Cool Tool of the Month: Xcatalog

by Whil Hentzen

Sooner or later, every developer gets tired of writing routines that rely on hard-coded data in their systems - the names of the tables, indexes, fields, and so on. They've usually heard of other developers using a "data dictionary" as a central repository for this system data, and finally reserve some time to set one up for themself. And instead of searching the marketplace for an existing solution, they'll most likely try to create one from scratch.

Their explanation for wanting to reinvent the wheel usually includes one or more of the following reasons: (1) late nights are more plentiful than cash, (2) the intellectual challenge and thrill of invention, (3) existing products are too constricting, too powerful, or not powerful enough, or (4) proprietary schemes are too limiting.

The Problem

However, midway into the development of their own data dictionary, a developer will usually face two problems. First, they don't know what to put in it, or how to structure it. After the initial glow of automating the file opening and reindexing processes has worn off, the hard work of analysis and design has begun. "I wonder what some other people have in theirs..." Second, more and more third parties are incorporating some sort of meta-data repository into their products, and it's possible to find oneself in the frustrating position of entering nearly identical information into two or three data dictionaries.

The developer faces the possibility of ending up with a set of continually envolving data dictionary programs and data files. The savings that were supposed to be realized by using a set of standard "black boxes" have dissappeared in the face of needing to maintain a series of programs that differ from the previous version by just enough to make life miserable. A second possible result is a half-baked collection of partially implemented routines that haphazardly hook into a poorly designed meta-data structure. This is no longer a maintenance nuisance - it's a maintenance nightmare. And a third possible result is simple paralysis when faced with multiple data dictionary schemes. Yes, the entire project gets shelved and it's back to hard coding.

The Solution

Enter Xcatalog, a public domain "system catalog" or "data dictionary" which is oriented around the Xbase programming language. It's architect, Tom Rettig, introduced Xcatalog last October and together with contributions from Alan Schwartz and Alan Griver last updated it on July 16, 1993. It is not intended to set a "standard" nor to crowd out other data dictionary products. Rather, it is an attempt to set up some conventions that we developers can all agree on until the systems software manufacturers and Xbase language vendors provide a set of standards. As Alan Schwartz said, "Until the vendors provide us with the mechanisms to play with the same toys, let's work on playing in the same sandbox and learning to share our toys."

According to Rettig, "Xcatalog's goals are (1) to provide a common base from which independent applications can begin to work together today, (2) begin to educate Xbasers about relational theory, and (3) to help smooth the Xbase transition into the real relational world."

You'll find that it's robust (it has all the features that you'll want), flexible (if it's missing something that you **gotta** have, you can add it), adaptable (there's a utility to map other data dictionaries with Xcatalog), and popular (several prominent parties whose products and methodologies are in wide use have had a hand in it's development.)

Before we look under the hood or turn the key, it's probably a good idea to explicitly state that Xcatalog is a "passive" data dictionary. In other words, it is an aid in organizing the data about your application - it, by itself, does not enforce rules or relationships used by your system. Again, to quote Rettig, "To make it work, you must add code to your application. Xcatalog's structures and architecture are intended to give you a head start on data driving your applications."

How To Use It

If you're used to a data dictionary that consists of a single table that contains database, field and tag names, you're in for a shock when you start examining Xcatalog. It's like going from a 1981 IBM PC to a dual-Pentium, EISA, VESA VL-Bus, SCSI-2 tower. After you unzip Xcatalog, you'll find a short READ.ME text file, two databases (SYSCAT and SYSMAP), three reports (SYSDET, SYSSUM, SYSGRP), and a utility program, XCATALOG, in your directory.

However, the complete XCatalog system consists of about 15 databases. There are separate tables for Tables, Relations, Fields, Indexes, Keys, Filters, Users, Useage, Input/Output Rules, Screens, Domains, Mapping, and Views. Where are they? Patience - the XCATALOG program will generate these files in a minute.

Your first step (after reading the READ.ME text file) will be to open up each of the three reports and run them to produce the

Xcatalog documentation. Alternatively, you can simply USE the SYSCAT database and scan the contents of each record's memo field. Either of these steps will familiarize you with the file structure of XCAT and the purposes of each database.

Now that you're comfortable with the contents of each of the tables, you'll run the XCATALOG program. If you're skipping ahead, you've already typed in

do XCATALOG

and ended up with an error message. The correct syntax is

do XCATALOG with "create"

(yes, include the quotation marks). If you're just doing this for educational purposes, you can run XCATALOG from anywhere. If you intend on building a data dictionary for a set of existing data files, you'll want to put XCATALOG.PRG and SYSCAT.* into the directory that contains those data files. After the program is done, you'll end up with the databases mentioned above. Now that you've created the "bare bones" data dictionary, you can run XCATALOG again, with the command

do XCATALOG with "table"

to add application-specific records to several of the tables - how many and to which tables depends on your application. At a minimum, however, you should see records specific to your application in the Tables, Fields and Indexes tables.

You're now ready to "add water and stir." Build the routines you want - a database opener, a reindexer, a memo packer, a field picker, and so on - make them into "black boxes" that can be used by multiple systems, and call them from your application.

The purpose of this article was not to do a long disseration on complex tricks with Xcatalog, but to introduce you to the tool, and to make you aware that you needn't reinvent the wheel if you're considering a data dictionary. Xcatalog provides the best of both worlds - a solid foundation with virtually everything you might need already thought of, but with sufficient flexibility that you can modify and customize as you desire. I'd be interested in hearing about the routines you create and innovative uses you come up with for Xcatalog.

Platforms

Currently, Xcatalog is implemented entirely in FoxPro 2.5 and will run under both the DOS and Windows versions.

Where To Find It

XCAT.ZIP is a 50K file in CompuServe's FoxForum Library 10. It's also on this month's companion diskette. It's in the public domain so there are no registration fees.

Whil Hentzen is president of Hentzenwerke, a 10 year old computer consulting firm that specializes in FoxPro and TRO-based strategic database applications and has commercial products and custom applications running throughout the U.S. and in 12 foreign countries. Author of The Ultimate FoxPro Reference, Whil heads up the Milwaukee Association of FoxPro Developers and has spoken on Rapid Application Development at developer conferences and user groups across North America. He can be reached at 414.332.9876 (voice), 414.963.4999 (fax), 70651,2270 (CompuServe).