http://www.nightwing.com.au/FileMaker/ CobaltSky@nightwing.com.au

FileMaker Pro™ Matrix Data Modelling [for FileMaker v6 and earlier]

Paper prepared by R J Cologon PhD

October 2002

Background

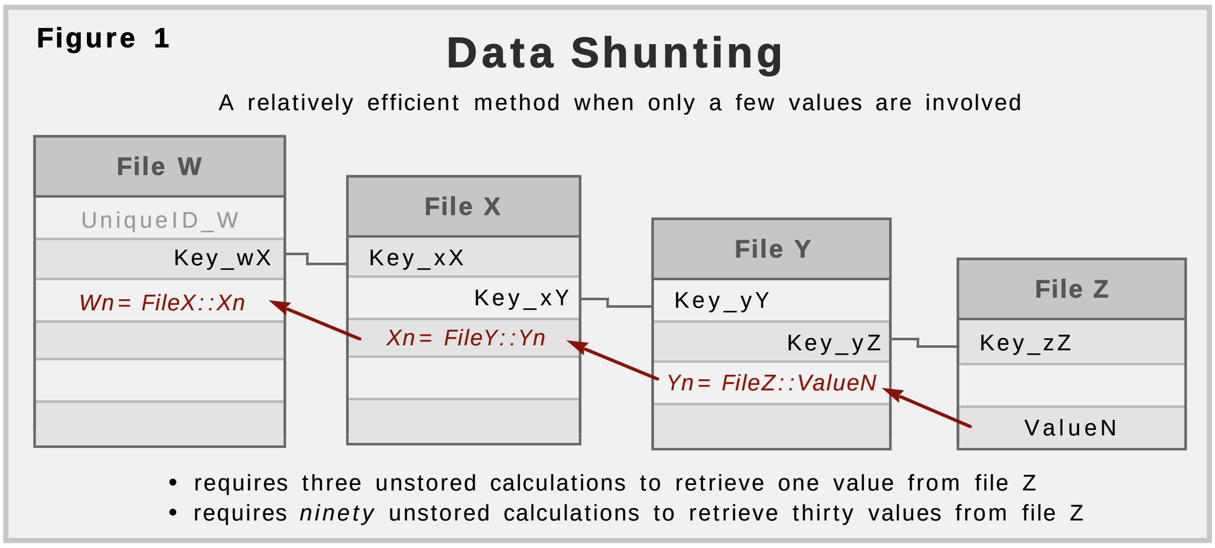
FileMaker Pro™ provides immediate and highly accessible means by which data from a related file can be accessed. However when it comes to accessing data which is related to a related file, the plot thickens considerably.

In multi-layered relational file structures, it is often necessary to access data from a table in a location which is several relationships away from the point where it is to be used. For example in a school solution, student names and details may be in a main table, the subjects the students are enrolled in may be in a related table, and attendance records for each subject may be in an attendance table which relates to the subject table. There will be occasions when it is desired to access attendance records against each student record in the main student file.

The most commonly used and understood technique in FileMaker Pro programming for accessing data from a remote table, is the creation of unstored calculation fields in each of the files in between the location where the data are stored and the file where they are needed.

This technique is sometimes referred to as 'shunting' or 'tunnelling'. It may be an effective solution in some cases, however it can cause the field definitions of files to become bloated with large numbers of unstored calculations serving the sole purpose of providing conduits for data being passed up (or down) the line to other files. A simple implementation is shown in the diagram at figure 1.

Figure 1


Data Shunting
A relatively efficient method when only a few values are involved
File W
UniqueID_W
Key_wX
Wn= FileX::Xn
File X
Key_xX
Key_xY
Xn= FileY::Yn
File Y
Key_yY
Key_yZ
Yn= FileZ::ValueN
File Z
Key_zZ
ValueN
• requires three unstored calculations to retrieve one value from file Z
• requires ninety unstored calculations to retrieve thirty values from file Z

Data shunts may do the job relatively effectively when the number of fields is small. As indicated on the diagram, the number of unstored calculations required is a multiple of the number of values to be retrieved and the number of files between the source and destination. This potential deluge of unstored calculations can be cumbersome and inefficient - and may also make it difficult to focus on the true role and purpose of the data within each file in a solution.

An Alternative Approach

It may be of interest that 'shunting' is not the only approach by which a value may be accessed from a file which is several relationships away in the relational structure. An alternative technique based on the implementation of data matrices provides a workable alternative which may be preferable in some situations.

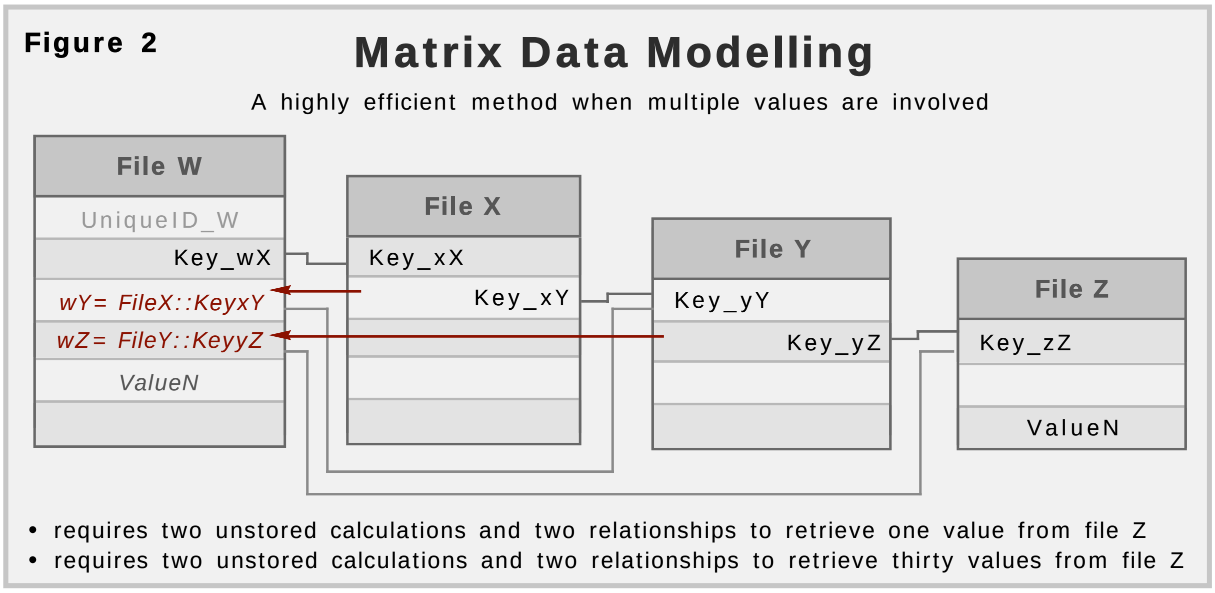
For the purposes of explanation, imagine a relational structure in which four files (let's call them W, X, Y and Z) are related in a linear fashion, so that file X is related to file W, file Y is related to file X and file Z is related to file Y. The question is then how, within the W->X->Y->Z structure, a value in file Z can be referenced directly against appropriate related values in file W.

The Basic Technique

The alternative method, a matrix model, works like this:

- You have a direct chain of relationships from file W to file X, from file X to file Y, from file Y to file Z.
- You have a field in W which links it to a corresponding field in X. For clarity, lets call the two fields Key_wX and Key_xX.
- X has a relationship to Y based on some other key which is stored in both X and Y. Lets call them Key_xY and Key_yY.
- To get at a field in Y directly from W, first create an unstored calculation field in W called Key_wY which picks up the value for Key_xY via the existing relationship to X. Then create a relationship from W to Y which matches Key_wY directly to Key_yY.
- Finally, to get at a field in Z directly from W, create an unstored calculation field in W called Key_wZ which picks up the value for Key_yZ via the relationship now established to Y. Then create a third relationship within W linking it to Z by matching Key_wZ directly to Key_zZ.

Figure 2


Matrix Data Modelling
A highly efficient method when multiple values are involved
File W
UniqueID_W
Key_wX
wY= FileX::KeyxY
wZ= FileY::KeyyZ
ValueN
File X
Key_xX
Key_xY
File Y
Key_yY
Key_yZ
File Z
Key_zZ
ValueN
• requires two unstored calculations and two relationships to retrieve one value from file Z
• requires two unstored calculations and two relationships to retrieve thirty values from file Z

As can be seen at figure 2 (above) when only a single value is to be retrieved the matrix data model is (arguably) no more efficient than the shunting method, in that it requires two unstored calculations and two relationships to achieve the same outcome. However as soon as there are two or more values to be retrieved, the matrix model is markedly more efficient, still requiring only two unstored calculations and two relationships as compared to six or more unstored calculations to achieve the same result with a shunting technique.

When the number of values to be retrieved from other files across a relational solution is significant, considerable efficiencies are to be gained by using a matrix-based technique. Moreover when the file size is large, when further calculations are dependent on the retrieved values, and/or when portal displays are required to include composite calculation results (eg calculations which draw on values from more than one file) a matrix model has the potential to offer considerable performance advantages.

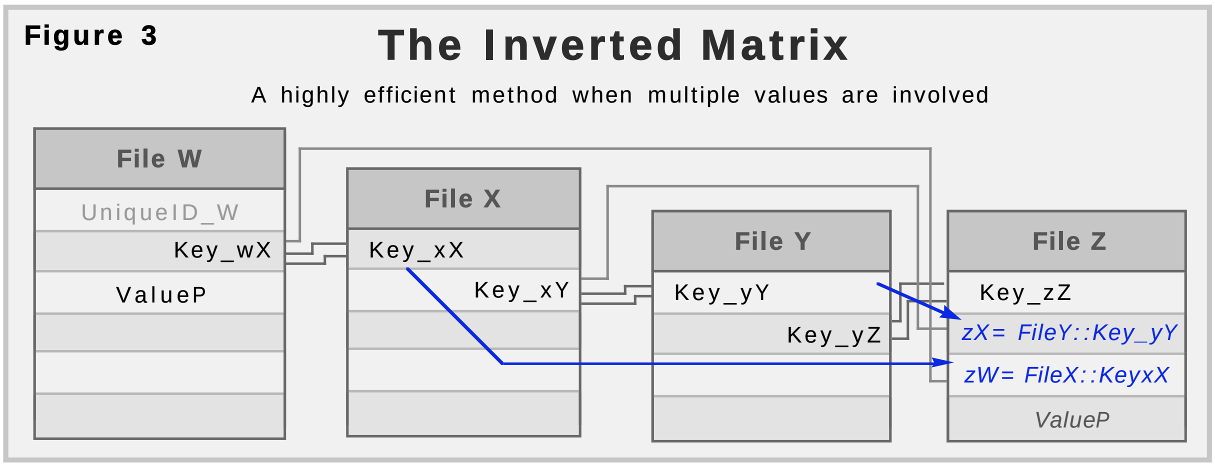
Whilst the example discussed here relates to four files in a linear relationship, the technique can be used in the reverse direction (eg to bring values from W into the Z file), can be established simultaneously across different branches of a complex structure, and can be implemented in bipolar formations to allow same tier values within different branches of the structure to be related.

The Inverted Relationship Matrix

Implicit within the principles of matrix modelling is the ability to establish logical connections in more than one direction. Thus in our data structure in which the four files W, X, Y and Z are related, a value in W can be accessed against corresponding (related) values in file Z in the following way:

- You have a field in Z which is the ‘foreign key’ for the relationship from Y to Z. In the preceding example, this key field was referred to as Key_zZ.
- A relationship can therefore be established in Z which uses the Key_Key_zZ field as the basis of an ‘upwards’ link to Y, by matching Key_zZ to the Key_yZ field in Y.
- Via this new relationship, it is then possible to create an unstored calculation field in Z which references the field Key_yY. This in turn can be used as a primary key in Z for a relationship to X which matches Key_zX in Z to Key_xY in X.
- Finally, to get at a field in W directly from Z, create an unstored calculation field in Z called Key_zW which picks up the foreign key value for Key_xX via the relationship now established to X. Then create a third relationship within Z linking it to W by matching Key_zW directly to Key_wX.

A rudimentary form of the inverted matrix is illustrated at figure 3 (below).



At this point it is possible to see that data can be passed both up and down the tiers of the relational data structure using direct referencing and without the aid of 'shunts' or 'tunnelling'.

Bipolar Matrix modelling

The full significance of the ability to establish matrix-based links in both directions, becomes clear within the context of a three dimensional model where multiple branches of data relationships can be inter-related by combining techniques for data matching working in both directions at simultaneously.

To use an example, lets suppose that file W holds student personal details, file X holds course enrolments. file Y holds subject selections and file Z holds Assignments set.

Now let's suppose that File W also has files V and U associated with it, where V holds data about halls of residence that students are living in, and U holds details of the peer support groups operating within each hall.

The challenge of determining the dates of meetings of peer support groups students who have been set a particular assignment are in requires that a relationship between Z and U be established. Depending on the nature of the data and keys upon which the relationships have been established, a direct relationship may be possible - but if not, bipolar matrix modelling is one means by which the required related keys can be acquired in Z to facilitate a direct relationship to U.

To bring this about, the following considerations apply:

- You have a field in W which links it to a corresponding field in V. In line with previous examples these fields can be referred to as Key_wV and Key_vV.
- V has a relationship to U based on some other key which is stored in both V and U which we are referring to as Key_vU and Key_uU.
- To get at a field in U directly from W, we will require an unstored calculation field in W called Key_wU which picks up the value for Key_vU via the existing relationship to V. Thus a relationship can be established from W to U which matches Key_wU directly to Key_uU.
- Finally, to get at a field in U directly from Z, create an unstored calculation field in Z called Key_zU which picks up the foreign key value for Key_wU via the inverted relationship previously decsibed which linked Z directly to W. Then create an additional relationship within Z linking it to U by matching Key_zU directly to Key_uU.

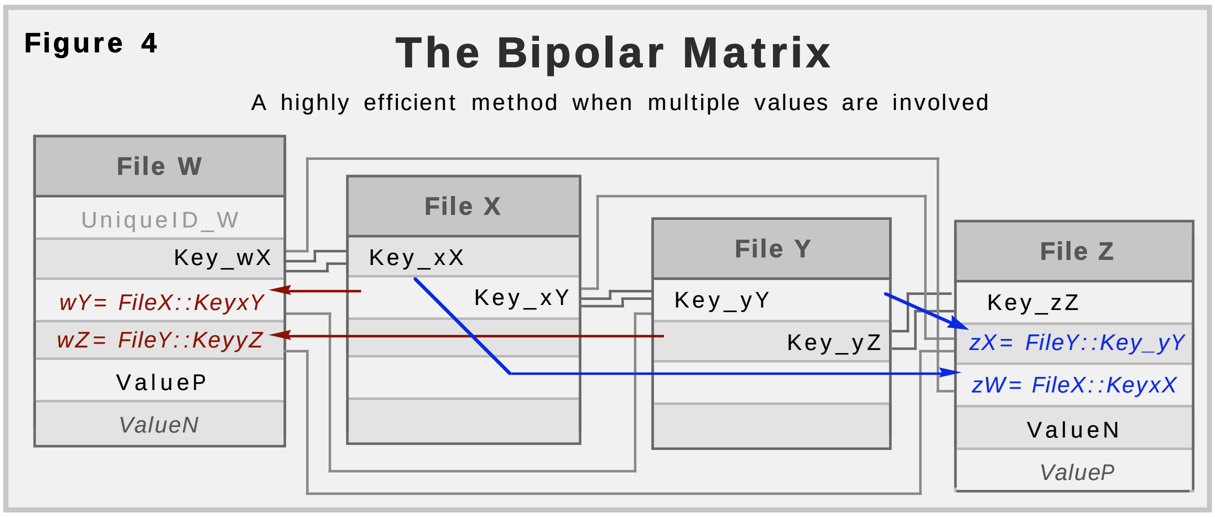
Not infrequently, however, it is desirable that values be retrieved in both directions within the same group of files. Thus the principles of modelling and inverted modelling may be necessary within the same subset of files, to pass a value across the data structure, while simultaneously passing values back in the other direction.

To illustrate, this, let's assume that in the first example of a matrix data model (provided on page 2 above) it is necessary to pass a value 'P' from W down to Z at the same time as retrieving the value 'N' from Z to W. To achieve this, it is possible to implement normal and inverted matrices simultaneously (ie overlaid upon one another within the same relational file structure). The resulting bipolar structure provides the potential for fully integrated data pathways throughout a solution.

This technique facilitates the passage of data in a way which would be almost beyond contemplation if using data shunting methodologies.

A rudimentary example of a bipolar matrix, bringing together the matrices illustrated in figures two and three (above) is provided at Figure 4.

Figure 4



Thus a three dimensional logical 'data map' can be assembled, based on a network of unstored key fields, through which related data can be drawn directly to any point in the data flow via a single step data call (based on the matrix key system) rather than via lengthy lines of data shunts.

An option to be considered in each case:

This approach is the core of matrix data modelling. Relationships based on these principles can be created from within each file and used to validate, summarise, verify and calculate with reference to the values local to that file, making it possible to simulate regressions within multi-dimensional data matrices.

Whilst it is by no means the purpose of this paper to suggest that the techniques described are a perfect or even a preferred solution for all cases, they nevertheless offer a valuable alternative which should be considered in the formulation of any complex data structure which is to be supported in a FileMaker Pro solution.

Prepared by:
R J Cologon, PhD
NightWing Enterprises
Melbourne, Australia
CobaltSky@nightwing.com.au
http://www.nightwing.com.au/FileMaker/