November, 1996

Advisor Answers

Visual FoxPro

Q: I have a form (MyForm) that has a grid (MyGrid) on it. The RecordSourceType for the grid is "Alias" and the alias is a view (MyView).

The view is created using:

```
Create SQL view MyView as ;
   select * from MyTable where <conditions>
```

If I create the view first, then USE it, then run the form, all is well.

Now my question: I want to be able to recreate the view on the fly with user supplied conditions from the same form the grid is on (MyForm) and then have the grid update to reflect the new view.

I've tried a number of things that don't work. Browsing the view makes it obvious that the view is not updated at all until it is closed and re-opened. When I close the initial view, the data disappears from my grid. When I then USE the modified view (with the new criteria) I can not, for the life of me, get the grid to refresh, although browsing again reveals that the new data is indeed there.

Any ideas are most appreciated!

–Ken B. Matson (via the Internet)

A: Grids are picky about their RecordSource. When you close the table or cursor a grid is based on, the data clears out. Re-opening a table or cursor with the same alias does *not* refill the grid. What's going on here is that the grid is tied in some way to the actual open table or cursor, not just to the name. (See this month's Tips, Tricks and Traps, however, for one workaround to this problem.)

There is a way to re-hook the grid to the alias - reset RecordSource like this:

```
THIS.RecordSource = THIS.RecordSource
```

However, in most cases, this isn't good enough. When you change a grid's RecordSource, the grid loses all customization that you've done. Any custom controls you've added or colors you've set or method code or anything else is gone. Most of the time, you'll want to keep those things, so another approach is needed.

This sounds like a perfect place to use a parameterized view. Let's review the basics of views.

A view is a query definition stored in a database. When you USE a view, VFP executes the query and creates a cursor containing the data specified. Ordinarily, a view definition is static. Given the same set of source data, it returns the same subset each time you open it.

However, views have a feature that makes them extremely powerful - the ability to accept parameters. In the WHERE and HAVING clauses of a view, you can precede a variable with a question mark (like ?MyVar) to indicate that that variable is a parameter to the query.

If the variable exists when you open the view, there's no difference between using a parameter and using any other variable. The value is substituted into the query before execution.

But, if the variable doesn't exist when you open the view, VFP prompts you for a value. While you wouldn't usually want end-users to see the View Parameters dialog, it's extremely handy while you're testing.

Here's an example of a parameterized view. We'll create a special database to hold the definition, but it could go in an existing database, either the one which contains its source tables or any other. This view chooses all the employees in a specified country.

```
CREATE DATABASE TestView
CREATE SQL VIEW EmpsByCountry AS ;
   SELECT * FROM TasTrade!Employee ;
      WHERE Country = ?m.cCountry
```

To open the view:

```
USE EmpsByCountry
```

Since the parameter cCountry doesn't exist, the View Parameters dialog prompts us for a value. Enter "USA" (without the quotes), then Browse the result. You'll see all the employees who live in the United States. Now close the view and open it again, but this time specify "UK" in the dialog. Now you get all the employees in the United Kingdom.

But we can do better. Close the view and open it again. This time, leave the dialog blank - just choose OK. Browse the result and you see all employees. FoxPro's partial string matching here means that every record matched the empty string. (If you keep ANSI set ON, you'll need to `SET ANSI OFF` for this example to work.)

Finally, let's create the parameter and then open the view:

```
cCountry = "France"
USE EmpsByCountry
BROWSE
```

You see all the employees in France. To get all employees, just set cCountry to "" before opening the view.

But we're still opening and closing the view to change which records it shows. Fortunately, we don't have to do it that way. The REQUERY() function tells a view to go back to the original source and refill itself based on its current parameters and the data now available at the source. As before, if the parameter exists when you issue REQUERY(), its current value is used. If not, you're prompted for a value. Try this:

```
cCountry = "USA"
USE EmpsByCountry
```

```
BROWSE
* Leaving the Browse visible
cCountry = "UK"
?REQUERY("EmpsByCountry")
cCountry = ""
?REQUERY("EmpsByCountry")
RELEASE cCountry
?REQUERY("EmpsByCountry") && This time, there's a prompt
```

You can base your grid on a parameterized view and simply REQUERY() the view each time the user changes the conditions. Put all the possible conditions into the view's WHERE clause. Let the user specify a value for each parameter.

The one case where this approach may be a problem is when some conditions are not based on character values. Comparisons of other types don't automatically match the empty value, so if you need to match an exact value, you'll have to be a little more creative in the query. For example, if you want the user to be able to specify employees born on a certain date, but that condition is optional, you'll need the WHERE clause to read something like:

```
WHERE EMPTY(?dBirthDate) OR Birth_Date = ?dBirthDate
```

Usually, however, conditions involving dates or numbers won't need an exact match, but will involve inequalities. Again, a little creativity in the WHERE clause will let you express such conditions so that they're evaluated only when you want them to be.

–Tamar