Part 18

•By Fenton Jones <manavesh@mymac.com>•

Labelling & Troubleshooting Fields, Calculations, Relationships & Scripts

In this article I'm going to give you a few tips on how to organize your database. In the rush of creation we often neglect these simple steps to document our own work. Then later we pay the price, as we've forgotten exactly what all those darn things are and what they do.

The first step in keeping track of what things are is to name them, and the first things to name (other than the files themselves) are the fields.
There are, as far as I can tell, two main systems for naming fields.

Alphabetical Field Names
The first is plain old alphabetical. If you don't have too many fields this may be all you'll need. The key to this system is to name all fields that you want to keep together with a similar beginning to the name, adding bits to the end for further identification.

Field Name	Field Type	Formula
Service	Calculation,	= ServRate*ServHours
Serv_Sum	Summary,	= Total of Service
Serv\Job_Total	Calculation,	= Sum(Self_Job ID::Service)*

*(Self_Service being the name of a self-relationship of the Job ID)

As you can see, there are several fields beginning with "Serv". Choosing view by Field Name in the Field Definitions will instantly bring them together.
You can adjust the order somewhat by using different punctuation. Sticky spaces between the words will also work, adding the advantage of keeping the words together in calculations. I often use that for IDs, since the extra space helps them stand out in lists. (But I'll spare our webmaster and just use regular spaces here.)

I also use the sticky space and a short suffix to identify types of fields. For example; " g" for global fields, " c" for calculation, " gc" for concatenated globals. That way I can quickly tell them apart from a regular field with the same name.

One problem I've found is to keep the distinction between actual Summary fields and Total fields which are based on aggregate functions. They are not always the same; the Summary depends on the found records, while an aggregate using a self-relationship does not. So I always use the letters "Sum" for Summaries and "Total" or "TOT" for Sum() aggregates. You could do it either way, but…

It is important to be consistent. If you name some fields carefully, but just slap on the first thing that comes to mind on others, you will soon have little idea what does what. It can get pretty sticky if there are a lot of fields with similar names. If you find it getting out of hand it's time to move up the next level of organization.

Custom Order Field Names
FileMaker remembers the field name order. So all you have to do is decide which fields go together and drag them into place. Choosing "Custom order" from the drop-down box in the Field Definitions dialog will restore them later if you add new ones.

A further refinement to this method is to name fields with a prefix to show the type of field, much like I did before with suffixes. "g Job ID", "g Service", etc.. If you do an alphabetical sort it will sort by "Type" of field as well.

Obsolete Fields
You can use prefixes and alphabetical sorting to get obsolete fields out of your way. Any field that you think you no longer need can be renamed with "`" (accent) in front. It will sort to the bottom.

Never delete a field if there is an Import order in a script. It will screw up the Import in a way that is very difficult to fix. Just change it to a global field and leave it. It will then be empty and unstored. If you need a new field later, just rename and redefine it. You could also use the "False Calculation" method (two

paragraphs down) to tell yourself just why you "retired" the field.

Conversely "#" and "•" prefixes will sort to the top. This is also a great way to track a field that may be referenced in other fields' definitions. Temporarily rename it, then scan the Field Definitions. It will stand out like a sore thumb in other fields' calculations.

Custom Separators
To really use the Custom Order method you should create custom labels for each of your carefully grouped fields, to create a "table of contents" look for the Field Definitions list. It's pretty easy. The quickest way is to just create a new global field. Use a few underline characters, then a descriptive name, eg. "_____Service", up to about 17 characters long. Drag it to be above your Invoice fields.

[Warning: the next two techniques are geeky.]

"False" Calculations
To get even more info in, you can create a "false" calculation field. Give it the name as above, then write more in the calculation options.
_____Service, Calculation= If (1=0, "Service Rates and a bunch of stuff", "")
The entire above sentence will display in Field Definitions.
This same trick can be used within real calculations, to remind you what they do.

For example:
Case (1, If(Max(DONATE::Date) = Date, Donation, ""), "Date is the latest")
In this "case" the first argument, Case(1) is always true, so the last comment is never evaluated. (Thanks to Steve Cassidy for this exercise in minimalism.)

Disabled Auto-Enter Options
Of course, the above could only be used in a calculation field. What about regular text and number fields? Well, believe it or not, someone (not me) has discovered a way. Click Options, then enter your descriptive text in the Auto-Enter Data box. Then uncheck the box. Your text will be grayed out, but will remain stored and readable. You just have to remember to look there for it.

Relationships

Now that we've got our fields under control, let's move on to relationships. When I was beginning, I tried to name relationships with names that described what they did, their function in the file. But I would forget what the necessarily short name

meant. And then, to make things worse, some relationships were used for several different operations, or a similarly named relationship went to a different file.

So now I name the relationships according to where they go (files) and what fields are involved, trusting that I can figure out what it might do by looking at the context of how it is used…and by reading my Comments in the scripts.

Relationship Names
The first part is the name of file, or part of it. The second part is at least one of the fields. Often relationships are between two identically named fields anyway. So a relationship between a client (Contacts file) and one of his jobs (Jobs file) would look like: from the Contacts file, "Job_Client ID", and from the Job file, "Contacts_Client ID".

Rather than name relationships within the same file with the file name, I prefix them with "Self", as in "Self-Client ID" (Contacts file), or "Self-Job ID" (Jobs file); whatever the field involved is named.

One of the great advantages of naming clearly like this is so you can pick the right one when choosing a relationship in the drop-down box for a field, in either a field definition, or in a calculation dialog. If you still can't tell what a relationship does, go ahead and choose it, then click again, drop down to the bottom of the list box, to "Define Relationships," and let go. The relationship dialog will open, already scrolled to that one, so you can see which fields are actually involved. If it's what you wanted, just close the dialog. (An excellent piece of software engineering.)

If you still can't tell, it may require a trip to the Field Definitions and a study of the field(s) involved. Try to remember where you are before going that route, as you'll need to get back.

Relationship Order
When you're in the relationship dialog box, you may as well tidy up that list of relationships. If you group them according to which file they go to (easy to do now), then group them roughly according to what they do and relative importance, then it's a lot easier to find them in that list. Just drag them into place.
Remember to click on the one you wanted if you were in the middle of doing something, so it remains selected in the relationship drop-down box.

Scripts

Now we get into the really fun stuff. Carefully naming scripts and documenting

script steps is the only way to preserve your sanity. The problem is not too awful at first, but wait a little while, then come back and try to modify a complex script. You won't remember exactly what's happening, much less the unforeseen effects on other scripts—and I don't think it's only me that has this problem.

Fortunately FileMaker has a few ways to help. The first is to use a system to name your scripts. The names can be pretty long, and there aren't really any restrictions on characters you can use, so there's less need to be cryptic.

Script Groups

First I'll tell you how to "group" scripts, so you can find them on the list in the ScriptMaker dialog, and how to create a label with a script name. The first simple scripts that deal with such universal things as Sorting and navigation to Layouts I generally put at the top of the list. Scripts and buttons that are only called from one particular Layout, but do different things, will be grouped together. If several scripts operate within one particular area or function of the database, then I group them together.

Above each of these groups is an easy-to-see label. The label is basically just an empty script; just delete all the steps. Name the scripts with a few dashes or underlines then the group name.
"---------- Navigation" would be scripts that just deal with moving around the file.
"----------> External", scripts that go to, or call scripts in other files.
"----------< External", scripts that get called from other files.
"---------- Printing", "---------- Help Messages", etc..

The dashes set the names off to the right of the regular script names, so they are more visible. Drag the label above the groups. Duplicate one to make another. Often a single dash is used as the name of an empty script to create a separation line in the Scripts Menu list visible to the user. Just click the "Show in Menu" checkbox for those. Don't check it for these other ones.

Script Names

As you may have noticed above, I use arrows to show that there is an external script within a script, either going out or coming in. These can be tough to troubleshoot, so name them all.

To make them easy to see, I use a long arrow, "-->", or "<--" for external scripts (sometimes "<-->" for one that goes then returns). I also include at least one of the file names of the external file(s). Eg., -->JobEntry, Job Choice Lay

This tells me that this script just goes to the JobEntry file, to the Job Choice layout.

If a script calls another script within its own file, I just use a single arrow after it. I try to put the script it calls directly under it, with an arrow in front.

New Record>
>Empty Fields Check

This tells me that the New Record script runs a custom scripted check for empty fields before allowing me to create a new record. Another way to write the first, if several scattered scripts called the same subscript (very likely), would be, New Record >w/Empty Check. Another script might be, Main Menu >w/Empty Check.

This makes your life much easier if you ever modify a script. You can see that there will be effects on other scripts. It's especially useful when you are checking scripts between files, when the name is all that you can see easily from the other file.

Comment Step
There is a built-in mechanism to document within scripts. It is the Comment [""] step, down near the bottom of the Script Steps list. Use it liberally; you'll be glad later. It can be put immediately after steps which are otherwise difficult to decipher, complex Sorts, Finds, and Perform Script [External].

External Scripts
This last is especially important to comment after. The step only shows the name of the file, not the name of the external script. You can see which script it is by clicking on the file name box and letting go, triggering the dialog box; but this gets old in a hurry when you're trying to track down a problem.
Put the external script's name in the Comment, and you'll be able to see in a glance what is going on. Be sure to keep it updated if things change.

By the way, if the script you see in the dialog doesn't have its "<--" arrow, then you have either neglected to name it properly, or you're calling the wrong script. This is especially important in simple scripts that have slight variations but similar names. Sometimes the externally called one has only a single difference (such as a Halt Script step); unless it's labelled you can't pick it easily from the other file.

Troubleshooting Scripts
There are a couple more steps you can add to help find out where a script is screwing up (also must happen to other people). The first one I turn to is the

Pause/Resume Script [] step. Put one of these before each critical juncture in a complex script. If the problem is in fields not being set properly, you should have a demo layout that has all of the fields involved. Add a step to go to this layout if needed. Then you can walk through the script, watching what happens after each big step, hitting the Enter key to continue. Remember to take the Pauses out when you're done.

Another quick way to "hear" what's going on is to add a Beep step. It's especially useful to see "if" something is happening at all. Add two beeps or three beeps to check other possible branches of a conditional script.

If you think that a step (or series of steps) are the problem, and you would like to run the script without them (assuming it could), there is a way to "comment them out."

Put a false "If" step before them, and an "End If" after.
If [1=0]
    Script step
End If
The rest of the script to run.

Obsolete Scripts
A combination of techniques can help identify unused scripts before taking the drastic step of deleting them. In order of intensity:
1. Name them with a prefix like "###". You'll be able to spot them more easily if they are called by other scripts. Check for it in suspected calling External scripts, too.
2. Put a Beep in. If you have no other beeps, this will alert you that a marked script is still being used.
3. Display a custom message, "Script name is marked for deletion".
4. Create a global field, text, "ScriptName g", and add the step:
Set Field ["ScriptName g", Status(CurrentScriptName)]
This will capture the name of the script so you can see it afterwards.

If all fails and you're still not sure whether it's being used, leave it in. The possible damage from removing it is worse than the small disk space being wasted by leaving it in. Speaking of wasting space, I'll see you next month.

•Fenton Jones• <manavesh@mymac.com>

<http://www.smalldog.com>