

Se você quiser um conjunto de dados atualizáveis personalizado para o seu aplicativo, poderá utilizar visualizações. As visualizações combinam as qualidades de [tabelas](#) e de [consultas](#): como em uma consulta; você pode utilizar uma visualização para extrair um conjunto de dados de uma ou mais tabelas relacionadas; como em uma tabela, você pode utilizar uma visualização para atualizar as informações nela contidas e armazenar permanentemente os resultados no disco. Você também pode utilizar visualizações para colocar seus dados off-line e coletar ou modificar dados do seu sistema principal.

Este capítulo abrange a criação e a atualização de visualizações através da linguagem de programação, bem como a definição de [propriedades](#) para otimizar o desempenho de suas visualizações. Para obter maiores informações sobre bancos de dados, consulte o capítulo 6. Se quiser obter maiores informações sobre tabelas ou índices, consulte o capítulo 7. Para obter maiores informações sobre o **Criador de visualizações**, consulte o capítulo 5 no *Guia do Usuário*.

Este capítulo aborda os tópicos a seguir:

- [Criando uma visualização](#)
- [Utilizando visualizações](#)
- [Atualizando dados em uma visualização](#)
- [Combinando visualizações](#)
- [Trabalhando com dados off-line](#)
- [Otimizando o desempenho das visualizações](#)

Criando uma visualização

Você cria uma visualização quase da mesma maneira que cria uma [consulta](#). Selecione as tabelas e os campos que deseja incluir na visualização, especifique as condições de associação utilizadas para relacionar as tabelas e especifique [filtros](#) para escolher registros específicos. Diferentemente das consultas, nas visualizações você também pode selecionar como as alterações feitas nos dados de uma visualização serão enviadas para as tabelas originais, ou [base](#), a partir das quais a visualização é construída.

Quando você cria uma visualização, o Visual FoxPro armazena uma [definição de visualização](#) no banco de dados atual. A definição contém os nomes das tabelas e dos campos das visualizações selecionados e as definições de suas [propriedades](#). Quando você utiliza a visualização, a definição de visualização é utilizada para criar uma instrução SQL que define o conjunto de dados da visualização. Para obter informações sobre propriedades de visualização, consulte [Definindo propriedades de visualização e conexão](#) posteriormente neste capítulo ou consulte `DBGETPROP()` ou `CURSORGETPROP()`.

Você pode criar dois tipos de visualizações: [local](#) e [remota](#). Visualizações remotas utilizam [sintaxe SQL remota](#) para selecionar informações de tabelas em uma fonte de dados ODBC remota. As visualizações locais utilizam [sintaxe SQL do Visual FoxPro](#) para selecionar informações de tabelas ou visualizações. Você pode adicionar uma ou mais visualizações remotas a uma visualização local, o que lhe permite acessar informações do Visual FoxPro e de fontes de dados ODBC remotas da mesma maneira. Para obter informações sobre como acessar dados locais e remotos em uma única visualização, consulte a seção [Combinando dados locais e remotos em uma visualização](#) posteriormente neste capítulo.

Criando uma visualização local

Você pode criar uma visualização local com o **Criador de visualizações** ou com o comando `CREATE SQL VIEW`.

► Para criar uma visualização local

- No **Gerenciador de projetos**, selecione um banco de dados, em seguida, selecione **Visualizações locais** e **Novo** para abrir o **Criador de visualizações**.

– Ou –

- Utilize o comando **CREATE SQL VIEW** quando um banco de dados estiver aberto, para exibir o **Criador de visualizações**.

– Ou –

- Utilize o comando **CREATE SQL VIEW** com a cláusula **AS**.

Por exemplo, o código abaixo cria uma visualização que contém todos os campos da tabela `products`:

```
CREATE SQL VIEW product_view AS SELECT * ;  
FROM testdata!products
```

O nome da nova visualização será exibido no **Gerenciador de projetos**. Se você abrir o **Criador de bancos de dados**, a visualização será exibida da mesma maneira que uma tabela no [esquema](#), com o nome da visualização substituindo o nome da tabela.

No exemplo anterior, o nome da tabela é precedido, ou [qualificado](#), pelo nome do banco de dados da tabela e o símbolo "!". Se você qualificar o nome da tabela quando criar uma visualização, o Visual FoxPro irá procurar a tabela na lista de bancos de dados abertos, inclusive no banco de dados atual e em todos os bancos de dados anteriores, e no caminho de procura padrão da tabela.

Se você não qualificar uma tabela com um nome de banco de dados em uma [definição de visualização](#), o banco de dados deverá estar aberto para que a visualização possa ser utilizada.

Dica Quando você cria ou utiliza uma visualização no **Gerenciador de projetos**, o **Gerenciador de projetos** abre o banco de dados automaticamente. Se utilizar posteriormente uma visualização fora do projeto, você deverá abrir o banco de dados ou certificar-se de que ele está no escopo para que possa utilizar a visualização.

Criando visualizações com instruções SQL SELECT armazenadas

É possível utilizar a [substituição de macro](#) para armazenar a instrução SQL SELECT em uma [variável](#) que você pode chamar com a cláusula **AS** do comando **CREATE SQL VIEW**. Por exemplo, o código abaixo armazena uma instrução SQL SELECT na variável `emp_cust_sql`, que, em seguida, é utilizada para criar uma nova visualização.

```
emp_cust_sql = "SELECT employee.emp_id, ;  
customer.cust_id, customer.emp_id, ;  
customer.contact, customer.company ;  
FROM employee, customer ;  
WHERE employee.emp_id = customer.emp_id"  
CREATE SQL VIEW emp_cust_view AS &emp_cust_sql
```

Modificando visualizações

Você pode modificar as visualizações existentes no **Criador de visualizações** utilizando o **Gerenciador de projetos** ou a linguagem. Se quiser modificar a sequência SQL da visualização através da linguagem de programação, será preciso criar uma nova visualização. Em seguida, você pode salvar a nova [definição de visualização](#) e sobrescrever o nome da visualização existente. Para modificar as propriedades da visualização, consulte [Definindo propriedades de visualização e conexão](#) posteriormente neste capítulo.

Dica No **Criador de visualizações**, é possível abrir uma visualização existente e copiar uma sequência SQL somente para leitura como um atalho para criar uma nova visualização através da linguagem de programação.

► Para modificar uma visualização no Criador de visualizações

- No **Gerenciador de projetos**, selecione o nome da visualização e, em seguida, selecione **Modificar** para abrir o **Criador de visualizações**.

– Ou –

- Abra um banco de dados e utilize o comando **MODIFY VIEW** com o nome da visualização.

No **Criador de visualizações**, você pode utilizar o menu **Consulta** ou a **Barra de ferramentas criador de visualizações** para adicionar uma nova tabela à visualização. O código abaixo exibe `product_view` no **Criador de visualizações**:

```
OPEN DATABASE testdata
MODIFY VIEW product_view
```

Renomeando uma visualização

É possível renomear uma visualização no **Gerenciador de projetos** ou com o comando `RENAME VIEW`.

► Para alterar o nome de uma visualização

- No **Gerenciador de projetos**, selecione um banco de dados, em seguida, o nome da visualização e, em seguida, selecione **Renomear arquivo** no menu **Projeto**.
 - Ou –
- Utilize o comando `RENAME VIEW`.

Por exemplo, o código abaixo renomeia `product_view` para `products_all_view`:

```
RENAME VIEW product_view TO products_all_view
```

O banco de dados que contém a visualização deve estar aberto para que você possa renomear a visualização.

Excluindo uma visualização

É possível excluir uma [definição de visualização](#) de um banco de dados utilizando o **Gerenciador de projetos** ou o comando `DELETE VIEW`. Antes de excluir a visualização, certifique-se de que o banco de dados que contém a visualização está aberto e definido como o banco de dados atual.

► Para excluir uma visualização

- No **Gerenciador de projetos**, selecione um banco de dados, em seguida, o nome da visualização e, em seguida, selecione **Remover**.
 - Ou –
- Utilize o comando `DELETE VIEW` ou `DROP VIEW`.

Por exemplo, o código abaixo exclui `product_view` e `customer_view` do banco de dados:

```
DELETE VIEW product_view
DROP VIEW customer_view
```

Observação Esses comandos possuem o mesmo efeito; `DROP VIEW` é a sintaxe ANSI SQL padrão para excluir uma visualização SQL.

Criando uma visualização de várias tabelas

Para acessar informações relacionadas armazenadas em tabelas separadas, você pode criar uma visualização e adicionar duas ou mais tabelas, ou modificar uma visualização existente adicionando tabelas. Para adicionar as tabelas, você pode utilizar o **Criador de visualizações** ou o comando `CREATE SQL VIEW`. Depois de adicionar as tabelas, você pode expandir o seu controle dos resultados da visualização, utilizando a condição de associação definida entre as tabelas.

► Para criar uma visualização de várias tabelas

- No **Gerenciador de projetos**, crie uma visualização e adicione as tabelas que você deseja no **Criador de visualizações**.
 - Ou –
- Abra um banco de dados e utilize o comando `CREATE SQL VIEW`, adicionando nomes de tabelas à cláusula `FROM` e às condições de associação.

A simples adição das tabelas ao comando CREATE SQL VIEW gera um produto cruzado. Você precisa especificar uma [condição de associação](#) na cláusula FROM ou WHERE da instrução para fazer a correspondência de registros entre tabelas. Se houver [relacionamentos permanentes](#) entre as tabelas, estes serão automaticamente utilizados como condições de associação.

Definindo e modificando condições de associação

Geralmente, para definir uma condição de associação, você utiliza os relacionamentos estabelecidos nos campos-chave [primário](#) e [estrangeiro](#) entre as tabelas. Por exemplo, você pode querer encontrar informações sobre pedidos, incluindo informações sobre o cliente que fez o pedido. Você pode criar uma visualização utilizando as tabelas Cliente e Pedidos. Você especifica uma condição de associação para comparar valores nos campos em comum e, geralmente, retornar os que são iguais. No exemplo, tanto Cliente quanto Pedidos possuem um campo ID do Cliente.

► Para definir condições de associação em uma visualização

- No **Gerenciador de projetos**, crie ou modifique uma visualização e, em seguida, adicione as tabelas desejadas no **Criador de visualizações**.

– Ou –

Abra um banco de dados e utilize o comando **CREATE SQL VIEW**, adicionando nomes de tabelas e condições de associação à cláusula FROM.

Associações internas especificadas no Criador de visualizações e exibidas na instrução SELECT - SQL

The screenshot shows the 'Criador de visualizações - Visual4' window. It displays three tables: Customer, Orders, and Orditems. The Customer table has fields: cust_id, company, contact, title, address. The Orders table has fields: order_id, cust_id, emp_id, to_name, to_address. The Orditems table has fields: line_no, order_id, product_id, unit_price, quantity. The 'Associação' tab is selected, showing two inner joins: Customer.cust_id = Orders.cust_id and Orders.order_id = Orditems.order_id. A separate window titled 'Visual4 [Somente leitura]' displays the generated SQL query:

```
FROM testdata!customer INNER JOIN testdata!orders;  
INNER JOIN testdata!orditems ;  
ON Orders.order_id = Orditems.order_id ;  
ON Customer.cust_id = Orders.cust_id
```

O código a seguir cria a nova visualização conforme descrito no exemplo acima, utilizando a cláusula FROM para especificar as condições de associação para a visualização:

```
OPEN DATABASE testdata  
CREATE SQL VIEW cust_orders_view AS ;  
SELECT * FROM testdata!customer ;  
INNER JOIN testdata!orders ;
```

```
ON customer.cust_id = orders.cust_id
```

A condição de associação possui vários aspectos: o tipo de associação, os campos para fazer a associação e o operador para comparar os campos. Neste caso, [associação interna](#), somente as linhas da tabela `cliente` que correspondam a um ou mais registros na tabela `pedidos` serão incluídas no resultado.

Para alterar os resultados da visualização e atender às suas necessidades específicas, você pode determinar:

- [Campos](#) na associação
- Operadores de comparação entre os campos
- Uma seqüência de associações, se você tiver duas [tabelas](#) na sua visualização
- O tipo da [associação](#)

A especificação das associações nos campos em vez de chaves [primárias](#) e [estrangeiras](#) pode ser útil em várias instâncias, mas não é utilizada na maioria das visualizações.

Alterando o operador de comparação, você pode controlar quais registros são comparados e retornados de forma semelhante a um [filtro](#). Por exemplo, se estiver utilizando um campo de data na associação, você poderá utilizar o operador de comparação para incluir apenas registros antes ou depois de determinada data.

Para obter maiores informações sobre a seqüência de associações, consulte [Definindo várias condições de associação](#) posteriormente neste capítulo.

A escolha de um tipo de [associação](#) diferente permite que você expanda os resultados da sua [consulta](#) para incluir registros que correspondam à [condição de associação](#) e também aqueles que não correspondem. Se tiver mais de duas tabelas na visualização, você poderá alterar os resultados alterando a ordem das associações na cláusula FROM.

Você pode modificar os tipos de associação na sua visualização utilizando o [Criador de visualizações](#) ou a linguagem.

► Para modificar um tipo de associação

- Selecione a guia **Associação**.
 - Ou –
- Clique duas vezes sobre a linha de associação.
 - Ou –
- Abra um banco de dados e utilize o comando [CREATE SQL VIEW](#), adicionando nomes de tabelas e condições de associação à cláusula FROM.

Incluindo registros não-correspondentes nos resultados

Se você desejar incluir linhas não-correspondentes nos seus resultados, poderá utilizar uma [associação externa](#). Por exemplo, você talvez queira uma lista de todos os clientes e se eles fizeram ou não um pedido. Além disso, para os clientes que fizeram pedidos, você talvez queira que os números de pedido sejam incluídos na visualização. Quando você utiliza uma associação externa, os [campos](#) vazios das linhas não-correspondentes retornam [valores nulos](#).

Você também pode utilizar a linguagem para criar essa visualização utilizando o código a seguir:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_view AS ;
    SELECT * FROM testdata!customer ;
        LEFT OUTER JOIN testdata!orders ;
            ON customer.cust_id = orders.cust_id
```

Para controlar que registros não-correspondentes estão incluídos na visualização, escolha a partir dos tipos de associação a seguir.

Para

Utilize

Retornar apenas registros das duas tabelas que correspondem à condição de comparação definida entre os dois campos na condição de associação.	Associação interna
Retornar todos os registros da tabela à esquerda da palavra-chave JOIN e apenas os registros correspondentes da tabela à direita da palavra-chave.	Associação externa esquerda
Retornar todos os registros da tabela à direita da palavra-chave JOIN e apenas os registros correspondentes da tabela à esquerda da palavra-chave.	Associação externa direita
Retornar registros correspondentes e não-correspondentes das duas tabelas.	Associação externa completa

Definindo várias condições de associação

Se você criar visualizações ou [consultas](#) com mais de duas tabelas, poderá alterar os resultados pela ordem em que as suas condições de associação são especificadas. Por exemplo, talvez você queira encontrar informações sobre os pedidos, incluindo informações sobre o funcionário que fez a venda e o cliente que fez o pedido. Você pode criar uma visualização utilizando as tabelas `cliente`, `pedidos` e `funcionário` e especificar condições de associação interna nos campos que elas possuem em comum: tanto `cliente` quanto `pedidos` possuem um campo ID do cliente; tanto `pedidos` quanto `funcionário` possuem um campo ID do funcionário.

Essa visualização possui a instrução SQL base a seguir:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_emp_view AS ;
    SELECT * FROM testdata!customer ;
        INNER JOIN testdata!orders ;
            ON customer.cust_id = orders.cust_id ;
                INNER JOIN testdata!employee ;
                    ON orders.emp_id = employee.emp_id
```

Utilizando associações na cláusula WHERE

Você pode especificar suas [condições de associação](#) na cláusula WHERE; no entanto, não é possível especificar um tipo de associação igual às associações na cláusula FROM. Para visualizações remotas, a cláusula de associação sempre aparece na cláusula WHERE.

O código a seguir cria a mesma visualização que o exemplo anterior, utilizando a cláusula WHERE para especificar as condições de associação para a visualização:

```
OPEN DATABASE testdata
CREATE SQL VIEW cust_orders_emp_view AS ;
    SELECT * FROM testdata!customer, ;
        testdata!orders, testdata!employee ;
    WHERE customer.cust_id = orders.cust_id ;
    AND orders.emp_id = employee.emp_id
```

Acessando dados remotos

Quando quiser utilizar dados localizados em um servidor remoto, crie uma [visualização remota](#). Para tal, você deve primeiro conectar-se a uma [fonte de dados](#).

Conectando-se a uma fonte de dados remota

Uma fonte de dados remota é normalmente um servidor remoto para o qual está instalado um driver de ODBC e está configurado um nome de fonte de dados ODBC. Para obter uma [fonte de dados](#),

válida, certifique-se de que o [ODBC](#) está instalado. No Visual FoxPro, você pode definir uma fonte de dados e conexões.

Para obter maiores informações sobre como configurar uma fonte de dados ODBC, consulte o capítulo 1, [Instalando o Visual FoxPro](#), no *Guia de Instalação e Índice Principal*.

Definindo uma conexão

No Visual FoxPro, você pode criar e armazenar uma [conexão definida](#) em um [banco de dados](#), à qual você poderá referir-se pelo nome quando criar uma [visualização remota](#). Você também pode definir [propriedades](#) na [conexão definida](#) para otimizar a comunicação entre o Visual FoxPro e a [fonte de dados](#) remota. Quando você ativa uma visualização remota, a [conexão](#) da visualização passa a ser o canal para a fonte de dados remota.

► Para criar uma conexão definida

- No **Gerenciador de projetos**, selecione **Conexões** e, em seguida, selecione **Novo** para abrir o **Criador de conexões**.
 - Ou –
- Abra um banco de dados e utilize o comando **CREATE CONNECTION** para abrir o **Criador de conexões**.
 - Ou –
- Utilize o comando **CREATE CONNECTION** com um nome de conexão.

Por exemplo, para criar uma conexão no banco de dados `testdata` que armazene as informações necessárias para conectar-se à fonte de dados ODBC `sqlremote`, você pode digitar o código a seguir:

```
OPEN DATABASE testdata
CREATE CONNECTION remote_01 DATASOURCE sqlremote userid password
```

O Visual FoxPro exibe `remote_01` como o nome da conexão no **Gerenciador de projetos**.

A criação de uma conexão definida no banco de dados não utiliza nenhum recurso remoto ou de rede, pois o Visual FoxPro só ativa a conexão quando você utiliza a visualização. Até você ativar a conexão, a conexão definida existe apenas como uma definição de conexão armazenada como uma linha no arquivo `.DBC` do banco de dados. Quando você utiliza uma visualização remota, o Visual FoxPro utiliza a conexão definida referenciada na visualização para criar uma conexão real com a fonte de dados remota e, em seguida, envia o pedido de dados para a fonte remota, utilizando a conexão ativa como canal.

Você pode criar, opcionalmente, uma visualização que especifique somente o nome da [fonte de dados](#), em vez de um nome de [conexão](#). Quando você utiliza a visualização, o Visual FoxPro utiliza as informações do [ODBC](#) sobre a fonte de dados para criar e ativar uma conexão com a fonte de dados. Quando você fecha a visualização, a conexão é fechada.

Precedência de nomes de conexões e bancos de dados

Quando utiliza o comando **CREATE SQL VIEW** com a cláusula **CONNECTION**, você especifica um nome que representa uma [conexão](#) ou uma [fonte de dados](#). O Visual FoxPro procura uma conexão com o nome especificado, primeiro, no banco de dados atual. Caso não exista uma conexão com o nome especificado neste banco de dados, o Visual FoxPro irá procurar uma fonte de dados ODBC estabelecida com o nome especificado. Se o banco de dados atual contiver uma conexão definida com o mesmo nome de uma fonte de dados ODBC do sistema, o Visual FoxPro irá localizar e utilizar a conexão definida.

Exibindo avisos de logon de ODBC

Quando você utilizar uma visualização cujas informações de logon de [conexão](#) não estão totalmente especificadas, o Visual FoxPro poderá exibir uma caixa de diálogo específica da [fonte de dados](#), solicitando as informações que estão faltando.

Você pode controlar se o Visual FoxPro exibe avisos solicitando informações que não foram especificadas na hora da conexão.

► Para controlar a exibição de avisos de logon de ODBC

1. No **Gerenciador de projetos**, selecione o nome da conexão e, em seguida, selecione **Modificar** para abrir o **Criador de conexões**.
2. Na área **Exibir avisos de logon do ODBC**, selecione uma opção.
 - Ou –
 - Utilize a propriedade DispLogin das funções **DBSETPROP()** ou **SQLSETPROP()**.

Utilizando uma conexão existente

É possível utilizar uma [conexão definida](#) existente para criar uma [visualização remota](#). Para exibir uma lista das conexões disponíveis em um banco de dados, utilize o **Gerenciador de projetos** ou o comando **DISPLAY CONNECTIONS**.

► Para determinar conexões existentes

- No **Gerenciador de projetos**, selecione um banco de dados e, em seguida, selecione **Conexões**.
 - Ou –
- Utilize o comando **DISPLAY CONNECTIONS**.

Por exemplo, o código abaixo exibe as conexões no banco de dados testdata:

```
OPEN DATABASE testdata
DISPLAY CONNECTIONS
```

Criando uma visualização remota

Dispondo de uma [fonte de dados](#) ou de uma [conexão definida](#), você poderá criar uma [visualização remota](#) utilizando o **Gerenciador de projetos** ou a linguagem. Uma visualização remota é semelhante a uma [visualização local](#), mas, ao defini-la, você adiciona um nome de fonte de dados ou conexão. A instrução SQL da visualização remota utiliza o dialeto do servidor nativo.

► Para criar uma visualização remota

- No **Gerenciador de projetos**, selecione um banco de dados, selecione **Visualizações remotas** e, em seguida, selecione **Novo** para abrir o **Criador de visualizações**.
 - Ou –
- Utilize o comando **CREATE SQL VIEW** com a cláusula **REMOTE** e/ou **CONNECTION**.

Se você utilizar a cláusula **CONNECTION** com o comando **CREATE SQL VIEW**, não será necessário incluir a palavra-chave **REMOTE**. O Visual FoxPro identifica a visualização como remota através da presença da palavra-chave **CONNECTION**. Por exemplo, se a tabela `products` do banco de dados `Testdata` estiver em um servidor remoto, o código abaixo criará uma visualização remota da tabela `products`:

```
OPEN DATABASE testdata
CREATE SQL VIEW product_remote_view ;
    CONNECTION remote_01 ;
    AS SELECT * FROM products
```

Você pode utilizar um nome de fonte de dados em vez de um nome de conexão quando criar uma visualização remota. Também pode optar por omitir o nome da fonte de dados ou conexão quando utilizar o comando **CREATE SQL VIEW** com a cláusula **REMOTE**. O Visual FoxPro exibirá, em seguida, a caixa de diálogo **Selecionar conexão** ou a [caixa de diálogo Fonte de dados](#), onde você poderá selecionar uma fonte de dados ou uma conexão válida.

Depois de criar uma visualização, você poderá abrir o **Criador de bancos de dados** e verá que a visualização está no [esquema](#) exibido da mesma maneira que uma tabela, com o nome e ícone da visualização no lugar de um nome e ícone de tabela.

Se você associar duas ou mais tabelas no **Criador de visualizações remotas**, o criador utilizará associações internas (ou [equi-associações](#)) e irá inserir a condição de associação na cláusula WHERE. Se desejar utilizar uma [associação externa](#), o **Criador de visualizações remotas** só fornecerá associações externas esquerdas, a [sintaxe](#) suportada pelo ODBC. Se precisar de associações externas completas ou direitas ou se quiser utilizar uma sintaxe nativa para uma associação externa esquerda, crie a visualização através da linguagem de programação.

Utilizando visualizações

Depois de criar uma visualização, você poderá utilizá-la para exibir e atualizar dados. Também é possível modificar as [propriedades](#) de uma visualização para melhorar o seu desempenho. Você trata uma visualização como uma tabela:

- Abre a visualização com o comando [USE](#) e inclui o nome da visualização.
- Fecha a visualização com o comando USE.
- Exibe registros da visualização em uma [janela Pesquisar](#).
- Exibe aliases de visualizações abertas na [janela Sessão de dados](#).
- Utiliza a visualização como uma [fonte de dados](#), como em um texto ou em um controle de grade, formulário ou relatório.

É possível utilizar uma visualização através do **Gerenciador de projetos** ou da linguagem.

► Para utilizar uma visualização

- No **Gerenciador de projetos**, selecione um banco de dados, escolha o nome da visualização e, em seguida, selecione **Pesquisar** para exibir a visualização em uma janela **Pesquisar**.

– Ou –

- Acesse a visualização através da linguagem de programação com o comando [USE](#).

O código abaixo exibe product_view em uma janela **Pesquisar**:

```
OPEN DATABASE testdata
USE product_view
BROWSE
```

Quando é utilizada, a visualização é aberta como um [cursor](#) em sua própria [área de trabalho](#). Se a visualização estiver baseada em tabelas locais, o Visual FoxPro também abrirá as [tabelas base](#) em áreas de trabalho separadas. As tabelas base de uma visualização são as tabelas acessadas pela instrução SELECT - SQL incluída no comando [CREATE SQL VIEW](#) quando a visualização é criada. No exemplo anterior, a utilização de product_view também abre automaticamente a tabela products.

Janela Sessão de dados exibe a visualização e a sua tabela base



Quando uma visualização está baseada em tabelas remotas, as tabelas base não são abertas em áreas de trabalho. Somente o nome da visualização remota é exibido na janela **Sessão de dados**.

Limitando o escopo de uma visualização

Ao acessar uma [fonte de dados](#) remota, você está acessando um grande volume de dados. É possível limitar o escopo dos dados selecionados na visualização a apenas registros necessários em um determinado momento. Isto reduz o tráfego na rede e melhora o desempenho da visualização. Por exemplo, se você quiser visualizar informações sobre os clientes de um país específico e sobre seus pedidos, poderá melhorar o desempenho se transferir para a visualização somente os registros relativos a esse país, em vez de transferir todos os clientes.

Um método que você pode utilizar para limitar o escopo da visualização é adicionar uma cláusula WHERE à instrução SQL da visualização. Se desejar visualizar os registros dos clientes da Suécia, você poderá criar a cláusula SQL WHERE para a visualização a seguir:

```
SELECT * FROM customer ;  
WHERE customer.country = 'Suécia'
```

Este código limita efetivamente o escopo da visualização transferindo registros apenas dos clientes suecos, mas também requer que você crie uma visualização separada para cada país, pois o valor real de customer.country de um país está codificado de forma permanente na instrução SELECT da visualização.

Criando uma visualização com parâmetros

Você pode limitar o escopo de uma visualização sem criar uma visualização separada para cada subconjunto de registros, criando uma visualização com parâmetros. Uma visualização com parâmetros cria, na instrução SQL SELECT da visualização, uma cláusula WHERE que limita os registros transferidos apenas aos registros que correspondam às condições da cláusula WHERE criada utilizando o valor fornecido para o parâmetro. Este valor pode ser fornecido em tempo de execução ou passado através da linguagem de programação para a visualização.

No caso do exemplo anterior, você pode criar uma visualização que lhe permita transferir registros de qualquer país, simplesmente digitando o nome do país quando utilizar a visualização.

► Para criar uma visualização com parâmetros

- No **Criador de visualizações**, selecione **Parâmetros de visualização** no menu **Consulta**.
– Ou –

- Utilize o comando **CREATE SQL VIEW** com o símbolo “?” e um [parâmetro](#).

O parâmetro fornecido é avaliado como uma [expressão](#) do Visual FoxPro e o valor é enviado como parte da instrução SQL da visualização. Em caso de erro na avaliação, o Visual FoxPro exibe um aviso solicitando o valor do parâmetro. Por exemplo, se a tabela cliente do banco de dados Testdata estiver em um servidor remoto, o código abaixo criará uma visualização remota com parâmetros que limita a visualização aos clientes cujo país corresponde ao valor fornecido para o parâmetro ?cCountry:

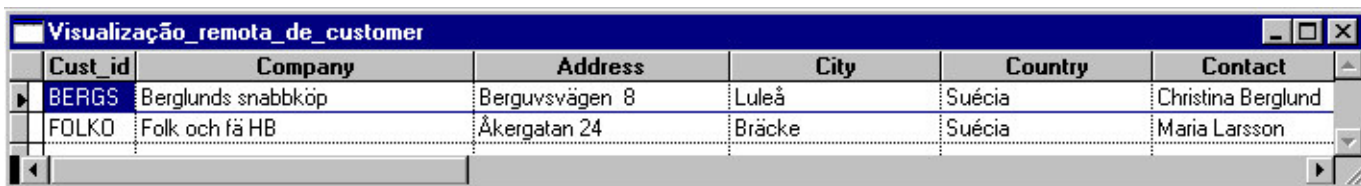
```
OPEN DATABASE testdata  
CREATE SQL VIEW customer_remote_view ;  
CONNECTION remote_01 ;  
AS SELECT * FROM customer ;  
WHERE customer.country = ?cCountry
```

Você pode fornecer um valor para ?cCountry através da linguagem de programação quando utilizar a visualização. Por exemplo, você pode digitar o código abaixo:

```
cCountry = 'Suécia'  
USE Testdata!customer_remote_view IN 0  
BROWSE
```

O Visual FoxPro exibirá os registros de clientes de empresas suecas na janela Pesquisar Customer_remote_view.

Visualização exibindo registros cujo país corresponde ao parâmetro fornecido



Cust_id	Company	Address	City	Country	Contact
BERGS	Berglunds snabbköp	Berguvsvägen 8	Luleå	Suécia	Christina Berglund
FOLKD	Folk och få HB	Åkergatan 24	Bräcke	Suécia	Maria Larsson

Dica Se o [parâmetro](#) for uma [expressão](#), coloque a expressão de parâmetros entre parênteses. Isto permite que a expressão inteira seja avaliada como parte do parâmetro.

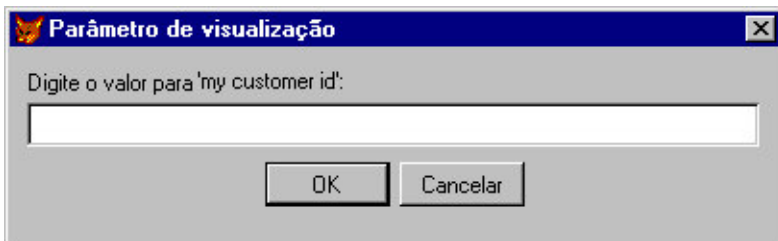
Exibindo um aviso solicitando que o usuário forneça um valor de parâmetro

Se o [parâmetro](#) não for uma [variável](#) ou uma [expressão](#), você pode querer exibir um aviso solicitando ao usuário que forneça o valor do parâmetro utilizando uma seqüência entre aspas como parâmetro da visualização. Quando você cria o parâmetro da visualização utilizando uma seqüência entre aspas após o símbolo "?", o Visual FoxPro não interpreta a seqüência como uma expressão. Em vez disso, você é avisado para digitar o valor do parâmetro em [tempo de execução](#). Por exemplo, o código abaixo cria uma visualização remota com parâmetros que avisa ao usuário para fornecer um valor para o parâmetro '?my customer id':

```
OPEN DATABASE testdata
CREATE SQL VIEW customer_remote_view ;
    CONNECTION remote_01 ;
    AS SELECT * FROM customer ;
    WHERE customer.cust_id = '?my customer id'
USE customer_remote_view
```

Quando você utiliza a visualização no exemplo anterior, é exibida a caixa de diálogo **Parâmetro de visualização**.

A caixa de diálogo **Parâmetro de visualização** solicita o valor na seqüência entre aspas



Depois que você digita uma identificação de cliente válida, o Visual FoxPro recupera o registro correspondente a essa identificação. Se você digitar o valor 'ALFKI' no exemplo anterior e, em seguida, pesquisar Customer_remote_view, o registro de clientes será exibido na janela Pesquisar.

Janela Pesquisar exibindo o registro de cust_id ALFKI



Cust_id	Company	Address	City
ALFKI	Alfreds Futterkiste	Obere Str. 57	Berlin

Utilizando uma seqüência entre aspas como um parâmetro de visualização, você certifica-se de que o Visual FoxPro solicitará sempre ao usuário o valor do parâmetro.

Abrindo várias instâncias de uma visualização

Você pode abrir várias instâncias de uma visualização em [áreas de trabalho](#) separadas, assim como abrir uma tabela em mais de uma área de trabalho. Ao contrário das [tabelas](#), as visualizações

carregam, como padrão, um novo conjunto de dados sempre que você utiliza a visualização.

► **Para abrir uma visualização em várias áreas de trabalho**

- No **Gerenciador de projetos**, selecione o nome da visualização e, em seguida, **Pesquisar** para exibir a visualização em uma janela **Pesquisar**. Repita o processo para abrir a visualização em outras áreas de trabalho.
 - Ou –
- Na janela **Sessão de dados**, selecione **Abrir** e, em seguida, o nome da visualização. Repita este processo para abrir a visualização em outras áreas de trabalho.
 - Ou –
- Acesse a visualização através da linguagem de programação com o comando **USE**.

Quando acessa a visualização através da linguagem de programação com o comando **USE**, você pode optar por abrir uma outra [instância](#) da visualização sem consultar novamente a [fonte de dados](#). Isto é particularmente útil quando você quer abrir uma visualização remota em várias [áreas de trabalho](#) sem esperar a transferência dos dados de uma fonte de dados remota.

► **Para utilizar novamente uma visualização sem transferir dados**

- Utilize a cláusula **NOREQUERY** com o comando **USE**.
 - Ou –
- Utilize a cláusula **AGAIN** com o comando **USE**.

O código abaixo utiliza a cláusula **NOREQUERY** para exibir o [cursor](#) carregado da primeira [instância](#) de `product_remote_view` em duas janelas **Pesquisar** sem consultar novamente a [fonte de dados](#) remota:

```
OPEN DATABASE testdata
CREATE SQL VIEW product_remote_view ;
    CONNECTION remote_01 ;
    AS SELECT * FROM products
USE product_remote_view
BROWSE
SELECT 0
USE product_remote_view NOREQUERY
BROWSE
```

É possível especificar um número de sessão com a cláusula **NOREQUERY**. Se não for especificado um número de sessão, o Visual FoxPro irá procurar em todas as [sessões](#). Caso um conjunto aberto seja localizado para a visualização, um [cursor](#) será novamente aberto no mesmo conjunto de resultados. Caso não seja localizado nenhum conjunto de resultados aberto, será carregado um novo conjunto de resultados para a visualização. Da mesma forma que com tabelas, se a visualização não for localizada, será aberto um novo cursor de visualização.

Se você quiser que o Visual FoxPro procure um conjunto de resultados aberto para a visualização apenas na sessão atual, poderá utilizar a cláusula **AGAIN**. O código abaixo exibe `product_remote_view` em duas janelas **Pesquisar**:

```
OPEN DATABASE testdata
USE product_remote_view
BROWSE
USE product_remote_view AGAIN in 0
BROWSE
```

Quando você utiliza a cláusula **AGAIN**, o Visual FoxPro procura um [cursor](#) de visualização existente na sessão atual e abre um [alias](#) adicional que indica este cursor de visualização. Abrir uma outra instância de uma visualização com a cláusula **AGAIN** equivale a emitir um comando **USE** com a cláusula **NOREQUERY** com o número da sessão atual.

Exibindo a estrutura de uma visualização

É possível abrir e exibir apenas a estrutura de uma visualização com a cláusula NODATA do comando USE. Este procedimento é particularmente útil quando você deseja visualizar a estrutura de uma [visualização remota](#) sem esperar a transferência dos dados.

► Para abrir uma visualização sem dados

- Acesse a visualização através da linguagem de programação com o comando **USE** e a cláusula NODATA.

O código abaixo exibe customer_remote_view sem dados em uma janela Pesquisar:

```
OPEN DATABASE testdata
USE customer_remote_view NODATA in 0
BROWSE
```

O uso de uma visualização com a cláusula NODATA sempre abre um novo [cursor](#) de visualização. A cláusula NODATA é a maneira mais rápida de se obter a estrutura da visualização, pois ela cria o menor cursor possível na fonte de dados remota. Quando esta cláusula é utilizada, o Visual FoxPro cria para a visualização uma cláusula WHERE que retorna sempre um valor falso. Visto que nenhum registro na [fonte de dados](#) corresponde à condição da cláusula WHERE, não será selecionada nenhuma linha no cursor da fonte de dados remota. A visualização é recuperada rapidamente, pois você não fica esperando que a fonte de dados remota construa um cursor potencialmente grande.

Dica Utilizar a cláusula NODATA é mais eficiente do que utilizar a definição 0 da propriedade MaxRecords na visualização ou no cursor. Quando você utiliza a propriedade MaxRecords, é necessário aguardar enquanto a fonte de dados remota constrói um cursor para a visualização contendo todas as linhas de dados correspondentes às condições normais da cláusula WHERE da visualização. Em seguida, as linhas do cursor da visualização remota completa são transferidas de acordo com a definição da propriedade MaxRecords.

Criando um índice em uma visualização

É possível criar [índices](#) locais em uma visualização, exatamente da mesma forma que em uma tabela, utilizando o comando **INDEX ON**. Ao contrário dos índices construídos em uma [tabela](#), os índices locais criados em uma visualização não são armazenados permanentemente: eles desaparecem quando a visualização é fechada.

Dica Considere o tamanho do conjunto de resultados da visualização ao decidir se criará um índice local em uma visualização. A indexação de um conjunto de resultados grande poderá levar muito tempo e reduzir o desempenho da visualização.

Para obter maiores informações sobre como criar índices, consulte o capítulo 7, [Trabalhando com tabelas](#), ou consulte [INDEX](#).

Criando relacionamentos temporários em visualizações

É possível criar [relacionamentos temporários](#) entre índices de visualização ou entre índices de visualização e índices de tabela com o comando **SET RELATION**.

Para obter um melhor desempenho quando utilizar o comando SET RELATION para relacionar uma visualização e uma tabela, faça a visualização do objeto pai e da tabela objeto filho no relacionamento. Este procedimento é mais eficiente porque o índice estrutural da tabela é mantido de forma constante, é rapidamente acessado e pode ser utilizado pelo [ambiente de dados](#) para ordenar os registros. O índice na visualização deve ser reconstruído toda vez que a visualização for ativada e demorar mais do que o índice na tabela. Um índice em uma visualização não é parte da definição da visualização; por isso, se você utilizar um ambiente de dados, a visualização não poderá ser o objeto filho, pois o índice do objeto filho deve fazer parte da definição, o que não é suportado por visualizações.

Definindo propriedades de visualização e conexão

Ao ser criada, a visualização herda definições de propriedades, como UpdateType e UseMemoSize,

do cursor do ambiente, ou cursor 0 da [sessão](#) atual. Para alterar essas definições padrão, utilize a função [CURSORSETPROP\(\)](#) com 0 como o número do cursor. Depois de criar a visualização e armazená-la em um banco de dados, você poderá alterar suas propriedades com a função [DBSETPROP\(\)](#). As alterações feitas em propriedades da visualização em um banco de dados são armazenadas de maneira contínua.

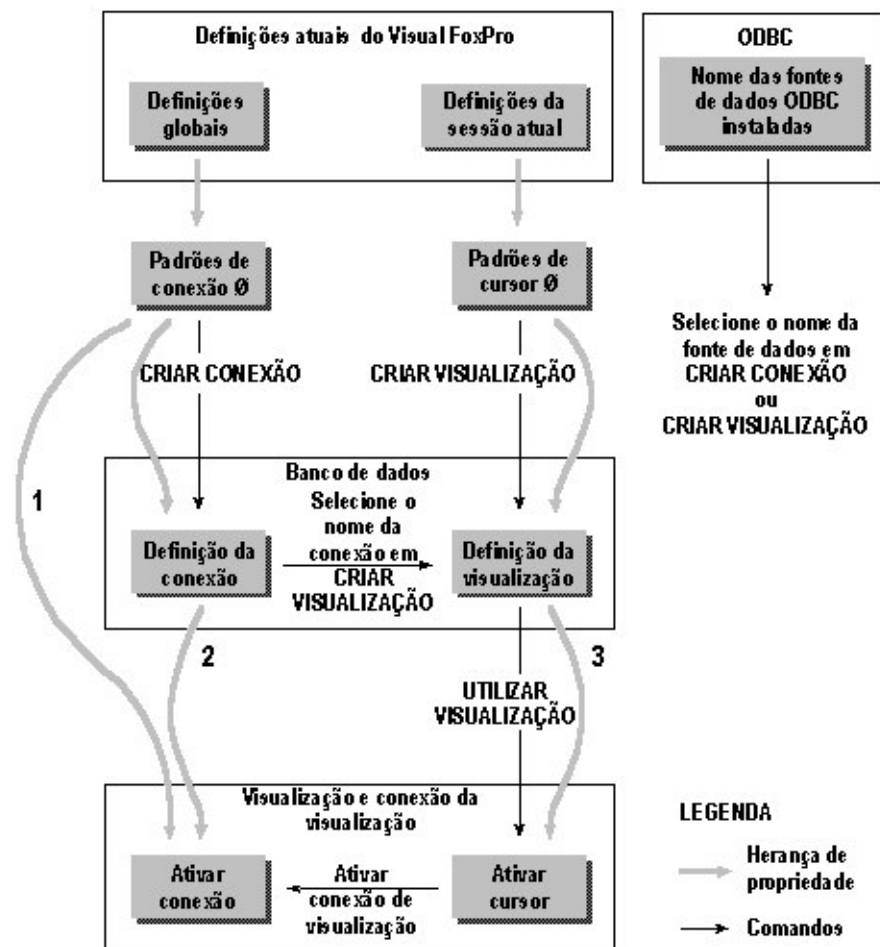
Quando você utiliza uma visualização, as definições de propriedades armazenadas para a visualização no banco de dados são herdadas pelo cursor de visualização ativo. Para alterar estas propriedades no cursor ativo, utilize a função [CURSORSETPROP\(\)](#) para o cursor de visualização. As alterações feitas com a função [CURSORSETPROP\(\)](#) são temporárias. As definições temporárias da visualização ativa desaparecem quando ela é fechada; as definições temporárias do cursor 0 desaparecem quando você fecha a sessão do Visual FoxPro.

As [conexões](#) herdam propriedades de modo semelhante. As propriedades padrão da conexão 0 são herdadas quando uma [conexão definida](#) é criada e armazenada em um banco de dados. Você pode alterar as definições destas propriedades padrão da conexão 0 com a função [SQLSETPROP\(\)](#). Depois de criar e armazenar a conexão em um banco de dados, você pode alterar as suas propriedades com a função [DBSETPROP\(\)](#). Quando uma conexão é utilizada, as definições de propriedades armazenadas para ela no banco de dados são herdadas pela conexão ativa. Para alterar estas propriedades na conexão ativa, utilize a função [SQLSETPROP\(\)](#) para o identificador de conexão.

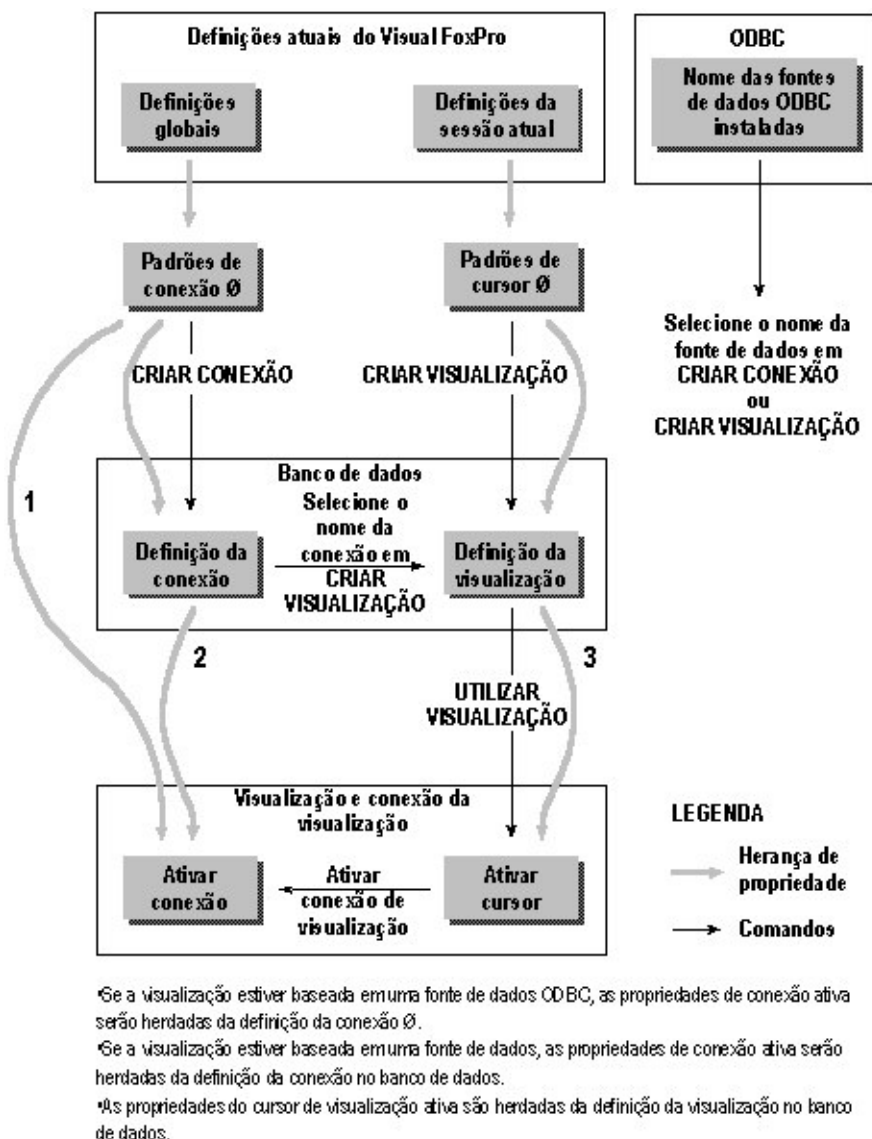
As [visualizações](#) e as [conexões](#) podem utilizar uma [fonte de dados](#) ODBC definida. Se você utilizar uma fonte de dados ODBC em uma visualização, a conexão herdará propriedades dos padrões da [sessão](#).

O diagrama abaixo ilustra a herança de propriedades em visualizações e conexões. As linhas cinzas representam o fluxo da herança de propriedades; as linhas pretas representam comandos do Visual FoxPro.

Propriedades de visualizações e conexões e sua herança



*Se a visualização estiver baseada em uma fonte de dados ODBC, as propriedades de conexão ativa serão herdadas da definição da conexão 0.



Alterando tipos de dados padrão ao transferir visualizações remotas

Quando uma visualização é criada, a propriedade `DataType` para todos os campos na visualização é definida para um valor padrão. O valor é a letra do tipo de dado (D, G, I, L, M, P, T, Y) para tipos de dados de tamanho fixo e a letra seguida por parâmetros de precisão e escala entre parênteses (B(*d*), C(*n*), N(*n*,*d*)) para tipos de tamanho variável. Essa propriedade é somente leitura para visualizações locais. Para obter uma lista dos tipos de dados padrão, consulte “Descarregando e Carregando Dados de Visualizações Remotas” no capítulo 21, [Implementando um aplicativo cliente/servidor](#).

Você pode modificar a definição da propriedade `DataType` para o campo de visualização remota com a função `DBSETPROP()` conforme indicado nesta tabela.

Tipo de dado ODBC do campo remoto	Tipos de dados possíveis no cursor do Visual FoxPro
SQL_CHAR SQL_VARCHAR SQL_LONGVARCHAR	Caractere ou Memo ¹ (padrão); também Geral ou Figura
SQL_BINARY	Memo (padrão); também Caractere, Geral

SQL_VARBINARY SQL_LONGVARBINARY	ou Figura
SQL_DECIMAL SQL_NUMERIC	Numérico ou Moeda ² (padrão); também Caractere, Inteiro ou Duplo
SQL_BIT	Lógico (padrão); também Caractere
SQL_TINYINT SQL_SMALLINT SQL_INTEGER	Inteiro (padrão); também Caractere, Numérico, Duplo ou Moeda
SQL_BIGINT	Caractere (padrão); também Inteiro, Numérico, Duplo ou Moeda
SQL_REAL SQL_FLOAT SQL_DOUBLE	Duplo (padrão); o número de casas decimais é o valor de SET DECIMALS no Visual FoxPro; também Caractere, Inteiro, Numérico ou Moeda
SQL_DATE	Data (padrão); também Caractere ou DataHora
SQL_TIME	DataHora ³ (padrão); também Caractere
SQL_TIMESTAMP	DataHora ⁴ (padrão); também Caractere ou Data

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

2 Se o campo do servidor contiver um tipo de dado de dinheiro, este se tornará um tipo de dado Moeda no Visual FoxPro.

3 A data assume o padrão de 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 DataHora do Visual FoxPro.

Utilizando a propriedade DataType

Você pode utilizar a propriedade DataType para escolher um [tipo de dado](#) diferente do padrão. Por exemplo, convém descarregar um campo de marca de DataHora do servidor para o Visual FoxPro. Entretanto o mapeamento do tipo de dado padrão em um campo de DataHora do Visual FoxPro poderia truncar qualquer fração de segundos armazenada na marca de DataHora do servidor. Você pode utilizar a propriedade DataType para mapear o campo de marca de DataHora remoto em um campo de caractere do Visual FoxPro para preservar as frações de segundos.

Atualizando dados em uma visualização

Você atualiza dados em uma visualização da mesma maneira que atualiza em uma tabela. Com uma visualização, você também pode atualizar as [tabelas base](#) da visualização. Como padrão, as visualizações utilizam buffer de linha otimista. É possível alterar este padrão para utilização de buffer de tabela; para obter maiores informações sobre a utilização de buffer, consulte o capítulo 17, [Programando para acesso compartilhado](#).

Você pode atualizar dados em uma visualização através da interface ou da linguagem. O primeiro passo é tornar a visualização atualizável. Na maioria das vezes, as definições de propriedades padrão preparam automaticamente a visualização para que seja atualizável, mas as atualizações só são enviadas para a fonte de dados quando você instrui o Visual FoxPro para fazê-lo, ativando a propriedade SendUpdates.

Uma visualização utiliza cinco propriedades para controlar atualizações. Estas propriedades encontram-se listadas abaixo com suas definições padrão:

Propriedades de atualização da visualização e suas definições padrão

Propriedade da visualização	Definição padrão
Tables	Inclui todas as tabelas que possuem campos atualizáveis e pelo menos um campo de chave primária.
KeyField	Os campos-chave do banco de dados e as chaves primárias remotas da tabela.
UpdateName	Nome_tabela.nome_coluna de todos os campos.
Updatable	Todos os campos, exceto os campos de chave primária.
SendUpdates	Adota o padrão da sessão, que originalmente está definido como falso (.F.); se você alterá-lo para verdadeiro (.T.), este passará a ser o padrão para todas as visualizações criadas na sessão.
CompareMemo	Adota o padrão de verdadeiro (.T.) e significa que os campos Memo são incluídos na cláusula WHERE e utilizados para detectar conflitos de atualização.

Embora todas as cinco propriedades sejam necessárias para atualizar dados, a propriedade SendUpdates funciona como uma “opção mestre” que controla se as atualizações são ou não enviadas. No desenvolvimento do aplicativo, você poderá desativar a propriedade SendUpdates e, em seguida, configurar as outras propriedades para permitir atualizações nos campos que deseja atualizar. Quando estiver pronto para testar o aplicativo, você poderá ativar a propriedade SendUpdates para iniciar o fluxo das atualizações.

Em determinadas situações mais complexas, as definições de atualizações padrão podem não fornecer atualizações para uma visualização criada através da linguagem. Para permitir atualizações, observe as definições padrão de cada propriedade de atualização e ajuste-as quando necessário. Também é possível especificar outras propriedades, como UpdateType, WhereType, etc., de acordo com as suas preferências. Para obter uma lista completa das propriedades de visualizações, consulte [DBGETPROP\(\)](#).

► Para tornar uma visualização atualizável no Criador de visualizações

- No **Criador de visualizações**, selecione a guia **Critério de atualização** e verifique as definições padrão.

As definições padrão das visualizações que podem ser criadas através do **Criador de visualizações** geralmente preparam a visualização para ser atualizável; você só precisa selecionar a caixa de verificação Enviar Atualizações SQL para ativar as atualizações. Você também pode modificar as opções de atualização, cláusula SQL WHERE, campos e tabelas, conforme desejar.

► Para tornar uma visualização atualizável definindo propriedades de atualização de visualizações

- Examine as definições padrão atuais com o comando [DISPLAY DATABASE](#) e, em seguida, modifique propriedades da definição de visualização, como desejar, com a função [DBSETPROP\(\)](#).

o exemplo abaixo lista os passos que você deve seguir para especificar as cinco propriedades de atualização de visualizações utilizando a linguagem de programação:

Observação As propriedades padrão podem fornecer todas as informações necessárias para atualizar a visualização

- 1 Defina a propriedade Tables com pelo menos um nome de tabela.

Por exemplo, se a visualização está baseada na tabela `customer` denominada `cust_view`, você poderá definir o nome de tabela com a função a seguir:

```
DBSETPROP('cust_view', 'View', 'Tables', 'customer')
```

Dica Se uma tabela aparecer como um qualificador na propriedade UpdateName, mas não estiver incluída na lista padrão da propriedade Tables, ela poderá não possuir um campo de chave primária especificado. Torne a tabela atualizável adicionando o campo que você considera um campo-chave à lista da propriedade KeyField e, em seguida, adicione a tabela à lista da propriedade Tables.

- 2 Defina a propriedade KeyField com um ou mais nomes de campos locais do Visual FoxPro que, juntos, definam uma chave única para a tabela de atualização.

Utilizando o mesmo exemplo, você pode tornar cust_id o campo-chave utilizando o código abaixo:

```
DBSETPROP('cust_view.cust_id','Field','KeyField',.T.)
```

Importante Certifique-se de que o(s) campo(s)-chave especificado(s) define(m) uma chave única na tabela base que você deseja atualizar e na visualização.

- 3 Mapeie os campos da visualização para os respectivos campos da tabela base com a propriedade UpdateName. Esta propriedade é particularmente útil quando a visualização está baseada na associação de duas tabelas com um nome de campo comum ou quando os campos têm aliases na visualização. Para atualizar a tabela base desejada, mapeie o nome do campo da visualização do Visual FoxPro para o nome da tabela e o campo da tabela base.

```
DBSETPROP('cust_view.cust_id','Field','UpdateName',;  
          'customer.cust_id')
```

Dica Para evitar a criação de campos sinônimos na visualização, pode-se qualificar nomes de campo na instrução SQL utilizada para construir a visualização. E, em seguida, utilizar a propriedade UpdateName da visualização para mapear cada campo qualificado para o nome correto de campo e tabela base.

- 4 Especifique o escopo dos campos que deseja atualizar com a propriedade UpdateField. Você deve especificar apenas os campos também especificados com a propriedade UpdateName.

```
DBSETPROP('cust_view.cust_id','Field','Updatable',;  
          .T.)
```

- 5 Defina a propriedade SendUpdates como verdadeira (.T.). Esta é a opção mestre que instrui o Visual FoxPro a criar e enviar atualizações para qualquer tabela e campo especificados como atualizáveis.

```
DBSETPROP('cust_view','View','SendUpdates',.T.)
```

Quando você utiliza **DBSETPROP()** para definir propriedades em uma visualização antes de utilizá-la, as definições são armazenadas no banco de dados e utilizadas automaticamente sempre que a visualização é ativada. Depois de ativar a visualização, você poderá utilizar **CURSORSETPROP()** para alterar as definições de propriedades desta visualização ativa. As definições de propriedades definidas em uma visualização ativa com **CURSORSETPROP()** não são salvas quando você fecha a visualização.

Atualizando várias tabelas em uma visualização

É possível atualizar várias tabelas base em uma visualização. Quando a visualização combinar duas ou mais tabelas, defina propriedades para certificar-se de que somente o lado “muitos” da consulta de visualização é atualizável.

As visualizações são atualizadas tabelas por tabelas. Você deve certificar-se de que para cada tabela acessada em uma visualização, o campo-chave definido é uma chave única, tanto para o conjunto de resultados da visualização como para a tabela base.

► Para tornar uma visualização de várias tabelas atualizável

- No **Criador de visualizações**, selecione a guia **Crêterios de atualização** e, em seguida, selecione as tabelas e os nomes de campos que deseja atualizar.
 - Ou –
- Utilize a função **DBSETPROP()**.

Na maior parte dos casos, os valores padrão fornecidos pelo Visual FoxPro preparam uma

visualização de várias tabelas para ser atualizável, mesmo quando ela é criada através da linguagem de programação. O exemplo de código a seguir cria e define propriedades de maneira explícita para atualizar uma visualização de duas tabelas. Você pode utilizar este exemplo como um guia para personalizar definições de propriedades de atualização em uma visualização.

Atualizando várias tabelas em uma visualização

Código	Comentários
CREATE SQL VIEW emp_cust_view AS ; SELECT employee.emp_id, ; employee.phone, customer.cust_id, ; customer.emp_id, customer.contact, ; customer.company ; FROM employee, customer ; WHERE employee.emp_id = customer.emp_id	Cria uma visualização que acessa campos de duas tabelas.
DBSETPROP('emp_cust_view', 'View', 'Tables', 'employee, customer')	Define as tabelas que deverão ser atualizadas.
DBSETPROP('emp_cust_view.emp_id', 'Field', ; DBSETPROP('emp_cust_view.phone', 'Field', ; DBSETPROP('emp_cust_view.cust_id', 'Field', ; DBSETPROP('emp_cust_view.emp_id1', 'Field', ; DBSETPROP('emp_cust_view.contact', 'Field', ; DBSETPROP('emp_cust_view.company', 'Field', ;	Define nomes de atualização. 'UpdateName', 'emp 'UpdateName', 'emp 'UpdateName', 'customer. 'UpdateName', 'customer. 'UpdateName', 'customer. 'UpdateName', 'customer.
DBSETPROP('emp_cust_view.emp_id', 'Field', ;	Define uma chave única de campo único para a tabela Funcionário. 'KeyField', .T.)
DBSETPROP('emp_cust_view.cust_id', 'Field', ; 'KeyField', .T.) DBSETPROP('emp_cust_view.emp_id1', 'Field', ; 'KeyField', .T.)	Define uma chave única de dois campos para a tabela Cliente.
DBSETPROP('emp_cust_view.phone', 'Field', ; 'UpdatableField', .T.) DBSETPROP('emp_cust_view.contact', 'Field', ; DBSETPROP('emp_cust_view.company', 'Field', ;	Define os campos atualizáveis. Normalmente, campos-chave não são 'UpdatableField', .T.) atualizáveis. 'UpdatableField', .T.)
DBSETPROP('emp_cust_view', 'View', ; 'SendUpdates', .T.)	Ativa a funcionalidade de atualização.
GO TOP REPLACE employee.phone WITH "(206)111-2222" REPLACE customer.contact WITH "John Doe"	Modifica dados na visualização.
TABLEUPDATE()	Grava as alterações atualizando as tabelas base Cliente e Funcionário.

Personalizando visualizações com o dicionário de dados

Visto que as visualizações são armazenadas em um banco de dados, você pode criar:

- [Legendas](#)
- Comentários para a visualização e campos da visualização
- [Valores padrão](#) para campos da visualização
- [Regras](#) de linhas e de campos e mensagens de erro das regras

Os recursos do dicionário de dados para visualizações são funcionalmente semelhantes aos seus correspondentes em tabelas de bancos de dados. No entanto, você utiliza a linguagem e não o **Criador de tabelas** para criar legendas, comentários, valores padrão e regras para visualizações.

Criando valores padrão para campos de visualizações

Da mesma forma que os valores padrão para campos de tabelas, os [valores padrão](#) para campos de visualizações são armazenados no banco de dados e estão disponíveis sempre que a visualização é utilizada. O Visual FoxPro não compara os valores padrão criados localmente com os valores padrão estabelecidos na [fonte de dados](#) remota. Você deve criar valores padrão aceitáveis para a fonte de dados.

► Para atribuir um valor padrão a um campo de visualização

- Na guia **Campos** no **Criador de visualizações**, selecione um campo e, em seguida, escolha **Propriedades** e digite o valor padrão para o campo.
– Ou –
- Utilize a propriedade DefaultValue da função `DBSETPROP()`.

Por exemplo, você pode querer que o aplicativo limite o volume de mercadorias que pode ser encomendado por um novo cliente até que você tenha tido tempo para concluir uma análise de crédito e determinar o volume de crédito que está disposto a estender a esse cliente. O exemplo abaixo cria um campo `maxordamt` com o valor padrão de 1000:

```
OPEN DATABASE testdata
USE VIEW customer_view
```

```
?DBSETPROP ('Customer_view.maxordamt', 'Field', 'DefaultValue', 1000)
```

Você também pode utilizar os [valores padrão](#) para preencher algumas linhas automaticamente para o usuário. Por exemplo, você pode adicionar um [controle Grid](#) a um formulário de entrada de pedidos baseado em uma visualização remota de uma tabela de itens da fila de pedidos. O campo `order_id` é o campo-chave que mapeia cada linha da grade para a linha correspondente na tabela remota de itens da fila de pedidos. Visto que a `order_id` para cada linha da grade será a mesma para um pedido, você pode utilizar um valor padrão para reduzir pressionamentos de teclas, preenchendo o campo `order_id` automaticamente.

Dica Caso uma das regras comerciais do aplicativo exijam que um campo contenha uma entrada, se você fornecer um valor padrão, ajudará a garantir que uma regra específica [em nível de campo](#) ou de [registro](#) não seja violada.

Criando regras em linhas e campos da visualização

Você pode criar versões locais de regras da fonte de dados remota para:

- Reduzir o tempo de resposta.
- Reduzir o impacto nos recursos da rede.
- Testar os dados antes de enviá-los para a fonte de dados remota.
- Evitar o envio de dados com erro para a fonte de dados remota.
- Visual FoxPro não compara as regras criadas localmente com as regras remotas. Você deve criar [regras](#) aceitáveis para a [fonte de dados](#). Caso as regras remotas sejam alteradas, você deve alterar as regras locais de acordo.

► Para criar uma regra em uma linha ou campo da Visualização

- Na guia **Campos** no **Criador de visualizações**, selecione um campo e, em seguida, escolha **Propriedades** e digite a expressão da regra e o texto da mensagem para o campo.
– Ou –
- Utilize as propriedades RuleExpression e RuleText da função `DBSETPROP()`.

Por exemplo, o código abaixo cria uma regra em nível de campo em `orditems_view` que impede a entrada de uma quantidade inferior a 1:

```
OPEN DATABASE testdata
USE VIEW orditems_view
```



```
DBSETPROP('Orditems_view.quantity','Field', ;
           'RuleExpression', 'quantity >= 1')
DBSETPROP('Orditems_view.quantity','Field', ;
           'RuleText', ;
           '"Quantities must be greater than or equal to 1"')
```

Também é possível utilizar a função `DBSETPROP()` para criar regras em nível de linha.

Combinando visualizações

É possível construir uma visualização baseada em outras visualizações. Talvez você queira fazer isso se precisar de um subconjunto das informações disponíveis em várias outras visualizações ou se desejar combinar dados remotos e locais em uma única visualização. Uma visualização baseada em outras visualizações, ou em uma combinação de tabelas locais e visualizações locais ou remotas, é chamada de [visualização de vários níveis](#). A visualização que combina outras visualizações é a visualização de nível superior. Pode haver vários níveis de visualizações entre a visualização de nível superior e as tabelas base remotas ou locais. Ao utilizar uma visualização de vários níveis, as visualizações nas quais a visualização de nível superior está baseada e todas as tabelas base do Visual FoxPro utilizadas nas visualizações de nível superior ou intermediário são exibidas na [janela Sessão de dados](#). As tabelas remotas não são exibidas na janela Sessão de Dados.

Combinando dados locais e remotos em uma visualização

É possível combinar dados remotos e locais em uma visualização, criando uma nova visualização local baseada em uma visualização local e em uma visualização remota.

► Para criar uma visualização que combine dados remotos e locais

- No [Gerenciador de projetos](#), selecione um banco de dados, escolha **Visualizações locais** e selecione **Novo** para abrir o [Criador de visualizações](#). Adicione qualquer combinação de tabelas, visualizações locais e visualizações remotas à sua visualização.
– Ou –
- Utilize o comando `CREATE SQL VIEW`.

Por exemplo, para criar uma visualização local que combine informações da tabela local `Funcionário` e da tabela remota `Pedidos`, você pode utilizar o código abaixo:

```
OPEN DATABASE testdata
CREATE SQL VIEW remote_orders_view ;
    CONNECTION remote_01 ;
    AS SELECT * FROM orders
CREATE SQL VIEW local_employee_remote_orders_view ;
    AS SELECT * FROM testdata!local_employee_view, ;
    testdata!remote_orders_view ;
    WHERE local_employee_view.emp_id = ;
    remote_orders_view.emp_id
```

Atualizando dados locais e remotos em uma visualização

Quando você atualiza dados em uma [visualização de vários níveis](#), as atualizações descem um nível, até a visualização na qual a visualização de nível superior está baseada. Se quiser atualizar as [tabelas base](#) com base nas quais uma visualização de vários níveis é construída, você deve emitir um comando `TABLEUPDATE` para cada visualização na estrutura.

Trabalhando com dados off-line

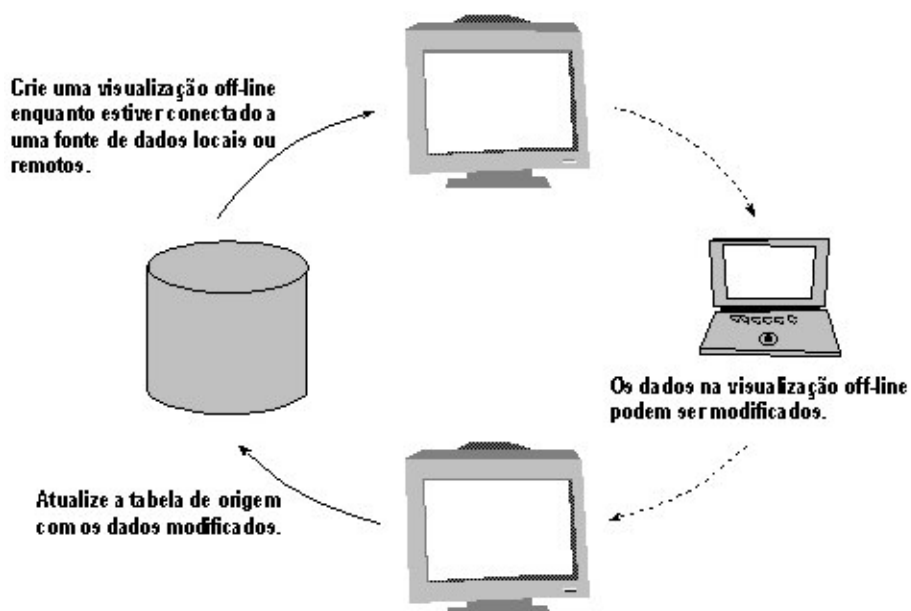
Você talvez queira exibir, reunir ou modificar dados independentemente do banco de dados no computador central. Com a utilização dos recursos de visualização off-line no Visual FoxPro, você pode utilizar as visualizações para se conectar a um banco de dados no computador central e criar um subconjunto de dados para uso off-line. Em seguida, trabalhando off-line, você pode utilizar a

visualização diretamente ou através do aplicativo criado. Quando tiver terminado, você poderá carregar as alterações armazenadas na visualização para o banco de dados no computador central.

Alguns cenários onde as visualizações off-line são úteis incluem:

- Uma situação de armazenamento de dados, onde os grandes bancos de dados são mantidos centralmente em servidores MIS. Se você só estiver interessado em dados relativos ao departamento de Marketing, por exemplo, poderá criar uma visualização incluindo apenas os dados relevantes. Em seguida, poderá colocar os dados off-line, permitir que vários usuários no departamento de Marketing atualizem os dados e depois gravar os dados alterados no banco de dados fonte.
- Um local geograficamente remoto que requer que você utilize um subconjunto de dados em um laptop, modifique os dados independentemente do banco de dados no computador central e, em seguida, atualize o banco de dados no computador central com os dados alterados posteriormente.
- Dados sensíveis à data. Por exemplo, você talvez queira atualizar dados que reflitam aumentos de salário dos funcionários antes que os novos valores entrem em vigor.

Trabalhando com visualizações off-line



Para criar e utilizar dados de visualização off-line, você pode utilizar os recursos de linguagem a seguir:

- A função **CREATEOFFLINE()**
- O comando **USE SQLViewName** com as cláusulas **ADMIN** e **ONLINE**
- A função **TABLEUPDATE**
- A função **DROPOFFLINE()**

Se planeja utilizar a visualização off-line em uma máquina que não seja aquela na qual a visualização foi criada, você deve preparar o destino off-line criando uma cópia do arquivo de banco de dados no computador central (.DBC); certificando-se de que a fonte de dados ODBC utilizada pela visualização existe na máquina de destino e analisando suas exigências de dados para determinar o conteúdo da visualização necessária.

Observação Utilize o programa ODBC Administrator para instalar as fontes de dados em uma máquina. Você pode acessar o programa ODBC Administrator do grupo de programas do Visual FoxPro ou do Painel de Controle.

Criando visualizações off-line

Da mesma forma que com os dados on-line, analise suas exigências antes de criar visualizações off-line para determinar as visualizações necessárias no banco de dados off-line. Depois de saber qual o subconjunto de dados a ser utilizado off-line, você pode começar com uma visualização existente ou criar uma nova visualização. Se houver uma visualização que retorne os registros que você deseja utilizar off-line, você poderá usá-la ou criar uma através da linguagem de programação. A visualização colocada off-line é armazenada em um arquivo .DBF no recipiente do banco de dados off-line.

Observação Se você planeja modificar dados em uma visualização off-line, certifique-se de tornar a visualização atualizável antes de colocá-la off-line. Depois que a visualização estiver off-line, você só pode definir suas propriedades de atualização através da linguagem de programação; não é possível modificar uma visualização off-line no **Criador de visualizações**.

► Para utilizar uma visualização existente off-line

- Utilize a função **CREATEOFFLINE()** e o nome da visualização.

Por exemplo, se quiser ir para locais de cliente para atualizar contas, adicionar clientes e registrar as novas vendas, você precisará das informações do cliente assim como dos pedidos atuais e das descrições do produto on-line. Você pode ter uma visualização denominada `customerinfo` que combina informações das tabelas `Clientes`, `Pedidos` e `OrderItems`. Para criar a visualização, utilize este código:

```
CREATEOFFLINE("customerinfo")
```

► Para criar uma visualização off-line através da linguagem de programação

- Utilize o comando **CREATE SQL VIEW**, seguido pelo comando **CREATEOFFLINE()**.

Por exemplo, o código a seguir cria uma visualização que exibe dados das tabelas `Products` e `Inventory` a partir do banco de dados on-line. Como nenhum critério de atualização está especificado, essa visualização é somente para leitura.

```
CREATE SQL VIEW showproducts ;  
    CONNECTION dsource ;  
    AS SELECT * FROM Products INNER JOIN Inventory ;  
    ON Products.ProductID = Inventory.ProductID ;  
CREATEOFFLINE("showproducts")
```

Exibindo e modificando dados off-line

Depois que você criar a visualização para seus dados off-line, poderá usá-la como qualquer visualização do seu aplicativo: poderá adicionar, alterar e excluir registros. Vários usuários podem acessar a visualização off-line ao mesmo tempo utilizando o mesmo banco de dados em modo compartilhado. Caso você não queira manter as alterações, poderá reverter as informações para refletir as informações originais.

Utilizando dados off-line

Utilizando a visualização off-line, você pode exibir e atualizar dados como na forma on-line com os mesmos formulários, relatórios ou aplicativos. Por exemplo, o código a seguir abre a visualização `Showproducts`:

```
USE Showproducts
```

Dica Se não estiver obtendo o subconjunto de dados esperado, verifique as definições de otimização para a visualização remota. Se você definir a propriedade `MaxRecords` utilizando a função **DBSETPROP()**, apenas esses registros aparecerão nas suas visualizações off-line. No entanto, se você incluir um campo `Memo` na lista de campo da sua visualização, ele será automaticamente incluído no conjunto de resultados mesmo se `FetchMemo` for definido como falso (.F.).

Administrando dados off-line

Em alguns casos — especialmente em ambientes de vários usuários onde os dados são modificados por várias pessoas — você pode querer examinar as alterações feitas à visualização off-line antes de gravar as alterações para o banco de dados fonte. Com o comando **USE** e a cláusula **ADMIN**, você pode ver todas as alterações gravadas em uma visualização desde que foi colocada off-line. Em seguida, pode reverter de forma seletiva as alterações feitas sem estar conectado à fonte de dados. Por exemplo, o código a seguir abre a visualização `Showproducts` no modo administrador:

```
USE Showproducts ADMIN
```

Atualizando dados on-line

Ao terminar de trabalhar off-line, você pode atualizar os dados no servidor utilizando as mesmas transações de atualização de tabela que são normalmente utilizadas com dados on-line. Ao trabalhar com dados remotos, lembre-se das dicas a seguir:

- Para atualizações de um único registro, utilize transações automáticas.
- Para atualizações em lote, utilize transações manuais.
- Conforme necessário, inclua código para detectar conflitos de atualização, crie um registro de conflitos e resolva-os.

Antes de processar suas atualizações, você precisa utilizar o comando **USE** e a palavra-chave **ONLINE** para reconectar-se ao banco de dados no computador central. Depois de emitir o comando, o Visual FoxPro tenta localizar o banco de dados no computador central utilizando as informações de fonte de dados armazenadas na visualização. Depois que a conexão é estabelecida, você pode utilizar **TABLEUPDATE()** para processar as atualizações armazenadas nos dados off-line.

Para certificar-se de que as informações de conexão estão corretas independentemente da localização do computador central ou das tabelas de visualização, você precisa utilizar a sintaxe de sequência de conexão em vez de uma conexão definida.

Atualizando lotes de registros em tabelas locais

Para processar um lote de alterações em tabelas locais, você pode utilizar transações manuais que permitem processar o lote inteiro de alterações com uma única transação em vez de uma série de transações separadas.

Atualizando tabelas locais com visualizações off-line

Código	Comentários
<pre>USE myofflineview ONLINE EXCLUSIVE</pre>	Reconecta ao computador central e abre a visualização
<pre>BEGIN TRANSACTION IF TABLEUPDATE (2, .F., "myofflineview") END TRANSACTION ELSE MESSAGEBOX("Erro ocorreu: Atualização sem sucesso.") ROLLBACK ENDIF</pre>	Verifica os conflitos de atualização e atualiza conforme adequado.

Atualizando lotes de registros em tabelas remotas

Para processar um lote de alterações em tabelas remotas, utilize transações manuais: comece com **TABLEUPDATE()** e termine o processamento com **SQLCOMMIT()** ou **SQLROLLBACK()**.

Para definir a conexão de forma a gerenciar suas transações manualmente, você precisa utilizar **CURSORGETPROP()** no cursor da visualização para obter o identificador de conexão e definir a propriedade **Transactions** para o modo manual.

No código a seguir, a identificação da conexão atual para a visualização, `myview`, é armazenada em `hConn1`. `hConn1` é utilizado para definir a propriedade Transactions como “2” para transações manuais.

```
hConn1 = CURSORGETPROP("CONNECTHANDLE","myview") ;
SQLSETPROP(hConn1,"TRANSACTIONS",2)
```

Depois de definir a conexão para gerenciar as atualizações, você poderá utilizar `TABLEUPDATE()` para manipular as transações.

Se as tabelas no computador central residirem em um servidor remoto, como SQL Server, você talvez utilize o código a seguir como uma diretriz.

Atualizando tabelas remotas com visualizações off-line

Código	Comentários
<code>USE myofflineview ONLINE EXCLUSIVE</code>	Reconecta ao computador central e abre a visualização.
<code>SQLSetProp(liviewhandle,"transactions",2)</code> <code>SQLSetProp(custviewhandle,"transactions",2)</code> <code>SQLSetProp(ordviewhandle,"transactions",2)</code>	Define as conexões nas visualizações para lidar com as transações manualmente.
<code>IF NOT TABLEUPDATE(.T.,.F.,"lineitemsview")</code> <code>=SQLROLLBACK(ordviewhandle)</code> <code>=MESSAGEBOX("Can't update line items table")</code> <code>IF NOT TableUpdate(.T.,.F.,"ordersview")</code> <code>=SQLROLLBACK(liviewhandle)</code> <code>=MESSAGEBOX("unable to update the orders table")</code> <code>IF NOT TABLEUPDATE(.T.,.F.,"customerview")</code> <code>=SQLROLLBACK(custviewhandle)</code> <code>=MESSAGEBOX("Can't update customer table")</code> <code>Else *# check out failure scenarios</code> <code>IF NOT SQLCOMMIT(liviewhandle)</code> <code>=SQLROLLBACK(liviewhandle)</code> <code>IF NOT SQLCOMMIT(ordviewhandle)</code> <code>=SQLROLLBACK(ordviewhandle)</code> <code>IF NOT SQLCOMMIT(custviewhandle)</code> <code>=SQLROLLBACK(custviewhandle)</code> <code>ENDIF</code> <code>ENDIF</code> <code>ENDIF</code> <code>ENDIF</code> <code>ENDIF</code> <code>ENDIF</code>	Gerencia atualizações e conflitos de atualização.

Atualizando um registro

Se estiver atualizando uma única linha, poderá utilizar transações automáticas. Como cada instrução para processar uma atualização, exclusão ou inserção é gerenciada como uma transação separada, não é possível fazer rollbacks das instruções de transação anteriores.

```
USE customerview ONLINE EXCLUSIVE
GO TO 3
    IF TABLEUPDATE (0, .F. workarea)
        * conflict handling code
    ENDIF
```

Dica Para atualizar um único registro em uma tabela local, utilize a função `GETNEXTMODIFIED()`.

Cancelando atualizações off-line

Caso você queira excluir os dados off-line e converter a visualização em uma visualização on-line, poderá utilizar a função `DROPOFFLINE()`.

► Para cancelar atualizações off-line

- Utilize `DROPOFFLINE()` com o nome da visualização.

Certifique-se de verificar os valores de retorno. Verdadeiro (.T.) indica sucesso e falso (.F.) indica que a visualização não foi fechada antes da emissão do comando.

O código a seguir abandona todas as alterações feitas ao subconjunto de dados em `myview`. A visualização mantém parte do banco de dados, mas seu conjunto de dados atual é abandonado.

```
DROPOFFLINE("myview")
```

Você pode excluir registros off-line, mas não pode utilizar os comandos `PACK`, `ZAP` ou `INSERT` com uma visualização off-line.

Otimizando o desempenho das visualizações

É possível otimizar o desempenho das visualizações definindo propriedades de visualização.

Controlando o tamanho da carga no carregamento progressivo

É possível controlar o número de linhas que o Visual FoxPro carrega progressivamente por vez do banco de dados no computador central, utilizando a propriedade `FetchSize` da visualização e do cursor de visualização ativo. Utilize `DBSETPROP()` e `CURSORSETPROP()` para definir estas propriedades.

Controlando a carga de memo

É possível utilizar o recurso de carga de memo retardada para acelerar a recuperação dos dados da visualização. Quando você seleciona carga de memo retardada, o Visual FoxPro não recupera o conteúdo de um campo Memo até você abrir e exibir o campo. Visto que o Visual FoxPro precisa do nome da tabela e do campo-chave para localizar uma linha na fonte de dados remota, você deve definir a propriedade `UpdateName` ou `UpdatableFieldList`, a propriedade `KeyField` ou `KeyFieldList` e a propriedade `Tables` para que a carga de memo retardada funcione. No entanto, não é necessário ativar as propriedades `SendUpdates` ou `Updatable` para que a carga de memo retardada funcione.

Definindo o número máximo de registros transferidos

Você pode controlar o volume de dados transferidos quando uma visualização é aberta definindo a propriedade `MaxRecords`. Quando o Visual FoxPro envia uma instrução SQL para a fonte de dados para criar uma visualização, a fonte de dados constrói e armazena um conjunto de resultados. A propriedade `MaxRecords` especifica o número máximo de linhas carregadas do conjunto de resultados remoto para a visualização. A definição padrão é `-1`, que transfere todas as linhas do conjunto de resultados.

► Para controlar o número de linhas transferidas para uma visualização

- No menu **Ferramentas**, selecione **Opções** e selecione a guia **Dados remotos**. Na área **Padrões de visualização remota**, ao lado da caixa **Máximo de registros a carregar**, desmarque **Todos**, digite um valor na caixa de texto e, em seguida, selecione **OK**.

– Ou –

- Utilize a propriedade `MaxRecords` da função `DBSETPROP()` ou `CURSORSETPROP()`.

Por exemplo, o código abaixo altera a definição de visualização para limitar em 50 o número de linhas transferidas para a visualização, independente do tamanho do conjunto de resultados construído com base na fonte de dados remota:

```
OPEN DATABASE testdata
USE VIEW remote_customer_view
?DBSETPROP('Remote_customer_view', ; 'View', 'MaxRecords', 50)
```

É possível utilizar a função `CURSORSETPROP()` para definir o limite `MaxRecords` para uma

visualização ativa.

Dica Não é possível utilizar a propriedade MaxRecords para parar uma consulta disparada, pois a propriedade MaxRecords não controla a construção do conjunto de resultados. Utilize a propriedade QueryTimeout para controlar o tempo de execução na fonte de dados remota.

Otimizando filtros e associações

Para tomar decisões de otimização para uma visualização ou consulta, você precisará conhecer o [plano de execução](#): a ordem em que as [associações](#) e cláusulas de [filtro](#) serão avaliadas. Utilizando a função `SYS(3054)`, você pode exibir um dos três níveis de otimização [Rushmore](#). Os três níveis indicam o grau em que as condições de filtro ou associação foram capazes de utilizar a otimização Rushmore. Os níveis são: completamente (Completo), parcialmente (Parcial) ou nenhum (Nenhum).

► Para exibir o plano de execução para filtros

1 Na janela **Comando**, digite `SYS(3054,1)` para ativar SQL ShowPlan.

2 Digite a instrução SQL SELECT.

Por exemplo, você pode digitar:

```
SELECT * FROM customer, orders ;  
AND Upper(country) = "MEXICO"
```

3 Leia o plano de execução na tela.

Para este exemplo, a tela pode exibir:

```
Using Index Tag Country to optimize table customer  
Rushmore Optimization Level for table customer: Full  
Rushmore Optimization level for table orders: none
```

4 Na janela **Comando**, digite `SYS(3054,0)` para desativar SQL ShowPlan.

Em seguida, você pode passar 11 para a função SYS para avaliar as associações nas cláusulas FROM ou WHERE.

► Para exibir o plano de execução para associações

1 Na janela **Comando**, digite `SYS(3054,11)` para ativar SQL ShowPlan.

2 Digite a instrução SQL SELECT.

Por exemplo, você pode digitar:

```
SELECT * ;  
FROM customer INNER JOIN orders ;  
ON customer.cust_id = orders.cust_id ;  
WHERE Upper(country) = "MEXICO"
```

3 Leia o plano de execução na tela.

Para este exemplo, a tela pode exibir:

```
Using Index Tag Country to optimize table customer  
Rushmore Optimization Level for table customer: Full  
Rushmore Optimization level for table orders: none  
Joining table customer and table orders using Cust_id
```

4 Na janela **Comando**, digite `SYS(3054,0)` para desativar SQL ShowPlan.

Controlando a avaliação da associação

Se o [plano de execução](#) para as suas [associações](#) não corresponder às suas necessidades específicas, você poderá forçar a ordem de associação para executar exatamente conforme escrita sem otimização do processador. Para forçar a ordem de avaliação da associação, você precisa adicionar a palavra-chave FORCE e colocar as condições de associação na cláusula FROM. As condições de associação colocadas na cláusula WHERE não estão incluídas em uma avaliação de associação forçada.

Observação Você não pode utilizar a palavra-chave FORCE em instruções de passagem SQL ou visualizações remotas porque essa palavra-chave é uma extensão do Visual FoxPro do padrão [ANSI](#)

e não é suportada em outros dicionários SQL.

A cláusula **FORCE** é global e, portanto, aplica-se a todas as tabelas na cláusula **JOIN**. Certifique-se de que a ordem na qual as tabelas de associação aparecem seja exatamente a ordem na qual devem ser associadas. Também é possível utilizar parênteses para controlar a ordem de avaliação das associações.

Neste exemplo, a primeira associação especificada também é a primeira associação avaliada. A tabela **Clientes** é associada primeiro com a tabela **Pedidos**. O resultado desta associação é em seguida associado com a tabela **OrdItems**.

```
SELECT * ;  
FROM FORCE Customers ;  
INNER JOIN Orders ;  
ON Orders.Company_ID = Customers.Company_ID ;  
INNER JOIN OrdItems;  
ON OrdItems.Order_NO = Orders.Order_NO
```

Neste exemplo, a associação entre parênteses para as tabelas **Pedidos** e **OrdItems** é avaliada primeiro. O resultado desta associação é utilizado na avaliação da associação com **Customers**.

```
SELECT * ;  
FROM FORCE Customers ;  
INNER JOIN (orders INNER JOIN OrdItems ;  
ON OrdItems.Order_No = Orders.Order_No) ;  
ON Orders.Company_ID = Customers.Company_ID
```

Compartilhando conexões para várias visualizações remotas

Você pode utilizar uma conexão ativa como canal de informações para várias visualizações remotas, compartilhando uma [conexão](#). Ao compartilhar uma conexão ativa, você:

- Reduz o número de conexões em um servidor remoto.
- Reduz os custos de conexões com servidores que cobram por conexão.

Para compartilhar conexões, estabeleça a definição de visualização de modo que utilize uma conexão compartilhada ao ser ativada. Quando a visualização é utilizada, o Visual FoxPro conecta-se à fonte de dados remota utilizando a conexão compartilhada existente (se houver uma). Caso uma conexão compartilhada não esteja em uso, quando a visualização for aberta, o Visual FoxPro criará uma conexão exclusiva que, depois, poderá ser compartilhada com outras visualizações.

Apenas uma instância ativa de uma determinada definição de conexão é compartilhada durante uma sessão do Visual FoxPro. Caso várias instâncias da mesma definição de conexão estejam ativas, a primeira instância a ser utilizada como uma conexão compartilhada se tornará a conexão compartilhada designada. Todas as visualizações que utilizem essa definição de conexão e empreguem o compartilhamento de conexão irão acessar o servidor remoto através da conexão compartilhada designada.

As demais conexões não serão compartilhadas. O compartilhamento de conexão não se aplica ao escopo de [sessões](#).

► Para compartilhar uma conexão

- No menu **Ferramentas**, selecione **Opções** e selecione a guia **Dados remotos**, em seguida, selecione a caixa de verificação **Compartilhar conexão** na área **Padrões de visualização remota** e selecione **OK**.
 - Ou –
- Utilize o **Criador de visualizações**.
 - Ou –
- Utilize o comando **CREATE SQL VIEW** com a cláusula **SHARE**.

O código abaixo cria uma visualização que, quando ativada com o comando **USE**, compartilha uma

conexão:

```
CREATE SQL VIEW product_view_remote ;  
    CONNECTION remote_01 SHARE AS ;  
    SELECT * FROM products  
USE product_view_remote
```

Testando se uma conexão está ocupada

Quando uma conexão está ocupada, como quando o Visual FoxPro está carregando progressivamente dados em um [cursor](#), não é recomendável iniciar uma outra carga ou enviar atualizações na mesma conexão. Para determinar se a conexão está ocupada, utilize a propriedade ConnectBusy, que retornará um valor verdadeiro (.T.) se a conexão estiver ocupada. Você pode utilizar esta propriedade no aplicativo para testar se uma conexão está ocupada antes de enviar um pedido através de uma conexão compartilhada para uma fonte de dados remota.

► Para determinar se uma conexão está ocupada

- Utilize a propriedade ConnectBusy da função [SQLGETPROP\(\)](#).

Para utilizar a função SQLGETPROP(), você precisa do identificador de conexão. Para definir o identificador de conexão de uma visualização ativa, utilize a propriedade ConnectHandle da função [CURSORGETPROP\(\)](#). O código abaixo define um identificador de conexão e, em seguida, utiliza esse identificador para testar se uma conexão está ocupada:

```
nConnectionHandle=CURSORGETPROP('ConnectHandle')  
SQLGETPROP(nConnectionHandle, "ConnectBusy")
```