FileMan 21: Twelve Steps to Better Sorting

by Tami Winn and Rick Marshall

This month's column reveals the new features, choices, and efficiency gains in FileMan version 21's sort and print modules. The overhaul of these print generation routines represents a sizable chunk of version 21's scope of work. As a result of these changes, FileMan users will enjoy greater control and efficiency in their reports than ever before. Site managers installing FileMan version 21 should train their computer users in the new sorting features brought in by this release, and train their developers to use the new features of the API (application programming interface).

To continue advancing into new database techniques and technologies, computer professionals need to invest in infrastructure to change the way they do business. As with the programming hooks we've discussed in past articles, sort and print templates often have included nonstandard code and structures to accomplish their tasks. While in the past developers have accepted the use of these nonstandard techniques in practice, for FileMan to move forward developers need to take responsibility for ensuring their templates are FileMan-compatible so they can reap the benefits of this evolution. Developers should take this opportunity to scrutinize their old sort-and-print templates to ensure they use only standard features.

As in so many other areas of the computer arena these days, surging ahead with new classes of features requires all programmers to tighten up the way they do business; the FileMan team needs to have the freedom to abandon old internal structures while maintaining a backward-compatible API. This freedom pays handsome dividends in features like these:

1. Speed: Sort Optimization

FileMan sorts faster. After FileMan users select the fields by which to sort a report, FileMan 21 samples the data to determine both the quickest route to loop through the file initially, and also the most efficient way to check whether records meet the sort criteria. Because FileMan can now make use of regular cross-references on most fields to do its initial loop through the file, this new feature often makes a dramatic difference in how long it takes to run a report. The FileMan team will use this same sort-optimization routine in a query tool planned for future development.

2. Speed: Sort Template Recompilation

Compiled sort templates respond to changes in their files. Compiling complex sort criteria has always given FileMan users faster reports, by converting a user's description into executable M routines. Now that FileMan can respond flexibly to changes in a file by changing how it sorts that file, a drawback in the compilation strategy has emerged: precompiled, templates do not change as the file changes under them, and thus over time can end up using a sort strategy Not as effective as it was when originally compiled. FileMan 21 lets compiled sort templates in on the benefits of sort optimization by recompiling the sort criteria each time the users run its report. This way, whenever the file changes enough to make some new strategy more efficient, its compiled sort templates will automatically shift to that new strategy. Users will especially notice the benefits of this with templates involving many sort fields or relational jumps.

3. Choice: Collation Control

FileMan users can pick the collation sequence of sort values. FileMan, like M, has always sorted purely numeric values like *1* completely ahead of mixed alphanumeric values like *1E*. However, FileMan users sometimes wish to sort mixed data as if entirely alphanumeric, with the purely numeric values mixed in throughout the report rather than clumped at the beginning, especially when sorting files that include ward names or Social Security numbers. FileMan 21 introduces a new sort qualifier—;TXT—to place after a free text sort field to make it sort as text rather than data. FileMan users can type this in at the sort prompts, while application programmers can include it in the FLDS input variable to the EN1^DIP call. This gives the choice of collation to those who need it.

To illustrate the difference, in a file containing the following ward names:

1, 1E, 1N, 1S, 1W, 2, 2E, 2W, 3, 3E, 3W, 4, 4E, 4W

if the user sorts from 2 to 3W, the report will show:

Without ;TXT	With ;TXT
2	2
3	2E
2E	2W
2W	3
3E	3E
3W	3W

4. Choice: Empty Report Notification

FileMan users now may decide what empty reports should show. If FileMan finds no file entries matching the user's selection criteria, it can now either display nothing as in the past, or print the report's header with a message declaring that FileMan found no matches. This lets the user decide whether to conserve paper with the traditional approach or to underscore the results of the report with the new approach.

5. Choice: Displaying Sort Criteria in Report Headers

FileMan users can make their reports show the sort criteria in each page header. This should greatly help third parties to identify the nature of a printed report.

6. Help: Sort Range Assistance

FileMan now offers users help in selecting the sorting range. If the user enters a question mark at the "START WITH" or "GO TO" prompt, FileMan will help in picking the range.

7. Architecture: New M]] Operator

FileMan uses the new M]] operator to decide whether a value falls within the sort range. This architectural change replaces a series of checks with a simple operation, and begins the process of making FileMan properly handle multiple character-set profiles. This change should interest FileMan's international users.

8. Architecture: Sort Criteria & FileMan Compatibility

FileMan stores the sort criteria for each sort template in a FileMan-compatible format. Now that FileMan does not keep this information in an arbitrary M format, FileMan's suite of file-manipulation tools can operate on all of the sort-related information in a template, making troubleshooting far easier.

9. Architecture: Easier Troubleshooting

Sort templates now encapsulate more of their own code. FileMan stores the software used to extract sort fields and to determine whether they meet the sort criteria as part of the sort template itself. This makes troubleshooting and enhancing sort templates easier, since more of the important information needed to do so is now kept with them, where troubleshooters would expect to find it.

10. Architecture: Greater Consistency

The FileMan team has modernized the sort routines. M has come a long way since FileMan began, and many of the new language features open the way to vastly more readable code. FileMan's sort logic now uses FileMan's Reader to handle all user prompts and many of M's more recent variable scoping features to keep the symbol table cleaner. In addition, FileMan 21 leaves behind many of the historical quirks and idiosyncracies that have lurked in the sort logic. Overall, this more consistent, readable, and maintainable code now meets DHCP's (Decentralized Hospital Computer Program) inhouse standards and conventions, and can more easily accept enhancements and modifications down the road.

11. Programmer Tools: Embedded Commas in Sort Values

Programmers can now use commas in the values that mark the edges of a sort range. The FileMan sort entry point, EN1^DIP, used commas to delimit values in its range specification input variables, FR and TO. This prevented programmers from using range values that include commas (e.g., "JANEWAY, KATHRYN"), since FileMan would interpret the commas as separators rather than as part of the values. EN1^DIP now accepts new input arrays under FR and TO that can accept values with embedded commas, giving programmers more flexibility in setting up their FileMan sorts.

12. Programmer Tools: Sort Control

Programmers now have many ways to control how FileMan sorts happen as means to make them more efficient. EN1^DIP lets programmers decide: which cross-reference to sort by first, to sort by a list of record numbers in a temporary global, to sort using search results in a search template, the sort ranges and subheader information for each subscript, and whether to apply further sorts to the data. This degree of control lets programmers rapidly eliminate unwanted records from the sort, letting FileMan avoid working on records that will not end up in the final report anyway. If programmers use this wisely, they can increase the speed of their sort-intensive reports.

Conclusion

FileMan 21 represents two years of work across a broad spectrum of functional areas. While past articles in this series have focused on the Database Server's role in opening the doors to graphical user interface and client-server applications, intensive work in other areas, such as the sort module, will delight FileMan users with new features and speed that affect existing reports. Reaping these benefits does not require the programming or retooling needed to make use of the DBS or programming hooks.

M Technology Association Staff

Maureen Lilly
Director

Sandra Reynolds
Associate Director

Judith Brown
Publications Assistant

Marlo Brown
Receptionist

Kristi Hanson
Communications/Administrative Associate

Ed Hinkle
Computer Services

Rita Mincavage
Communications Director

Marsha Ogden
Managing Editor

Don Piccone
MDC Secretariat

Susan G. Rathbone
Director, Member Services

Jill Todd-Klonzaris
Financial Services

Next time *Focus on FileMan* will continue its indepth look at the Database Server in Part 3 of the series on using it to supply FileMan data to a windowing application. **M**

Forward your FileMan questions or topics you would like to see addressed in this column to G.FILEMAN DEVELOPMENT TEAM@FORUM.VA.GOV, or write to VAISC6-San Francisco, Suite 600, 301 Howard Street, San Francisco, CA, 94105.

Tami Winn and Rick Marshall work for VA's San Francisco Information Systems Center, where they write and program for the FileMan development team. Rick also teaches the Kernel class at MTA's annual meeting, and works in MDC as the editor of the 2001 M Standard. Tami designed and developed FileMan 21's sort module enhancements.
