This free back issue of
THE QUICK ANSWER is provided courtesy of…

The Monthly Newsletter for Sesame Database Manager
http://www.insidesesame.com
Read a Free Issue of Inside Sesame!

AND

Makers of Sesame Database Manager
Compatible with Symantec Q&A™
http://www.lantica.com

The Ultimate Application Switcher

TOM MARCELLUS

The ability to swap application macro files on-the-fly hasn't been an easy feature to implement—until now.

WHEN you design Q&A applications for others, your top priority has to be ease of use. And a major step toward that goal is to give users a brutally simple way to navigate between the various applications or modules that comprise the Q&A system you're providing.

Suppose you're designing a system that includes modules such as customer maintenance, order entry, accounts payable, inventory control, and payroll. Each of these is a separate self-contained application, so each might have its own macro file and custom main menu. (Here, the terms *module*, *application*, and *macro file* are synonymous, and I'll use them interchangeably. See the sidebar on page 4.)

At the top of the whole system should be a *master menu* that makes switching between modules a no-brainer. That is, if a user entering sales orders needs to quickly check a customer's record, it should take no more than a couple of keystrokes or mouse clicks to pull up the main menu for the customer database. It should *not* require typing filenames or having to remember which special keys to press.

What makes module-switching such a hassle

The problem with switching modules is the effort it takes to load the macro file containing the custom menu(s) and macros for the module you're switching to. Q&A offers no easy way to do this. You must either use the Macro Menu, or have some way to exit and restart Q&A (or another copy of Q&A) using switches to load the new macro file and custom menu.

The Macro Menu method is hard on users. They have to press Shift-F2, select *Get Macros*, then type in the name of (or navigate to and select) the macro file to load—assuming they know *which* macro file they need and *where* it is. But they're still not done. Now they have to remember the shortcut key combination that displays the main custom menu for the new module. To switch to another module, they have to go through this gut-wrenching process all over again.

As for the exit-and-restart option, techniques range from devoting a separate Windows desktop shortcut (or Main menu Alternate Program slot) to each module, to overwriting the default macro file via batch files (one for each module), to starting another copy of Q&A for each module change. All these techniques, in my view, suffer from major drawbacks, which I won't go into here.

I've long suspected that there must be an easier way to swap macro files—one without drawbacks that anybody with little or no Q&A experience can use to quickly navigate between modules. And it turns out *there is* such a way. The keys are two simple batch files, and Q&A 5.0's ability to modify a batch file on-the-fly. Let's start with the batch files.

A one-two punch

Starting Q&A with a batch file is a common practice. But by starting Q&A with a Windows 95/98 desktop shortcut, you can do much more. In your

Q&A shortcut's properties, you can specify not just one but *two* batch files to run when you start Q&A. Figure 1 (page 3) shows the Program tab for a sample Q&A 5.0 shortcut. The pertinent portions of that tab are the following two fields that specify different batch files:

```
Cmd line:    D:\QA\SWAP.BAT

Batch file:  D:\QA\DEL.BAT
```

With your shortcut's Program properties set up this way (your paths

Letters

Toggle Display On and Off in Macros

In the "Macro Prevents Display Update" answer in the June 1999 @Helps, you suggest that the macro's **<vidoff>** be changed to **<vidon>** so that Q&A will display the target record even if it's in use by another. Actually, you could leave the <vidoff> at the beginning of the macro, and add the <vidon> just before the final <f10>. This will keep the interim screens from flashing by and still display the record in view-only mode. The macro might look like this:

 <begdef><nokey><name>"Go<sp>to<sp> cust<sp> record"vidoff>...*actions*...<vidon> <f10><enddef>.

—CARL UNDERWOOD

What Carl points out is true. However, we've noticed that some Q&A 5.0 macros can inexplicably fail when video is off. At this point, we don't know whether it's a bug, if certain commands in a macro might trigger the failure, or if it might be processor- or video-related. We "err" on the side of caution by keeping video on in all our Q&A 5.0 macros. The problem does not appear in Q&A 4.0 macros.
—*Bill Halpern*

Merge Large Q&A Text Fields with Word Docs

Thank you for your May 1999 article on mail-merging Q&A data in MS Word documents. I discovered a potential problem and solution that may be of help to your readers.

I need to mail-merge one Q&A record at a time. When exporting to dBase IV format, your procedure works fine for money, date and short text fields. But if you include a multi-line field, any characters after about the 256th are lost. Limiting my Q&A database fields to 256 characters was not an option, so I found another way.

I modified the macro that exports the Q&A data to a dBase IV file, adding steps that separately export my large text field and assign it a consistent name, such as MERGE1.TXT, overwriting the previous version of that file. (This exports just the lengthy field to Standard ASCII, using the *No quotes* and *Return as field delimiter* options.)

When designing the MS Word merge document, I went to where I wanted the data from this large text field to appear, and used Word's *Insert/File* command to insert MERGE1.TXT. This way, whenever I open the same Word document (after my Q&A macro has generated new dBase and MERGE1.TXT files from a different record in my database), the new MERGE1.TXT file automatically appears without my having to do anything. When I then run the merge, I end up with not only the new data in the form letter, but also with the new large text field appearing at the original insertion point.

The Word document is designed to be sent to customers as an email attachment, so each time I run the merge, I click on the *Merge to File* button on the Merge toolbar and give the document a unique filename. I use the order number from my Q&A database, so a document might be named something like AB123456.DOC.

Variations on this procedure could include exporting more than one large field from Q&A for auto-insertion into the Word document. Even with several large text fields, the process should run quickly and produce the MS Word results you need from your Q&A data.

—AL BEAMER

Swap Macro Files... continued from page 1

and batch file names may be different), when you double-click on the shortcut to start Q&A, Windows will run DEL.BAT first, followed by SWAP.BAT.

Here are the contents of DEL.BAT:

```
ECHO D\QA\QA > SWAP.BAT
```

Here's what SWAP.BAT contains:

```
D:\QA\QA
```

Here's what happens. When you start a new Q&A session, Windows runs DEL.BAT, which overwrites SWAP.BAT. SWAP.BAT contains "D:\QA\QA." Windows then runs SWAP.BAT, which starts Q&A in the usual way with the default macro file (QAMACRO.ASC) in memory. (You might know enough about batch files to realize that SWAP.BAT can intially be empty since DEL.BAT overwrites it.)

Now, when you switch modules in Q&A, a special command I'll show you later *adds a new line* to SWAP.BAT and exits Q&A. Here's what SWAP.BAT might look like after you've switched between modules five times during a Q&A session:

```
d:\qa\qa
d:\qa\qa -ald:\qa\macros\cust.asc -m6
d:\qa\qa -ald:\qa\macros\payables.asc -m5
d:\qa\qa -ald:\qa\macros\orders.asc -m6
d:\qa\qa -ald:\qa\macros\inventry.asc -m6
d:\qa\qa -ald:\qa\macros\cust.asc -m6
```

Remember that Windows initially started Q&A via SWAP.BAT's sole command (D:\QA\QA). Each time the special command adds a line to SWAP.BAT then exits Q&A, Windows notices that there's *one more line* in SWAP.BAT that hasn't been run, so Windows runs it. This restarts Q&A with the macro file for the new module now in memory, and the module's
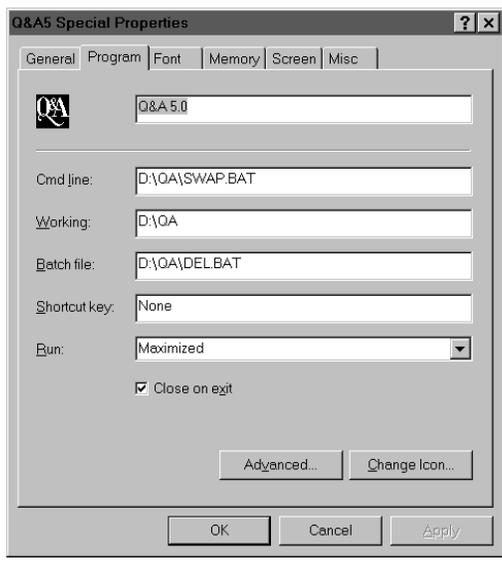
Figure 1. The Program tab of a Q&A shortcut properties sheet. Two batch files are specified. DEL.BAT runs first, followed by SWAP.BAT

main custom menu now displayed, ready for you to select the task you want to perform within the new module.

When you finally quit Q&A by exiting in the usual way through the Main menu, Windows sees that there are no more new lines in SWAP.BAT, and so doesn't restart Q&A.

But when you start your *next* Q&A session, DEL.BAT again overwrites SWAP.BAT and you start once more from the ground floor with the default macro file in memory.

What all this means is that you can easily (two clicks) switch modules in Q&A as many times as you need during the course of a Q&A session without incurring any overhead, and still start each new session the same way every time.

If you grasp these dynamics, you can see the potential power a technique like this offers. I'm showing you just one way to take advantage of that power.

Alternate Program slot needed

Earlier, I mentioned a *master system menu*—a user interface that facilitates fast and easy module-switching. You want to make displaying and using the master menu an absolute no-brainer. For the master menu, I use a one-record Q&A database named MODULES.DTF. Figure 2 shows a sample MODULES master menu, and Figure 3 shows MODULES on the Design screen.
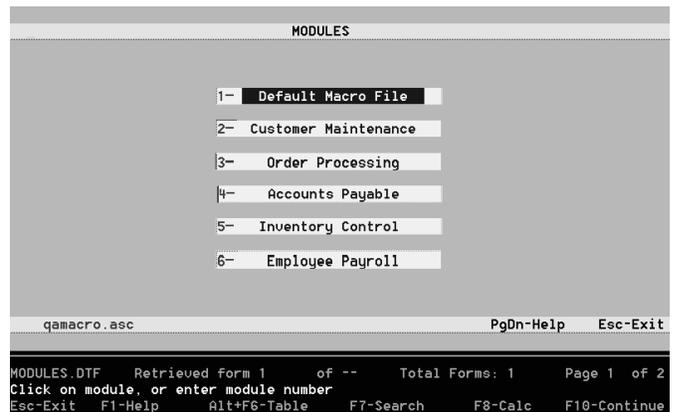
Figure 2. MODULES.DTF (the master menu). This sample screen shows that the default macro file (QAMACRO.ASC) is active.

Figure 3. MODULES.DTF on the Design screen.

You display the MODULES master menu via an Alternate Main menu program (Utilities / Alternate Programs). The last selection on the Alternate Programs screen might look like this:

```
Alternate program 6: altk
Menu selection.....: Modules
```

The Alt-K macro (you can use a different key combination if you like) that displays the master menu looks like this:

```
<begdef><altk><name>"<caps,>altk<caps.>"<vidoff>fs
<capsf4><f10>MODULES.DTF<enter><f10><enddef>
```

Each swappable macro file in the system must contain this Alt-K macro in addition to at least two other macros. For example, the first portion of the macro file for the *Customer Maintenance* module (CUST.ASC) might look like this:

```
<begdef><altk><name>"<caps,>altk<caps.>"<vidoff>fs<capsf4>
<f10>MODULES.dtf<enter><f10><enddef>
*
<begdef><nokey><name>"Swap<sp>macro<sp>file"<vidoff><esc>
y<esc>x<enddef>
*
<begdef><alt6><name>"<caps,>alt6<caps.>"<vidoff><menu>
"*Main"<enddef>
*
<begdef><nokey><name>"Main"<menu>"CUSTOMER<sp>
MAINTENANCE,FAN:/1<sp>-<sp>Selection<sp>1,/2<sp>-
<sp>Selection<sp>2,/3<sp>-<sp>Selection<sp>3,/4<sp>-
<sp>Selection<sp>4,/5<sp>-<sp>Selection<sp>5,/6<sp>-
<sp>Selection<sp>6,/7<sp>-<sp>Selection<sp>7,/8<sp>-
<sp>Selection<sp>8,/9<sp>-<sp>Selection<sp>9,/"<enddef>
```

The first macro is the Alt-K macro. The second one is a nokey macro named *Swap macro file.* Its job is to escape from the menu and exit Q&A. The third macro displays the startup custom menu (named *Main*) for the module, and the fourth macro contains the structure for that *Main* custom menu. (What's not shown in the above sample macro file are all the other macros that are specific to the customer maintenance module.)

In this case, by using the master menu to switch to the customer maintenance module, the next thing you'd see is the Customer Main menu shown in Figure 4

Why Multiple Macro Files?

There are advantages to using several smaller macro files (one per module) rather than a huge global one:

- It's easier to keep track of numerous macros in relation to the modules that use them.

- You can reuse key identifiers (Alt-A, Alt-B for example) and macro names in different macro files.

- There's little chance of a macro designed for use with *another* database or application being inadvertently triggered and damaging something.

- Q&A doesn't have to devote as much memory to macros. With a single global macro file, you can run out of macro buffer memory at around 30K. With smaller macro files, you won't have to worry about this.

(with your actual selections on it, of course). Press Esc from this menu, and you're back at Q&A's Main menu.

All your other swappable macro files (modules) must be similarly-structured. They all need the same first three macros, followed by the custom *Main* menu macro, then the module-specific macros. (When recording your macros, be sure the macro file you intend to save them to is in memory.)

The Lookup Table

To make the MODULES master menu system easily configurable for different macro file names and locations, I place all the parameters in MODULES' Lookup Table rather than hard-coding them in the programming. Figure 5 shows the Lookup Table. The idea is for you, the developer, to have an easy way to customize all your module-switching parameters from a single screen.

Figure 4. The Main menu for the Customer Maintenance module. Each module has a similar custom Main menu

Figure 5. MODULES.DTF's Lookup Table contains the variable parameters for the programming.

The first two lines in the table tell the programming (as you'll see) where to find QA.COM and SWAP.BAT when switching modules (macro files).

The next group of entries, starting in the Key column, are the field names that contain the descriptive module names, followed by the module descriptions themselves (in column 1), the corresponding macro file paths (in column 2) and the autostart macro switches (column 3).

The final six Key column entries are the filenames of the six macro files with their corresponding MODULES fieldnames in column 1. This group is strictly for display. When you pull up the master menu, the macro file that's currently in memory shows on the screen. Not essential, but a nice touch I thought, particularly during the development phase.

You can, of course, have fewer or more than six modules. You simply construct your Lookup Table accordingly and modify one programming statement (as you'll see). Q&A will then know how to handle the module-switching requirements of your Q&A system.

Programming

Considering what the module-switcher does, it doesn't require all that much programming. I'll show you the programming (moving left to right down the form), then follow it with an explanation:

Start:
```
<
Rem "Fill selection fields from Lookup Table";
Item1 = @Lu("Item1", 1);
Item2 = @Lu("Item2", 1);
Item3 = @Lu("Item3", 1);
Item4 = @Lu("Item4", 1);
Item5 = @Lu("Item5", 1);
Item6 = @Lu("Item6", 1);

Rem "Determine macro file currently in memory
    by isolating last line in SWAP.BAT";
Rem "Insert SWAP.BAT";

Start = @Insert(@Lu("Swap", 1));

Rem "Remove carriage returns";
Start = @Replace(Start, "
", "");

Rem "Replace last \ with |";
Start = @Repllas(Start, "\", "|");

Rem "Leave only filename & switch";
Start = @Del(Start, 1, @Instr(Start, "|"));

Rem "Display filename & switch";
If @Left(Start, 2) = "QA" Then
Start = "qamacro.asc";
MacroFile = Start;

Rem "Strip autostart macro switch";
If @Instr(Start, " ") Then
Start = @Left(Start, @Len(Start) -4);

Rem "Get fieldname from table";
Start = @Lu(Start, 1);

Rem "Highlight pertinent field";
@Color(@(Start), 15, 4);

Clear(Start, Check, Shell);

@Msg("Click on module, or enter module number")
```

Check:
```
<
If Start >= 1 and Start <= 6 Then
Goto @("Item" + @Str(Start))
Else Goto Start
```

Item1:
```
< Shell = "Item1"; Goto Shell
```

The Item2 through Item6 fields are programmed identically except for the Item*n* field reference.

Shell:
```
<
Rem "Builds the new line for SWAP.BAT, writes the
line to SWAP.BAT, then runs the macro that exits
Q&A. Windows executes the new SWAP.BAT line, which
restarts Q&A and autoloads the new macro file";

If Shell <> "" Then {Shell =
@Shell("echo " + @Lu("Q&A", 1) + " "
+ @Lu(Shell, 2) + " " + @Lu(Shell, 3)
+ " >> " + @Lu("Swap", 1));
@Macro("Swap Macro File") }
Else CHome
```

The **MacroFile** and **Help** fields require no programming.

How the programming works

The **Start** field is the first field on the form. The first thing its program does is fill the module selection fields (Item1 through Item6) with the corresponding descriptions from the Lookup Table. Revise this block of code to accommodate the actual number of modules you have.

The program next inserts SWAP.BAT into the field (consulting the Lookup Table for SWAP.BAT's location). It isolates the current macro filename and autostart macro switch and displays this in the **MacroFile** field for reference. Remember, SWAP.BAT can contain many lines if you've switched modules a lot during your the Q&A session. The active or current macro filename and switch will always be at the end of SWAP.BAT's last line.

Next, the program consults the Lookup Table to determine the Item field (Item1 through Item6) that corresponds to the active macro filename, and highlights that field by reversing its colors. This way, with the master menu on screen, you can see at a glance which module is active and its corresponding macro filename.

The final part of the Start field's program supplies user instructions (a prompt) via an @Msg message.

The programming gives you two ways to switch modules. You can simply click on the name of the module you want to use, or you can enter the module number. (See Figure 2.)

If you enter a number (in this case, 1 through 6), the program in the **Check** field adds that digit to the "Item" string (*Item3*, for example) then uses indirection to pass control to the matching Item3 field.

Now we're into the programs for the **Item1** through **Item6** fields. They're all identical except for the fieldname reference passed to the Shell field.

**EDITED BY GORDON MEIGS
AND WILLIAM HALPERN**

Confusing "Second Copy" Message

We run Q&A 5.0 on a Novel 4.11 network. Some of our PC's show "2nd copy" at the bottom of the opening screen while others do not. Why is this?

Graham Bolon, South Africa

Your people are using Q&A 5.0's External Programs feature to run a second copy of Q&A. That's what the "Copy 2" in the lower right-hand corner of the screen denotes. (When you exit the second copy, Q&A returns you to where you left off in the first copy.)

External Programs lets you run another copy of Q&A from anywhere in Q&A (even from a database record or document) simply by pressing Alt-F9. Since the Main menus in the two copies are identical, the "Copy 2" message lets you know "where you are."

To check for a second copy of Q&A on the External Programs menu, from the Main menu go to Utilities / External Programs. If you see a command line with *QA.COM* on it, that's what starts the second copy when the user presses Alt-F9 and selects it. If you don't want to use this feature, simply remove the entry and press F10 to resave the External Programs screen.

After responding to Graham via email, he added this:
I'm aware of why some machines show the "Copy 2" message and how you set up the second copy. The question is why some of the machines *do not* show the message. The machines that show it always show it [when in the second copy], while other machines never show it. Unfortunately, my PC is one that never shows the message.

This is because you've customized your Main menu by adding additional selections to it. As soon as you do this, the "Copy 2" message won't appear on that PC. The way we work around this is to start the second copy of Q&A with a *different* macro file. Our External Programs command line that runs a second copy of Q&A looks like this:

```
qa.com -ald:\qa5\qamac2.asc
```

This second macro file includes a custom *replacement* Q&A Main menu that readily identifies the copy as the Copy 2. (See Figure 1.) The custom menu structure looks like this at the Application Menu Options screen:

```
Menu name............: Q&A Main Menu
Display..............: [Full screen]  Overlay
Status...............: [Active]    Inactive
Menu returns.….......:  Yes    [No]
On Escape, show menu.:
Menu Title...........: Second Copy

Item 1: X - Close 2nd Copy    Macro Name: x
Item 2: F - Search or Add     Macro Name: f
```

Figure 1. With a custom Main menu for the second copy of Q&A, you'll always know you're in Copy 2 when you display its Main menu.

Another Way to Run a Second Copy of Q&A

You can also use Q&A 5.0's @Shell command in a database program to put yourself in a second copy of Q&A. This way, you can go out and check some data or perform an operation without losing your place in the record you're working on. What's more, there's a QACOUNT utility you can call via @Shell to tell you which copy you're in. This lets you conditionally (*If Copy2 Then...*) run an @Macro to return you to where you left off in the first copy. See the April 1997 issue, page 11, and download QACOUNT.EXE from the Freebies page at www.quickanswer.com.

Stumped?

Send your Q&A questions to **@Help, The Quick Answer, Marble Publications, Inc., 1927A Harbor Blvd, Costa Mesa, CA 92627** or email to **mailbox@quickanswer.com**. Include your name, address, phone, and your Q&A version number (and whether DOS or Windows) and a detailed description of the problem. We'll publish those questions we feel are of general reader interest; individual responses aren't possible.

Using Lesspace with Large Drive Partitions

I recently downloaded the Lesspace utility and can't figure out how to make it work. The instructions in the readme file are Greek to me. Can you help me out? Thanks.

Niels Ewing via the Internet

DOS programs such as Q&A won't operate correctly from a hard drive partition larger than 1.8G. (This includes the program files, as well as database, document, and temporary files.) Lesspace is a DOS-based TSR (terminate and stay resident) program you can use instead of creating a new, smaller hard drive partition just for Q&A on a much larger drive. Lesspace works with Q&A for DOS 4.0 or 5.0. In effect, it "fools" Q&A into thinking that it's operating in a 500M partition in which there's 500M of free disk space. To use Lesspace, you must start it *just before* you start Q&A, and quit it when you quit Q&A. You can have Windows do this automatically by configuring your Q&A desktop shortcut to run a batch file that starts Lesspace then Q&A. Using any word processor (Q&A Write is fine), type the following, substituting your own paths to Q&A and Lesspace.com:

```
C:
C:\Utils\lesspace.com
CD\QA4
QA.COM
C:\Utils\lesspace.com /u
```

Save this file as a text or ASCII file called QALS.BAT in your Q&A program files directory.

Now, go to your desktop and right-click on your Q&A icon. Select Properties and click on the Program tab. Make sure that the command line reads: C:\QA4\QALS.BAT (substitute the actual path to your Q&A program files). From now on, when you double-click your Q&A shortcut, Windows will start Lesspace then Q&A, and will remove Lesspace from memory when you quit Q&A.

Installing Q&A On a Large Hard Drive

Q&A 4.0 for DOS, during installation, is extremely sensitive to hard drive partition size, and Lesspace can't help you with this particular problem. The only way to install Q&A 4.0 in a partition greater than about 1.8G—where you can then use Lesspace to run it—is to copy all the files to the new PC's drive from a previously installed copy on the other PC using direct cables or a network connection. If you do this, make sure your Utilities / Set Global Options screen, along with your macros and programmed XLookups in the new setup, reflect any different paths to your files. You can avoid this by setting up your Q&A directory structure on the new drive exactly the same way as on the old PC. For example, if the path to your databases on the old PC is c:\qa4\data\, make sure you copy your databases to a directory named c:\qa4\data\ on the new PC.

Print Preview Problem in Windows 95/98

My Q&A Print Preview works fine and returns to full screen when I run Q&A strictly in DOS. But it doesn't work so well with Q&A running in Windows 95. I wind up at a tiny screen when returning from Print Preview to the document. Is there a solution other than *not* to run Q&A 5.0 under Windows 95 ?

Patrick McCarthy via the Internet

Q&A will return from Print Preview to a minimized window in Windows 95/98 *only* if you invoked the Print Preview with Q&A running in a Window. When this happens, simply click on the Maximize icon in the upper right hand corner of the Q&A window and the window will be restored to its original size. (If you run Print Preview in *full-screen* mode in Windows 95/98, it will return to full-screen, just like it did under DOS.) The additional keystroke seems like a small price to pay for picking up the extra Widows features available by running Q&A in a window (copy/paste to/from the Windows Clipboard, to name just one.)

"Disk Drive Door Open" Error Message

I keep getting a *"Disk drive door open"* message when I try to open Q&A write documents. What's the problem? There's nothing in the floppy drive!

Several Q&A users via the Internet

We've seen this same error message at some of our clients after designing a document for them offsite then installing the document onsite. It seems to occur only with documents that were saved with assigned fonts.

Apparently, Q&A is looking in a particular location for the fonts (font file) that were assigned to that document the last time it was saved. If the document has been moved or copied to a new computer, or the printer has been changed, Q&A might not be able to find the correct fonts for the document, and the error message (albeit a rather strange one) appears.

The solution we've found is to press the Esc key when the message appears. This opens the document. Then, we press Ctrl-F9 to display the Font Assignments screen, and make sure that each of the font lines is filled out correctly. We then resave the font assignments by pressing F10, then resave the document. This way, the problem doesn't recur with that document.

So far, we haven't been able to isolate the cause of the "disk drive door open" bit. (Is Q&A expecting a diskette with the correct font file for the document?) If and when we do, we'll let you know what we've found via this column.

Bill Halpern and Gordon Meigs own Professional Computer Technology Associates in Newtown, Pennsylvania, 215-598-8440, www.pcta-usa.com, email pcta@comcat.com.

Subdividing a Mature Database—Part 2

JOHN T. DOW

Breaking up is hard to do. But your database's health might require it

LAST month, in part 1 of this case study, I described my encounter with a client's (law firm) database which, over the years, had been stuffed with so many fields and had swollen to so many records that it was about to "burst at the seams." I covered the inherent flaws in the database's design and the reasons for breaking up this single huge database into several smaller, more efficient and manageable databases. This month, I'll show you how we did just that.

The final design

What came out of the redesign process was a compromise that involved some of what the database purists would want but is still very much in the Q&A camp.

First, there is a new database called C (for CLIENT). Each client has just one record in this database, even if he has multiple cases. This database was populated with information from the original database by eliminating duplicate client records, which I'll describe later.

Second, the records in the original database were split out into multiple databases based on the type of case. A few examples of types of cases are: Appeals, Divorce, Social Security. Appeals cases records went to the appeals database (A), divorce cases to the divorce database (D), and so forth.

All the new databases (A, D, S, and so on) have common header information on the first page, including client information in read-only fields brought over from the C database each time the record is accessed.

The design for A and the other new databases includes two identifying numbers. First, each stores the client number (123456, for example). This number uniquely identifies the client, but not the case, so it does not uniquely identify the record in the database either.

There is also a case number field so that each case (and record) has a unique identifier. The case number is created from the client number by appending information specific to the database. For example, in the Appeals database (A), the case number format is "123456A 19990525." It includes the client number, the database identifier (A), and the date, making it unique within the database and the entire system (excluding payments, which I'll cover later). When various manipulations are done across databases, it's convenient to have the

database identified like this as part of the key value.

Some cases involve several persons. Those records were split out not only into a new database (for example, "B" for Beneficiaries) but also each person's information was split out into his or her own record. For this type of case database, the case number looks like this: "123456B MARY", meaning that Mary is a beneficiary for client 123456. All information pertinent to Mary will be in her own record in the B database.

Thus, one of the lists (group of line-item-like fields) in the original database resulted in one record per list entry in the new database. This solves the problem of searching for people in the list. If you want information for a Mary Jones, you can go to the B database and easily search the beneficiary first and last name fields for "Mary" and "Jones." (Note that there is also a client first name field and a client last name field in the same record.)

Other than the people list, the other lists in the original database remained lists in the split out databases. How can this be, given the discussion in last month's Part 1 detailing the various problems with lists in a database? First, the list that's searched frequently is the people list, and we did split that out into separate records. The other lists are only viewed (such as comments) or are not searched but are used for reports (for example, critical actions must be reported regularly to supervisors). Our solution to the reporting problem was to compromise and do some batch processing at night, which I'll describe later. We realized that we couldn't compromise data entry and viewing ease for the sake of reporting.

The payment information was in a list in the original database. It's one on which we compromised. In the new scheme, it is still in a list, but in its own database that has payment data and virtually nothing else except information to identify the client and the case.

The business rule is to treat each case separately when it comes to payments. In the original database, each case record had its own payment list. This remains true, but even though the case records have been separated into multiple databases (A, B, D, S, and so on), all payments are in one database (R for receipts). If there's an appeal, then there will be a payment record for that case. The client number will of course be "123456" and the case number will be "123456A 19990525." At the same time, there could be a record of payments from Mary, having a case number "123456B MARY." That is, the case number field in the payments database exactly matches the case

number in the database containing the case information, thus tying them together.

To summarize, there is a database that identifies clients and provides client numbers. Other databases were created to store information appropriate for specific types of cases. In some of these databases, there may be multiple cases per client, so each has a way of generating case numbers (using the client number as a base) so each record has a unique identifier. Each record also has the client number in its own field. Finally, there is only one payment database for all types of cases. Its records are linked to the appropriate case via the case number.

Normalization still needed

Our design is a compromise. Much of the information is still in lists in various databases as in the original file. For instance, doing a report of payments in the last week remains difficult because payments are in lists—they haven't been normalized. Doing a report of appointments scheduled for tomorrow is likewise difficult.

Refer again to Tom Marcellus' normalization article in the November 1998 issue. We decided to use a technique similar to the one he describes. Data is extracted from the data entry databases and loaded into a simple database used solely for reporting.

The automated procedure is run by a scheduler every night. A batch file first cleans up from the previous day's work. Then it copies empty databases and other files to the local TEMP directory. The data entry databases from which data is to be extracted are then copied to the TEMP directory also. In Tom's article, he describes how a Q&A report can be used to extract and normalize data. Since we were already using my DTFDATA program for various other purposes and since it doesn't have any limitations on the size of the list that needs to be extracted, we use it instead of reports to extract and normalize the data. After DTFDATA has extracted all the data to ASCII files, Q&A is started with a macro that imports the data into databases for reporting. Q&A then exits, the updated report databases are copied to the server, and everything is ready for the next day and whatever reports need to be run.

Navigation between multiple databases

When you have just one database, navigating around the data for a case is easy. You press the PageUp and PageDown keys a few times. When the database has been split into multiple databases, it isn't so easy. You have to go from one database to the other. Of course, this should be easy for the user to do. Getting to the correct record in the new database should require little or no effort.

All databases in the new design have the client number in the upper left corner of each database's first page. That's the secret for going from one database to another while staying with the same client. Here's a macro that will go to the A database from any of the others:

```
<begdef><altA><name>"A"<vidon><home><home><home><f11>
<capsf10>sA<enter><f12><f10><enter><enddef>
```

The Alt-A the macro "presses" <home> three times to move to the top field on the first page, where the client number field is. (Note that it can't be a read-only field for the cursor to enter it. You may wonder about this, since the client number is a field that should never be changed. The answer is that there is programming that resets the field should the user change it mistakenly.) On reaching the field, the macro presses F11 to copy the client number to the clipboard. Then it exits from this record in this database (<capsf10>) and goes to database A, pasting the client number into the client number field into that database's Retrieve Spec (<f12>) and finding the record(s) that match (<f10>).

If the user is looking at a record for client 123456 in database D, pressing Alt-A will jump to the data for that client in database A. That's pretty easy to do.

If database A doesn't have a record for client 123456, which may very well happen, then of course the search will fail. The user can get back on track by telling Q&A to search another database and then can press F12, F10 in the Retrieve Spec to get right back to that client's record(s) in the newly selected database.

The Command Center: finding, navigating, adding

Remember that in the original database it was easy to search for a client but difficult (or slow) to search on other people in the database because they were in a list (a series of name fields in the same database). Remember also that a search was really for a case, not a client, since the original database had no way of identifying more than one case belonging to the same client.

An interesting question is whether we're any better off right now. We do have the client database (C), which will let us search for a client and get a client number. However, other than identifying the client, the client database doesn't tell us much. All of the really useful information is in other databases. The problem is, we don't know which. We can't expect a user to go searching through a dozen databases, even if the client number is known. Some method of showing the user which databases actually have data for the client is needed.

So the idea of a "Command Center" was born. This was first implemented within the C (client) database. However, that led to some confusion, so a new database was created just to serve the purpose. It's similar to one-record databases that people use for menus and such. It stores no data but merely serves as a user interface for accessing other databases.

The menu database provides several very important services. First, you can do a name search for the client (searching the C database using first and last names). The name search uses the DTFDATA utility and a new utility, DTFLIST. With these programs, the search results in a nearly full-screen popup listbox that scrolls vertically using the arrows keys, PageUp and PageDown keys, and Home and End keys. The currently selected line is highlighted and selection is made by pressing Enter. The

following information is displayed to allow selection of the client: last name, first name, middle name, date of birth, address, city, phone number, and so forth. Over 200 characters are displayed on each line, requiring horizontal scroll to see everything if needed.

Second, you can do a similar name search for beneficiaries (searching the B database using first and last names). The goal here is to identify the client. Therefore the result of the search is a client number.

Third, if either search fails to find the client, you can add a new client via a macro that goes to the C database to add a new record.

Fourth, and the most important, is that after a client is selected using one of the two searches, you can then ask for the "command center display"—another popup listbox. In it are lines showing all the client's cases in all the databases. Simply by selecting a line in this listbox you are taken to that case record in the appropriate database. There are lines at the end of the listbox for making new entries. Here's a typical command center listbox this:

```
Beneficiary    111000B MARY     :Mary Francis
Receipts       111000B MARY     :Balance due=760.00
Add A - Appeals
Add B - Beneficiary
Add D - Divorce
```

You can see from this that this client has one case, with Mary Francis as a beneficiary. The client currently owes $760.00 for this case also. If you select the first line, you will be taken to the B database to the "111000B MARY" record. If you select the second line, you will be taken to the R database (payments) to the "111000B MARY" record (payments for the above case). You can also select "Add A - Appeals" to create a new record in the A database, thereby creating a new appeals case. In this example I have not shown the full set of databases in which new records can be added by this mechanism.

This command center is crucial for the whole system of databases to work. It's a one-stop-shop where the user can identify a client and see what cases already exist. It's also how a user creates a new client or new cases.

This command center display is generated using my DTFDATA and DTFLIST programs. They are faster than Q&A and provide a nicer user interface. However, a series of XLookupr's into the various databases can extract the needed information; so with some programming an "organic" Q&A solution is possible.

What's important is not the technical details of how the command center was built, but that we built it at all. It's the glue that holds the system together. Without it, finding information distributed among the various databases would be a time-consuming, frustrating effort. The current system, with the command center, has won the hearty approval of the people who use it every day.

Converting the data

After designing and testing the new system of databases, we planned a conversion process and launched into it

after the close of business one evening.

First, we added a read-only field to the original database and did a mass update, assigning the field a record number using @Number. As data was extracted from this original database and moved into other databases, we made sure that this record number field was always included in the set of fields going to the new database. This enabled us to always track a record in any database back to the original. This has been very handy at times when analyzing questions and problems, and it can of course be used later with XLookups to bring over additional information that should have been included in a new database when it was first formed.

Then we made copies of this master database on multiple workstations so that we could simultaneously create new databases on different machines. Doing the work on local drives is also much faster than manipulating databases on a server.

The first step was to make subsets of the records. Redesigning the fields for specific databases to correspond to the specific types of cases was not part of the initial process. We made subsets of records by a variety of means. In some cases, we used Q&A's Copy feature with a Retrieve Spec. In others, we deleted all but the records we wanted. We also exported and then imported. Sometimes we used Q&A to export with a Sort Spec. Other times we used the data exported by DTFDOCTR.

Why so many ways of creating subsets? First, we used different techniques depending on the size of the subset being created. Also remember that for some new databases, one record in the old database would have data on multiple people, and so it would generate multiple records in the new database. But second, we had data problems that would prevent the Q&A Copy command from working. It seemed that in all of the 30,000 records there were a handful with mysterious problems that prevented the records from being copied.

At the same time that we were making subsets of the data, we were creating the client database. Remember that the client database has only one record per client, whereas the initial database had multiple records if the same client had multiple cases. To create the client database, we had to identify the same client in multiple records. This is not a trivial task when there are thousands upon thousands of records. You can't use your own eyes, it would take too long. And if the data isn't perfectly clean, doing it by computer will yield imperfect results. What we did was create a match based on 10 characters of the last name and six characters of the first name. This made a 16-character key, with duplicates of course. A mass update was used to lookup into the file using this key value, and the record number returned was used for the client number. When there are duplicate key values and you use XLookup, Q&A returns one of the records with that key value. Q&A is arbitrary as to the one it returns. And it's *consistently* arbitrary (until you change something anyway).

All records in the newly formed client database in

which the client number wasn't the same as the record number (see above) were considered to be duplicates and were moved into a spare holding copy of the client database called C_DUPS. With time, we learned that matching just on the names resulted in too many people with the same names (the "John Smith" sort of thing) getting the same client number. So another set of analysis reports followed by mass updates separated out these people based on date of birth. Other fields such as Social Security number could be used, but birth date seemed best. Since not all information had been gathered for all case records, anything along these lines was imperfect. For instance, if a record has no birth date, do you assume it matches or doesn't? The bottom line is that there are still some lingering client identification problems and there needs to be a process to clean them up as they are discovered. Having the C_DUPS database with the original record number is very helpful for sorting out who is who so that manual corrections can be made.

The point that I want you to be left with is that manipulating data like this will take time. You'll have to do things more than once. You'll have to do mass updates to correct problems. The original conversion will take hours and hours, and then there will be an ongoing maintenance process to maintain clean data and to resolve issues that are uncovered days or weeks later.

Maintaining database integrity

A very important issue to stay on top of is the linkages between the various databases. When everything is in one record in one database, no problemo. But when records in various databases are linked merely by a client number, you have to be very careful that that number isn't changed inadvertently in one of the records. If that happened, the record would be orphaned or, perhaps worse, linked to the wrong client.

The term *referential integrity* refers to making sure that every key value matches up properly from database to database. Losing referential integrity is a definite risk. Major big-ticket relational databases provide built-in solutions to this problem. They also hire big-ticket database administrators to manage such things. Our scale isn't quite the same. Nevertheless, we should take steps to ensure that there isn't a breakdown in integrity. I'd like to mention two types of things that can be done.

The first thing to do is to preserve identifying information in each record. I mentioned that there is common information about the client in read-only fields in each of the databases and that this information is automatically updated on-record-entry. The information that actually identifies the client is not updated. This includes the name, date of birth, social security number, and so forth. What *is* updated is more transient data such as the address and telephone number. Should a record become misplaced, you'd at least have a name and date of birth to identify who it really belongs to.

Second, reports can be designed to detect problems.

For example, the client database can be checked for duplicate client numbers. In theory this should never happen, but it should be checked periodically. You can write a Q&A report with a derived column spec to compare the client number from the previous record to the current record. Within the other databases in our design, the case number should always consist of the client number followed by other information, so a report that tests to make sure this is true can easily be done. You'd need this type of Retrieve Spec:

```
{ @Left(CaseNo,6) <> ClientNo }
```

For all records in a database that have the same client number, a test should be made to check that they all have the same name and date of birth. A Q&A report could do this check with a set of derived columns for all the fields to be compared.

Reports comparing records across databases should also be performed. The best way to do this is to extract a subset of fields from each database, together with a field positively identifying which database the data was extracted from. Then all of this extracted data should be merged into one database just for the purpose of analysis and reporting. The types of reports described above could also be performed on this composite database, thereby ensuring consistency across databases.

Sharing code in similar databases

One of the aspects of Q&A that's a blessing and a curse is the fact that the programming is in the same file with the data. It's a curse when you want the same programming to apply to data in more than one database. It has to be put into each database. In our scheme, that's about a dozen databases having very similar programming.

I made my life easier by devising a technique to actually reuse programming code. I created a field in each database called "ThisDB." I set the initial value for the field to be the database's file name. Example: C.DTF for the client file. (A mass update was also needed to initialize existing records to have the same value, and should a record be moved from one database to another for some reason, it is necessary to mass update this field pronto. As part of a regular database integrity check, any anomalies in this field should be reported.)

Now I write code that can test for those few exceptions to the common code. I simply write:

```
If ThisDB = "C.DTF" Then...
```

I can then develop code in one database and cut and paste it into others without change.

Updating common information rapidly

It was important to the users that information such as the client's address be automatically updated as they entered records in the various databases. This helps ease the pain caused by splitting information into multiple databases.

The information all comes from the client database. There are quite a few fields to be updated. Doing separate XLookups for each field takes a second or two on a lightly loaded network. (I have done tests and learned paradoxically that having XLookups return multiple fields at once is actually slower than using it once for each field.) In order to make this go faster, I changed C so that it has a composite field whose program is set to execute on-record-exit. This field lists the field names and the values for the fields to be updated in the other databases. Actually, the field names are the names in the destination databases. (Naturally we standardized field names across the databases.) A small piece of programming in each of the databases brings over the composite field and splits it out. For example, the composite data string might have

`First=Jimmy;Last=Jones;DOB=1962/04/20` (and so on)

The programming parses out First as the field name and Jimmy as the value, then uses indirection (@(fieldname)) to place Jimmy in the field named First.

Conclusion

We are probably pushing the limits of what you want to do with Q&A in terms of data complexity when we break it out into different databases like this. The reason, simply, is that Q&A wasn't designed for it. It doesn't have much in the way of tools to help with the design, with the transformation of the data, with ensuring the integrity of the data, and with accessing fields in other databases. Nevertheless, it is possible to pull it off and probably for much less effort than redoing the entire application in another database manager. I hope the experience I've recounted here will help you decide if you should do this and perhaps spare you some discomfort in doing it.

John Dow is a Q&A developer and creator of a host of useful utilities for Q&A. jtd@pgh.net, www.johntdow.com.

Improve Your Screen When Running Q&A DOS in a Window

When you run Q&A DOS in a window, you have a choice of screen fonts and window sizes. But they all look pretty crappy in my view—*except one*. The one that gives me an ideally-sized window with bold, easy-to-read text is the 8514OEM.FON 10 x 20 font. It looks good on my desktop PC at 1024x768 or 800x600 display resolution, and on my notebook at 800x600 resolution. It's the font/window size we use for the Q&A screen shots in *The Quick Answer*. But as I recently learned from two subscribers, some Windows PCs don't come with this attractive "DOS-box" font.

To see if yours does, put Q&A DOS in a window (press Alt-Enter from full-screen mode). If no toolbar appears at the top of the window, press Alt-Spacebar, click on *Toolbar*, then pull down the *Font* list. If you don't see a 10 x 20 font on the list, or if there's a 10 x 20 font but it's very light and hard-to read, then you probably don't have the 8514OEM.FON font. (To make sure, click on the large "A" icon on the toolbar to display the Font Properties tab, set *Available Types* to *Both types*, then try again.)

You can download the 8514OEM font from the Freebies page at www.quickanswer.com. Copy it to your WINDOWS\FONTS folder, then reboot your PC. Now, when you put Q&A in a window, you should have a 10 X 20 font option that gives you a nicely-sized window with very readable Q&A screens. DOS program screen fonts are "sticky," meaning that the one you choose will remain active until you change it.

Running Q&A in a window offers numerous advantages. For example, you can copy and paste text (into and out of Q&A) using the Windows Clipboard. (Be sure "Fast Pasting" in your Q&A shortcut's Miscellaneous properties is turned off.) Also, your Windows 95/98 task bar is exposed when you run Q&A in a window, so you can easily start and switch to other programs without having to exit or even minimize Q&A. For tips and articles on running Q&A for DOS in a window, search *TJ's Quick Answer Index Database* at www.quickanswer.com.

TOM MARCELLUS
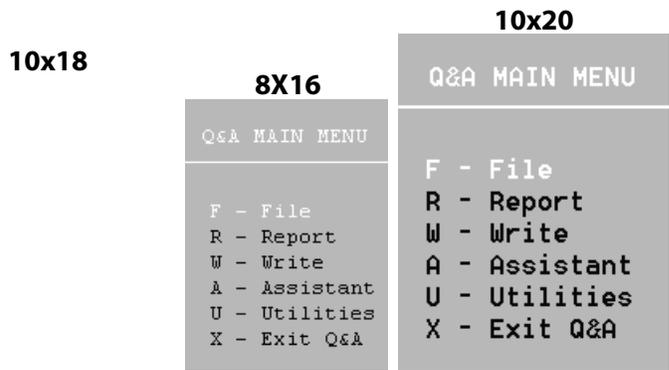
Figure 1. The 10x18 screen font **(Left)** looks blocky and squished. The 8x16 font **(Center)** is thin, light, and hard to read. The 8514OEM 10x20 font **(Right)** is well-proportioned, easy on the eyes, and produces a window size that looks good at 800x600 and 768x1024 display resolutions on both desktop and notebook PCs.

Replacing 'That' with 'This'

MARK MYERS

Need to replace part of a field value with something else—in dozens or hundreds of records?

IF you import records into a database, your data can get "messy" over time. Likewise, if you purchase data files for use in mailings, the information they contain might not be formatted exactly as you'd like it to be.

Q&A has useful tools for "scrubbing" your data. Among these, the replace functions (@Replace, @Replfir, and @Repllas) allow you to apply precision formatting changes to the contents of fields.

The general @Replace function is a good example of why these functions are so useful. Suppose you have a contacts database with a Keyword field containing the kinds of pets owned. The field might be filled from a picklist whose choices are *Dog, Cat* and *Bird*. As records can contain any combination of these three Keywords in any order, it would be a daunting challenge to change every instance of *Dog* to *Canine* without @Replace. With it, though, you could do the job quickly by using the following command in a Mass Update Spec:

```
@Replace(Pets, "Dog", "Canine")
```

A word of caution: Using @Replace imprecisely can have disastrous effects on your data. You should always backup your database before performing a Mass Update on it.

Suppose you wanted to change every occurrence of the word "On" to "Off" in a free-form text field. Applying *@Replace(Text Notes, "On", "Off")* to a record where the field contains "Honor Original Offer" would change it to "Hoffer Original Offer." Trying to restore the data by reversing the parameters wouldn't work in this case; the string would become "Honor Original Oner." The right way to update the field is to apply the statement with spaces around the words *On* and *Off*:

```
Text Notes = @Replace(Text Notes, " On ", " Off ")
```

This will ensure that you replace the word *On* (or *on*) with *Off* rather than replacing every instance of *o* followed by an *n* with the character string *off.*

Of course, this won't replace the word *on* when it occurs at the very beginning or end of a field. This is where the @Replfir (replace first) and @Repllas (replace last) functions come into play. The actual programming to update the entire field would be as follows:

```
Text Notes = @Replace(Text Notes, " On ", " Off ");
If @Left(Text Notes, 3) = "On " then
Text Notes = @Replfir(Text Notes, "On", "Off");
If @Right(Text Notes, 3) = " On" then
Text Notes = @Repllas(Text Notes, "On", Off");
```

The above case is a general way to describe some basic techniques for using these functions. However, my experience has been that these functions are most useful for standardizing addresses. As I mentioned earlier, lists obtained from outside sources often need to be scrubbed. And in-house data entry often leads to inconsistent data.

Names, street addresses, cities, states and ZIP codes all present challenges for anyone attempting to standardize a mailing list. For our purposes, we'll talk about some of the problems presented by working with names and how the replace functions can help you standardize your data.

A name might be entered into a database field in a number of different ways. Here are three examples:

```
John Q. Public Jr.
Mr John Q Public, Jr
Mr. John Q. Public Junior
```

To handle these differences, you first need to decide what form you want your data to end up in. You might want a period after every prefix and middle initial. Additionally, you might want a comma separating the suffix from the rest of the name and to use punctuated abbreviations for suffixes. While the programming for this clean-up can be initially laborious, it isn't hard to come up with general forms for each criteria you've established.

First, for handling the prefixes, you'd manipulate each prefix exactly as I've outlined below for *Mr.*

```
If @Left(Name, 3) = "Mr " then
Name = @Replfir(Name,"Mr ", "Mr. ")
```

In a similar manner, you can update every middle initial by following this form:

```
Name = @Replace(Name, " Q ", " Q. ")
```

Meeting the criteria for suffix formatting requires a three-step process. First, you need to handle cases where *Junior* and *Senior* are spelled out:

```
If @Right(Name, 7) = " Junior" then Name = @Repllas(Name,
"Junior","Jr.");
If @Right(Name, 7) = " Senior" then Name = @Repllas(Name,
"Senior","Sr.")
```

Little Time-Savers Add Up

ALEC MULVEY

Here are a few tips that can make data entry easier. The theme is saving keystrokes—that is, having Q&A automatically enter what needs to be entered, or to "second-guess" what users will want to enter.

Auto-Incrementing Invoice Numbers

By using @Number in a program or in the Initial Values Spec, Q&A can automatically generate an incremented number, such as an invoice or purchase order number. Sometimes, though, such a number is generated elsewhere and needs to be entered manually. Even so, it is still likely to be one higher than the last one. In this case, a program along these lines can save keystrokes by adding "1" to the highest invoice number already in the database:

```
< #1020: If InvNo = "" then InvNo = @XLR(@Fn, 99999,
"InvNo", "InvNo") + 1
```

If InvNo is a text field, you can add a leading zero if the number could increase to another digit (for example, from 0999 to 1000). The following more complex program generates text-field invoice numbers rather better since it handles a "roll-over" up to five characters:

```
< #1020: IF @Add And InvNo = "" Then InvNo =
@Rt("00000" + @Str(@Tn(@Num(@XLR(@Fn, 99999, "InvNo",
"InvNo"))) +1 ),5)
```

Both these programs allow you to change the invoice number if it's out of sequence. This might not be suitable for a network application if more than one user might be adding records to the same database at the same time. In this case, a conventional @Number generated after a number of key fields have been filled might be more appropriate. (Q&A automatically modifies the behavior of @Number in a shared database to avoid the risk of a number being duplicated. See the April 1998 issue, page 7, and the March 1999 issue, page 5.)

Q&A provides the current year by default

Many Q&A users, I've found, aren't aware that they don't need to enter the year portion of a date if the year is the current year. To allow omission of the year, even when entering December transactions in January, the following program can be useful:

```
> If @Add And Date > @Date + 30 Then Date =
@Replace(Date, @Yr(Date), @Yr(Date) -1 )
```

This subtracts a year from the date when the resulting date is more than 30 days in the future, such as when "15 Dec", or "15/12" is entered in the following January.

Duplicating data from the previous record

When adding records to a database, a field might routinely require the same value as the previous record. For a start, you can educate users about the F5 (Ditto) key that copies the value from the same field in the record last added. But how about doing it for them as an "initial value"? This program achieves that for two fields:

```
InvoiceType:
< If @Add And InvoiceType = ""
And Date = "" Then @Ditto(InvoiceType, Date)
```

Of course, this might prove to be more of a nuisance than a benefit if the previous value often needs to be changed. You need to consider each field individually. (Next month I'll show how to get the best of both worlds.)

Skipping over unnecessary fields

In some databases, fields within a series can be mutually exclusive. For example, if you're assigning a sale to one of three categories and the first one is chosen, you can use a navigation program to skip over the other two:

```
Cat1:    Cat2:    Cat3:    Other Field:

Cat1:  > If Cat1 <> "" Then Goto Other Field
Cat2:  > If Cat2 <> "" Then Goto Other Field
```

Splitting an amount between categories

Suppose a sale amount needs to be apportioned among several categories, and there are fields for categories such as Software, Hardware, Consumables, Training, and Other. The idea here is for Q&A to keep a running balance (in the "Other" field) of the portion of the amount that remains unassigned. The trick is to make it easy to allocate the balance to any one field—what users generally do. Here's how it works. The amount to be assigned is in an earlier field named "Net Amount." The first category field is programmed like this:

```
Software:
> If Other = 0 Then Goto DatePaid
  Else If Software = 1 Then { Software = Other;
  Goto DatePaid } Else Goto Hardware
```

This is repeated in each of the other fields, changing the names of the current field and the next field accordingly, to Hardware, Consumables, and Training.
And finally:

```
Other:
#1190: Other = NetAmount - @Sum(Software, Hardware,
Consumables, Training)
```

In this case, Program Calc mode is set to "Auto."

Here's how it works. The "Other" field shows the Net Amount less all amounts assigned to the various categories. As amounts are entered into one or more categories, the amount still to be assigned is in "Other." To allocate the remaining balance to any of the category fields, the user simply enters "1" to tell Q&A to make that field equal to "Other" (the unallocated balance) and go to the next field, Date Paid. (If the user enters an amount that leaves a zero balance, the cursor goes to Date Paid anyway.) Any amount not allocated to a category stays in "Other." A refinement is to add a text field (Other Text) that explains what "Other" is. Programming can be added to force an explanation if "Other" is non-zero:

```
Field After:
< #1290: If Other > 0 and Other Text = "" Then
{ @Msg("You must state what 'Other' is");
Goto Other Text }
```

This should prevent unassigned amounts being unintentionally "left over" in Other. Here, the "1" isn't a literal entry but a trigger. Q&A detects it and executes your additional programming. This prevents entry of an actual value of "1" in this field, but it's assumed that there'll never be an invoice for one pound or dollar.

These techniques might save only a few seconds here and a few keystrokes there. But when users are spending hours a day entering such data, the overall time savings is very real. What's more, users find data entry less of a drudgery if they feel the program is working with them.

Alec Mulvey owns Keyword Training & Consultancy in Ascot, near London, England. Alec has been building Q&A applications and training clients for nine years, and is the current president of the (Inter)National Q&A User Group. Keyword Training is also the UK distributor for the International English edition of Q&A and John Dow's Utilities. Fax +44-1344-884-111, alec@keywordtraining.com, www.keywordtraining.com.

Replacing Data. . .continued from page 13

Next, you need to make sure that you've punctuated your suffixes correctly. As with the prefixes, a form of this statement should be written for every possible suffix:

```
If @Right(Name, 3) = " Jr" then Name = @Repllas(Name,
"Jr", "Jr.");
```

Finally, to assure that you've separated the suffix from the rest of the name with a comma, you can use the following compound conditional statement:

```
If @Right(Name, 4) = " Jr." And @Right(Name, 5)<> ", Jr."
Then Name = @Repllas(Name, " Jr.", ", Jr.");
```

Again, repeat this for every possible suffix.

You've seen that the general format for @Replace is:

```
@Replace(Field, "Existing String", "Replacement String")
```

Field can be a field name or field ID number. *String* can include text, numbers, punctuation, spaces—even hard carriage returns. To replace all occurrences of "W3" in a field with *nothing*—that is, delete "W3" and close up the space—you can do it this way:

```
@Replace(Field, "W3", "")
```

To replace "(space)W3(space)" with a space, use:

```
@Replace(Field, " W3 ", " ")
```

Replace two contiguous spaces with one:

```
@Replace(Field, "  ", " ")
```

Replace every hard carriage return with a space. (After typing the opening quote for the *Existing String* parameter, press Enter then type the closing quote):

```
@Replace(Field, "
", " ")
```

Replace two successive hard carriage returns (a blank line) with a one. (Press Enter *twice* after the first quote):

```
@Replace(Field, "

", "
")
```

The *Replacement String* can optionally be another field's value. For example, if you wanted to replace all occurrences of "book" in FieldA with whatever whole value is in the same record's FieldB, you'd write your program this way:

```
FieldA = @Replace(FieldA, "book", FieldB)
```

If FieldA originally contained *The book was enjoyable*, and FieldB contained *magazine*, FieldA would contain *The magazine was enjoyable* after the update.

If you're running updates like these on enough records, you will probably have some that fall through the cracks or get updated incorrectly. In truth, there really is no substitute for manual inspection. However, for activities such as mass mailings where a certain level of inaccuracy is acceptable, the replace functions are great tools for cleaning your data quickly, efficiently and with a high degree of accuracy. And of course, once you've gone through the effort of creating the Update Spec the first time, save it so you can run it periodically to keep your list clean and to update records coming in from new sources.

Mark Meyers is a programmer/analyst at Bluegreen Corporation in Boca Raton, Florida. He can be reached at lofi@hotmail.com.

Swap Macro Files...

The program in the **Shell** field could be called the module-switching "engine." By the time the cursor reaches Shell, Shell contains an Item$n$ value. The program finds, in the Lookup Table, the location of QA.COM and SWAP.BAT, then, using the Item$n$ value, it retrieves the name and location of the corresponding macro file and autostart macro to run. From this information it constructs a command line such as the following example that loads the Accounts Payable module:

```
D:\QA\QA.COM -alD:\QA\MACROS\PAYABLES.ASC -m6
```

The program then executes an @Shell ECHO command, using the ">>" redirection symbol to write this new line to the end of SWAP.BAT. This is that "special command" I mentioned earlier.

The program continues, invoking the *Swap Macro File* macro which exits Q&A. Since SWAP.BAT now contains another line to run, Windows runs it. This restarts Q&A (in the wink of an eye) with the new module now in memory and its main custom menu displayed.

The final field, **Help**, is a multi-line field on screen page 2. Here you can place any help text or special instructions. The reference on screen page 1—*PgDn-Help*—tells the user to press PgDn if help is needed.

It's hard to appreciate the speed and simplicity of this little module-switching database without seeing it in action. And you can't get a more convenient (two mouse clicks) way of swapping macro files. Users won't have a clue as to how it works, but they won't care. To change modules, all they'll have to know is go to the Q&A Main menu and press M or click on *M - Modules*.

If you want the ability to display the master menu from anywhere in Q&A, simply record a hot-key macro that returns to the Q&A Main menu and "presses" *M*, then add this macro to all your swappable macro files.

Tom Marcellus is the editor of *The Quick Answer*.

About the Download Files

Modules.dtf and the related files featured in this article (the two batch files and six sample macro files) are included in the download file for Online Edition subscribers. Others may order the application for $39 using the order form at www.quickanswer.com, by sending a check to our address on page 2, or by fax with chargecard information.

July 1999 Issue