.

**Combinando expressões complexas**

| Expressão | Operador | Expressão | Resultado |
|---|---|---|---|
| Totalmente | AND | Totalmente | Totalmente |

| | | | |
|---|---|---|---|
| otimizável | | otimizável | otimizável |
| Totalmente otimizável | OR | Totalmente otimizável | Totalmente otimizável |
| Totalmente otimizável | AND | Parcialmente otimizável | Parcialmente otimizável |
| Totalmente otimizável | OR | Parcialmente otimizável | Parcialmente otimizável |
| Totalmente otimizável | AND | Não otimizável | Parcialmente otimizável |
| Totalmente otimizável | OR | Não otimizável | Não otimizável |
| — | NOT | Totalmente otimizável | Totalmente otimizável |
| Parcialmente otimizável | AND | Parcialmente otimizável | Parcialmente otimizável |
| Parcialmente otimizável | OR | Parcialmente otimizável | Parcialmente otimizável |
| Parcialmente otimizável | AND | Não otimizável | Parcialmente otimizável |
| Parcialmente otimizável | OR | Não otimizável | Não otimizável |
| — | NOT | Parcialmente otimizável | Não otimizável |
| Não otimizável | AND | Não otimizável | Não otimizável |
| Não otimizável | OR | Não otimizável | Não otimizável |
| — | NOT | Não otimizável | Não otimizável |

Você pode combinar expressões totalmente otimizáveis com o operador OR para criar uma expressão que também seja totalmente otimizável:

```
* Expressão completamente otimizável
(FIRSTNAME = "FRED" AND HIREDATE < {12/30/89}) ;
   OR (LASTNAME = "" AND HIREDATE > {12/30/88})
```

Para criar expressões parcialmente otimizáveis, combine uma expressão totalmente otimizável com uma expressão que não seja otimizável. No exemplo a seguir, o operador AND é usado para combinar as expressões:

```
* Expressão parcialmente otimizável
(FIRSTNAME = "FRED" AND HIREDATE < {12/30/89}) ;
   AND "S" $ LASTNAME
```

Expressões parcialmente otimizáveis podem ser combinadas para criar uma expressão que também seja parcialmente otimizável:

```
* Expressão parcialmente otimizável
(FIRSTNAME = "FRED" AND "S" $ LASTNAME) ;
   OR (FIRSTNAME = "DAVE" AND "T" $ LASTNAME)
```

A combinação de expressões que não sejam otimizáveis cria uma expressão que também não é otimizável:

```
* Expressão que não é otimizável
("FRED" $ FIRSTNAME OR "S" $ LASTNAME) ;
   OR ("MAIN" $ STREET OR "AVE" $ STREET)
```

## Otimizando formulários e controles

Você também pode fazer melhorias significativas nos formulários e controles de seu aplicativo.

**Dica** Para obter informações sobre como definir e obter propriedades com eficiência, consulte Fazendo referência a propriedades de objetos de modo eficiente posteriormente neste capítulo.

## Utilizando o ambiente de dados

Se você utilizar o ambiente de dados do **Criador de formulários** ou **Criador de relatórios**, o desempenho de abertura de tabelas será mais rápido do que a execução dos comandos USE, SET ORDER e SET RELATION no evento Load do formulário. Quando você utiliza o ambiente de dados, o Visual FoxPro utilizará chamadas de mecanismo de nível inferior para abrir as tabelas e configurar os índices e as relações.

## Limitando o número de formulários em um conjunto de formulários

Utilize conjuntos de formulários apenas quando for necessário que um grupo de formulário compartilhe uma sessão de dados privada. Ao utilizar um conjunto de formulários, o Visual FoxPro cria ocorrências de todos os formulários e controles em todos os formulários do conjunto de formulários, mesmo se o primeiro formulário no conjunto de formulário for o único exibido. Isto pode levar tempo e ser desnecessário se os formulários não tiverem que compartilhar uma sessão de dados privada. Em vez disso, você deve executar DO FORM para outros formulários, quando forem necessários.

No entanto, se você utilizar um conjunto de formulários, terá maior desempenho quando acessar os formulários no conjunto de formulários, pois estes já estarão carregados, mas não visíveis.

## Carregando controles de página dinamicamente em uma moldura de página

Assim como os conjuntos de formulários, as molduras de página carregam todos os controles de cada página quando a moldura de página é carregada, podendo causar um atraso nítido durante o carregamento da moldura de página. Em vez disso, você pode carregar dinamicamente os controles de página, conforme necessário, criando uma classe fora dos controles em cada página e, em seguida, carregando-os quando a página for ativada.

### Para carregar dinamicamente controles de página

1. Crie o seu formulário como você faria normalmente, incluindo todos os controles em todas as páginas.
2. Quando o projeto estiver concluído, vá para a segunda página da moldura de página e salve os controles encontrados ali como uma classe.
3. Abra a classe criada e certifique-se de que os controles já estejam dispostos devidamente.
4. Repita os Passos 2 e 3 para a terceira página e as páginas subseqüentes da moldura de página.
5. No evento Activate da segunda página e das páginas subseqüentes da moldura de página, adicione os objetos e torne-os visíveis.

   Por exemplo, se a sua classe de controles se chamasse `cnrpage1`, você adicionaria o seguinte código:

```
IF THIS.ControlCount = 0
   THIS.AddObject("cnrpage1","cnrpage1")
   THIS.cnrpage1.Visible = .T.
ENDIF
```

## Acoplando controles a dados dinamicamente

Você pode agilizar o tempo de carregamento de um formulário que contém vários controles ligados a dados, se você retardar o acoplamento destes controles até que eles sejam necessários.

### Para acoplar controles a dados dinamicamente

1. Coloque as tabelas e visualizações do seu formulário no ambiente de dados, de modo que sejam abertas quando o formulário for carregado.

2 Para cada controle de ligação, adicione código ao seu código de evento GotFocus, que acopla o controle ao valor dos dados. Por exemplo, o código a seguir acopla um controle de caixa de combinação ao campo `customer.company`:

```
* Verifica se o controle já foi ligado.
IF THIS.RecordSource = ""
   * Define a origem do registro com o valor correto
   * e define o tipo de origem de registro com "campos"
   THIS.RecordSource = "customer.company"
   THIS.RecordSourceType = 6
   THIS.Refresh
ENDIF
```

## Retardando atualização na tela

Se você tiver que fazer várias alterações na tela — por exemplo, alterar os valores de vários controles de uma vez — poderá reduzir o tempo geral necessário para atualizar a tela, retardando a atualização da tela até que todas as alterações sejam feitas. Por exemplo, Se você tornar os controles visíveis ou invisíveis, alterar as cores de controle ou mover os registros em controles de ligação, é muito mais eficiente retardar a pintura desses controle após todas as alterações terem sido concluídas:

**Para retardar a atualização na tela**

1 Defina a propriedade LockScreen do formulário como verdadeira.

2 Atualize os controles, conforme necessário.

3 Chame o método Refresh do formulário.

4 Defina a propriedade LockScreen do formulário como falsa.

Portanto, o exemplo a seguir altera as propriedades de exibição de várias propriedades de uma só vez, move para um novo registro e só atualiza a tela com as novas informações. Se a propriedade LockScreen não estivesse definida como verdadeira, cada uma dessas operações repintaria os controles afetados individualmente e o desempenho de atualização geral pareceria lento.

```
THISFORM.LockScreen = .T.
THISFORM.MyButton.Caption = "Save"
THISFORM.MyGrid.BackColor = RGB (255, 0, 0)   && Vermelho
SKIP IN customers
SKIP IN orders
THISFORM.Refresh
THISFORM.LockScreen = .F.
```

**Dica** Esta técnica não fornece qualquer benefício se você for atualizar apenas um único controle.

## Reduzindo código em métodos utilizados com maior freqüência

Como o método Refresh e o evento Paint são chamados freqüentemente, você pode otimizar o desempenho em formulários reduzindo a quantidade de código nestes métodos. De forma semelhante, para acelerar o tempo de carregamento de um formulário, é possível mover o código do evento Init para os eventos utilizados com menos freqüência, como Activate, Click e GotFocus. Em seguida, você utiliza uma propriedade no controle (como Tag ou uma propriedade personalizada) para verificar se o controle já executou o código que só precisa apenas ser executado uma vez.

# Otimizando programas

Ao escrever o código cuidadosamente, você poderá escrever os programas com a maior rapidez possível. Há várias maneiras de melhorar o desempenho do programa no Visual FoxPro:

- Seguindo as sugestões gerais de desempenho de programação abaixo.
- Utilizando expressões de nome em vez de substituição de macro.
- Fazendo referências a propriedades de objeto com eficiência.

## Sugestões gerais de desempenho de programação

Para escrever os programas o mais rápido possível, siga as recomendações listadas abaixo.

- Escolha o tipo de dados correto para os seus dados. Principalmente, utilize o tipo de dados Inteiro para as informações numéricas,sempre que possível, como este é processado com mais eficiência. Sempre que possível, utilize os tipos de dados Inteiro para valores de chave primária e estrangeira, que resultarão em arquivos de dados menores, índices menores (e, portanto, mais rápidos) e associações mais rápidas.

  **Observação** Para obter um exemplo que demonstre como criar um índice menor (e, portanto, mais rápido), execute SOLUTION.APP em VFP\SAMPLES\SOLUTION. Escolha **Visualizar exemplos por lista filtrada**, selecione **Índices** na lista suspensa e, em seguida, **Criar pequenos índices utilizando BINTOC( )** na lista que aparece.
- Evite reabrir os arquivos, o que reduz o desempenho. Em vez disso, atribua arquivos a áreas de trabalho ao abri-los e, em seguida, utilize o comando SELECT para escolher uma área de trabalho específica, conforme necessário.
- Utilize loops FOR … ENDFOR em vez de loops DO WHILE … ENDDO sempre que possível, pois eles são mais rápidos.
- Quando você copia dados de vários campos, SCATTER TO ARRAY é mais rápido do que SCATTER MEMVAR.
- Para utilizar a memória de forma mais eficiente, evite criar objetos antes de precisar deles e não se esqueça de limpar os objetos ao terminar de usá-los para liberar memória.

  **Dica** Você pode testar a quantidade de memória que cada objeto consome chamando a função SYS(1016).
- Envie saída para a janela superior sempre que for possível; atualizar janelas, que não seja a janela superior, é mais lento. Se a saída for exibida de modo a deslocar-se atrás de uma janela, a situação é ainda pior.
- Desative a exibição de status com o comando SET TALK OFF, eliminando o trabalho de atualizar a tela.
- Defina o comando SET DOHISTORY como desativado (OFF), para evitar a atualização da janela de comando sempre que um programa for executado.

## Utilizando expressões de nome em vez de substituição de macro

Se você utilizar expressões de nome em vez de substituição de macro, o desempenho do programa será muito mais aprimorado. Por exemplo, se você atribuir um valor à variável cFile, uma expressão de nome criada com cFile será mais rápida do que uma substituição de macro.

```
cFile = "CUST"
use &cFile      && Substituição de macro: lenta
use (cFile)     && Expressão de nome: mais rápida, preferida
```

## Fazendo referências a propriedades de objeto com eficiência

Ao compreender como o Visual FoxPro trabalha com propriedades e objetos, você pode executar os seus aplicativos com maior eficiência.

### Otimizando referências repetidas a uma propriedade

Ao fazer referência a uma propriedade de objeto com a sintaxe *objeto.propriedade*, o Visual FoxPro deve procurar pelo objeto antes que ele acesse a propriedade. Se você tiver que acessar a propriedade várias vezes, essa estratégia de procura poderá reduzir o desempenho.

Para evitar várias referências ao mesmo procedimento (como em um loop), leia o valor da propriedade em uma variável, faça alterações e, em seguida, defina a propriedade uma vez no final. Por exemplo, o código a seguir preenche a matriz de uma propriedade, criando primeiro uma matriz

na memória, preenchendo-a e, em seguida, definindo a propriedade apenas uma vez no final:

```
* Copia a seqüência para uma variável local
lcChar = THISFORM.cCharString
LOCAL laCharArray[256]  && Cria a matriz local
FOR nCounter = 1 to 256
   laCharArray[x] = SUBSTR(laChar,x,1)
ENDFOR
* Copia a matriz local para a matriz da propriedade
ACOPY(laCharArray,THISFORM.aCharArray)
```

### Fazendo referências a várias propriedades com eficiência

Se você atualizar mais de uma propriedade do objeto, o Visual FoxPro deverá procurar pelo objeto várias vezes, podendo afetar o desempenho. No exemplo a seguir, o código faz com que o Visual FoxPro procure em quatro objetos (como `THISFORM`, `pgfCstInfo`, `pgCstName` e `txtName`) para localizar a propriedade a ser definida. Como o código define duas propriedades, a procura quádrupla é realizada duas vezes.

```
THISFORM.pgfCstInfo.pgCstName.txtName.Value = ;
   "Fred Smith"
THISFORM.pgfCstInfo.pgCstName.txtName.BackColor = ;
   RGB (0,0,0)  & Vermelho-escuro
```

Para evitar este desgaste, utilize o comando WITH … ENDWITH. Este método faz com que o Visual FoxPro localize o objeto uma vez. Por exemplo, o exemplo a seguir realiza as mesmas tarefas que o anterior, só que mais rápido:

```
WITH THISFORM.pgfCstInfo.pgCstName.txtName
   .Value = "Fred Smith"
   .BackColor = RGB (0,0,0)  & Vermelho-escuro
ENDWITH
```

Você também pode armazenar a referência a um objeto em uma variável e depois incluir a variável no lugar da referência ao objeto:

```
oControl = THISFORM.pgfCstInfo.pgCstName.txtName
oControl.Value = "Fred Smith"
oControl.BackColor = RGB (0,0,0)  & Vermelho-escuro
```

## Otimizando controles de ActiveX

Se você utilizar controles de ActiveX ou Automação em seu aplicativo, poderá ajustar perfeitamente o aplicativo para obter o melhor desempenho fora dos controles de ActiveX e Automação.

## Utilizando controles de ActiveX com eficiência

Para obter melhor desempenho, ao utilizar controles de ActiveX em seus formulários, siga as seguintes sugestões:

- Inicie servidores de OLE com antecedência. Os controles ligados a campos gerais geralmente serão melhor executados quando os servidores destes tipos de dados (como o Microsoft Excel ou Word) já estiverem em execução na máquina do cliente.
- Insira objetos “Como Ícone”.  Ao inserir um controle de ActiveX em um campo, insira-o como um ícone ou marcador de lugar em vez de como um objeto inteiro. Este procedimento reduz a quantidade de espaço de armazenamento necessária, pois o Visual FoxPro armazena a imagem de uma apresentação com o objeto, podendo consumir uma grande quantidade de espaço de armazenamento. A inserção de um objeto como um ícone também aumenta o desempenho para o desenho de objetos.
- Utilize controles de imagem. Se você quiser exibir um bitmap (como o logotipo de uma empresa), os controles de imagem são mais rápidos do que os controles de Ligação de OLE.
- Utilize vínculos manuais sempre que possível. Os vínculos manuais em objetos são mais rápidos

pois evitam o tempo de notificação exigido para vínculos automáticos e porque o servidor não precisa ser iniciado para desenhar o objeto. Se você não tiver que atualizar um objeto com freqüência, utilize vínculos manuais.

# Otimizando o desempenho de automação

Se o seu aplicativo interagir com outros aplicativos, você poderá obter melhor desempenho utilizando as seguintes técnicas.

## Evitando várias instâncias do servidor

Em alguns casos, os servidores de OLE (como Microsoft Excel) sempre iniciarão uma nova instância, mesmo se uma já estiver em execução. Para remediar isto e aumentar o desempenho, utilize a função GetObject( ) em vez da CreateObject( ). Por exemplo, a chamada a seguir sempre utilizará uma instância existente, se esta existir:

```
x = GetObject(,"excel.Application")
```

Em comparação, a chamada a seguir cria uma nova instância:

```
x = CreateObject("excel.Application")
```

Se você chamar GetObject( ), mas o servidor ainda não estiver em execução, ocorrerá o erro 1426. Neste caso, você poderá localizar o erro e chamar CreateObject( ):

```
ON ERROR DO oleErr WITH ERROR()
x = GetObject(,"excel.application")
ON ERROR  && restaura o manipulador de erro no sistema

PROCEDURE oleErr
PARAMETER mError
IF mError = 1426 then
   x = CreateObject("excel.application")
ENDIF
```

## Fazendo referências a objetos com eficiência

A execução de expressões que utilizam objetos com o servidor de Ole pode ser dispendioso, especialmente quando avaliada várias vezes. É muito mais rápido armazenar referências a objetos em variáveis para referência. Para obter informações detalhadas, consulte Otimizando Referências Repetidas a uma Propriedade anteriormente neste capítulo.

# Otimizando aplicativos em ambientes multiusuários

Se você estiver escrevendo aplicativos para ambientes multiusuários, o desempenho será de extrema importância, pois a falta de eficiência é multiplicada. Além disso, se vários usuários estiverem acessando os dados, o seu aplicativo deverá lidar com questões de compatibilidade e acesso a rede.

Para lidar com essas questões, você pode:

- Ajustar o intervalo de tentativa de bloqueio.
- Utilizar o processamento de transação com eficiência.

É possível que você também possa tirar proveito das sugestões para o trabalho com os dados armazenados em servidores remotos. Para obter informações detalhadas, consulte Otimizando o acesso a dados remotos posteriormente neste capítulo.

## Ajustando o intervalo de tentativa de bloqueio

Se o aplicativo tentar bloquear um registro ou tabela e não obter êxito, você poderá fazer com que o Visual FoxPro tente automaticamente o bloqueio de novo, após um pequeno intervalo. No entanto, cada tentativa de bloqueio resulta em maior tráfego de rede. Se o tráfego de rede já estiver pesado,

enviar várias solicitações de bloqueio sobrecarregará a rede e acarretará na redução da velocidade para todos os usuários.

Para lidar com esta situação, você pode ajustar o intervalo entre as tentativas de bloqueio. Ao utilizar um intervalo maior (que resulta em menos tentativas por segundo), você poderá reduzir o tráfego de rede e melhorar o desempenho.

**Para ajustar o intervalo de tentativa de bloqueio**

- Chame a função SYS(3051), passando a ela o número de milissegundos a ser esperado entre cada tentativa de bloqueio.

### Utilizando processamento de transação com eficiência

Ao utilizar o processamento de transação, você deve criar transações para minimizar o impacto que causam sobre outros usuários. Enquanto uma transação estiver aberta, qualquer bloqueio definido durante a transação permanecerá bloqueado até que a transação seja gravada ou revertida o seu rollback. Mesmo que você emita explicitamente um comando UNLOCK, os bloqueios serão mantidos até o comando END TRANSACTION ou ROLLBACK.

Além disso, a inclusão de registros em uma tabela requer que o Visual FoxPro bloqueie o cabeçalho da tabela. O cabeçalho permanece bloqueado durante toda a transação, prevenindo que outros usuários também incluam registro.

Para minimizar o impacto causado pelas transações, crie-as para que comecem e terminem o mais próximo possível da atualização dos dados real; a transação ideal contém apenas as instruções de atualização de dados.

Se você estiver adicionando o processamento de transação a atualizações de dados feitas em um formulário, não abra uma transação, execute o formulário e, em seguida, grave a transação quando o formulário for fechado. Em vez disso, coloque as instruções de processamento de transação no código de evento do botão **Salvar** (por exemplo):

```
* Salva o método a partir do botão de comando cmdSave
BEGIN TRANSACTION
UPDATE PRODUCTS SET reorder_amt = 0 WHERE discontinued = .T.
END TRANSACTION
```

## Otimizando o acesso a dados remotos

A recuperação de dados, a partir de qualquer banco de dados remoto, é dispendiosa. Para obter dados a partir de um banco de dados do servidor, as seguintes etapas devem ser seguidas:

1. O cliente emite a consulta ao banco de dados remoto.
2. O servidor analisa e compila a consulta.
3. O servidor gera um conjunto de resultados.
4. O servidor notifica a conclusão do resultado ao cliente.
5. O cliente carrega os dados na rede a partir do servidor. Esta etapa pode ser realizada de uma só vez, ou o cliente pode solicitar que os resultados sejam enviados em partes, conforme solicitado.

Você pode utilizar várias técnicas para acelerar a recuperação (ou atualização) de dados. A seção a seguir aborda essas estratégias:

- Recuperando apenas os dados necessários
- Atualizando tabelas remotas com eficiência
- Enviando instruções em um lote
- Definindo o tamanho do pacote
- Retardando a recuperação de dados binários e memo
- Armazenando dados de procura localmente

- Criando regras locais

## Recuperando apenas os dados necessários

Na maioria dos aplicativos que utilizam dados remotos, os formulários e relatórios não precisam acessar todos os dados a partir de uma tabela de uma só vez. Logo, você pode acelerar o desempenho criando visualizações remotas que carreguem ou atualizem apenas os campos e registros necessários, minimizando a quantidade de dados que precisa ser transmitida pela rede.

Para criar consultas que minimizam as despesas da recuperação de dados a partir de fontes remotas, siga estas sugestões:

- Especifique apenas os campos necessários. Não utilize a instrução `SELECT * FROM customers` a não ser que você precise de todos os campos da tabela.
- Inclua uma cláusula WHERE para limitar o número de registros descarregados. Quanto mais específica for a cláusula WHERE, menos registros serão transmitidos para o computador e a consulta terminará mais rápido.
- No momento da criação, se você não conseguir determinar que valores utilizar em uma cláusula WHERE, poderá utilizar parâmetros na cláusula. Quando a consulta é executada, o Visual FoxPro utiliza o valor de uma variável de parâmetro ou solicita ao usuário que procure valores. Por exemplo, esta consulta permite que o aplicativo ou usuário preencha a região em tempo de execução:

```
SELECT cust_id, company, contact, address ;
   FROM customers ;
   WHERE region = ?pcRegion
```

- Defina a propriedade NoDataOnLoad do objeto cursor do ambiente de dados correspondente. Esta técnica é comumente utilizada em visualizações parametrizadas nas quais os dados do parâmetro são provenientes do valor de um controle em um formulário.

## Atualizando tabelas remotas com eficiência

Quando você utiliza uma visualização para atualizar uma tabela em uma fonte de dados remota, o Visual FoxPro deve verificar se o registro ou registros que estão sendo atualizados foram alterados. Para tal, é necessário que o Visual FoxPro analise os dados no servidor e os compare com os dados armazenados no computador. Em algumas instâncias, esta operação pode ser bastante demorada.

Para otimizar o processo de atualização de dados em fontes de dados remotas, você pode especificar como o Visual FoxPro deve verificar os registros alterados. Para tal, indique a cláusula WHERE que o Visual FoxPro deve gerar para executar a atualização.

Por exemplo, suponha que você esteja utilizando uma visualização com base em uma tabela de clientes em uma fonte de dados remota. Você criou a visualização utilizando uma instrução SELECT - SQL como esta:

```
SELECT cust_id, company, address, contact ;
   FROM customers ;
   WHERE region = ?vpRegion
```

Você deseja atualizar todos os quatro campos especificados na visualização, à exceção do campo-chave (`cust_id`). A tabela a seguir ilustra a cláusula WHERE que o Visual FoxPro gerará para cada opção disponível na cláusula SQL WHERE.

**Observação** A função OLDVAL( ) retorna a versão pré-atualizada dos campos modificados e a função CURVAL( ) retorna o valor atual armazenado na fonte de dados remota. Ao compará-las, o Visual FoxPro pode determinar se o registro foi alterado na fonte de dados remota a partir do momento em que foi descarregado no computador.

| Definição | Cláusula WHERE resultante |
|---|---|
| Somente campos-chave | `WHERE OLDVAL(cust_id) = CURVAL(cust_id)` |
| | `WHERE OLDVAL(cust_id) = CURVAL(cust_id) AND` |

| | |
|---|---|
| atualizáveis (padrão) | `OLDVAL(<mod_fld1>) = CURVAL(<mod_fld1>) AND`<br>`OLDVAL(<mod_fld2>) = CURVAL(<mod_fld2>) AND`<br>`...` |
| Campos-chave e modificados | `WHERE OLDVAL(cust_id) = CURVAL(cust_id) AND`<br>`OLDVAL(company) = CURVAL(company) AND`<br>`OLDVAL(contact) = CURVAL(contact) AND`<br>`OLDVAL(address) = CURVAL(address)` |
| Chave e datador | `WHERE OLDVAL(cust_id) = CURVAL(cust_id) AND`<br>`OLDVAL(timestamp) = CURVAL(timestamp)` |

Em geral, você deve escolher uma opção para a cláusula SQL WHERE nesta ordem de preferência:

1. **Chave e datador**, se o banco de dados remoto suportar campos datadores, que representam o meio mais rápido de informar se um registro foi alterado.
2. **Campos-chave e modificados**, pois, na maioria das vezes, os campos atualizados para o servidor são um subconjunto do número total de campos que podem ser atualizados.
3. **Campos-chave e atualizáveis**.
4. **Somente campos-chave**. Utilizar esta definição significa que o servidor remoto inserirá um registro completamente novo utilizando a chave alterada e excluirá o registro antigo.

## Enviando instruções em um lote

Alguns servidores (como o Microsoft SQL Server) permitem que você envie um lote de instruções SQL em um único pacote. Este procedimento aumenta o desempenho, pois reduz o tráfego em rede e porque o servidor pode compilar várias instruções de uma só vez.

Por exemplo, se você especificar um tamanho em lote de quatro e, em seguida, atualizar 10 registro em um banco de dados, o Visual FoxPro enviará quatro instruções como as seguintes para o banco de dados do servidor em um único lote:

```
UPDATE customer SET contact = "John Jones" ;
   WHERE cust_id = 1;
UPDATE customer SET contact = "Sally Park" ;
   WHERE cust_id = 2;
UPDATE customer SET company = "John Jones" ;
   WHERE cust_id = 3;
UPDATE customer SET contact = "John Jones" ;
   WHERE cust_id = 4
```

**Para enviar instruções em um único lote**

- Na caixa de diálogo **Opções**, escolha a guia **Dados Remotos** e, em seguida, em **Registros para atualização em lote**, especifique o número de registro a ser incluído em um lote.

  – Ou –

- Chame as funções DBSETPROP( ) ou CURSORSETPROP( ) para definir estas propriedades:
  - Defina Transaction como 2.
  - Defina BatchUpdateCount com o número de instruções a serem enviadas em um lote.

  – Ou –

1 Em **Criador de visualizações**, escolha **Opções avançadas** no menu **Consulta** para exibir a caixa de diálogo **Opções avançadas**.

2 Na área **Desempenho**, ao lado de **Número de registros para atualização em lote**, especifique o número de instruções a serem enviadas em um lote.

**Observação** Você deve experimentar diferentes valores para esta propriedade e a propriedade PacketSize para otimizar as atualizações.

## Definindo o tamanho do pacote

Você pode otimizar o acesso a servidores remotos ajustando com precisão o tamanho do pacote de

rede que será enviado para, e recuperado do, banco de dados remoto. Por exemplo, se a sua rede suporta tamanhos de pacote grandes (maiores do que 4096 bytes), você pode aumentar o tamanho do pacote no Visual FoxPro para enviar mais dados sempre que ler ou escrever para a rede.

**Para definir o tamanho do pacote**

- Chame as funções DBSETPROP( ) ou CURSORSETPROP( ) e defina a propriedade PacketSize com um valor inteiro positivo. O valor padrão é 4096.

**Observação** Fornecedores de rede distintos lidam com esta propriedade de maneiras diferentes; logo, você deve consultar a sua documentação de serviço de rede. A Novell® NetWare®, por exemplo, tem um tamanho de pacote máximo de 512 bytes, por isso, você não terá benefício algum definindo a propriedade PacketSize com um valor maior do que o especificado.

## Retardando a recuperação de dados binários e memo

Se você estiver armazenando dados binários ou memo em um servidor remoto, poderá aumentar o desempenho retardando o descarregamento destes dados até o momento em que o aplicativo realmente necessitar deles.

**Para retardar a recuperação de dados binários e memo**

- Na caixa de diálogo **Opções**, escolha a guia **Dados remotos** e, em seguida, em **Padrões de visualização remota**, defina **Carregar memo**.

  – Ou –

- Chame as funções DBSETPROP( ) ou CURSORSETPROP( ) para definir a propriedade FetchMemo.

## Armazenando dados de procura localmente

Vários aplicativos incluem dados de procura estáticos, como abreviações de estado, códigos postais e cargos de funcionários. Se o aplicativo contiver este tipo de dados e se a tabela não for muito grande, você poderá agilizar o aplicativo, mantendo cópias destas informações no computador de cada usuário, pois as procuras não geram tráfego de rede.

Esta técnica é particularmente útil para dados que nunca foram alterados ou que raramente são alterados. Se os dados não forem alterados na ocasião, você deverá criar uma estratégia para o descarregamento de uma nova cópia da tabela de procura para o computador de cada usuário.

## Criando regras locais

Você pode aumentar a eficiência do seu aplicativo criando regras locais de nível de campo e nível de registro no Visual FoxPro, em vez de confiar nas regras definidas no servidor. Estas regras podem evitar que os dados que não são compatíveis com as regras comerciais ou com os dados acessem o banco de dados.

Ao definir regras no Visual FoxPro, você captura os dados inválidos antes que eles sejam enviados pela rede, agilizando e fornecendo um melhor controle sobre o gerenciamento das condições de erro. No entanto, utilizar regras locais também significa que você deve coordená-las com as regras no servidor remoto. Por exemplo, se houver alterações nas regras do servidor, é possível que você tenha que alterar as suas regras locais por motivos de conformidade.

Para obter informações detalhadas sobre a criação de regras locais, consulte a seção Atualizando dados em uma visualização no capítulo 8, “Criando visualizações”.

# Otimizando aplicativos internacionais

Se você estiver desenvolvendo aplicativos internacionais, é possível que precise gerenciar a seqüência de ordenação dos dados para um perfeito desempenho. Esta seção aborda os seguintes tópicos:

- Utilizando seqüências de ordenação com eficiência.
- Utilizando SELECT - SQL com várias seqüências de ordenação.

## Utilizando seqüências de ordenação com eficiência

Se os seus dados não incluírem *sinais diacríticos,* como acentos (á) ou tremas (ü), você poderá aumentar o desempenho utilizando a seqüência de ordenação da máquina,pois:

- As chaves de índice, que não constam da máquina, são duplamente maiores pois contêm informações diacríticas.
- A ordenação que não consta da máquina utiliza várias regras especiais para que os caracteres de indexação retornem resultados corretos.

Como a seqüência de ordenação da máquina é mais rápida, geralmente é mais utilizada para associações e procuras, enquanto que outras seqüências de ordenação são perfeitas para ordenar registros.

Quando você cria um índice, o Visual FoxPro utiliza a definição atual de SET COLLATE. Portanto, se você desejar criar dois índices com duas seqüências de ordenação, poderá utilizar uma seqüência de comandos como a seguinte:

```
SET COLLATE TO "MACHINE"
INDEX ON lastname TAG _lastname    && associa/procura índice
SET COLLATE TO "GENERAL"
INDEX ON lastname TAG lastname  && classifica índice
```

Quando você quiser procurar, selecionar ou associar no campo `lastname`, emita o comando SET COLLATE TO “MACHINE” antes de executar a operação. A tecnologia Rushmore utilizará então o índice criado na seqüência de ordenação da máquina e a operação de procura será bastante rápida.

## Utilizando SQL SELECT com várias seqüências de ordenação

Quando você emite um comando SELECT - SQL, o Visual FoxPro utiliza a seqüência de ordenação atual para procura e para cláusulas ORDER BY e GROUP BY. Se você desejar procurar e classificar, utilizando seqüências de ordenação diferentes, poderá dividir seus comandos SQL em duas etapas conforme a seguir:

```
* Seleciona registros utilizando uma seqüência de ordenação
SET COLLATE TO "MACHINE"
SELECT * FROM table INTO CURSOR temp1 ;
    WHERE lname = "Müller"
* Ordena registros utilizando uma seqüência de ordenação diferente
SET COLLATE TO "GENERAL"
SELECT * FROM temp1 INTO TABLE output ORDER BY lastname
```