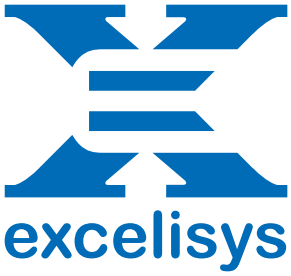
Virtually Possible by Michael Rocharde



Since its earliest days, FileMaker™ developers have come up with amazing workarounds to solve problems that could not be solved using the built in tools. As FileMaker™ developed and became more powerful, many of those workarounds disappeared and were able to be done natively. However there have always been, and will continue to be, developers who push the envelope. Andy Persons of Excelisys is one of those developers and is widely acknowledged as coming up with the drag-and-drop methodology that is now in use by FileMaker™ developers all over the world. Another such developer is Bruce Robertson who pioneered a technique called Virtual List, which may well be one of the most useful and versatile of all time.

Virtual List is a little hard to explain. In very simple terms, it consists of building variables (or global fields) with values from different fields and table occurrences with a carriage return separating each entry. The developer writes a script that will go to a layout and to a record, set specific values into the variables and end it with a carriage return. Then the script goes to the next record and does the same thing over and over again. After that it may close the current window or go to a different one and grab different values that are again written into the variables. Once all of the required values have been written into the variables, the script goes to a utility table that has 1000 (or more) pre-created records and sets the values from the variables into global fields. Calculated fields then parse that data into individual records which are then used to generate a report or populate a portal, etc.

So that's it in a nutshell. It is not as difficult or as complicated as it sounds but the only way to really get to understand it is to use it. I guarantee that you won't get it right on the first attempt but, eventually, you will and you will be staggered by it!

But what do we use it for? That's a very good question and the best answer that I can give you is that if you are aware of the technique, sooner or later you'll come up with something that you need to do for which VirtualList is the perfect solution.

In this article, I'm going to deal with a report generation problem that couldn't be done natively with FileMaker™s layout tools. The scenario is a quoting system with a Master Quote which could have multiple sub quotes and each sub quote could have multiple line items. So far, this could all be done by generating the quote (report) from the line items table using sub-summaries to display the sub quotes and master quote information. However, this particular situation was far more complicated as depending on the type of sub quote; the items had to be displayed and priced differently and some of them had to display a header line while others did not. If the items were for a conveyor, only the first line item had to be numbered but if not, all line items had to be. Additionally, if there were different quantities, those items had to be flagged with the price for each even thought the total would be different. As I said, it was a very complicated set of instructions to generate a perfectly formatted quote document that could be sent out.

Let's start by looking at the differences in the way that a sub quote is displayed depending on whether it is being quoted as a conveyor or as parts.

We are pleased to offer the following quotation:

Item #	Category	Model/Part #	Description	Qty	Price
1	**CONVEYOR**	**20-02042-R-N U1 RU-25A-20-11**		**2**	**$2,367.00** EA
	Conveyor	20-02042	Lite Series Module -- Size: 1.90" wide x 42" nominal length. 2.50" high, hard black anodized aluminum frame with integral .70" high side guides		
	Belt	U1-01.87-042			
	Drive Package	RI-025A-20-11			
	Bedplate	30-1502-042	.015" Thick UHMW Bedplate Cover for a 1.9 x 42" conveyor		
	Extrusion	40-7273-042	Standard 42" LP T-Slot Extrusion, Includes mounting Screws. Sold Individually - universal Left / Right side. Requires modification for 60/90 W side mount motor		
	Conveyor	20-7302-xx	1.90" Lite Drive Pulley Assembly, See Original Conveyor SO to Determine Drive Shaft Reqd. - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw		
	Belt	U1-01.87-042	Conveyor Belt -- U1 style, white urethane, 2" X 42". (Medium friction surface, Superior cut and abrasion resistance)		
	Conveyor	20-02042	Lite Series Module -- Size: 1.90" wide x 42" nominal length. 2.50" high, hard black anodized aluminum frame with integral .70" high side guides		
		Notes for #1	This space is for any notes pertaining to the conveyor		

The above example shows how it is displayed as a conveyor. Key things to notice here are that there is a header line (all in bold text) which displays the conveyor model # and has an item # (with all the items on that conveyor not having an item #. Finally the pricing shows the total for all of the individual items priced by each and does not show the price of each line item.

If we now look at how it is displayed when quoted as parts:

We are pleased to offer the following quotation:

Item #	Category	Model/Part #	Description	Qty	Price
1	Conveyor	20-02042	Lite Series Module -- Size: 1.90" wide x 42" nominal length. 2.50" high, hard black anodized aluminum frame with integral .70" high side guides	1	$1,000.00
2	Belt	U1-01.87-042		1	$60.79
3	Drive Package	RI-025A-20-11		1	$643.91
4	Bedplate	30-1502-042	.015" Thick UHMW Bedplate Cover for a 1.9 x 42" conveyor	1	$40.01
5	Extrusion	40-7273-042	Standard 42" LP T-Slot Extrusion, Includes mounting Screws. Sold Individually - universal Left / Right side. Requires modification for 60/90 W side mount motor	1	$29.09
6	Conveyor	20-7302-xx	1.90" Lite Drive Pulley Assembly, See Original Conveyor SO to Determine Drive Shaft Reqd. - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw	1	$30.87
7	Belt	U1-01.87-042	Conveyor Belt -- U1 style, white urethane, 2" X 42". (Medium friction surface, Superior cut and abrasion resistance)	1	$56.11
8	Conveyor	20-02042	Lite Series Module -- Size: 1.90" wide x 42" nominal length. 2.50" high, hard black anodized aluminum frame with integral .70" high side guides	1	$506.22

you'll notice that there is no header line and that each item has an item # and a price and there is no grand total.

Now we'll look at the entire body of the quote so you can see it in its entirety.

Virtually Possible by Michael Rocharde

We are pleased to offer the following quotation:

Item #	Category	Model/Part #	Description	Qty	Price
1	**CONVEYOR**	**20-02042-R-N U1 RU-25A-20-11**		**2**	**$2,367.00** EA
	Conveyor	20-02042	Lite Series Module -- Size: 1.90" wide x 42" nominal length. 2.50" high, hard black anodized aluminum frame with integral .70" high side guides		
	Belt	U1-01.87-042			
	Drive Package	RI-025A-20-11			
	Bedplate	30-1502-042	.015" Thick UHMW Bedplate Cover for a 1.9 x 42" conveyor		
	Extrusion	40-7273-042	Standard 42" LP T-Slot Extrusion, Includes mounting Screws. Sold Individually - universal Left / Right side. Requires modification for 60/90 W side mount motor		
	Conveyor	20-7302-xx	1.90" Lite Drive Pulley Assembly, See Original Conveyor SO to Determine Drive Shaft Reqd. - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw		
	Belt	U1-01.87-042	Conveyor Belt -- U1 style, white urethane, 2" X 42". (Medium friction surface, Superior cut and abrasion resistance)		
	Conveyor	20-02042	Lite Series Module -- Size: 1.90" wide x 42" nominal length. 2.50" high, hard black anodized aluminum frame with integral .70" high side guides		
		Notes for #1	This space is for any notes pertaining to the conveyor		
2	**CONVEYOR**	**20-04030**		**1**	**$536.66**
	Stand	30-7054	Leg Stand 30" to 36" Adjustable Height to Top of Belt. (Includes Frame Mounting Hardware.)		
	Stand	30-7061	Cross Brace for Leg Stands, 37" - 620". (Includes Mounting Hardware.)		
	Belt	P3-02.87-048			
	Pulleys	20-7312-50	11.90" Lite Drive Pulley Assembly, 1/2" Bore. - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw		
	Pulleys	20-7312-xx	11.90" Lite Drive Pulley Assembly, See Original Conveyor SO to Determine Drive Shaft Reqd. - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw		

Please Note: Due to the installation configuration by others, installation of SAFETY GUARDING is the responsibility of the end user.
Changes to quotations must be confirmed in writing

Item #	Category	Model/Part #	Description	Qty	Price	
	Pulleys	20-7312-xx	11.90" Lite Drive Pulley Assembly, - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw			
	Conveyor	30-7136-030	Universal Bed Plate, 3.900" Wide x 30" Nominal Length (actual length = 25.426") - Hole spacing @ 12.000"			
	Pulleys	40-5010	Timing Pulley, 16T 5M "Slave", .500 Bore with Flange			
		Notes for #2	This space is for notes pertaining to the accessories			
3	Stand	30-7051	Leg Stand 13" to 16" Adjustable Height to Top of Belt. (Includes Frame Mounting Hardware.)	1	$134.66	
4	Stand	30-7052	Leg Stand 16" to 22" Adjustable Height to Top of Belt. (Includes Frame Mounting Hardware.) -	2	$134.66	EA
5	Panasonic	30-7221	Panasonic AC Variable Speed Inverter/Controller Assembly 25/40 Watt, 100 V. - (Includes, Inverter/Controller # M1G4A1V1X, Enclosure, Cables, Etc., DVOP13802 Cable)	3	$423.63	EA
6	Pulleys	20-7302-xx	1.90" Lite Drive Pulley Assembly, See Original Conveyor SO to Determine Drive Shaft Reqd. - Includes: Pulley, Set Screw, Bearing, Bearing Adaptor, Tension Washer and Screw	1	$34.25	
7	Stand	30-7060	Cross Brace for Leg Stands, 12" - 36". (Includes Mounting Hardware.)	1	$41.02	
8	Extrusion	40-7273-060	Standard 60" LP T-Slot Extrusion, Includes mounting Screws. Sold Individually - universal Left / Right side. Requires modification for 60/90 W side mount motor	1	$33.86	
9	Guides	20-7220-060	1/4 x .50" High Alum. Lite Series Fixed Side Guide, 60" nominal conveyor length, (Includes mounting Hardware)	1	$62.24	

In this particular example, there are three sub quotes; two of which are being quoted as a conveyor and the third is quoted as parts. BTW the pricing is different for items that are sold as parts as opposed to being sold as part of a conveyor so when we build the quote, we also have to take that into account.

So let's deconstruct the script that runs all of this.

Script Name: BuildVLQuote

```
Allow User Abort [On]
Set Error Capture [On]
Perform Script ["Clear_for_VL"]
Set Field [MasterQuote::QuoteDate; Get(CurrentDate)]
Go to Related Record [Show only related records; From table: "Quote4Master"; Using layout: <Current Layout>]
Go to Record/Request/Page [First]
Set Variable [$mquote; Value:MasterQuote::_PKID]
Set Variable [$$file; Value:MasterQuote::MasterQuoteNumber&".pdf"]
Set Variable [$item#; Value:1]
Loop
  If [Quotes::PriceAs="Conveyor"]
    Set Field [Admin::G_Item#; Admin::G_Item#&$item#&"¶"]
    Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
    If [not IsEmpty(Quotes::Q_Series)]
      Set Field [Admin::G_Category; Admin::G_Category&TextStyleAdd("CONVEYOR";Bold)&"~¶"]
      Set Field [Admin::G_Part#; Admin::G_Part#&TextStyleAdd(Quotes::C_ModelNumber&"~¶";Bold)]
    Else
      Set Field [Admin::G_Category; Admin::G_Category&"¶"]
      Set Field [Admin::G_Part#; Admin::G_Part#&"¶"]
    End If
    Set Field [Admin::G_ItemName; Admin::G_ItemName&"~¶"]
    If [Quotes::ConveyorQuantity>1]
      Set Field [Admin::G_Each; Admin::G_Each&"EA¶"]
    Else
      Set Field [Admin::G_Each; Admin::G_Each&"¶"]
    End If
    Set Field [Admin::G_ItemQty; Admin::G_ItemQty&TextStyleAdd(Quotes::ConveyorQuantity&"~¶";Bold)]
    Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&TextStyleAdd(Quotes::Conveyor_Price&"~¶";Bold)]
    If [Count(QuoteItems::Serial_#)>0]
      Go to Related Record [Show only related records; From table: "QuoteItems"; Using layout: "QuoteItems" (QuoteItems); New window]
      Sort Records [Restore; No dialog]
      Go to Record/Request/Page [First]
      Loop
        Set Field [Admin::G_Item#; Admin::G_Item#&"~¶"]
        Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
        Set Field [Admin::G_Category; Admin::G_Category&QuoteItems::Category&"~¶"]
        Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(QuoteItems::ProductDescription;"¶";" - ")&"~¶"]
        If [QuoteItems::Item_Type="product"]
          Set Field [Admin::G_Part#; Admin::G_Part#&QI_Product::PDTNumber&"~¶"]
        Else
          Set Field [Admin::G_Part#; Admin::G_Part#&QI_Component::InHouseManuNumber&"~¶"]
          Set Variable [$$part#; Value:$$part#&QI_Component::InHouseManuNumber&"¶"]
        End If
        Set Field [Admin::G_ItemQty; Admin::G_ItemQty&"~¶"]
        Set Field [Admin::G_Each; Admin::G_Each&"¶"]
        Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&"~¶"]
        Go to Record/Request/Page [Next; Exit after last]
      End Loop
      Close Window [Current Window]
      If [not IsEmpty(Quotes::Notes4Cust)]
        Set Field [Admin::G_Item#; Admin::G_Item#&"~¶"]
        Set Field [Admin::G_Category; Admin::G_Category&"~¶"]
        Set Field [Admin::G_Part#; Admin::G_Part#&TextStyleAdd("Notes for #"&$item#&"¶";Bold)]
        Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(Quotes::Notes4Cust;"¶";" - ")&"~¶"]
        Set Field [Admin::G_ItemQty; Admin::G_ItemQty&"~¶"]
        Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&"~¶"]
        Set Field [Admin::G_Each; Admin::G_Each&"¶"]
        Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
      End If
      Set Variable [$item#; Value:$item#+1]
      Go to Record/Request/Page [Next; Exit after last]
    Else
      Go to Record/Request/Page [Next; Exit after last]
    End If
```

```
Else If [Quotes::PriceAs="parts" and Count(QuoteItems::Serial_#)>0]
  Go to Related Record [Show only related records; From table: "QuoteItems"; Using layout: "QuoteItems" (QuoteItems); New window]
  Sort Records [Restore; No dialog]
  Go to Record/Request/Page [First]
  Loop
    Set Field [Admin::G_Item#; Admin::G_Item#&$item#&"~¶"]
    Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
    Set Field [Admin::G_Category; Admin::G_Category&QuoteItems::Category&"~¶"]
    Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(QuoteItems::ProductDescription;"¶";" - ")&"~¶"]
    If [QuoteItems::Item_Type="product"]
      Set Field [Admin::G_Part#; Admin::G_Part#&QI_Product::PDTNumber&"~¶"]
      Set Variable [$$part#; Value:$$part#&QI_Product::PDTNumber&"¶"]
    Else
      Set Field [Admin::G_Part#; Admin::G_Part#&QI_Component::InHouseManuNumber&"¶"]
      Set Variable [$$part#; Value:$$part#&QI_Component::InHouseManuNumber&"¶"]
    End If
    Set Field [Admin::G_ItemQty; Admin::G_ItemQty&QuoteItems::QuantityOrdered&"~¶"]
    If [QuoteItems::QuantityOrdered>1]
      Set Field [Admin::G_Each; Admin::G_Each&"EA¶"]
    Else
      Set Field [Admin::G_Each; Admin::G_Each&"¶"]
    End If
    Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&QuoteItems::C_price&"~¶"]
    Set Variable [$item#; Value:$item#+1]
    Go to Record/Request/Page [Next; Exit after last]
  End Loop
  Close Window [Current Window]
  Go to Record/Request/Page [Next; Exit after last]
  Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
Else
  Go to Record/Request/Page [Next; Exit after last]
End If
Set Variable [$rec#; Value:$rec#+1]
End Loop
Perform Script ["GenerateQuote"]
```

Now we'll break it down

Script Name: BuildVLQuote

```
Allow User Abort [On]
Set Error Capture [On]
Perform Script ["Clear_for_VL"]
Set Field [MasterQuote::QuoteDate; Get(CurrentDate)]
Go to Related Record [Show only related records; From table: "Quote4Master"; Using layout: <Current Layout>]
Go to Record/Request/Page [First]
Set Variable [$mquote; Value:MasterQuote::_PKID]
Set Variable [$$file; Value:MasterQuote::MasterQuoteNumber&".pdf"]
Set Variable [$item#; Value:1]
```

The first thing we need to do is clear out all of the values in our global fields, so we have a script for that purpose:

Script Name: Clear_for_VL

```
Set Field [Admin::G_Item#; ""]
Set Field [Admin::G_Record#; ""]
Set Field [Admin::G_ItemQty; ""]
Set Field [Admin::G_ItemName; ""]
Set Field [Admin::G_ItemPrice; ""]
Set Field [Admin::G_Each; ""]
Set Field [Admin::G_Category; ""]
Set Field [Admin::G_Part#; ""]
New Window []
Go to Layout ["VL" (VL)]
Set Field [VL::G_Item#; ""]
Set Field [VL::G_ItemName; ""]
Set Field [VL::G_ItemQty; ""]
Set Field [VL::G_ItemPrice; ""]
Set Field [VL::G_Each; ""]
Set Field [VL::MQuote#; ""]
Set Field [VL::G_Part#; ""]
Set Field [VL::G_Category; ""]
Set Field [VL::G_record#; ""]
Close Window [Current Window]
```

Once we've done that we make sure that we have only got the sub quotes for the current master quote and that we are in the first record. We do that by going to the related records and then by going to the first record; admittedly the second step is pretty much overkill as you should always end up in the first record but I'd rather be 100% sure.

```
Set Field [MasterQuote::QuoteDate; Get(CurrentDate)]
Go to Related Record [Show only related records; From table: "Quote4Master"; Using layout: <Current Layout>]
Go to Record/Request/Page [First]
Set Variable [$mquote; Value:MasterQuote::_PKID]
Set Variable [$$file; Value:MasterQuote::MasterQuoteNumber&".pdf"]
Set Variable [$item#; Value:1]
```

You may notice that we are setting a variable **$item#** with the value of 1; as we go through the records, we will be using that variable where we need to display an item # (and then increasing its value by 1); otherwise we will not use it.

Now we are ready to start building the quote itself. As mentioned earlier, the instructions are different for sub quotes priced as a conveyor and as parts so the first thing we do is to check to see what the sub quote is priced at:

```
If [Quotes::PriceAs="Conveyor"]
```

If it is not, we skip to the second part of the loop but since, in this case, it is a conveyor, then we carry on:

```
If [Quotes::PriceAs="Conveyor"]
  Set Field [Admin::G_Item#; Admin::G_Item#&$item#&"¶"]
  Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
  If [not IsEmpty(Quotes::Q_Series)]
    Set Field [Admin::G_Category; Admin::G_Category&TextStyleAdd("CONVEYOR";Bold)&"~¶"]
    Set Field [Admin::G_Part#; Admin::G_Part#&TextStyleAdd(Quotes::C_ModelNumber&"~¶";Bold)]
  Else
    Set Field [Admin::G_Category; Admin::G_Category&"¶"]
    Set Field [Admin::G_Part#; Admin::G_Part#&"¶"]
  End If
  Set Field [Admin::G_ItemName; Admin::G_ItemName&"~¶"]
```

What we are doing here is setting our header row so that if it is a conveyor and there is a series #, we create a line with the category set to **CONVEYOR** followed by the Model # and both of them being displayed in bold text. If not, we simply append a carriage return ¶ to those fields. We also don't want an item name here so we set that also as ¶ prefixed by a tilde **~**. (Note: that the only reason the tilde is there is for debugging so that I can see that a line is being created. In the VL Table, those tildes are substituted out and replaced with nothing).

The next thing we have to do is determine whether we need to show the item priced as **EA** (Each) and we do this if the conveyor quantity is greater than 1

```
If [Quotes::ConveyorQuantity>1]
  Set Field [Admin::G_Each; Admin::G_Each&"EA¶"]
Else
  Set Field [Admin::G_Each; Admin::G_Each&"¶"]
End If
```

and finally we are finishing off our header row with the conveyor quantity and the price, also formatted in bold text:

```
Set Field [Admin::G_ItemQty; Admin::G_ItemQty&TextStyleAdd(Quotes::ConveyorQuantity&"~¶";Bold)]
Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&TextStyleAdd(Quotes::Conveyor_Price&"~¶";Bold)]
```

At this point, you are probably rolling your eyes and going "Oh, god!" but I urge you to persevere. It is a lot of work to do this and it does take some time to get it right but the results are staggering and, once you've got the hang of doing this, you will find it, relatively, easy to do again.

So moving on, we now need to 'write' the line items and, of course, the first thing we must do is check that there are actually line items

```
If [Count(QuoteItems::Serial_#)>0]
```

If there are, we go to those records and make sure that we are in the first record:

```
If [Count(QuoteItems::Serial_#)>0]
  Go to Related Record [Show only related records; From table: "QuoteItems"; Using layout: "QuoteItems" (QuoteItems); New window]
  Sort Records [Restore; No dialog]
  Go to Record/Request/Page [First]
```

You may notice that I've got a **Sort Records** in here. This is because the user may have changed the order the items should appear in and we need to take that into account as otherwise they will just appear in the order that they were created.

Now we run a **loop** through all of the found (related) records:

```
Loop
  Set Field [Admin::G_Item#; Admin::G_Item#&"~¶"]
  Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
  Set Field [Admin::G_Category; Admin::G_Category&QuoteItems::Category&"~¶"]
  Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(QuoteItems::ProductDescription;"¶";" - ")&"~¶"]
  If [QuoteItems::Item_Type="product"]
    Set Field [Admin::G_Part#; Admin::G_Part#&QI_Product::PDTNumber&"~¶"]
  Else
    Set Field [Admin::G_Part#; Admin::G_Part#&QI_Component::InHouseManuNumber&"~¶"]
    Set Variable [$$part#; Value:$$part#&QI_Component::InHouseManuNumber&"¶"]
  End If
  Set Field [Admin::G_ItemQty; Admin::G_ItemQty&"~¶"]
  Set Field [Admin::G_Each; Admin::G_Each&"¶"]
  Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&"~¶"]
  Go to Record/Request/Page [Next; Exit after last]
End Loop
Close Window [Current Window]
```

There is a slight twist to this and that is the items that were added to the quote may have come from different tables depending on whether they are a component or a part so depending on where they come from, we have to insert a different part #.

Once we're finished with the last quote item, we close the window and finally we look to see if there are any notes that need to be shown to the customer. If so, we add them:

```
If [not IsEmpty(Quotes::Notes4Cust)]
  Set Field [Admin::G_Item#; Admin::G_Item#&"~¶"]
  Set Field [Admin::G_Category; Admin::G_Category&"~¶"]
  Set Field [Admin::G_Part#; Admin::G_Part#&TextStyleAdd("Notes for #"&$item#&"¶";Bold)]
  Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(Quotes::Notes4Cust;"¶";" - ")&"~¶"]
  Set Field [Admin::G_ItemQty; Admin::G_ItemQty&"~¶"]
  Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&"~¶"]
  Set Field [Admin::G_Each; Admin::G_Each&"¶"]
  Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
End If
```

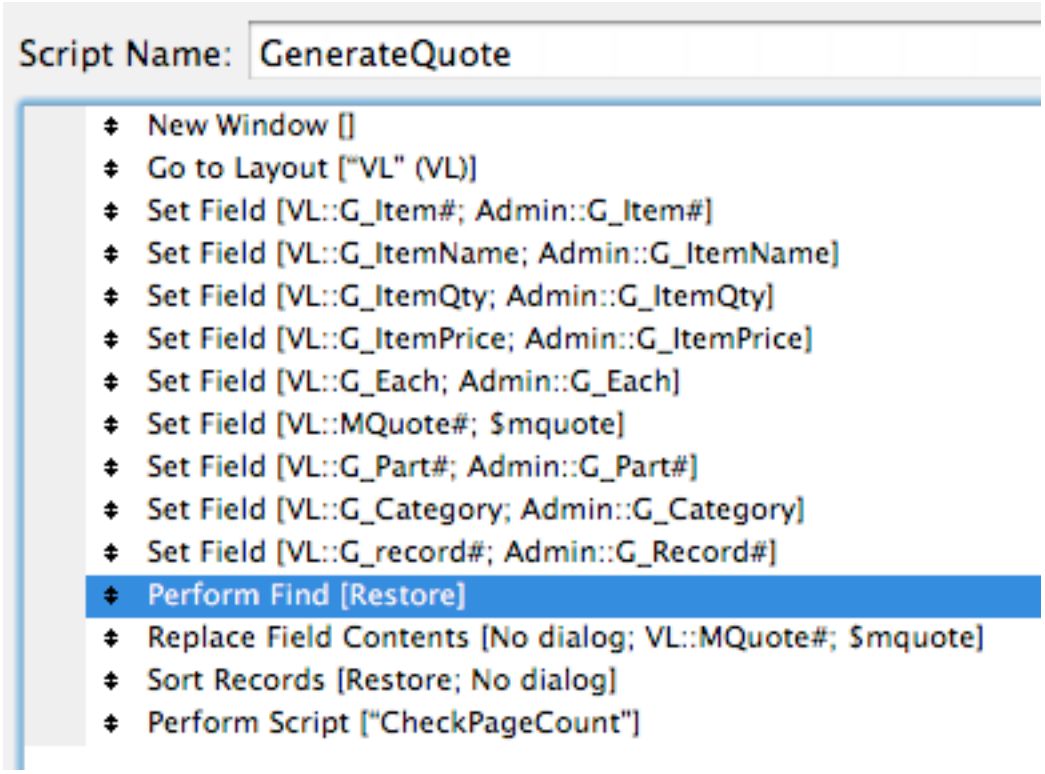
Everything we've done so far related to a sub quote that is being priced as a **conveyor**. If, however, it is being priced as **parts**, then the scripting is slightly different but the principle is exactly the same:

```
Else If [Quotes::PriceAs="parts" and Count(QuoteItems::Serial_#)>0]
  Go to Related Record [Show only related records; From table: "QuoteItems"; Using layout: "QuoteItems" (QuoteItems); New window]
  Sort Records [Restore; No dialog]
  Go to Record/Request/Page [First]
  Loop
    Set Field [Admin::G_Item#; Admin::G_Item#&$item#&"~¶"]
    Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
    Set Field [Admin::G_Category; Admin::G_Category&QuoteItems::Category&"~¶"]
    Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(QuoteItems::ProductDescription;"¶";" - ")&"~¶"]
    If [QuoteItems::Item_Type="product"]
      Set Field [Admin::G_Part#; Admin::G_Part#&QI_Product::PDTNumber&"~¶"]
      Set Variable [$$part#; Value:$$part#&QI_Product::PDTNumber&"¶"]
    Else
      Set Field [Admin::G_Part#; Admin::G_Part#&QI_Component::InHouseManuNumber&"¶"]
      Set Variable [$$part#; Value:$$part#&QI_Component::InHouseManuNumber&"¶"]
    End If
    Set Field [Admin::G_ItemQty; Admin::G_ItemQty&QuoteItems::QuantityOrdered&"~¶"]
    If [QuoteItems::QuantityOrdered>1]
      Set Field [Admin::G_Each; Admin::G_Each&"EA¶"]
    Else
      Set Field [Admin::G_Each; Admin::G_Each&"¶"]
    End If
    Set Field [Admin::G_ItemPrice; Admin::G_ItemPrice&QuoteItems::C_price&"~¶"]
    Set Variable [$item#; Value:$item#+1]
    Go to Record/Request/Page [Next; Exit after last]
  End Loop
  Close Window [Current Window]
  Go to Record/Request/Page [Next; Exit after last]
  Set Field [Admin::G_Record#; Admin::G_Record#&"1¶"]
Else
  Go to Record/Request/Page [Next; Exit after last]
End If
```

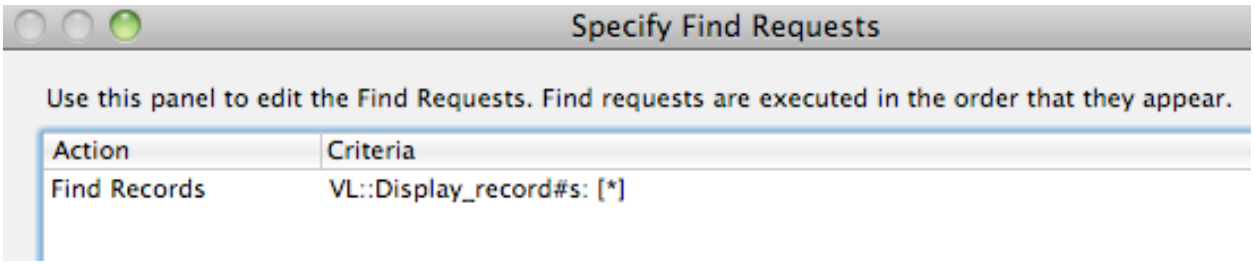
We've now come almost to the end. We've gone through and written all of the information that we need into a series of global fields. Where we had to information to write, we simply inserted a ¶ which is very important otherwise things will not line up. In the notes field, we also had to make sure that the user had not put a carriage return in there by substituting a dash for it:

```
Set Field [Admin::G_ItemName; Admin::G_ItemName&Substitute(Quotes::Notes4Cust;"¶";" - ")&"~¶"]
```

Finally we now create the quote itself:



You'll notice that there is a **Perform Find** here:



and you are probably wondering why. If you remember, earlier on we talked about our **VirtualList** table which we had populated with 1000 or more **blank** records. When we set the fields in that table with the values from our globals, we'll end up with as many records being used as there are rows from the global fields. Thus we want to only use/show records where there is a value. I've set this up so that every row that I create has a value in the record # field and thus the find isolates those records.

Let's look at the **VirtualList** table and see what happens there:

Table: VL 34 fields

Field Name	Type	Options / Comments (Click to toggle)
G_Each	Text	Global, Always Validate
Display_Each	Calculation	Unstored, from VL, = Substitute(GetValue(G_Each; Index);"~";"")
G_Item#	Text	Global, Always Validate
Display_Item#s	Calculation	Unstored, from VL, = Substitute(GetValue(G_Item#; Index);"~";"")
G_ItemDate	Date	Global, Always Validate
Display_ItemDates	Calculation	Unstored, from VL, = Substitute(GetValue(G_ItemDate; Index);"~";"")
G_ItemName	Text	Global, Always Validate
Display_ItemNames	Calculation	Unstored, from VL, = Substitute(GetValue(G_ItemName; Index);"~";"")
G_ItemPrice	Text	Global, Always Validate
Display_ItemPrices	Calculation	Unstored, from VL, = Substitute(GetValue(G_ItemPrice; Index);"~";"")
G_ItemQty	Text	Global, Always Validate
Display_ItemQty	Calculation	Unstored, from VL, = Substitute(GetValue(G_ItemQty; Index);"~";"")
G_record#	Text	Global, Always Validate
Display_record#s	Calculation	Unstored, from VL, = Substitute(GetValue(G_record#; Index);"~";"")
G_Category	Text	Global
DisplayCategory	Calculation	Unstored, from VL, = Substitute(GetValue(G_Category; Index);"~";"")
G_Part#	Text	Global
DisplayPart#	Calculation	Unstored, from VL, = Substitute(GetValue(G_Part#; Index);"~";"")

Our **global** fields are what we set the values with and the **display** fields split those into individual records based on the **Index** field which is very simple.

Index	Number	Indexed, Auto-enter Calculation replaces existing value

Index =

```
Get( RecordNumber )
```

Each of the display fields gets a value based on the record number.

Display_ItemNames =

```
Substitute(GetValue( G_ItemName; Index );"~";"")
```

We're only really concerned with getting the value (the highlighted part of the calculation); the wrapper is just to make sure that any tildes that were left in are substituted out.

Now when you run this script, the screen flashes very quickly as it jumps from screen to screen, grabs the information it needs and then closes that window. The process is incredibly fast. In the example we're using, it takes about 3-4 seconds for the quote to be displayed on the screen.

Summing up, there are many advantages to this methodology. Firstly it allows the user to produce something that could not be readily done in FileMaker™ and secondly it can be used over and over again for many different scenarios; such as a revision or even a completely different report/document. You can insert calculations to sum up the records within a grouping, format text any way you want to. In fact, the sky is really the limit here or at least your own imagination sets the boundaries.

There is one, important, caveat to this process. If you're using it on a multi-user system, it will not work reliably as you can have more than one person clearing and writing values. The simple solution to this is to put the VL table in its own file, make sure that network sharing (on that file) is set to off, and install the file on each users machine. That way, they control their own destiny, so to speak, and you won't run into any problems.

Go ahead and try it; I think you'll have fun ... eventually and definitely virtually!