White Paper

Approaches to Graph Modeling

9 Steps Toward Enlightenment With FileMaker Pro

Table of Contents

ABOUT THE AUTHOR:

Ray Cologon is the author of the FileMaker Pro 9 Bible. Ray has been working with FileMaker, both in house and as an independent contractor, for most of the last two decades. He is the director of NightWing Enterprises, based in Melbourne, Australia, a provider of FileMaker design, consulting and development services to clients worldwide. Ray is an award winning trainer and speaker in Australia and was recipient of the 2005 FileMaker award for Leadership and Technical Excellence in FileMaker Pro. Ray is also a FileMaker 7, FileMaker 8 and FileMaker 9 Certified Developer. Contact him at cologon@nightwing.com.au.

Preface:

If you are entirely certain that your work with the Relationship Graph within FileMaker Pro or FileMaker Pro Advanced is the best it can be, that you have weighed all the options and you know what you are doing – then this paper is not for you.

This paper assumes you have an intimate knowledge of the basic technologies of relationships in FileMaker Pro 7.x, 8.x and 9.x. This paper is not a primer on relationship graph fundamentals or a "how to" paper. Instead, it is a discussion of the principles and ideas that shape our work in FileMaker Pro. If you do not already know how the Relationship Graph works then you should consult other resources first before attempting to digest the information offered here.

The Relationship Graph examples appearing in this paper closely resemble those of real working solutions. However, any resemblance to the Graphs of your solutions is purely coincidental. Furthermore, the names of Table Occurrences have been changed to protect the identities of those responsible for them.

Context:

It was the evening prior to DevCon. The night was full of promise, and I was in the company of other FileMaker Pro developers from around the globe enjoying dinner in a quiet nook of a very pleasant restaurant.

Early in the evening, someone chanced to remark that "anchor-buoy was really the only viable graph modeling method" and it appeared there was general agreement at the table. I paused – contemplating the ice in my first drink of the evening – then drew a deep breath. Over the course of the evening's discussion, I named and briefly described each of the nine Graph modeling approaches mentioned in this paper.

Like every aspect of life, development with FileMaker Pro is a journey. We cannot go back and, for the most part, we would not wish to. The things we have learned cannot be unlearned. The solutions we worked on in previous years show us where we have been, but they do not limit or define us. At some point we must stand apart from what we do and what we have done, to consider who we are and where we are going.

I decided to share with my colleagues, a collection of perceptions about the thing known as the FileMaker Pro Relationships Graph, placing it within a broad frame of reference. As a result, it seems that the particular group of colleagues assembled on that evening left with a view they did not have upon arrival – but whether that is a good thing is for others to judge.

The subject is not exactly a light-hearted matter - yet deeply serious contemplations may sometimes provoke moments of spontaneous levity. Regardless, should any aspects of this paper appear whimsical, please be assured it is to be expected AND is an essential part of the clarity and purpose of what must be said.

If you are ready to share an adventure, read on. In the pages that follow, I will set out the essentials of the journey I sought to articulate on that Orlando summer evening.

What This Paper Is Not: Relational Theory

Let's be clear: this is not a paper about Relational Theory. It is about the Relationships Graph and how one works with it. Relational Theory and working with the Relationship Graph not the same thing – though many developers start out believing they are. If you are looking for an introduction to (or discussion of) relational theory, then this is not the paper for you. Other resources deal with that topic. One such resource resides in Chapter 12 ("Data Modeling in FileMaker") of the recently published FileMaker Pro 9 Bible; which I happily recommend.

If you are still here with me after reading the above, I will take it as a given that you have a healthy sense of humor along with a well-developed working concept of relational theory. I will take it that you are acquainted with terms such as Entity Relationship Diagram (ERD) and are familiar with their use. Moreover, I will also assume that you appreciate the distinction between data modeling, application or logical design and file architecture.

Notwithstanding the above, one of the first concepts essential to understanding the way the FileMaker Pro Relationships Graph works (and how it differs from an ERD) is to appreciate that tables on the graph (Table Occurrences – commonly referred to as TOs) are not the actual tables but merely pointers to them. Thus, the same underlying table may be represented by numerous TOs on the Graph, implicating it in different ways in the data flows and logical processes that are supported your graph design. You will find yourself adding TOs to the Graph for a variety of reasons. For example, you might add TOs to a Graph to…

- Filter a value list
- Display records in a portal
- Summarize line items by status
- Navigate between related records

Many of the above examples (and various others that will likely occur to you) have little or nothing to do with the data model of your solution.

While the FileMaker Pro Relationships Graph is a tool that interacts with the relational model, the logical model and the file architecture of your solutions (and may at times seem to dictate them), it is worthy of separate examination. The knowledge and skills that will be of benefit when embarking on the design and implementation of a Relationships Graph for your solution are not the same as those that equip you to arrive at a workable data model, an apt logical design or a suitable file architecture – though there is obviously some overlap. It is for this reason that I use the term "Graph Modeling"; it underscores a central distinction.

Overview: Graph Management

The FileMaker Pro Relationship Graph can be an enigma. It has prompted reactions ranging from awe to exasperation and from excitement and anticipation to confusion, uncertainty and even mild bewilderment. The Relationship Graph alone has been responsible for the permanent retreat of some faithful FileMaker Pro developers into the nether world of “FileMaker Previous”, never to emerge. Still others have adopted organized and stringent Relationship Graph strategies in an effort to move forward within an uncertain domain.

Whatever else may – or may not – be true, the post-FileMaker Pro 7 Relationship Graph cannot be fully explored or appreciated if we remain entirely tied to predictable, mundane or well-established thought patterns. It is different. It requires us to think different. Fortunately, many of us are good at that – it is what attracted us to FileMaker Pro in the first place.

Solution modeling via the Relationship Graph in FileMaker Pro is not a routine exercise. It is not merely a metaphor for known relationships between modeled realities. By its nature, it invites you to plot a path (knowingly or otherwise) towards metaphysical and conceptual dimensions. In doing so, the Graph becomes its own reality. However, even if you are interested only in concrete methods and techniques, you will nevertheless find things to think about and a variety of alternative approaches discussed in this paper.

What follows is my account of a journey of discovery. This paper uncovers nine distinct approaches to the use of the FileMaker Pro Relationships Graph – Nine operational models, each with direct and significant implications for your work patterns and your solution architectures. The journey follows a natural path that awaits anyone who chooses to follow it. The models described are loosely arranged in the order they naturally arise during the course of exploration – the further you delve into the Graph's mysteries, the more the insights recounted here will reveal themselves to you.

A majority of the Graph modeling approaches described here are functional approaches in the first instance. However, as methodologies, they encapsulate more than just technique. They each hinge on:

- How you think about the Graph as you work with it
- How the Graph represents or encapsulates the architecture (data architecture and/or logical frameworks) of your solution

- How you organize the elements of the Graph, regardless of their functional significance
- The deterministic relationship between the arrangement of elements on the Graph and the design of your solution

The purpose of this paper is exploratory and not meant to establish rules or rigid practices. You are encouraged to consider alternatives as they apply to the solutions that you create using FileMaker Pro. Whether you choose to embrace any, all or none of the techniques or concepts described here is entirely up to you. However, if you are open to considering alternatives and ready to contemplate concepts that may at first appear foreign then the following is food for thought.

Starting the Journey – Embarking on a Rethinking Process

Every developer or user who has worked with any version of FileMaker Pro from 7.0 onwards has encountered the Relationships Graph and made assumptions about it that have subsequently required revision, refinement, or a wholesale rethinking over time. Whatever the Graph is, one thing is established – it is not what it first appears to be.

One of the consequences of the deep nature of the Graph metaphor is that the first things it occurs to us to do with the Graph rarely prove to be the best things. Because of this, a part of the rethinking process involves coming up with alternative ways to use the Graph in the solutions that we develop. Some developers have rapidly settled into a pattern of behavior that they are comfortable with and see no reason for further exploration. Others have continued to revise and refine their approach, seeking new ways to understand and use the Graph. They have not been disappointed.

The process of coming to terms with the Graph has a natural evolution – there are certain common elements and predictable stages and patterns for most developers. The misconceptions that most of us began with caused many of us to produce something of a mess in our first attempts to produce a Graph for a complex solution, following the release of FileMaker Pro 7. The paths we took to extricate ourselves and our solutions from the tangles of those initial experiments have some identifiable commonalities. Let us begin to look at how the Graph journey unfolds.

Model 1 – The Amorphous / Chaotic Model

The first thing most developers do when confronted with the Graph and a complex solution is to connect everything to everything else. It seems the thing to do. No one told us not to. FileMaker Pro does not permit circular references, so there can be only one path between any two-table occurrences – but a single very large group of table occurrences frequently emerges in which there is little discernable organization and no apparent guiding framework. The Graph ends up pretty much however it grew. Sound familiar?

The Amorphous / Chaotic Graph model is therefore the default position for many first-time FileMaker Pro developers. It is the model you have when you do not have a model. Maybe more to the point, this model takes shape when the developer has not yet come to appreciate the need for a more purposeful approach to Graph Modeling. The Amorphous / Chaotic model of Graph management, despite some obvious drawbacks, remains the most common graph management methodology, since many solutions are either small or are developed by first time developers or end-users.

This first-time (and often one-time-only) modeling methodology includes structures commonly referred to among FileMaker developers as "The Spider" and "The Angel of Death" – so called in part because of their appearance, but perhaps also because of the sense of doom that accompanies them in the minds of some. This model comprises various approaches to the use of the Graph wherein typically, everything connects (ultimately) to everything else. Because of this superabundance of connectedness, these graphs tend to have lines going all over the place frequently become difficult to follow. The beginning of this process is shown in the Graph snippet in Figure 1.

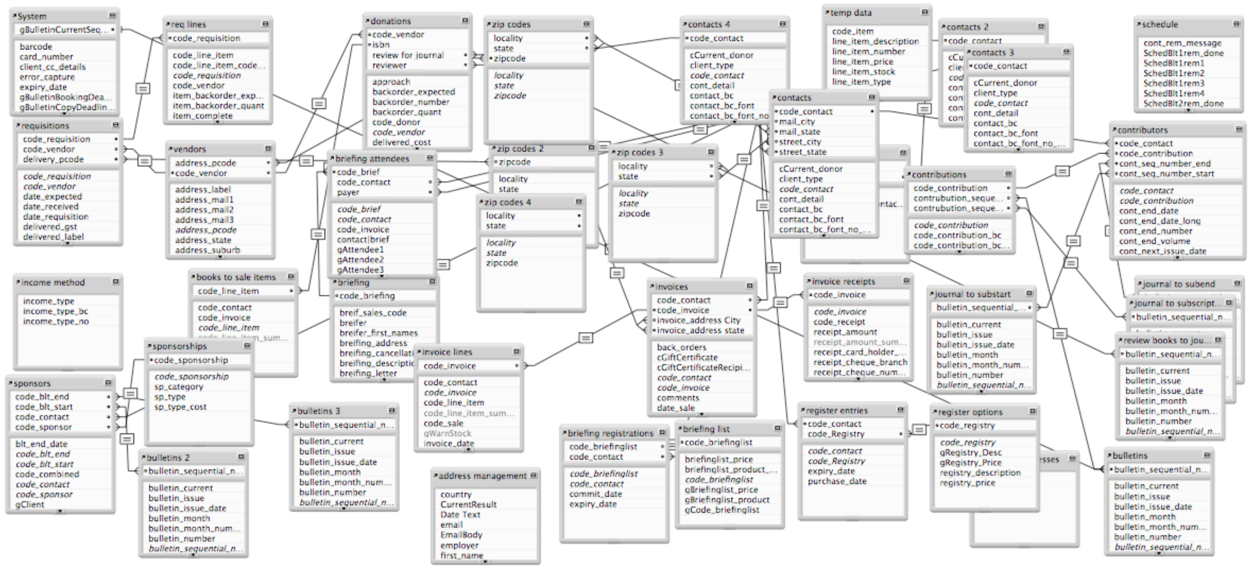

Figure 1 – *An Amorphous / Chaotic Graph design of an actual real-world solution comprising only 27 tables, where the data model is obscured by contrivances serving functional needs and the manageability of the graph is compromised.*

An Amorphous / Chaotic graph is the default position and may be considered to apply in all cases where the developer has not adopted an ordered and systematic methodology to aid in Graph management. Thus, even graph implementations where not everything connects to everything else may be considered Amorphous / Chaotic if no clear alternative organizing principle can be discerned.

In addition to the above, Graph designs that are built around the structures that result from migration of a solution from a previous version of FileMaker Pro (i.e. FileMaker Pro 6 or earlier) can also be considered as falling in this category if the developer has not imposed an alternative model for graph management. An example of such a graph is shown in Figure 2. Whereas the graph shown in Figure 1 has reached the point of being unmanageable, the graph in Figure 2 is of a smaller solution. This smaller solution is earlier in the development cycle and is arranged in a more orderly fashion (built upon the archetypal angel or spider structure).

Nevertheless, lacking any evident alternative organizing principle, both may be regarded as Chaotic / Amorphous examples.

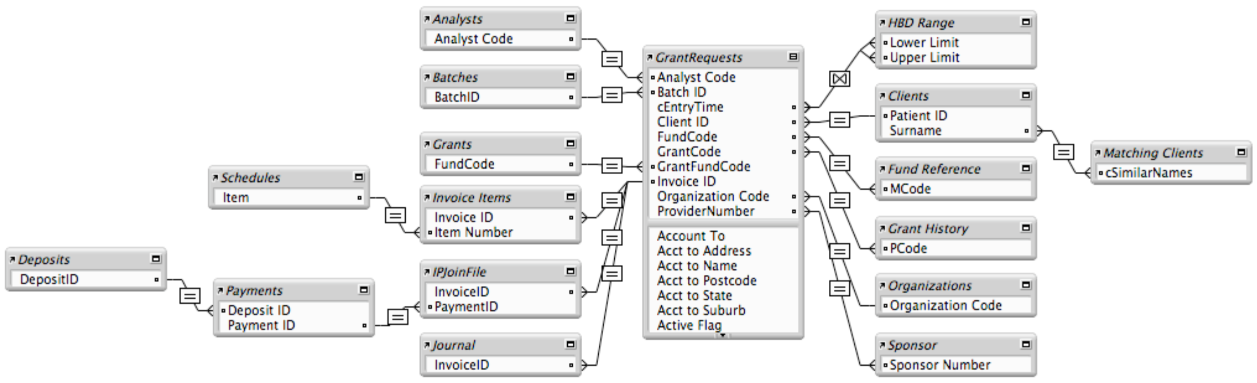

Figure 2 – *An Amorphous / Chaotic Graph design taking shape in a solution that was originally migrated from FileMaker Pro 6*

Despite the inadequate organizing principle (or lack of a coherent organizing principle) of the Amorphous / Chaotic Relationships Graph, solutions where such graphs are used frequently start with a comparatively clear concept or outline. When first confronted with the Relationships Graph, most developers begin with an abstract or hypothetical data model from which they are forced to diverge at regular intervals in order to accommodate the stringencies of the Graph model in FileMaker Pro (e.g. avoidance of circular references).

Departures from the developer's abstract or hypothetical data model spawn many Table Occurrences (TOs) to serve a variety of purposes in the management of core/underlying data relationships, procedural requirements (scripts, reports, summaries etc) and interface support (filtered portals, GTRR buttons etc). The Graph ends up carrying complexities that obscure the clear and (relatively) simple data model the developer first had in mind.

The resulting Graph may still encapsulate the essentials of a data model, however this has become increasingly obscured and interleaved with constructs of convenience to the point that any attempt to discern the elements of an ERD within the Graph becomes difficult or impossible. When viewing and editing more complex solutions, the manageability of the Graph is compromised by the constraints and imposed requirements of the use of the Graph in this way.

The Amorphous / Chaotic Graph Model is where most developers start out on their journey of discovery with the .fp7 versions of FileMaker Pro – but few developers persist with this approach for their second or subsequent solutions. For small and/or simple solutions, there may be no need to consider other options. However, as complexity increases, developers have sought alternative strategies with which to tame the growing complexity of the amorphous beast. Hence, Model 2.

Model 2 – The Squid / Anchor-Buoy Model

Shortly after the release of FileMaker Pro 7, as the unsuitability of a disorganized approach to the graph became clear to developers working on complex solutions, the quest began for an alternative approach – one that would introduce order into large and unwieldy graphs. Initially, many developers reflected on the more manageable simplicity of the relationship management model in FileMaker Pro 6, and some returned to the familiar tools of the past. Developers and users on various discussion lists yearned for list-based relationship definition dialogs and ways to constrain context so they could view relationships from the perspective of a single vantage point at a time. Before long, it occurred to many of those confronted by this problem, that the new graph-based relationship system in FileMaker Pro could be used to build such structures.

More-or-less simultaneously, developers in various parts of the globe moved to make sense of confusion by the imposition of order. An essential element for the imposed order was that it must improve the ability of the developer to see and manage context from any vantage point within the solution. The first methods to merge from these efforts were the Squid and Anchor-Buoy techniques (and any of several other variants which adhere to similar essential principles) in which developers designate a single TO as the basis of all layouts associated with a given base table, building separate structures (groups of related TOs) for each.

The basic concept of Squid/Anchor-Buoy and all its variants is that the TOs that layouts are based on (the Anchors or Squid-heads) always provide the context for layouts, so the developer is always "looking" in the same direction down the relationships that connect that TO to its associated TOs. By imposing a discipline in which all relationships are used in one direction only and only one Graph occurrence of each base table is used for layout context, a somewhat simplified set of rules emerges enabling consistency and order (at least in comparison to the Amorphous/Chaotic approach). An example of part of the graph of a solution where this approach has been used is shown in Figure 3.

While this graph management model provides several immediate benefits, it does so at a cost. In the first instance, the developer must forgo some of the newfound flexibility of the Graph (e.g. two-way relationships, reusable Graph elements). In fact, the practices of the various versions of Model 2 reinstate several of the essential constraints of the relationship model of FileMaker 6 (and earlier) in which relationships are only ever viewed and used from the perspective of one table at a time and all layouts associated with that table are tied to a single instance (TO) of that table. Anchor-Buoy and Squid and other variants are essentially what you end up with if you preserve and extend the relationship logic of a solution migrated from an earlier version of FileMaker Pro. Because of this, this model holds the hearts and minds of some developers captive with the familiarity of a known and understood model.

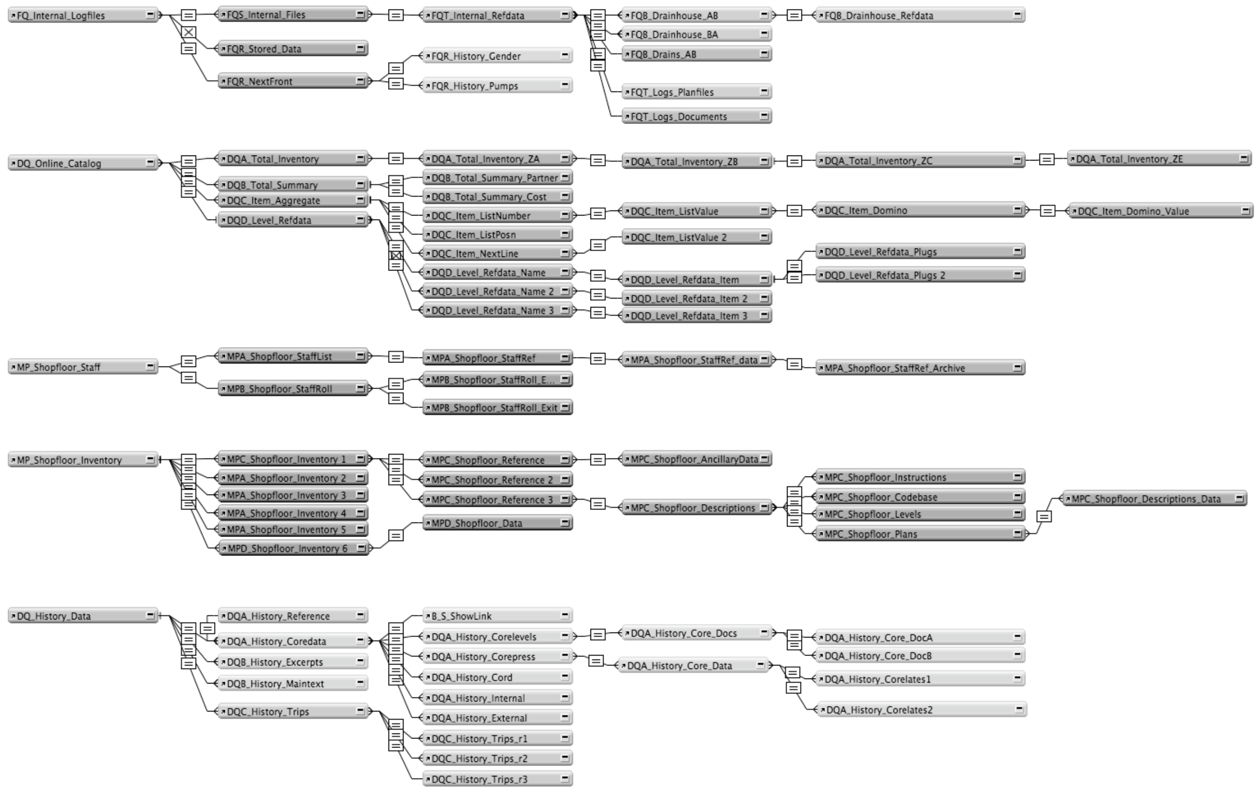

Figure 3 – *A portion of the Relationship graph of a twelve-table solution in which a variant of the Squid / Anchor-Buoy Graph Modeling technique has been used (for compactness, all the TOs are shown in their collapsed state).*

One of the accelerating costs, however, of the Squid/Anchor-Buoy model is the redundancy it introduces to the Graph (and consequently, the proliferation of joins), since essential relationship structures must be repeated for each context for which they are required to be available. In large and complex solutions, portions of the graph may need to be repeated many times to support the limitations of this model. This imposes a performance penalty through the number of redundant dependencies it introduces to the solution, and the join-caching burden they carry. It also requires the developer to maintain many redundant instances of similar structural elements. Moreover, it requires adherence to a naming convention to ensure that multiple instances of essentially the same structures can be reliably differentiated.

While some developers found themselves at ease with this innovation and got on with the business of building solutions, others have been less comfortable with the inherent compromises imposed by this method. If you are one of those experiencing frustration with the constraints, trade-offs, stringencies and costs of the second model, eventually you will begin to contemplate other possible paths to the desired end-points and it is at this nexus that the seed of Model 3 germinates.

Model 3 – The Modular/Centric Model

At the point where the shortcomings of models 1 and 2 call for a re-think, a melding of the elements of the first two produces a third distinct model. In this model, natural functional components of a system spawn modular centers of Graph elements. This is a logical next step and proves an evolution of the graph management model that is not too difficult to implement, yet provides some of the advantages (and perhaps fewer of the disadvantages) of either of the preceding models.

As the third model takes shape, you can envision a methodology where small core ERD-like structures are set in place separately (disconnected) for each main area of functionality of the solution. Radiating out from each of these functional centers, branches of supporting TOs are arranged, providing for the relatively discrete operations of a functional module of the solution. An example of a Modular/Centric graph structure is shown in Figure 4.

Since most solutions have several areas of overlapping functionality, this model allows a degree of natural separation of graph elements. Each modular center will require a few supporting TOs that point to base tables that have their "home" in another module, but the extent of redundancy will typically be moderate (considerably less than for Model 2). Most layouts will be associated with one of the core TOs in the module TOGs, however numerous relationships (particularly those between the core TOs for a module) will be re-used and utilized in both directions.

The modular approach introduces flexibility for the developer that is not available in the second model – and the resulting graph structures preserve the manageability of smaller solutions such as the one depicted in Figure 2, while supporting functionality that is considerably more complex than the solutions depicted in either figures 1 or 3. Moreover, the Modular/Centric approach reduces the ratio of table occurrences to tables and constrains join proliferation. This relieves the solution of some of the caching burden of more join intensive models (e.g. models 1 and 2), and at the same time moderates the need for exhaustive use of a strict naming convention. The modest number of table occurrences can be managed adequately with natural or descriptive names if preferred, in all but the largest of solutions.

An additional feature of the Modular/Centric graph Modeling approach is that it lends itself well to modularization of other aspects of the solution. For example, should you choose to place billing functionality in a separate file, the TOG associated with billing functions could be located in another file, without affecting the operations of the existing file. This is the case regardless of where in the file architecture the data tables for the billing function are located.

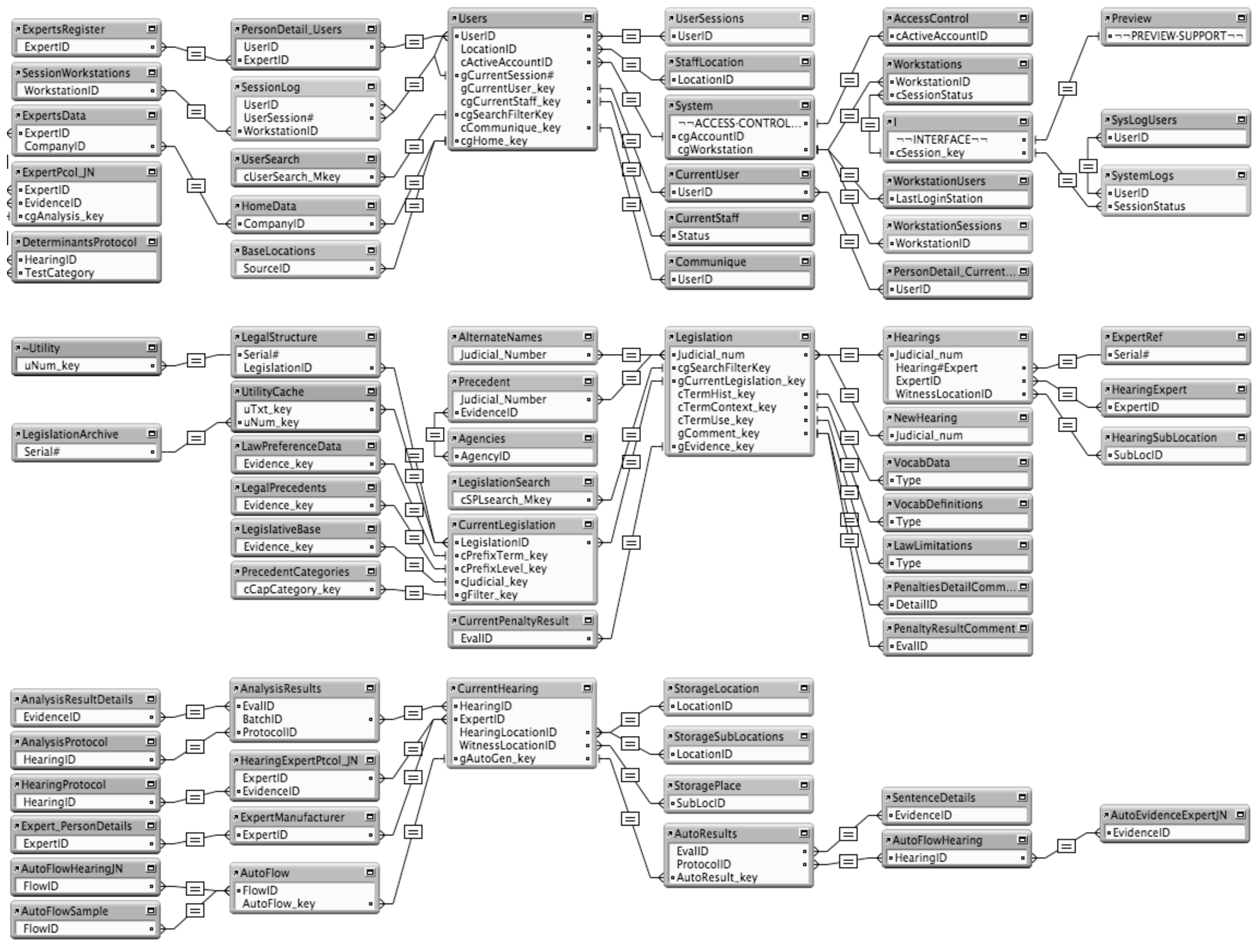

Figure 4 – *Part of the Relationship Graph of a solution of moderate complexity (35 tables in total) in which a modular / Centric graph Modeling technique is applied.*

The Modular/Centric approach provides an elegant and efficient framework for many solutions. Its benefits are immediate and enduring. However, its appropriateness depends on being able to identify a natural separation (modularization) of functionality within a given solution. Thus, Model 3 is highly successful in many cases, but may prove cramped and forced in others – particularly where there is a close interaction between all the essentials of a solution. In such cases, a different approach is called for and to move forward we must be prepared to re-think yet again.

Model 4 – The Flotilla or Cluster/Satellite Model

In seeking a further evolution of graph modeling, it is important to move forward, not backward – to preserve and build on the strengths of previous discoveries. One of the strengths of the third model is that it permits relatively straightforward modeling of the data structures for separate parts (modules) of functionality within your solution. In this respect, it is an improvement over all the flavors of the first and second models, especially when it comes to complex solutions. Thus, the data model of the solution although fragmented, is not altogether obscured.

In considering a move away from the techniques discussed to date, it seems a natural step to look for ways to further press the advantages gained from the improved clarity of the data model. This leads in the direction of exploring ERD-like structures and brings us to the next logical breakthrough – a fourth naturally occurring graph model.

In this fourth model, a single group of TOs forms as a cluster at the heart of the Graph and serves as the primary data model for the solution – remaining largely true to the model of the ERD. However when a requirement arises that cannot be accommodated in the central TOG (within the constraints FileMaker Pro imposes on the relationships graph – e.g. no circular references), a separate isolated group of two or three TOs is created outside the main TOG. This gives rise to a large central TOG and a series of smaller "satellite" or "flotilla" TOGs, arranged separately, as exemplified by the graph shown in Figure 5.

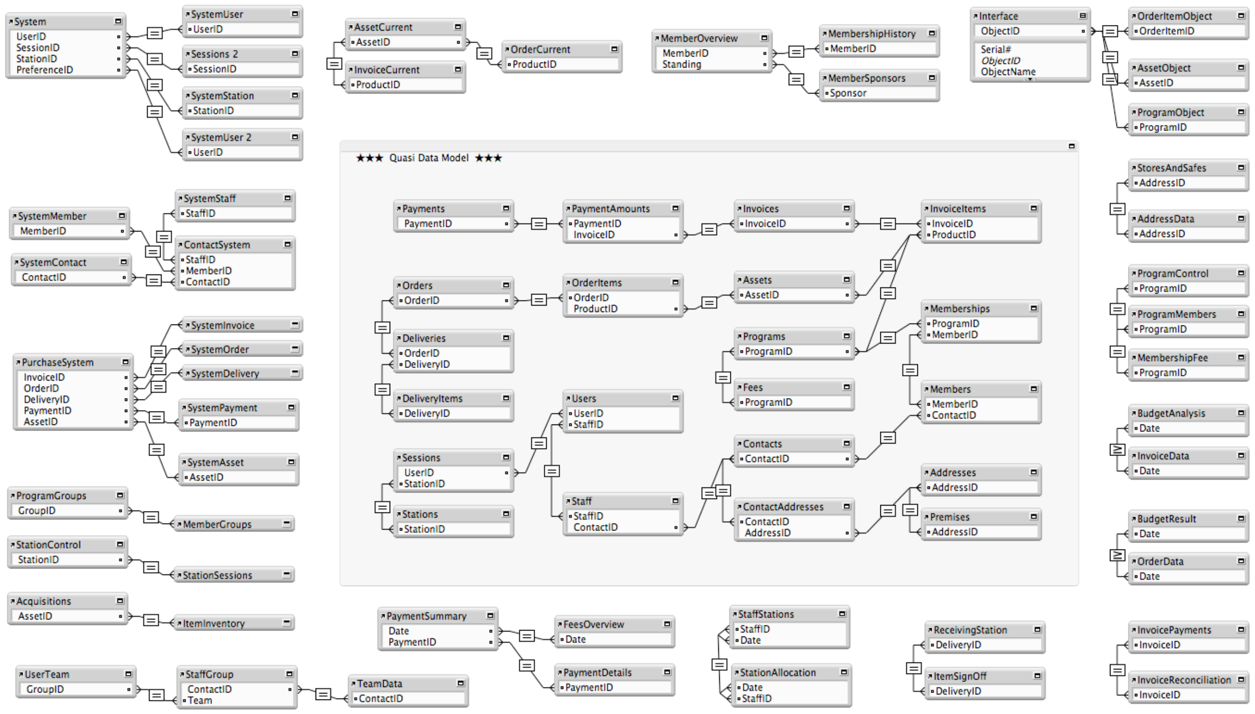

***Figure 5 –** The graph structure of a 23 table solution developed using the principles of the Flotilla/Satellite model.*

The Flotilla/Satellite method represents a radical re-think of the way the Relationships Graph supports the functionality of the solution, requiring a more heavily script-driven process model. To support the functionality of the growth of small special purpose TOs, a number of utility layouts are required and these are invoked by script as needed. In this model, the graph is comprised of a main central (ERD-like) data model and a group of (potentially) dozens of small special purpose (typically single-purpose) TOGs.

The Flotilla model provides an elegant, simple, integrated data model. It offers clarity and purposeful order to the graph. There is moderate redundancy (e.g. tables that are represented in the main TOG, which must be repeated in the satellite TOGs to support specific activities such as filtering, value lists, scripted GTRRs etc). Join proliferation is therefore moderate. Typically, the ratio of TOs to tables is lower than the Squid/Anchor-Buoy model, and may be lower than the Modular/Centric model, depending on the specific requirements of the solution.

The Flotilla model can be adapted to serve many purposes, even in some cases supporting the requirements of complex solutions. However, it can cause an increasing dependence on scripted control of the solution, wherein the context must "hop" from the main layouts to one or more utility layouts for various procedures.

Developers who have reached this point in their journey of discovery with the FileMaker Pro Relationships Graph have acquired a solid understanding of a number of options and possibilities the Graph offers and have a collection of alternative approaches from which to choose depending on the needs of a given solution. Some who have reached this point have found the available options satisfactory and have contented themselves with the knowledge they've gained.

Developers are nothing if not adventurous – and having read this far, you can be assured that for those with a thirst for exploration, the journey continues.

Model 5 – The Procedural Control Model

In some respects, the preceding model bends the functionality of FileMaker Pro to the will of the developer to make a less-than-obvious model workable. It requires that the developer adopt a tightly scripted and controlled application model where the Graph is supported by other elements to deliver needed functionality and provide process support. This begs the question as to whether there are other innovative application models that permit different approaches to the use of the Graph – and indeed there are.

More or less simultaneously during the course of 2005, several developers in different parts of the globe came up with innovative and quite different methods to provide solution and relational functionality without recourse to conventional Relationship Graph techniques. The only thing these methods had in common with each other – and with the Flotilla/Satellite technique, is a reliance on the interaction between scripts and other elements of the solution structure. That is, the methods use process control techniques in combination with simplified Graph structures to extend the capabilities of the application.

Examples of procedural control implementations include an implementation described and demonstrated by Michael Harris of Cerne Systems Inc, and Jonathan Stark's "Ginko" User Defined Data Model demonstrated as proof-of-concept at DevCon 2005.

The first of these implementations (first described by Michael Harris) has two calculated fields (one stored and one unstored) and a global field in each table. The unstored calc evaluates the content of the global (e.g. using GetField( )) to retrieve and prefix the value of any one of the other fields in the table. The stored calc concatenates (CR delimited) and prefixes all potential key field values in the table. A relationship is created from the unstored calc in one table to the stored calc in the next and so on, so that all the tables are joined in this way through a cascading array. An example of a graph structure using this approach to support a solution comprising 23 tables is shown in Figure 6.

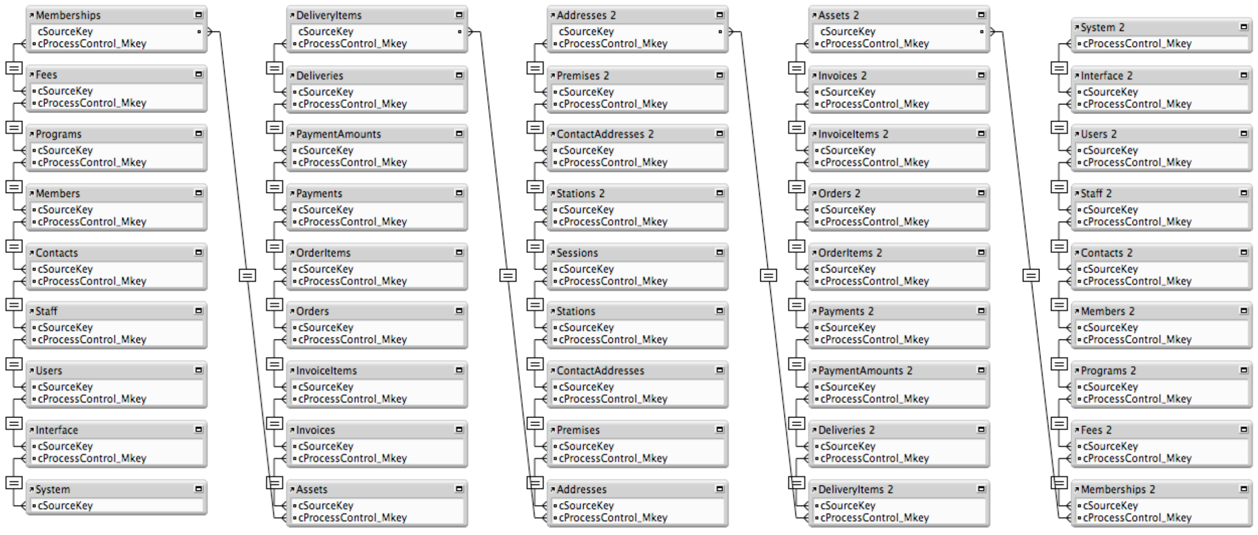

***Figure 6 -** The simplified array structure used to enable a Procedural Control implementation adapted from the method described by Michael Harris.*

Using the calculated key fields and a cascading array of TOs as shown in Figure 6, any relationship can be created programmatically between any two tables by setting the required key designations into the global fields in each of the tables. Thus, relational matches can be created and destroyed in real time via a script that resets the global assignments throughout. This permits relational configurations to be stored as data (an array of key values) and invoked at will by passing the array to a key configuration script. Navigation scripts taking the user to each layout of the solution are therefore also configured to establish the required relationship contexts to support the functional requirements of the destination layout.

It is not the purpose of this paper to fully explore or critique the various implementations of procedural control-based solutions, merely to acknowledge that they exist and that they provide a relatively open-ended alternative approach to solution design – and to Graph Modeling. As with the approach detailed above, the proof-of-concept provided by Jonathan Stark with his Ginko solution exposed another technique with similarly significant implications for graph design.

In implementations of Procedural Control application architecture, the structure and data design of the solution are defined in real time and as needed during the solution processes (typically via script) and the graph becomes a simplified and secondary supporting tool, configured to enable data model interventions which occur elsewhere in the solution. Whereas the Flotilla model required an increased reliance on scripted controls to manage context and support its graph modeling principle, the procedural control model relies almost entirely on run-time scripted processes to determine structure and context at every instant.

The defining concept of the Procedural Control model is abstraction. On fully comprehending and implementing such models, it becomes clear that the graph can be regarded as an enabling tool – that it need not define the solution nor dictate the parameters within which the developer or the solution will work. At this point the data model and the functional model of the solution exist outside the graph – perhaps on a whiteboard, in a spreadsheet, or in the developer's head. Not on the relationship graph.

While the approaches grouped together within this model in all their various permutations are revealing, liberating and intriguing, they do impose some stringencies and limitations on the developer. They involve uses of the FileMaker Pro platform that are somewhat at odds with the mainstream and not widely understood in the FileMaker Pro community. Moreover, the structures supported in any one implementation of a procedural control based solution are better suited to some solution requirements than to others – and not all of them scale well. Consequently, while this model in all its permutations represents a breakthrough in thinking, the number of solutions relying exclusively on procedural control implementations is likely to remain small.

Model 6 – The Hybrid Model

The preceding five models provide a rich source of alternative approaches to the use of the tools made available on the FileMaker Pro Platform. Each of the five models represents a different framework for solution implementation. Each has its own strengths and weaknesses and each requires that the developer acquire experience and understanding in order to employ the approach effectively.

In considered the merits and drawbacks of each of the models, it becomes clear that each model operates within its own "rule set" and with those rules come particular constraints. The spirit of exploration suggests that a new approach might be formed by combining elements of several models within the graph of one solution. That is, some of the shortcomings of earlier models such as the Modular/Centric and Flotilla/Satellite models might be addressed by incorporating some elements of the procedural control model – or by combining elements of other models.

For example, in a particular implementation of the Flotilla model, one might arrive at a graph implementation that comprises a main structural group of tables occurrences, plus fifty-five small purpose-specific groups, each with at least one associated utility layout. Some analysis might show that five of the purpose-specific TO groups have ongoing roles, but the remaining fifty can be replaced with a single implementation of a procedural control grouping. Doing so significantly reduces the complexity, redundancies and proliferation of joins of the Graph (and also the number of utility layouts required), while only marginally impacting the scripting model (the Flotilla model already has a significant reliance on scripted support for processes).

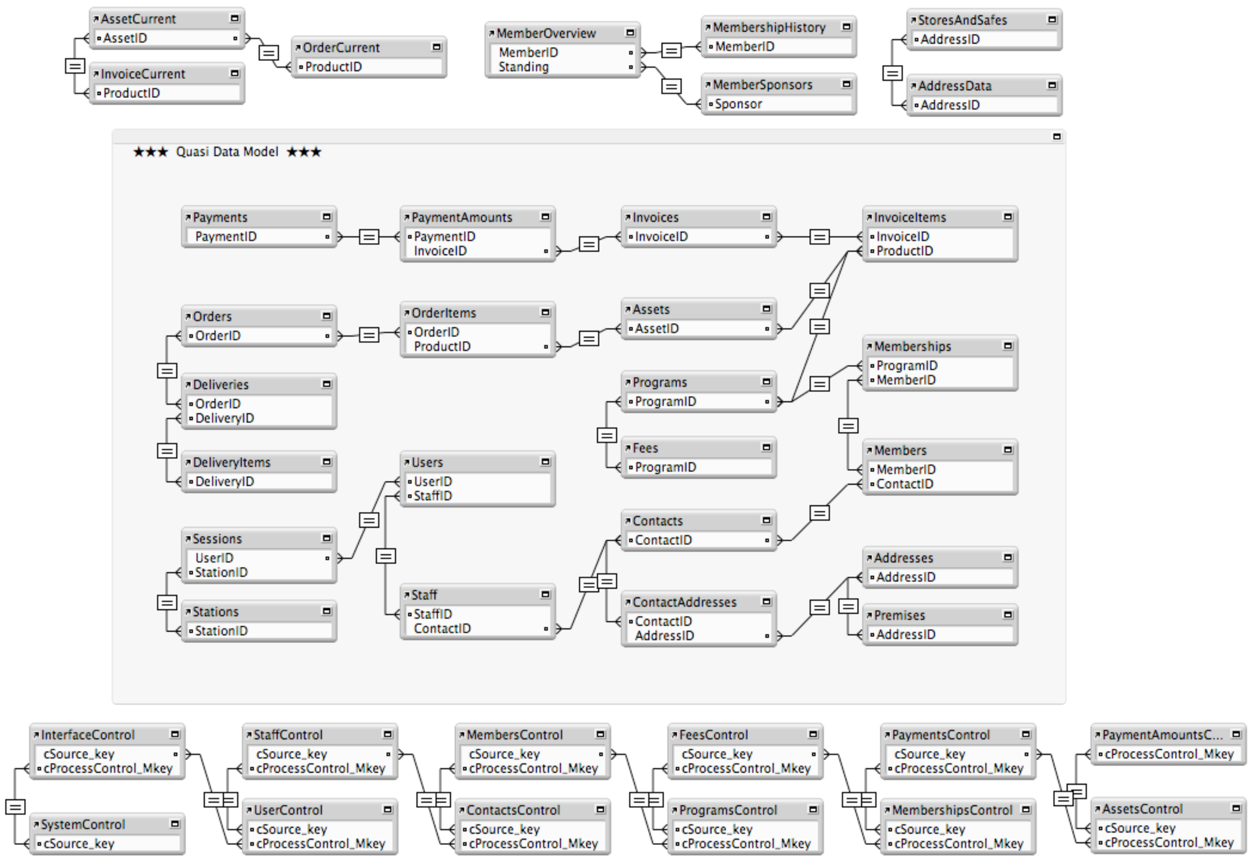

***Figure 7 -** An adaptation of the solution Graph of the Flotilla design from Figure 5, adapted to reduce redundancy by including "Procedural Control like" grouping in place of a majority of the original satellite TOGs.*

At this point, the Graph is no longer serving as the defining principle for the solution or the data model but rather, you are drawing on a variety of organizing principles according to the needs of (and best outcomes for) the solution as a whole. Graph methodologies are dictated by the importance of the solution design rather than the other way around. Although the Hybrid model revisits some of the principles articulated in earlier models, it does so within a different framework where the orthodoxies of those models are no longer a determinant of limits, structures or working processes. Flexibility and adaptation of a variety of known and well-defined methods become the overarching central principles of Graph design.

Having embarked on implementations of Hybrid modeled graphs, you will begin to note that though your graphs contain defined structural elements and segments of order that is recognizable as conforming to the patterns of one of the various clearly delineated models described previously, no comprehensive organizing principle pervades the entire solution. The graph is highly organized, but the organization conforms to a needs model rather than a unified set of principles.

After working for a time with the Hybrid model, it becomes clear that – perhaps without conscious effort or without particular notice, you have in fact begun to drift toward a further distinct model.

Model 7 – The “It’s Really Not That Hard” Model

Once you have become comfortable with the concept of the Hybrid model, it becomes clear that, in fact you are able to complete fully functioning solutions without adherence to any one graph modeling system. Yet, your solutions can be elegant and efficient in operation, perhaps more so than they could be if operating under the guise of any one of the first five orthodoxies of graph management. You have begun to embrace a new reality in the realization that you do not in fact need a graph model at all – as such.

As long as you are able to visit the graph and find a way to use the graph (drawing on any principle known to you) to solve an immediate problem that presents itself during development, your solution will work and development will proceed quickly. Moreover, if you use the method that seems most efficient at each moment, the overall efficiency both of the design and the development process will be enhanced. There is a certain freedom in the abandonment of orthodoxies and models, though the result may appear somewhat messy and even ugly, as shown in Figure 8.

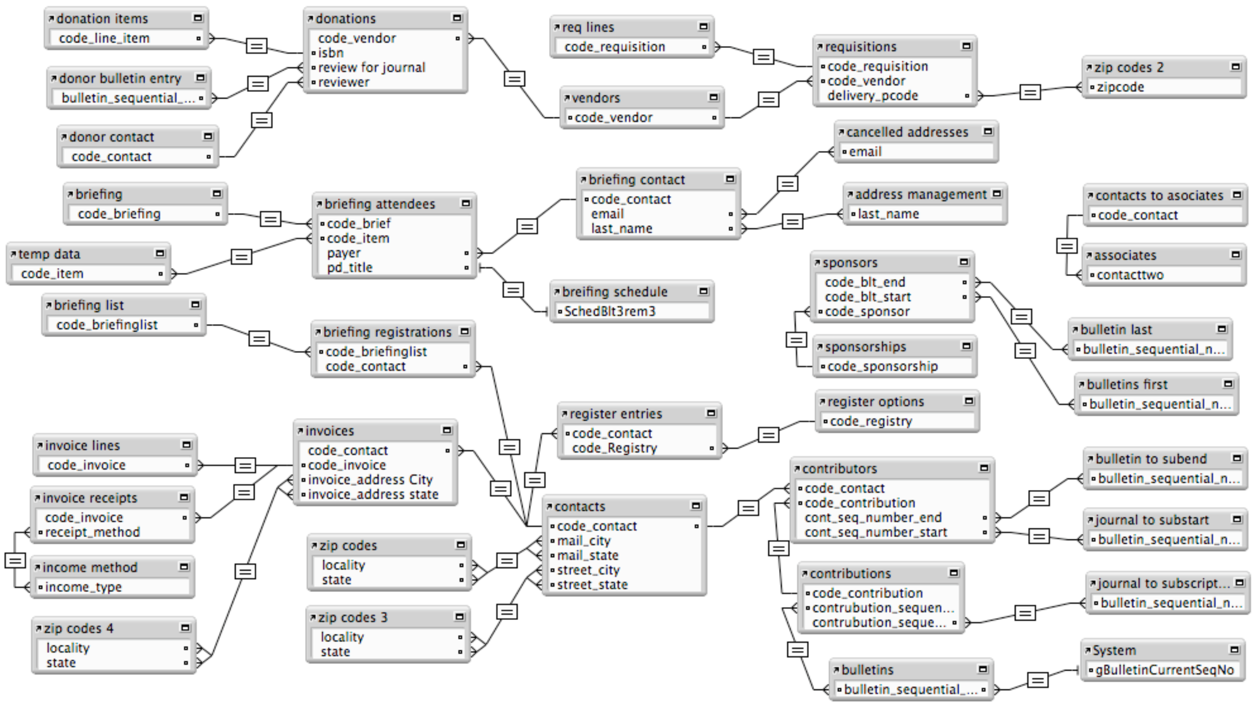

Figure 8 - *A reworking of the Amorphous/Chaotic graph of the same solution featured in Figure 1, applying Really-Not-That-Hard principles.*

At this point, it has become clear that it's not that hard. You can just do whatever works best right now and so long as that is what you always do, the dots will connect up, outcomes will blend and the graph will support a breadth of functional requirements – fluidly and with a certain ungainly grace.

We have now moved beyond abstraction (see model 5) and convenience (see model 6) to a genuinely conceptual model of graph design. A non-model, if you prefer to think existentially. Each time you visit the graph, you will create a microcosm of order with symmetry and purpose and you will achieve the advancement of the functional design of the solution. Furthermore, you will not be constrained to follow the same path to order on your next visit to the graph. Then, as now, you will create structures with their own inherent order serving the purpose of the moment. The over-riding rule and principle of this model is that there are really no (overall graph organizing) rules. You have discovered you no longer need them. Your graphs grow in a cellular and organic way as multiple microcosms of order build up and intersect.

The “it's-really-not-that-hard model” is therefore analogous to a Zen graph technique, wherein all time is now and purpose emerges from experience, order emerges from purpose and guiding purpose (graph doctrine) is immaterial. Your focus is on solving the specific problems of the now, with no thought of adherence to any rigid over-arching organizing principle. Your graphs are composed entirely of microcosms of order. Your solution works (beautifully), so you are not disturbed by the fact that viewed as a whole, your Relationships Graphs do not have a coherent nor even cohesive appearance. Or so you tell yourself.

All is well, except for the nagging feeling of concern about the lack of a coherent organizing principle for the Graph. But we do not really need it, right? Or do we?!

Model 8 – The Recognition of Human Frailty Model

The discovery of the Zen principle of graph management from model 7 is indeed a significant step forward. Many of the complexities and requirements of earlier conceptions of Graph management drop away and you have embraced a delicate yet powerful regime of fluidity and elegance. However, something is still not right. No matter how well your solutions work – and however efficient your development processes, however powerful your conception of the tools at your command, you cannot escape the reality that the lack of an overall order in the graph makes you uncomfortable. You know it should not, but it does.

It is at this point that it becomes necessary to act in the recognition of the flaws and failings of your own humanity, to impose some over-arching order to the graph. Even though you recognize that the exercise is in part cosmetic; you do not really need a unified rule of order on the graph to create or support the data model or functional requirements of your solutions. You remain committed to the rule that there are no rules, having embraced the liberation of the Zen principles of model 7. In any event, it will make you feel better (and you will cringe less) when other developers see your graphs. That alone has to be worth something!

The Recognition of Human Frailty model emerges as you maintain your commitment to the framework-free discipline of model 7 – accepting that all time is now and that the graph need only be composed of small pieces of order – but you adopt a variety of methods to create a appearance of overall order to the graph as a whole. These methods will make it easier to find things (use of labels and colors) and perhaps easier to see what is where (repositioning of elements so that lines do not cross and rows are neat and ordered) but mostly, they will make you feel better. Look at the example shown in Figure 9 and you will feel better already.

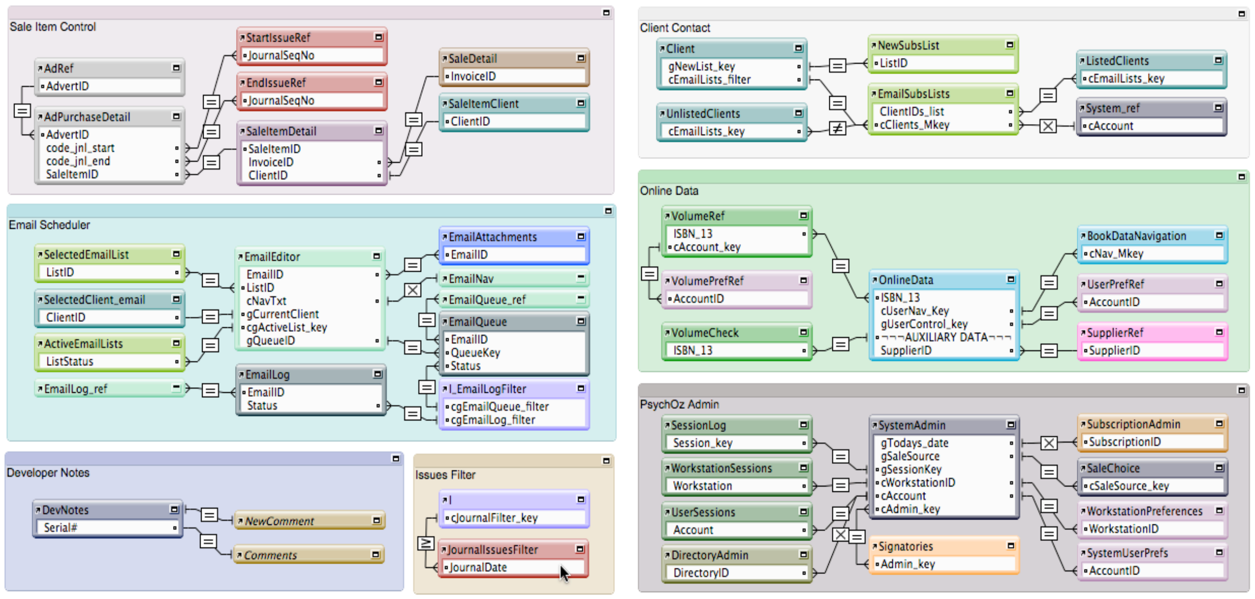

***Figure 9** - In recognition of human frailty, a Graph that is really-not-that-hard can be coaxed into a semblance of order and given a less jarring appearance.*

And it is better. It takes a little additional time to create an impression of overall order that encapsulates your microcosms of order – but it becomes clear that things are easier to find and that saves you time. It is with a new lightness of spirit, therefore, that you embrace each new day, confident that you are free (of imposed disciplines and orthodoxies), but not *too* free.

Model 9 – The Transcendental Model

Unfortunately, not all is what it seems, as there remains something that has been overlooked. One cannot truly be without rules while one continues to adhere to the rule that there shall be no rules. Like the prohibition against circular references on the graph, this presents a paradox that cannot go unresolved indefinitely. You are not as free as you thought you were.

Fortunately, at this point, a brief meditation is all that is required to arrive at the ultimate answer. To truly transcend a rule-driven belief, you must abandon even the rule that there will be no rules and allow yourself to have rules when they serve the purpose of the moment – or when you just feel like it (after all, you have recognized human frailty at model 8). This leads you to the transcendental model of Graph management.

Having reached transcendence, you are free to embrace those principles which serve the needs of the solution (somewhat as you did when using the Hybrid model) - but you preserve the insight that it is really not that hard, that all time is now and that human frailties must be recognized. So you allow yourself the luxury of rules that seem appropriate, that help you to develop effectively and serve the best outcomes for the solution, while avoiding adherence to orthodoxies or thought-prison regimes. Your Graphs are well organized in a way that aids comprehension and presents an aesthetically pleasing unified whole. They will undoubtedly contain microcosms of order, because that is the way the universe chooses to organize itself (well, FileMaker, at any rate) and they may contain semblances of other over-arching structures – sometimes. At other times your graph may amuse or surprise you – or lead you toward abstract meditations, as shown somewhat whimsically in Figure 10

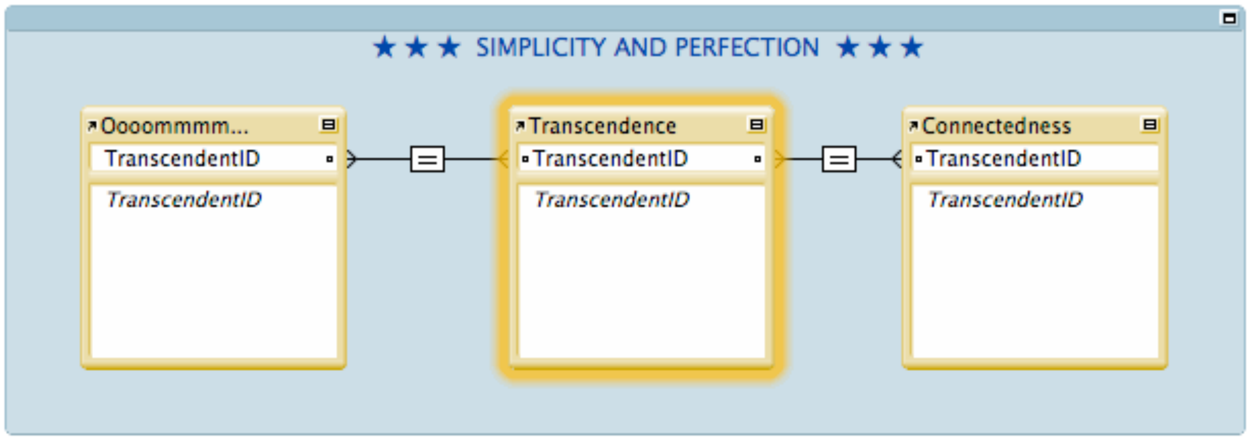

Figure 10 *- The FileMaker Relationships Graph as a meditation on the oneness of all being.*

The transcendental model of Graph management is not something every developer should aspire to. It is not necessary. Each of the preceding models described in this paper can be called into service in some way, to solve some groups of solution design problems. If you have found that any of the other models routinely works well for you, you should allow yourself the latitude to embrace what works, looking further afield only when necessity drives you to do so.

Know, however, that all nine steps toward enlightenment are contained within the last step, so that you can reach for it if the occasion demands. Until then, thank you for joining this FileMaker pilgrimage of discovery.

Conclusion

If you work on a number of FileMaker solutions – whether of your own design or the work of others – you will encounter various approaches to graph management. Few developers continue in exactly the same mode as they start out. Whether you choose to consider one approach the "right" approach (and all others inherently flawed) or whether you choose to remain open to a variety of strategies, it will be helpful to you to be aware of the differing models described here, and the different frames of reference that apply to each of them.

Inevitably, as your preferences, skills and perceptions are adjusted over time and with experience, some changes will take place in the ways you conceptualize your work with the relationships graph. Consequently, over time, with rework and extension, many solutions tend towards a Hybrid graph model – one that incorporates elements of design and concept from different points along the evolutionary journey.

The most important message of this paper is that rigid adherence to a single approach to the Relationship Graph is neither necessary nor desirable. That acquiring a broadly based set of perceptions and skills will afford you the flexibility to deal with situations as they arise and to choose techniques to fit the special needs of each case and each solution.