Figures:
File for Subscriber Downloads:

What's *Really* New in VFP 7.0

Whil Hentzen

By now you may well be sick and tired about hearing about VFP 7.0. It's been demonstrated ad naseum for nearly two years now, and you probably think you know it all by now.

However, there's a technique that the various product teams in Redmond use when demonstrating unreleased products that you may you may not be aware of. The timeworn tactic for pre-release demos is to gradually open the kimono, bit by bit, instead of ripping it off all at once. That's why you saw the first sneak preview to VFP 7 at DevCon 10 in Palm Desert, but then a longer demonstration at DevCon 11 in Miami in 2000. However, the kimono wasn't completely open even then; at FoxTeach 2001, it was announced that the Setup Wizard would be replaced with a specially designed version of InstallShield. But not even at the Canadian Launch of VFP 7 was it all revealed.

The product team has saved the best for last, and has asked, once again, FoxTalk to disclose the last new feature added to VFP 7. And this one comes back to where we live – data.

To fully appreciate this new feature, I need to discuss a bit of history, and I need to explain a bit of complex computing as well.

History of the Fox, Part I

FoxPro 1.0 gave IBM PC developers a GUI on top of a DOS foundation. We suddenly had GUI-style controls, like command buttons, option groups, and combo boxes, just like a Mac, available to our dBASE III+ applications. Better, though, was the addition of SCATTER and GATHER to take the place of the timeworn REPLACE command. You could move data in and out of an entire record with a single command, instead of a laboriously hand-coded routine with dozens of field names.

FoxPro 2.0 then went deep under the hood, and gave us Rushmore. Suddenly tens of thousands of records were scoffed at, hundreds of thousands of records were dismissed in short order, and even million record tables were only grudgingly allowed to be considered 'large.' Rushmore technology could access any record in a blimptosecond, and we could write apps that were comparable, dollar for dollar, with any system on the planet.

FoxPro 3.0, called Visual FoxPro, introduced a new data feature called buffering. Prior to the introduction of buffering, developers had to resort to any number of intricate constructs in order to handle multi-user record locking. One typical approach was to create a temporary record that held the original data, and then use that temporary record as a baseline for comparison after multiple users had made changes in order to determine which data ought to be saved. Buffering was Visual FoxPro's built-in version of this temporary record concept.

The developer now had the choice to use the old-fashioned scatter/gather, use no buffering at all, use pessimistic, or optimistic locking with row or table level buffering, and attach that to a table or a form.

History of Computing, Part 27

In the beginning of computing, there were mechanical devices. Expensive and hard to construct, these never became popular. Then the first computers featuring vacuum tubes, such as ENIAC, were introduced. They worked, but were difficult to maintain because vacuum tubes were very fragile. Then the transistor was invented, and the IC, and computers became smaller and faster. Gordon Moore, one of the founders of Intel, postulated the well known law that computing power would continue to double every 18 months.

However, even silicon and other elements have limits, and electrical engineers and computer scientists are thinking that this limit might be reached in the next ten to twenty years. As a result, people have been experimenting with a completely new type of computer called a quantum computer, based on quantum theory.

Quantum theory, essentially, states that a quantum computer processor can answer multiple questions (or perform multiple calculations) at the same time, instead of sequentially, as we normally think of processors working now. The details of how that works are complex, but one offshoot of this is that data can

occupy multiple values at the same time, and the final value will be that which the user views at the completion of a calculation.

The last new feature of VFP 7

A couple of the rocket scientists in Redmond who are responsible for thinking great thoughts about data were talking about these quantum computer processors last fall, and one of them made the jump from individual quantum bits to quantum bytes – and then to quantum fields in databases.

Fox being the favorite testing ground for advanced technologies at Redmond, they immediately proposed the concept of quantum buffering to the Fox product team. The slowest part of a database is the writing of data, because there's a lot going on – finding the record, writing to disk, confirming the write, and all of it involving that yucky hardware.

Why, they asked, couldn't we improve the speed of a database by introducing quantum buffering during the read and write of database accesses? In other words, why can't we have a data field contain both values – the original value and the saved value – at the same time? This would speed up processing significantly, wouldn't it? Naturally, the Fox team was all ears when it came to improving the speed of their database tool, and a working prototype was put together within six weeks of the initial proposal. Just as when Rushmore was released, the specific internals are a secret, but here's a rough sketch of what will happen as far as the developer is concerned:

Before a read or write operation occurs, a field will contain a value, say, "A". Then the user will perform an action, causing a read or write operation to occur. Invoking the laws of quantum theory allows that field to contain both the original and the new value at the same time. Once that operation has been completed, the field will contain the appropriate value. The user will view the result, and see the final answer.

In other words, there will be a piece of data on disk, and when accessing that record, the record will be quantum buffered so the user can view the data – whether it's the original value or the new value immediately – before the data is actually read. However, the actual value of the data will not be known until the user reads the record, and the value will only be decided at that point.

Performance Improvements

You may be skeptical at this point. "This sounds pretty good in theory, but what's the performance really like in the real world?" Given that this is still in beta, the final results aren't in, but so far the results are very promising. The speed improvement in a read is approximately a factor of 15 – that's 15 times faster. The speed improvement in a write is much more significant – by a factor of 80 to 90 times – that's almost 100 times faster.

The only downside is that the value of the data is unknown – it can contain both values - until the user sees it again. This may be a problem for systems that rely on or retrieve data without the user viewing it first – such as compilation queries, and so on. At this point, however, this merely appears to be an implementation issue.

Fox has always been fast – the fastest kid on the block, in fact. With this new advancement of quantum buffering, Fox will be even faster than ever. I'm reminded of the limerick that played on Einstein's theory of relativity:

```
There was a young lady named Bright
Whose speed was much faster than light
She set out one day
In a relative way
And returned the previous night
```

With quantum buffering in Visual FoxPro 7, we'll be able to execute a query, say, in the middle of the month, like April 18, and have the results returned days or weeks before – like on April 1st. I know I can't wait until the product ships!