February, 1999

Advisor Answers

Exploring database contents

Visual FoxPro 6.0, 5.0, 3.0

Q: I am looking for a way to query the .DBC to determine what tables and views are housed there. I can close the database container and open it as a free table, but that causes problems for the app in a multi-user environment. Ideally, I need a way to query the container, and get back a cursor containing the tables and views which can then be used to populate a list or tree view.

-- Jason Tryon (via Advisor.Com)

A: While it doesn't create a cursor, the ADBObjects() function is the ticket to your solution. It examines the open database and creates an array of contained objects. You tell it what kind of objects you're interested in. For example, to get a list of tables in the current database, issue:

```
nTableCount = ADBObjects(aTableList, "TABLE")
```

Similarly, for a list of views, use:

```
nViewCount = ADBObjects(aViewList, "VIEW")
```

It doesn't take much code to go from these arrays to the cursor you want. Something along these lines should do the trick:

```
* Make sure the right database is selected
SET DATABASE TO MyDatabase

* Create a cursor for results
CREATE CURSOR DBCContents (cItemName C(254), cType C(1))

* Get table information
nTableCount = ADBObjects(aTableList, "TABLE")
* Copy array to cursor when there's a table in the DBC
IF nTableCount > 0
   * Redimension array to 2-D
   DIMENSION aTableList[ nTableCount, 1]
   * Now copy from array to cursor
   APPEND FROM ARRAY aTableList
ENDIF
* Mark as tables
REPLACE ALL cType WITH "T"

* Get view information
nViewCount = ADBObjects(aViewList, "VIEW")
* Copy array to cursor when there's a view in the DBC
IF nViewCount > 0
   * Redimension array to 2-D
   DIMENSION aViewList[ nViewCount, 1]
   * Now copy from array to cursor
   APPEND FROM ARRAY aViewList
```

```
ENDIF
* Mark as views
REPLACE cType WITH "V" FOR EMPTY(cType)
```

ADBObjects() can also tell you about connections and persistent relations. For tables, views and connections, the resulting array has one column. For persistent relations, the function creates a five-column array containing information about the parent table, the child table, the tags used to create the relation, and any referential integrity constraints based on that relation.

A number of functions let you retrieve other information about your databases and their contents. ADatabases() fills an array with a list of all open databases. AFields() loads field information into an array. DBGetProp() lets you delve into the database in great detail to find out the properties of the tables, views, fields and connections. For example, to find out the primary key for the table MyTable, use:

```
cPrimary = DBGetProp("MyTable","Table","PrimaryKey")
```

Some of the properties returned by DBGetProp() can also be changed using DBSetProp().

There's also a full collection of functions to tell you about indexes. Take a look at TAG(), KEY() and TAGCOUNT() to see what indexes you have, and PRIMARY(), CANDIDATE(), FOR() and DESCENDING() to learn more about each index.

FoxPro is generally very good about letting your explore your data, the environment, your class libraries and objects and just about anything you can create. When you're trying to figure out how to dig into a particular item, there are a number of places to look:

- the functions beginning with "A" – many of them (like ADBObjects()) look up some information and put it into an array;
- the SET() functions – if you can set it with a SET SOMETHING command, you can almost always find out the current setting with a SET("SOMETHING") call;
- the SYS() functions – while the names of these are obscure and many of them are duplicated by better-named functions, there are still some settings that are only available through this mechanism.

In addition to looking in Help for what you need, the sample code and utilities that come with VFP are another rich source of ideas. For example, for your problem, a good place to look is GenDBC, a utility that generates a program to recreate your database. Clearly, that program needs some way to determine what's in the database and, in fact, it uses several calls to ADBObjects(). The Solutions sample included with VFP 5 and 6 provides ideas for a number of common problems. Look at those examples and the code behind them to learn how to use various of VFP's features.

-- Tamar