September, 1996

Advisor Answers

Q: Do you know of a way to link two tables so that combo boxes (drop down lists) will automatically reflect a one to many relationship. What do I need to set on the combo boxes?

–Paul Frankel (via the Internet)

A: The good news is that you can do this without too much work. The bad news is that you can't do it by simply setting up an appropriate relationship. There are actually several ways to handle this kind of setup. All of them have one thing in common - they use the second combo's Requery method to repopulate that combo as needed.

Here's one solution—the form I built using it is shown in Figure 1 and included on the Companion Disk. In my example, the relationship is based on the Supplier and Product tables from VFP's sample TasTrade database. The left-hand combo shows suppliers. When you choose a supplier, the right-hand combo shows products available from that supplier.

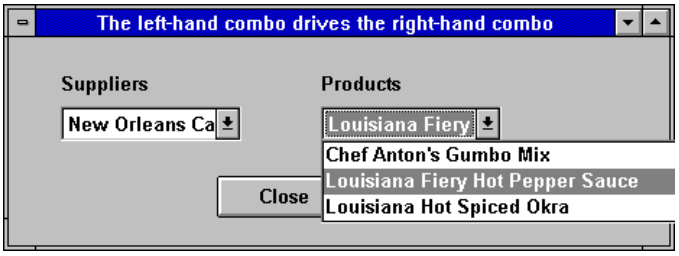

Figure 1 Paired combos. When the supplier combo is used, it calls the products' combo's Requery method to update the list.

The first step is to create a parameterized view of the many side of the relationship. The parameter is the id of the item chosen on the one side. I put the view in a separate work database, but it could go in the same database as your original data:

```
CREATE DATABASE DualComb
CREATE SQL VIEW SupplierProducts AS ;
   SELECT * FROM TasTrade!Products ;
      WHERE Supplier_Id = ?THISFORM.cboSuppliers.Value
```

Note that this view is specifically intended for use in the form, so the parameter is the value of the first combo. However, if you simply USE SupplierProducts, VFP happily prompts you for THISFORM.cboSuppliers.Value as shown in Figure 2. (See Will Phelps article "Change Your Point of View for Better Design" in the July issue for more on the tricks you can do with the View Parameters dialog.)

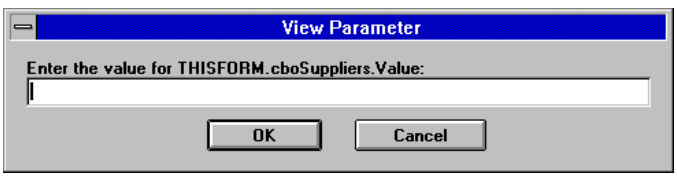

Figure 2 View Parameters dialog. Even outside a form, VFP is happy to prompt you for a parameter like THISFORM.cboSuppliers.Value.

Now we can create the form. In the example, the left-hand combo box is named cboSuppliers while the right-hand combo is cboProducts.

In the data environment, add the table for the left-hand combo (Supplier in the example) and the view for the right-hand combo. Set the view's NoDataOnLoad property to .T. to prevent the form from looking for the parameter too soon.

Now set the combo properties. For cboSuppliers, set:

```
Name = cboSuppliers
Style = 2 (Dropdown List)
RowSourceType = 6 (Fields)
RowSource = "Supplier.Company_Name,Supplier_Id"
BoundColumn = 2
```

The combo uses the Supplier table, showing the company name, but storing the company's id in Value.

For cboProducts, set:

```
Name = cboProducts
Style = 2 (Dropdown List)
RowSourceType = 6 (Fields)
RowSource = "SupplierProducts.English_Name"
```

Now we need just a little method code. We want to repopulate the Product combo every time the Supplier changes. We'll do that by issuing REQUERY() for the view (which tells it to take the current value of its parameter and rebuild the cursor) and then using the combo's Requery method to refill it. The most transparent way to do this is to put the call to REQUERY() for the view in the combo's Requery method. Then, a call to the combo's Requery does the whole job and the caller doesn't need to know how the combo is built. Put this code in the Requery method of cboProducts:

```
=REQUERY("SupplierProducts")
THIS.ListIndex = 1
```

The second line resets the highlight in the combo to the first product for this supplier. The code takes advantage of the fact that the default behavior of a method happens *after* any code in that method, so we can rebuild the view, then the combo gets refilled.

To update the products combo every time the user chooses a different supplier, call the products combo's Requery method from the Valid method of the suppliers combo:

```
THISFORM.cboProducts.Requery()
```

The only task left is to fill the products combo in the first place. We opened the view with no data because the supplier combo didn't have a Value at that point. In the Init method for the Supplier combo, we can give it a Value and requery the products combo to get it set up:

```
THIS.Value = Supplier.Supplier_ID
THISFORM.cboProducts.Requery()
```

One more line of code puts the highlight on the first item in the Products combo when we start the form. In cboProducts' Init method, put:

```
THIS.ListIndex = 1
```

Finally, I put a standard Close button on the form. It's Click method contains THISFORM.Release().

With a little more work, it's possible to build this whole thing as a Container class you could just drop into a form as needed.

–Tamar