O Visual FoxPro fornece as ferramentas para você criar *aplicativos cliente/servidor* eficazes. Um aplicativo cliente/servidor do Visual FoxPro combina a força, velocidade, interface gráfica com o usuário e o sofisticado sistema de consulta, relatórios e processamento típicos do Visual FoxPro, incluindo o acesso multiusuário direto, armazenamento de grande quantidade de dados, segurança interna, processamento robusto de transações, criação de resumos e sintaxe nativa do servidor característicos de uma fonte de dados ou servidor ODBC. A sinergia dos pontos fortes do Visual FoxPro e do servidor fornece aos usuários uma solução cliente/servidor eficiente.

A etapa mais importante na construção de aplicativos cliente/servidor eficazes é o desenvolvimento de um bom projeto. Este capítulo baseia-se nas informações de desenvolvimento de aplicativos multiusuário fornecidas nos capítulos anteriores. A partir dessa base, foi definida uma metodologia para o desenvolvimento de aplicativos cliente/servidor.

Para obter informações sobre como criar ou migrar um protótipo local, consulte o capítulo 20, Migrando bancos de dados do Visual FoxPro. Para obter informações sobre a utilização da tecnologia de passagem SQL, consulte o capítulo 21, Implementando um aplicativo cliente/servidor. ". Para acelerar a recuperação e o processamento de dados, consulte o capítulo 22, Otimizando o desempenho do aplicativo cliente/servidor.

Este capítulo aborda os seguintes tópicos:

- Objetivos da criação do cliente/servidor
- Criando para alto desempenho
- Desenvolvendo aplicativos rapidamente
- Criando com precisão e integridade de dados

Objetivos na criação do cliente/servidor

Quando você cria um aplicativo cliente/servidor, combina várias exigências diferentes. Deseja criar o aplicativo mais rápido e produtivo para os seus usuários. Também deseja garantir a integridade dos dados do aplicativo, aproveitar ao máximo o investimento feito no hardware existente e criar uma capacidade de escala para o futuro. Além disso, como um desenvolvedor do Visual FoxPro, você certamente deseja tornar o processo de desenvolvimento o mais padronizado e eficiente possível.

A melhor maneira de atender a essas exigências é ter esses objetivos em mente ao criar seus aplicativos. Vamos começar indicando as técnicas que fornecem o melhor desempenho do aplicativo cliente/servidor.

Criando para alto desempenho

A criação de um aplicativo cliente/servidor rápido e de alto desempenho com o Visual FoxPro envolve o aproveitamento de sua enorme velocidade de operação. Isso se consegue com novas técnicas, como a utilização da técnica de acesso a dados baseados em conjuntos em vez da navegação local tradicional, criação de consultas parametrizadas para descarregar apenas os dados necessários, localização de tabelas na plataforma adequada e nivelamento entre o Visual FoxPro e procedimentos armazenados remotamente.

Antes de poder aproveitar as novas técnicas, você deve analisar os sistemas que serão utilizados. Ao criar um aplicativo local ou de servidor de arquivos, você determina as consultas, formulários, menus e relatórios que o aplicativo utilizará ou criará. Ao criar um aplicativo cliente/servidor, você executa todas as análises normais de sistema e uma análise extra relacionada especificamente com aplicativos cliente/servidor. Precisa pensar onde serão localizados os dados utilizados para consultas, formulários, menus e relatórios e como você acessará essas informações. Por exemplo, você pode se fazer perguntas do tipo:

- Que tabelas serão armazenadas no servidor remoto quando o aplicativo estiver implementado?
- Que tabelas poderiam ser melhor armazenadas como tabelas de pesquisa local?
- Que modos de visualização você precisa para acessar dados remotos?

- Que regras de operação são impostas pelo servidor e como o seu aplicativo interage com elas?

Uma vez determinados os componentes básicos do seu aplicativo cliente/servidor, você pode começar a elaborar como ele acessará e atualizará os dados.

Descarregando só os dados necessários

Um dos fatores mais importantes na criação de um aplicativo cliente/servidor rápido e eficiente é a minimização da quantidade de dados transferidos do servidor. Como os aplicativos cliente/servidor podem acessar potencialmente grandes quantidades de dados em um servidor remoto, a utilização de técnicas tradicionais de navegação local pode resultar em um aplicativo cliente/servidor lento. Para acelerar o desempenho, utilize as técnicas de acesso a dados baseados em conjuntos para filtrar a quantidade de dados transferidos.

Acessando dados baseados em conjuntos com eficiência

Os dados remotos são *baseados em conjuntos:* para acessá-los, você seleciona um *conjunto* em uma área de armazenamento de dados extensos, utilizando instruções SELECT - SQL. A principal diferença entre a criação de um aplicativo local tradicional e a criação de um aplicativo cliente/servidor é o contraste entre as técnicas de navegação tradicionais do Visual FoxPro e as técnicas de acesso a dados baseados em conjuntos do servidor.

Utilizando as técnicas de navegação tradicionais

Na programação tradicional de banco de dados local, você pode acessar quantidades isoladas e geralmente grandes de dados com o comando GOTO BOTTOM, que pode ser utilizado para consultas. Você pode navegar pelos dados emitindo o comando SET RELATION para criar um relacionamento temporário entre as duas tabelas, emitindo em seguida um comando SKIP para deslocar-se pelos registros relacionados.

Esse método de navegação pelos registros pode ser utilizado com dados remotos, mas pode ser ineficiente com grandes áreas de armazenamento de dados remotos. Por exemplo, se você criar uma visualização remota que acesse uma tabela grande em uma fonte de dados remota e, em seguida, emitir o comando GOTO BOTTOM, deverá esperar até que todos os dados visualizados sejam recuperados da fonte de dados, enviados pela rede e transferidos para o Cursor de visualização do seu sistema local.

Utilizando consultas parametrizadas

Uma abordagem mais eficiente para acessar dados remotos consiste em descarregar apenas os dados necessários e depois fazer uma nova consulta para obter registros específicos adicionais ou novos. Utilize uma instrução SELECT com base em parâmetros para transferir um conjunto pequeno de dados; em seguida, para acessar novos registros, utilize o comando REQUERY( ) para solicitar um novo conjunto de dados.

Você não emite o comando GOTO BOTTOM para os dados do servidor remoto porque poderia:

- Sobrecarregar desnecessariamente os recursos da rede ao transferir grandes quantidades de dados.
- Diminuir o desempenho do aplicativo, manipulando dados desnecessários.
- Reduzir potencialmente a precisão dos dados no Cursor local, porque as alterações nos dados remotos só serão refletidas no Cursor local quando você fizer uma nova consulta.

Por exemplo, se deseja criar um aplicativo cliente/servidor que acesse os pedidos de um cliente específico, crie uma visualização remota que acesse a tabela Customer. Você cria outra visualização remota que acesse a tabela Orders, utilizando os parâmetros baseados no campo `cust_id`. Em seguida, utilize o registro do cliente atual como o parâmetro para a visualização da tabela Orders.

Você pode utilizar o parâmetro para definir a quantidade correta de dados para o conjunto

transferido. Se solicitar poucos dados, o desempenho poderá ser afetado porque terá que consultar o servidor remoto com mais freqüência. Se solicitar muitos dados, poderá perder tempo transferindo dados que não serão utilizados.

Escolhendo o melhor projeto cliente/servidor

Os exemplos a seguir descrevem como obter os benefícios da tecnologia cliente/servidor e evitar as armadilhas das técnicas precárias de programação. O primeiro método utiliza práticas de programação tradicionais para chamar todos os dados de uma fonte de dados remota para Cursores locais, que são então relacionados com o comando SET RELATION. O segundo, terceiro e quarto métodos adotam técnicas cada vez mais aprimoradas de captura de dados, limitando com eficiência a quantidade de dados transferidos com uma metodologia 'just-in-time' que fornece os dados mais atualizados e um melhor tempo de resposta na rede.

Utilizando uma estratégia cliente/servidor não otimizada

Um aplicativo cliente/servidor simples, não otimizado, utiliza técnicas de navegação de dados locais com dados remotos. Por exemplo, se você tiver 10 milhões de registros de clientes e 100 milhões de registros de pedidos em uma fonte de dados remota, poderá criar um aplicativo ineficiente que transfira todos os registros de clientes e pedidos para Cursores locais. Você pode então indexar 100 milhões de registros de pedidos, criar uma relação temporária entre as tabelas Customer e Order nos seus Cursores locais e utilizar o comando SKIP para navegar pelos registros.

Este método não é otimizado para desempenho, mas poderá ser útil se o lado "um" for local e o lado "n" for remoto.

Filtrando o lado "n"

Um aplicativo cliente/servidor ligeiramente melhorado limita o lado "n" do relacionamento, mas chama todo o lado "um" de modo que você possa ignorar os registros. Neste cenário, você cria uma visualização remota do lado "n" do relacionamento, a tabela Orders, parametrizada na identificação do cliente. Você então transfere toda a tabela Customer.

Criar uma visualização parametrizada na tabela Orders é um avanço em relação à transferência de todos os pedidos, mas você continua baixando informações desnecessárias ao transferir toda a tabela Customer. A tabela Customer também vai ficando desatualizada à medida que os outros usuários alteram o sistema. Este método pode ser vantajoso se o lado "um" do relacionamento contiver um conjunto de dados pequeno.

Filtrando o lado "um"

Uma técnica melhor de programação cliente/servidor cria visualizações remotas para todos os dados remotos. Você limita o número de registros de Customer transferidos para a visualização remota da tabela Customer, utilizando a instrução SELECT na visualização para selecionar apenas os clientes para uma região. Em seguida, você cria uma visualização remota do lado "n" do relacionamento, a tabela Orders, parametrizada na identificação do cliente.

Este cenário baixa um conjunto menor de registros. Utilize o comando SKIP para passar para o lado "um" do relacionamento (a visualização de Customer). A função REQUERY( ) é utilizada para acessar novos dados no lado "n" (Orders).

Nesse exemplo, você limita, ou filtra, o lado "um" e o lado "n" do relacionamento e ainda pode utilizar o comando SKIP para navegar pelos dados filtrados. Esse método pode ser recomendado se o lado "um" do relacionamento, mesmo depois de filtrado, ainda for suficiente para fornecer informações para um conjunto de consultas sucessivas antes de consultar novamente o servidor remoto.

Utilizando a chave primária para acessar o relacionamento um-para-n

O paradigma de programação cliente/servidor mais eficiente abre mão do luxo de utilizar o comando

SKIP e cria um formulário que solicita informações ou a seleção da identificação do cliente, que é então utilizada como um parâmetro para a visualização remota da tabela Customer. O parâmetro também é utilizado como um parâmetro para uma visualização remota da tabela Orders.

Por exemplo, você pode criar um formulário um-para-n, no qual as informações do cliente formam o lado “um” e um controle Grid exibe o lado “n” do relacionamento. O controle Grid pode ser ligado à identificação do cliente escolhida no lado “um” do formulário. Em seguida, você pode definir a propriedade CURSORSETPROP( ) como 1 e utilizar o código a seguir para preencher o lado “um” do formulário:

```
SELECT * FROM customer WHERE customer.cust_id = ?cCust_id
```

Quando o usuário deseja visualizar um registro de um outro cliente, fornece ou seleciona uma nova identificação de cliente. O formulário consulta novamente a fonte de dados para os pedidos de nova identificação de cliente e atualiza o controle Grid com os novos dados de pedido.

Utilizando essas técnicas, seu aplicativo transfere apenas os dados necessários, no momento necessário. Você acelera a resposta pela rede, limitando a quantidade de dados transferidos e fornece informações atualizadas ao usuário, consultando novamente a fonte de dados antes de exibir as informações solicitadas.

Este método é recomendado quando você deseja acessar o relacionamento um-para-n aleatoriamente utilizando qualquer valor de chave primária. Quando abrir o formulário, convém transferir as chaves primárias para um controle, como uma lista suspensa, e depois fornecer um controle que o usuário possa selecionar para atualizar a lista de valores de chave primária necessários.

Utilizando o ambiente de dados em aplicativos cliente/servidor

Quando você utilizar dados remotos em um formulário, inclua as visualizações do ambiente de dados do formulário. Você pode definir a propriedade AutoOpenTables para o ambiente de dados como falsa (.F.), de modo que possa especificar quando o aplicativo atualizará as visualizações com os dados remotos. Defina a propriedade ControlSource para as caixas de texto ou outros controles ligados a dados depois de chamar o método OpenTables do ambiente de dados, geralmente no código associado com o evento Init do formulário. Para obter maiores informações sobre como definir propriedades do formulário, consulte o capítulo 9, Criando formulários.

Localizando dados na plataforma ideal

Você obtém desempenho máximo quando armazena dados e outros atributos do seu banco de dados na plataforma ideal. A melhor plataforma para determinado elemento depende de como ele é acessado e atualizado. Por exemplo, convém armazenar uma cópia local de uma tabela de servidor (uma tabela tipo diretório de códigos postais, por exemplo), que é utilizada como uma tabela de consulta, e atualizar a cópia local apenas quando a tabela especializada for alterada.

A tabela a seguir lista alguns elementos comuns de aplicativos e exemplos de onde encontrá-los para obter melhor desempenho.

Localizando Elementos pela Plataforma

Elemento	Localização	Tipo	Observações
Tabelas	Local	Cópias locais das tabelas de consulta do servidor; tabelas pequenas, pouco alteradas	Utilize uma marca de data e hora, se suportado pelo seu servidor remoto, para comparar e opcionalmente atualizar a tabela local para fazer coincidir qualquer alteração feita na tabela fonte

			especializada.
	Remota	Tabelas grandes ou muito alteradas	
Regras	Local	Regras em visualizações remotas	Você pode utilizar DBSETPROP( ) para armazenar regras em nível de campo ou registro em uma visualização remota. O seu aplicativo pode utilizar essas regras locais para verificar a validade dos dados antes de enviá-los para as tabelas especializadas como atualizações de tabelas remotas.
	Remota	Regras em nível de linha e coluna em tabelas de base remota	
Procedimentos armazenados	Local	Procedimentos armazenados do Visual FoxPro	
	Remoto	Procedimentos armazenados do servidor especializado	Utilize a função de passagem SQLEXEC( ) de SQL para chamar os procedimentos armazenados do servidor.
Transações	Local	Transações do Visual FoxPro	
	Remota	Transações do servidor	
Disparadores	Visualizações locais	Nenhum disparador nas visualizações	
	Remotas	Disparadores do servidor	

Para reduzir o tráfego na rede durante as consultas, você pode optar por armazenar não apenas as tabelas de consulta com alterações freqüentes, mas também aquelas com poucas alterações feitas localmente. Por exemplo, você pode transferir a lista de clientes da sua empresa e atualizá-la apenas quando as informações do cliente forem alteradas.

Para conseguir isso, você pode programar o aplicativo para comparar as marcas de data e hora na

cópia local da tabela com as marcas de data e hora dos dados especializados (se o servidor remoto aceitar marcas de data e hora) e atualizar a cópia local se a tabela do servidor tiver sido alterada. Você também pode adicionar um botão de comando ao formulário, forçando uma transferência imediata da tabela e permitindo aos usuários atualizar a sua cópia da tabela local sob demanda.

Escolhendo os métodos corretos

Você pode utilizar as visualizações remotas, a passagem SQL, ou ambas para criar o seu aplicativo cliente/servidor. Pode combinar os dois métodos para obter bons resultados: utilize as visualizações para a maioria das suas exigências de gerenciamento de dados e a passagem SQL para aumentar a força do aplicativo.

Utilizando visualizações

Você pode utilizar as visualizações como o principal método para desenvolver um aplicativo cliente/servidor potente. As visualizações remotas são uma tecnologia poderosa, criada para permitir apenas a seleção dos dados necessários de um servidor remoto para um Cursor local do Visual FoxPro, que pode ser utilizado para visualizar e atualizar dados remotos. Uma visualização é basicamente o conjunto de resultados de uma instrução SQL SELECT.

As visualizações são permanentes: a definição da visualização está armazenada em um banco de dados. As definições das visualizações possuem propriedades que podem ser definidas, depois personalizadas, para o cursor da visualização ativa. As visualizações são a melhor ferramenta para a definição de dados de um conjunto de resultados atualizáveis.

Você pode utilizar as visualizações locais para criar um protótipo local, depois utilizar o **Assistente de upsizing** para transformar as visualizações locais em visualizações remotas. Para obter informações sobre como utilizar o **Assistente de upsizing**, consulte o capítulo 20, Migrando bancos de dados do Visual FoxPro.

Se os usuários de seus aplicativos desejarem utilizar dados para trabalho em trânsito, você pode empregar visualizações off-line. Visualizações off-line criam dados portáteis, permitindo aos usuários de laptops ou outros computadores portáteis trabalharem com cópias armazenadas de dados fonte, que podem ser atualizados em trânsito. Quando o usuário se reconectar ao servidor, poderá mesclar as mudanças feitas off-line com as tabelas fonte.

Você poderá também utilizar a tecnologia de visualização off-line para permitir aos usuários locais trabalharem “off-line”, mesclando seus dados mais tarde. Para mais informações sobre o trabalho off-line, consulte o capítulo 8, Criando visualizações de várias tabelas.

Utilizando a passagem SQL

A tecnologia de passagem SQL fornece acesso direto a um servidor remoto com as funções de passagem SQL do Visual FoxPro. Essas funções permitem o acesso e o controle adicional do servidor além da capacidade de visualização; por exemplo, você pode efetuar a definição de dados no servidor remoto, definir propriedades do servidor e acessar procedimentos nele armazenados.

A passagem SQL é a melhor ferramenta para criar conjuntos de resultados somente para leitura e para utilizar qualquer outra sintaxe nativa SQL. Em oposição a uma visualização, que é o conjunto de resultados de uma instrução SQL SELECT, a passagem SQL permite que você envie qualquer item desejado ao servidor, utilizando a função SQLEXEC( ). A tabela a seguir lista as funções de passagem SQL do Visual FoxPro.

Funções de passagem SQL

SQLCANCEL( )	SQLCOLUMNS( )	SQLCOMMIT( )
SQLCONNECT( )	SQLDISCONNECT( )	SQLEXEC( )
SQLGETPROP( )	SQLMORERESULTS( )	SQLPREPARE( )
SQLROLLBACK( )	SQLSETPROP( )	SQLSTRINGCONNECT( )

SQLTABLES( )

Você mesmo pode criar Cursores utilizando a tecnologia de passagem SQL. Embora a passagem SQL forneça acesso mais direto ao servidor, esse acesso é menos permanente do que as visualizações. Diferentemente das visualizações, cujas definições são armazenadas de forma permanente em um banco de dados, os Cursores que utilizam a passagem SQL existem apenas para a sessão ativa. Para obter maiores informações sobre como utilizar a tecnologia de passagem SQL, consulte o capítulo 21, Implementando um aplicativo cliente/servidor.

Combinando visualizações e a passagem SQL

O paradigma mais poderoso para a criação de um aplicativo cliente/servidor do Visual FoxPro combina as tecnologias de visualização e passagem SQL. Como essas visualizações são fáceis de criar e fornecem as capacidades de utilização de buffer e atualização, elas são utilizadas para a maior parte das tarefas de gerenciamento de dados. Você utiliza a passagem SQL para realizar tarefas específicas no servidor remoto, como a definição de dados e a criação e execução de procedimentos armazenados pelo servidor.

Desenvolvendo aplicativos rapidamente

Independente do método de programação escolhido, você precisa de uma boa estratégia para tornar o desenvolvimento de aplicativos cliente/servidor rápido e eficiente. Como o Visual FoxPro facilita a criação rápida de protótipos e aplicativos, você pode optar por criar e construir um protótipo local do seu aplicativo, depois migrar e implementá-lo gradualmente em uma fonte de dados remota. Se você tiver acesso a uma fonte de dados remota durante o processo de desenvolvimento, poderá optar por escolher o protótipo para a fonte de dados remotos, utilizando as visualizações remotas.

Criando um protótipo com visualizações

A primeira etapa no desenvolvimento de um aplicativo cliente/servidor do Visual FoxPro pode ser criar um protótipo. Ao criar um protótipo para o seu aplicativo, talvez módulo a módulo, você descobre as alterações e melhorias que devem ser feitas na estrutura do aplicativo no início do processo de desenvolvimento. Você pode ajustar o seu projeto de forma eficiente para áreas pequenas de armazenamento de dados de exemplo antes de adicionar a complexidade inerente a trabalhos com grandes conjuntos de dados remotos e heterogêneos. A criação de um protótipo está descrita no capítulo 20, Migrando bancos de dados do Visual FoxPro.

Criando um protótipo local com visualizações locais

Um protótipo local para um aplicativo cliente/servidor é um aplicativo do Visual FoxPro que utiliza visualizações locais para acessar tabelas locais. Você utiliza as visualizações no protótipo do cliente/servidor porque o aplicativo final cliente/servidor utiliza visualizações remotas para acessar dados remotos. Criando um protótipo do aplicativo com visualizações locais, você está mais próximo do aplicativo final.

A criação de um protótipo local é extremamente prática quando você não tem acesso constante a uma fonte de dados remotos durante o desenvolvimento ou quando não quer utilizar os dados remotos para criar um protótipo para o aplicativo. As visualizações locais acessam tabelas locais do Visual FoxPro, em vez de tabelas de uma fonte de dados remota. Os dados locais são criados para imitar a estrutura dos dados no servidor. A utilização de dados locais para representar dados remotos é um método de desenvolver e testar rapidamente a estrutura básica do aplicativo. Você também pode acelerar o desenvolvimento, limitando a quantidade de dados selecionados nas visualizações. Para obter maiores informações sobre como criar visualizações locais e remotas, consulte
o capítulo 8, Criando visualizações.

Planejando a migração

Migração é o processo que cria um banco de dados no servidor remoto com a mesma estrutura de tabela, dados e potencialmente muitos outros atributos do banco de dados original do Visual FoxPro. Com a migração, você pega um aplicativo do Visual FoxPro existente e faz a migração para um aplicativo cliente/servidor. Para obter maiores informações sobre com migrar, consulte o capítulo 20, Migrando bancos de dados do Visual FoxPro.

Quando você cria um aplicativo que será migrado, você escolhe a estrutura da arquitetura do aplicativo e o modelo de programação com base na obtenção de desempenho máximo para uma fonte de dados remotos. Essas opções foram descritas anteriormente neste capítulo, na seção Criando para alto desempenho.

Criando protótipos com visualizações remotas

Se você tiver acesso a uma fonte de dados remota e quiser utilizar os dados remotos diretamente ao desenvolver o aplicativo cliente/servidor, poderá criar o seu protótipo utilizando visualizações remotas. Ao criar protótipos utilizando visualizações remotas, você pula a etapa de migração, porque os dados já estão localizados em um servidor remoto e você já possui visualizações remotas para acessar esses dados.

Implementando o aplicativo cliente/servidor

Você pode simplificar o processo de teste e depuração do aplicativo, implementando o seu protótipo de aplicativo por etapas. Ao implementá-lo por etapas, você adiciona melhorias para o ambiente multiusuários, move os dados para a fonte de dados remota e testa e depura o aplicativo, módulo a módulo, de forma sistemática.

Ao implementar o aplicativo, você pode utilizar a sintaxe do servidor local e acessar as funções específicas do servidor, como a tecnologia de procedimentos armazenados no servidor e a passagem SQL. Para obter informações sobre a passagem SQL , consulte o capítulo 21, Implementando um aplicativo cliente/servidor.

Otimizando o aplicativo

Depois que o aplicativo tiver sido totalmente implementado para os dados remotos e você tiver concluído a fase de testes e depuração, pode ajustar a velocidade e o desempenho do aplicativo inteiro. Para obter maiores informações sobre as melhorias que podem ser introduzidas em um aplicativo implementado, consulte o capítulo 22, Otimizando o desempenho do aplicativo cliente/servidor.

Criando com precisão e integridade de dados

Você pode combinar os recursos das regras de validação de dados do Visual FoxPro e procedimentos armazenados com as regras de validação de dados fonte e procedimentos armazenados para criar aplicativos cliente/servidor que protegem a integridade dos dados.

Mantendo a integridade dos dados

Você pode criar versões locais das regras de validação do servidor remoto para fornecer mensagens amigáveis ao usuário, por exemplo, sobre atualizações que não serão permitidas quando enviadas ao servidor remoto, porque os dados fornecidos violaram a integridade relacional do servidor ou regra de validação de dados.

Utilizando as regras do Visual Foxpro em uma visualização remota ou off-line

Você pode criar regras em nível de campo e registro em visualizações off-line para validar os dados inseridos localmente, antes que os dados sejam enviados para a fonte de dados remota. Como o objetivo dessas regras de visualização é evitar o envio de dados para a fonte de dados que será rejeitada pelas regras de integridade de dados do servidor, você deve replicar as regras de fonte de

dados nas regras criadas para a visualização remota. Utilize a função DBSETPROP( ) para criar regras para as visualizações.

Dica Você pode criar uma regra de validação local em uma visualização remota, a qual chama um procedimento armazenado de servidor remoto e envia o valor que você deseja validar para o servidor como um parâmetro. No entanto, utilizar um procedimento armazenado remoto aumenta o tempo de processamento durante a entrada de dados.

Utilizando regras do servidor

Você pode optar por seguir as regras estabelecidas no servidor para a validação dos dados. Se ocorrer um erro, a rotina de manipulação de dados poderá chamar a função AERROR( ) para obter informações, incluindo o número da mensagem de erro, o texto da mensagem de erro remota e o identificador de conexão associado com o erro.

Utilizando disparadores do servidor

Embora seja possível criar disparadores do Visual FoxPro em tabelas locais, não é possível criá-los em visualizações. No entanto, você pode utilizar os disparadores na fonte de dados remota. Os disparadores do servidor podem ser utilizados para processar atualizações de dados secundários, como por exemplo, atualizações ou exclusões em cascata. Utilizar os disparadores do servidor para atualizações secundárias é mais eficiente do que enviar vários comandos ao servidor remoto do aplicativo Visual FoxPro.

Protegendo contra a perda de dados

O Visual FoxPro e a maioria das fontes de dados remotas fornecem capacidades de registro de transações para proteção contra a perda de dados. Para obter maiores informações sobre como utilizar transações do Visual FoxPro, consulte o capítulo 17, Programando para acesso compartilhado.

Você pode utilizar as transações do Visual FoxPro para os protótipos locais e para o processamento local de dados. Utilize as transações do servidor para atualizações, inserções e exclusões de dados remotos. Para obter maiores informações sobre como utilizar transações remotas, consulte o capítulo 22, Otimizando o desempenho do aplicativo cliente/servidor.