FileMaker® Pro Tune Up™

Study Guide

Released 07/05/01

208 Stone Ave., Suite 3

Lexington, KY 40508

(859) 225 1572

a2soft@aol.com

Technical Solutions for Human Needs™

Introduction

NOTE: This document originally written in 1997 as a handout for a classroom setting. FileMaker Pro has since advanced by two or three major revisions, but the guidelines and examples presented in this document **ARE STILL ACCURATE**.

Welcome to our class. This class is designed to help non programmers take advantage of the key advanced features of FileMaker Pro 3. Since its introduction in 1985, FileMaker Pro has been an easy to use tool for managing name lists and printing forms. However, with the introduction of FileMaker Pro version 3, FileMaker Pro has also become a powerful programming environment for professional developers.

In half a day, we can not master all the tools available in FileMaker Pro version 3, but we can unlock the fundamental powers of version 3 to dramatically improve most database files. In this course we will study the three fundamental differences between databases created in FileMaker Pro versions 2 and 3, and how to best apply them. These are:

- Relational data handling and database design.
- Using merge fields in display and print layouts.
- "Programming" FileMaker Pro with new status and error scripts.

Not all of the classroom material is covered in this study guide. Portions of the guide have been left open ended to allow for a variety of classroom discussion.

What's New in FileMaker® Pro 3.0

There are literally hundreds of individual new features in the latest version of FileMaker Pro. We will not have time during our half day course to mention all of them. Some of the most significant include:

- Ability to store, display, and calculate relational data.
- Merge fields.
- Nearly doubled selection of script commands, including conditional and status statements.
- 32 bit native in Windows 95, support for long filenames, and OLE.
- Support for other networks than AppleTalk, such as TCP/IP and IPX.
- Global fields.
- Unstored calculations.
- Complex auto enter and validation of fields.
- Complex and dynamic value lists.
- Improved layout editing tools.

Things to Do Before Coming to Class

Our class will move quickly, so please take some time to review basic FileMaker Pro skills before the class day. We will expect you to be able to perform the following:

1) Know how to create a new database file.
2) Know how to define a text, date, number, or basic calculation field.
3) Know how to create a simple layout, add or remove layout “parts”, draw basic graphics and text, and add or remove fields.
4) Know how to create a basic script.
5) Know how to “find” records, and what is meant by the “found set”.
6) Read through this study guide and make a list of questions you may have.

In addition, we would like you to take a look at, but not necessarily understand:

1) The “Define Relationships…” choice in the FILE menu.
2) The “Paste Special > Merge Field” choice in the EDIT menu while in layout mode.

Class Program

Hour 1: Relational Structures

What is Relational Data?

A relational data structure splits a large collection of data into many smaller linked data sets. This creates smaller files for the computer to work with, thus speeding up operations. It also eliminates duplicate information, conserving disk space and memory.

Database files created in FileMaker Pro are not automatically relational. You must actively design your database files to be relational. In order to implement the relational functions of FileMaker Pro, two things *must* be done:

1) You must define one or more relationships between files.
2) You must plan your database as two or more logically related files.

A few things to remember about FileMaker Pro version 2 databases that you may have converted to version 3:

- Multiple files which perform lookups on each other ARE NOT a relational database.
- When FileMaker Pro version 3 converts your version 2 files, it defines relationships to maintain the functional lookups. *This does not make your database files relational.*

Sidetrack: Self Relations

You may occasionally hear the term “self relation” in advanced database design. A self relation is not a relational data structure, but is a programmer’s method of manipulating a single flat file using relational tools. This advanced method is beyond the scope of this class.

The Common Flat File Database

A flat file database is the most common and easily understood form of database. They are called “flat file” because when printed out the file looks like a flat wall chart. A list of names and addresses in a spreadsheet is an example of a flat file database. A HyperCard stack is a flat file database. A list of names and sales in a word processor is a flat file database. All FileMaker Pro databases version 2 or earlier are flat file databases. In a flat file database, *all* the information necessary for a display or printout is stored in a single file. Any additional files serve only to store templates or references for speeding data entry in new records.

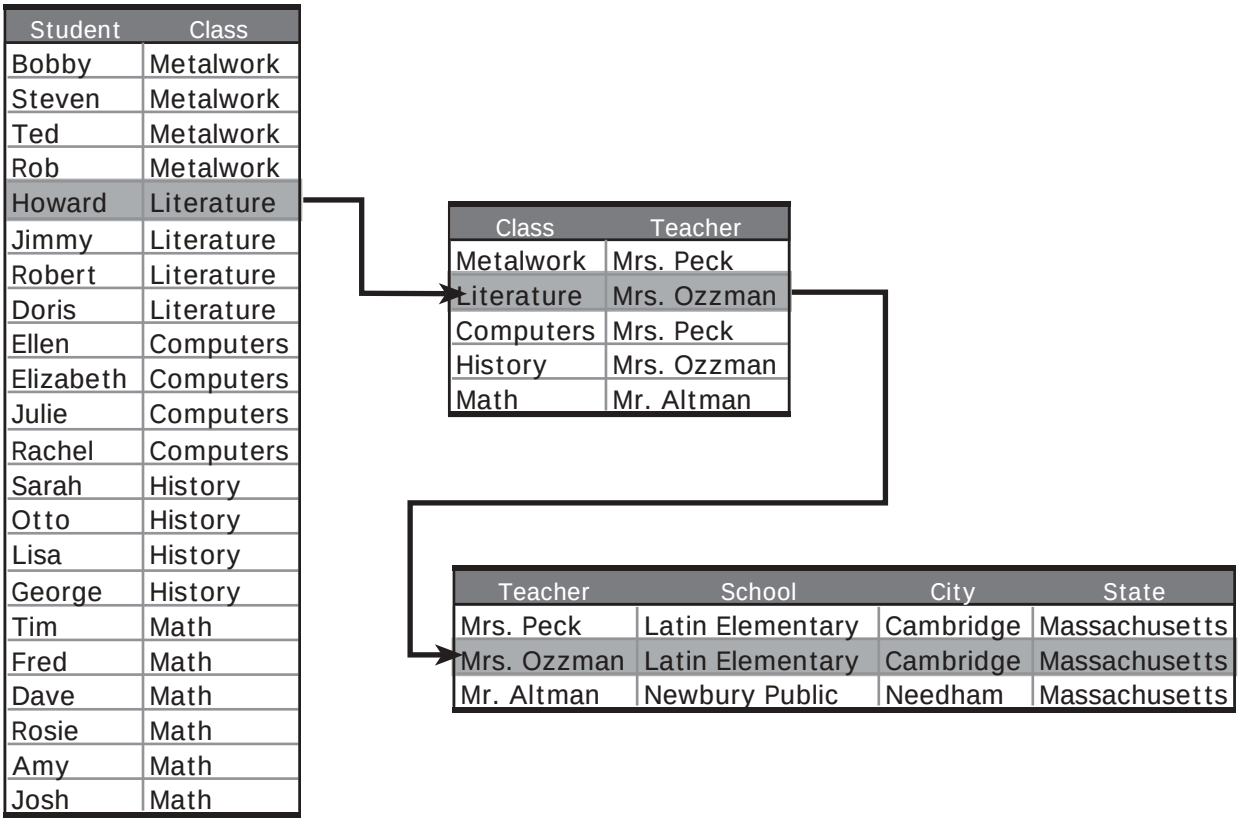
The simplicity of the flat file database limits its power. Storing all necessary information in one file makes the file very large, and this large file requires more RAM and CPU power to manipulate. Below is a visual display of a flat file database.

Student	Class	Teacher	School	City	State
Bobby	Metalwork	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Steven	Metalwork	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Ted	Metalwork	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Rob	Metalwork	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Howard	Literature	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Jimmy	Literature	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Robert	Literature	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Doris	Literature	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Ellen	Computers	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Elizabeth	Computers	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Julie	Computers	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Rachel	Computers	Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Sarah	History	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Otto	History	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Lisa	History	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
George	History	Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Tim	Math	Mr. Altman	Newbury Public	Needham	Massachusetts
Fred	Math	Mr. Altman	Newbury Public	Needham	Massachusetts
Dave	Math	Mr. Altman	Newbury Public	Needham	Massachusetts
Rosie	Math	Mr. Altman	Newbury Public	Needham	Massachusetts
Amy	Math	Mr. Altman	Newbury Public	Needham	Massachusetts
Josh	Math	Mr. Altman	Newbury Public	Needham	Massachusetts

A Flat File Database: the flat file above contains 132 cells or pieces of information.

Making the Flat File Relational

In a world of limited RAM and computer power, we need to find a way to minimize file size. We can do this by dividing the flat file into logical groups and dynamically linking these groups, as below.



Student	Class
Bobby	Metalwork
Steven	Metalwork
Ted	Metalwork
Rob	Metalwork
Howard	Literature
Jimmy	Literature
Robert	Literature
Doris	Literature
Ellen	Computers
Elizabeth	Computers
Julie	Computers
Rachel	Computers
Sarah	History
Otto	History
Lisa	History
George	History
Tim	Math
Fred	Math
Dave	Math
Rosie	Math
Amy	Math
Josh	Math

Class	Teacher
Metalwork	Mrs. Peck
Literature	Mrs. Ozzman
Computers	Mrs. Peck
History	Mrs. Ozzman
Math	Mr. Altman

Teacher	School	City	State
Mrs. Peck	Latin Elementary	Cambridge	Massachusetts
Mrs. Ozzman	Latin Elementary	Cambridge	Massachusetts
Mr. Altman	Newbury Public	Needham	Massachusetts

The Relational Version: in contrast, the three relational files above contain half as many (66) cells, but still contain all the information.

Designing the Relational Database

To create a relational database, it is extremely important to map out your data on paper similar to the illustration above before trying to create the database on the computer. In class, we will try mapping out a relational database on paper.

Hour 2: Merge Fields and Scripts

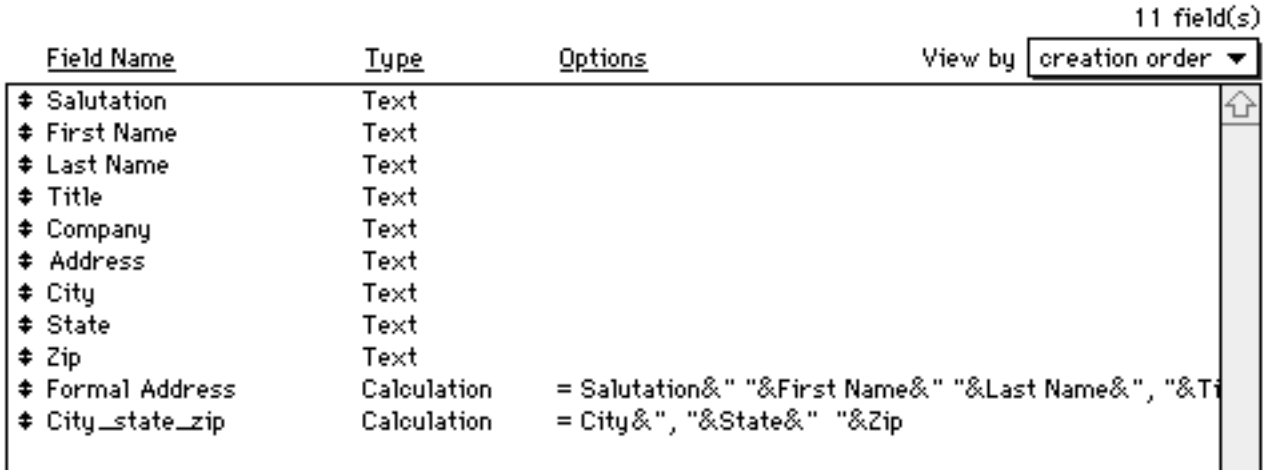
The Merge Field

We love merge fields! The merge field is a powerful way to display a field value without actually placing a field on your layout. Once you learn how to place a merge field on a layout, a huge variety of applications will appear to you. Some things you can do with merge fields:

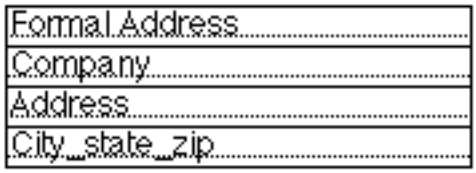
- Seamlessly mix field values and fixed text in both display and print layouts.
- Display a field value while preventing anyone from editing that field value.
- Replace a calculated text field used only for display and printing.

What's It For?

We have all been there. You want to print or display the full name and address, but when you place the fields on the layout, you can't get that "natural" look. You can use "sliding" to make the fields pull together on a printed report, but you can't get the commas and other punctuation between fields. So you create some calculated fields that format the text for you, as below:



You then format a layout with the address as below:

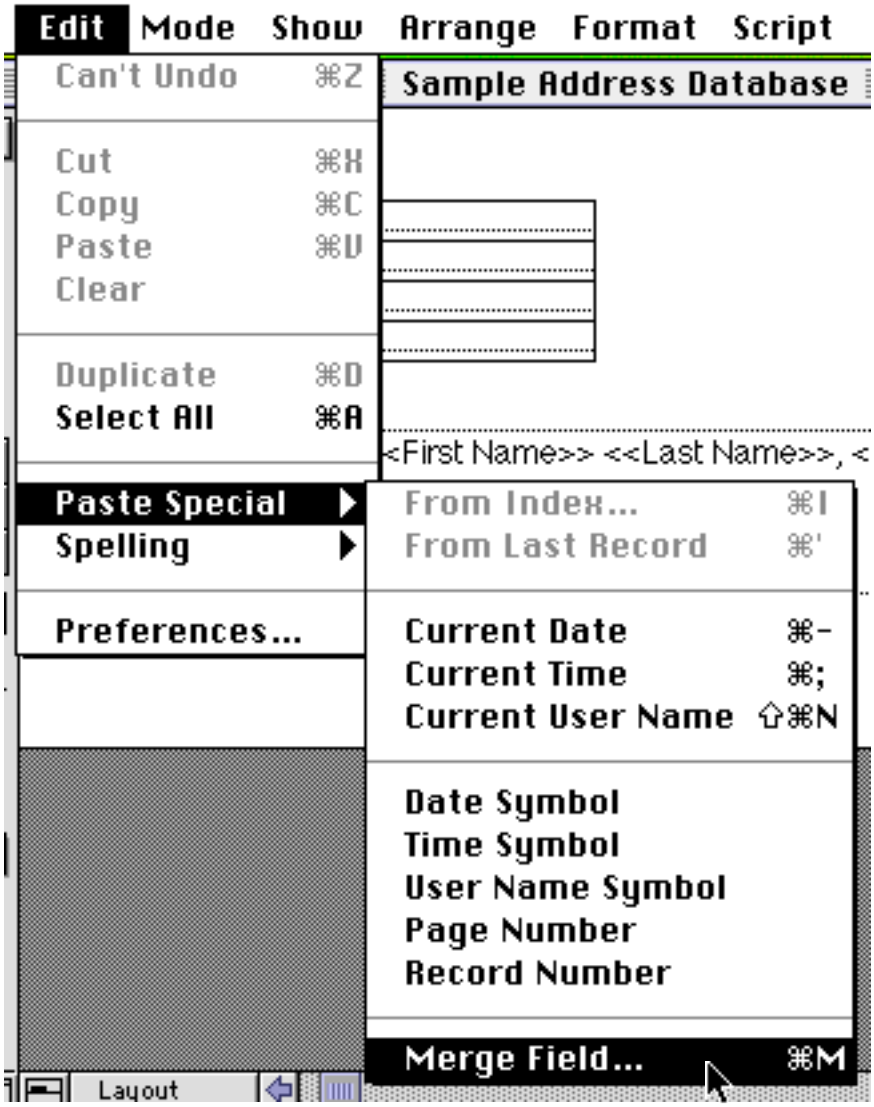


Address Formatted with Calculations: this is how an address formatted with calculation fields appears in layout mode.

Of course, the problem is that after you do this a few times, you introduce lots of calculations for your database to perform, the database file grows, and performance slows.

Try a Merge Field

FileMaker Pro 3 offers a solution. While in layout mode, select "Merge Field…" as shown below:



How to Paste a Merge Field: you may also type in the merge field by typing the field name bracketed by double arrows, thus "<<field name>>".

<<Salutation>> <<First Name>> <<Last Name>>, <<Title>>
<<Company>>
<<Address>>
<<City>>, <<State>> <<Zip>>

Address Formatted with Merge Fields: this is how an address formatted with merge fields appears in layout mode.

Using calculations	Mr. Steven Harrod, President Applied Arts Ltd. P.O. Box 54 Needham, MA 02192
Using merge fields	Mr. Steven Harrod, President Applied Arts Ltd. P.O. Box 54 Needham, MA 02192

The Results: as you can see, both methods give the same result, but the merge fields eliminate two calculation fields.

Form Letters

A similar problem exists in form letters, but you can't "fake" these with calculations, because the text of the form letter may exceed the maximum allowable calculation length (32,767 characters), and it really isn't practical anyway. A form letter formatted with calculations makes a huge data field, and that slows down the database. Without merge fields, your layout and results look like this:

Dear Salutation Last Name ,

We are proud to announce that you, First Name Last Name are our grand prize winner! Please call 800 570 2787 immediately to collect your prize.

Dear Mr. Harrod ,

We are proud to announce that you, Steven Harrod are our grand prize winner! Please call 800 570 2787 immediately to collect your prize.

Looks Pretty Awful: extreme example, but you get the point. Top, layout mode; bottom, as printed.

So let's redo this with merge fields:

Dear <<Salutation>> <<Last Name>>,

We are proud to announce that you, <<First Name>> <<Last Name>>, are our grand prize winner! Please call 800 570 2787 immediately to collect your prize.

Dear Mr. Harrod,

We are proud to announce that you, Steven Harrod, are our grand prize winner! Please call 800 570 2787 immediately to collect your prize.

Much Better: text reflows cleanly and without gaps.

Coffee Break!

New Scripting Features "Status" and "Control"

Before FileMaker Pro 3, scripts were basically "dumb". A script started, did its thing, and was over. The script

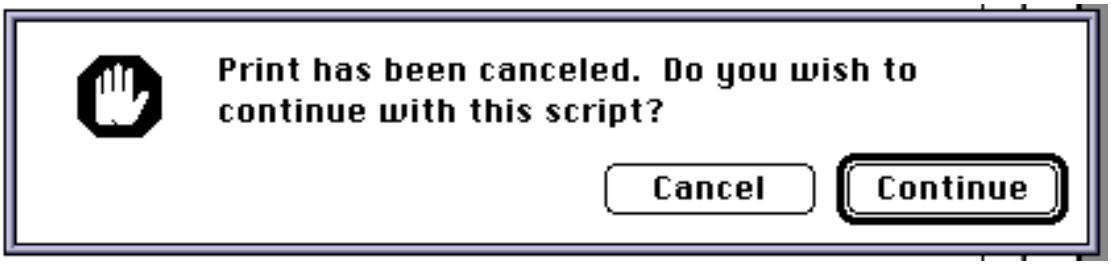
could not change what it did based on the current situation. If an error occurred, some bland and generic FileMaker Pro error message would appear with the choice of canceling or continuing, and the user probably tossed a coin to choose between the two.

All of that has changed for the better. “IF-ELSE-END IF” subroutines and “LOOP” routines are now available in the script library. These two control features make it possible for scripts to be intelligent and modify their behavior based on data and current conditions. “Status” functions are implemented as calculation functions, but have their greatest impact on IF subroutines. Together, these features allow us to:

- Intercept FileMaker Pro error messages and either respond to them or replace them.
- Create interactive scripts that can give users choices.
- Use FileMaker Pro as a programming environment.

Customizing Error messages

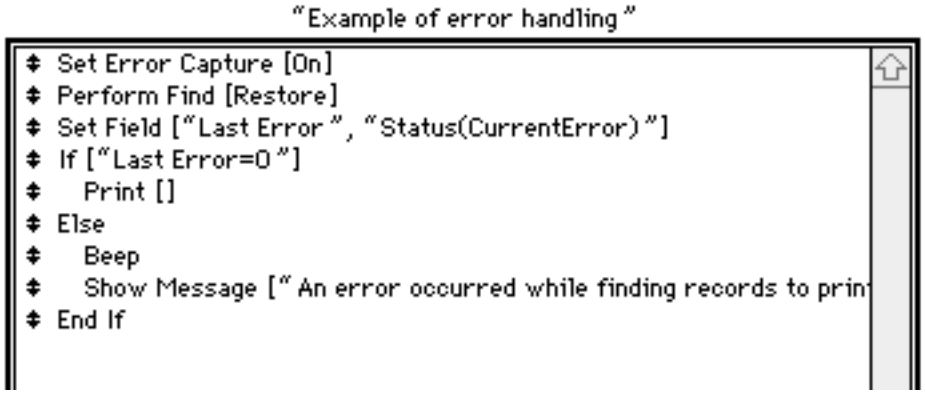
The default error messages in scripts do not tell a user what has gone wrong.



Default Error Message: this standard script error message confuses users and rarely provides the right answer.

With new control and status functions in FileMaker Pro 3, we can intercept errors and either show a message or perform another action. To put a customized error handler into a script, we need three things:

1) A “Set Error Capture” step at the beginning of the script.
2) A “set field” step at the point(s) in the script we are testing, to store the error value in a global field.
3) An “IF” subroutine to test the error value and react to it.



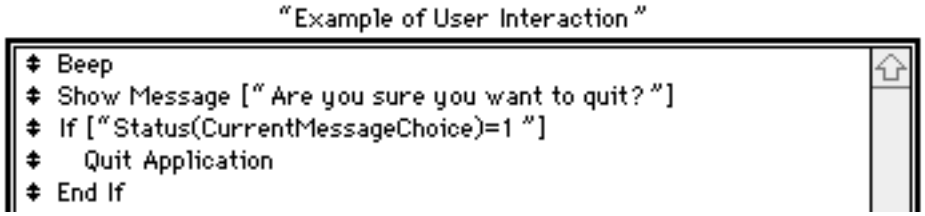
Simple Error Handling: this is the simplest example of error handling. Right after the FIND step, the error code is stored in the global field “Last Error”. Zero means no error occurred. The if statement tests the value of “Last Error” and makes a decision whether to print or not. More complex error checking can be achieved by adding multiple IF and ELSE statements. The

global field is necessary because the value of “status(current error)” is only accurate IMMEDIATELY after the action you are testing.

A complete list of error codes is available from the online help section of FileMaker Pro 3 under the definition of “Status(CurrentError)”.

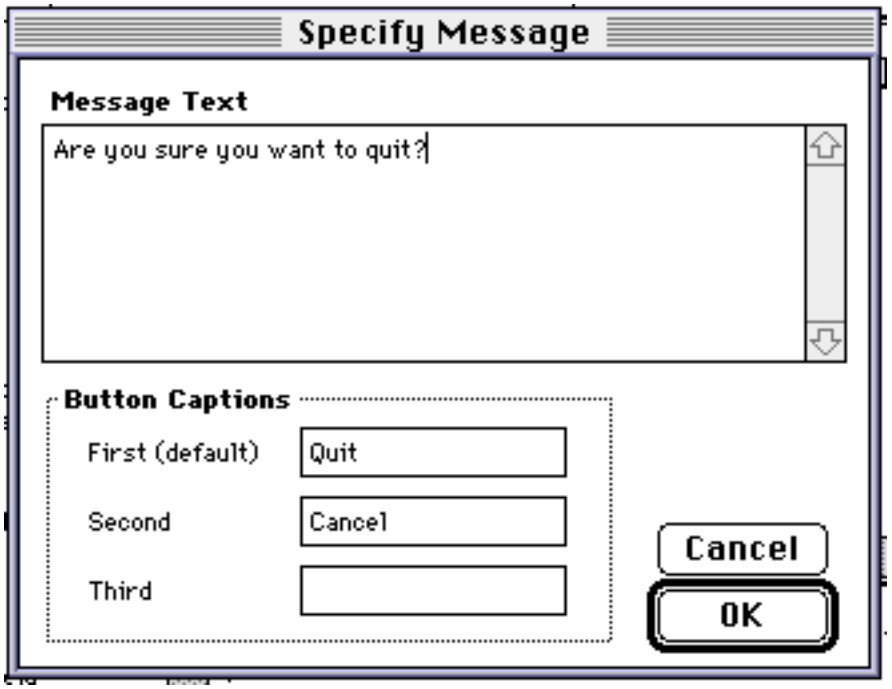
Giving Users Choices

It sure is a lot nicer to ask your users permission before charging off and doing something like deleting hundreds of files or starting a big print job. With FileMaker Pro’s “Show Message” script, it is very easy.



Simple User Choice: this is the simplest example of responding to a script message.

Immediately after a message is displayed, the number corresponding to the button pressed by the user may be retrieved by the function “Status(CurrentMessageChoice)”. If you want to check this value more than once (for example, in a nested IF subroutine), you also need to store the value in a global field, as we did previously for error handling.



Show Message: regardless of what you type into the button captions, the “Status(CurrentMessageChoice) function always returns “1” for the first button, “2” for the second, and “3” for the last.

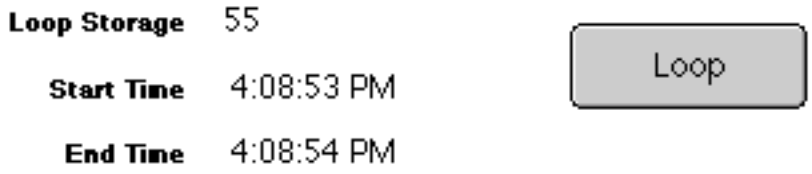
Programming

Almost any logical program you can write in Basic or Pascal can be written in FileMaker Pro 3 script.

"Example of Programming"

```
Set Field ["Loop Storage", "0"]
Clear [Select, "End Time"]
Set Field ["Start Time", "Status(CurrentTime)"]
Loop
   Exit Loop If ["Status(CurrentTime)=Start Time+1"]
   Set Field ["Loop Storage", "Loop Storage+1"]
End Loop
Set Field ["End Time", "Status(CurrentTime)"]
Exit Record/Request
```

Simple Loop Calculation: this loop shows a basic iterative loop.



Program Results: when the button is pressed, the script starts counting at 1 and keeps counting until one second has passed. The results vary by the split millisecond at which the button is pressed and by how busy the computer processor is.

Hour 3: Speed Issues

The only operations that should be time consuming on a FileMaker Pro database, running on a Pentium or PowerPC computer, are:

- Sorting.
- Indexing a field (which only needs to happen once).
- Complex replace operations.
- Importing or exporting large numbers of records (a thousand or more).
- Complex scripts that step through every record in the database.

Finds, screen displays, printing, and data entry should all be quick and reliable. If your database has you going on coffee breaks on a regular basis, chances are it is not FileMaker Pro's fault.

FileMaker Pro databases grow old like houses, they get patched over and extended over time until new lifestyles and new technology demand they be gutted and renovated. Like an old house, the traffic patterns and demands placed on FileMaker Pro databases overwhelm the old design and life becomes cramped and compromised. Common causes of slow performance in FileMaker Pro usually result from patched database files left over from previous versions of FileMaker Pro, and include:

- Poor or no implementation of relational data structure.
- Too many calculation fields.
- Poorly placed or designed calculation fields.

Although network traffic and low computer power can slow FileMaker Pro databases, it is our experience that this is an easy excuse, and blamed much too often.

We have already covered relational data structures and how to eliminate calculation fields with merge fields. The last item, design and placement of calculation fields, requires an understanding of stored and unstored calculation fields.

Stored vs. Unstored Calculation Fields

In FileMaker Pro version 2 or earlier, all calculations were stored; that is, each calculated result was actually written to the database file, and the file increased in size accordingly. New in FileMaker Pro version 3, calculations may be “unstored”, which means they are not written to disk, but only calculated when displayed.

	Stored	**Unstored**
When Calculated	Calculated when values changed	Calculated when displayed on screen or used.
Increases File Size?	Yes	No
Limits	Can not reference a related, global, or summary field, or another unstored calculation.	None
Use in Relationships?	Yes	No
Indexable for Finds?	Yes	No

So, with reference to speed, the stored calculation is usually faster. This is because the correct value already exists in the file, whereas an unstored calculation must be recalculated every time it is used, regardless of whether the result has changed or not.

For example, if you perform a find on a calculated field, the find will proceed much faster if the calculation is stored and indexed. If the calculation is not stored, you will have to wait through a long progress bar while the results are calculated and indexed temporarily before performing the search.

In a complex set of related files, some calculations refer to data in the related files. If you have a layout with many records in a list, these calculations must be calculated for every visible record. The time delay while the calculation looks for data in related files will cause a noticeable slowdown in the screen display.

Conclusion

We hope this class has helped you unlock the potential of your FileMaker Pro database. FileMaker Pro is quickly becoming the professional developer's tool of choice. Its powerful features combined with a compact, easy to understand design interface makes it the most economical investment of valuable labor hours of any database system.

Remember, when designing your FileMaker Pro database for best performance:

- Maintain a clean, organized user interface.
- Use relational data structures to have the smallest data files possible.
- Use merge fields instead of calculation fields.
- Use stored or unstored calculations as appropriate, and be careful where you display them.
