Figures:
File for Subscriber Downloads:

Still Using DBFs? Get Real.

Whil Hentzen

There's been a lot of talk recently about how using DBFs is not a good thing. And there are a lot of things wrong with a DBF.

First of all, a DBF is prone to corruption. Everyone has a favorite story about how a memo file became disassociated with it's parent DBF, and gobs of data was lost. The indexes aren't any better – a fractured CDX file can cause all sorts of weird, difficult-to-trace problems. This is why our industry has all sorts of repair utilities – like, dSalvage, and FoxFix, and… Ooops, I guess those are 2.x utilities, aren't they? As far as 6.0 goes, I can't think of any applicable utilities right off the top of my head, but I'm sure there are dozens, because this corruption is such a problem.

Second, the DBF file format doesn't have any intelligence. If you want to create a table without a primary key, you can do it. If you want to generate a primary key, you have to write the code yourself. If you want to stuff the same primary key value into multiple rows, there's nothing preventing you from doing so.

Third, if you want relations between files – the type of relations that Codd and Date are interested in, not Clinton and Kennedy – you have to handle that yourself. You have to define the primary and foreign keys, you have to write the code to handle the joins between tables, and you can, thus, do all sorts of bad things.

Fourth, there's absolutely no inherent security available for a DBF. If you want to protect the contents of your DBF, good luck. You have to roll your own security, or buy a third party tool that may or may not integrate very well, and of course you'll suffer a performance hit with such a solution.

And, finally, it doesn't have that sexy MDB or MDF extension, and, just like in Detroit, image is very important in our business. (Note to self: Is it obvious yet that I'm being sarcastic, or should I be more explicit?)

Sure, you might argue that you can get around all of these problems using a database container, but that's really what you're doing – getting around the problems. The database container isn't a true database, for many reasons that have been discussed here and other places ad nauseum, and you still have to do a lot of work yourself to address these issues.

In addition, the increased friendliness and performance of SQL Server make it a great choice for applications that are growing up, or that need increased security needs. Most of my customers' apps have moved to SQL Server over the past three years.

But SQL Server (and other alternatives) are not without their costs. SQL Server, even in it's current form, requires at least a part time DBA – while applications running on Fox data can run forever without the interference of an administrator. There are a remarkable number of FoxPro for DOS apps still out there whose major problem is dealing with upgrades to the operating system.

And a true database requires a big fat wallet – SQL Server licenses cost a lot more than the freely available DBF – and the corresponding bill from Oracle will cause you to drop your teeth in your soup. And the cost isn't just the license – but the administration needed to keep track of how many seats or users are on, and what happens if you add a group of eight more users from another department to the system.

"I'm not quite dead yet. I think I'll go for a walk."

OK, so I was kidding earlier. Sure, the DBF isn't perfect. Neither is my lawn, but that doesn't mean I'm going to install AstroTurf in the front yard. The DBF still has a lot of things going for it.

First of all, everyone and their brother knows how to get data in and out of a DBF. There are hundreds of thousands of old dBASE programmers who can still manipulate a DBF without even thinking about it.

There are lots of data types – sure, everyone can think of one more they'd like to see – but for the most part, you can stuff most any type of data you want in a DBF. It's well suited for writing relational programs, and the last time I looked, you were all database programmers – isn't writing relational programs what you do for a living?

And as we all know, the combination of Visual FoxPro and a DBF make up the very fastest desktop database program on the planet. (Hey, that's a cool phrase!) I'm still fond of remembering Microsoft demonstrations of Access a few years ago – they'd open up a form on a brand new 486 notebook, and query a Zip Code, and get the results back in a tenth of a second – that's right – a query on a 45,000 record table would be returned by the time you heard the click of the mouse.

Then another Microsoft program manager would do the same demo somewhere else with Visual FoxPro, and get back their results in the same tenth of a second or so. Except that the query wasn't against a 45,000 record Zip Code table, it was against a 1.6 million record table that contained every city street in the entire country.

Nowadays, other databases have gotten faster – Fox's Rushmore has made it, piece by piece, into Access and SQL Server. But just because other tools have been catching up doesn't mean that Fox has gotten slower.

And, finally, it's got a standard format used the world over – everyone has an import/export facility for DBFs. You've probably read my articles on Satellite Forms, a forms package for the Palm OS that uses DBFs as the data repository for the piece that runs on the PC.

But there's another very popular program – one that's probably on your PC right now – that's been using the DBF file format to store all of its data – and I'll bet you didn't even know it. Open up Explorer, drill down into Program Files, Real, Real Jukebox, db, and you'll see that the list of files that store your music files all have DBF, CDX and FPT extensions.

CD, CDTRAX, NAMES, PLAYGRPS, PLAYLIST, TRACKS, TRAKINFO, VALUES – yes, there're all there, and all available for you to spelunk through. Before you get all super-excited, note that the CD DBF appears to be either encrypted or otherwise contains data that's not in human-readable format. Still, the mind boggles at the possibilities (that's a hint for someone out there to investigate what's in there and write an article for us.)

So I'd say the reports of the death of the DBF are greatly exaggerated. It's not for everybody, and it's not a panacea for everyone's data storage needs, but nothing is. And for many purposes, it's the perfect repository. Don't give up on it yet.