June, 2005

Advisor Answers

Memo field bloat

VFP 9/8/7

Q: When SET EXCLUSIVE is OFF, I get memo bloat. However if SET EXCLUSIVE is ON, the .FPT file does not increase in size. I set up a test with a free table located on a file server. It had 2 fields, an integer and a memo. I placed about 100 characters of text in the memo field in record 1. (There were no other records in the table.)

My code looped 10,000 times, scattering, then gathering that one record. With SET EXCLUSIVE OFF, the FPT file grew from 832 bytes to 3,200,832 bytes. Repetitive attempts increased the FPT size every time.

With SET EXCLUSIVE ON, the size of the FPT file did not change. If I ran this test with the bloated FPT, the size stayed the same, and if I reduced the FPT with PACK MEMO, the size stayed the same. In other words, it appears that with SET EXCLUSIVE ON, memo bloat does not occur. (Also, the performance over the network improved dramatically with SET EXCLUSIVE ON.)

Is there a technical explanation for this behavior?

–Steven Merar (Naperville, IL)

A: You've run into one of the two main issues regarding growth of memo fields in VFP. Both behaviors go back to the early days of Fox and, while annoying, are actually necessary.

When EXCLUSIVE is ON, the VFP engine can be sure that no other session is reading or writing to the memo field as you replace it. Therefore, if the data fits, the engine can put it back into the same position in the file.

However, when EXCLUSIVE is OFF, it's possible that another VFP session (or even some other application using ODBC or OLE DB) is reading that data at the time you want to write it. Therefore, it must store the data in a new location on the disk, and update the pointers to point to the new data. Doing so allows the read in progress to be completed (as the other session won't see the update right away), and allows your session to save the change.

Thus, in a multi-user system in which memo fields are saved (as opposed to only being read), some memo field bloat is inevitable.

The second thing that contributes to memo field bloat is frequent rewriting of the field. I encountered this a few years ago with a developer who was running into so much bloat that his application was becoming unusable. (I think he was running into the 2GB file size limit.) It turned out he had code that looked something like this:

```
REPLACE MyMemo WITH MyMemo + "Something"
REPLACE MyMemo WITH MyMemo + " else. "
REPLACE MyMemo WITH MyMemo + "More stuff"
```

The actual code segment he showed me went on for 20 or 30 lines, each of which stuffed a little more data into the memo field. (Of course, in the application, it wasn't just fixed strings—it was data derived from a business process.) The problem here is that each time he resaved the memo field, it was moved to a new location on the disk. Even if this hadn't been a multi-user system (which it was), the increases in the size of the memo field would have caused the problem, though it might have been less acute.

I convinced him to change his code to look more like this, with the memo field updated only once:

```
cAdditions = "Something "
cAdditions = cAdditions + " else. "
cAdditions = cAdditions + "More stuff"
REPLACE MyMemo WITH MyMemo + cAdditions
```

He made the change, and the problem was immediately resolved.

You have some control over the way VFP allocates space in memo fields, even when working in a multi-user environment. The SET BLOCKSIZE command lets you determine how much space is allocated at a time. The command's syntax is simple:

```
SET BLOCKSIZE TO nSize
```

nSize can be any number from 0 to 32,768, but its meaning varies with the value you specify. Table 1 shows the various possibilities:

Table 1. Specifying memo field allocation. The meaning of the blocksize setting varies with the value specified.

nSize	**Meaning**
0	Allocate memo fields one byte at a time
1-32	Allocate memo fields in blocks of nSize*512 bytes. That is, specifying nSize=2 allocates memo fields 1024 bytes at a time.
33-32,768	Allocate memo fields in blocks of nSize. So specifying nSize = 40 allocates space for memo fields 40 bytes at a time.

Although you can control BLOCKSIZE, the truth is that setting it to 0 is almost always the best choice. For shared access, every replacement will move the data, so a blocksize of 0 minimizes the new space allocated. For exclusive access, you're still likely to find a blocksize setting of 0 the most efficient. The one case where a different setting might make sense is when you're frequently adding a little bit of data to memo fields. In that situation, having the extra space allocated could speed your application up at the cost of a little bit of disk space.

Block size applies at the time you create the table. You can't change the block size of an existing table, except by rewriting it with a command like COPY TO, MODIFY STRUCTURE or PACK.

–Tamar