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

Auto-Print Multiple Labels with Variable Text

TOM MARCELLUS

When you need to print a batch of identical labels with a variable line of text—here's a quick and easy way to do it.

(5.0)

Q&A makes printing merge labels or a batch of identical labels easy. But how would you go about printing three identical shipping labels with something like *Carton 1 of 3*, *Carton 2 of 3*, and *Carton 3 of 3* (see Figure 1)?

A scenario

Suppose you enter a customer order in your ORDERS database, then print it for the shipping department. Later that day, shipping tells you the order is packed in seven cartons, and seven shipping labels are needed. You'd like to print *Carton 1 of 7*, *Carton 2 of 7*, and so forth on the labels, but that would take too much time. So you wind up printing seven identical labels, and handwriting the variable information on each one.

I'll show you how to automate the process. This way, when an order requires multiple labels, you'll simply display the invoice or sales order, and tell Q&A to print that many.

The technique works with any kind of label—and in Add Data or Search/Update. (It doesn't require the record be saved to disk.) What's more, you can optionally print a line of custom *fixed* text—such as *Ship via UPS Ground*—if that's what you want to appear on the batch of labels you're printing.

If you're long on interest but short on time, you can order the two ready-to-use databases (ORDERS and LABELS) from Marble Publications. See "Resource" at the end of the article.

Add two fields to your orders database

The programming and other elements I'll describe assume your order entry database (the database you'll be printing the labels from) contains the following fields: *Invoice No*, *Purchase Order No*, *Company*, *Address*, *City*, *State*, *Zip*, *Number of Labels*, and *Fixed Text Line*. If your fields have different names, or you want to merge other fields with your labels, you can modify the programming and other elements accordingly. (You might want to create a sample order entry database to test the technique before redesigning your
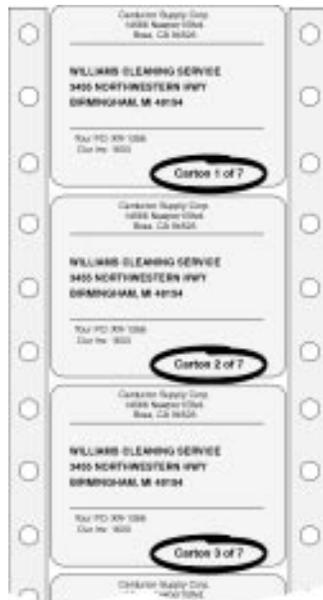
Figure 1. Shipping labels with a variable text line at the bottom.

A Second Copy of Q&A 5.0 Makes a Great Utility Program

TOM MARCELLUS, EDITOR

Something I really like about Q&A 5.0 is the way it lets me temporarily suspend whatever I'm working on, run an automated procedure in a second copy of the program, then return to my original screen. This lets me use that second copy as a powerful utility program—and that's how I think of it.

A case in point: Without leaving the unsaved record I'm working on, I sometimes need to print 2 or 3 or more labels with the same name and address but with a different line of text—such as *Carton 1 of 5*, *Carton 2 of 5*, and so forth—on each label. To do this, I run a routine that tells Q&A how many labels to print and the variable text to include on them, then run the rest of the procedure in my utility program—the second copy of Q&A. This way, when I need to print multiple labels for the customer in the current record, I simply select how many, and in a few seconds I'm back where I started, and the printer is churning out the labels. I'll show you how to do it with a technique you can modify to automate most any multiple label-printing task.

How would you like to be able to search your drives and directories—even select and load files—all from inside a database record? If you've been looking for this kind of capability, you'll find **Jeff Nitka's** Program Spec column irresistible.

Simply knowing that a field has been changed might not be sufficient. If the field contains particularly sensitive information—sensitive enough to track changes to it—you might need to know not only when it was changed, but what older values the more recent values replaced. **A.C. Tynes** shows you the ropes.

I'm pleased to welcome **T.J. Shuflin** as *The Quick Answer'*s new @Help column editor. Those long involved with the National Q&A Users Group think of the soft-spoken, keen-witted T.J. as a kind of brother. Not only has he served as the group's president and guiding light, over the years he has been very active in various Q&A help forums on the electronic services. Dave Reid edited the @Help column for more than four years, and has taken on new responsibilities at Symantec. QA

Auto-Print Multiple Labels . . .

"live" database.)

Place the *Number of Labels* and *Fixed Text Line* fields on the form where they won't get in the way (see Figure 2), and format them for text. In use, when you move to *Number of Labels*, its program asks if you want to add a fixed line of label text instead of the default variable text. If you say Yes, you're prompted to type the line and press Enter. A pop-up list then lets you select the quantity of labels to print.

When you've selected the quantity, the program creates two macros: one that tells the supplementary LABELS.DTF database (which I'll cover later) how many merge records to create and what information from the current record to include in them, and another that auto-prints the labels then returns you to the original record where you started the process.

The program for *Number of Labels* is shown in **Listing 1**. An example of the macro file it creates is shown in **Figure 3**. The macro file is saved to disk (LABELS.ASC) by the *Generate Macro* macro, which I'll describe later. Each time you print labels, *Generate Macro* overwrites the existing LABELS.ASC file that printed the previous batch of labels.

The *Fixed Text Line* field's program is shown in **Listing 2**. It launches a second copy of Q&A, and specifies LABELS.ASC as the macro file to use, and Alt-1 as the autostart macro.

In addition to the programs, add the following navigation commands at the Field Navigation Spec:
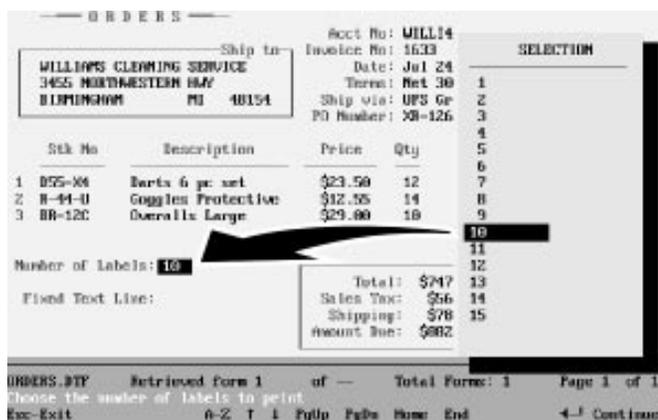

Figure 2. The ORDERS database can be any database from which you might want to print multiple labels for the current record. Here, the Number of Labels field (at the bottom) pops up a selection list for the quantity labels to print. The Fixed Text Line field gives you the option of substituting a fixed text line for the variable text in the current batch of labels.

```
<begdef><alt1><name>"<caps,>alt1<caps.>"<vidoff>fr
labels<enter>s<f10><rgt><enter><esc>a<enter>WILLIAMS
<sp>CLEANING<sp>SERVICE|3455<sp>NORTHWESTERN<sp>HWY|
BIRMINGHAM|MI|48154|5|XR-1266|1633<tab><enddef>
*
<begdef><nokey><name>"Print<sp>Multi-Labels"<vidoff>
<esc>y<esc>wmmultilabel<enter><f2><f10><wait><enter>
<f10><enter><vidoff><esc><esc>x<enddef>
```

Figure 3. A typical LABELS.ASC macro file (for variable label text) created by the Number of Labels program. In this example, the Alt-1 macro tells LABELS.DTF to create five shipping label records for Williams Cleaning Service's on PO XR-1266 and your invoice 1633. The Print Multi-Labels macro, which is run after the LABEL records have been auto-created, selects a label named MultiLabel and prompts you to put the labels in the printer.

```
Number of Labels: > Goto Fixed Text Line
Fixed Text Line: > Goto Number of Labels
```

Listing 1. Program for the Number of Labels field. Substitute your own field names for those shown. In the Usl (Userselect) command, precede the 1 through 9 selections with a space so the list appears in numeric order. Also, be sure to end the two lines shown in italics with a hard carriage return.

```
< REM "****************************************
        Make sure no label fields are empty.
        ****************************************";

  If Company = "" or Address = "" or City = ""
  or State = "" or Zip = "" or Invoice No = ""
  or Purchase Order No = "" Then
  { @Msgbox("One of these fields is blank:",
    "Company, Address, City, State, Zip,",
    "Invoice No, Purchase Order No"); Goto Zip };

REM "****************************************
     Find out if user wants to substitute a
     FIXED text line for the variable text.
     ****************************************";

  If Fixed Text Line = "" Then
  If @Askuser("Do you want to add",
            "a line of FIXED text",
            "to print on ALL the labels?")
  Then { Number of Labels = "*";
  @Msg("Type the fixed text and press ◊fŸ");
  Goto Fixed Text Line };

  REM "****************************************
     Display selection list for number of
     labels to print, then construct macro.
     ****************************************";

  @Msg("Choose the number of labels to print");
  Usl(" 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15",
  Number of Labels);

  If Number of Labels > 0 Then
  If @Askuser("Print " + @Str(Number of Labels) +
            " labels for",
            Company + "?","") Then
  {
  Number of Labels =
  "<begdef><alt1><name>" + @Chr(34)
  + "<caps,>alt1<caps.>" + @Chr(34) +
    "<vidoff>frlabels<enter>s<f10>" +
    "<rgt><enter><esc>a<enter>"
  + Company + "|" + Address + "|"
  + City + "|" + State + "|"+ @Str(Zip) + "|"
  + @Str(Number of Labels) + "|"
  + @Str(Purchase Order No) + "|" + Fixed Text Line + "|"
  + @Str(Invoice No) + "<tab><enddef>" + "
" + "*" + "
" + "<begdef><nokey><name>" + @Chr(34) + "Print<sp>" +
    "Multi-Labels" + @Chr(34) + "<vidoff><esc>y<esc>" +
    "wmmultilabel<enter><f2><f10><wait><enter><f10>" +
    "<enter><vidoff><esc><esc>x<enddef>";
```

```
REM "*****************************************
    Replace spaces with macro <sp> commands.
    Delete extra `|' if no fixed text.
    *****************************************";

  Number of Labels =
  @Replace(Number of Labels, " ", "<sp>");
  Number of Labels =
  @Replace(Number of Labels, "||", "|");

REM "*****************************************
    Print macro to LABEL.ASC disk file,
    overwriting existing LABEL.ASC file.
    *****************************************";

  @Macro("Generate Macro")
 }

  Else { Clear(Number of Labels); Goto Zip }
```

Listing 2. Program for the Fixed Text Line field. Type the entire @Shell command on one line, substituting your own paths for those shown.

```
< REM "*****************************************
        Shell out to 2nd copy of Q&A.
        Autoload LABELS.ASC macro file, and
        run Alt-1 autostart macro.
        *****************************************";

  If Number of Labels = "*" Then
  {Clear(Number of Labels); STOP };

  If Number of Labels <> "" Then
  Fixed Text Line = @Shell("D:\QA\QA1 -alD:\QA\DOCS\
                            LABELS.ASC -m1");

REM "*****************************************
    On return from 2nd copy of Q&A, clear
    the two label-related fields, and
    let user know labels are printing.
    *****************************************";

  Clear(Fixed Text Line, Number of Labels);
  @Msgbox("Labels should be printing now.","","");
  Goto Zip
```

Create the LABELS database

The supplementary LABELS database (see Figure 4) auto-creates the records needed to merge-print the labels. The Alt-1 autostart macro in LABELS.ASC removes LABELS' records from the previous label-printing job, then dumps the label information string into the *Parse* field and tabs to
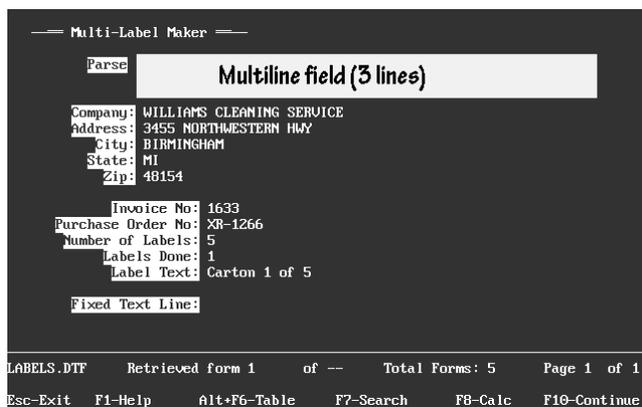

Figure 4. The LABELS database auto-creates a merge-record for each label to be printed. All the fields except Parse, Number of Labels, Labels Done, and Fixed Text Line are merged with the shipping label shown in Figure 1.

the *Company* field. (Be sure to make *Parse* a multiline field of at least three lines.)

Company's program parses the label information string into the appropriate fields, then saves the record via @Save, which displays another blank record. In this second and subsequent records, the on-field-entry program in *Parse* copies the information from the previous record into the current record, courtesy of @Ditto.

In each record added to LABELS, the *Labels Done* field is incremented by 1. If no fixed label text line was specified, then the *Label Text* field is likewise incremented with "Carton 2 of 7," Carton 3 of 7," and so on.

When *Labels Done* equals *Number of Labels*, the *Print Multi-Labels* macro (the second macro in LABELS.ASC) is invoked to print the labels then return you to the original Orders record.

LABELS.DTF requires no customization other than the programs for the *Parse* and Company fields. (See Listings 3 and 4.)

Listing 3. Program for LABELS.DTF's Parse field. You can edit the Label Text = statements in both programs to change "Carton" to "Box," "Package," or whatever you need.

```
< REM "*****************************************
        Alt-1 autostart macro dumps label info
        into Parse field of first record, then
        tabs out of field. As additional records
        are created, @Ditto copies fields from
        previous record.
        *****************************************";

  If @Add Then {

  @Ditto(Company, Address, City, State, Zip,
        Invoice No, Purchase Order No,
        Number of Labels, Labels Done,
        Fixed Text Line);

REM "*****************************************
    In each new record, Labels Done increments
    by 1, and Label Text is updated. When Labels
    Done equals Number of Labels, the label-
    printing macro is invoked.
    *****************************************";

  If Labels Done <> "" and
  Labels Done = Number of Labels Then
  @Macro("Print Multi-Labels")

  Else { Labels Done = Labels Done + 1;

  If Fixed Text Line = "" Then

  Label Text = "Carton " + @Str(Labels Done)
            + " of " + @Str(Number of Labels)

  Else Label Text = Fixed Text Line;

@Save }}
```

Listing 4. Program for the LABELS.DTF Company field.

```
< REM "*****************************************
        Ditto label info from previous record.
        In first record @Ditto returns nothing.
        *****************************************";

  If @Add Then {

  @Ditto(Company, Address, City, State, Zip,
        Invoice No, Purchase Order No,
        Number of Labels, Labels Done, Fixed Text Line);
```

```
REM "****************************************
    In first record, Company is blank, with
    the label info string in the Parse field.
    Parse the string into the fields.
    ****************************************";

If Company = "" Then {

Company = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

Address = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

City = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

State = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

Zip = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

Number of Labels = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

Purchase Order No = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

Fixed Text Line = @Left(Parse, @Instr(Parse, "|") -1);
    Parse = @Del(Parse, 1, @Instr(Parse, "|"));

Invoice No = Parse; Clear(Parse);

REM "****************************************
    Fill Labels Done and Label Text, save
    the record, and get another blank one.
    ****************************************";

  Labels Done = "1";

  If Fixed Text Line = "" then

  Label Text = "Carton " + @Str(Labels Done)
              + " of " + @Str(Number of Labels)

  Else Label Text = Fixed Text Line;

  @Save
}}
```

Create the macro

The two LABELS.ASC macros are created by the
programming, but you'll need to create the following
Generate Macro macro, and save it to your regular default
macro file (usually QAMACRO.ASC):

```
<begdef><nokey><name>"Generate<sp>Macro"<vidoff><f6><f2>
1<dn>0<dn><home><dn><home><rgt><dn><end><dn><end><dn>
<f10>ilabels.asc<enter>y<f6><tab><enddef>
```

The *Generate Macro* macro creates a disk file from the
macros constructed by the Orders database program. It
opens *Number of Labels*' field editor, fills in the Field Print
Options screen, then saves the file to disk in the default
documents directory, overwriting the existing
LABELS.ASC file.

Using the procedure

When all is in place, take the following steps before
running the label-printing procedure for the first time:

- The *Generate Macro* macro expects a LABELS.ASC file
  to overwrite, so be sure you have a file by that name
  in your default documents subdirectory. (You can
  always change the code to save and retrieve the file
  from any directory.)

Resource

Both databases featured in this article (ORDERS and LABELS)
plus the *Generate Macro* macro are available on disk for $39
plus $2 shipping from Marble Publications, PO Box 9034,
Gaithersburg, MD 20898-9034. If ordering by Visa, Mastercard,
or American Express, you can call 800-780-5474, or fax your
order to 301-424-1658.

You can use the supplementary LABELS database as is,
and optionally copy the two programs from ORDERS for use
in your own order entry or other database. The programming
isn't locked, so you can edit or copy it as needed.

- The *Print Multi-Labels* macro expects a mailing label
  named *MultiLabel*. Name your label *MultiLabel*, or
  modify the program that creates the macro (see Listing
  1), substituting the name of your own label. Don't
  forget to include the *Label Text* merge field on the label.

- The macros are designed to run with Automatic
  Execution set to Yes at the Utilities / Global Options
  screen. If you don't use Automatic Execution, you'll
  have to modify the macros and programs to add the
  Enter command that must follow any Q&A menu
  selection.

To use the procedure, display an order entry record in
Search/Update mode, or add a new record in Add Data
mode. (If you're adding a new record, fill in the fields that
will print on the labels.

Move to the *Number of Labels* field. If you want to
replace the default variable text line with a custom fixed
text line, press "Y" when prompted, type the custom text
line, press Enter, then select the number of labels to print
from the pop-up list. The program prompts you to
confirm that you want to print the selected quantity of
labels for the company in the current record.

You can use the same database to print one or more
labels for non-order-related parcels. For example, if you
mail a weekly package of documents to the home office,
you might want to print up a dozen or more labels in
advance and keep them in your desk drawer. Start a new
Orders record, and fill out the home office name, address,
and so forth. Type "N/A" in the Purchase Order No and
Invoice No fields, and "First Class Mail" in the Fixed
Label Text field, then select the number of labels to print.
When the program returns you to the unsaved record,
simply abandon it. QA

Tom Marcellus is editor of *The Quick Answer* and author of *PC World Q&A
Bible* (IDG Books). His QuickClick Calendar Plus—a time- and activity-
tracking database for Q&A 5.0—is available from Marble Publications,
publisher of *The Quick Answer*.

Surf Your Drive with Q&A 5

JEFF NITKA

Can you imagine a Q&A database that lets you select directories, list your files, and even view them like Q&A's List Manager or Windows' File Manager? Jeff has worked out all the code.

In last month's Program Spec column, I offered some programming to display a pop-up list of directories on your hard drive. This month, I'll show you a neat little Q&A 5.0 database that lets you select any file in any directory on any of your computer's drives. When you've selected a file—as long as it's an ASCII text file or a Q&A write document—you can even insert it into the record to view it.

I use this technique in my FAXMAN database (see "The No-Overhead Way to Fax from Q&A" in last month's issue) to view and fax the contents of document files. Also, I maintain a database of all my correspondence (see "Create a Documents Catalogue" in the July 1995 issue), and use the technique to select documents from pop-up lists to insert into records.

Shooting the curl

To surf your drive, you'll simply add a record to the database (let's call it SURFDRVE.DTF). A pop-up list prompts you to select one of the drives specified in the database's lookup table. When you do, another pop-up list gives you the following five choices:

- **List Files** displays another selection list where you choose the type of files to be displayed (ASCII text files are *.TXT, Write files are *.WRI, and document files are *.DOC). Choosing the file type displays a list of all associated filenames in the currently displayed directory. Choosing a filename from the list inserts it in a field and opens the editor so you can view it.

- **Forward** displays a selection list of all available directories in the currently displayed directory (if none exist, nothing happens). Choosing a directory from the list places your choice in the *Directory* field, and you can continue surfing.

- **Backward** returns you to the previously displayed directory. For example, if the Directory field displays C:\QA5\DOCS, and you choose *Backward*, then Directory changes to C:\QA5, and you can continue surfing.

- **Change Drive** lets you to change to the drive you want to surf.

- **Quit** lets you to exit the database.

You can use the database as is, or add the fields and programming to an existing database. You can also simplify the code if all you want, for example, is to display a selection list of Write *.DOC files in a particular directory. If you don't have the time or inclination to type all the programming code, order SURFDRVE.DTF ready-to-use from Marble Publications. See "Resource" at the end of the article.

Creating SURFDRVE.DTF

All you need are five text fields in this order: *Directory*, *Sub*, *Type*, *FileName*, and *View*. When you've finished adding the fields to the database, proceed to the Programming menu.

First, choose Read Only Fields, and make the *Directory, Sub*, and *Filename* fields read-only. Then, choose Field Navigation, and type > *goto Sub* in the *View* field.

Next, choose Edit Lookup Table. In the first row of the Key column, type *Drives*. In the column labeled *1*, list your computer's available drives (even floppy and CD ROM drives if you like), separating each with a comma. For example, if your drives are A:, C:, D:, and F:, then in column 1 of the *Drives* row, type *A:,C:,D:,F:*

In the second row of the Key column type *Hard Drive*. This drive will be used by SURFDRVE to record some data about your computer's files. It can be a network hard drive or a local hard drive, but it should be permanent, and it can't be write-protected. Most likely, you'll enter your *C:* (local hard drive) or the name of a network drive in column 1 of the table.

Select Program Form, and enter the programs shown in Listings 1 through 5. When typing the programs, you'll encounter three occurrences of the following statement:

```
        Sub = @Us( @Replace( Sub ,"
",",") );
```

The statement isn't wrapped for publication. You must type it exactly as shown. At the end of the first line, you must press Enter to insert a hard a carriage return.

Listing 1. The program for the Directory field.

```
< If @lu("Drives",1) = ""  or
    @lu("Hard Drive",1) = "" then
{ @MsgBox("Lookup table entry DRIVES or HARD",
"DRIVE does not contain data. Please enter the",
"necessary data in the lookup table.");
              Directory = "Quit"; Goto Sub }
else { @Msg("Select the drive you wish to surf.");
    Directory = @Us( @lu("Drives",1) );
    if Directory = "" then Goto Directory
    else Goto Sub }
```

Listing 2. The program for the Sub field.

```
< If Directory = "Quit" then Goto Filename else
{ Filename = "";
  Sub = @lt(@Us("1-List Files,2-Forward," +
          "3-Backward,4-Change Drive,5-Quit"),1);

  if Sub = "" then goto Sub        else
  if Sub = 5  then {Type="Quit"; Goto Type} else
  if Sub = 4  then
  { Clear(Directory,Sub,Filename,View);
    Goto Directory } else
   if Sub = 3
  then { Sub = @in(Directory,"\");
         if Sub = 0 then Goto Sub
         else { Sub = @Repllas(Directory,"\","~");
             Directory = @lt(Sub,@in(Sub,"~")-1);
               Goto Sub } }
  else
  if   Sub = 2
  then { Sub = @Shell("dir " + Directory +
        "\*.* /A:d /b > " + @lu("Hard Drive",1)
        + "\DocsList.txt");
        Sub = @Insert(@lu("Hard Drive",1) +
                   "\DocsList.txt");
        if @len(Sub) = 0 then Goto Sub
        else { @Msg("Select from following " +
               "directories in " + Directory );
               Sub = @Us( @Replace( Sub ,"
",",") );
               Directory = Directory +
               @txt(Sub<>"","\" + Sub);
               Goto Sub }
        }
  else Goto Type }
```

Listing 3. The program for the Type field.

```
< If Type <> "Quit" and Directory <> "Quit" then
{ @Msg("Choose file type. For all files," +
      " press ENTER twice.");  Type = "";

  Type = @Us("*User Entered*,Write Files (*.WRI),"
      + "Documents (*.DOC),Text Files (*.TXT)");
  Type = @lt(Type,3);

  if Type = ""    then Goto Sub      else
  if Type = "Wri" then Type = "*.WRI"  else
  if Type = "Doc" then Type = "*.DOC"  else
  if Type = "Tex" then Type = "*.TXT"  else
     Type = "";

  if Type <> "" then goto Filename else
  @Msg("Enter a filetype, such as *.DTF,*.EXE, " +
      "etc. (leave BLANK for ALL)")
}
else { Clear(Filename,View,Type,Sub);
     @Msg("You can escape from the record now.")}
```

Listing 4. The program for the FileName field.

```
< If Directory <> "Quit" then
{ @Msg("Select from the following files in " +
      Directory);

  if Type = "" then Type = "*.*";

  if Sub = "..Again.."
  then { Sub = @Insert(@lu("Hard Drive",1) +
                    "\DocsList.txt");
        Sub = @Replace( Sub ,"
",",");  Filename=@Us("*Change Directory*,"+Sub);
        if Filename = "*Change Directory*"
        then { Filename = ""; Goto Sub } }
  else { Sub = @Shell("dir " + Directory + "\" +
        Type + " /A:-d /b > " +
        @lu("Hard Drive",1) + "\DocsList.txt");
        Sub = @Insert(@lu("Hard Drive",1) +
             "\DocsList.txt");
        if @len(Sub) = 0 then
        { @MsgBox("There are no " + Type +
          " files in " + Directory,"or " +
          Directory + " does not exist.",
         "Please select a new directory.");
          Goto Sub }
        else { Sub = @Replace( Sub ,"
",",");
             Filename=@Us("*Change Directory*,"
                    + Sub);
             if   Filename="*Change Directory*"
             then { Filename = ""; Goto Sub } }
     };

  if   Filename = ""
  then { Sub = "..Again.."; Goto Filename }
  else { View = @Insert(Directory+"\"+Filename);
        Goto View }
}
else Goto Type
```

Listing 5. The program for the *View* field.

```
< If   Filename <> "" and View <> ""
  then @Fedit else goto Sub
```

When you've finished adding the programming, press F10 to save the Spec. Now, grab your surfboard, open SURFDRVE in Add Data mode, and "hang-ten"!

Jeff Nitka works for a chemical manufacturer and develops Q&A applications part-time for Epoch Software, 908-874-3989. Jeff is the author of the Program Evaluator (a Q&A program debugging utility) and FaxMan (a Q&A faxing database), both available from Marble Publications.

@Help

EDITED BY T.J. SHUFLIN

Finding Earliest and Latest Dates

My database contains a series of date fields. I need separate fields and programming to find and store the earliest and latest dates in the series. What do you suggest?

One solution is to use Q&A's @Max and @Min functions. Suppose your series of date fields are numbered #30, #50, #65, #80, #82, #84, and #90 in the Program Spec. You could use programs like these to find the earliest and latest dates:

```
Earliest Date: #200 = @Min(#30, #50, #65, #80..#84, #90)
  Latest Date: #210 = @Max(#30, #50, #65, #80..#84, #90)
```

Where a range of fields (such as #80..#84) can't be specified due to intervening non-date field numbers, list each field ID number separately. You can optionally combine the program in a single field, then make it an on-record-exit field so the information would be updated each time a record is saved:

```
Latest Date: #200 = @Min(#30, #50, #65, #82..#84, #90);
             #210 = @Max(#30, #50, #65, #82..#84, #90)
```

To specify an on-record-exit field, press F8 while at the Program Spec, and type in the pertinent field's ID number. (In this case, you'd type 200 without the "#" symbol.)

Use Current Date and Time to Generate Incident Numbers

We need Q&A to assign a unique Incident Number to each new record we add to our database. The number must contain abbreviations that tell us the day of the week, month, date, and time the record was created. We don't need the year (or seconds). The database is single-user.

If I understand you correctly, for a new record created on Sunday, December 1, at 2:47 p.m., you need Q&A to generate an Incident Number like *Su12/01-14:47*.

I advise adding separate fields for the date and time—they might come in handy for sorting or other purposes, but they're not required.

You can program the fields along the following lines:

```
Date: <#10: If @Add then {#10 = @Date; Cnext}
      else goto #40

Time: <#20: If @Add then #20 = @Time; Cnext

Incident Number: <#30: If @Add then {#30 =
@Left(@Dow$(#10),2)
              + @Right(#10,5) + "-"+ @Left(#20,5);
Cnext}

Next Field: #40
```

The Incident Number field is formatted for text. Format the Time field for 24-hour military time, with a colon separating the hours and the minutes.

These fields could be the first three in the database so that an on-field-entry program would create the Incident Number as soon as a new blank record is displayed. The fields should be made read-only to prevent accidental changes or tampering.

With the above programming, if the record is displayed in Search/Update mode, the cursor is simply moved to the Next field (#40). In Add mode, the date and time are read from the PC's system clock and stored in the Date and Time fields.

The Incident Number program generates the unique value by adding the first two characters of the day of the week to the last five characters of the date (Q&A stores dates in a YYYY/MM/DD format), then adding the first five characters of the time. (Q&A stores times of day in HH:MM:SS military format.) A "dash" is inserted between the date and time for readability.

To generate the Incident Number without Date or Time fields, you could use a program like this:

```
Incident Number:
<#100: If @Update then Cnext;
If @Add then #100 = @Left(@DOW$(@Date),2)
+ @Right(@Date,5) + "-" + @Left(@Time,5); Cnext
```

Of course, the PC's internal system clock must be accurate for the program to create an accurate Incident Number. Two or more records added to the database during the same minute would have the same Incident Number. If this could happen, you'll want to modify the program to include seconds.

The programs I've shown generate 13-character Incident Numbers, well within Q&A's 16-character XLookup limitation for unique values.

When @XLookup Draws a Blank

When my program triggers an @XLookup to retrieve information from an external database, my target fields in the displayed record are sometimes blanked out. Is there a way to prevent this from happening?

Q&A has an XLookup statement and an @XLookup function. Both are designed to find and retrieve information, usually from an external database, but they react differently to blank fields.

An XLookup statement includes the *destination* field as its final parameter. This way, you can tell Q&A where to place the information it retrieves from the matching record in the other database. If the matching field in the external database is blank, there's nothing for Q&A to place in the destination field, so it remains unchanged. (The same holds true if the XLookup statement can't find a matching record in the external database.)

The @XLookup function works differently. If the matching field in the external database contains nothing, or there's no matching record found, Q&A places that "nothing" in the destination field, blanking it out.

Another way a destination field can be blanked out is if it's formatted for information other than text, and the external field doesn't contain the "proper" type of data. If, for example, both fields are formatted for dates, but the external field contains a non-date value, Q&A returns nothing to the destination field, blanking it out.

Because the @XLookup function doesn't include a destination field parameter, you can use it in programming expressions in merge documents and labels, as a variable in formulas, or even in report derived columns.

Indent a Report Column

I have a database field for the sport name, and a multi-line field for the description. I'd like to print a report with these two fields as columns, and indent just the first lines of the descriptions. The Description column should be about 40 characters wide.

You'll need a derived column trick that uses a special non-printing ASCII character. The first technique I'll show uses two derived columns—the other uses just one. Your Column/Sort Spec should look like this:

```
Sport: 1,AS        (For an ascending alphabetical sort)
Description: 4,I   (The column made invisible)
```

From the Column/Sort Spec, press F8 for Derived Columns, and create the following two columns. Later, I'll explain how to insert the non-printing character (NPC):

```
Heading:        Indent Trick
Formula:        "NPC     "
Column Spec:    2,I

Heading:        40: Description
Formula:        #2 + #4
Column Spec:    10
```

To create the formula containing the NPC in the first derived column, type the quotation mark, then press Alt-F10. Press and hold the Alt key down, and press 7 on the numeric keypad. (Be sure NumLock is on.) Release the Alt key, press the spacebar five times to indent five spaces, then type the closing quote mark. When you released the Alt key, a small solid bullet character should have appeared. This is the non-printing ASCII character that facilitates the indent.

Your printed report should look something like this:

```
 Title      Description
 --------   ----------------------------------------
 Baseball   Baseball is a game played on a
            field called a "diamond." The field
            has four "bases" that constitute the
            diamond shape of the infield.

 Golf       Golf is a sport where the object
            is to place a small hard ball into a
            small hole in the ground from a long
            distance.

 Tennis     Tennis is a game played on a court
            marked with white lines. In singles
            play, one opponent faces another, ...
```

The derived column heading, *40: Description*, forces the column width to 40 characters. The *#2 + #4* formula "adds" the five spaces to the beginning of the first line of the Description column. The column containing the non-printing character and the original Description field are made invisible so they don't appear on the report.

The single derived column method looks like this:

```
Heading:        40: Description
Formula:        @Text(5, @Chr(7)) + #4
Column spec:    10
```

To increase or decrease the number of characters to indent, simply change the 5 the formula to the number of characters you want Q&A to indent the first line.

Importing an Almost-Fixed ASCII File

I have an ASCII file where all the data *except* the last field is in Fixed ASCII format. The file contains names, addresses, phone numbers, and some additional fields. The last field varies from five to 30 characters, and is followed by a hard carriage return record delimiter.

I've tried importing the data into my database as a Fixed ASCII file, but that final variable-length field causes problems. Any suggestions?

I assume that since it's a Fixed ASCII file it contains no quote marks, and no characters as field delimiters. (In Fixed ASCII files, fields are delimited by their starting column number and length.) This is important because the technique I'll show you involves importing the file as a *Standard* ASCII rather than a Fixed ASCII file.

Start by adding the following four fields to the beginning of the destination database :

Field Name	Special Formatting
Full Import	Multi-line field large enough to store an entire record from the imported file.
Total Field Length	Number field.
Field Length minus *xxx*	Number field. (I'll explain the *xxx* value later.)
Info for Lib#	Text field, 30 characters minimum

The following table shows the remaining fields in your database corresponding to the fields in the Fixed ASCII file. The Length column shows the field lengths in the Fixed ASCII file, so your corresponding database fields should be at least that length:

Field Name	Length
Name	31
Street	41
City	21
State	6
Zip	13
Phone	22
Occup	42
Referred	15
Date	18
Lib#	5 to 30

The Lib# field in the Fixed ASCII file varies from five to 30 characters, so make it 30 characters in the database.

With your database ready, import the ASCII file into it, importing each record—the entire record—into the Full Import Field. To do this, select File / Utilities / Import, and choose *Standard* ASCII. Type in the name of the ASCII file, followed by the destination database. Leave the Merge Spec blank, and press F10 to continue. At the ASCII Options screen, set *Quotes around text* to No, and set the *Field delimiter* to a semicolon.

Press F10 to perform the import, then verify that each ASCII file record's data (and no more) wound up in the

Full Import field (the first field) of each record in the database.

Next, you'll need a program to determine the length of the value in the Full Import Field. Since the lengths of all the Fixed ASCII fields (except the last one) are known, you can subtract this known length from the total length to determine the length of the final variable-length field.

Once that's done, programs in the other fields can parse (separate) the Full Import Field, copying the appropriate data into the respective fields. Here's the Program Spec:

```
Full Import:            #10
Total Field Length:     <#20: #20= @Len(#10)
Field Length minus 209: <#30: #30= #20-209
Info for Lib#:          <#40: #40=@Right(#10,#30);
#100=#40
Name:                   <#70: #70=@Left(#10,31)
Street:                 <#72: #72=@Mid(#10,32,41)
City:                   <#74: #74=@Mid(#10,73,21)
St:                     <#76: #76=@Mid(#10,94,6)
Zip:                    <#78: #78=@Mid(#10,100,13)
Phone:                  <#80: #80=@Mid(#10,113,22)
Occup:                  <#82: #82=@Mid(#10,135,42)
Referred:               <#84: #84=@Mid(#10,177,15)
Date:                   <#86: #86=@Mid(#10,192,18)
Lib#:                   #100
```

The program in Total Field Length determines the length of the Full Import Field value (the imported record). Once that's known, 209 (the total of the fixed field lengths) can be subtracted to find the length of the pesky variable-length field. The remaining statements parse the remaining information into the respective fields.

Because the programs are on-field-entry, you can step through each field in a number of records to view how the data is parsed. When you're confident the programming is doing the job, you can remove the on-field-entry symbols from each field, then Mass Update the imported records.

When performing the Mass Update, press F10 at the Retrieve Spec to select all the records. Leave the Update Spec blank, press F8 for the Calculation Options screen, and set Calculation statements to Yes. QA

T.J. Shuflin is a forensic chemist and lab director of a crime laboratory in Alexandria, Louisiana. Since serving as the first president of the National Q&A User Group in 1991–92, he remains active with the group by maintaining its Internet Web site at http://www.timetrend.com/~tshuflin.

When Was That Field Changed, and What Was It Before?

A. C. TYNES

When you need to monitor changes to sensitive fields in your database, use this technique to tell you not only when a field was changed, but to maintain a history of its previous values.

Knowing that a field was changed at some point isn't always enough. If it contains sensitive information—sensitive enough to track changes to it—you might need to know when it was changed, and what older values newer values replaced. For example, has a customer's credit rating or limit been updated? If so, you might want to know when, and what the earlier rating or limit was. Likewise, you might want to maintain a history of changes to the costs or selling prices of your inventory items, or changes to your customers' addresses.

Using @Update or @Modified to track changes to fields isn't likely to give you what you need. @Update programming can execute when you simply view a record. And @Modified executes when *any* field is changed. Though you can use @XLookup to compare a field's value in the saved version of the record to the changed value in the current record, external lookups can slow things down, particularly on a network.

There's another solution that works for all versions of Q&A, and you can use it to track changes to any number of sensitive fields. All you need are three additional fields for each field you want to monitor, and a few lines of programming.

An example

To illustrate the technique, let's take a simple example. Suppose you want to monitor changes to ZIP codes to determine if a retiree is eligible for a certain medical plan. Among other fields, the database contains ID Number (#5) and Zip (#10) fields. Add Last Changed (#20), History (#30), and Compare Zip (#40) fields to the database, formatting Last Changed for Dates, and the other two for text. (You can make Last Changed and Compare Zip read-only, and place Compare Zip on an unused page since you won't need to see it.)

Next, go to the Program Spec, and type the following program in the Zip field:

Tracking Changes to a Number Field

I always format ZIP code fields for text to accommodate ZIP+4 (with a dash) entries. But what if you're monitoring a money field, such as a parts price?

For such cases, I use an additional field to convert the money amount to a text value before adding it to the History field. This makes it easier to include the dollar sign, commas, and trailing zeros, which would otherwise be missing when converting money values to text.

Suppose fields #10 and #40, mentioned earlier, are money fields, and you've added field #50 and named it Money as Text. Change the reference on line 4 from #40 to #50, and insert the following statements immediately after the "#40 = #10" on line 4:

```
If #40 = 0 then #50 = "$.00"
   Else Begin #50 = @Str(#40+.001);
   #50 = @Left(#50,(@Len(#50)-1));
   If @Len(#50) < 7 then #50 = "$" + #50
   Else
   If @Len(#50) < 10 then #50 = "$" +
   @Left(#50,(@Len(#50) -6)) + "," +
      @Right(#50,6)
   Else
   #50 = "$" + @Left(#50,(@Len(#50)-9)) + "," +
   @Mid(#50,(@Len(#50)-8),3) + "," + @Right(#50,6);
   End
```

The comparison number is zero if #10 is either blank or zero. If so, the program places the proper text value in #50. Again, the action begins with line 2. The program converts the dollar amount in #40 to a string after adding 0.001 to ensure the string contains three decimal places. The third line uses the @Len function to remove the 1 that's forced into the third decimal place. If you need more decimal places, simply add the zeros to the @Str function.

The subsequent statements use @Len to determine the length of the string in order to insert no, one, or two commas (for amounts up $999,999,999.99).

This technique is a reversed and extended version of Jim Pogany's QuickTip on converting text values to money in the January 1996 *The Quick Answer*.

```
>#10: If #10 = #40 then goto #5
     Else Begin #30 = #30 + "; " + @Str(@Right(#20,5))
     + "/" + @Str(@Mid(#20,3,2)) +
     ": " + #40; #20 = @Date; #40 = #10;
     If #30 = "; /:" then #30 = "";
     If @Left(#30,2) = "; " then #30 =
     @Right(#30,(@Len(#30)-2)); goto #5; End
```

The ">" (greater than symbol) sets the program to execute when the cursor leaves field #10 (Zip). Line 1 compares Zip with Compare Zip. If there's a match, the cursor moves to field #5 (ID Number) and the program ends.

If the two ZIP codes don't match, the action begins with line 2. There are several steps to perform, so I use a "Begin" on line 2 and an "End" at the end of the program. I prefer these over the left and right French braces.

First, you want to move the old ZIP code and Last Changed date into the History field (#30). Since there might already be data in History, the program adds a semicolon as a separator, converts the date in field #20 from Q&A's yyyy/mm/dd format to an mm/dd/yy format using the @Right and @Mid functions, inserts a colon, and adds the old ZIP code from field #40.

The @Date function then puts the current date into Last Changed (#20), and the newly entered ZIP code is copied to field #40, Compare Zip.

At this point, the fields have been updated, but History might not look right. If this is the first time a ZIP code has been entered, History shows only "; /" from lines 2 through 4 because there is no data in fields #20 and #40 to copy. The If statement beginning on line 5 checks for this condition and blanks History as necessary.

This might also be the first time a date and ZIP code have been added to History, leaving a leading ";" from the program. The second If statement tests for this condition.

If it exits, you need to get rid of the first two characters. The @Len function determines the number of characters in History, then the @Right function keeps two less than that number.

Now that the History field has been cleaned up, the cursor moves to the ID Number field, and the database shows everything you need to know about the current and prior ZIP codes. (You could also add the User ID to the mix if the database were multi-user and you needed to track *who* changed the field.)

This technique can be used to monitor changes to multiple fields in any database. You simply add the extra fields and appropriate programming. The History field and related programming can be left out if you're interested only in tracking the date of the last change to a field and don't care about its previous value or values. QA

A. C. Tynes is a Certified Public Accountant in Metairie, Louisiana. 504-835-6378.

Which Copy of Q&A are You In?

If you sometimes work in a second copy of Q&A and forget which copy you're in, there's a quick way to tell. From the Main menu, press *U*, then press Enter. Most of the selections on the Utilities menu aren't accessible in any but the first copy, and a message appears telling you so. QA

Tom Marcellus

Q&A 4.0 vs. 5.0 Sorting

When Q&A indexes and sorts data using a collation sequence table built in the database, it treats uppercase and lowercase letters—as well as certain other characters—as equals. However, the Q&A 4.0 table differs from its 5.0 counterpart.

Take the letter *A*. In Q&A 4.0, six variations of *A* (a, â, ä, à, Ä, and á) are treated the same as *A*, while in Q&A 5.0, only *A* and *a* are treated as equals—the other five are treated differently because in languages other than English they're distinct vowels.

Knowing a bit of German, I experimented with some German words. Q&A 4.0 treats the letters *a* and *a-umlaut* (ä) as though they were the same, while Q&A 5.0—showing its German genesis—doesn't.

I found it interesting that a "native" Q&A 4.0 database retains its version 4.0 collation sequence even after it has been upgraded to version 5.0. Moreover, some non-alphabetic characters sort differently. For example, in 4.0 the left square bracket "[" sorts after the letters of the alphabet, while in 5.0 it sorts before them. QA

John Dow

Using QEXTRACT on Corrupted Databases

JOHN DOW

If you've ever used Symantec's QEXTRACT data extraction utility, you might have received confusing results. Here's a possible explanation.

Most *Quick Answer* readers probably know about QEXTRACT, the Symantec utility designed to extract data, form designs, and programming from corrupted Q&A databases. (See "Q&A Utilities to the Rescue" in the October 1994 issue.) With regard to QEXTRACT's data extraction capabilities, I've found two reasons to be cautious about it:

- It's output can scramble the field order.
- It might not retrieve all records.

These sound like serious drawbacks—and they are. Depending on the condition of the database (which I'll discuss later), the utility might create multiple .STR output files with the fields in a different order. When this happens, you have to use different Merge Specs to import the extracted data back into a database. For some users, the problem of missing records could be even more serious. But read on, QEXTRACT has virtues despite these potential problems.

Generations

When you add records to a database, redesign it, and then add more records to it, QEXTRACT might not extract them properly. The QEXTRACT.DOC document file that comes with the utility describes what its makers call *data shift*, which has to do with adding or removing fields from a database.

But what if you simply change the order of the fields? The document doesn't discuss this, but it should.

To see what I mean, create a database with two fields, *Lastname* and *Firstname*. Enter *Dow* and *John*, respectively, into the two fields in a new record, like this:

```
 Lastname: Dow
Firstname: John
```

Now, redesign the database and swap the order of the fields, being sure to move their internal tags (AA and AB)

with them. You now have the *Firstname* before the *Lastname* field.

Add another record, this time entering *Harry* in the *Firstname* field, and *Houdini* in the *Lastname* field. The database now contains these two records:

```
Record 1    Firstname: John
            Lastname : Dow

Record 2    Firstname: Harry
            Lastname : Houdini
```

If you now use QEXTRACT to extract these two records, then look at the 2.STR file it generates, you'll see this:

Wrong (what QEXTRACT does)	Right
"Harry","Houdini"	"Harry","Houdini"
"Dow","John"	"John","Dow"

The fields containing John and Dow actually appear in the record internally with Dow followed by John. When QEXTRACT extracts the fields, it doesn't realize that the order of the data fields no longer corresponds to the order of the form fields.

Missing records

The QEXTRACT.DOC file mentions that the utility might not retrieve all the records, but doesn't elaborate. The problem might be more common than its makers would have us believe.

For example, I had a small database with 10 records, and QEXTRACT extracted only seven of them. There was nothing wrong with this database as far as I could tell. In fact, Q&A itself thought there were 10 records. My own DTFDATA utility (see "Resource" at the end of this article) reported 10 records, and extracted all 10.

(Reporting the number of records and extracting them are independent tasks that use different information in the .DTF file. The fact that both resulted in 10 meant that the .DTF file was internally consistent.)

I then ran QEXTRACT on a database of 1,432 records. (At least that was the number of records Q&A reported it contained. It was also the number DTFDATA said it had, and the number DTFDATA actually extracted.) But QEXTRACT extracted only 1,428 records, missing four.

I suspected the problem might be related to

QEXTRACT's inability to recognize that the order of the fields had changed. So, out of curiosity, I ran DTFDATA on the little two-record database mentioned earlier using the following switches. DTFDATA extracted "Harry","Houdini" and "John","Dow" in the correct order:

```
DTFDATA databasename FORMAT=Q outputfile
```

The reason I suspected QEXTRACT might misbehave in this manner is that it creates multiple output files according to the number of fields in each data record. This implied that it wasn't using the generation information stored in the .DTF file. If it had, it would have been able to put the fields into the correct order, and make sure all the records had the correct number of fields.

Because DTFDATA uses the generation information, it can create a single output file regardless of how many generations are represented in the database. You don't have to work with multiple output files, and you don't have to fool around with multiple Merge Specs when importing the data.

A saving grace

I speculated that Symantec might have designed QEXTRACT to work this way so that it could extract records from extremely badly corrupted .DTF files. So, I created a small database, then performed a "lobotomy" on

Resource

DTFDATA, John's latest utility for Q&A, can perform magic on Q&A databases, including the following:

- Extract data to import into another database.
- Extract data in real time for use within another application.
- Extract all data or selected fields from selected records.
- Print extracted data to the screen or save it to a file.
- Create multicolumn and wide column picklists for Q&A 5.0.
- Perform predefined queries in under 1/2 second.
- Instantly retrieve the number of records in a database.

DTFDATA is $55 postpaid. Make your check payable to John Dow, and mail it to 6560 Rosemoor St, Pittsburgh, PA 15217. A free demo of DTFDATA is available from Symantec's Q&A library on CompuServe, or from John's Home page, http://dropit.pgh.net/~jtd.

it, deleting a part of the file I knew to be crucial to Q&A (and crucial to DTFDATA as well). Here are the results:

Q&A reported System error C-2 Ref#:06A0

DTFDATA didn't recognize it as a Q&A .DTF file.

QEXTRACT correctly extracted all records. It was even able to correctly prompt for a User ID and password.

I concluded that QEXTRACT, irrespective of *how* it works, is the best utility to extract data from a severely corrupted database.

I also concluded that because QEXTRACT tries to find and extract the data from records without using information in the database that's crucial for Q&A and DTFDATA, QEXTRACT can sometimes be wrong—that is, it might fail to recognize some data records in the file.

Protecting your data

Here are my recommendations for reducing the possibility of data corruption, and what you can do if you encounter it:

- Use Q&A itself or DTFDATA to extract your data before resorting to QEXTRACT. When using Q&A, copying records to another database is better than exporting them to a Standard ASCII file if the file contains special characters, such as embedded carriage returns. (DTFDATA's *format=Q* switch preserves nearly all special characters.)
  (As far as I can tell, Q&A's *Recover database* utility is designed to recover disk space and rebuild indexes, not fix data corruption.)

- Keep your database's records clean. Experienced users know that you can run a "null" Mass Update on a database to bring all its records up to the current generation. If you've redesigned a database form, you could save yourself a lot of grief later on by performing this simple step.
  If you want to see the generations in your data records, get version 1.01 of my DTFCHECK utility. (You can download it for free from the Symantec Q&A file library on CompuServe). Running DTFCHECK with the *-v* and *-d* switches displays the generation information for each record in the database.

- Routinely backup your databases. QA

John Dow has been working with computers professionally since the 1960s. Currently, he's director of development at The Lackner Group in Pittsburgh. John is the creator of DTFSPECS, a Q&A programmer's tool reviewed in the October 1995 The Quick Answer. 412-521-1577, Internet jtd@pgh.net, CompuServe 102701,2751.

Upgrading 4.0 Databases to Q&A 5.0

GORDON MEIGS

Q&A 5.0 automatically upgrades 4.0 databases. But you can head-off potential problems by performing a few preliminary steps on your own.

You upgrade a Q&A 4.0 database to Q&A 5.0 simply by opening it in 5.0. However, some Q&A users have reported problems with databases upgraded this way. To ensure a smooth upgrade to 5.0 and minimize potential problems with your databases, here are some steps you can perform:

1. Be sure all reports in the 4.0 database are the latest generation. You can use any of the following methods to do this:

   Method 1—Run DTFCHECK on the database. (See "Report Retrieve Spec Glitch Finally Solved," in the October 1996 issue.) If no warning message appears, your reports are okay. If you receive a warning message, DTFCHECK advises which reports are in danger or already "bad."

   For reports that are in danger, bring them up to the latest generation by selecting Design/Redesign a report for each report in the database and pressing F10 until you've re-saved each one. For "bad" reports, look for garbage characters in the report's Specs—particularly in the last field of the Retrieve Spec. Delete any you find before re-saving the report.

   Method 2—Select Design/redesign a report for each report in the database, and press F10 until you've re-saved each one. (Check the last field in each report's Retrieve Spec as in Method 1.)

   Method 3—If you have the latest version of DTFSPECS, you can use it to generate macro files that automatically fix reports in the database. (See the article mentioned earlier.)

   Method 3 also lets you copy report designs to another database with the same form design. You might want to do this to add a newly designed report from your master database to a client's copy of the database, or to copy a report design from an active database (INVOICE.DTF, for example) to an archive version of the database.

2. Use the File / Copy / Copy design only command to copy the database design to a new name, creating a new, single generation database. Follow this by copying all the records from the old database to the new one, then renaming the new database to the old filename.

2. Backup the database.

4. Select Redesign a database, press F10 to get to the Global Format Options screen, and change the date display format to date format 1. Then, go to the Initial Values Spec, and delete any initial @Date values. (As far as we know, @Date is the only initial value that can cause a problem).

5. Go to the Speed-up Spec, make a note any Speedy fields (S, SU, and SE), delete any Speedy codes, and re-save the Spec.

6. Use the File / Utility / Recover command to recover the 4.0 database. Then, open the database in Q&A 5.0 to upgrade it.

7. Re-specify any @Date Initial Values, and change the database's date display format (at the Global Format Options screen) to date format 1, 3, 5, 11, 13, 14, 15, 16, 19, or 20. (See "Date Bug Can be Double Trouble" in the March 1996 issue.)

8. Go to the Speed-up Spec and redefine your Speedy fields.

9. Recover the upgraded database, and run a Mass Update on it using a blank Update Spec.

10. If you're using Q&A 5.0 on a peer-to-peer network, set the database's Sharing Mode (select File / Design a file / Redesign a file / Secure a file / Declare sharing mode) to *Allow*.

11. The upgraded database will have Q&A's default color scheme. You can change the color palette by selecting File / Design a file / Customize a file / Change palette.

12. A bug in Q&A 5.0 can crash the PC if a default Font Assignments screen exists and F6 is pressed at the Retrieve Spec for a mail merge document. (See "5.0 Mail Merge Bug Uncovered" on page 7 of the April 1996 issue.) To work around the bug, delete the default Font Assignments screen, and create a *Set*

Fonts macro to auto-fill the Font Assignments screen (assigning the desired default fonts) for new documents.

By following these steps, you increase the chances that you're upgraded databases will be trouble-free. QA

Gordon Meigs is vice president and general manager of Professional Computer Technology Associates of Newtown, Pennsylvania. He teaches courses and does corporate training on Q&A, and has been designing and installing advanced Q&A business applications for more than nine years. 215-598-8440, CompuServe 71023,356.

Quick Tip

Two Shapes for Multiline Fields

When you need a field for notes or comments, you can design a multiline field into the form or make the field any size, then use the Field Editor to expand it.

With a multiline field, you can display more text without having to move the cursor to it and press F6 to expand it. However, a multiline field takes up more form space, and you can't place fields to the left or right of it, or surround it with Line Draw.

You create a multiline field by starting with a field label and colon, then ending the field one or more lines below it with the right angle bracket end field delimiter (">"). You'll have a text entry area like the top of Figure 1. However, If you use the left angle bracket ("<") in place of the colon, the same multiline field gives you the rectangular text entry area shown at the bottom of Figure 1. Either way, a multiline field can still be expanded by pressing F6. QA

Tom Marcellus

Figure 1. Using the left angle bracket instead of the colon, you can make a rectangular multiline field.