March, 2007

Advisor Answers

Save Table Structure as Text

VFP 9/8/7

Q: Is there any way to save the structure of a table as a text file and then use the text file to create a new table with the same structure?

A: VFP offers several ways to save enough information to recreate a table, but none of them offer exactly what you're asking for. Let's take a look at them and see what might solve your problem.

First, and perhaps easiest, is the GENDBC utility that comes with VFP. When you run GENDBC (found in the Tools\GENDBC directory) against a database, it creates a program capable of recreating the database's structure, including tables, views, indexes and stored procedures. However, GENDBC doesn't work with free tables, and you can't tell it to generate code for just a single table.

Another approach to creating a new table with the same structure as an existing table is using the AFIELDS() function to populate an array with the structure information and then calling CREATE TABLE with the FROM ARRAY clause to generate a new table. The code looks like this (assuming a table is open in the current work area):

```
AFIELDS(aStructure)
CREATE TABLE NewTable FROM ARRAY aStructure
```

However, there's no direct way to store the content of the array created by AFIELDS() to a text file. If the result doesn't have to be plain text, you could write to a .MEM file with SAVE TO and read it back using RESTORE FROM. Otherwise, you'd have to write some code to save it and read it back.

Yet another way to create a table like an existing table is the COPY STRUCTURE command. As the name suggests, this command creates a new table with the same structure as the table open in the current work area, like this:

```
COPY STRUCTURE TO NewTable
```

But this approach, again, doesn't store the information in a text file. It simply creates the table. However, a variant of COPY STRUCTURE does create a file containing structure information.

COPY STRUCTURE EXTENDED creates a new table that contains the structure information of the open table. The number of fields in the new table depends on the version of VFP you're working with, and is the same as the number of columns in the array created by AFIELDS(). In VFP 8 and 9, the table has 18 fields. There's one row in the new table for each field of the original.

The table created by COPY STRUCTURE EXTENDED can be used by the CREATE FORM command to create a new table. Here's an example, assuming the table you want to copy is open in the current work area:

```
COPY STRUCTURE EXTENDED TO TableStructure
CREATE NewTable FROM TableStructure
```

COPY STRUCTURE EXTENDED has the advantage that it creates a table with a disk presence, so you can use it in a future session, e-mail it to someone else, or take it to another site. However, it still doesn't create a text file.

The LIST STRUCTURE command saves structure information to a text file, but it doesn't save it in a format that can be used to create a new table. You'd have to parse the text to remove header information and to format the rest into an appropriate format for creating a table.

One option for generating a text file that can be used to create a new table is to save the structure information in XML. I've written a pair of functions that use COPY STRUCTURE EXTENDED and CREATE FROM together with CursorToXML() and XMLToCursor() to store table information into a text file and then create a new table from that text file. StructureToXML.PRG saves the structure information as XML; it has one required parameter, the name for the XML file, and one optional parameter, the workarea or alias containing the table. If the second parameter is omitted, StructureToXML works in the current work area.

```
* StructureToXML.PRG
* Save the structure of a table in XML format

LPARAMETERS cXMLFile, uWorkArea
* uWorkArea - (numeric) work area or (character) alias.
*             If omitted, use current work area.

LOCAL nCurrentArea, cStruFile, cOldSafety
```

```
IF VARTYPE(m.cXMLFile) <> "C" OR EMPTY(m.cXMLFile)
   ERROR 11
   RETURN .f.
ENDIF

nCurrentArea = SELECT()
IF NOT EMPTY(m.uWorkArea)
   SELECT (m.uWorkArea)
ENDIF

IF NOT USED()
   ERROR 52
   SELECT (m.nCurrentArea)
   RETURN .F.
ENDIF

cOldSafety = SET("Safety")

cStruFile = FORCEPATH("Stru.DBF", SYS(2023))
SET SAFETY OFF
COPY STRUCTURE EXTENDED TO (m.cStruFile)
SET SAFETY &cOldSafety

SELECT 0
USE (m.cStruFile) ALIAS __Stru

CURSORTOXML("__Stru", m.cXMLFile, 1, 8 + 512, 0, "1")

* Clean up
USE IN __Stru
ERASE (m.cStruFile)
ERASE FORCEEXT(m.cStruFile, "FPT")

SELECT (m.nCurrentArea)

RETURN
```

TableFromXML.PRG reads the XML file to create a new table. It expects two parameters, the name of the XML file and the name of the new table.

```
* TableFromXML.PRG
* Create a new table from an XML structure file

LPARAMETERS cXMLFile, cTableFile

IF VARTYPE(m.cXMLFile) <> "C" OR EMPTY(m.cXMLFile)
   ERROR 11
   RETURN .f.
ENDIF

IF VARTYPE(m.cTableFile) <> "C" OR EMPTY(m.cTableFile)
   ERROR 11
```

```
   RETURN .f.
ENDIF

LOCAL nCurrentArea, cStruTable, cOldSafety

nCurrentArea = SELECT()
cOldSafety = SET("Safety")

XMLTOCURSOR(m.cXMLFile, "__Stru", 512)

* Have to create an actual table
cStruTable = FORCEPATH("Stru.DBF",SYS(2023))

SELECT __Stru
SET SAFETY OFF
COPY TO (m.cStruTable)
SET SAFETY &cOldSafety

CREATE (m.cTableFile) from (m.cStruTable)

* Clean up
ERASE (m.cStruTable)
ERASE FORCEEXT(m.cStruTable, "FPT")

SELECT (m.nCurrentArea)

RETURN
```

Both functions SET SAFETY OFF before creating the structure DBF, but otherwise leave it under the control of the calling program, so you can decide whether users get a chance to avoid overwriting the XML file and the new table.

What I like most about this solution is that it lets you combine some very old Xbase commands with much newer elements of VFP to get useful results. The two functions are included on this month's PRD.

–Tamar