July, 1997

Advisor Answers

Visual FoxPro

Q: I have a number of views which I use for reporting. They differ only in the sort order. There doesn't seem to be a way to use a parameter in the ORDER BY clause. Is there a solution other than maintaining multiple, similar, views?

–Eldor Gemst (via CompuServe)

A: Maintaining multiple views that need to remain in synch isn't a great idea. If a field name changes or you need to add a field, you have to remember to update all of the multiple views.

As you've noted, view parameters don't extend to the ORDER BY clause. They're restricted to the WHERE and HAVING clauses. You can use a macro in the ORDER BY clause, but it's tricky to get it stored with the view and, of course, works only for local views, not remote.

An alternative solution is to use a single view and index it as needed once it's been opened. This technique is most useful when the number of records in the view is reasonably small since the user will have to wait for the view to be indexed.

The tricky part is figuring out where to put the INDEX command. When you use the Data Environment and set AutoOpenTables to .T., views are automatically opened on the way into a form or report. The BeforeOpenTables event fires just before the tables and views open, but there's no corresponding AfterOpenTables event. The actual opening of tables and views occurs as the very last step in OpenTables, after all custom code there has been executed. The sequence of code on the way into a form or report with AutoOpenTables = .T. is:

1. Any custom code in the OpenTables method runs
2. Any code in the BeforeOpenTables method runs
3. Any tables and views in the DE are opened

In a report, the Inits for the cursors in the Data Environment fire next. In a form, the next thing to happen is the form's Load. So where can you put the INDEX command you need?

The secret is to change the order in which things happen. The keyword NODEFAULT and the new DODEFAULT() function or the scope resolution operator (::) let us move things around.

Every method in VFP has some built-in behavior that fires when it's called. For OpenTables, the built-in behavior is opening the tables and views listed in the Data Environment. This built-in behavior is the last thing to happen when the method runs.

Putting the keyword NODEFAULT anywhere in a method keeps that method from firing its built-in behavior.

In VFP5, DODEFAULT() calls up the class hierarchy to the current method one level up. In VFP3, we can't do this generically, but :: lets us specify a class and method to call.

We can use this combination to make the tables and views open earlier, giving us a place to put code that should be executed after the tables and views are open. Here's the structure for the OpenTables method:

```
* any code that should execute before
* tables and views open

* now force them to open right away
DODEFAULT()  && or in VFP3, DataEnvironment::OpenTables

* prevent the tables and views from being
* opened again at * the end of OpenTables
NODEFAULT

* now you can do anything you want with
* the tables and views
* For example:
SELECT MyView
INDEX ON SomeField TAG SomeTag
```

Even though VFP's reports are not objects, the data environment they use is an object and this technique works equally well in reports and views. The technique of forcing the default behavior to occur earlier can be useful in other methods as well.

–Tamar