August, 1998

Advisor Answers

Q: I can modify the Send Updates setting of a view through code, but I can't see how to modify the view's update criteria. How do you modify a view's update criteria through code rather than with the View Designer?

–Chico (via Advisor.Com)

A: In general, view properties are modified using the DBSetProp() function. However, the mapping between what you see in the View Designer and what you write in code is not totally intuitive.

DBSetProp() is used to set properties of all kinds of objects in a database. When you use it, you have to specify not just the name of the object to be updated, but its type (table, view, connection, field) as well. The syntax for DBSetProp() is:

```
lSuccess = DBSetProp( cObject, cType, cProperty, uNewValue)
```

When you handle view properties in code, you need to address properties of both the view and its fields. The properties you're most likely to set at the view level are:

SendUpdates–determines whether updates are sent to the original table. This corresponds to the "Send SQL Updates" checkbox on the Update Criteria page of the View Designer (shown in Figure 1).

Tables–a list of the tables that can be updated. This more or less corresponds to the "Tables" dropdown on the Update Criteria page.

UpdateType–determines whether updates use the SQL UPDATE command or SQL DELETE followed by SQL INSERT. This setting corresponds to the "Update using" option buttons on the Update Criteria page.

WhereType–determines how records in the view are matched with records in the original tables to perform updates. This one corresponds to the "SQL WHERE clause includes" option buttons.

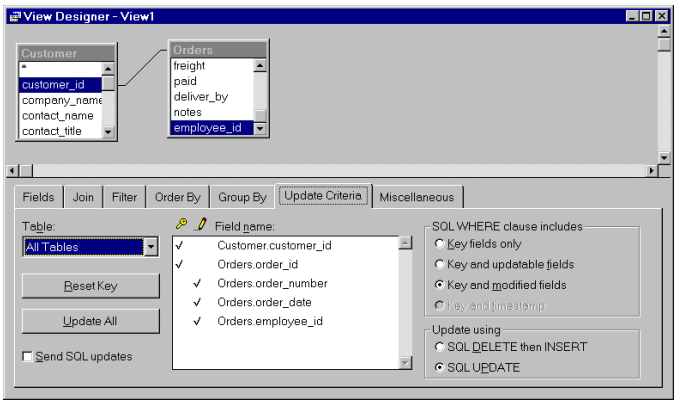

Figure 1. Update Criteria - This tab of the View Designer lets you indicate how changes to the view are sent back to the original tables.

For example, to indicate that a view called MyView should be updated based on the key field and modified fields, you'd issue this call:

```
DBSetProp("MyView","View","WhereType",3)
```

Because you can choose whether to update individual fields, that information is specified at the field level. The most important property for updating is **Updatable**, which indicates whether the field should be updated. This corresponds to the "pencil" column on the Update Criteria page.

DBSetProp() has one additional wrinkle when used on fields. For the object, you have to specify both the view (or table) name and the field name, using the usual FoxPro alias.field notation. So, to set the field cName in MyView as updatable, use:

```
DBSetProp("MyView.cName","Field","Updatable",.T.)
```

There are a couple of other field properties that are important for updatable views. First, the **KeyField** property indicates whether this field is part of the primary key for the table. This information is used to match records in the view with records from the original table. KeyField corresponds to the "key" column on the Update Criteria page.

Finally, **UpdateName** lets you deal with cases when a field in the view has a different name than in the original table. Specify the original name as UpdateName to indicate that this field should be matched to the field you name on the back-end. For example:

```
DBSetProp("MyView.MyField","Field","UpdateName", ;
          "OrigTable.OrigField")
```

says that the field MyField in MyView should be matched with OrigField in OrigTable when updates are performed.

DBSetProp() sets these properties in the database itself, so that they apply each time the view is opened. These properties can also be set up for a particular use of the view with the CursorSetProp() function. Properties set that way apply only as long as the view is open.

In CursorSetProp(), there's no distinction between view properties and field properties. Everything applies to the view as a whole. Because of this, the field-level properties take

a different form. Rather than a field-level KeyField property, for example, CursorSetProp() offers a view-level KeyFieldList that accepts a comma-delimited list of key fields.

Similarly, UpdatableFieldList takes a comma-delimited list of field names that can be updated. UpdatableNameList gets a comma-delimited list of original field names and their corresponding field names in the view. For example:

```
CursorSetProp("UpdateNameList", ;
   "MyFld OrigTable.OrigFld, MyOtherFld OrigTable.OtherFld")
```

The combination of DBSetProp() and CursorSetProp() gives you the best of both worlds. You can set up each view in the database the way you're most likely to use it, but make changes for individual situations as needed.

–Tamar