

A construção de seu [banco de dados](#) envolve a criação de [tabelas](#). Ao projetar o banco de dados, você especificou os [campos](#) e relacionamentos necessários para seu aplicativo. Agora, à medida que as tabelas forem criadas, você fará escolhas mais detalhadas quanto aos tipos de dados, legendas e prováveis valores padrão para cada campo, os disparadores para cada tabela, além dos índices de tabela que serão criados para estabelecer os relacionamentos entre tabelas. Este capítulo descreve o processo criação, refinamento e relacionamento de tabelas e índices durante o desenvolvimento de um aplicativo. Ele se concentra basicamente na utilização da linguagem para trabalhar com tabelas e registros, porém explica também como utilizar a interface para lidar com tarefas comuns.

Este capítulo aborda os tópicos a seguir:

- [Criando tabelas](#)
- [Trabalhando com registros](#)
- [Indexando tabelas](#)
- [Utilizando várias tabelas](#)

## Criando tabelas

É possível criar uma tabela em um banco de dados ou uma tabela livre que não esteja associada a um banco de dados. Se você colocar a tabela em um banco de dados, é possível criar nomes extensos de tabela e de campos para tabelas de banco de dados. Também é possível utilizar os recursos de [dicionário de dados](#) para tabelas de banco de dados, nomes extensos de campos, valores padrão de campos, regras de [campos](#) e [registros](#), além de [disparadores](#).

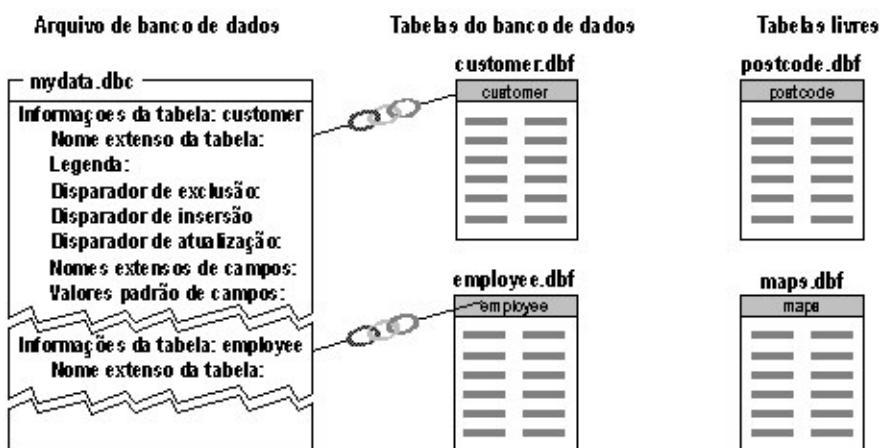
## Estruturando tabelas de banco de dados ou tabelas livres

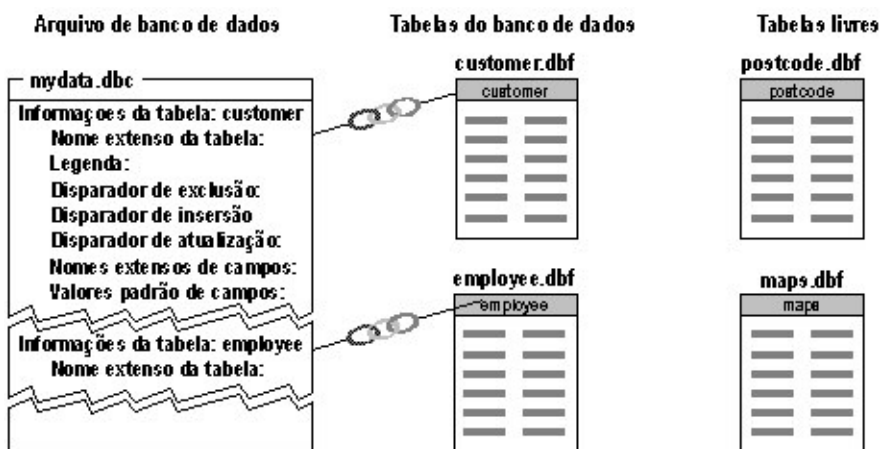
Uma tabela do Visual FoxPro, ou arquivo .DBF, pode existir em um dos estados a seguir: como uma tabela de banco de dados (uma tabela associada a um banco de dados) ou como uma tabela livre que não esteja associada a qualquer banco de dados. As tabelas associadas a um banco de dados possuem diversas vantagens em relação às tabelas livres. Quando uma tabela é parte de um banco de dados, é possível criar:

- Nomes extensos para a tabela e seus campos.
- [Legendas](#) e comentários para cada campo da tabela.
- [Valores padrão](#), [máscaras de entrada](#) e formatos para campos da tabela.
- [Classe de controle](#) padrão para campos de tabela.
- Regras de [campos](#) e de [registros](#).
- [Índices](#) de chave primária e relacionamentos de tabela para suportar regras de [integridade referencial](#).
- Um [disparador](#) para cada evento INSERT, UPDATE ou DELETE.

Alguns recursos podem ser utilizados somente com tabelas de banco de dados. Para obter informações sobre como associar tabelas a um banco de dados, consulte o capítulo 6, [Criando bancos de dados](#)

### As tabelas de um banco de dados têm propriedades que as tabelas livres não têm





É possível projetar e criar uma tabela de forma interativa utilizando o **Criador de tabelas**, que pode ser acessado a partir do **Gerenciador de projetos** ou do menu **Arquivo**. Também é possível criar uma tabela utilizando a linguagem de programação. Esta seção descreve principalmente como criar uma tabela utilizando a linguagem. Para obter informações sobre como utilizar o **Criador de tabelas** para criar tabelas de forma interativa, consulte o capítulo 2, **Criando tabelas e índices**, no *Guia do Usuário*.

Os comandos a seguir podem ser utilizados para criar e editar uma tabela utilizando a linguagem de programação:

#### Comandos para criar e editar tabelas

ALTER TABLE	CLOSE TABLES
CREATE TABLE	DELETE FILE
REMOVE TABLE	RENAME TABLE
DROP TABLE	

### Criando uma tabela de banco de dados

Você pode criar uma nova tabela em um banco de dados utilizando o sistema de menus, o **Gerenciador de projetos** ou a linguagem. Ao criar a tabela, você pode criar nomes extensos de tabela e de campos, valores padrão de campos, regras de campos e de registros, além de disparadores.

#### ► Para criar uma nova tabela de banco de dados

- No **Gerenciador de projetos**, selecione um banco de dados, depois **Tabelas**, depois **Novo** para abrir o **Criador de tabelas**.  
– Ou –
- Utilize o comando **CREATE TABLE** quando um banco de dados estiver aberto.

Por exemplo, o código abaixo cria a tabela `smalltbl` com um coluna, denominada `name`:

```
OPEN DATABASE Sales
CREATE TABLE smalltbl (name c(50))
```

A nova tabela será associada automaticamente ao banco de dados que estiver aberto quando ela for criada. Essa associação é definida por um [vínculo retroativo](#) armazenado no registro de cabeçalho da tabela.

### Criando uma tabela livre

Uma tabela livre é uma tabela que não está associada a um banco de dados. As tabelas livres são úteis, por exemplo, para armazenar informações de pesquisa compartilhadas por muitos bancos de dados.

#### ► Para criar uma nova tabela livre

- No **Gerenciador de projetos**, selecione **Tabelas livres** e, em seguida, **Novo** para abrir o **Criador de tabelas**.
  - Ou –
- Utilize a palavra-chave **FREE** com o comando **CREATE TABLE**.

Por exemplo, o código abaixo cria a tabela livre `smalltbl` com uma coluna, denominada `name`:

```
CLOSE DATABASES  
CREATE TABLE smalltbl FREE (name c(50))
```

Se nenhum banco de dados estiver aberto no momento em que você criar a tabela, não será necessário utilizar a palavra-chave **FREE**.

## Nomeando uma tabela

Ao emitir o comando **CREATE TABLE**, você especifica o nome de arquivo para o arquivo `.DBF` criado pelo Visual FoxPro para armazenar sua nova tabela. Esse nome de arquivo é o nome padrão da tabela tanto para tabelas de banco de dados quanto para tabelas livres. Os nomes de tabela podem conter letras, dígitos ou caracteres de sublinhado e devem começar com uma letra ou caractere de sublinhado.

Caso sua tabela esteja em um banco de dados, também será possível especificar um nome extenso para a tabela. Os nomes de tabela extensos podem conter até 128 caracteres e podem ser utilizados no lugar de nomes de arquivo reduzidos para identificar a tabela no banco de dados. O Visual FoxPro exibe os nomes extensos de tabela, caso tenham sido definidos, sempre que uma tabela é exibida na interface, por exemplo no **Gerenciador de projetos**, no **Criador de bancos de dados**, no **Criador de consultas** e no **Criador de visualizações**, assim como na barra de título de uma janela **Pesquisar**.

#### ► Para atribuir um nome extenso a uma tabela

- No **Criador de tabelas**, digite um nome extenso na caixa de texto **Nome da tabela**.
  - Ou –
- Utilize a cláusula **NAME** do comando **CREATE TABLE**.

Por exemplo, o código abaixo cria a tabela `vendintl` e atribui a ela um nome mais fácil de entender, `vendors_international`:

```
CREATE TABLE vendintl NAME vendors_international (company C(40))
```

Também é possível utilizar o **Criador de tabelas** para renomear tabelas ou adicionar nomes extensos a tabelas que foram criadas sem eles. Por exemplo, quando você adicionar uma tabela livre a um banco de dados, é possível utilizar o **Criador de tabelas** para adicionar um nome extenso à tabela. Os nomes extensos podem conter letras, dígitos ou caracteres de sublinhado e devem começar com uma letra ou caractere de sublinhado. Você não pode utilizar espaços em nomes de tabela extensos.

## Renomeando uma tabela

Você pode renomear tabelas de bancos de dados através da interface porque está alterando o nome extenso. Se remover a tabela do banco de dados, o nome de arquivo para a tabela retém o nome original. As tabelas livres não possuem um nome extenso e somente podem ser renomeadas utilizando a linguagem.

#### ► Para renomear uma tabela em um banco de dados

- 1 No **Criador de bancos de dados**, selecione a tabela a ser renomeada.
- 2 No menu **Banco de dados**, selecione **Modificar**.
- 3 No **Criador de tabelas**, digite um novo nome para a tabela na caixa **Nome da tabela** na guia **Tabela**.

### ► Para renomear uma tabela livre

- Utilize o comando [RENAME](#).

---

**Cuidado** Se você utilizar o comando RENAME em tabelas associadas a um banco de dados, o comando não atualizará o [vínculo retroativo](#) no banco de dados e poderá causar erros de acesso à tabela.

---

## Excluindo uma tabela de banco de dados

Se uma tabela estiver associada a um banco de dados, é possível excluir a tabela ao mesmo tempo em que você a remove do banco de dados. Entretanto, excluir uma tabela não é o mesmo que removê-la de um banco de dados. Caso você queira somente remover uma tabela de um banco de dados, mas não queira excluí-la do disco fisicamente, consulte “Removendo uma tabela de um banco de dados” no capítulo 6, [Criando bancos de dados](#)

### ► Para excluir do disco uma tabela de um banco de dados

- No [Gerenciador de projetos](#), selecione o nome da tabela, escolha **Remover** e depois **Excluir**.  
– Ou –
- A partir do [Criador de bancos de dados](#), selecione a tabela, escolha **Remover** no menu **Banco de dados** e depois escolha **Excluir**.  
– Ou –
- Para excluir a tabela e todos os índices primários, valores padrão e regras de validação associados à tabela, utilize o comando [DROP TABLE](#).  
– Ou –
- Para excluir somente o arquivo de tabela (.DBF), utilize o comando [ERASE](#).

---

**Cuidado** Se você utilizar o comando ERASE em tabelas associadas a um banco de dados, o comando não atualizará o [vínculo retroativo](#) no banco de dados e poderá causar erros de acesso à tabela.

---

Por exemplo, o código a seguir abre o banco de dados `testdata` e exclui a tabela `orditems` e seus índices, valores padrão e regras de validação.

```
OPEN DATABASE testdata
DROP TABLE orditems
```

Se a tabela for excluída com a cláusula DELETE do comando [REMOVE TABLE](#), o arquivo de memo (.FPT) e o arquivo de índice estrutural .CDX associados também serão removidos.

## Excluindo uma tabela livre

Se uma tabela não estiver associada a um banco de dados, é possível excluir o arquivo da tabela por meio do **Gerenciador de projetos** ou utilizando o comando DELETE FILE.

### ► Para excluir uma tabela livre

- No [Gerenciador de projetos](#), selecione a tabela livre, escolha **Remover** e depois **Excluir**.  
– Ou –
- Utilize o comando [DELETE FILE](#).

Por exemplo, se `sample` for a tabela atual, o código a seguir fechará a tabela e excluirá o arquivo do disco:

```
USE
DELETE FILE sample.dbf
```

O arquivo que você deseja excluir não pode estar aberto quando o comando DELETE FILE for emitido. Caso você exclua uma tabela que tenha um arquivo de memo (.FPT) ou arquivos de índice (.CDX ou .IDX), lembre-se de excluir também esses arquivos. Por exemplo, se o arquivo

sample.dbf tivesse um arquivo de memo associado, você poderia excluir ambos com os comandos a seguir:

```
USE
DELETE FILE sample.dbf
DELETE FILE sample.fpt
```

## Duplicando uma tabela

Você pode criar uma cópia de uma estrutura de tabelas, seus [procedimentos armazenados](#), expressões de [disparadores](#) e valores de campo padrão utilizando a linguagem. Não há opção de menu para executar a mesma função. Esse procedimento não copia o conteúdo da tabela.

### ► Para duplicar uma tabela

- 1 Abra a tabela original.
- 2 Utilize o comando [COPY STRUCTURE](#) para fazer uma cópia da tabela original.
- 3 Abra a tabela vazia criada com o comando COPY STRUCTURE.
- 4 Utilize o comando [APPEND FROM](#) para copiar os dados da tabela original.

## Copiando e editando estruturas de tabela

Para modificar a estrutura de uma tabela existente, você pode utilizar o [Criador de tabelas](#) ou [ALTER TABLE](#). Como alternativa, você pode criar uma nova tabela com base na estrutura de uma tabela existente e, em seguida, modificar a estrutura da tabela nova.

### ► Para copiar e editar uma estrutura de tabela

- 1 Abra a tabela original.
- 2 Utilize o comando [COPY STRUCTURE EXTENDED](#) para produzir uma nova tabela contendo as informações estruturais da tabela antiga.
- 3 Edite a nova tabela contendo as informações estruturais para alterar a estrutura de qualquer tabela nova criada a partir dessas informações.
- 4 Cria uma nova tabela utilizando o comando [CREATE FROM](#).  
A nova tabela está vazia.
- 5 Utilize [APPEND FROM](#) ou um dos comandos de cópia de dados para preencher a tabela se necessário.

## Criando campos

Ao criar os campos da tabela, você determina como os dados são identificados e armazenados na tabela especificando um nome de campo, um tipo de dado e uma largura para o campo. Você pode também controlar que dados são permitidos no campo especificando se o campo permite [valores nulos](#), se possui um valor padrão ou deve atender às regras de validação. A definição das propriedades de exibição permite que você especifique o tipo de controle de formulário criado quando o campo é adicionado a um [formulário](#), o formato para o conteúdo dos [campos](#) ou a [legenda](#) que rotula o conteúdo do campo.

**Observação** As tabelas no Visual FoxPro podem conter até 255 campos. Se um ou mais campos contiverem valores nulos, o número máximo de campos que a tabela poderá conter será reduzido em um, de 255 para 254.

## Nomeando campos

Os nomes dos campos são especificados quando você cria uma nova tabela. Esses nomes podem ter até 10 caracteres para tabelas livres ou 128 caracteres para tabelas de banco de dados. Caso você remova uma tabela de um banco de dados, os nomes de campo extensos da tabela serão truncados para 10 caracteres.

### ► Para nomear um campo de tabela

- No **Criador de tabelas**, digite um nome de campo na caixa **Nome**.  
– Ou –
- Utilize o comando **CREATE TABLE** ou o comando **ALTER TABLE**.

Por exemplo, para criar e abrir a tabela `customer` com três campos, `cust_id`, `company` e `contact`, você utilizaria o comando a seguir:

```
CREATE TABLE customer (cust_id C(6), company C(40), contact C(30))
```

No exemplo acima, o `C(6)` significa um campo com dados de Caractere e largura 6. A escolha dos tipos de dados para os campos é abordada posteriormente nesta seção.

Utilizando o comando **ALTER TABLE**, você adiciona os campos `company` e `contact` a uma tabela existente denominada `customer`:

```
ALTER TABLE customer ;  
    ADD COLUMN (company C(40), contact C(30))
```

### Utilizando nomes de campos reduzidos

Quando você cria uma tabela em um banco de dados, o Visual FoxPro armazena os nomes extensos referentes aos campos da tabela em um registro do arquivo `.DBC`. Os primeiros 10 caracteres do nome extenso também são armazenados no arquivo `.DBF` como o nome do campo.

Se os primeiros 10 caracteres do nome de campo extenso não forem únicos nesta tabela, o Visual FoxPro gerará um nome equivalente aos primeiros *n* caracteres do nome extenso acrescidos de um número seqüencial, de tal forma que o nome do campo contenha 10 caracteres. Por exemplo, estes nomes de campo extensos serão convertidos para os nomes de 10 caracteres a seguir:

Nome extenso	Nome reduzido
customer_contact_name	customer_c
customer_contact_address	customer_2
customer_contact_city	customer_3
...	...
customer_contact_fax	customer11

Enquanto uma tabela estiver associada a um banco de dados, você deve utilizar os nomes de campo extensos para fazer referência aos campos da tabela. Não é possível utilizar os nomes de campo de 10 caracteres para fazer referência aos campos de uma tabela em um banco de dados. Caso você remova uma tabela de seu banco de dados, os nomes extensos serão perdidos e você deverá utilizar os nomes de 10 caracteres (armazenados no `.DBF`) como nomes de campo.

É possível utilizar nomes de campo extensos compostos de caracteres, não números, nos arquivos de [índice](#). Contudo, caso você crie um índice utilizando nomes extensos e depois remova do banco de dados a tabela à qual ele se refere, seu índice não funcionará. Nesse caso, é possível reduzir os nomes no índice e depois recriar o índice ou excluí-lo e recriá-lo utilizando nomes de campo reduzidos. Para obter informações sobre como excluir um índice, consulte [Excluindo um índice](#), posteriormente neste capítulo.

As regras para criar nomes de campo extensos são as mesmas utilizadas para criar qualquer identificador do Visual FoxPro, exceto pelo fato de que os nomes podem conter até 128 caracteres. Para obter maiores informações sobre como nomear identificadores do Visual FoxPro, consulte [Criando nomes no Visual FoxPro](#).

### Selecionando tipos de dados

Ao criar um campo da tabela, você também escolhe um tipo de dado para os dados que o campo armazenará. Quando escolhe o tipo de dado de um campo, você está decidindo:

- Que tipos de valores serão permitidos no campo. Por exemplo, não é possível armazenar texto

em um campo [Numérico](#).

- Quanto espaço de armazenamento o Visual FoxPro deverá reservar para os valores armazenados naquele campo. Por exemplo, qualquer valor com o [tipo de dado Moeda](#) utiliza 8 bytes de armazenamento.
- Que tipos de operações poderão ser executados sobre os valores daquele campo. Por exemplo, o Visual FoxPro pode somar valores do tipo Numérico ou Moeda, mas não valores do tipo [Caractere](#) ou [Geral](#).
- Se o Visual FoxPro poderá indexar ou classificar os valores no campo. Você não pode classificar nem criar um índice para campos [Memo](#) ou [Geral](#).

**Dica** Para números de telefone, números de peças e outros números que você não pretende utilizar para cálculos matemáticos, selecione o tipo de dados [Caractere](#), em vez de [Numérico](#).

#### ► Para selecionar um tipo de dado para um campo

- No [Criador de tabelas](#), escolha um tipo de dado na lista **Tipo**.  
– Ou –
- Utilize o comando [CREATE TABLE](#).

Por exemplo, para criar e abrir a tabela `products` com três campos, `prod_id`, `prod_name` e `unit_price`, você poderia utilizar o comando a seguir:

```
CREATE TABLE products (prod_id C(6), prod_name C(40), unit_price Y)
```

No exemplo acima, o 'Y' depois do nome do campo `unit_price` especifica um tipo de dado [Moeda](#).

Para obter maiores informações sobre tipos de dados específicos, consulte [Tipos de dados e campo](#).

### Adicionando um índice regular rapidamente

À medida que você adiciona um campo, poderá definir rapidamente um [índice regular](#) no campo especificando ascendente ou descendente na coluna Índice do [Criador de tabelas](#). O índice criado é automaticamente adicionado à guia **Índice** e utiliza o campo como a expressão. Para modificar o índice, você pode alternar para a guia **Índice** para alterar o nome e o tipo do índice ou adicionar um [filtro](#).

### Utilizando valores nulos

Ao criar uma nova tabela, você pode especificar se um ou mais campos da tabela aceitarão valores nulos. Ao utilizar um valor nulo, você está registrando o fato de que uma informação que normalmente estaria armazenada em um campo ou registro não está disponível no momento. Por exemplo, os benefícios de assistência médica ou o valor dos impostos de um funcionário podem ainda não ter sido determinados na hora em que um registro é preenchido. Em vez de armazenar um zero ou um valor em branco, que poderiam ser interpretados como tendo algum sentido, você poderia armazenar um valor nulo no campo até que a informação em questão estivesse disponível.

#### ► Para controlar a entrada de valores nulos para cada campo

- No [Criador de tabelas](#), selecione ou limpe a coluna **Nulo** para o campo.  
Quando a coluna **Nulo** está selecionada, é possível inserir valores nulos no campo.  
– Ou –

- Utilize as cláusulas `NULL` e `NOT NULL` do comando [CREATE TABLE](#).

Por exemplo, o comando a seguir cria e abre uma tabela que não permite valores nulos nos campos `cust_id` e `company` mas permite valores nulos no campo `contact`:

```
CREATE TABLE customer (cust_id C(6) NOT NULL, ;  
company C(40) NOT NULL, contact C(30) NULL)
```

Você pode também controlar se os valores nulos são permitidos ou não em campos da tabela utilizando o comando [SET NULL ON](#).



► **Para permitir que valores nulos sejam utilizados em todos os campos da tabela**

- No **Criador de tabelas**, selecione a coluna **Nulo** para cada um dos campos da tabela.  
– Ou –
- Utilize o comando **SET NULL ON** antes de utilizar o comando **CREATE TABLE**.

Quando você emitir o comando **SET NULL ON**, o Visual FoxPro marcará automaticamente a coluna **NULO** para cada campo da tabela à medida que você adicionar campos no **Criador de tabelas**. Se você emitir o comando **SET NULL** antes de emitir **CREATE TABLE**, não precisará especificar as cláusulas **NULL** ou **NOT NULL**. Por exemplo, o código abaixo cria uma tabela que permite valores nulos em todos os campos da tabela:

```
SET NULL ON
CREATE TABLE test (field1 C(6), field2 C(40), field3 Y)
```

A presença de valores nulos afeta o comportamento das tabelas e [índices](#). Por exemplo, se você utilizar **APPEND FROM** para copiar registros de uma tabela contendo valores nulos para uma tabela que não permite valores nulos, os campos incluídos que contiverem valores nulos serão tratados como em branco, vazios ou sendo iguais a zero na segunda tabela.

Para obter maiores informações sobre como os valores nulos interagem com os comandos do Visual FoxPro, consulte [Gerenciando valores nulos](#).

## **Acrescentando comentários aos campos**

Depois que você cria uma tabela em um banco de dados aberto, é possível adicionar uma descrição de cada campo da tabela para que seja mais fácil entender e atualizar as tabelas. O Visual FoxPro exibe o texto com comentários sobre um campo no **Gerenciador de projetos** quando você seleciona o campo na lista de campos da tabela.

► **Para acrescentar um comentário a um campo em uma tabela de banco de dados**

- No **Criador de tabelas**, digite o texto do comentário na caixa **Comentário do campo**.  
– Ou –
- Utilize a função **DBSETPROP()**.

Por exemplo, você talvez queira deixar mais claro o que está armazenado no campo `unit_price` da tabela `orditems`. Para fazer isso, você digita “Preço atual de venda por unidade” como um texto de comentário para este campo:

```
?DBSETPROP('orditems.price', 'field', 'comment', ;
           'Preço atual de venda por unidade')
```

Para obter maiores informações sobre como utilizar **DBSETPROP()** para definir propriedades em campos de uma tabela de banco de dados, consulte o capítulo 6, [Criando bancos de dados](#)

## **Criando valores padrão para os campos**

Se quiser que o Visual FoxPro preencha automaticamente o conteúdo de um campo sempre que você adicionar um novo registro, você poderá criar um [valor padrão](#) para o campo. O valor padrão é utilizado não importando se os dados foram digitados em um [formulário](#), em uma janela **Pesquisar**, uma [visualização](#), ou através da linguagem de programação, e permanece no campo até que um novo valor seja fornecido.

Você cria valores padrão através do **Criador de tabelas** ou através da linguagem. Você pode especificar valores padrão para qualquer tipo de dado exceto o tipo **Geral**.

► **Para atribuir um valor padrão para um campo de uma tabela**

- No **Criador de tabelas**, digite o valor na caixa **Valor padrão** na área **Validação do campo**.  
– Ou –
- Utilize a cláusula **DEFAULT** do comando **CREATE TABLE**.



Por exemplo, você talvez queira que seu aplicativo limite a quantidade de mercadorias que um novo cliente pode encomendar até que tenha sido possível completar uma verificação de crédito e determinar a quantidade de crédito que deverá ser oferecido a esse cliente. O exemplo a seguir cria um campo `maxordamt` com um valor padrão de 1000:

```
CREATE TABLE customer (cust_id C(6), company C(40), contact C(30), ;  
                        maxordamt Y(4) DEFAULT 1000)
```

Se a tabela `customer` já possuíse uma coluna `maxordamt`, você poderia acrescentar um valor padrão para a coluna com este comando:

```
ALTER TABLE customer ALTER COLUMN maxordamt SET DEFAULT 1000
```

## Utilizando valores padrão para facilitar a entrada de dados

Você pode utilizar [valores padrão](#) para facilitar a entrada de dados para os usuários de seu aplicativo, permitindo que eles ignorem um campo a não ser que queiram digitar um valor diferente. Por exemplo, se seu ramo de negócios lida basicamente com clientes de seu próprio país, você talvez queira que o campo `country` na tabela `customer` seja preenchido automaticamente com o nome de seu país. Caso esteja digitando um registro de cliente referente a um cliente internacional, você poderá sobrescrever o nome de seu país com o nome do país do cliente.

**Dica** Caso uma das regras de operação de seu aplicativo exija que um campo contenha sempre algum valor, fornecer um valor padrão ajuda a garantir que as regras de [campos](#) ou de [registros](#) não serão violadas.

Caso você remova ou exclua uma tabela de um banco de dados, todos os valores padrão associados àquela tabela serão excluídos do banco de dados. Os [procedimentos armazenados](#) aos quais é feita referência por um valor padrão removido ou excluído permanecerão, mesmo após a remoção desse valor padrão.

Quando você não especificar um valor padrão, um valor em branco (conforme definido para cada tipo de dado) será inserido, a menos que [SET NULL](#) esteja ativado (ON). Isso mantém a compatibilidade com as versões anteriores de código FoxPro que você possa ter.

É possível utilizar `.NULL.` como um valor padrão se você quiser que o campo utilize valores nulos. Se `.NULL.` for utilizado como valor padrão, o Visual FoxPro inserirá `.NULL.` para todos os comandos, exceto `APPEND BLANK`, quer `SET NULL` esteja ativado (ON) ou desativado (OFF).

## Valores padrão permitidos

Você pode especificar valores padrão que sejam valores escalares (como 'um número') ou expressões que resultem em uma quantidade escalar. Também é possível especificar qualquer expressão válida de Xbase que retorne um valor consistente com o tipo de dado do campo.

O Visual FoxPro avalia o tipo de dado das [expressões](#) quando a estrutura da tabela é fechada. Se o tipo de dado não corresponder ao tipo de campo associado, o Visual FoxPro gerará um erro. Se a expressão for uma [função definida pelo usuário](#) (UDF) ou contiver uma UDF, ela não será avaliada.

Quando você cria o valor padrão utilizando a linguagem, os comandos `CREATE TABLE` ou `ALTER TABLE` gerarão um erro se os tipos de dados não forem correspondentes. Se a expressão for uma função definida pelo usuário (UDF) ou contiver uma UDF, ela não será avaliada ao executar `CREATE` e nenhum erro será retornado.

## Quando valores padrão são utilizados

Os [valores padrão](#) são avaliados (se necessário) e colocados nos campos adequados quando os comandos `APPEND`, `APPEND BLANK` ou `INSERT` são emitidos.

Quando você atribui valores com os comandos `APPEND FROM` ou `INSERT - SQL`, o Visual FoxPro atribui valores padrão para todos os campos que não estiverem explicitamente designados. Os comandos `APPEND FROM` e `INSERT - SQL` também respeitam valores padrão. Contudo, quando qualquer um desses comandos for emitido, os padrões não irão sobrescrever valores já existentes

nos campos. Se os campos incluídos ou inseridos contiverem valores, os valores existentes serão mantidos quando o registro for incluído ou inserido e o valor padrão não será utilizado.

## Utilizando valores padrão para preencher automaticamente campos Não Nulos

Os [valores padrão](#) são particularmente úteis para preencher automaticamente campos que não permitam [valores nulos](#). Quando você adiciona um novo registro, os valores padrão são utilizados primeiro, e depois o programa verifica cada campo, em ordem de definição, para determinar se falta alguma informação. Isso assegura que os campos designados como NOT NULL possam ser preenchidos com valores padrão antes que a restrição NOT NULL seja aplicada.

## Especificando uma máscara de entrada

Ao especificar uma [máscara de entrada](#), você define os atributos de pontuação, espaçamento e de outros formatos de valores à medida que são digitados no campo. Os valores são armazenados de forma uniforme podendo reduzir erros de entrada de dados e tornar o processamento mais eficiente. Por exemplo, adicionar uma máscara a um campo numérico que armazena números de telefone ajuda o usuário a preencher rapidamente o campo porque a pontuação e os espaços já são fornecidos pela máscara.

### ► Para fornecer uma máscara de entrada

- No **Criador de tabelas**, digite a máscara na caixa **Máscara de entrada** na área **Exibir**.  
– Ou –
- Utilize a função `DBSETPROP( )` para definir a propriedade InputMask.

Por exemplo, o código a seguir especifica uma máscara de entrada para uma data:

```
DBSetProp("orders.postalcode", "field", "InputMask", "99999-9999")
```

## Controlando a exibição de um campo

As propriedades adicionais para campos permitem que você controle como um campo e seus valores aparecem em [formulários](#), janelas **Pesquisar** e relatórios. Você pode especificar um formato de exibição, uma [legenda](#) de campo padrão e uma [classe](#) padrão e [biblioteca de classe](#).

## Definindo um formato

Um formato fornece uma máscara de saída que determina o modo como o valor de um campo é exibido em um formulário, janela **Pesquisar** ou relatório. Por exemplo,

### ► Para fornecer um formato

- No **Criador de tabelas**, digite a máscara na caixa **Formato** na área **Exibir**.  
– Ou –
- Utilize a função `DBSETPROP( )` para definir a propriedade Format.

Por exemplo, o código a seguir especifica o formato de exibição para um código postal:

```
DBSetProp("orders.postalcode", "field", "Format", "@R 99999-9999")
```

## Criando legendas para os campos

Você pode criar uma [legenda](#) para cada campo de uma tabela de banco de dados. O Visual FoxPro exibe o texto da legenda de um campo como o cabeçalho de coluna em uma janela **Pesquisar** e como nome padrão de cabeçalho em uma grade de formulário.

### ► Para acrescentar uma legenda a um campo em uma tabela de um banco de dados

- No **Criador de tabelas**, digite o texto da sua legenda na caixa **Legenda** da área **Exibir**.  
– Ou –
- Utilize a função `DBSETPROP( )`.

Por exemplo, você pode querer criar uma legenda para o campo `fax` em sua tabela `supplier` digitando “Supplier\_Fax” como legenda para o campo:

```
?DBSETPROP('supplier.fax', 'field', 'caption', 'Supplier_Fax')
```

Para obter maiores informações sobre como utilizar `DBSETPROP()` para definir propriedades de campos de tabela de banco de dados, consulte o capítulo 6, [Criando bancos de dados](#)

## Definindo uma classe padrão

Para economizar tempo mais tarde quando estiver criando [formulários](#), você poderá definir uma [classe](#) padrão para um campo. Depois de definido, toda vez que você adicionar um campo a um formulário, o [controle](#) no formulário utilizará a classe especificada como padrão. Por exemplo, os campos de caractere aparecem automaticamente como controles de caixa de texto quando você os adiciona a um formulário. Se quiser criar automaticamente um controle de caixa de combinação quando utilizar o campo em um formulário, você poderá definir esta classe como padrão para o campo. Você pode também utilizar as [bibliotecas de classe](#) que você criou.

### ► Para definir uma classe padrão

- No **Criador de tabelas**, selecione uma classe e uma biblioteca nas caixas **Classe padrão** e **Biblioteca padrão**.

Se verificar que está alterando com frequência a biblioteca e a classe dos seus campos, você poderá mapear os tipos de dados dos campos para uma biblioteca e classe na [caixa de diálogo Opções](#). Para obter maiores informações sobre como mapear os tipos de dados de campo para classes, consulte o capítulo 3, [Configurando o Visual FoxPro](#), no *Guia de Instalação e Índice Principal*. Para obter maiores informações sobre como criar classes, consulte o capítulo 3, [Programação orientada a objetos](#), neste manual.

## Executando regras de Business

Você pode executar regras de Business para a entrada de dados através da criação de regras em nível de campo e de registro, denominadas [regras de validação](#), para controlar os dados fornecidos nos campos e registros de tabelas de banco de dados. As regras de campos e de registros comparam os valores digitados com as expressões que você definiu como regras. Se o valor digitado não atender aos requisitos da regra, ele será rejeitado. As regras de validação somente existem em tabelas de banco de dados.

As regras de campos e de registros possibilitam que você controle o tipo de informação digitado em uma tabela, não importando se os dados são acessados através de uma [janela Pesquisar](#), de um [formulário](#), ou através da linguagem. Elas permitem que você aplique, de forma consistente, a regra de um campo utilizando menos código do que seria necessário caso você escrevesse a regra como código em uma cláusula `VALID` em um formulário, ou em uma seção de código de um programa. Além disso, as regras que você estabelece para um banco de dados são aplicadas a todos os usuários da tabela, não importando quais sejam os requisitos do aplicativo.

Você pode também criar índices [candidatos](#) ou [primários](#) que evitem entradas duplicadas em um campo, além de disparadores para assegurar a [integridade referencial](#) ou executar outras ações quando os dados de seu banco de dados forem alterados.

## Quando as restrições são executadas

As restrições do banco de dados são escolhidas com base no nível em que você deseja aplicar uma regra de operação ou de integridade referencial, e também com base na ação que faz com que a restrição seja ativada. A tabela a seguir lista as restrições de validação dos dados na ordem em que são aplicadas pelo mecanismo do Visual FoxPro, o nível em que são aplicadas e o momento em que a validação é ativada.

Mecanismo de execução	Nível	Ativado
-----------------------	-------	---------

---

Validação de NULO	Campo ou coluna	Quando você sai do campo/coluna em uma pesquisa, ou quando o valor do campo muda durante um INSERT ou REPLACE.
Regras em nível de campo	Campo ou coluna	Quando você sai do campo/coluna em uma pesquisa, ou quando o valor do campo muda durante um INSERT ou REPLACE.
Regras em nível de registro	Registro	Quando ocorre a atualização do registro.
Índice candidato/primário	Registro	Quando ocorre a atualização do registro.
Cláusula VALID	Formulário	Quando você sai do registro.
Disparadores	Tabela	Quando os valores da tabela são alterados durante um evento INSERT, UPDATE ou DELETE.

As restrições são ativadas na ordem em que estão colocadas na tabela. A primeira violação de qualquer teste de validação interrompe o comando.

Os índices candidato e primário são explicados posteriormente neste capítulo, na seção [Controlando valores duplicados](#).

## Limitando os valores de um campo

Para controlar o tipo de informação que um usuário pode digitar em um campo, quando for possível validar os dados de um campo independentemente de qualquer outra entrada no registro, utiliza-se uma regra de validação em nível de campo. Por exemplo, é possível utilizar uma regra de validação de campo para assegurar que o usuário não digitará um número negativo em um campo que deve conter somente valores positivos. Também é possível utilizar uma regra de campo para comparar os valores digitados em um campo com valores existentes em outra tabela.

Você não deve criar regras de campos ou de registros que sejam específicas de um aplicativo. Utilize as regras de validação de campos ou de registros para aplicar a integridade de dados e as regras de operação que sempre se apliquem aos dados em seu banco de dados, independente de quem for acessar os dados. Por exemplo, você pode criar uma regra que compare a entrada do campo `postal_code` de uma tabela com uma tabela de pesquisa contendo os códigos postais de seu país, e que rejeite qualquer valor não existente nesta tabela.

### ► Para criar uma regra de campo

- No **Criador de tabelas**, digite a expressão da regra na caixa **Regra** da área **Validação de campo**.  
– Ou –
- Utilize a cláusula CHECK do comando **CREATE TABLE**.  
– Ou –
- Utilize a cláusula SET CHECK do comando **ALTER TABLE**.

Por exemplo, o código abaixo acrescenta uma regra de validação de campo à tabela `orditems`. Essa regra exige que os números digitados no campo `quantity` sejam iguais ou maiores que 1:

```
ALTER TABLE orditems
    ALTER COLUMN quantity SET CHECK quantity >= 1
```

Quando o usuário tentar digitar um valor menor que 1, o Visual FoxPro exibirá uma caixa de diálogo de erro e o valor será rejeitado.

Você pode personalizar a mensagem exibida quando a regra é violada, acrescentando um texto de validação ao campo. O texto que você digitar será exibido no lugar da mensagem de erro padrão.

► **Para acrescentar uma mensagem de erro personalizada a uma regra em nível de campo**

- No **Criador de tabelas**, digite a mensagem de erro desejada na caixa **Mensagem** na seção **Validação de campo**.

– Ou –

- Utilize a cláusula opcional ERROR com a cláusula CHECK do comando **CREATE TABLE** ou **ALTER TABLE**.

Por exemplo, o código abaixo acrescenta uma regra de validação de campo para a tabela `orditems`, exigindo que os números digitados na coluna `quantity` sejam iguais ou maiores que 1. É acrescentada, também, uma mensagem de erro personalizada:

```
ALTER TABLE orditems ;  
    ALTER COLUMN quantity SET CHECK quantity >= 1 ;  
    ERROR "As quantidades devem ser maiores que ou iguais a 1"
```

Quando o usuário tentar digitar um valor menor que 1, o Visual FoxPro exibirá uma caixa de diálogo de erro com a mensagem de erro personalizada que você definiu, rejeitando o valor inválido. Você pode também utilizar a cláusula SET CHECK do comando ALTER TABLE com a cláusula opcional ERROR para criar uma mensagem de erro personalizada.

### Quando as regras de campos são verificadas

As regras de campos são verificadas quando o valor do campo muda. Ao contrário dos [disparadores](#), essas regras são ativadas mesmo que um buffer esteja sendo utilizado. Quando você trabalha com dados em uma janela **Pesquisar**, um formulário ou outra janela da interface com o usuário, o Visual FoxPro verifica as regras de campos quando você sai do campo. Se o valor de um campo não tiver sido alterado, a regra não será verificada. Isso significa que você pode tabular pelos campos sem que o sistema valide os dados.

### Verificação de regras em nível de campo

Método de entrada de dados	Janela ou comando	Verificação da regra
Interface com o usuário	<b>Janela Pesquisar</b> <b>Formulário</b> Outra janela	Quando você sai do campo, se o valor do campo tiver sido mudado. (Se o valor do campo não tiver sido alterado, a regra não será verificada.)
Comandos que não especificam campos	<b>APPEND</b> <b>APPEND GENERAL</b> <b>APPEND MEMO</b> <b>BROWSE</b> <b>CHANGE</b> <b>DELETE</b> <b>EDIT</b> <b>GATHER</b>	Quando o valor do campo muda, na ordem da definição dos campos.
	<b>APPEND BLANK</b> <b>INSERT</b> <b>INSERT - SQL</b>	Quando o registro é incluído ou inserido.
Comandos que especificam campos	<b>UPDATE</b> <b>UPDATE - SQL</b> <b>REPLACE</b>	Na ordem em que os campos são especificados no comando.

## Validando valores em nível de registro

As regras de validação em nível de registro são utilizadas para controlar o tipo de informação que um usuário pode digitar em um registro. As regras de validação em nível de registro geralmente comparam os valores de dois ou mais campos no mesmo registro para verificar se eles seguem as regras de operação estabelecidas para o banco de dados. Por exemplo, é possível utilizar uma regra de validação de registro para assegurar que o valor de um campo é sempre maior do que o de outro no mesmo registro.

### ► Para criar uma regra de validação de registro e uma mensagem de erro personalizada

- Na guia **Tabela** do **Criador de tabelas**, digite a regra e a mensagem de erro desejadas nas caixas **Regra** e **Mensagem**.

– Ou –

- Utilize a cláusula CHECK do comando **CREATE TABLE** ou **ALTER TABLE**.

Por exemplo, você talvez queira assegurar que todos os funcionários tenham 18 anos ou mais ao serem empregados. O código a seguir acrescenta uma regra de validação de registro e um texto de erro para a tabela `employee`, exigindo que a data de contratação digitada na coluna `hire_date` seja maior do que ou igual à data de nascimento mais 18 anos:

```
ALTER TABLE employee SET CHECK ;  
    hire_date >= birth_date + (18 * 365.25) ;  
    ERROR "Os funcionários devem ter 18 anos ou mais na data de contratação "
```

Se o usuário digitar um registro de funcionário com uma data inválida, o Visual FoxPro exibirá uma caixa de diálogo de erro com a mensagem de erro personalizada que você definiu, e não atualizará o registro.

Você pode também utilizar a cláusula SET CHECK do comando **ALTER TABLE** para criar uma regra de validação de registro. Você deve certificar-se de que as regras especificadas para os campos não entrem em conflito semântico com as regras definidas para a tabela. O Visual FoxPro não verifica a consistência entre as expressões em nível de campo e de registro.

## Quando as regras de registros são verificadas

As regras de registro, assim como as regras de campos, são ativadas quando o valor do registro muda. A maneira como você trabalha com os dados não importa: seja em uma janela **Pesquisar**, em um formulário, em outra janela da interface com o usuário ou através de comandos que alteram dados, o Visual FoxPro verifica as regras de registros quando o ponteiro do registro é movido para outro registro. Caso nenhum valor tenha sido alterado dentro do registro, as regras de registros não serão verificadas quando você mover o ponteiro do registro. Você pode mover-se pelos registros sem que o sistema valide os dados.

Caso você modifique um registro, mas não mova o ponteiro do registro, e depois feche a janela **Pesquisar**, ainda assim a regra será verificada, você será avisado sobre eventuais erros e a janela **Pesquisar** será fechada.

---

**Cuidado** Não inclua nas regras de validação comandos ou funções que tentem mover o ponteiro do registro na [área de trabalho](#) atual (ou seja, na área de trabalho cujas regras estão sendo verificadas). A inclusão de comandos ou funções como SEEK, LOCATE, SKIP, APPEND, APPEND BLANK, INSERT ou AVERAGE, COUNT, BROWSE e REPLACE FOR em regras de validação pode fazer com que elas sejam ativadas de forma recursiva, gerando uma condição de erro.

---

Ao contrário dos [disparadores](#), as regras de registros disparam mesmo que os dados estejam no buffer. Quando uma regra em nível de registro dispara durante a execução de um aplicativo, você precisa incluir código de gerenciamento de erros. Geralmente, isso implica em não permitir que o aplicativo saia do formulário (ou altere o ambiente ativo, de forma mais genérica) até que o usuário corrija o erro relatado ou cancele a atualização.



## Removendo de um banco de dados uma tabela com regras associadas

Caso você remova ou exclua uma tabela de um banco de dados, todas as regras de campos e de registros que estejam vinculadas a essa tabela serão excluídas do banco de dados. Isso porque as regras são armazenadas no arquivo .DBC, e remover uma tabela de um banco de dados quebra o vínculo entre o arquivo .DBF e seu arquivo .DBC. Contudo, os [procedimentos armazenados](#) aos quais é feita referência nas regras removidas ou excluídas não são excluídos automaticamente, já que podem estar sendo utilizados por regras de outras tabelas que permanecem no banco de dados.

## Utilizando disparadores

Um disparador é uma expressão vinculada a uma tabela que é chamada quando um dos registros da tabela é modificado por um dos comandos especificados de modificação de dados. Os disparadores podem ser utilizados para executar qualquer tipo de operação paralela de que um aplicativo de banco de dados necessite quando os dados forem modificados. Por exemplo, é possível utilizar disparadores para:

- Registrar modificações no banco de dados.
- Executar a [integridade referencial](#).
- Criar uma nova ordem automática para um produto cujo estoque está baixo.

Os disparadores são criados e armazenados como [propriedades](#) de uma tabela específica. Caso você remova uma tabela de um banco de dados, os disparadores associados a essa tabela serão excluídos. Os disparadores serão ativados depois que todas as outras verificações, como regras de validação, verificação de chave primária e verificação de valores nulos, forem executadas. E, ao contrário das regras de validação em nível de campo e de registro, os disparadores não são ativados quando os dados estão no buffer.

## Criando disparadores

Para criar disparadores, utilize o **Criador de tabelas** ou o comando CREATE TRIGGER. Para cada tabela, você pode criar um disparador para cada um dos três eventos: INSERT, UPDATE e DELETE. Uma tabela pode ter no máximo três disparadores simultaneamente. Um disparador deve retornar um valor verdadeiro (.T.) ou falso (.F.).

### ► Para criar um disparador

- Na guia **Tabela** do **Criador de tabelas**, digite a expressão do disparador ou o nome de um procedimento armazenado que contenha a expressão do disparador na caixa **Disparador de inserção**, **Disparador de atualização** ou **Disparador de exclusão**.  
– Ou –
- Utilize o comando [CREATE TRIGGER](#).

Por exemplo, a cada vez que o Tasmanian Traders vender um item, eles querem comparar o número de `Units_in_stock` com o `Reorder_level` e receber um aviso caso o item deva ser encomendado. Pode ser criado um **Disparador de atualização** na tabela `products` para este fim. Este seria o disparador adequado, e não os de Exclusão ou Inserção, porque o disparador deve ser ativado a cada vez que um produto for vendido, e o campo `Units_in_stock` é atualizado a cada vez que um produto é vendido para refletir o número de itens restantes no estoque.

Para criar o disparador, é possível especificar `updProductsTrigger( )` como **Disparador de atualização** para a tabela `products`. É possível adicionar um campo à tabela, denominado `reorder_amount`, que armazena o número de itens a encomendar em cada pedido de um item, e criar uma tabela `reorder` com os campos a seguir: `product_id` e `reorder_amount`. É possível, então, adicionar o código a seguir ao seu procedimento armazenado:

```
PROCEDURE updProductsTrigger
  IF (units_in_stock+units_on_order) <= reorder_level
  INSERT INTO Reorder VALUES (Products.product_id, ;
```



```
Products.reorder_amount)
ENDIF
ENDPROC
```

Você pode criar disparadores parecidos para um evento de inserção ou exclusão utilizando a cláusula FOR INSERT ou FOR DELETE, respectivamente, no lugar da cláusula FOR UPDATE. Caso você tente criar um disparador que já exista para um determinado evento e tabela enquanto **SET SAFETY** estiver ativado (ON), o Visual FoxPro perguntará se você quer sobrescrever o disparador existente.

## Removendo ou excluindo disparadores

Você pode remover um disparador de uma tabela de banco de dados através da interface ou com o comando DELETE TRIGGER.

### ► Para excluir um disparador

- Na guia **Tabela** do **Criador de tabelas**, selecione e exclua a expressão do disparador na caixa **Disparador de inserção**, **Disparador de atualização** ou **Disparador de exclusão**.
  - Ou –
- Utilize o comando **DELETE TRIGGER**.

O exemplo a seguir remove o **Disparador de atualização** da tabela `customer`:

```
DELETE TRIGGER ON customer FOR UPDATE
```

Caso você remova ou exclua uma tabela de um banco de dados, todos os disparadores vinculados a essa tabela serão excluídos do banco de dados. Contudo, os [procedimentos armazenados](#) aos quais é feita referência no disparador removido ou excluído não serão excluídos.

## Modificando disparadores

Você pode modificar disparadores utilizando o **Criador de tabelas** ou a linguagem.

### ► Para modificar um disparador

- Na guia **Tabela** do **Criador de tabelas**, digite a nova expressão do disparador na caixa **Disparador de inserção**, **Disparador de atualização** ou **Disparador de exclusão**.
  - Ou –
- Emita o comando **SET SAFETY OFF**, seguido do comando **CREATE TRIGGER**.

Ao modificar um disparador emitindo primeiro o comando **SET SAFETY OFF** e depois recriando o disparador, a antiga expressão do disparador será automaticamente excluída e substituída pela expressão que foi recriada.

## Utilizando disparadores para criar integridade referencial

O Visual FoxPro fornece um **Construtor de integridade referencial** para gerar disparadores e procedimentos armazenados que irão aplicar e manter a Integridade Referencial (IR) de seu banco de dados. Para obter maiores informações sobre como utilizar o **Construtor de integridade referencial**, consulte o capítulo 6, [Criando bancos de dados](#).

## Modificando a estrutura da tabela

Depois que você tiver criado uma tabela, é sempre possível modificar a estrutura da tabela e suas propriedades. Você pode querer adicionar, alterar ou excluir nomes e larguras de campos, tipos de dados, alterar valores padrão ou regras, ou ainda adicionar comentários ou legendas.

Você abrir o **Criador de tabelas** para modificar a estrutura de uma tabela ou efetuar as mudanças através da linguagem, utilizando o comando ALTER TABLE. Certifique-se de que você tem acesso exclusivo à tabela antes de modificar sua estrutura.

► **Para modificar a estrutura de uma tabela utilizando o Criador de tabelas**

- No **Gerenciador de projetos**, selecione o nome da tabela e escolha **Modificar**.  
– Ou –
- No **Criador de bancos de dados**, selecione a tabela no esquema e escolha **Modificar** no menu **Bancos de dados**.  
– Ou –
- Utilize o comando **MODIFY STRUCTURE**.

Por exemplo, você pode modificar a estrutura da tabela de banco de dados `employee` com os comandos a seguir:

```
OPEN DATABASE testdata
USE employee EXCLUSIVE
MODIFY STRUCTURE
```

Todas as opções anteriores abrem o **Criador de tabelas**.

► **Para modificar a estrutura de uma tabela utilizando a linguagem de programação**

- Utilize o comando **ALTER TABLE**.

O comando **ALTER TABLE** possui diversas cláusulas que permitem adicionar ou retirar campos da tabela, criar ou retirar chaves primárias ou exclusivas, ou marcas de chaves estrangeiras, e renomear os campos já existentes. Algumas cláusulas dizem respeito somente a tabelas associadas a um banco de dados. Alguns exemplos específicos foram incluídos nesta seção.

## Adicionando campos

Você pode adicionar um novo campo a uma tabela com o **Criador de tabelas** ou utilizando a linguagem.

► **Para adicionar um campo a uma tabela**

- No **Criador de tabelas**, escolha **Inserir**.  
– Ou –
- Utilize a cláusula **ADD COLUMN** do comando **ALTER TABLE**.

Por exemplo, o comando a seguir adiciona um campo denominado `fax` à tabela `customer` e permite que o campo tenha valores nulos:

```
ALTER TABLE customer ADD COLUMN fax c(20) NULL
```

## Excluindo campos

Você pode excluir um campo já existente de uma tabela utilizando o **Criador de tabelas** ou a linguagem.

► **Para excluir um campo de uma tabela**

- No **Criador de tabelas**, selecione o campo e escolha **Excluir**.  
– Ou –
- Utilize a cláusula **DROP COLUMN** do comando **ALTER TABLE**.

Por exemplo, o comando a seguir exclui o campo `fax` da tabela `customer`:

```
ALTER TABLE customer DROP COLUMN fax
```

Quando um campo é removido de uma tabela, a definição de valor padrão, as definições de regras e a legenda do campo também são removidas. Se expressões de chave de índice ou disparador fizerem referência ao campo, as expressões se tornarão inválidas quando o campo for removido. A expressão inválida de chave de índice ou disparador não irá gerar um erro até a [hora da execução](#).

## Renomeando campos

Você pode renomear campos já existentes na tabela de duas formas.

► **Para renomear um campo de tabela**

- No **Criador de tabelas**, digite um novo nome para o campo na caixa **Nome**.  
– Ou –

- Utilize a cláusula RENAME COLUMN do comando **ALTER TABLE**.

Por exemplo, para renomear a coluna `company` na tabela `customer`, utilize o comando a seguir:

```
ALTER TABLE customer RENAME COLUMN company TO company_long_new_name
```

No exemplo acima, o novo nome utiliza o recurso de criar nomes de campo extensos nas tabelas de bancos de dados.

## Definindo ou alterando regras de campos ou tabelas

Você pode definir novas expressões e textos de regras de campos ou de tabelas, assim como alterar regras e textos definidos com os comandos CREATE TABLE ou ALTER TABLE.

► **Para alterar uma regra já existente**

- No **Criador de tabelas**, selecione a guia **Tabela** e digite a nova expressão ou texto de regra nas caixas **Regra** e **Mensagem** na seção **Validação de registros**.

– Ou –

- Utilize o comando **ALTER TABLE**.

A função **DBGETPROP( )** pode ser utilizada para visualizar a expressão ou texto de regra atuais; esses valores são somente para leitura em se tratando de tabelas e somente podem ser alterados através do comando ALTER TABLE.

## Definindo ou alterando valores padrão

Você pode definir ou alterar valores padrão de campos da tabela depois que a tabela tiver sido construída.

► **Para alterar um valor padrão já existente**

- No **Criador de tabelas**, digite o novo valor na caixa **Valor padrão** da guia **Campos**.

– Ou –

- Utilize o comando **ALTER TABLE**.

A função **DBGETPROP( )** pode ser utilizada para visualizar o valor padrão atual de um campo; esses valores são somente para leitura em se tratando de tabelas e somente podem ser alterados através do comando ALTER TABLE.

## Trabalhando com registros

Após ter projetado e criado a estrutura de uma tabela, você pode armazenar dados na tabela acrescentando novos registros. Posteriormente, você irá alterar e excluir registros já existentes. Cada uma dessas tarefas pode ser executada através da interface ou utilizando comandos. Esta seção trata basicamente de como trabalhar com registros utilizando a linguagem de programação. Para obter maiores informações sobre como trabalhar com registros através da interface, consulte o capítulo 2, **Criando tabelas e índices**, no *Guia do Usuário*.

## Adicionando registros

Ao criar uma tabela do Visual FoxPro, ela estará aberta, porém vazia. Caso você tente armazenar dados em uma tabela sem antes criar um registro na tabela, nada acontecerá. O primeiro passo para adicionar registros a uma nova tabela é adicionar linhas para armazenar os novos dados.

► **Para adicionar registros a uma tabela**

- Utilize o comando [INSERT - SQL](#).

O comando [INSERT - SQL](#) pode ser utilizado para inserir valores especificados através do próprio comando ou para inserir valores de uma [matriz](#) ou [variável](#) de memória. Por exemplo, para inserir um novo registro na tabela `customer` do banco de dados `TasTrade`, você emitiria o comando a seguir:

```
INSERT INTO customer (cust_id, company, contact) ;
VALUES ("SMI007", "Smith's Delicatessen", "Sarah Smith")
```

O comando [INSERT - SQL](#) é útil para dados remotos, já que utiliza uma sintaxe SQL de acordo com a norma [ANSI](#).

Você pode também utilizar o comando [APPEND BLANK](#) seguido pelo comando [REPLACE](#) para adicionar um registro em branco a uma tabela e depois armazenar dados em um campo. O comando [APPEND BLANK](#) inclui um novo registro, em branco, em uma tabela. O comando [REPLACE](#) substitui o valor atual de um campo, mesmo que vazio, por um novo valor.

O comando [REPLACE](#) requer:

- Uma [tabela](#) aberta.
- Um [registro](#) já existente.
- O nome do [campo](#) no qual o valor deve ser armazenado.
- Um valor para cada campo que seja válido para o [tipo de dado](#) do campo.

O exemplo a seguir utiliza o comando [APPEND BLANK](#) para criar um registro no qual é possível armazenar dados utilizando o comando [REPLACE](#):

```
APPEND BLANK                                && o registro foi criado
REPLACE lastname WITH "SMITH"                && armazena um valor do tipo caractere no campo
```

Você pode utilizar o comando [UPDATE - SQL](#) em vez do comando [REPLACE](#) para atualizar registros em uma tabela.

## Incluindo registros de outra tabela

Outra forma de armazenar dados em registros é copiá-los de outras tabelas ou arquivos. Por exemplo, você pode [incluir](#) registros de outra tabela ou arquivo.

### ► Para incluir registros de outro arquivo

- Utilize o comando [APPEND FROM](#).
- Ou –
- Utilize o comando [IMPORT](#).

Os registros podem aceitar dados diretamente, como no exemplo anterior, onde o comando [INSERT](#) especificou o texto a ser inserido em campos específicos da tabela `customer`. Os dados podem também vir de [constantes](#), [variáveis](#), [matrizes](#), [objetos](#) e outras fontes de dados. Para obter maiores informações sobre outras formas de importar dados, consulte o capítulo 9, [Importando e exportando dados](#), no *Guia do Usuário*.

## Adicionando registros no modo de pesquisa

Se você deseja adicionar um novo registro enquanto visualiza uma tabela no modo de pesquisa, selecione **Incluir registro** no [menu Tabela](#). Por outro lado, se você deseja impedir que os usuários possam incluir um novo registro enquanto estão no modo de pesquisa, utilize a cláusula [NOAPPEND](#) do comando [BROWSE](#).

## Inserindo dados em uma tabela

Você pode inserir dados a uma tabela de forma interativa, através de uma [janela Pesquisar](#), ou utilizando a linguagem de programação, com os comandos [REPLACE](#) ou [UPDATE - SQL](#). Ao utilizar

REPLACE ou UPDATE -SQL em um aplicativo multiusuário, você pode utilizar um buffer de registro ou de tabela, que permite a você editar os dados sem bloquear o registro até que você queira gravar as alterações fisicamente. Para obter maiores informações sobre a utilização de buffers de registro e de tabela, consulte o capítulo 17, [Programando para acesso compartilhado](#).

## Editando registros em uma tabela

Você pode exibir e editar os registros já existentes em uma tabela através da interface ou utilizando a linguagem de programação.

### ► Para exibir os registros para edição

- Utilize o comando [EDIT](#).  
– Ou –
- Utilize o comando [CHANGE](#).

Por exemplo, o código abaixo exibe a tabela `customer` em uma janela **Pesquisar** no modo de edição:

```
USE customer
EDIT
```

Se você deseja utilizar um [formulário](#) para editar um registro, crie uma [caixa de texto](#) em seu formulário e defina a respectiva [propriedade](#) DataSource como sendo o nome da tabela que você deseja editar. Para obter maiores informações sobre formulários, consulte o capítulo 9, [Criando formulários](#).

Você também pode utilizar os comandos [CHANGE](#) e [EDIT](#) para alterar campos específicos de uma tabela.

## Adicionando gráficos a uma tabela

Para armazenar gráficos em uma tabela do Visual FoxPro, crie um [campo do tipo Geral](#) e importe ou cole objetos OLE, como bitmaps ou gráficos, no campo. O comando [APPEND GENERAL](#) coloca um objeto OLE em um campo do tipo Geral. O exemplo a seguir armazena um arquivo de gráfico do Microsoft Excel, localizado no diretório padrão do Visual FoxPro, em um campo do tipo Geral denominado Chart:

```
APPEND GENERAL Chart FROM "CHART1.CLX" CLASS EXCELCHART
```

Para obter maiores informações sobre como trabalhar com objetos OLE nas tabelas do Visual FoxPro, consulte o capítulo 16, [Adicionando OLE](#)

## Inserindo valores nulos nos campos

Você pode inserir um [valor nulo](#) em um campo utilizando a linguagem, com `.NULL.`, ou através da interface, utilizando uma combinação de teclas, se o campo aceitar valores nulos.

### ► Para armazenar um valor nulo em um campo

- Em uma janela **Pesquisar** ou controle de formulário, pressione CTRL+0 (zero).  
– Ou –
- Utilize o token `NULL`.

Por exemplo, o código abaixo substitui o valor existente no campo `automobile` por um valor nulo:

```
REPLACE automobile WITH NULL
```

**Observação** Utilize o comando [SET NULLDISPLAY](#) para especificar o texto exibido para valores nulos.

## Excluindo registros

Para excluir registros, você os marca para exclusão e depois remove os registros excluídos. Até que

você remova os registros sinalizados para exclusão, estes ainda estarão em disco e poderão ser desmarcados e restaurados. Esta seção descreve como marcar, desmarcar e remover registros de sua tabela.

## Marcando registros para exclusão

É possível marcar registros para exclusão através da interface ou utilizando o comando **DELETE - SQL**.

### ► Para marcar um registro para exclusão

- Em uma janela **Pesquisar**, clique sobre o marcador de exclusão para ativar o sinalizador do registro.
  - Ou –
- No menu **Tabela**, escolha **Excluir registros**.
  - Ou –
- Utilize o comando **DELETE - SQL**.

Você pode utilizar o comando **DELETE -SQL** para especificar um intervalo de registros, assim como uma condição baseada em uma expressão lógica que os registros devem atender para serem marcados para exclusão. Por exemplo, o código abaixo marca para exclusão todos os registros de produtos com .T. no campo `Discontinuo`:

```
USE products
DELETE FROM products WHERE discontinuo = .T.
BROWSE
```

Os registros marcados para exclusão somente serão removidos fisicamente da tabela quando você emitir um comando **PACK**. Ao visualizar a tabela na janela **Pesquisar**, você verá que o marcador de exclusão está ativado para todos os registros excluídos, porém o registro ainda estará visível na tabela, caso **SET DELETED** esteja desativado (OFF). Caso **SET DELETED** esteja ativado (ON), os registros marcados para exclusão não serão exibidos na janela **Pesquisar**.

A definição do comando **SET DELETED** também determina se os registros marcados para exclusão poderão ser acessados por comandos que operam em registros.

## Recuperando registros marcados para exclusão

Você pode desmarcar registros que foram marcados para exclusão, utilizando o comando **RECALL**. Esse comando somente pode recuperar os registros caso você não tenha emitido um comando **PACK** ou **ZAP**, que excluem fisicamente os registros da tabela.

### ► Para desmarcar um registro marcado para exclusão

- Em uma janela **Pesquisar**, clique no marcador de exclusão para desmarcar o registro.
  - Ou –
- No menu **Tabela**, escolha **Reintegrar registros**.
  - Ou –
- Utilize o comando **RECALL**.

Você pode utilizar o comando **RECALL** para especificar um intervalo de registros, assim como uma condição baseada em uma expressão lógica que os registros devem atender para serem desmarcados para exclusão. Por exemplo, o código abaixo desmarca para exclusão todos os registros de produtos com .T. no campo `discontinuo`:

```
USE products
RECALL FOR discontinuo = .T.
BROWSE
```

Quando você visualizar a tabela na janela **Pesquisar**, verá que o marcador de exclusão não está ativado para os registros.

## Removendo registros marcados para exclusão

Após marcar registros para exclusão, você pode removê-los do disco de forma permanente através da interface ou da linguagem.

### ► Para remover do disco registros marcados para exclusão

- Em uma janela **Pesquisar**, escolha **Remover registros excluídos** no menu **Tabela**.
  - Ou –
- Utilize o comando **PACK**.

O comando PACK possui duas cláusulas: MEMO e DBF. Quando você emitir PACK sem a cláusula MEMO ou DBF, tanto os registros do arquivo da tabela quanto os registros do arquivo de memo associado serão removidos. Por exemplo, o código abaixo remove os registros marcados para exclusão:

```
USE customer EXCLUSIVE  
PACK
```

Para excluir somente os registros do arquivo da tabela, deixando o arquivo de memo intacto, utilize PACK DBF.

## Conservando espaço

As informações contidas nos **campos Memo** são armazenadas em um arquivo de memo associado, com o mesmo nome da tabela e a extensão .FPT. Se você quiser remover o espaço não utilizado do arquivo de memo, mas não quiser remover da tabela os registros marcados para exclusão, emita o comando **PACK** com a cláusula MEMO. Certifique-se de que você tem acesso exclusivo à tabela.

## Removendo todos os registros de uma tabela

Se você desejar remover todos os registros de uma tabela, deixando somente a estrutura da tabela, utilize o comando **ZAP**. Este comando equivale ao comando DELETE ALL seguido por PACK, porém ZAP é muito mais rápido. Certifique-se de que você tem acesso exclusivo à tabela.

---

**Cuidado** Os registros removidos da tabela atual com ZAP não poderão ser reintegrados.

---

## Indexando tabelas

Para navegar, visualizar ou manipular registros da tabela em uma determinada ordem, você deve utilizar um índice. O Visual FoxPro utiliza índices como mecanismos de ordenação para proporcionar flexibilidade e capacidade à medida que você desenvolve seu aplicativo. Você tem a flexibilidade de criar e utilizar diversas chaves de índice para a mesma tabela, permitindo que você trabalhe com registros em diferentes ordens, de acordo com as necessidades de seu aplicativo. Você tem a capacidade de criar relacionamentos personalizados entre tabelas, com base nos índices, permitindo que você acesse somente os registros desejados.

Um índice do Visual FoxPro é um arquivo contendo ponteiros que são ordenados de forma lógica de acordo com os valores de uma chave de índice. O arquivo de índice é distinto do arquivo .DBF da tabela e não altera a ordem física dos registros na tabela. Em vez disso, ao criar um índice, você cria um arquivo que mantém ponteiros para os registros do arquivo .DBF. Quando deseja trabalhar com os registros da tabela em uma determinada ordem, você escolhe um índice para controlar a ordem e aumentar a velocidade na qual a tabela é visualizada e acessada.

## Criando um índice

Quando você cria uma tabela, o Visual FoxPro cria o arquivo .DBF da tabela e, caso sua tabela inclua campos do tipo **Memo** ou **Geral**, o arquivo .FPT associado. Nenhum arquivo de índice será gerado neste momento. Os registros digitados na nova tabela serão armazenados na ordem em que você os digitar; quando você pesquisar a nova tabela, serão exibidos na ordem em que foram



digitados.

É geralmente útil visualizar e acessar os registros de uma tabela em uma ordem específica. Por exemplo, você talvez queira visualizar os registros da tabela de clientes em ordem alfabética pelo nome da empresa. Para controlar a ordem na qual os registros serão exibidos e acessados, você deve criar um arquivo de índice para a tabela, criando o primeiro cenário de ordenação, ou chave de índice, para ela. Você pode então ordenar a tabela de acordo com a nova chave de índice e acessar os registros da tabela na ordem nova.

#### ► Para criar uma chave de índice para uma tabela

- No **Criador de tabelas**, escolha a guia **Índice** e digite as informações para uma chave de índice. Escolha **Normal** para o tipo do índice.  
– Ou –
- Utilize o comando **INDEX**.

Por exemplo, o código abaixo utiliza a tabela `customer` e cria uma chave de índice utilizando o campo `city`. A palavra-chave `TAG` e a palavra “city” depois dela especificam um nome, ou marca, para a nova chave de índice utilizando o campo `city`.

```
USE customer  
INDEX ON city TAG city
```

No exemplo acima, a marca para a chave de índice utiliza o mesmo nome do campo indexado. Os nomes não precisam ser correspondentes—você poderia ter escolhido outro nome para a chave de índice.

Quando você cria um índice utilizando o comando **INDEX**, o Visual FoxPro utiliza automaticamente o novo índice para definir a ordem dos registros na tabela. Por exemplo, caso você tenha digitado alguns dados na tabela criada no exemplo acima e depois pesquisado a tabela, os registros seriam exibidos em ordem de `city`.

## Criando um arquivo de índice

Ao criar a primeira chave de índice para sua tabela no exemplo acima, o Visual FoxPro criará automaticamente um novo arquivo, `CUSTOMER.CDX`, para armazenar a nova chave de índice. O arquivo de índice `.CDX`, denominado [índice composto estrutural](#), é o tipo mais comum e mais importante de arquivo de índice que você criará no Visual FoxPro. O arquivo `.CDX` estrutural:

- É aberto automaticamente quando você abre uma tabela.
- Pode conter múltiplos cenários de ordenação, ou chaves de índice, no mesmo arquivo de índice.
- É mantido automaticamente quando você adiciona, altera ou exclui registros da tabela.

Se uma tabela do Visual FoxPro possuir algum arquivo de índice associado a ela, geralmente será um arquivo `.CDX` estrutural. O termo “estrutural” se refere ao fato de que o Visual FoxPro trata o arquivo como uma parte intrínseca da tabela, abrindo-o automaticamente quando você utiliza uma tabela. Se você utilizar o **Criador de tabelas** ou a forma mais simples do comando **INDEX**, conforme mostrado no exemplo anterior, o Visual FoxPro criará o arquivo `.CDX` com o mesmo nome da tabela atual e armazenará as informações de indexação da nova chave, ou marca, dentro dele. Os arquivos `.CDX` estruturais são empregados para chaves de índice frequentemente utilizadas, como para ordenar registros para visualização diária, entrada de dados, criação de vínculos utilizando **SET RELATION**, otimização, [Rushmore™](#) para visualização de registros ou relatórios impressos com frequência.

O Visual FoxPro possui dois outros tipos de arquivos de índice: o arquivo `.CDX` não-estrutural e o arquivo `.IDX` de chave única. Uma vez que o `.CDX` (índice composto estrutural compacto) é o tipo de índice mais importante, a maioria dos exemplos desta seção abordará a utilização de chaves de índice nos arquivos `.CDX` para ordenar registros de tabela. Os outros dois tipos de arquivo de índice são menos utilizados, e serão abordados no final desta seção.

## Visualizando informações sobre o índice

Você pode ver quantos registros são indexados durante o processo de indexação ativando **TALK**. O intervalo entre registros exibido durante a indexação pode ser especificado com **SET ODOMETER**. Para obter maiores informações sobre arquivos de índice abertos, utilize o comando **DISPLAY STATUS**. Esse comando lista os nomes de todos os arquivos de índice abertos, seus tipos (estrutural, .CDX, .IDX), suas expressões de indexação e o nome do arquivo de índice mestre ou marca mestre.

O número de arquivos de índice (.IDX ou .CDX) que podem ser abertos somente é limitado pela memória e recursos do sistema.

## Controlando valores duplicados

O Visual FoxPro aceita quatro tipos de índice: primário, candidato, único e normal. Esses tipos de índice determinam se valores duplicados são ou não permitidos em campos e registros da tabela.

### Evitando valores duplicados

Um índice primário é um índice que nunca permite a existência de valores duplicados nos campos ou na expressão especificada. Os índices primários são utilizados sobretudo dentro da tabela primária ou “referenciada” para estabelecer a integridade referencial em um relacionamento permanente. Você somente pode criar um único índice primário para uma tabela. O Visual FoxPro retornará um erro caso seja especificado um índice primário em qualquer campo que já contenha dados duplicados.

Um índice candidato é um índice que nunca permite valores duplicados nos campos ou expressão especificada. O nome “Candidato” se refere ao status do índice; uma vez que os índices candidatos não permitem a duplicação de valores, são considerados “candidatos” para serem selecionados como índice primário de uma tabela.

Você pode criar diversos índices candidatos para uma tabela. Índices candidatos são utilizados tanto como índice referenciado quanto como índice de referência em um relacionamento permanente para estabelecer a integridade referencial.

O Visual FoxPro retornará um erro caso seja especificado um índice candidato para qualquer campo que já contenha dados duplicados.

### Definindo um índice primário ou candidato

Os índices primários e candidatos são criados com os comandos **CREATE TABLE** ou **ALTER TABLE**. Você pode utilizar esses dois tipos de índice ao definir o lado “um” de um relacionamento permanente um-para-n ou um-para-um.

#### ► Para criar um índice primário ou candidato

- No **Criador de tabelas**, escolha a guia **Índice** e crie um índice, selecionando **Primário** ou **Candidato** como tipo do índice.
  - Ou –
- Utilize o comando **ALTER TABLE**.

Por exemplo, ambos os comandos a seguir tornam `cust_id` a chave primária da tabela `customer`:

```
ALTER TABLE customer ADD PRIMARY KEY cust_id TAG cust_id
ALTER TABLE customer ALTER COLUMN cust_id c(5) PRIMARY KEY
```

Os índices primários e candidatos são armazenados no arquivo .CDX estrutural para uma dada tabela, assim como no banco de dados com as propriedades “Primário” ou “Candidato”. Não é possível armazenar estes tipos de índice em outros arquivos .CDX, assim como não é possível utilizar arquivos .IDX para esses tipos de índice. A razão principal para isso é que o arquivo de índice contendo estes índices deve sempre ser aberto quando a tabela associada for aberta.

As [chaves primárias](#) são parte de uma tabela em um banco de dados. Se você liberar uma tabela de um banco de dados, a chave primária será removida.

Se você utilizar uma [função definida pelo usuário](#) em uma expressão de indexação associada a um banco de dados, o Visual FoxPro tratará a expressão da mesma forma como trata as expressões de [regras](#) e [disparadores](#) que contêm UDFs.

## Permitindo valores duplicados

No Visual FoxPro, um [índice único](#) não impede que sejam criados valores duplicados. Em vez disso, um índice único armazena somente a primeira ocorrência dos valores no arquivo de índice. Neste sentido, a palavra “único” diz respeito somente às entradas do arquivo de índice, que contém somente valores únicos, já que nunca armazena uma determinada chave mais do que uma vez, ignorando as ocorrências posteriores de um valor não-único. Uma tabela indexada por um índice único pode conter valores duplicados. Os tipos de índice únicos são fornecidos basicamente para fins de compatibilidade com versões anteriores.

Um [índice normal](#) é simplesmente um índice que não é único, nem primário, nem candidato. Os índices normais são utilizados para ordenar e procurar registros, mas não para garantir que os dados nesses registros sejam únicos. Você também pode utilizar um índice normal como o lado n de um relacionamento permanente [um-para-n](#).

### ► Para criar um índice normal

- No **Criador de tabelas**, escolha a guia **Índice** e crie um índice, selecionando **Normal** como tipo do índice.
  - Ou –
- Utilize o comando [INDEX](#).

Por exemplo, os comandos a seguir tornam `city` uma chave normal para a tabela `customer`:

```
USE customer
INDEX ON city TAG city
```

## Criando vários índices

À medida que for trabalhando com os registros de sua tabela, você descobrirá a necessidade de acessar os registros da tabela em várias seqüências diferentes. Por exemplo, você pode querer ordenar a tabela `customer` por telefone, para encontrar rapidamente um nome, ou por código postal, para gerar etiquetas de mala direta já ordenadas e facilitar a postagem.

Para criar e armazenar diversos cenários de ordenação para sua tabela, basta criar diversas chaves de índice para a mesma tabela. Isso permite que, em momentos distintos, você ordene os registros da tabela de acordo com suas diferentes necessidades.

### ► Para criar chaves de índice adicionais para uma tabela

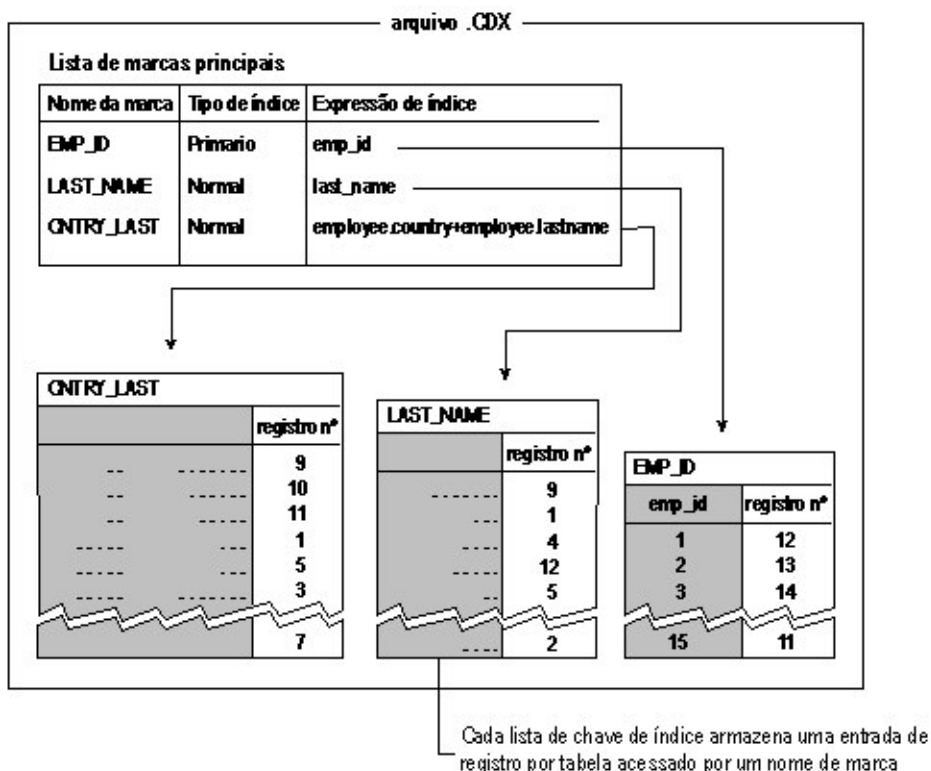
- No **Criador de tabelas** escolha a guia **Índice** e digite as informações relativas às chaves de índice adicionais.
  - Ou –
- Utilize o comando [INDEX](#).

Por exemplo, o código a seguir cria duas novas chaves de índice na tabela `employee`: uma no campo `last_name` e outra no campo `country`.

```
USE employee
INDEX ON last_name TAG last_name
INDEX ON country TAG country
```

Quando você criar uma marca de índice sem especificar o nome de um arquivo de índice, a marca será automaticamente acrescentada ao arquivo de índice `.CDX` estrutural da tabela. O diagrama a seguir mostra um arquivo de índice `.CDX` com três marcas de índice.

O índice .CDX contém diversas marcas representando diversos cenários de ordenação



Duas das marcas no diagrama, `emp_id` e `last_name`, representam índices baseados em campos individuais. O índice `cntry_last` ordena os registros utilizando uma expressão de indexação simples de dois campos. Para obter maiores informações sobre como construir um índice baseado em mais de um campo, consulte [“Indexando por expressões”](#), posteriormente neste capítulo.

## Controlando a ordem na qual os registros são acessados

Após criar chaves de índice para a tabela `customer` utilizando os campos `company`, `city` e `country`, você pode acessar e exibir a tabela utilizando ordens diferentes. Basta escolher a chave de índice preferida. Utilize o comando **SET ORDER** para escolher uma chave de índice específica como chave de ordenação para a tabela.

Por exemplo, o código abaixo abre uma [janela Pesquisar](#), exibindo os registros da tabela `customer` por país:

```
SET ORDER TO country
BROWSE
```

## Definindo a ordem dos registros em tempo de execução

O comando **SET ORDER** permite que você especifique o arquivo ou marca de índice que estará no controle. Vários arquivos de índice podem estar abertos ao mesmo tempo para somente uma tabela. Contudo, a ordem na qual os registros serão exibidos ou acessados será determinada pelo arquivo de índice único (.IDX) (o arquivo de índice de controle) ou marca de um arquivo de índice composto (.CDX) (a marca de controle) que for definida como índice de controle. Alguns comandos, como **SEEK**, utilizam a marca de índice de controle para procurar registros. Você não precisa de **SET ORDER** para executar [consultas](#).

## Definindo a ordem dos registros de forma interativa em um formulário

Você pode utilizar **SET ORDER** em [tempo de execução](#) para alterar a ordem dos registros em um [formulário](#). Por exemplo, você talvez queira permitir que os usuários de seu aplicativo reordenem os registros em uma [grade](#) clicando no cabeçalho da coluna pela qual eles querem ordenar.

### ► Para ordenar por colunas os registros em uma grade

- 1 Crie um formulário com um controle [Grid](#).
- 2 Defina a propriedade [ColumnCount](#) da grade como o número de campos a serem exibidos na grade.
- 3 No evento [Click](#) do cabeçalho de cada coluna da grade, insira um código que:
  - Defina a ordem dos registros utilizando uma chave de índice baseada na coluna.
  - Atualize o formulário.

Por exemplo, se você criar um formulário baseado na tabela Customer do banco de dados Testdata com uma grade contendo as quatro colunas — `company`, `contact`, `postal code` e `phone` — a grade estará inicialmente classificada em ordem alfabética, já que os registros dessa tabela foram digitados nessa ordem.

### Tabela Customer em uma grade, em ordem alfabética por nome de empresa



	Company	Contact	Postal Code	Telephone
	Alfreds Futterkiste	Maria Anders	12209	030-0074321
	Ana Trujillo Emparedados y	Ana Trujillo	05021	(5) 555-4729
	Antonio Moreno Taquería	Antonio Moreno	05023	(5) 555-3932
	Around the Horn	Thomas Hardy	WA1 1DP	(71) 555-7789
	Berglunds snabbköp	Christina Berglund	S-958 22	0921-12 34 6
	Blauer See Delikatessen	Hanna Moos	68306	0621-08460
	Blondel père et fils	Frédérique Citeaux	67000	88.60.15.31
	Bólido Comidas preparadas	Martín Sommer	28023	(91) 555 22 8
	Bon app'	Laurence Lebihan	13008	91.24.45.40
	Bottou Deller-Markets	Elizabeth Lincoln	T3F 0M4	(604) 555 47

Para permitir que o usuário visualize a grade em ordem de `contact` ou `postal_code`, insira o código abaixo no evento [Click](#) do cabeçalho de cada coluna:

### Exemplo de código de evento para ordenar registros em uma Grade clicando no cabeçalho da coluna

Código	Comentário
SET ORDER TO company GO TOP THISFORM.Refresh	No código de evento <a href="#">Click</a> do cabeçalho de <code>Company</code> , reordena a grade utilizando a chave de índice <code>company</code> e atualiza o formulário para exibir os registros ordenados por empresa.
SET ORDER TO contact GO TOP THISFORM.Refresh	No código de evento <a href="#">Click</a> do cabeçalho de <code>Contact</code> , reordena a grade utilizando a chave de índice <code>contact</code> e atualiza o formulário para exibir os registros ordenados por nome de contato.
SET ORDER TO postalcode	No código de evento <a href="#">Click</a> do cabeçalho de

GO TOP  
THISFORM.Refresh

a chave de índice `postalcode` e atualiza o formulário para exibir os registros ordenados por código postal.

Já que uma classificação por número de telefone não é relevante para este aplicativo, deixe o código de evento Click do cabeçalho de `Phone` em branco.

Neste exemplo, quando o [formulário](#) for exibido inicialmente, a [grade](#) será exibida em ordem alfabética por empresa. Quando o usuário clicar sobre o cabeçalho da coluna `Contact`, o Visual FoxPro exibirá os registros na grade em ordem alfabética por nome de contato.

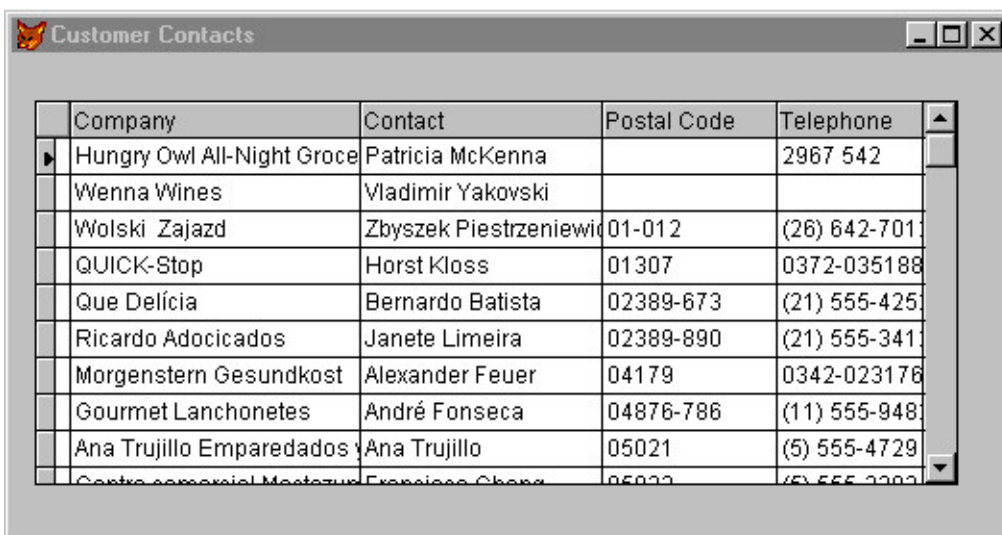
#### A tabela **Customer** na grade, reordenada em ordem alfabética por nome de contato



	Company	Contact	Postal Code	Telephone
	Romero y tomillo	Alejandra Camino	28001	(91) 745 6200
	Morgenstern Gesundkost	Alexander Feuer	04179	0342-023176
	Ana Trujillo Emparedados	Ana Trujillo	05021	(5) 555-4729
	Tradição Hipermercados	Anabela Domingues	05634-030	(11) 555-2167
	Gourmet Lanchonetes	André Fonseca	04876-786	(11) 555-9483
	Eastern Connection	Ann Devon	WX3 6FW	(71) 555-0297
	La maison d'Asie	Annette Roulet	31000	61.77.61.10
	Antonio Moreno Taquería	Antonio Moreno	05023	(5) 555-3932
	Familia Arquibaldo	Aria Cruz	05442-030	(11) 555-9857
	Spit-Nail Book & More	Art Breuninger	03520	(203) 555-4671

Se o usuário clicar no cabeçalho da coluna `Postal_code`, a grade será reordenada e exibida por código postal.

#### A tabela **Customer** na grade, reordenada pelo código postal



	Company	Contact	Postal Code	Telephone
	Hungry Owl All-Night Groce	Patricia McKenna		2967 542
	Wenna Wines	Vladimir Yakovski		
	Wolski Zajazd	Zbyszek Piestrzeniewicz	01-012	(26) 642-7011
	QUICK-Stop	Horst Kloss	01307	0372-035188
	Que Delícia	Bernardo Batista	02389-673	(21) 555-4250
	Ricardo Adocicados	Janete Limeira	02389-890	(21) 555-3411
	Morgenstern Gesundkost	Alexander Feuer	04179	0342-023176
	Gourmet Lanchonetes	André Fonseca	04876-786	(11) 555-9483
	Ana Trujillo Emparedados	Ana Trujillo	05021	(5) 555-4729
	Centro comercial Mestizo	Fernando Cheng	05022	(5) 555-3302

Já que não há nenhuma necessidade aparente de classificar os contatos por número de telefone no aplicativo exemplificado, não será inserido código de [SET ORDER](#) no evento Click para o cabeçalho da coluna `phone`. Quando o usuário clicar no cabeçalho de coluna `Phone`, a exibição da grade não



será alterada.

## Utilizando outros tipos de índice

Além do índice mais comum — o índice .CDX composto estrutural compacto — o Visual FoxPro aceita dois outros tipos de arquivos de índice: o .CDX não-estrutural e o índice .IDX independente. Os índices .CDX não-estruturais são utilizados para marcas de chave múltipla utilizadas com menos frequência. Os índices independentes, ou .IDX, são utilizados para índices de chave única temporários ou que sejam raramente utilizados, e estão disponíveis basicamente para fins de compatibilidade com versões anteriores.

A tabela a seguir é um resumo dos três tipos de índice, seus nomes, o número de chaves que podem conter e as limitações no número de caracteres de cada um.

### Tipos de índice do Visual FoxPro

Tipo de índice	Descrição	Número de chaves	Limites
.CDX estrutural	Utiliza o mesmo nome do arquivo de tabela; é aberto automaticamente com a tabela	Expressões utilizando chaves múltiplas, chamadas de marcas	Limite de 240 caracteres para a expressão avaliada
.CDX não-estrutural	Deve ser aberto explicitamente; o nome difere do nome da tabela	Expressões utilizando chaves múltiplas, chamadas de marcas	Limite de 240 caracteres para a expressão avaliada
.IDX independente	Deve ser aberto explicitamente; o nome do arquivo .IDX é definido pelo usuário	Expressão utilizando chave única	Limite de 100 caracteres para a expressão avaliada

## Utilizando índices .CDX não-estruturais

Um índice .CDX não-estrutural é útil quando você deseja criar várias marcas de índice com uma finalidade especial, porém não quer sobrecarregar seu aplicativo com a manutenção regular desses índices. Por exemplo, o aplicativo pode conter um conjunto especial de relatórios que analisam os dados com base em campos que, normalmente, não estão indexados. O programa aplicativo pode criar um índice .CDX não-estrutural com as marcas de índice necessárias, executar os relatórios especiais e depois excluir o arquivo .CDX não-estrutural.

### ► Para criar uma marca de índice .CDX não-estrutural

- Utilize o comando **INDEX** com as cláusulas TAG e OF.

A cláusula OF é utilizada com o comando INDEX para instruir o Visual FoxPro a armazenar a marca em outro arquivo que não o arquivo de índice .CDX estrutural da tabela. Por exemplo, o comando a seguir cria marcas denominadas `title` e `hire_date` na tabela `employee` e armazena-as em um arquivo .CDX não-estrutural denominado `QRTLYRPT.CDX`:

```
USE employee
INDEX ON title TO TAG title OF QRTLYRPT
INDEX ON hire_date TO TAG hiredate OF QRTLYRPT
```

## Utilizando índices independentes



O arquivo de índice independente, baseado em uma expressão de chave única, é armazenado como um arquivo .IDX. Ao contrário dos índices .CDX, que podem armazenar expressões utilizando chaves múltiplas, o índice .IDX armazena somente uma única expressão chave.

Os índices independentes são geralmente utilizados como índices temporários, criados ou reindexados quando for necessário usá-los. Por exemplo, você pode ter um índice utilizado somente para um relatório de resumo trimestral ou anual. Em vez de incluir este índice pouco utilizado no .CDX estrutural, onde seria mantido todas as vezes em que você utilizasse a tabela, é possível criar um índice independente .IDX. Você pode criar quantos arquivos .IDX quiser para uma determinada tabela.

#### ► Para criar um índice independente .IDX

- Utilize a cláusula COMPACT do comando [INDEX](#).

– Ou –

- Utilize o comando [COPY TAG](#).

O comando INDEX, utilizado com a cláusula COMPACT, cria um novo índice independente em um arquivo de índice pequeno e que pode ser acessado rapidamente. Você pode omitir a cláusula COMPACT se quiser criar um arquivo .IDX independente não-compacto para manter a compatibilidade com os antigos formatos de índice do FoxBASE+® e FoxPro® versão 1.0.

O código a seguir cria um arquivo .IDX independente utilizando `order_date` na tabela `orders`, ordena a tabela pelo novo índice e depois abre uma [janela Pesquisar](#) exibindo os pedidos em ordem de `order_date`:

```
USE ORDERS
INDEX ON order_date TO orddate COMPACT
SET ORDER TO orddate
BROWSE
```

Você pode utilizar o comando COPY TAG para gerar um arquivo de índice independente a partir de uma marca de índice em um arquivo .CDX já existente. Por exemplo, você pode descobrir que um dos índices atualmente mantidos no .CDX estrutural somente é utilizado para fazer relatórios trimestrais ou anuais. O código abaixo cria um índice independente a partir de uma marca `birth_date` na tabela `employee`:

```
COPY TAG birth_date to birthdt COMPACT
```

Após criar um índice independente a partir de uma marca em um arquivo .CDX, normalmente a marca do arquivo .CDX, que agora tornou-se desnecessária, deve ser excluída. A próxima seção explica como excluir um índice.

## Excluindo um índice

Você pode excluir índices que não estão mais sendo utilizados. Para fazer isso, exclua a marca dentro do arquivo .CDX, ou exclua o próprio arquivo .IDX, no caso de índices independentes. A exclusão de marcas de índice melhora o desempenho, pois o Visual FoxPro não precisa atualizar marcas não utilizadas para refletir as mudanças nos dados de uma tabela.

## Excluindo uma marca do arquivo .CDX estrutural

Você pode excluir uma marca do arquivo .CDX estrutural utilizando o **Criador de tabelas** ou a linguagem.

#### ► Para excluir uma marca de índice do .CDX estrutural

- No **Criador de tabelas**, utilize a guia **Índice** para selecionar e excluir o índice.

– Ou –

- Utilize o comando [DELETE TAG](#).

– Ou –

- Utilize as cláusulas DROP PRIMARY KEY ou DROP UNIQUE TAG do comando [ALTER TABLE](#). Por exemplo, se a tabela `employee` contiver uma marca denominada `title`, será possível excluí-la utilizando o código a seguir:

```
USE employee
DELETE TAG title
```

Se a marca a ser excluída fosse a chave primária da tabela `employee`, você poderia utilizar o comando `ALTER TABLE`:

```
USE employee
ALTER TABLE DROP PRIMARY KEY
```

## Excluindo uma marca de um arquivo .CDX não-estrutural

Um índice .CDX não-estrutural e suas marcas não são visíveis no [Criador de tabelas](#). Utilize a linguagem para excluir uma marca de um arquivo .CDX não-estrutural.

### ► Para excluir um índice de um arquivo .CDX não-estrutural

- Utilize a cláusula OF do comando [DELETE TAG](#).

Utilize a cláusula OF com o comando `DELETE TAG` para instruir o Visual FoxPro a excluir uma marca de um arquivo .CDX que não seja o .CDX estrutural. Por exemplo, caso haja um arquivo .CDX não-estrutural denominado `QRTLYRPT.CDX` com uma marca denominada `title`, a marca `title` poderá ser excluída com o comando a seguir:

```
DELETE TAG title OF qtrlyrpt
```

Para excluir todas as marcas de um arquivo .CDX estrutural ou não-estrutural, utilize a cláusula `ALL` do comando `DELETE TAG`.

## Excluindo um arquivo de índice .IDX independente

Já que um arquivo de índice independente contém uma única expressão de chave de índice, para excluir a expressão, basta excluir o arquivo .IDX do disco.

### ► Para excluir um arquivo .IDX

- Utilize o comando [DELETE FILE](#).

Por exemplo, o código a seguir exclui o arquivo de índice .IDX independente `ORDDATE.IDX`:

```
DELETE FILE orddate.idx
```

Você pode também utilizar um utilitário como o Windows Explorer para excluir um arquivo .IDX independente desnecessário.

## Indexando por expressões

Você pode aumentar o poder de seus aplicativos criando índices baseados em expressões. Essas expressões podem ser simples ou complexas, dependendo do que você quer fazer.

### Indexando por expressões simples

Expressões de índice simples são índices baseados em campos únicos ou na concatenação de dois ou mais campos de caractere para formar uma chave de campos múltiplos. Por exemplo, você pode querer criar um índice para a tabela `Customer` no banco de dados `TasTrade` baseado na expressão:

```
country + region + cust_id
```

Ao [pesquisar](#) a tabela `Customer` utilizando esta marca de índice, você verá os clientes ordenados por país, região e número de identificação.

### Utilizando uma expressão para impedir duplicatas em uma combinação de campos

Para impedir a duplicação de valores em uma combinação de campos, você pode criar um índice

[primário](#) ou [candidato](#) baseado em uma expressão que combine esses campos.

Por exemplo, você pode ter uma tabela que armazene o código DDD e o número de telefone em duas colunas:

Código DDD	Número de telefone
206	444-nnnn
206	555-nnnn
313	444-nnnn


Tanto o campo de código DDD quanto o de número de telefone contêm valores duplicados. Contudo, não existem números de telefone duplicados, porque a combinação dos dois campos forma um valor separado. Se o índice primário ou candidato especificar ambas as colunas na expressão do índice, as linhas do exemplo não seriam consideradas duplicadas. Se você tentasse inserir um valor que correspondesse exatamente ao mesmo código de área e número de telefone de uma das linhas existentes, o Visual FoxPro rejeitaria a entrada como sendo duplicada.

### Utilizando valores nulos nas expressões de índice

Você pode criar índices utilizando campos que contenham [valores nulos](#). As expressões de índice que resultarem em .NULL. serão inseridas no arquivo .CDX ou .IDX antes das entradas não-nulas. Todos os valores nulos serão colocados no início do índice.

O exemplo a seguir demonstra um dos efeitos de se indexar valores nulos. Este é o estado da tabela antes de o índice ser aplicado:

#### Há valores nulos no campo SocSec de dois registros



Record	Lastname	Firstname	Socsec
1	Silva	Sergio	111-000-2222
2	Pereira	Paulo	555-22-9999
3	Machado	Eduardo	.NULL.
4	Moreira	Marcos	222-33-4444
5	Pachiardi	Lucio	.NULL.

O valor .NULL. em dois registros representa o fato de que os números de previdência social de Anne Dunn e Alan Carter são desconhecidos ou não estão disponíveis. Você cria, então, um índice com o número de previdência social utilizando o exemplo a seguir:

```
INDEX ON SocSec + LastName + FirstName TAG MyIndex
```

Ao visualizar a tabela classificada por este índice, a ordem de classificação aparecerá como na próxima figura.

#### Depois de indexar por SocSec, os registros contendo valores de SocSec nulos aparecem primeiro



Record	Lastname	Firstname	Socsec
3	Machado	Eduardo	.NULL.
5	Pachiardi	Lucio	.NULL.
1	Silva	Sergio	111-000-2222
2	Pereira	Paulo	555-22-9999
4	Moreira	Marcos	222-33-4444

Quando a expressão de índice contém valores nulos, os registros cujos valores de SocSec são .NULL. são classificados primeiro (por LastName), seguidos pelos registros cujos valores de

SocSec não são nulos. Repare que há duas entradas para Alan Carter. Como o registro 5 contém um valor nulo, ele está indexado antes do registro 2.

## Indexação utilizando expressões complexas

Você também pode criar índices baseados em expressões mais complexas. As expressões de chave de índice do Visual FoxPro podem incluir as [funções](#) do Visual FoxPro, [constantes](#) ou [funções definidas pelo usuário](#).

O resultado da expressão criada deve ter menos de 100 caracteres para um índice independente (.IDX) ou menos de 240 caracteres para uma marca de índice .CDX. Para utilizar campos de [tipos de dados](#) diferentes juntos em uma única marca, converta os componentes individuais da expressão em dados de caractere.

Para tirar vantagem da otimização [Rushmore](#), a expressão de índice deve coincidir exatamente com o critério.

## Utilizando as funções do Visual FoxPro em uma marca de índice

As funções do Visual FoxPro podem ser utilizadas em uma marca de índice. Por exemplo, é possível utilizar a função `STR()` para converter um valor numérico em uma seqüência de caracteres. Para criar uma marca de índice para a tabela `customer` que combinasse o campo `cust_id` com o campo `maxordamt`, você poderia converter o campo `maxordamt` do tipo Moeda, com uma largura de 8, em um campo de 8 caracteres com 2 casas decimais, utilizando o código abaixo:

```
INDEX ON cust_id + STR(maxordamt, 8, 2) TAG custmaxord
```

Se você quer reduzir o tamanho dos índices para campos com valores inteiros, você pode converter os valores inteiros em uma representação de um caractere binário utilizando a função `BINTOC()`. Você pode também converter os valores binários em valores inteiros por meio da função `CTOBIN()`.

Se você deseja criar um índice para classificar uma tabela em ordem cronológica, é possível utilizar a função `DTOS()` para converter um campo de data em uma seqüência de caracteres. Para acessar a tabela `employee` por `hire_date` e `emp_id`, você pode criar a expressão de chave de índice a seguir:

```
INDEX ON DTOS(hire_date) + emp_id TAG id_hired
```

## Incluindo procedimentos armazenados ou funções definidas pelo usuário

Para aumentar o poder de seu índice, você pode fazer com que a expressão de índice faça referência a um [procedimento armazenado](#) ou uma [função definida pelo usuário](#). Por exemplo, você pode utilizar um procedimento armazenado ou uma UDF para extrair um nome de rua de um campo que inclua tanto o número quanto o nome da rua. Se o número de rua for sempre numérico, o procedimento armazenado ou UDF poderão retornar a parte do campo composta por caracteres e preencher o campo com o número de espaços necessários para criar uma chave de índice de comprimento constante. Essa chave de índice pode, então, ser utilizada para acessar registros na tabela ordenados pelo nome das ruas.

Caso sua tabela esteja associada a um banco de dados, você talvez prefira utilizar um procedimento armazenado em vez de uma UDF na marca de índice. Uma vez que as UDFs são armazenadas em um arquivo separado do banco de dados, você pode mover ou excluir o arquivo da UDF, o que faria com que a marca de índice que fizer referência a essa UDF se tornasse inválida. Por outro lado, o código dos procedimentos armazenados é armazenado no arquivo .DBC, podendo sempre ser localizado pelo Visual FoxPro.

Outra vantagem de utilizar um procedimento armazenado em uma marca de índice é que referenciar um procedimento armazenado garante que o índice será baseado no código especificado. Caso você utilize uma UDF na expressão de índice, qualquer UDF que se encontre no escopo no momento da indexação e tenha o mesmo nome que a UDF referenciada em seu índice será utilizada.

**Observação** Fique atento ao fazer referência a um procedimento armazenado ou UDF em uma

expressão de índice, uma vez que isso aumenta o tempo necessário para criar o índice.

## Utilizando dados de um campo de outra tabela em uma marca de índice

Você pode criar uma marca de índice que se refira a uma tabela aberta em outra [área de trabalho](#). É conveniente utilizar um índice independente (.IDX) para qualquer marca que se refira a mais de uma tabela. Isso porque, se você incluísse uma marca que fizesse referência a outra tabela em um arquivo .CDX estrutural, o Visual FoxPro não permitiria que você abrisse a tabela antes de ter aberto a tabela referenciada na marca de índice.

## Acessando registros em ordem descendente

Para visualizar os registros em ordem descendente, crie um índice descendente ou leia um índice já existente em ordem descendente.

### ► Para criar um índice descendente

- Na guia **Índice** do **Criador de tabelas**, escolha o botão de direção à esquerda da caixa **Nome de modo** que a seta fique apontando para baixo.
  - Ou –
- Utilize a cláusula DESCENDING com o comando **INDEX ON** para criar um índice descendente.

Para criar arquivos de índice estrutural composto, você pode utilizar qualquer um desses métodos. Para criar outros tipos de arquivos de índice, você pode utilizar o segundo método. Por exemplo, você poderia criar um novo índice descendente ordenando sua tabela `product` do maior `unit_price` para o menor e pesquisar a tabela na nova ordem com o código a seguir:

```
USE products
INDEX ON unit_price TAG unit_price DESCENDING
BROWSE
```

### ► Para ler um índice existente em ordem descendente

- Utilize a cláusula DESCENDING do comando **SET ORDER** para ler um índice existente em ordem descendente.

Ler um índice já existente em ordem descendente permite que você empregue um índice já existente em vez de criar um novo. Por exemplo, talvez já haja um índice que ordene a tabela `product` por `unit_price`, criado com o código a seguir:

```
USE products
INDEX ON unit_price TAG unit_price
```

Como padrão, a ordem é ascendente. Você poderia pesquisar a tabela em ordem descendente com o código a seguir:

```
USE products
SET ORDER TO unit_price DESCENDING
BROWSE
```

Os exemplos anteriores se concentraram em como acessar as informações em ordem descendente. Tanto **SET ORDER** quanto **INDEX** também possuem uma cláusula ASCENDING. Você pode combinar esses dois comandos para ganhar uma enorme flexibilidade em seu aplicativo. Por exemplo, caso você utilize a cláusula ASCENDING ou DESCENDING para criar um índice de acordo com a ordem mais freqüentemente utilizada, você pode utilizar a cláusula oposta com o comando SET ORDER para visualizar ou acessar as informações na ordem inversa quando esta for mais conveniente.

## Filtrando dados

Você pode limitar os registros acessados somente aos dados que você deseja empregar, utilizando um índice filtrado. Ao criar um índice filtrado, somente os registros que corresponderem à expressão de filtro estarão disponíveis para exibição e acesso.

### ► Para filtrar dados utilizando um índice filtrado

- No **Criador de tabelas**, escolha a guia **Índice** e digite uma expressão de filtro na caixa **Filtro** para o índice que você deseja filtrar.  
– Ou –
- Utilize a cláusula opcional FOR com o comando **INDEX**.

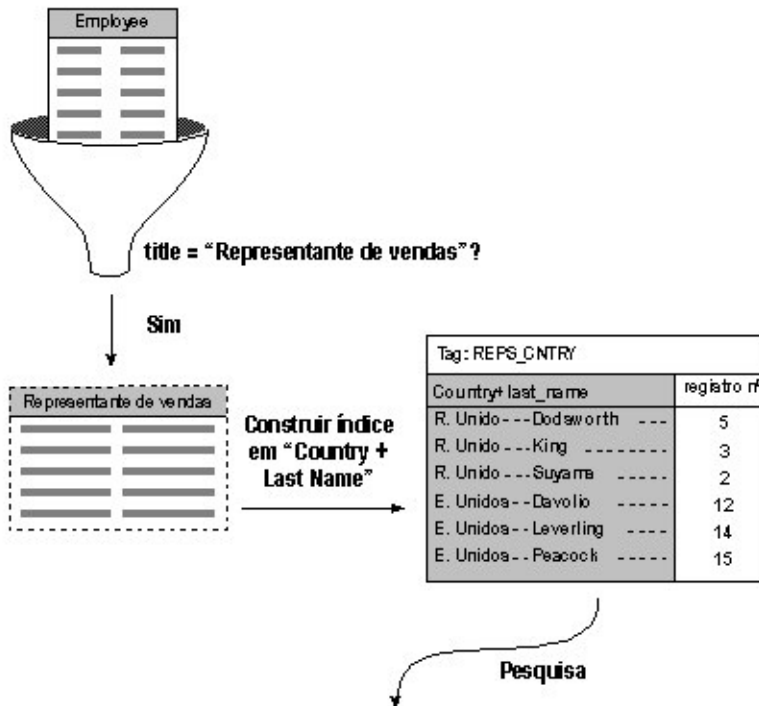
Caso você inclua a cláusula opcional FOR no comando INDEX, o arquivo de índice agirá como um filtro na tabela. Serão criadas chaves de índice no arquivo de índice somente para os registros que corresponderem à expressão de filtro. Por exemplo, caso esteja preparando uma mala direta dirigida aos representantes de vendas de sua empresa e deseje classificar essa correspondência por país, você poderá criar um índice que filtre a tabela `employee` de forma que somente os registros de representantes de vendas sejam exibidos, ordenados por país e sobrenome. O código a seguir cria um índice filtrado e exibe os dados filtrados em uma janela **Pesquisar**:

```
USE employee
INDEX ON country+last_name FOR title = "Sales Representative" ;
TAG reps_cntry
BROWSE
```

Ao visualizar a janela **Pesquisar**, somente os representantes de venda serão exibidos; os registros dos outros funcionários não aparecerão na janela **Pesquisar**.

**Um índice filtrado cria um índice somente para os registros que correspondem à expressão de filtro.**

**INDEX ON country+last\_name FOR title = "Representante de vendas" TAG reps\_cntry**

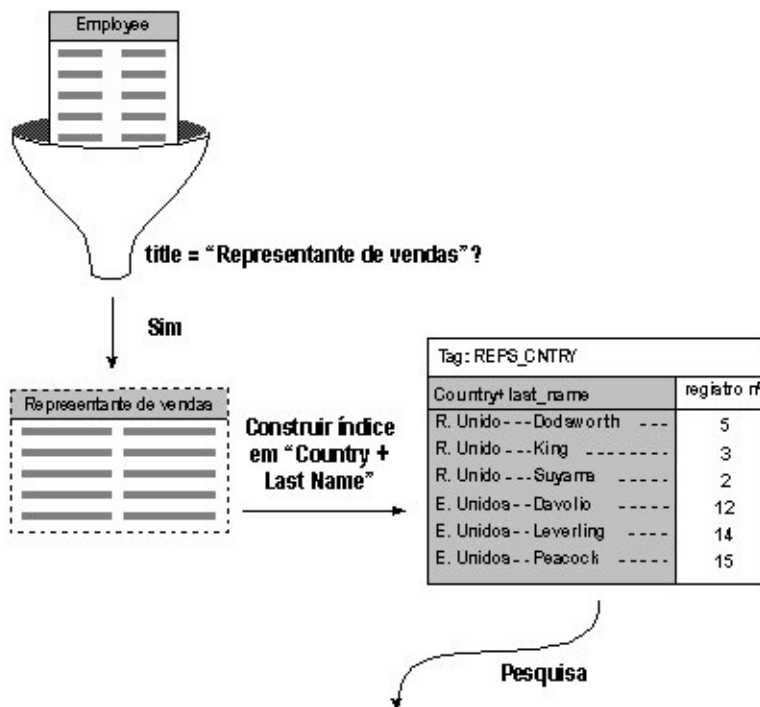


Pesquisa

	Last_name	First_name	Title	Address
▶	Davolio	Nancy	Representante comercial	507 - 20th Ave. E., Apt. 2A
	Dodsworth	Anne	Representante comercial	7 Houndstooth Rd.
	King	Robert	Representante comercial	Edgeham Hollow, Winchester Way
	Leverling	Janet	Representante comercial	722 Moss Bay Blvd.
	Peacock	Margaret	Representante comercial	4110 Old Redmond Rd.
	Suyama	Michael	Representante comercial	Coventry House, Miner Rd.



INDEX ON country+last\_name FOR title = "Representante de vendas" TAG reps\_cntry



Employee				
	Last_name	First_name	Title	Address
▶	Davolio	Nancy	Representante comercial	507 - 20th Ave. E., Apt. 2A
	Dodsworth	Anne	Representante comercial	7 Houndstooth Rd.
	King	Robert	Representante comercial	Edgeham Hollow, Winchester Way
	Leverling	Janet	Representante comercial	722 Moss Bay Blvd.
	Peacock	Margaret	Representante comercial	4110 Old Redmond Rd.
	Suyama	Michael	Representante comercial	Coventry House, Miner Rd.

## Filtrando dados temporariamente

Você pode utilizar o comando **SET FILTER** para filtrar dados temporariamente, sem criar um índice filtrado especial. Esse comando é particularmente útil quando você deseja especificar uma condição temporária que os registros de uma tabela devem satisfazer para serem acessados. Para desligar o filtro da tabela atual, você pode emitir SET FILTER TO sem uma expressão. Por exemplo, o comando a seguir seria emitido para filtrar a tabela `customer` de forma a mostrar somente os clientes da Alemanha:

```
USE customer
SET FILTER TO country = "Germany"
BROWSE
```

O comando SET FILTER aceita qualquer expressão lógica e válida do Visual FoxPro como condição de filtro. Uma vez que o comando SET FILTER tenha sido emitido, somente os registros que satisfizerem a condição de filtro estarão disponíveis na tabela. Todos os comandos que acessam a tabela respeitam a condição SET FILTER. Você pode definir um filtro separado para todas as tabelas abertas.

## Utilizando índices de forma eficiente

O desempenho das tabelas indexadas pode ser melhorado se os índices forem sempre atualizados



e se forem utilizadas expressões otimizáveis nos índices.

## Reconstruindo um arquivo de índice ativo

Os arquivos de índice ficam desatualizados quando você abre uma tabela sem abrir os arquivos de índice correspondentes e modifica os campos chave da tabela. Os arquivos de índice pode também tornar-se inválidos como resultado de uma falha no sistema ou quando uma tabela for acessada e atualizada utilizando outro programa que não o Visual FoxPro. Quando os arquivos de índice ficam desatualizados, você pode atualizá-los fazendo uma reindexação com o comando REINDEX.

### ► Para reconstruir um arquivo de índice ativo

- No menu **Tabela**, escolha **Reconstruir índices**.
  - Ou –
- Utilize o comando **REINDEX**.

Por exemplo, o código a seguir atualiza o arquivo de índice da tabela Customer:

```
USE customer  
REINDEX
```

REINDEX atualiza todos os arquivos de índice abertos na [área de trabalho](#). O Visual FoxPro reconhece cada tipo de arquivo de índice (arquivos .CDX de índice composto, arquivos .CDX estruturais e arquivos .IDX de índice único) e os reindexa da forma adequada. Ele atualiza todas as marcas nos arquivos .CDX e atualiza os arquivos .CDX estruturais, que são abertos automaticamente com a tabela.

Você pode também atualizar arquivos de índice desatualizados com o comando REINDEX.

## Reindexando em tempo de execução

Uma reindexação leva tempo, particularmente ao reindexar grandes tabelas. Reindexe somente quando necessário. Você pode melhorar o desempenho se a reindexação for feita durante a inicialização ou finalização de seu programa, em vez de ser executada durante a parte principal de um aplicativo.

## Utilizando índices para otimizar consultas

Você pode utilizar índices para acelerar [consultas](#) e outras operações. Para obter informações sobre como criar expressões de índice otimizáveis por Rushmore, consulte o capítulo 15, [Otimizando aplicativos](#)

## Utilizando várias tabelas

Para utilizar mais de uma tabela, são utilizadas sessões de dados para controlar tabelas disponíveis para formulários e áreas de trabalho para definir as tabelas abertas. Uma [área de trabalho](#) é uma região numerada que identifica uma tabela aberta. Você pode abrir e manipular as tabelas do Visual FoxPro em 32.767 áreas de trabalho. As áreas de trabalho são normalmente identificadas nos aplicativos por meio do alias da tabela aberta na área de trabalho. Um alias de tabela é um nome que se refere a uma tabela aberta em uma área de trabalho.

## Utilizando sessões de dados

Além das áreas de trabalho visíveis na janela **Sessão de dados**, o Visual FoxPro cria automaticamente um ambiente separado para cada instância de um [formulário](#) ou [conjunto de formulários](#) através das [sessões de dados](#). Uma sessão de dados é uma representação do atual ambiente dinâmico de trabalho utilizado por um formulário, conjunto de formulários ou relatório. Cada sessão de dados contém seu próprio conjunto de áreas de trabalho. Essas áreas de trabalho contêm as tabelas abertas nas áreas de trabalho, seus índices e relacionamentos. Para obter informações sobre como utilizar sessões de dados, consulte o capítulo 17, [Programando para](#)

acesso compartilhado

## Visualizando áreas de trabalho

Você pode ver a lista de tabelas aberta em uma sessão do Visual FoxPro abrindo a janela **Sessão de dados**.

### ► Para abrir a janela Sessão de dados

- No menu **Janela**, escolha **Sessão de dados**.
  - Ou –
- Utilize o comando **SET**.

Quando você digitar SET na janela **Comando**, o Visual FoxPro abrirá a janela **Sessão de dados** e exibirá os aliases das áreas de trabalho para as tabelas abertas na sessão de dados atual.

### Janela Sessão de dados sem nenhuma tabela aberta



## Abrindo tabelas em áreas de trabalho

Você pode abrir uma tabela em uma área de trabalho utilizando a janela **Sessão de dados** ou o comando USE.

### ► Para abrir uma tabela em uma área de trabalho

- Na janela **Sessão de dados**, escolha **Abrir**.
  - Ou –
- Digite **USE** na janela **Comando**.

Para abrir uma tabela na primeira área de trabalho disponível, utilize a cláusula IN do comando USE com uma área de trabalho 0. Por exemplo, se houver tabelas abertas nas áreas de trabalho de 1 a 10, o comando a seguir abrirá a tabela customer na área de trabalho 11.

```
USE customer IN 0
```

Você pode também escolher **Abrir** no menu **Arquivo** para abrir uma tabela em uma área de trabalho.

## Fechando tabelas em áreas de trabalho

Você pode fechar uma tabela em uma área de trabalho utilizando a janela **Sessão de dados** ou a linguagem.

### ► Para fechar uma tabela em uma área de trabalho

- Na janela **Sessão de dados**, selecione o alias da tabela e escolha **Fechar**.  
– Ou –
- Digite **USE** sem um nome de tabela.  
– Ou –
- Utilize a cláusula IN do comando USE para fazer referência à área de trabalho da tabela que você deseja fechar.

Ao emitir o comando USE sem um nome de tabela, se houver um arquivo de tabela aberto na área de trabalho selecionada, essa tabela será fechada. Por exemplo, o código abaixo abre a tabela `customer`, exibe uma janela **Pesquisar** e depois fecha a tabela:

```
USE customer
BROWSE
USE
```

Você também fecha uma tabela automaticamente ao abrir outra tabela na mesma área de trabalho, ou ao emitir o comando USE com a cláusula IN e fazer referência à área de trabalho atual. O código abaixo abre, exibe e depois fecha a tabela `customer` emitindo o comando USE IN e o alias de tabela `customer`:

```
USE customer
BROWSE
USE IN customer
```

Não é possível haver mais de uma tabela aberta em uma área de trabalho ao mesmo tempo.

## Referenciando uma área de trabalho

Você pode fazer referência à próxima área de trabalho disponível antes de abrir uma tabela, utilizando o número da área de trabalho conforme mostrado abaixo:

```
SELECT 0
```

## Utilizando aliases de tabela

Um alias de tabela é o nome que o Visual FoxPro utiliza para fazer referência a uma tabela aberta em uma área de trabalho. O Visual FoxPro utiliza automaticamente o nome do arquivo como alias padrão quando você abre uma tabela. Por exemplo, caso você tenha aberto o arquivo `CUSTOMER.DBF` na área de trabalho 0 com os comandos abaixo, o alias padrão `customer` será automaticamente atribuído à tabela:

```
SELECT 0
USE customer
```

Você pode então utilizar o alias `customer` para identificar a tabela em um comando ou função. Você pode também criar seu próprio alias.

## Criando aliases definidos pelo usuário

Você pode atribuir um alias definido pelo usuário para uma tabela quando esta for aberta.

### ► Para abrir uma tabela com um alias definido pelo usuário

- Digite **USE** com um nome para o alias da tabela.

Por exemplo, para abrir o arquivo `CUSTOMER.DBF` na área de trabalho 0 e atribuir-lhe um alias `people`, utilize o comando a seguir:

```
SELECT 0
USE customer ALIAS people
```

Você deve então utilizar o alias `people` para fazer referência à tabela aberta. Um alias pode consistir em até 254 letras, dígitos ou caracteres de sublinhado e deve começar com uma letra ou caractere de sublinhado. O Visual FoxPro criará um alias automaticamente se o alias fornecido contiver um caractere não permitido em aliases.

## Utilizando aliases atribuídos pelo Visual FoxPro

O Visual FoxPro atribui automaticamente um alias para uma tabela em determinadas circunstâncias:

- Caso você abra uma mesma tabela em diversas áreas de trabalho ao mesmo tempo incluindo a cláusula AGAIN com o comando [USE](#) e caso você não especifique um alias ao abrir a tabela em cada área de trabalho.
- Caso ocorra um conflito com o alias.

Os aliases padrão atribuídos nas primeiras 10 áreas de trabalho são as letras das áreas de trabalho, de A a J; os aliases atribuídos nas áreas de trabalho de 11 a 32767 são de W11 a W32767. Você pode utilizar esses aliases atribuídos pelo Visual FoxPro da mesma forma como qualquer alias padrão ou alias definido pelo usuário seriam utilizados, para fazer referência a uma tabela aberta em uma área de trabalho.

## Selecionando uma áreas de trabalho utilizando um alias

Você pode mover-se de uma área de trabalho para outra utilizando o comando SELECT. Por exemplo, se CUSTOMER.DBF estiver aberto em uma área de trabalho e o alias padrão CUSTOMER tiver sido atribuído, você pode mover-se para essa área de trabalho com o comando SELECT a seguir:

```
SELECT customer
```

## Fazendo referência a tabelas abertas em outras áreas de trabalho

Você pode também fazer referência a campos em outras áreas de trabalho utilizando o nome de alias e um ponto, ou o operador →, como prefixo do nome de campo. Por exemplo, caso você esteja em uma área de trabalho e queira acessar o campo `contact` da tabela Customer, aberta em outra área de trabalho, utilize o seguinte para referir-se ao campo:

```
customer.contact
```

Se a tabela a que você deseja fazer referência tiver sido aberta com um alias, você pode utilizar o nome do alias. Por exemplo, se a tabela Customer tiver sido aberta com o alias `people`, você pode se referir ao campo `lastname` da forma a seguir:

```
people.lastname
```

Utilizar o nome da tabela ou o alias da tabela identifica especificamente a tabela desejada, não importando em qual área de trabalho a tabela esteja aberta.

## Definindo relacionamentos temporários entre tabelas

Ao estabelecer um [relacionamento temporário](#) entre tabelas, você faz com que o ponteiro de registro de uma tabela (a tabela filho) siga automaticamente os movimentos do ponteiro de registro na outra tabela, a tabela pai. Isso permite que você selecione um registro no lado 'um', ou lado pai, de um relacionamento, acessando automaticamente os registros relacionados no lado 'n', ou lado filho, do relacionamento de tabelas.

Por exemplo, você pode querer relacionar as tabelas `customer` e `orders` de forma que, ao mover o ponteiro de registro na tabela `customer` para um determinado cliente, o ponteiro de registro na tabela `orders` se mova para o registro que possua o mesmo número de cliente.

As áreas de trabalho e os aliases de tabela são utilizados para estabelecer os relacionamentos entre duas tabelas abertas com o comando SET RELATION. Caso você esteja utilizando um [formulário](#) para trabalhar com tabelas, você pode armazenar esses relacionamentos como parte do [ambiente de dados](#) do formulário.

## Relacionando tabelas de forma temporária

Utilize a janela **Sessão de dados** ou a linguagem para criar relacionamentos temporários entre

tabelas.

► **Para criar relacionamentos temporários entre tabelas**

- Na janela **Sessão de dados**, selecione as tabelas e utilize o botão **Relações** para criar relacionamentos.
  - Ou –
- Utilize o comando **SET RELATION**.

Utilize o comando SET RELATION para estabelecer um relacionamento entre uma tabela aberta na área de trabalho atualmente selecionada e outra tabela aberta em outra área de trabalho. Você normalmente irá relacionar tabelas que têm um campo em comum, e a expressão utilizada para estabelecer o relacionamento será geralmente a expressão de indexação do índice que controla a tabela filho.

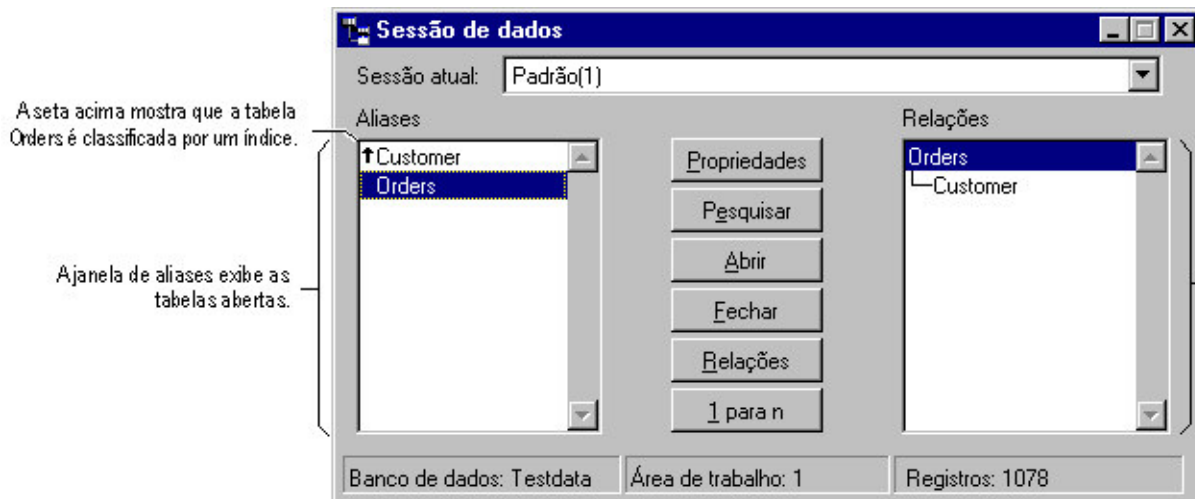
Por exemplo, um cliente pode ter muitos pedidos associados a seu registro. Caso você crie um relacionamento entre um campo comum a ambas as tabelas, você pode ver facilmente todos os pedidos de qualquer cliente. O programa a seguir utiliza um campo, `cust_id`, comum às duas tabelas, para criar um relacionamento entre o campo `cust_id` na tabela `customer` e a marca de índice `cust_id` na tabela `orders`.

**Utilizando SET RELATION para estabelecer um relacionamento entre duas tabelas**

Código	Comentários
USE customer IN 1	Abre a tabela <code>customer</code> (tabela pai) na área de trabalho 1.
USE orders IN 2	Abre a tabela <code>orders</code> (tabela filho) na área de trabalho 2.
SELECT orders	Seleciona a área de trabalho filho.
SET ORDER TO TAG cust_id	Especifica a ordenação da tabela para a tabela filho utilizando a marca de índice <code>cust_id</code> .
SELECT customer	Seleciona a área de trabalho pai.
SET RELATION TO cust_id INTO orders	Estabelece o relacionamento entre a tabela pai e o índice que controla a tabela filho.
SELECT orders BROWSE NOWAIT SELECT customer BROWSE NOWAIT	Abre duas janelas <b>Pesquisar</b> ; repare que mover o ponteiro do registro na tabela pai muda o conjunto de dados visualizado na tabela filho.

A janela **Sessão de dados** exibirá as duas tabelas abertas, `Orders` e `Customer`, bem como o relacionamento estabelecido pelo comando SET RELATION.

**A janela Sessão de dados exibe os aliases das tabelas abertas e os relacionamentos temporários.**





Você criou um índice para a tabela filho, `orders`, para organizar os registros dessa tabela em grupos, de acordo com o cliente que efetuou o pedido. Quando você criar um relacionamento entre a tabela pai e o índice da tabela filho, o Visual FoxPro selecionará somente os registros da tabela filho cuja chave de índice corresponda à chave de índice do registro pai selecionado.

O exemplo anterior estabeleceu um relacionamento único entre duas tabelas. Você pode também utilizar o comando **SET RELATION** para estabelecer relacionamentos múltiplos entre uma única tabela pai e diversas tabelas filho.

## Salvando relacionamentos de tabela em um ambiente de dados

Ao criar um [formulário](#) que utilize mais de uma tabela, você pode utilizar o [ambiente de dados](#) para criar relacionamentos entre tabelas e armazená-los com o formulário. Os relacionamentos estabelecidos no Ambiente de dados serão abertos automaticamente quando você executar o formulário. Para obter informações sobre como criar um ambiente de dados, consulte o capítulo 9, [Criando formulários](#)

## Relacionando registros de uma mesma tabela

Você pode também criar um relacionamento entre registros de uma mesma tabela. Esse relacionamento, conhecido como relação de auto-referência, pode ser útil nas situações em que você tenha todas as informações necessárias armazenadas em uma única tabela. Por exemplo, você talvez queira percorrer os gerentes na tabela `Employees` e fazer com que os funcionários subordinados a cada gerente sejam automaticamente alterados à medida que você mover o ponteiro de registro de gerente em gerente.

### ► Para relacionar de forma temporária registros em uma mesma tabela

- Na janela **Sessões de dados**, selecione as tabelas e utilize o botão **Relações** para criar relacionamentos.  
– Ou –
- Utilize o comando **SET RELATION**.

Para criar uma relação de auto-referência, a mesma tabela deve ser aberta duas vezes, a primeira em uma [área de trabalho](#) e a segunda, com o comando **USE AGAIN**, outra área de trabalho. Depois, você utiliza um índice para relacionar os registros. Por exemplo, o código abaixo estabelece e pesquisa um relacionamento de auto-referência, criando uma marca de índice denominada `mgr_id`, que ordena a tabela `Employee` pelo campo `reports_to`:

```
SELECT 0
USE employee ALIAS managers
SELECT 0
USE employee AGAIN ALIAS employees
INDEX ON reports_to TAG mgr_id
SET ORDER TO mgr_id
SELECT managers
SET RELATION TO emp_id INTO employees
BROWSE
SELECT employees
BROWSE
```

Quando você mover o ponteiro de registro na janela **Pesquisar** de `managers`, a janela **Pesquisar** de `employees` será atualizada para mostrar somente os empregados subordinados ao gerente selecionado.

## Definindo relacionamentos permanentes com índices

Os índices são utilizados para estabelecer [relacionamentos permanentes](#) entre tabelas de um banco de dados. Relacionamentos permanentes são relacionamentos entre tabelas de bancos de dados que são armazenados no arquivo de banco de dados e são automaticamente utilizados como condições padrão de associação nos **Criadores de consulta** e de **Visualizações**. Relacionamentos permanentes também são exibidos no **Criadores de bancos de Dados** como linhas associando índices de tabelas, e como relacionamentos padrão quando as tabelas são utilizadas no Ambiente de dados.

Ao contrário dos [relacionamentos temporários](#) definidos com o comando **SET RELATION**, [relacionamentos permanentes](#) não precisam ser recriados a cada vez que você utilizar as tabelas. Contudo, uma vez que os relacionamentos permanentes não controlam os relacionamentos entre ponteiros de registro nas tabelas, será necessário utilizar tanto relacionamentos temporários do tipo **SET RELATION** quanto relacionamentos permanentes ao desenvolver aplicativos do Visual FoxPro. Para obter maiores informações sobre como definir relacionamentos permanentes, consulte o capítulo 6, [Criando bancos de dados](#)