Conditional Headers and Footers

by John Mark Osborne
www.databasepros.com

A tremendous amount of reporting power has been packed into FileMaker Pro with an incredibly easy-to-use interface. The power and simplicity behind adding and deleting parts to organize your pages almost makes it too easy to create a sophisticated printed report. Unfortunately, there is no way to attach a calculation formula to a part much like you can with Access Privileges to create conditional records. Either a part exists or it doesn't. While it is possible to slide through most parts, this doesn't really help unless you can create a field that is conditional. With sliding turned on, the problem becomes even more complex because you never know how many records will appear on each page. Creating a calculation that adjusts itself dynamically to the number of records is crucial to this technique.

Dynamic programming is at the core of the Philosophy of FileMaker. Within reason, it is wiser to design FileMaker with the assumption that fields, layouts, groups and other pieces of the FileMaker puzzle will be added at a later time. If your solution can adjust to the newly added objects, you don't have to remind yourself how a script, calculation or technique works to program in the new object. It just works! Your clients will be much happier because you don't have to bill them for additional work and you can move onto exciting new projects instead of spending all your time maintaining a system.

A Good Example

This is a tough technique to grasp unless you've worked with a lot of subsummary reports. It's not complex because there are long formulas. Rather, the technique is complex because you need to have a good understanding how FileMaker works with records inside of parts. In order to make this technique easier to understand, let's use an example that should be familiar to everyone reading this article. Let's assume you have a standard invoicing solution with an invoices, products and line items files. When printing an invoice, you don't print from the invoices file because portals weren't designed by FileMaker, Inc. to be printed. Portals don't slide in all scenarios and don't page break gracefully. Instead, you use the Go to Related Record script step to locate all the related records in the line items file and print from the line items file. As a general rule, it is almost always best to print from the child file rather than the parent if multiple related records need to be printed.

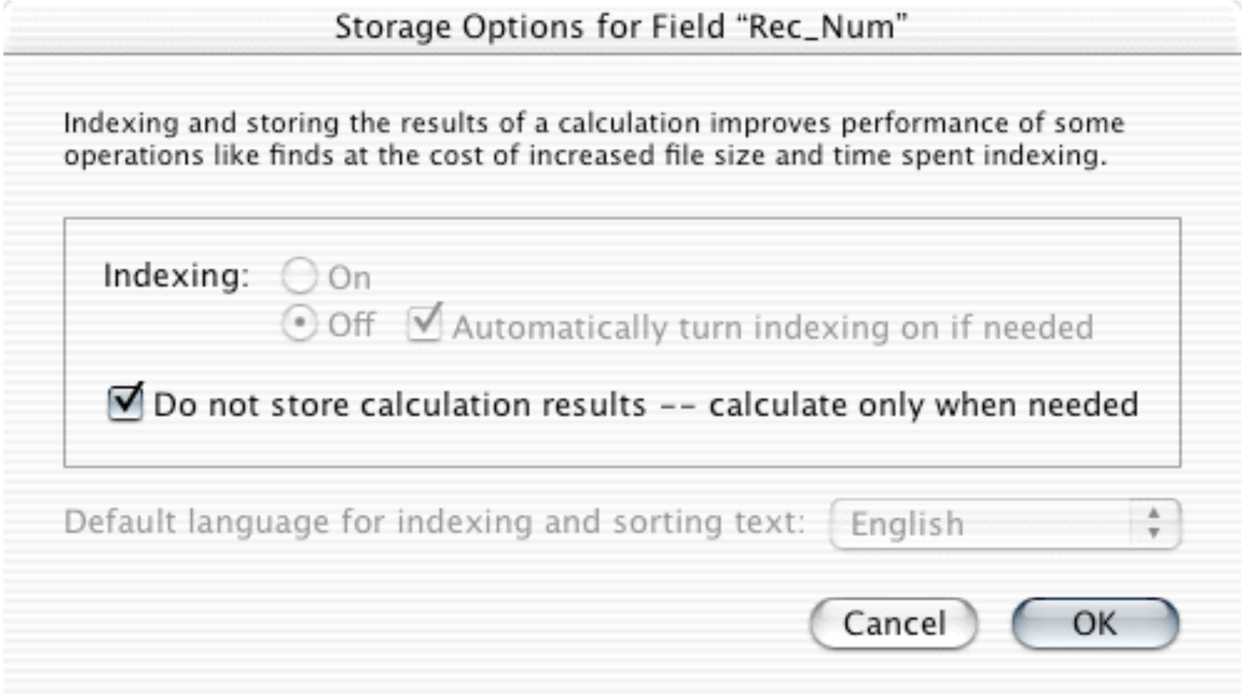
Reverse relationships bring in any data not contained in the line items file. Therefore, it is important to make sure the line items file contains the key field from any file you want to borrow data. In this example, the invoice ID is contained on every record in the line items file because a portal was used to enter the products. All you have to do is create a relationship to the invoice file originating from the line items file using the invoice ID as the key.

This all works flawlessly until you have a two page invoice. You could place the invoice total at the end of the second page using a trailing grand summary part. This works great as long as you want the total to appear after the last record which could place the total in the middle of the page. Many systems place the totals at the bottom of the last page so they are easy to find in a consistent location. All of a sudden, the problem is not so easy for FileMaker to handle!

Conditioning a Footer

The first calculation you'll need is Rec_Num in the line items file. It is a simple formula that returns the Status(CurrentRecordNumber). Make sure it is set to unstored in storage options or it will not update properly (see Figure 1). This is true for any Status or Design function that is placed into a Define Fields calculation. Status functions were designed for scripts and only update when called. If a Define Fields calculation only contains a reference to a Status function, it will never update beyond the status available at the time you created the calculation field or when the record was created. If you make the calculation field unstored, the calculation will recalculate every time the screen refreshes. The screen refreshes every time the database changes records, layouts, files or modes. Otherwise, the only other time a Define Fields calculation field containing a Status function will update is if a field referenced in the formula is modified. While this may work in some situations, most scenarios require the result to update as the status changes.

Figure 1: Status(CurrentRecordNumber) - Make sure the Rec_Num calculation field is set to be unstored or it won't update properly.



Storage Options for Field “Rec_Num”

Indexing and storing the results of a calculation improves performance of some operations like finds at the cost of increased file size and time spent indexing.

Indexing: ○ On
● Off ☑ Automatically turn indexing on if needed

☑ Do not store calculation results -- calculate only when needed

Default language for indexing and sorting text: English

There is only one Status function that doesn't need to be unstored. That function is Status(CurrentRecordID). The Record ID is defined when a record is created and can't change. Therefore, there is no need to update the calculation result. There are also several Status functions that will never update properly even when they are set to unstored. One example is Status(CurrentFieldName). You might want to create a user interface feature that displays the selected field in a calculation so the user knows what to type without looking where the cursor currently is. Unfortunately, tabbing from one field to another is not enough to refresh the window and the calculation will not update.

Next, create a summary field in the line items file titled Max_Num that returns the maximum of the Rec_Num field. Comparing the Rec_Num and Max_Num fields enables other calculations to determine the last page of the report and display a conditional result. The following calculation shows or hides the result depending on the currently selected page. The formula is a simple Case statement comparing the Max_Num and Rec_Num fields.

```
Case(

Max_Num = Rec_Num,

Total,

TextToNum("")
```

)

The Total field referenced in the formula is a simple summary field that totals the extended price. The TextToNum function is used to prevent a zero from appearing as the null result for the formula. The zero null result can be hidden via the Number Format dialog but this technique is better so you don't have to format the field on every layout where it is needed. In addition, this technique can be used with Date and Time results in order to prevent a question mark result. Just use DateToText("") or TimeToText("").

How it Works

At first, the Total [Cond] formula is very short and seems very easy to understand! The difficult part is understanding what values are being compared. This requires a good understanding of how records are handled in relation to the different parts. Simply put, the footer part displays values from the last record on the page. When the Max_Num and Rec_Num field are compared, they only equal each other on the last record. Since the footer part displays the last record on the page, only the last page of the report displays a value for the Total [Cond] formula. It's really that simple.

A More Complex Example

Subsummary reports in FileMaker are sophisticated yet easy to implement. However, there are a few features that would be nice to add. A common occurrence in a report is for the records within a single subsummary category to break across a page. Unfortunately, a subsummary part has no option for repeating the fields or objects within it when the records it subsummarizes span multiple pages. The result is a listing of records without a subsummary title.

One solution is to add a header part that replicates the information found in the subsummary. Unfortunately, this creates another problem. If the header and subsummary parts both display at the top of the same page, you get the same information twice. This solution gets around the problem by creating a smart header. Instead of just placing the data or fields from the subsummary part in the header part, conditional calculations are utilized. These calculations determine whether the record at the top of the current page is a continuation from the previous subsummary category or the beginning of a new subsummary part.

What you Need

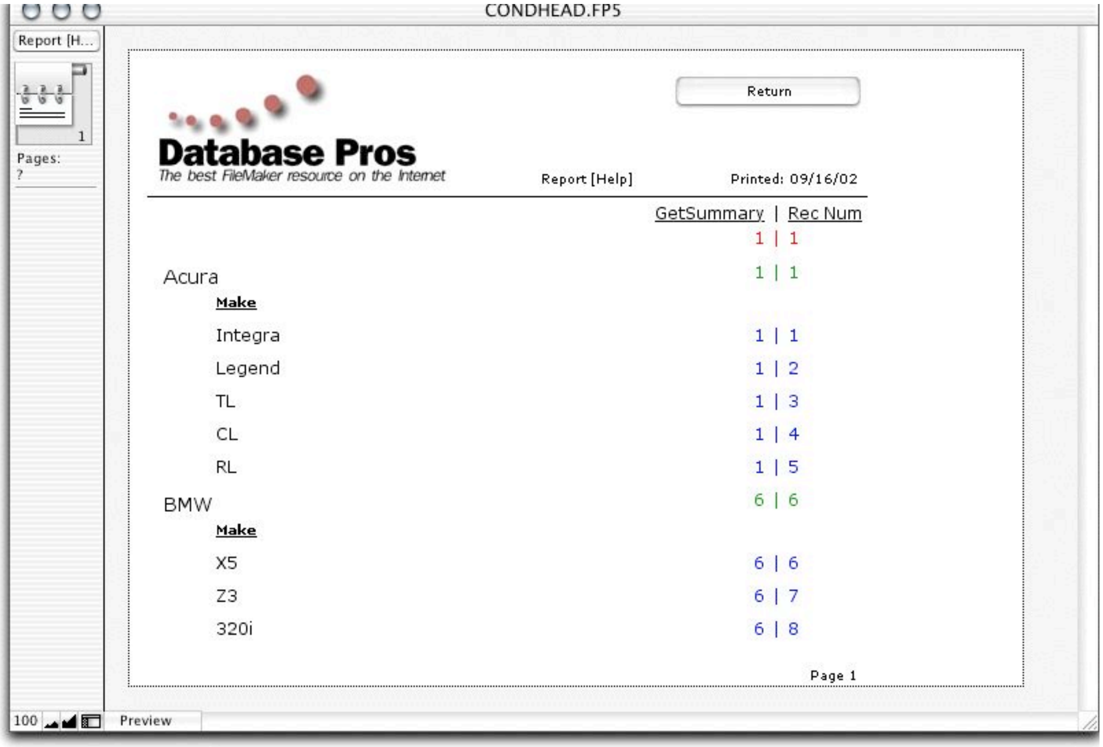
The first calculation you will need is the same as the previous example: Rec_Num. It is a simple formula that calculates the Status(CurrentRecordNumber). Make sure it is set to unstored in storage options. Next, create a summary field titled Min_Num that returns the minimum of the Rec_Num field. These two fields will be incorporated into each calculation field that replicates the data from the subsummary part.

The conditional calculation below shows or hides the result depending on whether the subsummary part is a continuation from the previous page. This formula comes from an example that summarizes makes of cars by their manufacturer. Here is the formula for Manufacturer_Header:

```
Case(

GetSummary(Min_Num, Manufacturer) <>
Status(CurrentRecordNumber),

Manufacturer

)
```

The GetSummary function is like a subsummary report but in a calculation field. It uses a summary field and a break/sort field to summarize values for a group of records within the current found set. Let's look at the values displayed by a report on car manufacturers (see Figure 2 and 3) to better understand how the formula creates a smart header. Start by studying the GetSummary and Rec_Num fields down the right side of the report.

Figure 2: First Page of the Report - Notice that the GetSummary and Rec_Num fields in red are the same.



Database Pros
The best FileMaker resource on the Internet

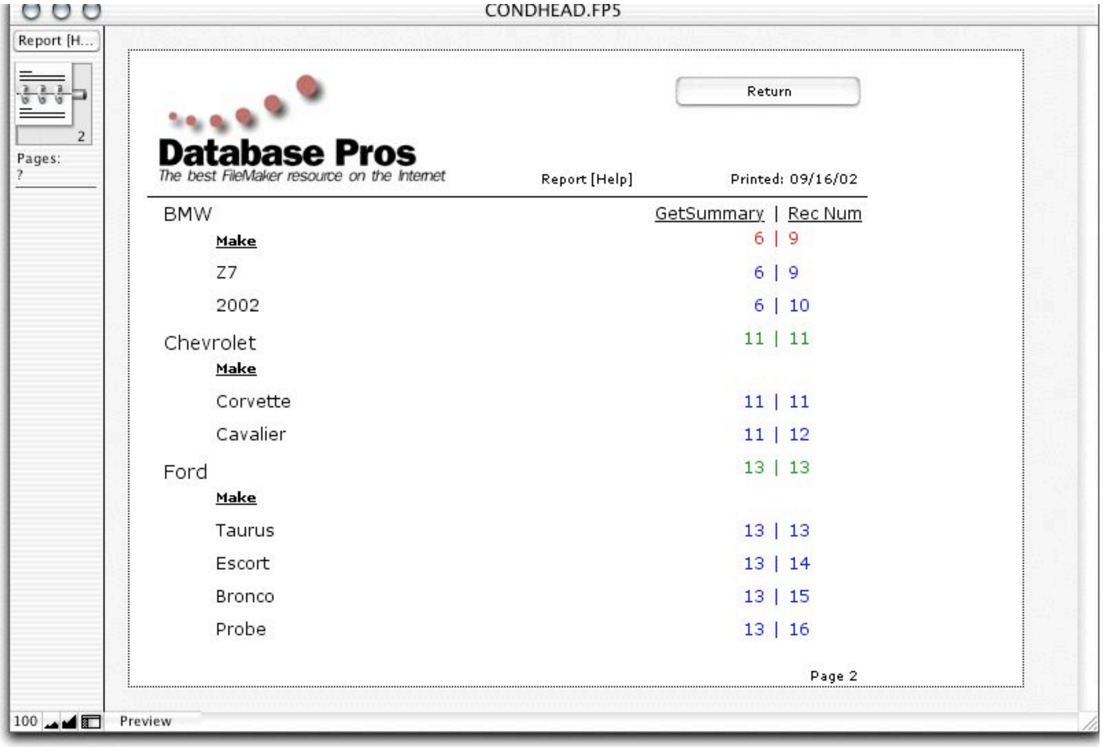
Report [Help]

Printed: 09/16/02

	GetSummary	Rec Num
	1	1
Acura	1	1
Make		
Integra	1	1
Legend	1	2
TL	1	3
CL	1	4
RL	1	5
BMW	6	6
Make		
X5	6	6
Z3	6	7
320i	6	8

Page 1

Figure 3: Second Page of the Report - Notice that the GetSummary and Rec_Num fields in red are different.



Database Pros
The best FileMaker resource on the Internet

Report [Help]

Printed: 09/16/02

	GetSummary	Rec Num
BMW		
Make	6	9
Z7	6	9
2002	6	10
Chevrolet	11	11
Make		
Corvette	11	11
Cavalier	11	12
Ford	13	13
Make		
Taurus	13	13
Escort	13	14
Bronco	13	15
Probe	13	16

Page 2

The GetSummary calculation field is simply the GetSummary portion of the formula shown above. The values all come from the same fields but the color of the result changes depending on the part. Red indicates the fields are located in the header part, blue shows the results from the body part and green is the result from the subsummary part.

Notice that the GetSummary and Rec_Num fields (in red) equal each other on the first page but don't equal each other on the second page. When you place a field in a header part and preview or print, it displays a value from the first record on the page. Therefore, the Manufacturer_Header calculation field is comparing values from the first record on the page. The Status(CurrentRecordNumber) is different on every record but the GetSummary function returns the same minimum value on every record within each set of records organized by the same break value. If these two values equal each other, the first record on the page is the first record in the subsummary part and the calculation should return a null value. If these two values do not equal each other, the first record on the page is NOT the first record of the subsummary and the calculation should return a value.

Considerations

One drawback of this technique is that header parts do not support all sliding features. Whenever the calculations in the header part return a blank result, it would be nice to reduce the size of the header part. Unfortunately, reducing the size of the enclosing part is not possible in a header or footer part. The result is a blank space at the bottom of the header which is noticeable when you compare a page that displays results for these calculations versus a page that doesn't. This may or may not affect your decision to implement this solution. That doesn't mean you shouldn't study the technique. You never know how you can apply the knowledge from one technique to another solution. Learn as much as you can about FileMaker and you will be rewarded in the long run.