

Se você criou e migrou um protótipo local de trabalho ou desenvolveu o aplicativo utilizando dados remotos através de visualizações remotas, obteve acesso a grandes áreas de armazenamento de dados geralmente disponíveis em um banco de dados do servidor. Você pode também aproveitar a segurança e as capacidades de processamento de transações do servidor remoto. Ao mesmo tempo em que as visualizações remotas lidam com as principais tarefas de gerenciamento de dados, você pode melhorar o aplicativo utilizando a tecnologia de passagem SQL (SPT) para criar objetos no servidor, executar procedimentos armazenados no servidor e executar comandos utilizando sintaxe nativa do servidor.

Este capítulo aborda as técnicas para implementar tecnologia cliente/servidor em um aplicativo de trabalho que utiliza visualizações remotas. Se quiser saber mais sobre como projetar e criar um aplicativo cliente/servidor, consulte o capítulo 19, [Criando aplicativos cliente/servidor](#), e o capítulo 20, [Migrando bancos de dados do Visual FoxPro](#). Para obter maiores informações sobre como criar visualizações remotas, consulte o capítulo 8, [Criando visualizações](#).

Este capítulo abrange os tópicos a seguir:

- [Utilizando a tecnologia de passagem SQL](#)
- [Trabalhando com dados remotos utilizando a passagem SQL](#)
- [Manipulando erros da passagem SQL](#)

Utilizando a tecnologia de passagem SQL

O aplicativo cliente/servidor pode acessar dados do servidor utilizando:

- Visualizações remotas
- Passagem SQL

As visualizações remotas fornecem o método mais comum e fácil para acessar e atualizar dados remotos. Os **Assistentes de upsizing** podem criar automaticamente visualizações remotas para o banco de dados como parte da migração ou você pode utilizar o Visual FoxPro para criar visualizações remotas depois da migração. Para obter maiores informações sobre visualizações remotas, consulte o capítulo 8, [Criando visualizações](#).

A tecnologia de passagem SQL permite que você envie instruções SQL diretamente a um servidor. Já que elas podem ser executadas no servidor especializado, são formas eficazes de melhorar o desempenho dos aplicativos cliente/servidor. A tabela a seguir compara as visualizações remotas com a passagem SQL:

Comparação entre as tecnologias de visualização remota e passagem SQL

Visualização remota	Passagem SQL
Baseada em uma instrução SQL SELECT.	Baseada em qualquer instrução SQL de servidor nativa, ativando instruções de definição de dados ou a execução de procedimentos armazenados no servidor.
Pode ser utilizada como fonte de dados para os controles no momento da criação.	Não pode ser utilizada como uma fonte de dados para controles.
Não fornece capacidade para executar comandos DLL em fontes de dados.	Fornecer métodos de utilização de comandos DLL em fontes de dados.
Carrega um conjunto de resultados.	Carrega um ou vários conjuntos de resultados.
Fornecer gerenciamento interno de conexões.	Requer gerenciamento de conexão explícito.

Fornece informações de atualização padrão interna para atualizações, inserções e exclusões.

Fornece execução SQL implícita e carga de dados.

Não fornece manipulação de transações.

Armazena propriedades de forma permanente no banco de dados.

Utiliza carga progressiva assíncrona durante a execução de SQL.

Não fornece informações de atualização padrão.

Fornece execução de SQL e controle de carga de resultados explícitas.

Fornece manipulação de transações explícita.

Fornece propriedades temporárias para o cursor de passagem SQL, com base nas propriedades de sessão.

Fornece suporte total à carga assíncrona através da linguagem de programação.

A tecnologia de passagem SQL oferece as vantagens a seguir em relação a visualizações remotas:

- Você pode utilizar a funcionalidade específica do servidor, como procedimentos armazenados e funções intrínsecas baseadas no servidor.
- Você pode utilizar extensões à SQL suportadas pelo servidor, assim como comandos de definição de dados, de administração do servidor e de segurança.
- Você tem mais controle sobre as instruções de Atualização, Exclusão e Inserção da passagem SQL.
- Você tem mais controle sobre as transações remotas.

Dica O Visual FoxPro pode manipular consultas de passagem SQL que retornam mais do que um único conjunto de resultados. Para obter maiores informações, consulte [Processando vários conjuntos de resultados](#) mais adiante neste capítulo.

As consultas de passagem SQL também têm desvantagens:

- Como padrão, uma consulta de passagem SQL sempre retorna um instantâneo não atualizável de dados remotos, que é armazenado em um cursor de visualização ativo. Você pode tornar o cursor atualizável, definindo propriedades com a função `CURSORSETPROP()`. Uma visualização remota atualizável, por outro lado, geralmente não requer a definição de propriedades antes da atualização de dados remotos, pois as definições de propriedade são armazenadas no banco de dados com a definição da visualização.
- Você deve digitar comandos SQL diretamente na janela **Comando** ou em um programa, em vez de utilizar o **Criador de visualizações** gráfico.
- Você cria e gerencia a conexão à fonte de dados.

Se você utilizar visualizações remotas ou passagem SQL, poderá consultar e atualizar dados remotos. Em muitos aplicativos, você usará ambas.

Utilizando funções da passagem SQL

Para utilizar a passagem SQL a fim de conectar a uma fonte de dados ODBC, você primeiro chama a função `SQLCONNECT()` do Visual FoxPro para criar uma conexão. Em seguida, utiliza as funções de passagem SQL do Visual FoxPro para enviar comandos à fonte de dados remota para execução.

► Para utilizar as funções da passagem SQL do Visual FoxPro

- 1 Confirme a capacidade do sistema de conectar o seu computador à fonte de dados. Lance mão de um utilitário como ODBC Test para ODBC.
- 2 Estabeleça uma conexão com sua fonte de dados utilizando a função `SQLCONNECT()` ou com `SQLSTRINGCONNECT()`.

Por exemplo, se estiver conectando o Visual FoxPro à fonte de dados de SQL Server `sqlremote`,

poderá efetuar o logon como administrador do sistema (identificação de usuário as) utilizando a senha secret com o comando a seguir:

```
nConnectionHandle = SQLCONNECT('sqlremote','sa','secret')
```

Observação Você pode também utilizar a função SQLCONNECT() para ligar-se a uma conexão definida.

- 3 Utilize as funções da passagem SQL do Visual FoxPro para recuperar dados para os cursores do Visual FoxPro e processar os dados recuperados com os comandos e funções padrão do Visual FoxPro.

Por exemplo, você pode consultar a tabela authors e pesquisar o cursor resultante utilizando este comando:

```
? SQLEXP(nConnectionHandle,"select * from authors","mycursorname")  
BROWSE
```

- 4 Desconecte-se da fonte de dados com a função SQLDISCONNECT().

Funções da passagem SQL do Visual FoxPro

A tabela a seguir lista as funções SQL do Visual FoxPro que suportam o trabalho com fontes de dados remotas, agrupadas de acordo com as tarefas.

Funções da passagem SQL do Visual FoxPro

Tarefa	Função	Objetivo
Gerenciamento de conexões	SQLCONNECT()	Conecta a uma fonte de dados para operações de passagem SQL..
	SQLSTRINGCONNECT()	Conecta a uma fonte de dados utilizando a sintaxe da sequência de conexões ODBC.
	SQLDISCONNECT()	Interrompe uma conexão com uma fonte de dados ODBC, tornando o identificador de conexão especificado obsoleto.
Execução e controle da instrução SQL	SQLCANCEL()	Cancela uma consulta SQL em execução assíncrona em uma conexão ativa.
	SQLEXP()	Executa uma consulta de passagem SQL em uma conexão ativa; retorna o número de conjuntos de resultados gerado ou 0 se SQLEXP() ainda estiver executando (processamento assíncrono).
	SQLMORERESULTS()	Coloca outro conjunto de resultados em um cursor. Retorna 0 se a instrução que cria o conjunto de resultados ainda estiver sendo executada.
	SQLPREPARE()	Pré-compila a instrução SQL na fonte de dados e

		acopla os parâmetros do Visual FoxPro, ou seja, salva as expressões de parâmetro reais para todos os parâmetros na instrução SQL.
	<code>SQLCOMMIT()</code>	Pede a gravação de uma transação.
	<code>SQLROLLBACK()</code>	Desfaz uma transação.
Informações da fonte de dados	<code>SQLCOLUMNS()</code>	Armazena uma lista com os nomes de colunas e informações sobre cada uma delas em um cursor. Retorna 1 se a função tiver êxito ou 0 se a função ainda estiver sendo executada.
	<code>SQLTABLES()</code>	Armazena os nomes de tabelas na fonte em um cursor. Retorna 1 se a função tiver êxito ou 0 se a função ainda estiver sendo executada.
Controle diverso	<code>SQLGETPROP()</code>	Obtém uma propriedade de conexão da conexão ativa.
	<code>SQLSETPROP()</code>	Define uma propriedade de uma conexão ativa.

As instruções `SQLEXP()`, `SQLMORERESULTS()`, `SQLTABLES()` e `SQLCOLUMNS()` podem ser canceladas em modo síncrono, pressionando ESC se SET ESCAPE estiver ativado (ON). Você pode cancelar estas instruções a qualquer momento em modo assíncrono, utilizando `SQLCANCEL()`.

Todas as outras instruções da passagem SQL funcionam de modo síncrono e não podem ser interrompidas.

Criando conjuntos de resultados

Quando você utiliza as funções da passagem `SQLEXP()` ou `SQLMORERESULTS()` para consultar dados, o Visual FoxPro retorna-os em um ou vários conjuntos de resultados. Estes têm origem nos cursores da fonte de dados do servidor e se tornam cursores no Visual FoxPro. O nome padrão para um conjunto de resultados é `SQLRESULT`.

Acessando procedimentos armazenados do servidor com funções de passagem SQL

Você pode utilizar a tecnologia de passagem SQL para criar e executar procedimentos armazenados em um servidor remoto. Eles podem aumentar muito a eficácia, a eficiência e a flexibilidade do SQL e melhorar sensivelmente o desempenho das instruções SQL e em lote. Muitos servidores fornecem procedimentos armazenados para a definição e a manipulação de objetos do banco de dados do servidor e para realizar a administração de usuário e do sistema do servidor.

Observação Os exemplos neste capítulo utilizam sintaxe de SQL Server da Microsoft, a não ser em casos especificados.

► Para chamar um procedimento armazenado do servidor

- Utilize a função `SQLEXP()` com o nome do procedimento armazenado.

Por exemplo, o código a seguir exibe os resultados da chamada de um procedimento armazenado denominado `sp_who` no SQL Server utilizando uma conexão ativa à fonte de dados `sqlremote`:

```
nConnectionHandle = SQLCONNECT('sqlremote')
? SQLEEXEC(nConnectionHandle, 'use pubs')
? SQLEEXEC(nConnectionHandle, 'sp_who')
BROWSE
```

Para obter maiores informações sobre a criação e a execução de procedimentos armazenados do servidor, consulte a documentação do servidor.

Retornando vários conjuntos de resultados

Se você executar um procedimento armazenado que contenha instruções `SELECT` da sintaxe do servidor nativo, cada conjunto de resultados será retornado para um cursor separado do Visual FoxPro. Você pode utilizar estes cursores para retornar valores ou parâmetros de um procedimento armazenado do servidor para o cliente do Visual FoxPro.

► Para retornar vários conjuntos de resultados

- Utilize a função `SQLEEXEC()` para selecionar vários conjuntos de resultados usando a sintaxe do servidor nativo.

Por exemplo, o código a seguir cria e executa um procedimento armazenado de SQL Server, `my_procedure`, que retorna três cursores do Visual FoxPro: `sqlresult`, `sqlresult1` e `sqlresult2`.

```
=SQLEEXEC(nConnectionHandle, 'create procedure my_procedure as ;
    select * from sales; select * from authors;
    select * from titles')
=SQLEEXEC(nConnectionHandle, 'execute my_procedure')
```

Como o servidor processa conjuntos de resultados e erros

Como o servidor compila cada procedimento armazenado ao ser criado, você receberá qualquer erro de sintaxe do servidor no momento da criação. Quando você executa o procedimento armazenado, o servidor executa as instruções SQL compiladas em sequência (como em um programa do Visual FoxPro) e o Visual FoxPro carrega cada conjunto de resultados de cada instrução SQL dentro do procedimento armazenado separadamente, na ordem executada.

Os conjuntos de resultados e erros são retornados na ordem recebida e o processamento é interrompido se um erro for encontrado. Por exemplo, se ocorrer um erro em tempo de execução quando o servidor executa a terceira instrução de um procedimento armazenado com quatro instruções, você receberá os dois primeiros conjuntos de resultados e, em seguida, o erro ocorrido durante o processamento do terceiro. O processamento é interrompido depois que um erro é retornado; o quarto conjunto de resultados não é recuperado. Você pode utilizar a função `AERROR()` para obter informações sobre o erro mais recente.

Observação É possível executar procedimentos armazenados do servidor do Visual FoxPro apenas utilizando as funções de passagem SQL do Visual FoxPro. As visualizações não aceitam procedimentos armazenados de servidor, pois cada visualização contém uma instrução SQL explícita em sua definição SQL.

Passando uma instrução SQL à fonte de dados

A função `SQLEEXEC()` permite que você envie uma instrução SQL à fonte de dados sem interpretação. Basicamente, isto ocorre com qualquer sequência incluída no segundo parâmetro da função `SQLEEXEC()` permitindo que você execute qualquer instrução utilizando a SQL nativa da fonte de dados.

Você pode também utilizar a função `SQLEEXEC()` para criar uma consulta com parâmetros ou passar extensões do ODBC para SQL à fonte de dados.

Criando uma consulta com parâmetros

Assim como você pode criar visualizações com parâmetros, utilizando a linguagem ou o **Criador de visualizações**, pode criar também com parâmetros uma consulta de passagem SQL.

► Para criar uma consulta com parâmetros com a passagem SQL

- Digite um símbolo de ponto de interrogação (?) após um parâmetro do Visual FoxPro e, em seguida, inclua o parâmetro em uma sequência SQL enviada com **SQLEXEC()**.

O parâmetro fornecido é avaliado como uma expressão do Visual FoxPro e o valor enviado como parte da instrução SQL da visualização. Se a avaliação falhar, o Visual FoxPro solicita o valor do parâmetro.

Dica Se o parâmetro for uma expressão, coloque a expressão de parâmetro entre parênteses. Isso garante que toda a expressão será avaliada como parte do parâmetro.

Por exemplo, se você tiver uma tabela `customer` do banco de dados `Testdata` em um servidor remoto, o código abaixo criará uma consulta com parâmetros que limita a visualização aos clientes cujos países correspondem ao valor fornecido para o parâmetro `?cCountry`:

```
? SQLEXEC(1, 'SELECT * FROM customer WHERE customer.country = ?cCountry')
```

Se quiser solicitar que o usuário forneça um valor de parâmetro, coloque a expressão de parâmetro entre aspas. Para obter maiores informações sobre como pedir um valor de parâmetro, consulte o capítulo 8, **Criando visualizações**.

A fonte de dados ODBC não aceita parâmetros nos locais a seguir:

- Em campos **SELECT** ou lista de tabelas.
- Como as duas expressões de um predicado de comparação.
- Como os dois operandos de um operador binário.

Uma fonte de dados ODBC não aceita parâmetros nos seguintes locais na cláusula **WHERE** ou **HAVING** de uma instrução **SELECT**:

- Como o primeiro e o segundo operandos de um predicado **BETWEEN**.
- Como o primeiro e terceiro operandos de um predicado **BETWEEN**.
- Como a expressão e o primeiro valor de um predicado **IN**.
- Como o operando de um operador monádico **+** ou **-**.
- Como o argumento de uma função **SET**.

Utilizando parâmetros de entrada/saída de SQL Server

Você pode utilizar parâmetros de entrada/saída para passar valores entre o Visual FoxPro e o SQL server. Os parâmetros de entrada/saída estão disponíveis somente utilizando passagem SQL; mas não em visualizações.

A tabela a seguir fornece um exemplo utilizando parâmetros de entrada/saída para passar valores do Visual FoxPro para um procedimento armazenado de SQL Server, retornando o resultado a uma variável do Visual FoxPro.

Utilizando parâmetros de entrada/saída com procedimento armazenado de SQL Server

Código	Comentários
<pre>resultCode = SQLExec(connHand, "CREATE PROCEDURE sp_test; @mult1 int, @mult2 int, @result int; OUTPUT AS SELECT @result = @mult1 * @mult2")</pre>	Cria um procedimento armazenado, <code>sp_test</code> , que multiplica duas variáveis (<code>mult1</code> and <code>mult2</code>), em seguida, armazena a quantidade resultante na variável <code>result</code> .

<code>outParam = 0</code>	Cria uma variável do Visual FoxPro para receber o valor do parâmetro de saída quando esta é passada de um SQL Server para o Visual FoxPro.
<code>resultCode = SQLExec(connHand, ; "{CALL sp_test (2, 4, ?@outParam)}")</code>	Executa o procedimento armazenado de SQL Server, passando os valores '2' e '4' para serem multiplicados juntos no procedimento armazenado.
<code>? "outParam =", outParam && o valor é 8</code>	Exibe o valor do parâmetro de saída.

Definindo parâmetros

A sintaxe para os parâmetros de saída é:

?@parâmetro_nome

Ao implementar os parâmetros de entrada/saída, defina as variáveis do Visual FoxPro que deseja incluir no comando de passagem SQL antes de utilizar as variáveis na instrução SQL. Para enviar e receber informações de maneira bem-sucedida com os parâmetros de entrada/saída, deve-se definir:

- Um parâmetro de procedimento armazenado, com um tipo de saída, que retorne o valor.
Por exemplo, se o parâmetro do procedimento armazenado for `@result`, você deverá atribuir um tipo de saída, como `int`, para `@result` e também um valor a `@result`.
- Uma expressão de parâmetro de saída (*@parâmetro_nome*) que avalia para uma variável existente do Visual FoxPro.
Por exemplo, se a expressão de parâmetro de saída for `?@outParam`, o aplicativo deverá ter definido a variável `outParam` do Visual FoxPro.

Observação Se você não utilizar um parâmetro de saída no Visual FoxPro ou em um procedimento armazenado, ou se não definir uma variável do Visual FoxPro para receber o valor de retorno, o valor do parâmetro não será alterado.

Convertendo tipos de dados

O Visual FoxPro converte valores de variáveis retornados utilizando as regras a seguir:

- As variáveis de tipo de dados de ponto flutuante (N, F, B) são convertidas em N.
- O tamanho da exibição é definido como 20.
- A definição decimal é definida como a da sessão atual, afetando apenas o formato de exibição padrão, mas não a precisão decimal.
- As variáveis de data e hora (D, T) são convertidas a variáveis de hora (T).

Você não pode utilizar os tipos de dados Memo, Geral, Figura ou NULO em parâmetros de entrada/saída.

Se o seu aplicativo utiliza campos de cursor como parâmetros, o Visual FoxPro tentará converter o resultado ao tipo de dados original.

Retornando valores de parâmetros

Os parâmetros de entrada/saída estão disponíveis somente após o último conjunto de resultados de uma instrução tiver sido carregado. Isso significa que os valores de entrada/saída são retornados ao Visual FoxPro somente depois que:

- **SQLEXEC()** retornar (1) no modo em lotes
– Ou –
- **SQLMORERESULTS()** retornar (2) no modo não em lotes.

Se a instrução **SQLEXEC()** exigir vários conjuntos de resultados, só será garantido que os parâmetros de saída estarão disponíveis depois que o último conjunto de resultados for carregado da fonte de dados.

Criando associações externas com dados remotos

Você pode utilizar passagens SQL para realizar associações externas em dados remotos usando a sintaxe do servidor nativo, se o servidor aceitar associações externas. Uma associação externa combina informações de uma ou mais tabelas não importa se são localizadas linhas coincidentes.

► Para realizar uma associação externa em um servidor

- Utilize a função **SQLEXEC()** com a sintaxe de associação externa do servidor.

Por exemplo, o código a seguir utiliza a função de passagem SQL do Visual FoxPro **SQLEXEC()** para exibir os resultados de uma associação externa no SQL Server utilizando a conexão definida ativa `sqlremote`:

```
? SQLEXEC(sqlremote, 'select au_fname, au_lname, pub_name ;
                    from authors, publishers ;
                    where authors.city != publishers.city')
```

BROWSE

Para obter maiores informações sobre a sintaxe e os tipos de associação externa, consulte a documentação de seu servidor. Para obter maiores informações sobre a criação de uma conexão definida, consulte “Definindo uma conexão” no capítulo 8, [Criando visualizações](#).

Utilizando as extensões ODBC para SQL

Você pode utilizar **SQLEXEC()** para executar as extensões ODBC para SQL, colocando a instrução SQL no Grupo de acesso SQL padrão ou na sintaxe de escape estendida. Para obter maiores informações sobre as extensões ODBC para SQL, consulte o apêndice Gramática SQL na documentação sobre o ODBC.

Criando associações externas utilizando a cláusula de escape de ODBC

Você pode utilizar a passagem SQL para realizar associações externas em dados remotos através da sintaxe de escape de ODBC, se o servidor aceitar associações externas. Uma associação externa combina informações de uma ou mais tabelas não importa se forem localizadas linhas coincidentes.

A sintaxe para associações externas utilizando a cláusula de escape de ODBC é:

{oj *expressão associação-externa*}

O exemplo a seguir cria um conjunto de resultados dos nomes e departamentos de funcionários que trabalham no projeto 544:

```
SELECT employee.name, dept.deptname;
FROM {oj employee LEFT OUTER JOIN dept;
      ON employee.deptid = dept.deptid};
WHERE employee.projid = 544
```

Para obter maiores informações sobre a sintaxe e os tipos de associações externas, consulte a documentação de seu servidor. Para obter maiores informações sobre a criação de uma conexão definida, consulte “Definindo uma conexão” no capítulo 8, [Criando visualizações](#).

Gerenciando conexões com a passagem SQL

Ao criar uma visualização remota, você escolhe, para ativá-la, um nome de fonte de dados ODBC ou

um nome de conexão que será usado como ferramenta pelo servidor remoto. Para acessar dados remotos diretamente através da passagem SQL, você deve ter o identificador de uma conexão ativa, que é um valor que se refere a um objeto; neste caso, se refere a uma conexão de fonte de dados. Para obter um identificador, você pede uma conexão com a fonte de dados utilizando a função [SQLCONNECT\(\)](#) ou [SQLSTRINGCONNECT\(\)](#). Se a conexão for bem-sucedida, o aplicativo receberá o identificador de conexão para uso nas chamadas subsequentes do Visual FoxPro.

O aplicativo pode pedir várias conexões para uma fonte de dados. Você também pode trabalhar com várias fontes de dados ODBC, solicitando uma conexão com cada fonte de dados que deseja acessar. Se desejar reduzir o número de conexões usadas, poderá configurar as visualizações remotas para compartilhar a mesma conexão. Você se desconecta de uma fonte de dados com a função [SQLDISCONNECT\(\)](#).

Dica O Visual FoxPro depende da definição da fonte de dados ODBC armazenada no arquivo ODBC.INI do Windows ou no registro do Windows NT para conectar-se a uma fonte de dados. Se você alterar o nome ou as informações de login para uma fonte de dados, lembre-se de que estas alterações poderão afetar a conexão do aplicativo que usa a fonte de dados com o servidor remoto desejado.

Controlando as propriedades de ambiente e conexão

O ambiente cliente/servidor é estabelecido toda vez que é aberto o Visual FoxPro. O ambiente existe para esta sessão e desaparece quando você fecha o Visual FoxPro. O ambiente cliente/servidor contém:

- Propriedades globais que agem como protótipos para as novas conexões.
- Valores de erro para erros que ocorrem fora de uma conexão específica.

Você pode utilizar um identificador 0, o identificador de ambiente, para fazer referência às definições de propriedade global. Utilize a função [SQLSETPROP\(\)](#) para controlar as definições de propriedade padrão no ambiente de conexão e as propriedades dentro de conexões individuais. Os métodos utilizados para fornecer valores de [SQLSETPROP\(\)](#) são consistentes para as conexões de ambiente e individuais:

- Propriedades especificadas com um dos dois valores podem utilizar um valor lógico (.F. ou .T.) para *Expressão*.
- Um nome de propriedade pode ser abreviado para a forma truncada menos ambígua. Por exemplo, você pode usar "Asynchronous", "Asynch" ou "A" para especificar a propriedade Asynchronous. Os nomes de propriedade desconsideram maiúsculas e minúsculas.

Quando uma conexão é iniciada, ela herda valores de propriedade de conexão padrão. Você pode utilizar [SQLSETPROP\(\)](#) para alterar estes valores.

Definindo propriedades de conexão

Para visualizar as definições de propriedade atuais para uma conexão, utilize [SQLGETPROP\(\)](#) com o respectivo identificador de conexão. A tabela a seguir lista as propriedades de conexão que podem ser acessadas com [SQLGETPROP\(\)](#).

Propriedades de conexão do Visual FoxPro

Para	Utilize esta propriedade	Objetivo
Exibir as informações utilizadas para criar a conexão ativa	ConnectionString	A seqüência de conexão de login.
	DataSource	O nome da fonte de dados como definido pelo ODBC.

	Password	A senha de conexão.
	UserID	A identificação de usuário.
Trabalhar com conexões compartilhadas	ConnectBusy	Verdadeiro (.T.) se uma conexão compartilhada estiver ocupada; caso contrário, falso (.F.).
Controlar a exibição de interface	DispLogin	Controla quando a caixa de diálogo Logon ODBC é exibida.
	DispWarnings	Controla se as mensagens de aviso de erro são exibidas ou não.
Controlar intervalos de tempo	ConnectTimeout	Especifica o tempo (em segundos) de espera antes de retornar um erro de tempo limite de conexão.
	IdleTimeout	Especifica o intervalo de tempo limite inativo (em segundos). Conexões ativas qualificadas são desativadas depois do intervalo de tempo especificado. ¹
	WaitTime	Controla a quantidade de tempo decorrido em milissegundos antes que o Visual FoxPro verifique se a instrução SQL foi executada por completo.
	QueryTimeout	Controla o tempo (em segundos) de espera antes de retornar um erro geral de tempo limite.
Gerenciar transações	Transactions	Determina como a conexão gerencia transações na tabela remota.
Controlar a carga dos conjuntos de resultados nos cursores de visualização	Asynchronous	Especifica se os conjuntos de resultados são fornecidos de forma síncrona (padrão) ou assíncrona.
	BatchMode	Especifica se SQLEXEC() retorna conjuntos de resultados de uma vez (padrão) ou individualmente com SQLMORERESULTS().
	PacketSize	Especifica o tamanho do pacote de rede utilizado pela conexão.
Exibir identificadores ODBC internos	ODBCdbc2	O identificador de conexão ODBC interno que pode ser utilizado por arquivos de

ODBCstmt²

biblioteca externos (arquivos .FLL) para chamar as funções da API do ODBC.

O identificador de instrução ODBC interno que pode ser utilizado pelos arquivos de biblioteca externos (arquivos .FLL) para chamar as funções da API do ODBC.

¹ Se estiver no modo de transação manual, a conexão não será desativada.

² Se uma conexão for desativada, os valores ODBCdbc e ODBCstmt não serão mais válidos. Não libere ou abandone esses valores na biblioteca do usuário.

Para obter maiores informações sobre as propriedades de conexão e suas definições padrão, consulte [SQLSETPROP\(\)](#).

Controlando as definições de propriedade de ambiente

Os valores definidos no ambiente do Visual FoxPro utilizando o identificador 0 são utilizados como protótipos ou valores padrão para cada conexão ou anexo subsequente.

► Para visualizar as definições de propriedade de ambiente atuais

- Utilize [SQLGETPROP\(\)](#) com 0 como o valor para o identificador.

O exemplo a seguir exibe na tela a definição da propriedade WaitTime do ambiente atual:

```
? SQLGETPROP(0, "WaitTime")
```

Se você definir a propriedade DispWarnings como (.T.), o Visual FoxPro exibe qualquer erro de ambiente deste ponto em diante e define também DispWarnings como verdadeiro (.T.) para as conexões recém-criadas.

Embora os valores definidos para o identificador 0 sejam utilizados como valores protótipo para cada conexão, você pode também definir propriedades personalizadas para uma conexão individual, emitindo [SQLSETPROP\(\)](#) para este identificador de conexão. As exceções são as propriedades ConnectTimeout, PacketSize, e DispLogin, cujas definições a conexão herda no momento da conexão. Se você alterar a definição das propriedades ConnectTimeout, PacketSize ou DispLogin, a nova definição só será utilizada quando você se reconectar.

Controlando objetos de conexão e visualização

Você pode controlar conexões e visualizações definindo propriedades no objeto de visualização ou conexão. As propriedades que controlam os bancos de dados, tabelas, campos de tabelas, definições e campos de visualizações, conexões definidas ou cursores de conexões ativas são chamadas *propriedades engine*. Você pode exibir ou definir propriedades de mecanismo com umas destas funções do Visual FoxPro:

Para exibir o uso de propriedades engine

[CURSORGETPROP\(\)](#)

[DBGETPROP\(\)](#)

[SQLGETPROP\(\)](#)

Para definir o uso de propriedades engine

[CURSORSETPROP\(\)](#)

[DBSETPROP\(\)](#)

[SQLSETPROP\(\)](#)

A função utilizada depende de se você deseja definir propriedades sobre objeto 0 (conexão 0 e cursor 0), a definição do objeto em um banco de dados (conexão definida ou definição de visualização) ou o objeto ativo (conexão ativa ou cursor de visualização ativo). A tabela a seguir lista objetos e as funções utilizadas para definir propriedades em cada objeto:

Para definir propriedades para

Conexão

Visualização

Objeto 0	SQLSETPROP()	CURSORSETPROP()
Definição de objeto em um banco de dados	DBSETPROP()	DBSETPROP()
Objeto ativo	SQLSETPROP()	CURSORSETPROP()

Propriedades engine

A tabela a seguir lista as propriedades de mecanismo em ordem alfabética com os objetos que utilizam cada propriedade.

LEGENDA

- Somente para leitura
- ◐ Somente para leitura local ;
leitura-gravação remota
- Leitura-gravação

Propriedades

	Banco de dados	Tabela	Campo da tabela	Definição da visualização	Campo da visualização	Definição da conexão	Conexão ativa	Cursor ativo
Asynchronous						●	●	
BatchMode						●	●	
BatchUpdateCount			●					●
Buffering								●
Caption		●		●				
Comment	●	●	●	●	●	●		
CompareMemo			●					●
ConnectBusy							○	
ConnectHandle								○
ConnectName			●				○	○
ConnectionString						●	○	
ConnectTimeout						●	●	
Database								○
DataSource						●	○	
DataType				○				
DefaultValue		○		●				
DeleteTrigger	○							
DispLogin						●	●	
DispWarnings						●	●	
FetchAsNeeded			●					●
FetchMemo			●					○
FetchSize			●					●
IdleTimeout						●	●	
InsertTrigger	○							
KeyField				●				
KeyFieldList								●
MaxRecords			●					●
ODBCdbc							○	
ODBCstmt							○	
Offline			○					○
PacketSize						●	●	
ParameterList			●					●
Password						●	○	
Path	○							
Prepared			●					●
PrimaryKey	○							
QueryTimeout						●	●	
RuleExpression	○	○	●	●				
RuleText	○	○	●	●				

LEGENDA

- Somente para leitura
- Somente para leitura local ;
leitura-gravação remota
- Leitura-gravação

Propriedades

	Banco de dados	Tabela	Campo da tabela	Definição da visualização	Campo da visualização	Definição da conexão	Conexão ativa	Cursor ativo
Asynchronous						●	●	
BatchMode						●	●	
BatchUpdateCount			●					●
Buffering								●
Caption		●		●				
Comment	●	●	●	●	●	●		
CompareMemo			●					●
ConnectBusy							○	
ConnectHandle								○
ConnectName			●				○	○
ConnectionString						●	○	
ConnectTimeout						●	●	
Database								○
DataSource						●	○	
DataType				○				
DefaultValue		○		●				
DeleteTrigger		○						
DispLogin						●	●	
DispWarnings						●	●	
FetchAsNeeded			●					●
FetchMemo			●					○
FetchSize			●					●
IdleTimeout						●	●	
InsertTrigger		○						
KeyField				●				
KeyFieldList								●
MaxRecords			●					●
ODBCdbc							○	
ODBCstmt							○	
Offline			○					○
PacketSize						●	●	
ParameterList			●					●
Password						●	○	
Path		○						
Prepared			●					●
PrimaryKey		○						
QueryTimeout						●	●	
RuleExpression		○	○	●	●			
RuleText		○	○	●	●			
SendUpdates			●					●
ShareConnection			●					○
SourceName								○
SourceType			○					○
SQL			○					○
Tables			●					●
Transactions						●	●	
Updatable				●				
UpdatableFieldList								●

RuleExpression		○	○	●	●				
RuleText		○	○	●	●				
SendUpdates				●					●
ShareConnection				●					○
SourceName									○
SourceType				○					○
SQL				○					○
Tables				●					●
Transactions							●	●	
Updatable					●				
UpdatableFieldList									●
UpdateName					●				
UpdateNameList									●
UpdateTrigger		○							
UpdateType				●					●
UseMemoSize				●					○
UserID							●	○	
Version		○							
WaitTime							●	●	
WhereType				●					●

Propriedade de mecanismo

Relativo a

Asynchronous	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
Batchmode	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
BatchUpdateCount ¹	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativos: consulte CURSORSETPROP() .
Buffering	Cursors de visualizações ativos: consulte CURSORSETPROP() .
Caption	Campos em tabelas, campos com definições de visualização: consulte DBSETPROP() .
Comment	Bancos de dados, tabelas, campos em tabelas, definições de visualização, campos em definições de visualização, definições de conexão: consulte DBSETPROP() .
CompareMemo	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativos: consulte CURSORSETPROP() .
ConnectBusy	Conexões ativas: consulte SQLGETPROP() .
ConnectHandle	Cursors de visualizações ativos: consulte CURSORGETPROP() .
ConnectName ¹	Definições de visualização: consulte DBSETPROP() . Conexões ativas: consulte SQLGETPROP() . Cursors de visualizações ativos: consulte CURSORGETPROP() .
ConnectionString	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLGETPROP() .
ConnectTimeout	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .

Database	Cursores de visualizações ativos: consulte CURSORGETPROP() .
DataSource	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLGETPROP() .
DataType	Campos em definições de visualização: consulte DBSETPROP() .
DefaultValue	Campos em tabelas, campos em definições de visualização: consulte DBSETPROP() .
DeleteTrigger	Tabelas: consulte DBGETPROP() .
DispLogin	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
DispWarnings	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
FetchAsNeeded	Definições de visualização: consulte DBSETPROP() . Cursores de visualizações ativos: consulte CURSORGETPROP() .
FetchMemo ¹	Definições de visualização: consulte DBSETPROP() . Cursores de visualizações ativos: consulte CURSORGETPROP() .
FetchSize1	Definições de visualização: consulte DBSETPROP() . Cursores de visualizações ativas: consulte CURSORSETPROP() .
IdleTimeout	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
InsertTrigger	Tabelas: consulte DBGETPROP() .
KeyField	Campos em definições de visualização: consulte DBSETPROP() .
KeyFieldList ²	Cursores de visualizações ativas: consulte CURSORSETPROP() .
MaxRecords ¹	Definições de visualização: consulte DBSETPROP() . Cursores de visualizações ativas: consulte CURSORSETPROP() .
ODBCHdbc	Conexões ativas: consulte SQLGETPROP() .
ODBCHstmt	Conexões ativas: consulte SQLGETPROP() .
Offline	Definições de visualização: consulte DBGETPROP() .
PacketSize	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
ParameterList	Definições de visualização: consulte DBSETPROP() . Cursores de visualizações ativas: consulte CURSORSETPROP() .
Password	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLGETPROP() .
Path	Tabelas: consulte DBGETPROP() .
Prepared	Definições de visualização: consulte DBSETPROP() .

PrimaryKey	Tabelas: consulte DBGETPROP() .
QueryTimeOut	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
RuleExpression	Tabelas, campos em tabelas, definições de visualização, campos em definições de visualização: consulte DBSETPROP() .
RuleText	Tabelas, campos em tabelas, definições de visualização, campos em definições de visualização: consulte DBSETPROP() .
SendUpdates ²	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativas: consulte CURSORSETPROP() .
ShareConnection	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativas: consulte CURSORGETPROP() .
SourceName	Cursors de visualizações ativas: consulte CURSORGETPROP() .
SourceType	Definições de visualização: consulte DBGETPROP() . Cursors de visualizações ativos: consulte CURSORGETPROP() .
SQL	Definições de visualização: consulte DBGETPROP() . Cursors de visualizações ativos: consulte CURSORGETPROP() .
Tables ²	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativas: consulte CURSORSETPROP() .
Transactions	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
Updatable	Campos em definições de visualização: consulte DBSETPROP() .
UpdatableFieldList ²	Cursors de visualizações ativas: consulte CURSORSETPROP() .
UpdateName	Campos em definições de visualização: consulte DBSETPROP() .
UpdateNameList ²	Cursors de visualizações ativas: consulte CURSORSETPROP() .
UpdateTrigger	Tabelas: consulte DBGETPROP() .
UpdateType	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativas: consulte CURSORSETPROP() .
UseMemoSize ¹	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativas: consulte CURSORGETPROP() .
UserID	Definições de conexão: consulte DBSETPROP() .

	Conexões ativas: consulte SQLGETPROP() .
Version	Bancos de dados: consulte DBGETPROP() .
WaitTime	Definições de conexão: consulte DBSETPROP() . Conexões ativas: consulte SQLSETPROP() .
WhereType	Definições de visualização: consulte DBSETPROP() . Cursors de visualizações ativos: consulte CURSORSETPROP() .

¹ Propriedade útil principalmente para visualizações remotas; a definição não tem efeito sobre o desempenho em visualizações locais. Você pode definir essa propriedade em visualizações remotas se desejar predefinir a propriedade na visualização local e, posteriormente, migrar para criar uma visualização remota.

² A propriedade deve ser definida para atualizações a serem enviadas a uma fonte de dados remota.

Utilizando transações com dados remotos

Você também pode quebrar transações em atualizações, exclusões e inserções em dados remotos utilizando um dos dois métodos a seguir:

- Modo de transação automático
- Modo de transação manual

O modo de transação selecionado determina como o Visual FoxPro lida com as transações em sua máquina local.

Utilizando o modo de transação automático

Como padrão, o Visual FoxPro quebra automaticamente todo comando de transação enviado para o servidor remoto em uma transação. Este tratamento padrão é fornecido quando a propriedade Transactions é definida como 1 ou DB_TRANSAUTO.

► Para utilizar o modo de transação automático

- Utilize a função [DBSETPROP\(\)](#) para definir a propriedade Transactions na conexão como 1 ou DB_TRANSAUTO.
– Ou –
- Utilize a função [SQLSETPROP\(\)](#) para definir a propriedade Transactions na conexão ativa como 1 ou DB_TRANSAUTO.

O processamento de transações para a tabela remota é automaticamente manipulado.

Observação Os comandos [BEGIN TRANSACTION](#) e [END TRANSACTION](#) do Visual FoxPro criam uma transação apenas para o cursor local do Visual FoxPro. Eles não estendem a transação para o servidor remoto.

Controlando as transações manualmente

Se você quiser controlar transações manualmente, pode definir a propriedade Transactions como 2 ou DB_TRANSMANUAL. Com o tratamento manual, o Visual FoxPro inicia automaticamente uma transação quando você emite a primeira instrução SQL. Entretanto, a função [SQLCOMMIT\(\)](#) ou [SQLROLLBACK\(\)](#) do Visual FoxPro deve ser submetida para que a transação seja finalizada.

► Para utilizar o modo de transação manual

- Utilize o comando [DBSETPROP\(\)](#) para definir a propriedade Transactions na conexão como 2 ou DB_TRANSMANUAL.
– Ou –
- Utilize o comando [SQLSETPROP\(\)](#) para definir a propriedade Transactions na conexão ativa como 2 ou DB_TRANSMANUAL.

O processamento de transações é tratado manualmente através de [SQLCOMMIT\(\)](#) e [SQLROLLBACK\(\)](#).

Depois de gravar ou fazer o rollback para a transação anterior, o Visual FoxPro automaticamente começa uma nova, quando você emite a próxima instrução SQL de transação. Para obter maiores informações sobre as transações, consulte o capítulo 17, [Programando para acesso compartilhado](#)

Transações aninhadas

O Visual FoxPro suporta transações aninhadas em até cinco níveis para dados locais. Um único nível de suporte de transações é interno à passagem SQL.

Se o servidor suportar vários níveis de transações, você pode utilizar a passagem SQL para gerenciar os níveis de transação explicitamente. No entanto, o gerenciamento explícito de transações é complexo porque pode ser difícil controlar a interação entre a transação interna e o tempo das transações do servidor remoto. Para obter maiores informações sobre gerenciamento explícito de transações, consulte a documentação sobre o ODBC.

Trabalhando com dados remotos utilizando a passagem SQL

Depois de recuperar um conjunto de resultados utilizando a passagem SQL, você pode visualizar e controlar as propriedades do cursor de conjunto de resultados utilizando as funções [CURSORGETPROP\(\)](#) e [CURSORSETPROP\(\)](#) do Visual FoxPro. Estas funções são as mesmas usadas para definir propriedades em um cursor de visualização ativo.

Observação Os cursores não são objetos e não estão ligados ao modelo de objeto. No entanto, você pode visualizar as suas propriedades, ou atributos, com [CURSORGETPROP\(\)](#) e definir suas propriedades com [CURSORSETPROP\(\)](#).

Definindo as propriedades de cursor para dados remotos

A tabela a seguir relaciona as propriedades de cursor do Visual FoxPro que suportam o trabalho com visualizações e conjuntos de resultados conectados, agrupados de acordo com categorias de tarefas.

Propriedades de cursor do Visual FoxPro

Tarefa	Propriedade	Objetivo
Visualizar definição do cursor	SQL	Contém a instrução SQL a partir da qual o cursor foi criado.
Controlar as interações entre o Visual FoxPro e o ODBC	ConnectHandle	Identificador para conexão remota utilizado pelo cursor.
	ConnectName	Nome da conexão utilizada pelo cursor.
	Prepare	Especifica se a consulta para a visualização está preparada antes de ser executada.
	FetchAsNeeded	Especifica se as linhas são carregadas automaticamente durante o loop ou somente em uma base conforme o necessário.
	CompareMemo	Especifica se os campos Memo e Geral participarão da cláusula WHERE de uma

		instrução UPDATE, independentemente da definição da propriedade UpdateType
	FetchMemo	Especifica se os campos Memo e Geral são carregados automaticamente com conjuntos de resultados ou carregados posteriormente, sob demanda, quando um desses campos é aberto.
	UseMemoSize	Especifica o tamanho mínimo da coluna (1 a 255) em conjuntos de resultados para os quais as colunas são retornadas em campos Memo.
	FetchSize	Especifica o número de linhas carregadas de cada vez do conjunto de resultados remoto.
	MaxRecords	Especifica o número máximo de linhas carregadas quando conjuntos de resultados são retornados.
Atualizar dados	SendUpdates ¹	Especifica se as atualizações ao cursor são enviadas às tabelas nas quais o cursor se baseia.
	BatchUpdateCount	Especifica o número de instruções de atualização enviadas para a tabela especializada à qual se utilizou buffer.
	Tables ¹	Lista delimitada por vírgulas de nomes de tabela na fonte de dados; utilizada para definir o escopo para as propriedades UpdateNameList e UpdatableFieldsList.
	KeyFieldList ¹	Lista delimitada por vírgulas de campos do Visual FoxPro que representam a chave primária do conjunto de resultados utilizado para atualizações.
	UpdateNameList ¹	Lista delimitada por vírgulas correspondendo campos do Visual FoxPro no cursor com os nomes de tabela e coluna dos campos aos quais deseja enviar atualizações.

UpdatableFieldList ¹	Lista delimitada por vírgulas dos campos do Visual FoxPro aos quais as atualizações são enviadas.
Buffering	Especifica o tipo de utilização de buffer executada no cursor.
UpdateType	Especifica se as atualizações devem ocorrer utilizando os comandos UPDATE ou DELETE e, em seguida, INSERT.
WhereType	Especifica o que deve ser incluído na cláusula WHERE para atualizações aos dados na tabela.


¹ As propriedades que devem ser definidas antes que você atualize dados.

Estas propriedades são utilizadas para controlar a forma como o aplicativo interage com dados remotos, estabelecendo o número de linhas recuperadas durante a carga progressiva e controlando a utilização de buffer e as atualizações aos dados remotos.

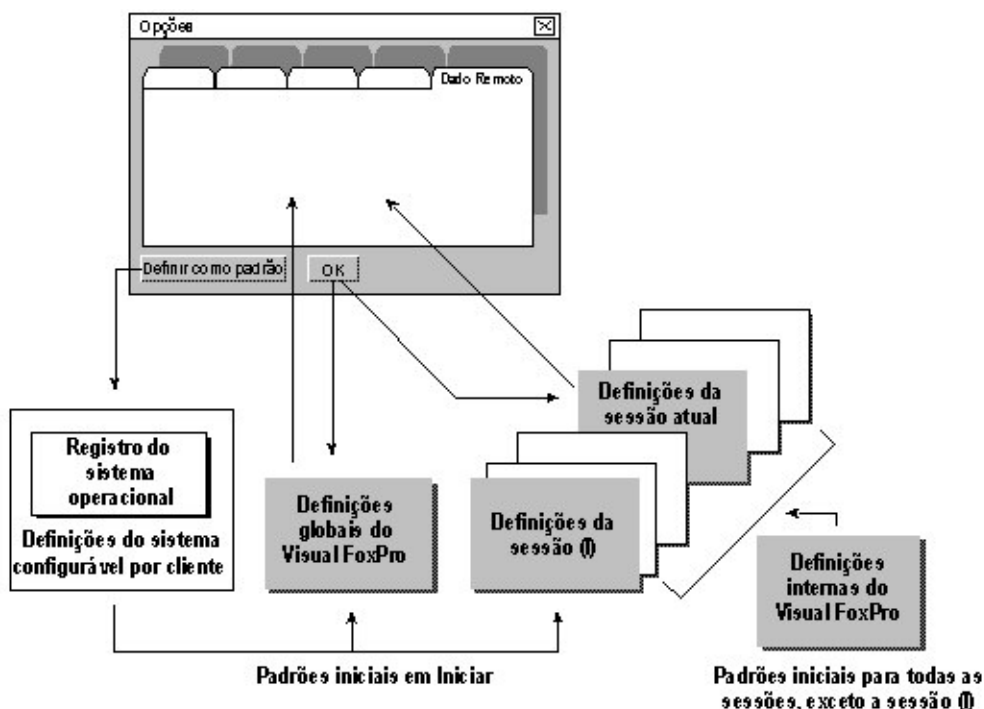
Utilizando a guia dados remotos na caixa de diálogo opções

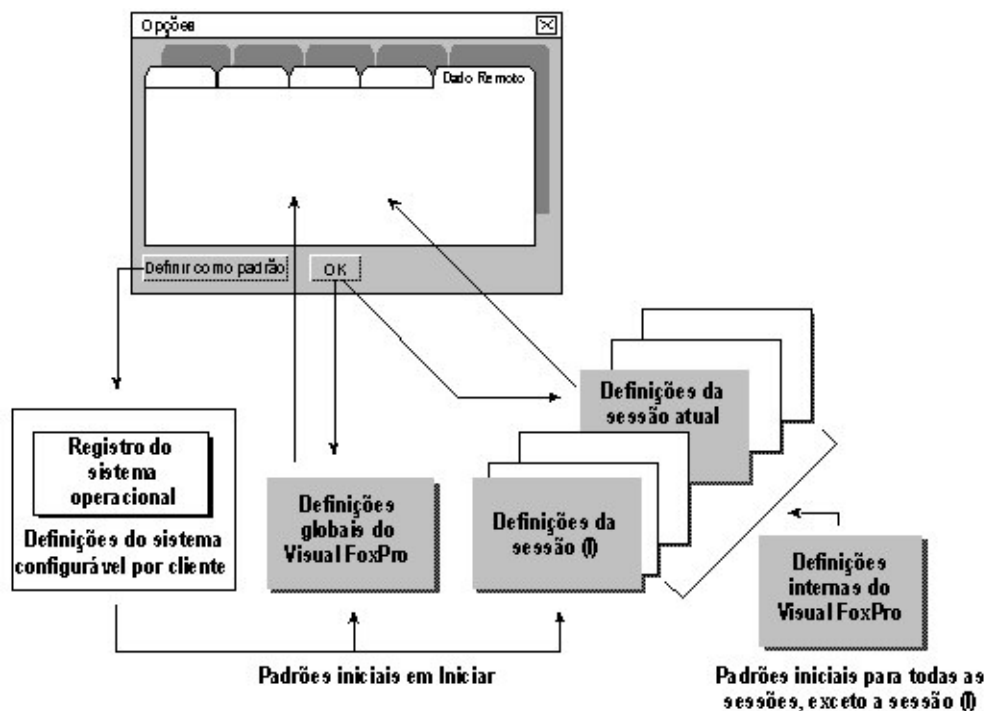
Algumas propriedades de cursor herdam seus valores iniciais do ambiente; outras só ficam disponíveis em nível do cursor. Algumas delas estão disponíveis aos cursores que representam visualizações remotas e tabelas conectadas através da passagem SQL ou ODBC.

Você pode controlar algumas definições de propriedade de cursor e conexão através da guia **Dados remotos** da caixa de diálogo **Opções**. Quando você exibe a guia **Dados remotos**, os valores na caixa de diálogo representam as definições de cursor para a sessão atual e as definições globais padrão do Visual FoxPro para a conexão. Quando você altera valores na guia **Dados remotos** e escolhe **OK**, os novos valores são salvos na sessão atual do cursor e nas definições globais padrão da conexão. Se você escolher **Definir como padrão**, os valores serão gravados nas definições de sistema configuráveis na sua máquina. O diagrama a seguir ilustra estas interações:

Para obter uma versão animada, clique aqui. 

Visualize e especifique definições globais e de sessão na caixa de diálogo Opções





Definindo propriedades com a passagem SQL


Quando você cria um cursor, ele herda as definições de propriedade, como UpdateType e UseMemoSize, do cursor de ambiente ou cursor 0 da sessão atual. Você pode alterar estas definições de propriedade padrão, utilizando a função `CURSORSETPROP()` com 0 como número de cursor.

Depois de criar um cursor de visualização com a passagem SQL, você pode alterar as definições de propriedade do cursor ativas, utilizando a função `CURSORSETPROP()` para o cursor de visualização. As alterações feitas com `CURSORSETPROP()` são temporárias: as definições temporárias para a visualização ativa desaparecem quando você fecha a visualização e as definições temporárias para o Cursor 0 somem quando você fecha a sessão do Visual FoxPro.

As conexões herdam propriedades de forma semelhante. As propriedades padrão para a conexão 0 são herdadas quando você cria e armazena uma conexão definida em um banco de dados. Você pode alterar estas definições de propriedade padrão para a conexão 0 com a função `SQLSETPROP()`. Depois que a conexão tiver sido criada e armazenada em um banco de dados, você poderá alterar as propriedades de conexão com a função `DBSETPROP()`. Quando você utiliza uma conexão, as definições de propriedade armazenadas para a conexão no banco de dados são herdadas pela conexão ativa. É possível alterar estas propriedades na conexão ativa, utilizando a função `SQLSETPROP()` para o identificador de conexão.

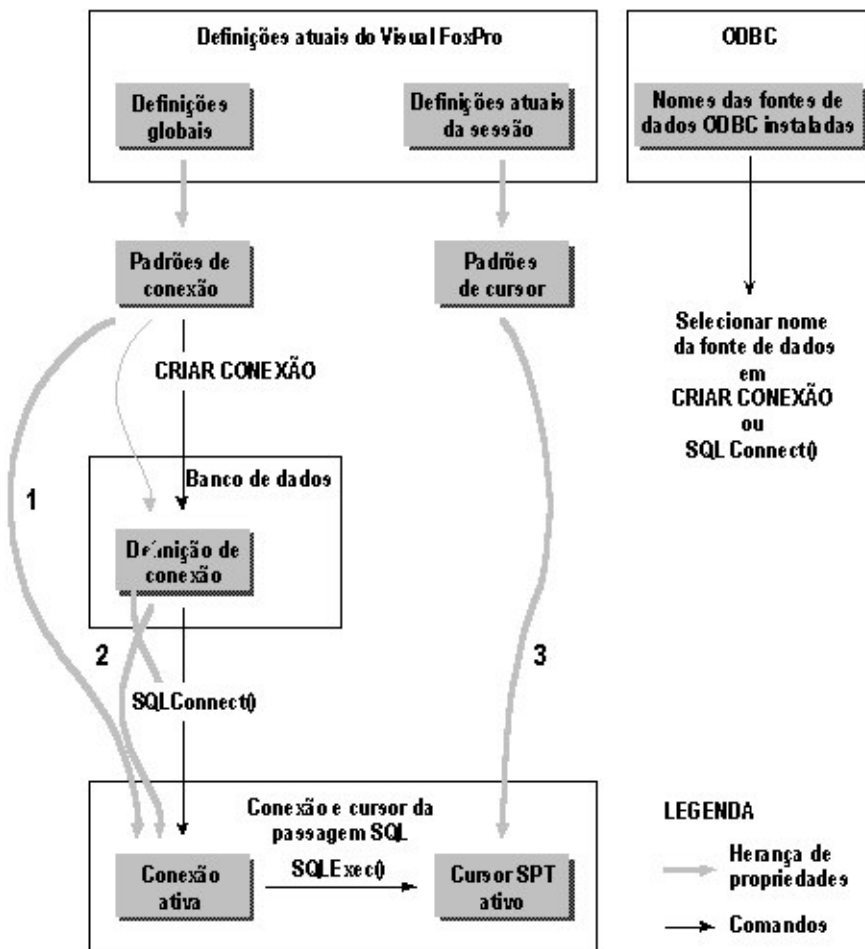
Tanto os cursores de visualização da passagem SQL quanto as conexões definidas podem utilizar uma fonte de dados ODBC definida. Se você utilizar uma fonte de dados ODBC em um cursor de visualização da passagem SQL, a conexão herda as propriedades dos padrões da sessão.

O diagrama a seguir ilustra a herança de propriedades dos cursores e conexões criados com a passagem SQL. As linhas cinzas representam o fluxo de herança de propriedade; as linhas pretas representam comandos do Visual FoxPro.

Para obter uma versão animada, clique aqui. 

Herança de propriedades do cursor e da conexão da passagem SQL (SPT).





- Se a conexão ativa estiver com base em uma fonte de dados ODBC, suas propriedades serão herdadas da conexão ODBC.
- Se a conexão ativa estiver com base em uma conexão de banco de dados, suas propriedades serão herdadas da definição de conexão no banco de dados.
- As propriedades do cursor de passagem SQL ativo são herdadas dos padrões de cursor ODBC.

Atualizando dados remotos com a passagem SQL

Quando você utiliza as funções da passagem SQL para atualizar dados em um servidor remoto, você controla se os dados são atualizados, assim como detalhes específicos sobre as atualizações, definindo as propriedades no cursor de conjunto de resultados. Quando uma atualização é solicitada, o Visual FoxPro verifica estas propriedades antes de gravar a atualização.

Para atualizar dados remotos, você deve definir cinco propriedades: Tables, KeyFieldList, UpdateNameList, UpdatableFieldList e SendUpdates. Você pode especificar propriedades adicionais como Buffering, UpdateType e WhereType para melhor atender às exigências do aplicativo.

► Para ativar atualizações em um cursor de visualização ativo

- Utilize a função `CURSORSETPROP()` para especificar as propriedades de atualização do cursor de visualização: Tables, KeyFieldList, UpdateNameList, UpdatableFieldList e SendUpdates.

Dica Os cursores de visualização da passagem SQL não são atualizáveis até que você especifique propriedades de atualização para eles. Se precisar armazenar definições de propriedade de atualização de forma permanente, crie uma definição de visualização. O Visual FoxPro fornece valores padrão que preparam a visualização para atualização quando você cria uma visualização utilizando a linguagem ou o **Criador de visualizações**. Você pode utilizar a função

CURSORSETPROP() para adicionar informações que complementam ou personalizam os valores padrão.

As propriedades de atualização definidas no cursor de visualização ativo têm nomes ligeiramente diferentes de seus correspondentes em [DBSETPROP\(\)](#). A tabela a seguir lista os nomes utilizados para as definições de visualização e os cursores ativos.

Propriedades de atualização de visualização e de cursor

Objetivo	Propriedades de definição de visualizações ¹	Propriedades do cursor ativo ²
Tornar tabelas remotas atualizáveis.	Tables	Tables
Especificar os nomes remotos para os campos de visualização.	UpdateName (propriedade em nível de campo)	UpdateNameList
Especificar os campos de visualização que você deseja usar como chave.	KeyField (propriedade em nível de campo)	KeyFieldList
Especificar os campos de visualização que são atualizáveis.	Updatable (propriedade em nível de campo)	UpdatableFieldList
Ativar as atualizações.	SendUpdates	SendUpdates

¹ Definido com DBSETPROP().

² Definido com CURSORSETPROP().

Para obter maiores informações sobre como definir propriedades de atualização, consulte o capítulo 8, [Criando visualizações](#), ou consulte [DBSETPROP\(\)](#) ou [CURSORSETPROP\(\)](#).

Cronometrando atualizações remotas

Você controla a utilização de buffer para atualizações de dados remotos, definindo a propriedade Buffering do cursor. Das cinco definições de propriedade de utilização de buffer possíveis, duas são válidas para visualizações remotas:

- 3 ou DB_BUFOPTROW, padrão, que causa um bloqueio otimista na linha.
- 5 ou DB_BUFOPTTABLE, que causa um bloqueio otimista na tabela.

O Visual FoxPro suporta apenas o bloqueio otimista em cursores remotos.

Observação As definições de utilização de buffer em tabelas e linhas pessimistas, 2 e 4, não se aplicam a visualizações remotas, porque o Visual FoxPro não realiza bloqueios nos dados do servidor. A definição 1 da propriedade Buffering não se aplica a visualizações remotas porque sempre são utilizados buffers nas visualizações.

Utilizando buffer de linha otimista

A definição padrão de Buffering, DB_BUFOPTROW, causa um bloqueio otimista nos dados remotos linha a linha. Por exemplo, se quiser que as alterações à tabela titles sejam gravadas linha a linha, como no caso da utilização do comando SKIP, você pode definir a propriedade Buffering como 3:


```
CURSORSETPROP('buffering', 3, 'titles')
```

Quando a propriedade Buffering é definida para utilização de buffer de linha, há dois métodos de enviar atualizações para o servidor remoto. Você poderá:

- Chamar a função [TABLEUPDATE\(\)](#).
- Utilizar um comando que move o ponteiro de registro para fora da linha, como [SKIP](#) ou [GO BOTTOM](#).

A função [TABLEUPDATE\(\)](#) atualiza o servidor sem mover o ponteiro de registro. Os comandos que o movem enviam atualizações para o servidor remoto como consequência da retirada do registro atualizado.

Se você utilizar o buffer de linha e quiser posteriormente reverter as alterações às linhas, deve quebrá-las em uma transação manual, com as funções de transação da passagem SQL.

Utilizando buffer de tabela otimista

Se você quiser que as alterações a uma tabela sejam gravadas um lote de cada vez, como ocorre quando o usuário clica sobre um botão **Salvar** ou **OK** em um formulário, deve definir a propriedade Buffering como 5 ou `DB_BUFOPTTABLE`. Você deve chamar a função [TABLEUPDATE\(\)](#) para enviar a atualização para o servidor.

No exemplo a seguir, você define a propriedade Buffering no código de inicialização do formulário e depois grava as alterações no código de salvamento:

Código	Comentários
<pre>CURSORSETPROP('buffering', 5, 'sqltitles')</pre>	Define no código Init
<pre>* Atualiza alterações em lote; * ignora alterações feitas por outros usuários TABLEUPDATE(.T., .T., 'titles')</pre>	Define no código Save

Para restaurar os valores originais para uma tabela e evitar que as atualizações sejam enviadas a um servidor remoto, chame [TABLEREVERT\(\)](#). Você pode controlar se uma única linha ou todas elas serão revertidas, combinando a definição da propriedade Buffering do cursor com o comando [TABLEREVERT\(\)](#). O exemplo a seguir reverte apenas a linha atual. Convém chamar este código quando o usuário clicar sobre o botão **Cancelar** em um formulário:

```
= TABLEREVERT(.F., 'titles')      && Reverte a linha atual
```

Se quiser reverter todas as linhas, como ocorre quando o usuário pressiona ESC para sair de um formulário, você pode usar o mesmo exemplo, desta vez alterando as definições da propriedade Buffering e o comando [TABLEREVERT\(\)](#) para reverter todas as linhas com o buffer inteiro da tabela:

```
= TABLEREVERT(.T., 'titles')      && Reverte todas as linhas
```

Para obter maiores informações sobre a utilização de buffer, consulte o capítulo 17, [Programando para acesso compartilhado](#).

Detectando alterações feitas por outros usuários

Em aplicativos multiusuários, os conflitos com atualizações de outros usuários são detectados pela consulta de Atualização SQL, gerada quando ocorre uma tentativa de gravação local. O nível de detecção depende da definição da propriedade WhereType. Para obter maiores informações sobre como definir a propriedade WhereType, consulte o capítulo 8, [Criando visualizações](#).

Forçando atualizações

Você pode utilizar a função [TABLEUPDATE\(\)](#) para controlar se as alterações feitas a uma tabela ou cursor por outro usuário em uma rede serão substituídas quando você enviar atualizações. Se definir o parâmetro Force de [TABLEUPDATE\(\)](#) como verdadeiro (.T.) e a propriedade UpdateType de

CURSORSETPROP() for definida como o valor padrão 1, os dados antigos serão atualizados com o envio de novos dados, desde que o valor no campo-chave do registro na tabela remota não tenha sido alterado. Se o valor no campo-chave da tabela remota tiver sido alterado, ou se a propriedade UpdateType for definida como 2, o Visual FoxPro enviará uma instrução DELETE e, em seguida, INSERT, para a tabela remota.

Resolvendo problemas com mensagens de erro de atualização

A tabela a seguir registra as mensagens de erro do Visual FoxPro e do ODBC que se aplicam especificamente a atualizações remotas. A coluna Ação contém a ação que deve ser seguida para resolver a condição de erro.

Mensagem de Erro	Significado	Ação
Nenhuma tabela de atualização especificada. Utilize a propriedade de cursor Tables.	A propriedade de cursor Tables não contém nomes de tabelas remotas. Pelo menos uma tabela é necessária para ativar as atualizações para o servidor remoto.	Utilize a propriedade Tables para especificar pelo menos uma tabela para o cursor.
Nenhuma coluna-chave especificada para a tabela de atualização " <i>nome_tabela</i> ". Utilize a propriedade de cursor KeyFieldList.	A chave primária para a tabela remota especificada na mensagem de erro não está incluída na propriedade de cursor KeyFieldList; é necessária uma chave primária para cada tabela que está sendo atualizada.	Utilize a propriedade KeyFieldList a fim de especificar a chave primária para a tabela remota.
Nenhuma tabela de atualização válida especificada para a coluna " <i>nome_coluna</i> ". Utilize as propriedades de cursor UpdateNameList e Tables.	A propriedade UpdateName para a coluna " <i>nome_coluna</i> " tem um qualificador de tabela inválido.	Defina o qualificador de tabela com a propriedade UpdateNameList, ou adicione o qualificador de tabela à definição da propriedade Tables, ou ambos.
A propriedade de cursor KeyField List não define uma chave exclusiva.	Mais de um registro remoto possui a mesma chave.	Utilize a propriedade KeyField List para definir uma chave exclusiva para a tabela remota.
Do ODBC: objeto ODBC inválido.	O ODBC não pode localizar a tabela ou coluna remota porque este nome não existe conforme definido. Os nomes de campo do Visual FoxPro são validados pelo Visual	Verifique o nome do objeto.

FoxPro; os nomes de tabela e coluna remotos são validados apenas pelo servidor remoto.

Para obter maiores informações sobre o tratamento de erros, consulte [Manipulando erros de passagem SQL](#) posteriormente neste capítulo.

Escolhendo um modo de processamento eficiente para a passagem SQL

O Visual FoxPro fornece dois modos de processamento para recuperar e atualizar dados remotos utilizando a passagem SQL: síncrono e assíncrono. Ao utilizar as funções da passagem SQL, você pode escolher o método preferido. Não é preciso escolher um método para visualizações remotas; o Visual FoxPro emprega automaticamente a carga progressiva e gerencia o modo de processamento nas visualizações remotas.

Vantagens do modo síncrono

Como padrão, as funções SQL do Visual FoxPro são processadas de forma síncrona. O Visual FoxPro não retorna o controle a um aplicativo até que uma chamada de função seja concluída. O processamento síncrono é útil quando você está trabalhando com o Visual FoxPro interativamente.

Vantagens do modo assíncrono

O processamento assíncrono fornece maior flexibilidade do que o processamento síncrono. Por exemplo, quando o aplicativo está processando uma função de forma assíncrona, ele pode criar um termômetro para exibir o andamento da instrução em execução, exibir o movimento do ponteiro do mouse, criar loops e definir cronômetros para permitir a interrupção de um processamento mais demorado.

Utilizando a passagem SQL de forma assíncrona

O seu aplicativo pode pedir o processamento assíncrono para as quatro funções que submetem pedidos a uma fonte de dados e recuperam dados: [SQLEXEC\(\)](#), [SQLMORERESULTS\(\)](#), [SQLTABLES\(\)](#) e [SQLCOLUMNS\(\)](#). O processamento assíncrono é ativado definindo-se a propriedade *Asynchronous* da conexão com a função [SQLSETPROP\(\)](#). Quando a comunicação assíncrona é estabelecida para a conexão, todas as quatro funções operam de forma assíncrona.

► Para verificar a definição da propriedade *Asynchronous*

- Utilize a função [SQLGETPROP\(\)](#) para visualizar a definição da propriedade *Asynchronous*. No exemplo a seguir, `nConnectionHandle` representa o número do identificador da conexão ativa:

```
? SQLGETPROP(nConnectionHandle, 'Asynchronous')
```

► Para ativar o processamento assíncrono

- Utilize a função [SQLSETPROP\(\)](#) para especificar a propriedade *Asynchronous*:

```
? SQLSETPROP(nConnectionHandle, 'Asynchronous', .T.)
```

No modo Assíncrono, você deve chamar cada função repetidas vezes até que ela forneça um valor diferente de 0 (ainda em execução). Você pode cancelar o processamento da função ainda durante sua execução, pressionando a tecla ESC se a propriedade SET ESCAPE estiver definida como verdadeiro (.T.).

Até que a função tenha terminado de processar, o aplicativo poderá usar um identificador de conexão apenas com [SQLCANCEL\(\)](#) ou com a função assíncrona — [SQLEXEC\(\)](#), [SQLMORERESULTS\(\)](#), [SQLTABLES\(\)](#), ou [SQLCOLUMNS\(\)](#) — originalmente associada ao identificador. Você não poderá chamar as outras três funções assíncronas ou [SQLDISCONNECT\(\)](#) com o mesmo identificador de conexão até que a função tenha terminado.

Processando vários conjuntos de resultados

O aplicativo recupera vários conjuntos de resultados quando você utiliza a função `SQLEEXEC()` para emitir mais de uma instrução SQL SELECT ou para executar um procedimento armazenado que emite várias instruções SELECT. Os resultados de cada instrução SQL SELECT são retornados em um cursor separado do Visual FoxPro.

O nome padrão `SQLRESULT` é utilizado para o primeiro cursor. Os cursores subseqüentes recebem nomes exclusivos a partir de um índice do nome padrão. Por exemplo, os nomes padrão para os cursores retornados por uma instrução `SQLEEXEC()` que requer três conjuntos de resultados são `Sqlresult`, `Sqlresult1` e `Sqlresult2`.

Em modo de processamento em lote, se uma função retornar vários conjuntos de resultados, os respectivos nomes de cursor no Visual FoxPro terão sufixos exclusivos e poderão ter até 255 caracteres. O exemplo a seguir define a propriedade `BatchMode` para modo de processamento em lote e, em seguida, emite uma instrução `SQLEEXEC()` contendo quatro instruções SQL SELECT que retornam quatro conjuntos de resultados.

```
? SQLSETPROP(nConnectionHandle, 'BatchMode', .T.)
? SQLEEXEC(nConnectionHandle, 'select * from authors ;
    select * from titles ;
    select * from roysched ;
    select * from titleauthor', 'ITEM')
```

Quando a função acima tiver completado o processamento, o Visual FoxPro retorna quatro conjuntos de resultados como os cursores `Item`, `Item1`, `Item2` e `Item3` do Visual FoxPro.

Você pode alterar o nome padrão, utilizando o parâmetro `cNomeCursor` com as funções `SQLEEXEC()` ou `SQLMORERESULTS()`. Se o nome especificado para um conjunto de resultados tiver sido utilizado, o novo conjunto de resultados substituirá o cursor existente.

Quando o aplicativo recupera vários conjuntos de resultados, você pode escolher entre os modos de processamento assíncrono ou síncrono e em lote ou individual.

Utilizando o modo de processamento em lote


A propriedade `BatchMode`, definida como `SQLSETPROP()`, controla como `SQLEEXEC()` retorna vários conjuntos de resultados. O valor padrão é `.T.`, para o modo de processamento em lote. Processamento em lote significa que o Visual FoxPro só retorna resultados de uma chamada `SQLEEXEC()` em execução quando todos os conjuntos de resultados individuais tiverem sido recuperados.

Utilizando o modo de processamento individual

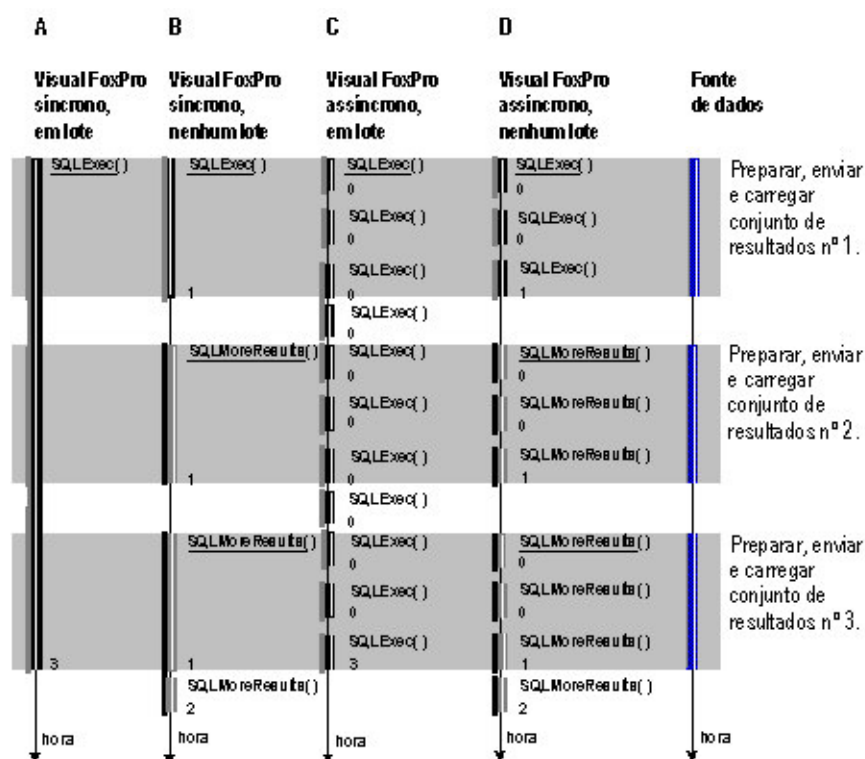
Se você utiliza `SQLSETPROP()` para definir a propriedade `BatchMode` como `.F.`, para o modo de processamento individual, cada conjunto de resultados é retornado individualmente. O primeiro conjunto de resultados é retornado pela chamada de função `SQLEEXEC()`. O aplicativo deve então chamar `SQLMORERESULTS()` repetidas vezes até que um valor 2 seja retornado, indicando que não há mais resultados disponíveis.

Em modo individual, o nome do cursor pode ser alterado em cada chamada `SQLMORERESULTS()` subseqüente. Assim, se, como no exemplo anterior, o nome do primeiro cursor em uma seqüência `SQLEEXEC()` for `Item`, e a segunda chamada `SQLMORERESULTS()` alterar o parâmetro `cNomeCursor` para `Otheritem`, os cursores resultantes serão chamados de `Item`, `Item1`, `Otheritem` e `Otheritem1`.

A seção a seguir descreve os modos de processamento em lote e individual com maiores detalhes sobre processamento síncrono e assíncrono. O diagrama a seguir fornece uma representação das quatro combinações possíveis de processamento. Os números 0, 1 e 2 representam os valores retornados quando você chama cada função.

Para obter uma versão animada, clique aqui. 

Modos de processamento síncrono e assíncrono do Visual FoxPro



O comportamento de cada tipo de processamento está descrito a seguir; as etiquetas A, B, C e D fazem referência ao diagrama anterior. Cada explicação considera a execução de uma instrução que fornecerá três conjuntos de resultados, representados no diagrama por três faixas horizontais.

Utilizando o processamento síncrono

No modo síncrono, o controle só volta ao aplicativo quando a execução de uma função for concluída.

A: modo de processamento em lote síncrono

Quando você executa uma instrução de passagem SQL de forma síncrona no modo de processamento em lote, o controle só é retornado quando todos os conjuntos de resultados tiverem sido recuperados. Você especifica o nome do primeiro cursor, utilizando o parâmetro `cNomeCursor` na função original. Se o cursor especificado já existir, será substituído pelo conjunto de resultados. Quando você solicita vários conjuntos de resultados em modo de processamento em lote síncrono, o Visual FoxPro cria os nomes de cursores adicionais indexando exclusivamente o nome do primeiro cursor.

B: modo de processamento individual síncrono

Quando você executa uma instrução de passagem SQL de forma síncrona em modo individual, a primeira instrução recupera o primeiro conjunto de resultados e retorna 1. Você deverá chamar a função `SQLMORERESULTS()` repetidas vezes e, opcionalmente, especificar um novo nome para o cursor. Se não fizer isso, serão criados vários nomes para vários conjuntos de resultados através da indexação exclusiva do nome base. Quando `SQLMORERESULTS()` retornar o valor 2, não haverá mais resultados disponíveis.

Utilizando o processamento assíncrono

No modo assíncrono, o aplicativo deve continuar chamando a mesma função de passagem SQL até

que esta retorne um valor diferente de zero (em execução). O nome do conjunto de resultado padrão, `Sqlresult`, pode ser alterado explicitamente com o parâmetro `cNomeCursor` na primeira vez em que a função é chamada. Se o nome especificado para um conjunto de resultados já tiver sido utilizado, o novo conjunto de resultados substituirá as informações no cursor existente.

C: modo de processamento em lote assíncrono

Quando você executa de forma assíncrona no modo de processamento em lote, cada chamada repetida da função original retorna a zero (ainda em execução) até que todos os conjuntos de resultados tenham sido retornados para os cursores especificados. Quando todos os resultados tiverem sido recuperados, o valor retornado é o número de cursores ou um número negativo que indica um erro.

D: modo de processamento individual assíncrono

Ao processar de forma assíncrona em modo de processamento individual, `SQLEXEC()` retorna o valor 1 quando completa a recuperação de cada conjunto de resultados. O aplicativo deve então chamar `SQLMORERESULTS()` repetidas vezes até o valor 2 ser retornado, indicando que não há mais resultados disponíveis.

Dica Os conjuntos de resultados remotos são recuperados em duas etapas: primeiro, o conjunto é preparado no servidor; em seguida, é carregado em um cursor local do Visual FoxPro. No modo assíncrono, você pode chamar a função `USED()` para ver se o Visual FoxPro iniciou a carga do cursor solicitado.

Controlando a conversão de tipo de dado

Quando você move dados entre um servidor remoto e o Visual FoxPro, talvez encontre diferenças na variedade de tipos de dados disponíveis no servidor ou no Visual FoxPro, pois raramente existe uma correlação de um para um entre os tipos de dados disponíveis em uma fonte de dados remota e os disponíveis no Visual FoxPro. Para lidar com estas diferenças, o Visual FoxPro utiliza tipos de dados ODBC para mapear tipos de dados remotos para tipos de dados locais do Visual FoxPro. Ao entender como os tipos de dados são mapeados entre o ODBC e o Visual FoxPro, você poderá prever como os dados remotos do servidor serão tratados no aplicativo do Visual FoxPro.

Se precisar, você também poderá ajustar os tipos de dados utilizados no servidor ou no aplicativo. O tipo de dado de campo padrão do Visual FoxPro pode ser substituído, criando uma visualização para o conjunto de dados remotos e, em seguida, definindo a propriedade de campo de visualização `DataType` no banco de dados. A propriedade `DataType` é uma propriedade de caractere indicando o tipo de dado desejado para cada campo de uma visualização remota. Para obter maiores informações sobre a propriedade `DataType`, consulte `DBSETPROP()`.

Descarregando e carregando dados de visualizações remotas

Quando você recupera dados de uma fonte de dados ODBC remota, o Visual FoxPro converte o tipo de dado de cada campo ODBC em um tipo equivalente do Visual FoxPro no cursor do conjunto de resultados. A tabela a seguir lista os tipos de dados disponíveis em fontes de dados ODBC e seus equivalentes no Visual FoxPro.

Tipo de dado ODBC de campo remoto	Tipo de dado de campo no cursor do Visual FoxPro
SQL_CHAR SQL_VARCHAR SQL_LONGVARCHAR	Caractere ou Memo1
SQL_BINARY SQL_VARBINARY SQL_LONGVARBINARY	Memo

SQL_DECIMAL SQL_NUMERIC	Numérico ou Moeda ²
SQL_BIT	Lógico
SQL_TINYINT SQL_SMALLINT SQL_INTEGER	Inteiro
SQL_BIGINT	Caractere
SQL_REAL SQL_FLOAT SQL_DOUBLE	Duplo; o número de casas decimais é o valor de SET DECIMAL no Visual FoxPro
SQL_DATE	Data
SQL_TIME	Data e Hora ³
SQL_TIMESTAMP	Data e Hora ⁴

¹ Se a largura do campo ODBC for menor do que o valor da propriedade de cursor UseMemoSize, ele se tornará um campo do tipo Caractere no cursor do Visual FoxPro; caso contrário, se tornará um campo do tipo Memo.

² Se o campo do servidor requerer um tipo de dado monetário, irá se tornar um tipo de dado Moeda no Visual FoxPro.

³ A data assume como padrão 1/1/1900.

⁴ Se o valor no campo SQL_TIMESTAMP contiver frações de segundos, as frações serão truncadas quando o valor for convertido em um tipo de dado Data e Hora do Visual FoxPro.

Observação Valores nulos em campos de fontes de dados ODBC tornam-se nulos no cursor do Visual FoxPro, independente da definição de SET NULL no Visual FoxPro no momento em que o aplicativo estiver recuperando dados remotos.

Convertendo parâmetros do Visual FoxPro para tipos de dados de visualizações remotas

Se houver dados do Visual FoxPro em um cursor resultante de dados remotos, estes voltam ao tipo ODBC original quando enviados para o servidor remoto. Se você enviar dados originados no Visual FoxPro para o servidor remoto via passagem SQL, as conversões a seguir serão aplicadas.

Tipo de dado Visual FoxPro	Tipo de dado ODBC
Caractere	SQL_CHAR ou SQL_LONGVARCHAR ¹
Moeda	SQL_DECIMAL
Data	SQL_DATE ou SQL_TIMESTAMP ²
Data e Hora	SQL_TIMESTAMP
Duplo	SQL_DOUBLE
Inteiro	SQL_INTEGER
Geral	SQL_LONGVARBINARY
Lógico	SQL_BIT
Memo	SQL_LONGVARCHAR
Numérico	SQL_DOUBLE

¹ Se a variável de memória do Visual FoxPro que mapeia um parâmetro criar uma expressão cuja largura seja menor do que 255, ela se tornará um tipo SQL_CHAR na fonte de dados ODBC; caso contrário, se tornará um tipo SQL_LONGVARCHAR.

² Os dados de Data do Visual FoxPro são convertidos em SQL_DATE para todas as fontes de dados ODBC, exceto o SQL Server, onde se tornará SQL_TIMESTAMP.

Mapeando um parâmetro do Visual FoxPro em um tipo de dado remoto

Você pode mapear um valor de parâmetro do Visual FoxPro para um tipo de dado remoto específico, formatando o parâmetro como uma expressão de caractere que utiliza a sintaxe para o tipo de dado remoto desejado. Por exemplo, se o servidor retornar um tipo de dado Data e Hora, você pode criar o parâmetro do Visual FoxPro como uma expressão de caractere no formato utilizado pelo seu servidor para representar os dados de Data e Hora. Quando o servidor receber o valor do parâmetro, tentará mapear os dados formatados para o tipo de dado Data e Hora.

Observação Quando você enviar um parâmetro para o servidor remoto, certifique-se de que o tipo de dado na cláusula WHERE corresponde com o tipo de dado utilizado para a expressão de parâmetro.

Manipulando erros da passagem SQL

Se uma função de passagem SQL retornar um erro, o Visual FoxPro armazena a mensagem de erro em uma matriz. A função **AERROR()** retorna informações sobre erros que são detectados em qualquer um dos níveis de componente: o Visual FoxPro, a fonte de dados ODBC ou o servidor remoto. Ao examinar os valores retornados por **AERROR()**, você pode determinar o erro que ocorreu no servidor e o respectivo texto da mensagem de erro.

Importante Você deve chamar **AERROR()** imediatamente para obter informações de erro. Se gerar qualquer outro antes de chamar **AERROR()**, as informações de erro serão perdidas.