A Wrinkle in Timeliness Pt1 - Perfecting FileMaker Recursion & Virtualization, by Andrew Persons of Excelisys

Our client had an interesting problem. They needed to recreate Microsoft Project in FileMaker.

Well, not quite *recreate* it, but they did need to replicate one important aspect of it.

They had large projects with hundreds of tasks whose time frames depended on each other. Each task could rely on multiple other tasks that needed to be completed before it could be started, as well as multiple tasks that depended on it. With hundreds of tasks per project and tens of thousands of occurrences, they needed to quickly (within a second or two) update all affected tasks any time a task's estimated time frame changed.

A Proliferating Headache

First, a quick overview. Each project started with a Concept task, with all other tasks "downstream" from it. Each task had a projected duration (in days).

- Concept (2)
    - Task A (2)
    - Task B (4)

Task A and B couldn't start until Concept was complete. So, if Concept started on January first, they would each start on January third. Then each one needed to display a projected completion date based on its projected duration:

- Concept (2) - 1/3
    - Task A (2) - 1/5
    - Task B (4) - 1/7

So far, so good. A straightforward loop through the records starting from the root (Concept) could update each task's projected completion. If Concept was pushed back a day, Task A and B would also be pushed back a day and so would any tasks that depend on them. No problem!

Not so fast. Let's throw a monkey wrench in:

- Concept (2) - 1/3
  - Task A (2) - 1/5
    - Task B (4) - 1/9
    - Task C (5) - 1/10
  - Task B (4) - 1/7

Now Task B depends on BOTH Task A and Concept. It can't start until both are finished, so it won't be finished until 1/9 (the latest completion date of Task A and Concept, plus its own projected duration).

Suddenly, it's a much thornier problem. Here's a snapshot of what a very small portion of the full structure could look like if each task occurrence were "expanded":

```
Project Personnel Assignments
   Design File Initiation
      Evaluation C
         Prototype C
            Design C
               Evaluation B
                  Executed Engineer Assignments
                     Approved Needs Assessment
                        Concept
                     Approved Engineer Assignments
                        Concept
                  Prototype B
                     Design B
                        Evaluation A
                           Prototype A
                              Design A
                                 Concept

Create Prints
   Test Prototypes
      Build Prototypes
         Preliminary Quotes for Purchased Items
            Evaluation C
               Prototype C
                  Design C
                     Evaluation B
```

Executed Engineer Assignments
Approved Needs Assessment
Concept
Approved Engineer Assignments
Concept
Prototype B
Design B
Evaluation A
Prototype A
Design A
Concept
Create Prototype Prints
Evaluation C
Prototype C
Design C
Evaluation B
Executed Engineer Assignments
Approved Needs Assessment
Concept
Approved Engineer Assignments
Concept
Prototype B
Design B
Evaluation A
Prototype A
Design A
Concept
Create Prototype Prints
Evaluation C
Prototype C
Design C
Evaluation B
Executed Engineer Assignments
Approved Needs Assessment
Concept
Approved Engineer Assignments
Concept
Prototype B
Design B
Evaluation A

```
                                    Prototype A
                                       Design A
                                          Concept
```

Multiply that by a thousand and you get an idea of the scope. How do you resolve that?!

Let's look at some of the different possible approaches.

Calculate It!

Let's get the simplest brute force approach out of the way. Create an unstored calculation in the Tasks table that looks at its parents (tasks it directly depends on), takes the latest date and adds its duration to that. It would refer to itself and look something like this:

date_calculated =

Max (Tasks_Parents::date_calculated) + duration

This works in theory, but be prepared for a lot of downtime. Lacking masochistic tendencies, I didn't actually try it. I have no doubt it would bring FileMaker to a grinding halt, if not crash it.

Getting Loopy

Another option would be to loop through the tasks, starting with the root task (Concept).

This approach would go to Concept, store its completion date and then loop through the tasks immediately dependent on it, setting their projected completion date. The first pass would set Task A's projected completion date to 1/5 correctly, but Task B would be set to 1/7. We'd then have to repeat the process: go to Task A, store its completion date and loop through ITS "children". If the task already had a date, set it to the new date only if it's greater than the existing one.

For this to work, we would have to use a "recursive" approach; that is, we would have to create a script that calls itself. It would look something like this:

Update Tasks =

- Set date to [Get (ScriptParameter) + duration]
- Go to related children
- Loop
    - Perform Script [Update Tasks (date)]
    - Go to Next Record [Exit After Last]
- End Loop

There are two major problems with this approach.

1. For a recursive approach to work, it has to maintain its "scope". In this case, it means that the found set would have to be preserved each time we called "Update Tasks". We could accomplish this by opening a new window when we call the script and closing it when we exit the script, but this would result in hundreds of windows being generated through the course of the script. This could potentially crash FileMaker through excessive memory usage. It would also slow things down considerably with the overhead of creating and closing all those windows.

2. It would require us to update the same task many, many times. A project might have a few hundred tasks, but have tens of thousands of occurrences of those tasks. This would be excruciatingly slow.

Exercising Our Options

OK, if not the previous two approaches then what? Next time, we'll look at some other possible approaches, and begin to break down the approach we eventually used.

Andy Persons is a Lead Senior FileMaker Pro Developer with Excelisys: Andy has been an industry leading FileMaker Pro developer creating FileMaker Pro solutions for over 17 years. In addition to being one of the lead developers of three top-rated and most-downloaded FileMaker Pro solutions of all-time; the FileMaker Business Tracker and the Excelisys eX-BizTracker & eX-BizTracker Pro jump-start solutions, he has shared his incredible and advanced talents by authoring numerous Tips-n-Tricks files and white papers, including Hierarchical Portals, Recursive Calcs, Audit Logs and Drag-and-Drop using FileMaker Pro.

• www.excelisys.com • 866•592•9235 • info@excelisys.com •
http://www.excelisys.com/filemaker-tips-tricks-demos-downloads.php