



The History of FoxPro

[Home TOC](#)



The Product That Won't Die

Tom Spitzer

Published in DBMS Magazine February 1997

The other night I went out to dinner with a colleague who specializes in developing sophisticated business applications and who prefers to use Microsoft Visual FoxPro (VFP) as her development tool. During the course of dinner, she asked me why Microsoft does not appear to be aggressively promoting that product, particularly because they recently released version 5.0 that is amazingly rich in features and performance. I was not at all surprised by the question; discussions such as this one are fairly commonplace among VFP application developers. At the FoxPro Developer conference in November 1996, a large room full of developers voiced the same concerns at a feedback session with Microsoft's product marketing team.

I explained to my colleague that I thought Microsoft had purchased Fox Software Inc. and FoxPro back in 1992 for three reasons. First, the FoxPro database engine code contained sophisticated optimization techniques that could be applied to Microsoft's other database products. The talented folks who developed FoxPro made a valuable addition to the Microsoft development effort at a time when the company was just starting to get serious about databases. Second, FoxPro had a large and loyal following among PC application developers. Microsoft figured that by acquiring FoxPro, and over time merging it into their other development tools, it would obtain the allegiance of the PC application development community.

Finally, at the time of the FoxPro acquisition, Microsoft had not yet released Access, nor had the Visual Basic boom begun. Microsoft had not succeeded in previous attempts to develop a viable local database engine and data-oriented application development technologies. FoxPro provided a fallback position in case Access did not succeed in garnering user or developer support or if the Xbase development model remained dominant. I do not believe that in 1992 even Microsoft imagined a future where there would be mass adoption of Visual Basic develop database applications.

Everything that happened between 1992 and the end of 1995 promoted Microsoft's core database and tools strategy and increasingly marginalized Visual FoxPro, which was released late and was not promoted outside of the FoxPro community. Access has been wildly successful among both end users and application developers. The growing number of developers using Visual Basic for client/server database applications is astounding. Borland fed its LAN database products cyanide, with the dual results that the Xbase industry disintegrated and Paradox (recently sold to Corel

Corp.) faded as a significant competitor to Access. Meanwhile, Internet architectures emerged as the paradigm for the future of network computing, and Microsoft had to focus significant resources (significant even for Microsoft) on creating development and deployment platforms for this paradigm. Against this backdrop, Visual FoxPro was at best a distraction.

Death and Rebirth

After the initial release of Visual FoxPro, application developers complained about the product's performance and stability, as well as about Microsoft's unwillingness to recommend it as a serious solution at general developer conferences and in corporate marketing presentations. A simmering situation reached the boiling point in early 1996 with the publication of several articles that indicated that Microsoft would withdraw the product entirely. I believe that Microsoft would have loved to eliminate the product right then and there and leaked the news of its impending demise as a trial balloon. I further believe that the ensuing firestorm in the FoxPro development community included credible threats of legal action if Microsoft had chosen to orphan some significant users. The way I read the situation, this threat forced Microsoft instead to continue with the VFP 5.0 development. Why would Microsoft want to kill Visual FoxPro? From a resource standpoint, there is a group of extremely talented people on the FoxPro team who could be deployed far more strategically. Certainly there is too much product overlap. A white paper titled "Choosing the Appropriate Database Development Tool" -- published by Microsoft in October 1996 -- presents a Database Engine Decision Worksheet indicating that the FoxPro engine is only appropriate in environments where around-the-clock availability is not required, where total data storage is less than one gigabyte and there are fewer than 50 concurrent users, where security is not a concern, and where applications are not transaction-oriented. A companion Visual Tool Decision Worksheet recommends developing with Visual FoxPro only when you require Macintosh portability or rapid development, but not if you are new to Microsoft visual tools. On the development tools side, it's a much cleaner story to pitch Access for small applications, Visual Basic for medium applications, and Visual C++ for large or packaged applications. On the database side, it's much easier to promote JET for fewer than 20 users and SQL Server for anything else. Simple stories resonate.

Okay, then, so how did Visual FoxPro 5.0 become such a great product -- and I think it is -- and why did Microsoft announce that the Fox team is working to build another version? Just as the Fox team is very talented, I have a very high regard for the FoxPro development community. By and large, FoxPro developers have been FoxPro developers for a relatively long time. Typically, they come from business rather than computer backgrounds, and their work is more business- than systems-oriented. This group's creativity is shown in the way it has built a wide array of tools and development frameworks that fill in the gaps left by shortcomings in prior versions of the product -- and by the unwillingness of commercial development infrastructure tool vendors to support those versions. The Fox team at Microsoft and the development community feed off of each other, with the result that Visual FoxPro 5.0 is an extremely creative, solutions-oriented tool. Microsoft seems to be accepting that FoxPro will persist, and the company is beginning to take all of these folks' efforts a little more seriously. VFP 5.0's full citizenship in the gallery of ActiveX technologies reflects this change of attitude.

VFP 5.0 Catches Up

Visual FoxPro 3.0 was not a finished product. It had performance problems when rendering controls, and it did not fully support OLE controls and OLE automation servers. The main objectives of the version 5.0 development project were to correct performance problems and to address areas where VFP lacked capabilities that were available in competitive tools. In the final analysis, however, the whole is greater than the sum of its parts. VFP 5.0 sports several team-oriented development enhancements that bring the product into line with other application development tools. As in Visual C++ and Visual Basic, the VFP project manager now integrates tightly with Microsoft's Visual SourceSafe version-control system, promoting coordinated work on application components in a multi-developer environment. VFP's database container now permits a developer to make changes in metadata while it is open for other users or developers (exclusive access is no longer required). One of my bugaboos about modern-day development environments is that they force you to write numerous, relatively small but sometimes complex event and method procedures using a very bare-bones editor. In the migration from linear character-mode applications to event-driven Windows applications, we lost the ability to use the editor of our choice. In VFP 5.0, the enhanced program editor becomes an effective substitute for my favorite editor, with the ability to color-code keywords, literals, strings, variables, and operators; to generate a list of all of the procedures and functions in a module and jump to one in the list; to automatically indent and comment selected blocks of text; and to open an expression-builder dialog where you can create complex formulas and paste them into your code. Rounding out the enhancements to the developer environment, the debugger is now worthy of its name. It includes separate windows to display watch points, properties of active objects, a call stack, and a trace window. During debug sessions, you can have the debugger record all instances of selected event firings or generate reports of which blocks of code are exercised.

Creating Some Excitement Where VFP gets exciting is in its ability to make ActiveX development accessible to high-level business application developers like me. By allowing developers to create Automation servers, VFP lets them leverage its powerful data processing power in other development environments. To demonstrate this capability, I created a single-form application and instantiated it from a Visual Basic application, using the CREATEOBJECT function that is common to the Microsoft languages. My little demonstration application tied together a series of VFP's strengths. The database view behind my form was based on a multi-table join against a Microsoft Access for Windows 97 database. I assigned the form's new ShowWindow property the Top-Level Form value, which caused it to run outside of Visual FoxPro's context. I displayed the results of the join in VFP's dynamic grid control, which allowed me to embed a edit control in one column to display a memo field inside each of the cells in that column. To make the form available for instantiation by controller applications such as Visual Basic, I had to create a class with the OLE Public attribute and give that class a single task: Run my form. Then I built an executable from the project manager, which created an out-of-process OLE server. The build dialog also offers the option of creating an OLE DLL, which creates an in-process OLE server that would run within the address space of the client application. During the build process, VFP registered an OLE-type library for the viewer class that I declared as OLE public. In my project settings, I had the

option of creating a multi-use class or a single-use class. In a multi-use class, each request for an instance of the class by an OLE client will check if the server is already running and create the instance from an already running server if it finds one. I could also choose to deploy Automation servers I create with VFP on application servers where my users can share them. VFP uses the same remote automation control manager as Visual Basic to support this remote automation capability.

Taken together with some of its other features, VFP's OLE automation capabilities have some profound ramifications. For one, I can instantiate VFP's application object and assign it to an object variable in a OLE controller application. Through the DoCmd, Eval, and RequestData methods, I can execute the full range of Visual FoxPro capabilities from inside that application. This functionality, together with the ability to create an application without any user interface, means that I can create application servers that run in the background or on a network server that quietly receive data values as input and perform a series of tasks. VFP is ideally suited for use in this manner. The only drawback I can see is that it's not multi-threaded, so that each instance of the out-of-process executable will require a separate allocation of system resources.

VFP includes an intriguing demonstration of its use as an Automation server that works with Microsoft's Internet Information Server ISAPI API. Part of this demonstration is a C++ program that supports ISAPI and whose sole purpose is to invoke OLE Automation objects. There is also a VFP project that includes a base-class form that is not declared as OLE public, and a sample application form derived from the base class that is. The base class includes a series of methods that do a lot of work: The startup method creates a session-tracking database and calls another method that generates an HTML representation of the active form. For each entry field on the VFP form, the generator creates a row in a table with a literal value in one cell and an HTML input variable in the other.

The process of generating the HTML representation of the form leverages VFP's sophisticated text merge feature, which merges literal text and embedded expressions. It also leverages the ease with which VFP developers can call Windows API functions, which it does in this case in order to read the FOXISAPI INI file. When the Web browser triggers the link to the desired form, the ISAPI DLL calls the Automation server, which renders the form as HTML with the fields populated for the selected row in the appropriate table. Controls on the form enable navigation through the record set as well as updates.

Database Engine Enhancements

Database engine enhancements should also interest DBMS readers. Fortunately, there are several to consider, including addition of explicit join syntax to VFPs SQL-SELECT statement, the ability to formulate TOP queries, the ability to display execution plan information, a FORCE clause, and offline views. Since version 2.0, FoxPro has offered a SQL-SELECT command for constructing queries, in addition to using the Xbase style record navigation and table filtering semantics. Although not quite consistent with standard SQL SELECT, FoxPro's SELECT offered substantial functionality and facilitated working with both FoxPro engine-based tables and SQL-based tables from inside FoxPro. Version 5.0 advances the tradition

of a powerful, albeit idiosyncratic, `SELECT` statement.

The most significant addition in this version is a `JOIN` statement that provides a subset of the functionality in the `SQL-92` specification. In previous versions of FoxPro, you would indicate relationships between source tables in the `WHERE` clause; the explicit `JOIN` clause now provides control over the type of the relationship and the order in which multiple relationships are processed. Relationships defined in the `WHERE` clause were interpreted as inner joins, but the `JOIN` clause provides `LEFT` and `RIGHT OUTER JOIN` syntax to enable constructing sets that include all of the members of one table and only matching members of the other. The typical syntax for joins is `SELECT FROM JOIN ON JOIN ON .` (See Figure 1.) VFP creates an intermediate result set from the first join and then joins that with the third table. Using this syntax permits you to manage the order in which VFP processes join expressions, which was essentially impossible in previous versions.

VFP also provides control over query execution by providing a `FORCE` clause. When `FORCE` precedes a table name, that table is joined first to the next table specified. `FORCE` overrides the VFP optimizer, which makes its decisions based on guesses as to the size of intermediate result sets. A scenario in which you might want to `FORCE` the order in which VFP evaluates conditions is when you expect a query to return only a few rows from a large table and many more rows from a smaller table. In this instance, it may be faster to direct VFP first to create a small subset of the larger table, rather than to accept its normal behavior, under which it would try to process the smaller table first. `FORCE` is a VFP-specific extension and not an ANSI standard, so it can't be used in SQL passthrough queries or remote views.

Historically, constructing optimizable multi-table queries in VFP has been a fairly arduous process. Its optimizer is very particular about only using criteria or join conditions that exactly match index key expressions. FoxPro users have long requested a tool to report on the order in which query conditions are processed and which indexes are used. VFP 5.0 provides a new function that enables on-screen display of which indexes are being used and the order in which joins are processed. This diagnostic tool should be used outside the context of a program to report sequence of intermediate result sets in a multi-table `JOIN` expression; it will provide useful insight into how VFP is responding to your attempts to communicate your intentions via nesting and parentheses.

Another useful feature for developing analytical applications is the ability to create a query that returns only a specific number of rows based on their rank order. VFP 5.0 provides the `TOP` clause in its select statement, which delivers this functionality. can be a specific number or a percent of the total number of rows in the result set. VFP's `TOP` does not provide any fine-tuning control over its operation. For instance, it deals with ties by returning all of the rows that include the tie value, so it's possible to get more than rows in the result, but it stops as soon as it finishes delivering rows with the last tied value. If you want to get the top distinct values, you must construct your own set of queries to do so.

Offline Views

Offline views present a built-in way to support users of applications who have

transient connections to their servers. Traveling sales and service personnel carrying around subsets of company databases on laptops spring immediately to mind. In fact, another such category of users is those whose database must be taken offline for routine processes such as backing up or reindexing, which require exclusive use of tables. From this perspective, offline views provide a way to extend VFP into environments where round-the-clock operations are mandatory. In VFP 5.0, one function takes a query-based view and converts it into a table that is accessible to the user that created it. Subsequently, VFP will route references to the view to the new table, and it will permit the user application to interact with this table as if it were the underlying view, adding, deleting, and editing data. When in a position to reconnect to the home database, the user would open the table with the ONLINE keyword, which would load pending updates into the underlying table. Typically, all of this would happen under your applications control; one approach would be to enable FoxPro's row and table buffering facilities and write code to compare change timestamps and resolve conflicts with changes made since the data was taken offline.

Server Connectivity Enhancements

Microsoft has also worked to improve VFP's behavior as a client to common RDBMS products. VFP 3.0 introduced an upsizing wizard that automated the conversion of a FoxPro database to SQL Server. The upsizing wizard built scripts that would define the schema of the tables on the server, optionally executed them on the server to create the tables and indexes, and bulk-loaded data from the FoxPro database to Microsoft SQL Server. VFP 5.0 adds an upsizing wizard for Oracle. Having spent a fairly significant amount of time and money trying unsuccessfully to build such a thing about a year ago, I appreciate the effort, and I'm grateful for the breadth of functionality that the Oracle upsizing wizard provides. In addition to doing default data type mapping, the wizard allows selection or creation of the tablespace to move your FoxPro database onto, and it supports creating Oracle clustered tables. In addition to creating tables and indexes and moving data, you can choose to migrate persistent relationships as foreign keys, referential-integrity constraints, and validation rules into CHECK constraints. Another significant enhancement to using VFP as a client/server development tool is the ability to pass and receive parameters to stored procedures on SQL Server. The inability to process output parameters was a serious restriction to developing scalable client/server applications with VFP 3.0. VFP 5.0 essentially uses ODBC escape-clause syntax to call stored procedures, with the name of the output parameter preceded by ?@. A few other nice features have been added to VFP's client/server toolkit: VFP now passes through the ODBC SQLPREPARE() function, which was missing from VFP 3.0; this new setup enables precompiling SQL statements on the server for later execution. Views defined against server tables now support a FETCHASNEEDED property; with this property, VFP automatically retrieves additional rows when the user scrolls beyond the number of rows initially fetched in accordance with the setting of the FETCHSIZE parameter.

The Sum of the Parts

I've tried to describe a wide range of enhancements and distill them into a discussion of what's significant about the VFP 5.0 upgrade. I neglected, however, to address fairly extensive improvements in the forms design environment and forms runtime engine. I also neglected to discuss how the VFP development team has obtained

noticeable improvements in both design-time and runtime performance, which should restore VFP's position as the fastest development tool available. These improvements, although important to developers, are not strategic and probably not the primary concern of IS decision-makers. More to the point are questions about whether VFP can be an effective part of a consistent tools and database strategy. Corporate information systems require scalable three-tier tools optimized for Internet deployment. In this environment, separating programming tools from database products, and providing robust APIs for implementing database applications with the development tool of choice, is essential. VFP's tight integration of programming environment and language from database engine works well for LAN applications, but it's not scalable. As a database engine, VFP requires tools for replicating data from a server RDBMS to the local engine, and closer adherence to SQL standards, to become a viable corporate solution.

VFP 5.0 is a great tool for custom application developers working to provide solutions for free-standing, medium-sized businesses. For VFP developers, the integration of language and engine makes for a series of extremely productive design surfaces. VFP gives developers an environment where they can build reusable application components and use those components to reduce the time it takes to complete and deploy custom applications. By adding reasonably good ActiveX and COM support, Microsoft has given these developers a good reason to stay with the product, as well as a viable way for them to start implementing applications that adhere to Microsoft's vision of distributed application architecture.

(Source: *DBMS Magazine*)

Tom Spitzer is Vice President, Product Technologies at *The EC Company*, a Silicon Valley startup entering the electronic commerce marketplace.

[Home TOC](#)

If you have any information that can help us to compile this history, please, [send us a message](#). Thanks!