1

In This Chapter

New Dashboard...

Component Binding

Changing Default Dashboard Properties

Data Source

Creating a New Dashboard

Linking Dashboard Records to Contact Records

Manually Typed Data Sources

Pivot Table

Note

Mr. Iain Wicks has agreed to permit me, albeit for a fee, to distribute his 10 Dashboard Videos along with this book. If you think that these are helpful to your needs, you may purchase the entire collection of videos at a reduced cost of ***$339.15*** *US from:*

http://www.djhunt.us/DWSite/Docs/GoldMine/Video_GMPE.html

Dashboards really say it all as far as the new tool on the block in GoldMine Premium. I can't tell you how excited and then disappointed we were to see this module in GoldMine Premium. Excited because everyone has been waiting for this tool for such a long time. Disappointed because there is no documentation on Dashboards at all. My partner James McCracken may be writing a book on the Dashboards in the future, but there is very little to date.

Hence this, the Dashboards chapter, replaces the former Chapter 1 that was published in The Hacker's Guide series of books. I will do the best that I can to explain the basics of Dashboards, however, it would take an entire book to do Dashboards justice, and I simply don't have room in this book.

Dashboards are a graphical way of displaying your GoldMine Premium data, as opposed to, let's say, GoldMine Reports. Accessing the Dashboards is very simple. One can simply click on **Dashboards** on the Outlook style toolbar to the left of the GoldMine Premium screen or one could utilize the GoldMine Premium menu:

Go To
Dashboards

Either way will bring forth the **Dashboards** dialog form as shown below in Figure 1-1:

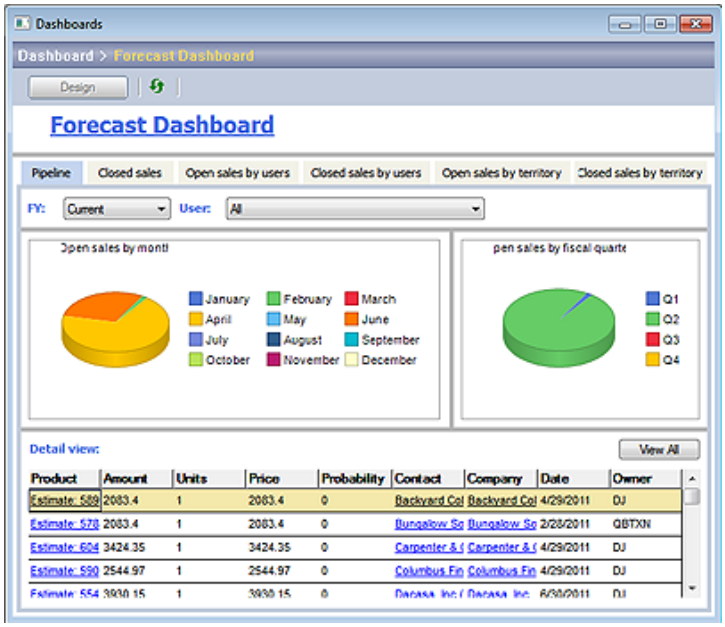

Figure 1-1

Please excuse any distortion in this image. So as to fit this image within the constraints of this page, there had to be a large amount of image compression.

This particular screen shot is that of the **Forecast Dashboard** with its 6 tab that cover **Pipeline**, **Closed sales**, **Open sales by users**, **Closed sales by users**, **Open sales by territory**, and lastly **Close sales by territory**.

While the Dashboards are open let's take a look at the Quick Toolbar to the left in GoldMine Premium as it should not contain all of your installed Dashboards. You may see what mine looks like on the next page in Figure 1-2, and yours should look similar to this. In the default state, one has 2 categories, **Management** containing 7 dashboards, and **Sales** containing 3 dashboards.

Tip

*The **Activity Aging Random Example** is not one of the GoldMine Premium default dashboards as it is one that I Imported as a test of the Dashboard Import feature during the Beta cycle.*

*Additionally, the **Undefined** category shown in Figure 1-2 was developed as part of my testing.*

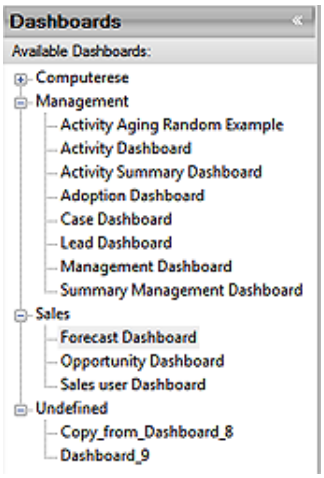

Figure 1-2

A right-click over any of the dashboards and you will have the ability to:

New
Edit
Copy
Properties
Delete
Import
Export

Notice that there are no buttons with which to do these activities within the Dashboards dialog form. There is one hotlink at the bottom of the Quick Toolbar that will allow one to create a **New Dashboard...**, but beyond that your only way to access these activities is via the old right-click method on a Quick Toolbar category or item.

I do not plan on covering the included Dashboards as I think that you should be able to discover those capabilities on your own. What I am going to cover is the basics of creating your own Dashboards. So while you are reviewing the default Dashboards on your own, pay particular attention to the items contained within the Dashboard dialog form like multiple (tabbed) dashboards, the various Filtering capabilities. Many of these features may exceed my capabilities to cover them within this book, however, I will do my best to incorporate as many as possible.

New Dashboard...

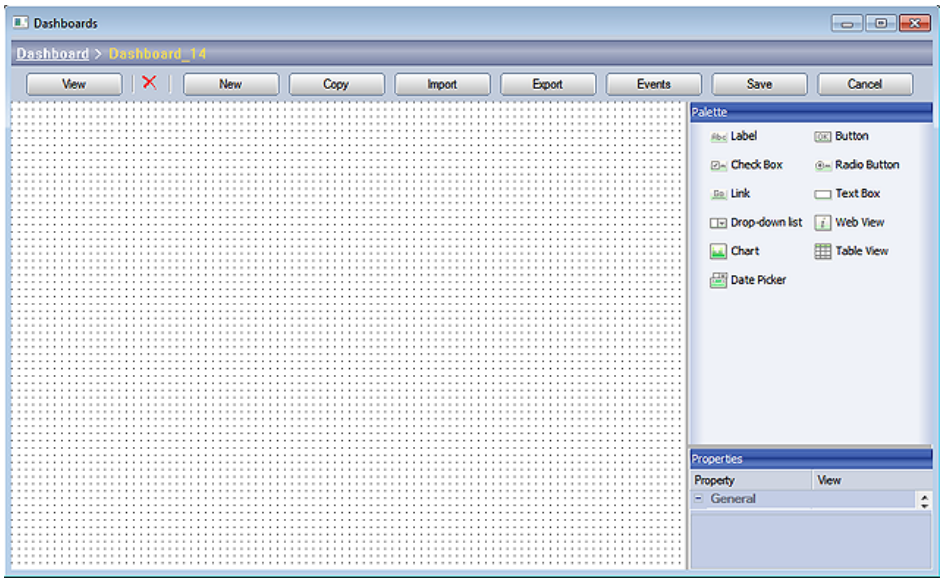

Figure 1-3

As the section title indicates, let's begin a **New Dashboard...**. Go ahead, and click on the **New Dashboard...** hotlink to bring forth the **Dashboards** designer dialog form as shown above in Figure 1-3 along with its' corresponding Quick Toolbar shown in Figure 1-4 on the next page. Again, pardon any distortion in Figure 1-3 as the compression was again quite severe. From here on I'll try to keep my screen shots to the segments of the dialog against which I am discussing.

Basically, as the screens that we will be discussing later in this book, this dialog form, the grid area in particular, represents a blank slate upon which we will be designing our dashboard(s). Referring to Figure 1-3, we have our general buttons above the grid area, and the **Palette** and **Properties** to the right of the grid area. The buttons permit you to **View**, **X** delete, create **New**, **Copy**, **Import**, **Export**, **Events**, and the standard **Save** and **Cancel** operations. On the other hand, your **Palette** contains the components that one may utilize in the creation of a dashboard, and we'll look at a couple of these a little later in this chapter. If you reviewed the default Dashboards, then you may have observed some of these components in action. They labels are pretty self descriptive so I won't repeat them all here.

Next we have the **Properties** container, which is an editible area that maintains some of the properties of your dashboard. As this area is compressed in Figure 1-3, I will list the default properties of **Category**, **Grid Spacing**, **Name**, **Timer** which are all in the **General** section, while the **Owner** property is part of the **Security** section. In turn, as you add components to your dashboard, this Properties container will change to reflect the properties of the component that was added or is highlighted.

Why don't we jump right in and change a couple of these properties. Let's change the **Category** to **Definitive Guide Examples**. We'll leave the **Grid Spacing** at its default setting, and we'll change the **Name** to **Definitive Guide Example 1**. That's it for this go around. If one were to click on the **View** button at this point, one would be able to see the new category of Definitive Guide Example along with its node of Definitive Guide Example 1. Voilà, you're a dashboard designer.

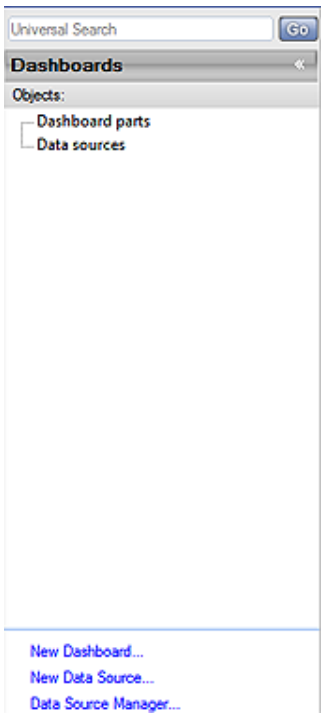

Figure 1-4

We need to add a few components to our dashboard, however, each component will eventually need to be linked to a data source. When you are looking at a dashboard, you are always looking at the component with its contained data source. Without this pairing you would have just a **Table View** say, without it having any data. Not a very good dashboard, is it? We control the various data sources by using the **Data Source Manager...** as shown in Figure 1-4, however, we will discuss that in a little more detail later in this chapter.

For now we just want to add a couple of components to give us a feel for how the components enhance your dashboards. Almost every dashboard will have at least two components, the **Chart** component allowing for the graphical representation of your bound data, and the **Table View** component allowing for the record presentation of the bound data.

Why not drag & drop the Chart component on the grid first placing it in the upper left hand corner of your grid leaving at least a 1 grid spacing to the top and left sides of the component. I then ask you to grab and drag the lower right hand corner handle such that your chart is approximately 6" wide and 3" in height.

Next, we'll want to modify a few of the properties for this chart. Did you notice that the **Properties** section changes when you dropped the Chart component on your gird? Before we forget, we would like to modify the **Anchors** which are contained in the **Layout** section of the **Properties**. And their definition, according to FrontRange is:

> **Anchors**
> Defines the edges of the container to which a certain control is bound.

For the purposes of this book, the container is our grid drawing surface while the control is what I refer to as the component. I suppose if I wanted be faithful to all of the designers among us, I would stick to the designers terminology, but I am trying to bring this book down to a more end user reading level.

Ah well, I wandered off on a tangent. By binding the edges of our component to the edges of our grid drawing space, we permit the component to anchor those edges such that if an end user changes the size of the dashboard, your component will change proportionally as well. Your design is maintained regardless of the Dashboard size set by the end user.

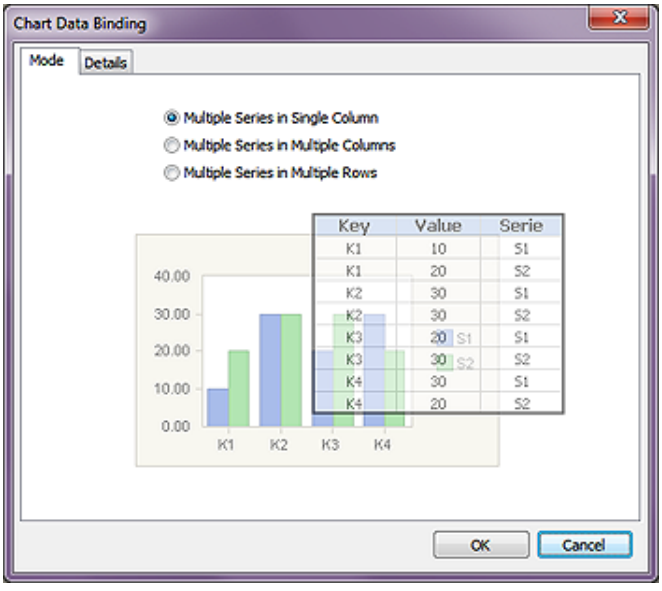

Figure 1-5a

I would like you to change the **Left**, and **Top** Anchors from **False** to **True**. For the time being we'll leave the **Right**, and **Bottom** Anchors at **False**. You may either do this on the single **Anchors** property (don't forget to Tab out when done) or via the individual properties below the **Anchors** property.

Without worrying about Data Management at the moment, we are now going to Bind our component to a data source. We can do this because the developers at FrontRange have predefined 20 categories of data sources, with each category containing

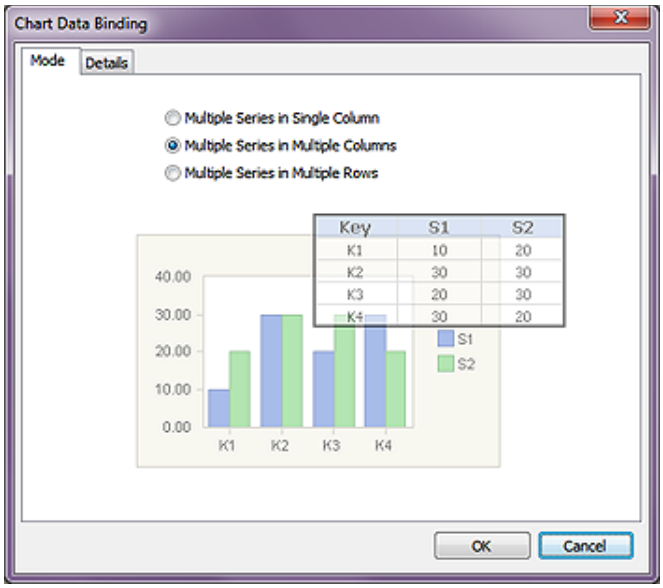

Figure 1-5b

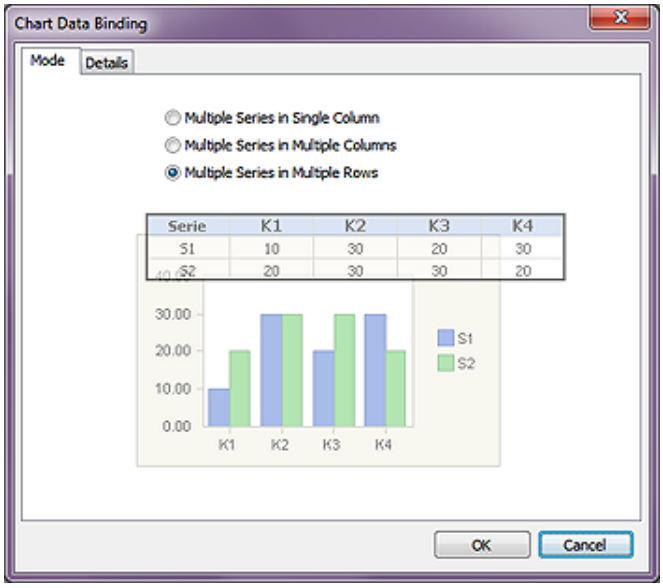

Figure 1-5c

multiple data sources. So let's walk through the minimal Binding process. In the **Properties** section, click on the ellipsis (**...**) to the right of the **Data Binding** property.

On the first tab page of the **Chart Data Binding** dialog form, we see that you have three radio button options from which to choose on the **Mode** tab. Notice that I have supplied an image of each screen as each option was selected, and pay particular attention to the displayed example table view associated with each radio button selection. This series of images is shown in Figures 1-5a on the previous page, and 1-5b & 1-5c on this page.

Although the GUI chart of the bound data never changes through the series of **Mode** selections, the table views of the selected data certainly do change. For this exercise I would like the default selection as shown in Figure 1-5a on the previous page.

Now, if you will, I would have you click on the **Details** tab of the **Chart Data Binding** dialog form which should result in the dialog form as shown in Figure 1-6 on the upcoming page. If you are not following in your GoldMine Premium, you may want to review the Figure 1-6 in this book at this time.

So in picking a category that is not too awfully complicated for this book, I have decided to utilize the **Data source category:** of **Closed sales**. Once I have made that decision I am forced to work within that constraint. For instance, the **Data source:** drop list was built based on our category decision, and contains some 8 possibilities in its default state of which I will select **Closed sales amount by user**.

This selection, in turn, populates the drop list for the remaining 5 selections. For the **Data are summarized by:** choice, I have selected **Activity_owner**. For the **Summarizing key legends (optional):** choice, I have again selected **Activity_owner**. Finally, in this section anyway, for the **Summarized values:** choice, I have selected **sum_amount**.

Now we have one more section to worry about or not (optional settings) on the **Details** tab of the **Chart Data Binding** dialog form, and that as stated on the dialog form is:

> **If the source contains multiple series of data, specify the following options.**

For both the **Series distinguishing field (optional):** and the **Series legends (optional):** choices, based on my previous selections, I have to leave them at their default state of **(empty)**.

If you have been following along up to this point then I would ask you to now click on the **OK** button, and then the **View** button in turn. Hopefully, you have been using GoldMine Premium as it was designed, and you users have Forecast and Closed their Sales. If so, your chart should look similar

to my chart as displayed in Figure 1-7.

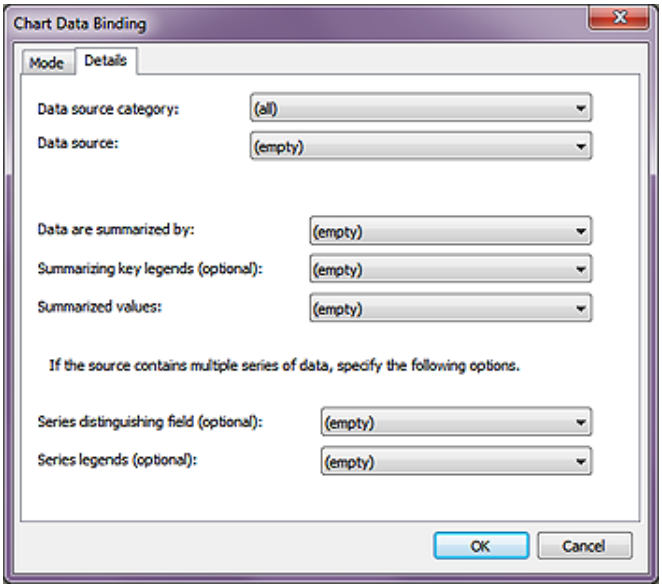

Figure 1-6

Looking at Figure 1-7, one might instantly notice that **BOB**, **LYNN**, and **JAMES** haven't had any sales. Visually, precisely and quickly, one can determine who the slackers are, and who the doers are in your organization.

What I immediately notice is that I require some sort of filtering capability because, in fact, Bob, Lynn and James are not even involved in sales within our organization. They are or were part of the Computerese Inc signature GoldMine Premium Support Consultant Team.

How about we add yet another component? Why don't you drag the **Table View** component below our **Chart** component leaving 1 grid spacing between the two while aligning the left edges. While we're at it, why don't we align right hand edges as well, and make the height approximately 4".

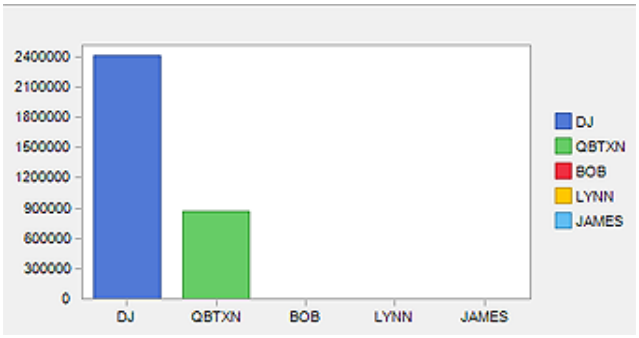

Figure 1-7

Here I have set the **Layout Anchor Properties** as **Left | True**, **Top | True**, **Right | False**, and finally **Bottom | False**. For the heck of it I also set the **Properties** | **General** | **Name** to **Closed Sales Table**. You should feel free to ad lib here if you wish.

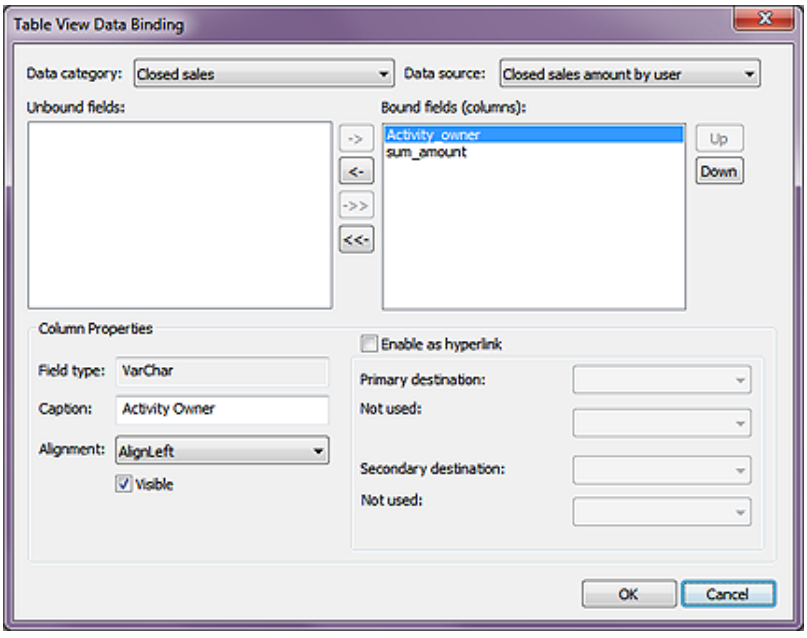

Figure 1-8

I think that we should begin to discuss the data binding for this component. We begin the binding process exactly as we did for the last component, however, this time the binding dialog form is dif-

Note

*Drag & Drop does not function between lists in the **Table View Data Binding** dialog form. To move all of the fields from one list to the other list would require that one click on either the **->>** or the **<<-** button whichever is appropriate for the direction you wish the fields to travel in.*

ferent. Notice, Figure 1-8 on the previous page, that the **Table View Data Binding** dialog form for this component is a single dialog form. We are going to bind this control to the same data source as we previously used because we want to display the data from which the chart was created for comparison purposes. Our **Data category:** will be **Closed sales**, while our **Data source:** will be **Closed sales amount by user** just as before. Immediately, **Activity_owner** and **sum_amount** will appear in the **Unbound fields:** list of fields. We want to move both of those fields over into the **Bound fields (columns):** list.

Clean up time. Next, we'll highlight the field **Activity_owner**, and change the **Caption:** for that column to a more dashboard display readable format of **UserID**, while we leave the column **Alignment:** value at its default for **VarChar Field type:** of **AlignLeft**. Additionally, we also want to leave the default value of ☑ **Visible**. We are not going to check the option to ☐ **Enable as hyperlink** as neither of these two fields links to anything that identifies the Contact record. Now let's highlight the **sum_amount** field, and change its **Caption:** to simply **Amount**.

Note

*Because of the nature of the data contained within this dashboard and depending on your organizations policies, you may want to consider changing this dashboards **Properties | Security | Owner** to that of a UserID or a User Group that you have established within your GoldMine Premium.*

UserID	Amount
DJ	2408653.49999999
QBTXN	873120.52
BOB	4871.5
LYNN	750
JAMES	375

Click on the **OK** button on the **Table View Data Binding** dialog form, and then again on the **View** button. Figure 1-9 shows you the results of the **Table View** once my data has been bound.

Figure 1-9

Did you notice anything unusual in Figure 1-9? Now I can clearly see that Bob, Lynn and James have all had sales. Graphically, because of the scale of DJs graph, their sales were not available in a visual manner where they are in the table grid. I bring this to your attention to show you a solid reason for utilizing a Chart and a Table View in your dashboards to show you the same data displayed in two unique formats. The more information that you can supply to your end users, the more power that they will have to wield.

Before I move us on to the next step I want to save a backup of this Dashboard. The way that we do that is by clicking on the **Export** button, and exporting this dashboard to a safe location. We'll restore from there later, so keep it handy.

Let's look at display a second screen within this Dashboard. The first approach that we will look at is the ability to subdivide the screen either horizontally or vertically into two independent screens. For this exercise I will work with the Vertical Splitter. With the Dashboard in the **Design** mode, right-click over the grid area, and select **Add vertical splitter** from the Local Menu. Now drag it to within one grid spacing from your existing components right edges. Technically, you now have two frames (screens) each independent of the other.

Want me to prove that? Drag the **Web View** component such that its left edge is one grid spacing to the right of the Vertical Splitter, and one grid spacing from the Dashboard on the three other sides. Under the **Properties** for this component, we will not do any **Data Binding**. I have changed the **Name** property on mine to **CNN World News**, and I have changed the **URL** property to **http://www.CNN.com/World/**. Lastly, I have set all of the **Anchors** to **True**.

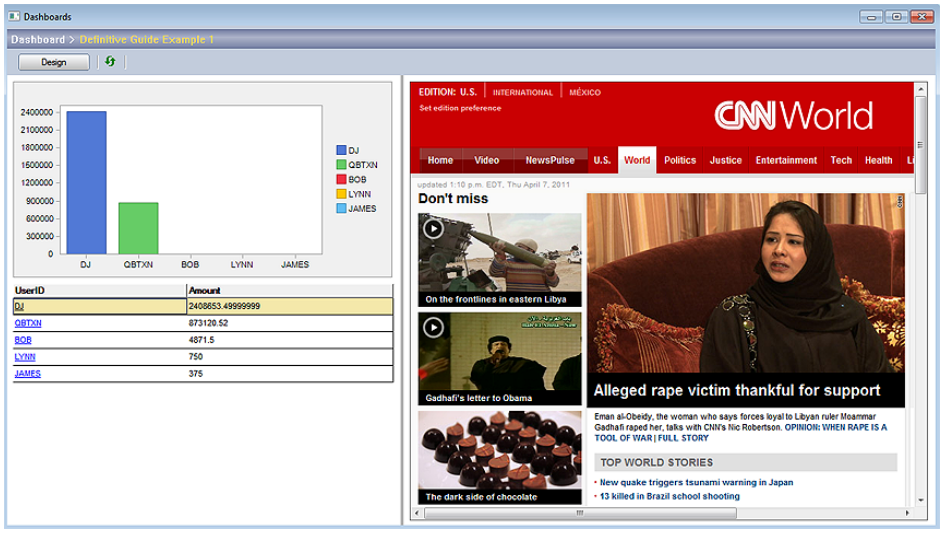

Figure 1-10

In Figure 1-10, you can see the consequences of my actions. And the acid test for independent frames (screens), drag the Vertical Splitter to the left and watch what happens. If everything worked

as expected the browser frame should have resized to the new width while the Graph/Table View frame should have gotten partially or totally blocked from view depending on how far you moved the Vertical Splitter to the left.

Let's look at another way to approach this. First we want to remove the splitter. I move my cursor over the splitter until it changes to parallel vertical lines with arrows pointing left and right. Next I right-click, and then I select the only option of the Local Menu the **Delete** option. When you are finished with that I want you to right-click over the **Web View** component, and choose **Delete control** from that Local Menu.

We've made it back to our base screen now so let's right-click over the grid area, and choose **Change to tab frame** from that Local Menu. I named my tab **Closed Sales by UserID**. I would then ask you to right-click just to the right of this tab, and select **Add tab** from the Local Menu. I named this tab **World News**. While this tab is active, add your **Web View** component just as we did previously. Click on the **View** button, and play around with your work. Again, we have two independent screens, but this time they each have their own tab, hence, they are visually separate.

Note

*To activate a Tab Frame, simply click on the tab, whether you are in the **Design** or **View** mode.*

Let's click on the **Design** button again, and this time I want you to click on the **Import** button. Because you did not delete the Dashboard that you were working on, you will receive the message shown here in Figure 1-11. Since this is only a practice run I would ask you to accept the default setting Replace existing dashboard, although, you could as easily have just created a new Dashboard from this import.

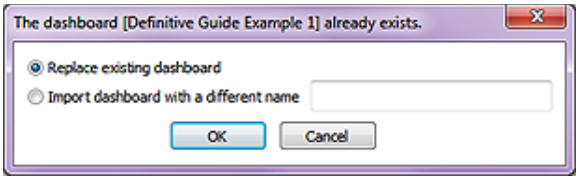

Figure 1-11

Let's try another component. Slide your Closed Sales Chart and Closed Sales Table down about 3/4" on the screen, and then Drag & Drop a **Label** component into the exposed grid area positioning it neatly. Some of the **Properties** that I change on mine are **Font | Tahoma, 18, Bold**, **Text Color | DarkBlue**, **Name | Closed Sales Title**, and **Text | Closed Sales by UserID**. Now, you'll need to switch between **Design** and **View** modes and stretch and heighten your **Label** component until your dashboard title looks correct and you are pleased with your design.

Tip

During the design stage, you want to remember to export your design after you have completed a stage with which you are satisified.

Backup Backup Backup

Again, just to be safe, I am going to Export this Dashboard, and you would be wise to do the same if you have been following along.

Component Binding

As of now I have 3 independent components on my Dashboard. Would it be great if the **Chart** component and the Table View component were bound such that when one clicks on a specific graph in the Chart component that the Table View reflects the data against which the Chart graph was built? I like to call this **Component Binding**, and within GoldMine Premium Dashboards, this is accomplished with the use of **Events**.

Note

A special thanks to Jamal Ahmad of FrontRange Solutions and Chris Wettre of W-Systems for their assistance in this area of my Dashboard education.

So let's begin by selecting and setting the focus to the Chart component. We can now click on the **Events** button to bring up the **Events and actions** dialog form shown here in Figure 1-12 although your **List of actions:** should be empty. Mine, on the other hand, is not empty as I have already added the **Drill down** action in preparation for this chapter in the book.

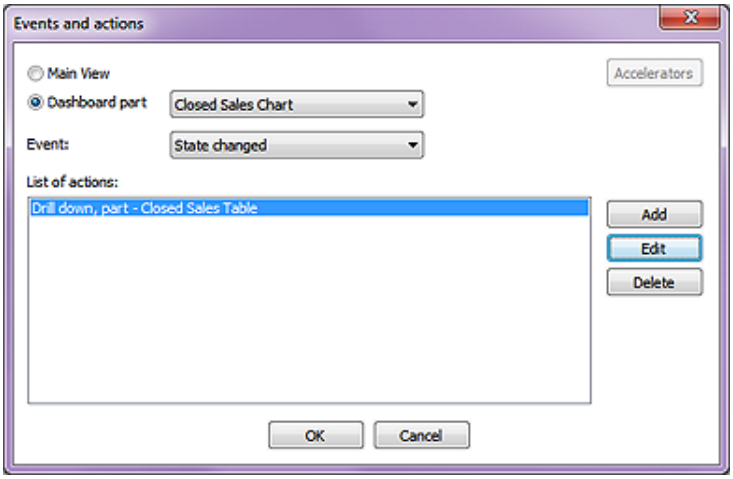

Figure 1-12

In fact, this is the very action that we need to add so I would have you click on the **Add** button while I will click on the **Edit** button. Figure 1-13 on the next page shows the **Action's properties, part - Closed Sales Chart** as I have already completed it in my pretesting for this book, however, I will go over each section as if I were creating it new.

First, notice that the dialog form **Title** has included the Dashboard component (part) **Name** property as developer information.

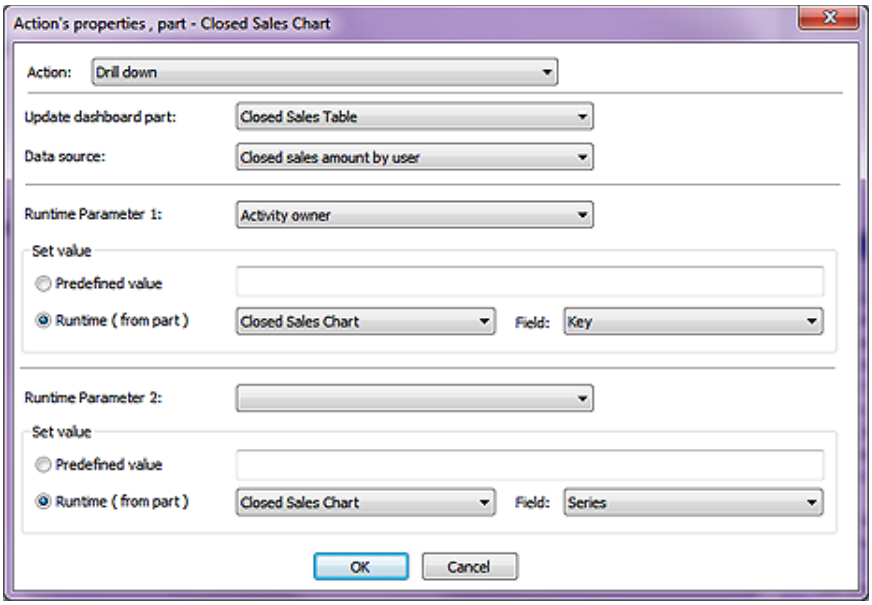

Figure 1-13

The **Action:** drop list is defaulted to **Drill down**, and as that is the action which I desire, I am willing to just accept it as is. One is next required to select the dashboard component which they would like to have updated as a result of this action, and this is done via the drop list for the **Update dashboard part:** field. You'll notice that only the components that you have already added to your dashboard will be listed in this drop list, and they will be listed utilizing the value as entered in their **Name** property. In this section of the dialog form you can also see the **Data source:** drop list, however, that is preselected for you this time as we had only selected a single data source for our **Closed Sales Table** component. I, therefore, graciously accept the **Data source:** of **Closed sales amount by user** for this entry.

Note

*The **Runtime Parameters** are predefined as part of the **Data Source**, and only those runtime parameters that were predefined for the selected data source will be made available to you via the **Runtime Parameter 1:** and **Runtime Parameter 2:** drop lists.*

For the **Runtime Parameter 1:** I have selected **Activity owner** from the available runtime parameters listed. Based on my selections so far the ◉ **Runtime (from part)** of **Closed Sales Chart**, and its' **Field:** of **Key** have been pre selected for me. Again, that is how I would have chosen my settings for this runtime parameter, however, I could have as easily chosen O **Predefined value** and added my own criterion.

Having done everything that is needed for this exercise, I would ask you to click on the **OK** button, and then on the **OK** button for the **Events and actions** dialog form. It might also be a good time to **Export** your Dashboard to another xml named file. I'm curious. Did you know that an xml is a structured document that contains name/value pairs representing your dashboard data in this particular case? Here's a the structure of the xml that I just saved:

```
<?xml version="1.0" encoding="UTF-16" standalone="no" ?>
<serialization xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="gmdashboard.xsd">

 <property id="View">
  <property id="Recid">FEWA1I5$H7B7R#Y</property>
  <property id="Name">Definitive Guide Example 1</property>
  <property id="Category">Definitive Guide Examples</property>
  <property id="Timer">0</property>
  <property id="Grid size">8</property>
  <property id="Owner"></property>
  <property id="Frames">
   <value>
    <object id="FramePanel">
     <property id="ID">frame</property>
     <property id="Parent_id"></property>
     <property id="Type">1</property>
     <property id="BaseArea">0 0 1168 865</property>
     <property id="MinSize">0 0</property>
     <property id="Visible">1</property>
     <property id="Pos">0</property>
    </object>
   </value>
  </property>
  <property id="Controls">
   <value>
    <object id="ChartView">
```

```
      <property id="Type">7</property>
      <property id="Frame">frame</property>
      <property id="Rect">8 64 568 328</property>
      <property id="InitRect">8 64 568 328</property>
      <property id="PrintRect">0 0 100 100</property>
      <property id="LpRect">212 1693 15028 8678</property>
      <property id="ID">FEWA1SW)=KPLR#Y</property>
      <property id="Name">Closed Sales Chart</property>
      <property id="Anchors">5</property>
      <property id="DataBinding">
       <object id="ChartMixedBinding">
        <property id="Data">EASPMA0%+/;\^Z!</property>
        <property id="KeyKeyField">Activity_owner</property>
        <property id="KeyVisualField">Activity_owner</property>
        <property id="SerieKeyField"></property>
        <property id="SerieVisualField"></property>
        <property id="ValueField">sum_amount</property>
       </object>
      </property>
      <property id="View Mode">2</property>
      <property id="3D Mode">0</property>
      <property id="3D Depth">20</property>
      <property id="Stacked">0</property>
      <property id="Gridlines">0</property>
      <property id="Color Scheme">0</property>
      <property id="Back Color">Control</property>
      <property id="Inside Color">Window</property>
      <property id="Title"></property>
      <property id="Keys">0</property>
      <property id="Keys Decimals">0</property>
      <property id="Values">0</property>
      <property id="Values Decimals">0</property>
      <property id="Keys Axis Title"></property>
      <property id="Values Axis Title"></property>
      <property id="Retain Source Order">1</property>
      <property id="Colored Values">1</property>
     </object>
    </value>
    <value>
     <object id="Grid">
      <property id="Type">6</property>
      <property id="Frame">frame</property>
      <property id="Rect">8 336 568 696</property>
      <property id="InitRect">8 336 568 696</property>
      <property id="PrintRect">0 0 100 100</property>
      <property id="LpRect">212 8890 15028 18415</property>
      <property id="ID">FEWA80E$4L#NR#Y</property>
      <property id="Name">Closed Sales Table</property>
      <property id="Anchors">5</property>
      <property id="Back Color">Window</property>
      <property id="Text Color">WindowText</property>
      <property id="Line Color">Black</property>
      <property id="Font">
       <property id="Height">-11</property>
       <property id="Width">0</property>
       <property id="Escapement">0</property>
       <property id="Orientation">0</property>
       <property id="Weight">400</property>
       <property id="Italic">0</property>
       <property id="Underline">0</property>
       <property id="StrikeOut">0</property>
       <property id="CharSet">0</property>
       <property id="OutPrecision">0</property>
       <property id="ClipPrecision">0</property>
       <property id="Quality">5</property>
       <property id="PitchAndFamily">0</property>
       <property id="FaceName">Tahoma</property>
      </property>
...

</serialization>
```

Of course I had to shorten it quite a bit to show you just this piece of the structure so that you could get a feel for how the xml document looks in its structured layout.

Any way, why not try your Dashboard now? First click on the **View** button if you haven't already. In the Chart, hover your cursor over one of the graphs, and watch what happens in the Legend, and the Table View. You'll see the Legend change before your eyes, however, the Table View remains constant.

Changing Default Dashboard Properties

Next click on that same graph in the Chart, and this time the Legend won't change (unless you move the cursor a hair after you click the graph), however, the Table View now displays only those records against which that particular graph unit was constructed.

I think that I'm going to stop diddling with this Dashboard for the time being so that we can move ahead. I want to look at some of the changeable properties on the default Dashboards. True, there aren't many over which you have any control, but there are a few.

Expand the **Management** branch in the Quick Toolbar if it is not already expanded. Selected any default dashboard, let's say the **Activity Dashboard**, and right-click to bring up the Local Menu. Next I want you to select **Properties** from that menu. The **Specify Dashboard's name** dialog form, as shown in Figure 1-14, becomes available, but not only can we modify the **Name:**, we could also move it into a different **Category:** or even create a totally new category here. Lastly, we have the ability to assign an **Owner:** which could be a specific UserID or a User Group should you have any predefined.

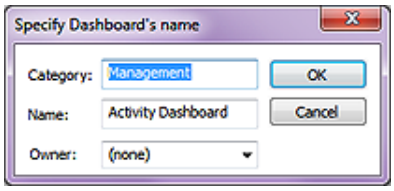

Figure 1-14

As is clearly shown, the default **Owner:** is **(none)** such that everyone has access to this Dashboard. This may be particularly useful for the **Forecast Dashboard** or the **Opportunity Dashboard** where you might want to permit only managers access to this information, remembering in the back of your mind that Master Rights users trump all settings.

Moving forward now, right-click on any bar in the **Activity Dashboard** to bring up the Local Menu which should contain:

Gallery ►
Color ►
Point Labels
Properties...

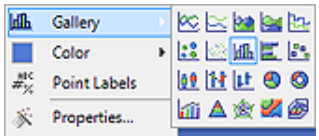

Figure 1-15

If one were to actually hover the cursor over the **Gallery ►** option, one should see the various graphs available for displaying the associated data. You can play around with this as you want to see the various types of graphs that are made available to you and how they appear in action.

I think that **Color ►** is fairly obvious as to what it represents, however, maybe **Point Labels** are not as obvious. If one were to select **Point Labels** while in the Local Menu for one of the bars or pie pieces you would see the labels appear over or on the various pieces of the charts as shown here in Figure 1-16.

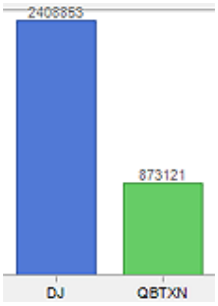

Figure 1-16

If one were to right-click off of the actual graph, but still on the Dashboard, one might see that they have the option of changing:

Toolbar
Data Editor
Point Labels
Gallery ►
Color ►
Edit Title
Point Labels
Font
Properties...

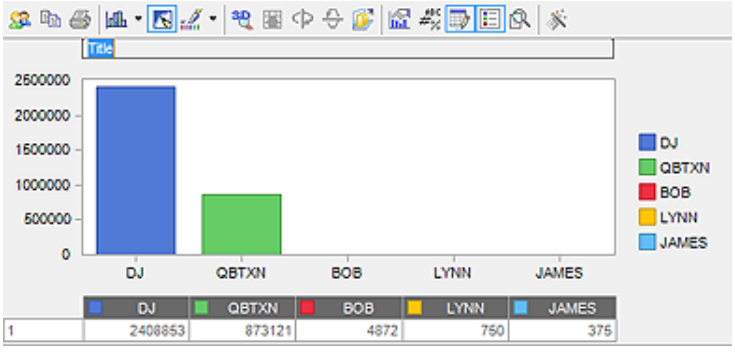

Figure 1-17

One will find that some of these settings are sticky, while some are only sticky to a point. Let's say that one selects **Data Editor**, **Edit Title** or **Toolbar**. Ones chart might look like the one shown above

Note

*Changing the graph through the **Gallery is** sticky. In other words, if you were to change a bar graph to a pie graph, leave that Dashboard for another, and return to the original Dashboard the pie graph style would have been maintained.*

Note

*Editing the Title of the default Dashboards **is not** sticky. Doing this will change the Title, however, only while you are utilizing that instance of the Dashboard.*

in Figure 1-17. And, one you may certainly change the title or any of the individual numbers that make up each graph unit. One could also **Print** or **Copy to Clipboard** this chart at this instance in time as manipulated by hand, however, click on any other dashboard and then on this dashboard again, and those edits that you had made will be gone completely. True the **Data Editor** and **Toolbar** will still be displayed there, but all of your modified entries will be gone.

Again, you'll want to play around (aka Trial & Error) with these features to see how they could be best utilized for your needs. In particular, a good area to play is with the **Properties...** dialog form as there are many display (beautification) items that can be modified in this area.

I don't think that I'm going to go into the functioning of the Grids here as they function as any grid within GoldMine. One could **Sort**, **Filter**, **Group**, etc on the grid to arrange the data in any of a number of presentations, and then as simply **Output to Excel** that very data among other avenues.

Data Source

Earlier I promised you that we would take a closer look at the **Data Source** feature of any **Dashboard**. It's about time we did just that. *What is a Data Source*, you might ask. Well to me, in Visual FoxPro terminology it is a Cursor which, in turn, is the result of executing a SQL Query. You have seen SQL Queries before within GoldMine maybe, if not, I will be discussing them in detail later in this book in Chapter 7 - **Gathering the Data**. For now let's look at:

Note

*Most of the Dashboard Options will not be available unless one of your dashboards is placed in the **Design** mode. You'll need to remember that throughout this chapter.*

Tools
SQL Query

We could look at any query at this point, however, for consistency, why don't we all look at this query:

```
select C1.AccountNo,
  C1.Company,
  C1.Contact,
  C1.Key1,
  CH.UserID,
  CH.OnDate,
  CH.Ref as [Reference]
from Contact1 C1,
  ContHist CH
where C1.AccountNo = CH.AccountNo
  and CH.RecID in
    (
    select max(RecID)
    from ContHist
    where sRecType = 'A'
    group by AccountNo
    )
order by CH.OnDate desc,
  C1.Key1
```

If you were to run this query, the resulting cursor would display the last appointment that was had with your respective contacts.

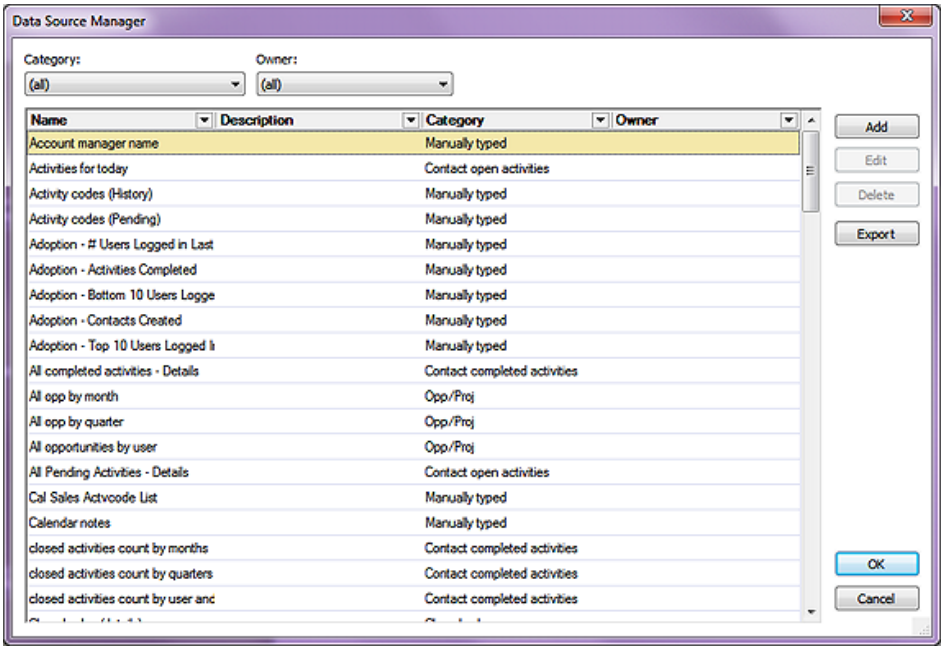

Figure 1-18

This is basically, what a Data Source is, and we have also learned that components in our dashboards are tied into a data source through the components properties.

While we are in the Design mode for our Dashboard, and in the Quick Toolbar to the left, let's click on the hotlink for **Data Source Manager...** to expose the **Data Source Manager** dialog form shown on the previous page in Figure 1-18.

Note

*Even a GoldMine User possessing Master Rights will not be able to **Edit** or **Delete** any of the GoldMine default Dashboards.*

From here it is easy to see that one may filter down the list by **Category:** or **Owner:** or both for that matter. Furthermore, we could **Add**, **Edit**, **Delete**, or even **Export** a data source.

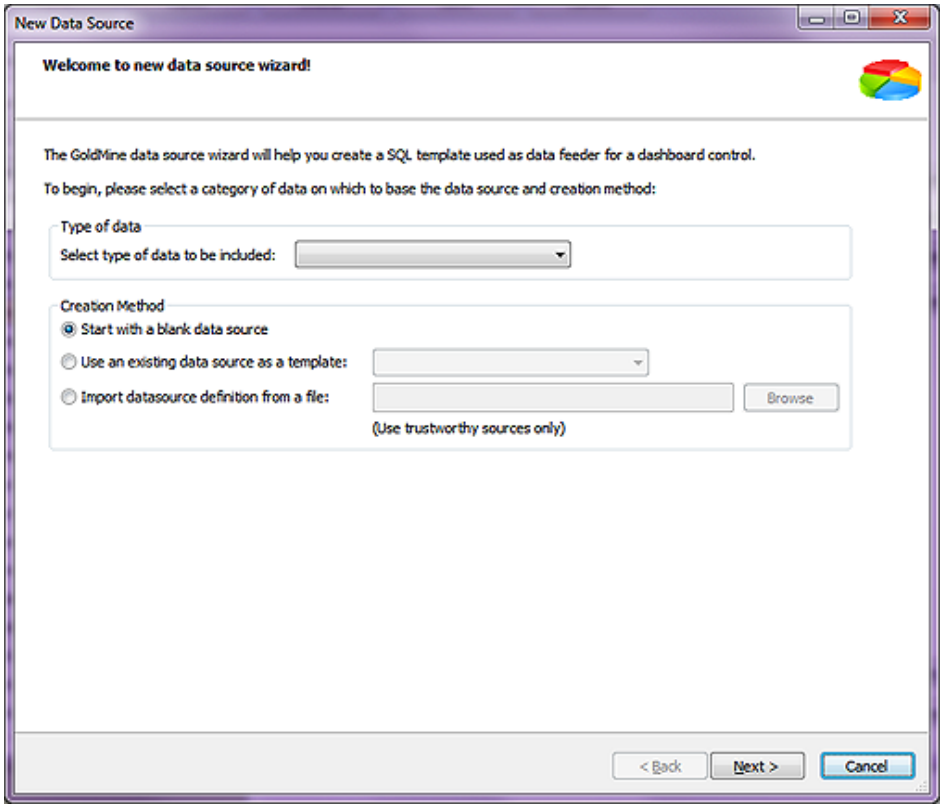

Figure 1-19

Note

I'm going to follow, in written form, the videos developed by Iain Wicks for this area so that you can always refresh yourself by using that video if you have it.

Let's begin now by building a Data Source of our own by clicking on the **Add** button to produce the **New Data Source** dialog form shown above in Figure 1-19. You should immediately notice two separate and distinct frames, the **Type of data** frame and the **Creation Method** frame.

If you focus on the first frame for the moment, you see that you have but one choice and that is the **Select type of data to be included:** which consists of a drop list of values to which you may not add to or edit any of the items within the list. Upon closer examination of this list, you might have noticed something familiar. Have you noticed that this list of items look suspicious like the Category list which you saw earlier? The observant among you will have. For this exercise let's just select the **Contact Info** category.

In the **Creation Method** frame you see that you have radio button options with the first being ◉ **Start with a blank data source** which just so happens to be the GoldMine default selection, and is the Creation Method that we will utilize in just a bit. The next option is to ○ **Use an existing data source as a template:** and is always a good place to start if you have an existing data source of the type that you need. While the final Creation Method, remember that **Export** button that I mentioned earlier, is to ○ **Import datasource definition from a file:** and one could then **Browse** for said import file. Of course, as stated, **(Use trustworthy sources only)**.

Let's trudge on down the road then, shall we, and click on the **Next >** button which will take you to the **General properties** dialog form of the Wizard as shown in Figure 1-20 at the top of the next page. You will notice that there are a few data fields and the **Output type** frame. The **Name:** field is simply for entering the name that you wish to remember this Data Source by later on when you select it from your list. You'd be wise to make it as descriptive as you can so that you have a chance of remembering the type of data which it pulls for you a year from the time when you create it and its function is clear in your mind. Ah, that goes without saying I guess, but I had to say it any way. You probably see me restate that a number of times throughout this book. Because we selected Contact

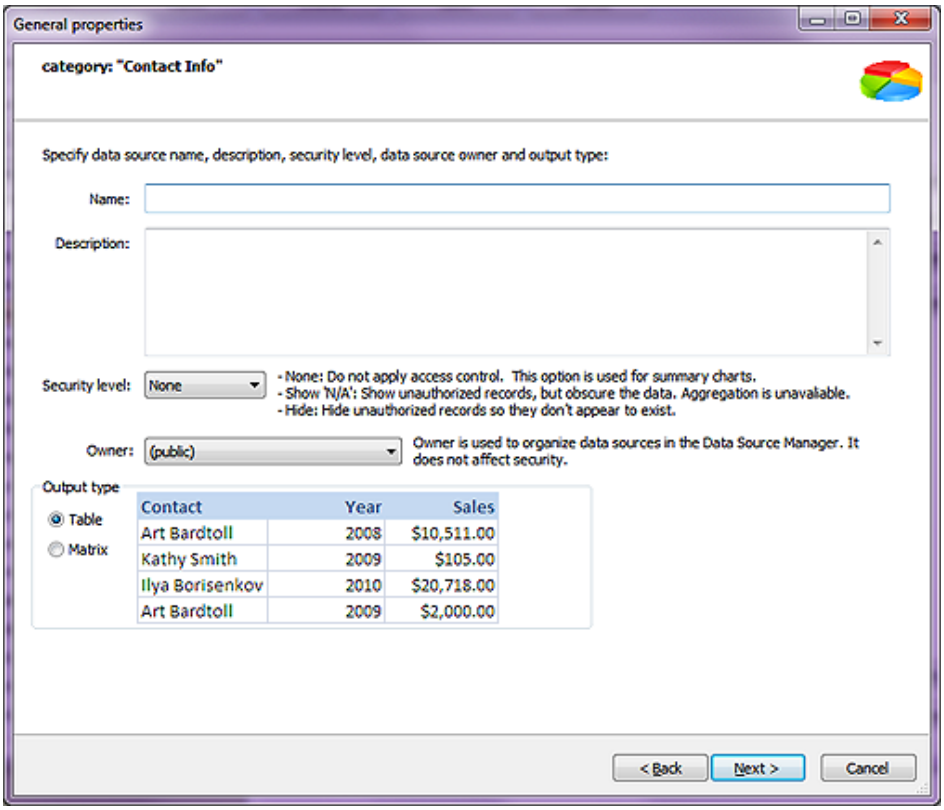

Figure 1-20

Info as our Category previously, why not just give this data source a **Name:** of **Contacts Information with Source Field**. I ask for this name as later on I plan to tie this data source in with another data source that will count against the Source field.

You can enter a **Description:** if you want to, and I highly recommend that you do that. Again, when you look at these a year from now, you are going to want to have as many ticklers to your brain cells as possible to help you in remembering exactly why you created this in the first place.

Next, you have the **Security level:** option, and FrontRange actually supplies you with a pretty good definition of this directly on the dialog form:

> **- None: Do not apply access control. This option is used for summary charts.**
> **-Show: 'N/A': Show unauthorized records, but obscure the data. Aggregation is unavailable.**
> **-Hide: Hide unauthorized records so they don't appear to exist.**

If you remember when I talked about Properties earlier, you could control who can view the dashboards by controlling that through the Properties of each dashboard. For this exercise, why not just leave this at its default setting of **None**.

Where next we come upon the **Owner:** field. Unlike the Property Owner, this Owner field does not affect security one bit. It is simply a means of organizing the Data Sources in the Data Source Manager. Odd that FrontRange would use such a security oriented label for an organizational task.

So now we have a radio button option in the **Output type** frame, and there are but two choices. A ◉ **Table** or a ○ **Matrix** (not to be confused with the cult movies). The **Table** type of output is nothing more than Columns identified in Row 1 which acts as the Header for the Columns, and Rows (Records) which contain the data delivered via the Data Source. The sample data for this type is displayed in Figure 1-20 above. Simple enough, and something with which a GoldMine user should be more than familiar.

On the other hand you might not recognize a **Matrix** as a Pivot Table or what some may call Cross Tabbing Table. As you may not be as familiar with this type of output I have used the GoldMine example here in Figure 1-21 to help you better visualize this output type. You'll notice that the Se-

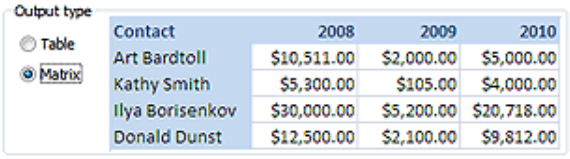

Figure 1-21

Tip

*Whenever you create a Data Source, remember to select your UserID as the **Owner:** of the Dashboard to aid in locating it later among the many Dashboards.*

ries is include in Row 1, the Header Row, while the Rows contain the Series Data as gathered by the Data Source.

You should now want to click on the **Next >** button to proceed to the **Select** dialog form of the Wizard as shown in Figure 1-22.

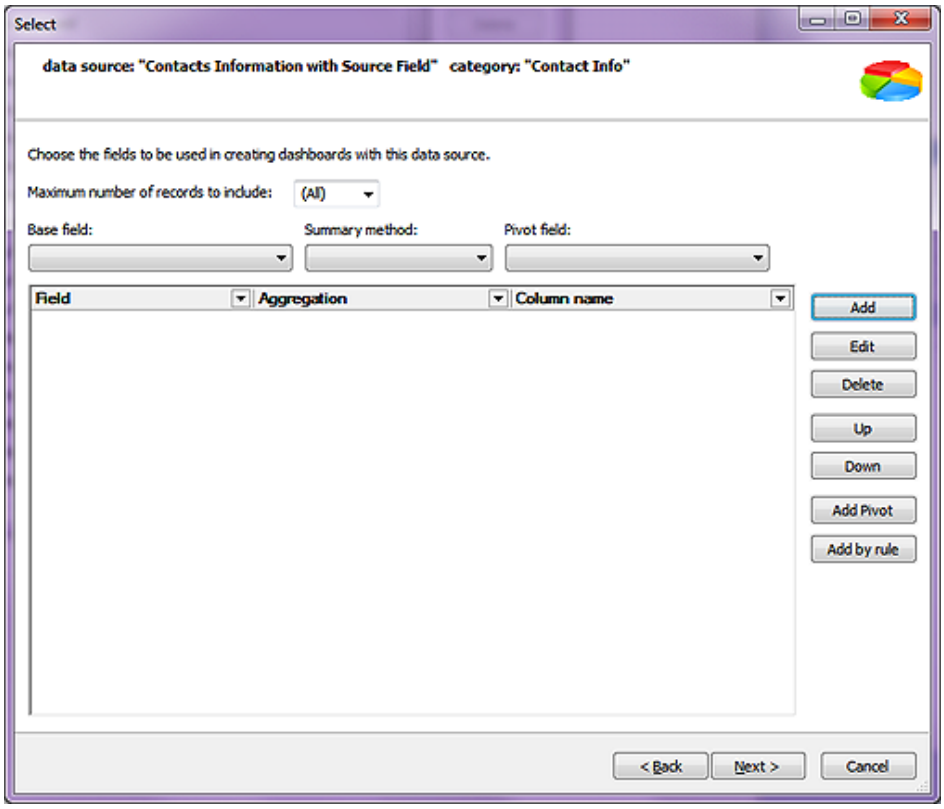

Figure 1-22

This is where you would choose the data that would be included as the **select** segment of your SQL Query. Some of the options are clearly displayed in Figure 1-22, but include the ability to **Add**, **Edit**, **Delete**, Move **Up** or **Down** in the selection order, to **Add Pivot**, or to **Add by rule**. By clicking on the **Add** button, we are presented with the **Add/Edit field** dialog form shown in Figure 1-23.

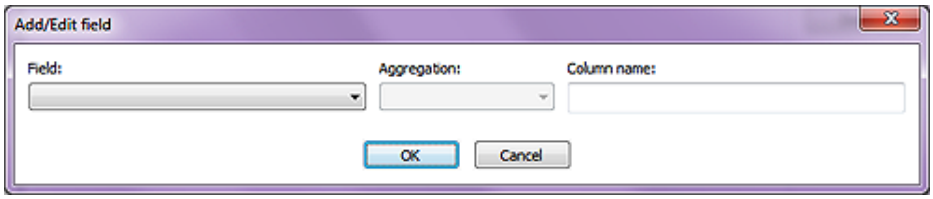

Figure 1-23

It is here that one could, obviously, Add a new selection or Edit an existing selection statement. The **Field:** drop list contains common name macros towards the top of the list like **Company**, and alias.field names like **CONTACT1(c1).COMPANY** further on down the list. As we selected a Category of Contact Info earlier, the **Field:** drop list was built with Contact1 and Contact2 fields or macros only. Here is the list of fields and their column names that I am proposing that you add for this example:

Field:	Column Name:
CONTACT1(c1).COMPANY	Company
Primary contact	Primary Contact
CONTACT1(c1).SOURCE	Source
CONTACT1(c1).KEY1	Contact Type
City	City
Phone	Phone

You'll notice that I have select to use some of the macros and some of the alias.fieldname fields so that you could get a feel for their workings. We can make the Column Name: anything at all that we desire or that would be meaningful to the dashboard viewer. Once you have added those fields, you'll want to click on the **Next >** button if you are following along in this example.

Note

*For this Data Source, I ask you to forget about the **Aggregation** value, although, I promise you that we will utilize that later.*

*All that you need to know at this moment is that the **Aggregation** value selection permits:*

AVG
COUNT
MAX
MIN
SUM

Hopefully, these are self-explanatory terms.

And doing so should bring you to the **Predefined Search Criteria** dialog page of the Wizard which you may better recognize as a SQL Query filtering condition or the **where** segment of a SQL Query, Figure 1-24.

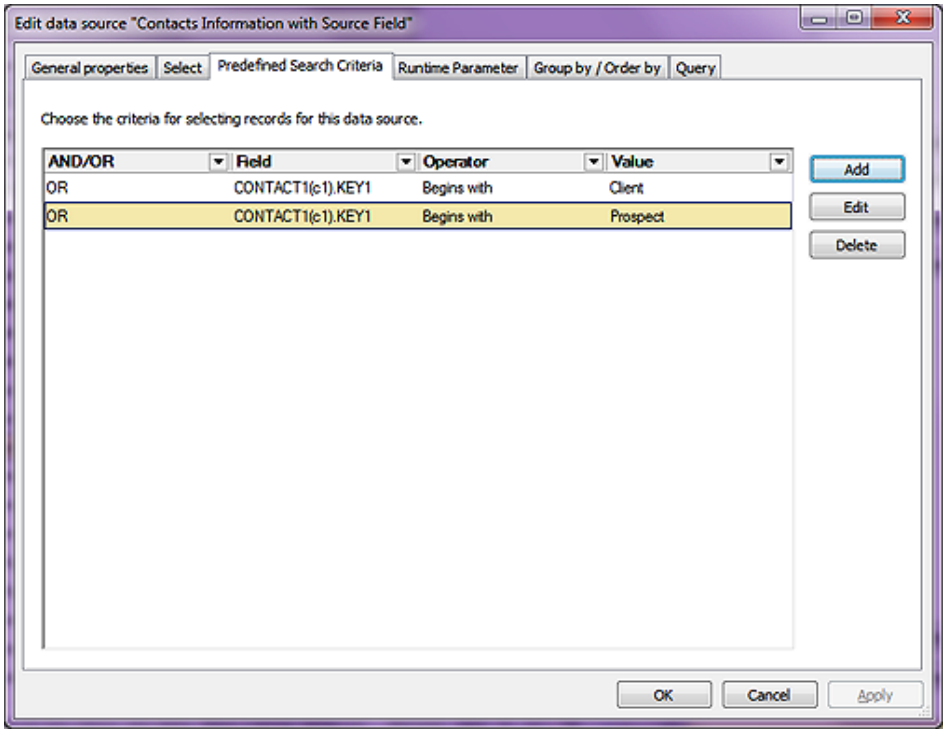

Figure 1-24

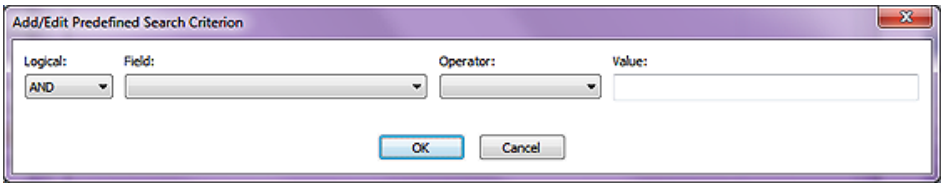

Figure 1-25

Since I have Vendors and GM Dealers in my **Contact1.Key1** field among others, I want to target this search to just my Clients and Prospects. Unfortunately, the **Value:** field, as shown in Figure 1-25, does not contain the F2 Lookup List as we're used to when creating filters such that you must manually type in the **Value:** so be extremely careful to check your spelling.

That is all that I need to add for the **Predefined Search Criteria** dialog page of the Wizard, and if I weren't now in the **Edit** mode, I would click on the **Next >** button as you should. However, because I have switched to the **Edit** mode, I will click on the **Runtime Parameter** tab so that I will be on the same dialog form as those of you following this exercise.

Note

*Please remember, from here on out through the rest of this section of this chapter, that my screenshots are taken from the **Edit** mode and will be slightly different from your Wizard displayed dialog form.*

I will be discussing them, however, as if I were still in the Wizard.

These figures are so large that it's hard to fit many on their proper page in a book of this size, so the **Runtime Parameter** dialog form is being shown in Figure 1-26 on the next page. Looking at what FrontRange has to say about this page of the Wizard, we can see that *"Runtime Parameter enable end-users to select the records to display in a dashboard. You can choose them on this page."*

Other than the crappy English, do you buy that statement? I don't, at least not as stated. We have already selected, via previous dialog form pages in this Wizard, everything that we actually want such that this is more of a staging area that the developers use to provision their data source selection code. True, one can add more runtime parameters or even modify or remove existing runtime parameters, yet for the most, you will leave this dialog form untouched.

These are normal runtime parameters that the GoldMine Wizard will add to each and every dashboard created via the Wizard. On the other hand, and for this exercise, I would like you to click on the **Add** button, see Figure 1-27 on the next page, as we are going to add a runtime parameter. Do you remember earlier that we named this dashboard: **Contacts Information with Source Field**? Well we are now going to add a runtime parameter for Source.

You'll notice in Figure 1-27, that I have already pre filled in my desired information, but I would like to explain each selection.

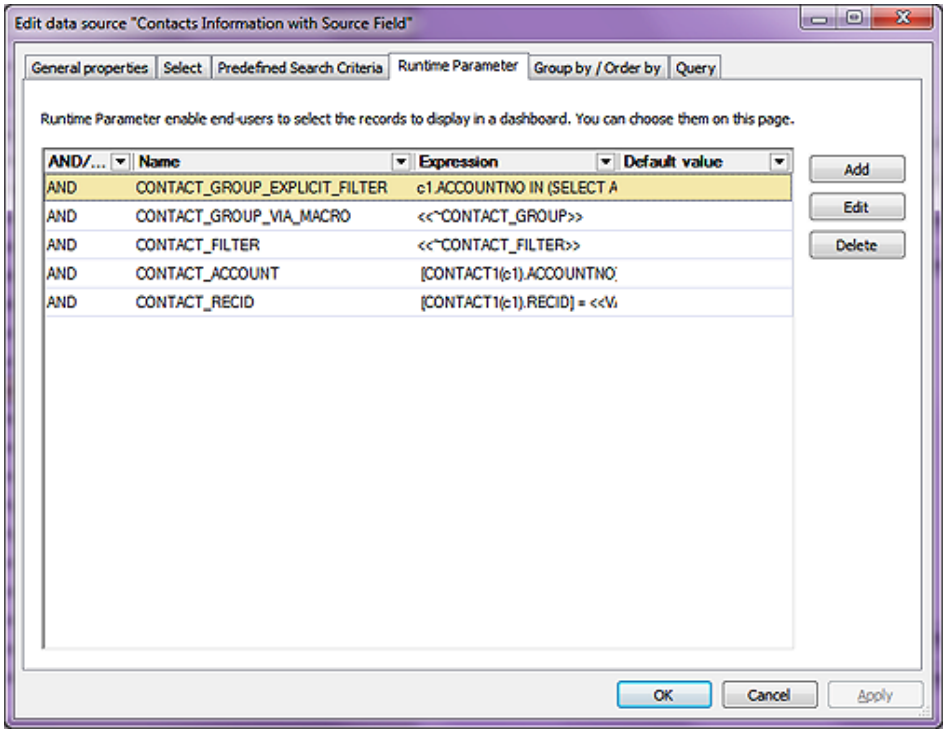

Figure 1-26

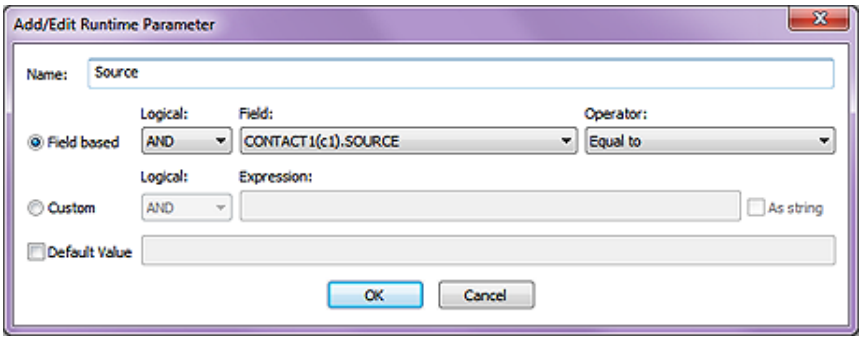

Figure 1-27

In the **Name:** field I simply ask you to enter **Source** which is a clear enough name for our runtime parameter in that I'll remember what that is next year if I rewrite this book.

I want you to accept the default ⦿ **Field based** option at this point, however, you could have just as easily selected to create a ○ **Custom** parameter or set a ☐ **Default Value**. You notice, Figure 1-26, that all of the default runtime parameters were established with **Logical: AND** instead of **OR** so I see no reason for not following suit. For the **Field:** I would like you to select **CONTACT1(c1).SOURCE**, and we accept the default **Operator:** of **Equal to**. A simple click on the **OK** button, and your runtime parameter will be added to the existing list of runtime parameters.

Now the question might be: "Why did we do that?", and the answer to which is that, later on, I may add a drill down on our Chart based on the Source, and we would want the Table View to reflect only that data for the selected Source. You'll see as we progress, but suffice it to say that we will need it for this example exercise.

Click on the **Next >** button, to bring up the **Group by / Order by** dialog form of the Wizard, Figure 1-28 on the next page. I think that the GoldMine explanation is worth repeating on this one: *"Specify how data will be grouped and sorted. You can choose among fields included to SELECT."*.

That's as good an explanation as any, however, I will ask you, for this example, to not use the **Group by:** section of this dialog form as grouping is unnecessary for this example. On the other hand, I would like you to use the **Sort by:** option. On the **Sort by:** drop list, I would have you select **Company**. **Do not** leave this screen as yet. If you do not click upon the **Add** button at this point, then the **Company will not** be added to the **Sort by:** listing, hence, the query. Click that **Add** button now please, and the click on the **Next >** button to proceed to the final dialog form of this Wizard which is the **Query** screen as shown in Figure 1-29 on the next page.

Note

An example of what a grouping and sorting statement might look like in a typical SQL Query might be:

```
select c1.Source,
   count(*) as [Count]
from Contact1 c1
group by c1.Source
order by Count desc
```

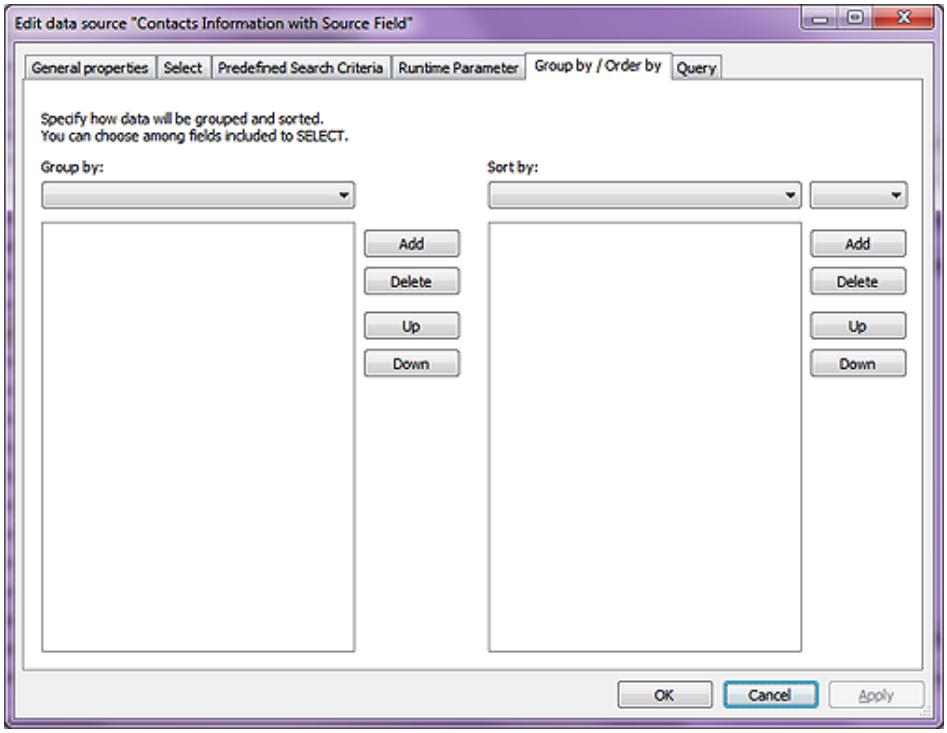

Figure 1-28

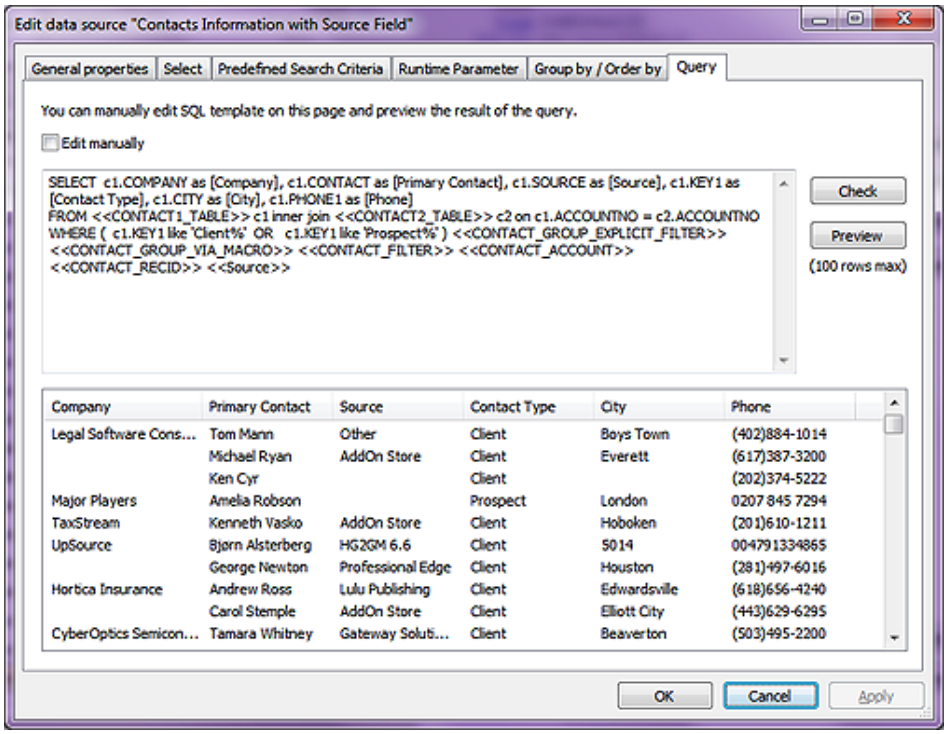

Figure 1-29

You will notice that your dialog form, at this point anyway, will not look quite like the one shown here in Figure 1-29. You see, I did two things to get it to this point. First I clicked on the **Check** button to check the integrity of our query. As we utilized the Wizard to build this query, it should have checked out just fine, and it did. So why is the **Check** button even here? Well, technically we could edit this query manually causing that integrity to become unstable. Notice the ☐ **Edit manually** checkbox just above the query.

As for the next step, I clicked on the **Preview** button to make sure that the query was pulling the correct data, and it appears to be doing that just fine. At this point, you could click on the **Finish** button, or I on the **OK** button, and I think that we should do just that.

Creating a New Dashboard

That's it. You've created your first **Data Source**, and it is ready for insertion into a Dashboard.

What say we put our new data source to some good use by creating a **New Dashboard...**. Go ahead, click on the **New Dashboard...** link in the Quick Toolbar, and then let's change the property of **Category** to **Definitive Guide Examples** as you had done previously. Why not change the **Name** property to **Definitive Guide Example 2** while you're at it.

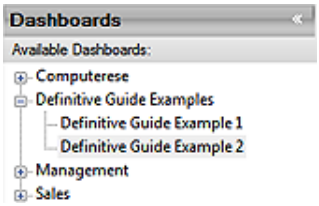

Figure 1-30

You won't need to do this as frequently later, but because you are still new at Dashboards, let's have you click on that **View** button to make certain that you're good so far. Your Quick Toolbar tree should look similar to mine as shown in Figure 1-30, and if it does, you should click on the **Design** button in preparation for moving forward on this example.

Let's go ahead an Drag & Drop a **Table View** on there now, and stretch it out to give it some size. Let's change the **Properties** just as we did in the **Changing Default Dashboard Properties** section of this chapter only this time I would have you make the **Table View Name** property **Contacts by Source**. Additionally, I would like you to change the **Data Binding** property to your new **Data Source**. Make certain that you throw in all of the columns which is most easily done by highlighting the first column in your **Unbound fields** list, and then clicking on the **->>** button to move them all at one time into the **Bound fields (columns):** list to the right. And after you have done all of this, and clicked on the **OK** button, please click on the **View** button again. If everything was correct, you should see your data in the grid just as it was seen earlier when you created your **Data Source**.

Let's click on that **Design** button once more, and make some additions. I want you to create a new **Data Source** on your own this time. I'll just give you the settings that I would be looking for you to include in this data source. The **Type of data** will again be **Contact Info**. This time you'll give it a **Name:** of **Count of Contacts by Source**, and set yourself as the **Owner:**. As well, you'll accept the default **Output type** of **Table**. For this **Data Source** you'll only require two columns, and the first of which should be **CONTACT1(c1).SOURCE**. Here's a new one for you, however, you are going to add a macro as the second column which you have not encountered as yet. Please add the **Items Count** macro as your second column. As you'll find out, this is basically an automated **Group By** statement.

Moving ahead now, please add the same two **Predefined Search Criteria** as you added when making your first **Data Source**. You won't need to add any **Runtime Parameters** as this is going to be the **Data Source** for a **Chart** on the dashboard that you are designing. While you are on the **Group by / Order by** dialog form of the Wizard, I want you to notice that the previously selected macro is now appearing in the **Group by:** listing automatically. When you get to the **Query** dialog form page of the Wizard, your query should look similar to this, only it will be unstructured:

```
SELECT  c1.SOURCE as [SOURCE],
  count(*) as [Items_Count]
FROM <<CONTACT1_TABLE>> c1
  inner join <<CONTACT2_TABLE>> c2 on c1.ACCOUNTNO = c2.ACCOUNTNO
WHERE (  c1.KEY1 like 'Client%'  OR   c1.KEY1 like 'Prospect%' )
  <<CONTACT_GROUP_EXPLICIT_FILTER>>
  <<CONTACT_GROUP_VIA_MACRO>>
  <<CONTACT_FILTER>>
  <<CONTACT_ACCOUNT>>
  <<CONTACT_RECID>>
GROUP BY c1.SOURCE
```

Does yours look something like this? I hope so. Click on the **Check**, and then the **Preview** button to see a cursor result set as extracted based upon your query. Click on the **Finish** button to wrap things up properly.

Drag & Drop a **Chart** onto your designer screen just above your **Contacts by Source Table View** much as we did before when creating a new dashboard. You should change the **Name** property to **Contacts by Source** as **Chart Name** won't be confused with your **Table View Name** as they are two separate and distinct components. Also, don't forget to change the **Data Binding** property. This time you'll want to set it up as I have done in Figure 1-31 on the next page.

Now go ahead and click on the **View** button to view the results of your working so far. You should have a properly functioning **Chart**, and a properly functioning **Table View** albeit functioning independently of each other. If you're with me so far, that's exactly where I want you to be.

At this point I'm going to have you make a modification to your **Contact Information with Source Field Data Source**. I would like you to **Sort by:** the **Company** field just in case you use Iains Video to watch what I am discussing in this book.

Note

*Have you noticed all of the extraneous statements in your query? For instance, you have no need for information out of the **Contact2** table, yet the Wizard has included an **inner join** statement joining the **Contact1** and **Contact2** tables on their respective **AccountNo** fields.*

Note

I had to do just this before I began writing this chapter. I had to contact FrontRange for instructions on exactly how to accomplish this. Alas, there was nothing written on this, and very few knew how to even design a dashboard.

Remember now that we have two independent components functioning in our dashboard, each using its own independent data source. Wouldn't it be nice if one had the ability to click on a graph unit on the chart and have the data that comprises that graph unit display below in our table view?

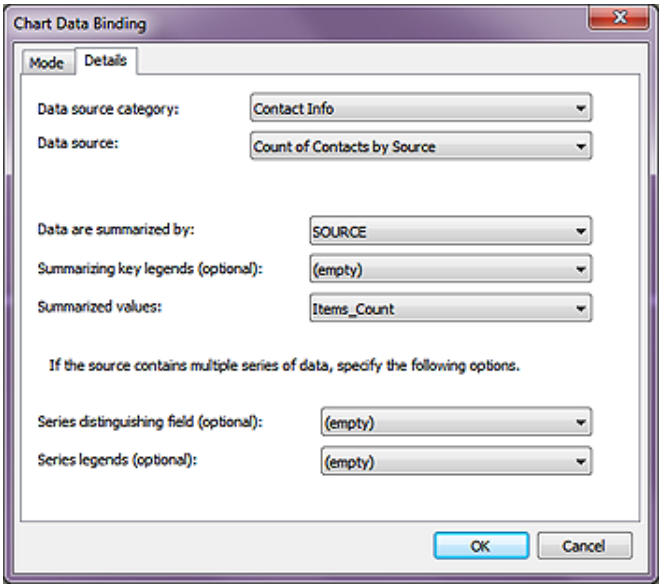

Figure 1-31

I certainly think so. Let's attempt to do just that right now by clicking on the **Events** button in the **Dashboard Design** mode to produce the dialog form as shown in Figure 1-32.

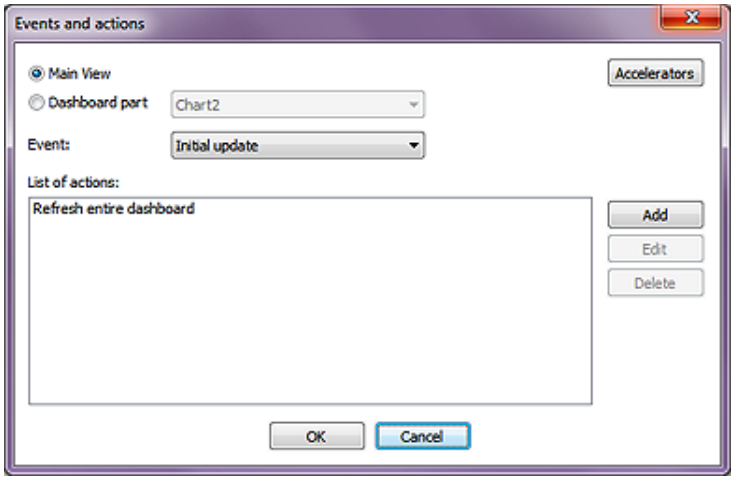

Figure 1-32

Events are the Triggers and Actions for components on the dashboards. First, one must select the component that this Trigger will function for, and then one must select the Trigger **Event:** that would cause this Trigger to be pulled. We know that this particular Trigger component is not the ⦿ **Main View**, so we'll need to select the ○ **Dashboard part** which in this case is the **Count of Contacts by Source**.

We'll then have to supply a Trigger **Event:** for which to watch by clicking on the associated drop list and selecting an event. What's that you say, *"There is only one Event."*, and you'd be absolutely correct. The **Event:** is **State changed**, and will monitor this component for any changes in state like say a **MouseDown/MouseUp** (a mouse click) **State change**.

The **Accelerators** button disables the instant that one clicks on the ○ **Dashboard part** radio button, but would have permitted one to a hotkey against to ⦿ **Main View** in case one wanted to pull the Trigger by issuing a hotkey stroke. More importantly, you have not added any actions in your **List of actions:**. That's right, you have multiple actions occur as the result of the Trigger being pulled. Because my screenshot was utilizing the GoldMine default settings there is already one action, **Refresh entire dashboard**, displayed in the list, however, as we selected the ○ **Dashboard part** radio button, we must now supply an action or actions to our empty list. I'll have you do just that by clicking on the **Add** button which brings forth the dialog form shown at the top of the next page in Figure 1-33. This is the same dialog form as I showed you back in Figure 1-13, however, I have modified this dialog form from its default state.

The **Action:** is **Drill Down** while the **Update dashboard part** is set to **Contacts by Source** which automatically picks up its **Data source:** of **Contacts Information with Source Field**. Then the **Runtime Parameter 1:** that you should use is the **Source**, and the ⦿ **Runtime (from part)** remains as set by GoldMine at this point. Click on the **OK** button at this point, and then the **OK** button on the parent dialog form, and you should be finished with this section of the example.

If you have done everything properly (always a big if), you should be able to click on the **View** button at this point and test your dashboard out. When you click on one of the graph units, your table view should actually change to display just those records that were utilized in constructing that particular graph unit on your chart.

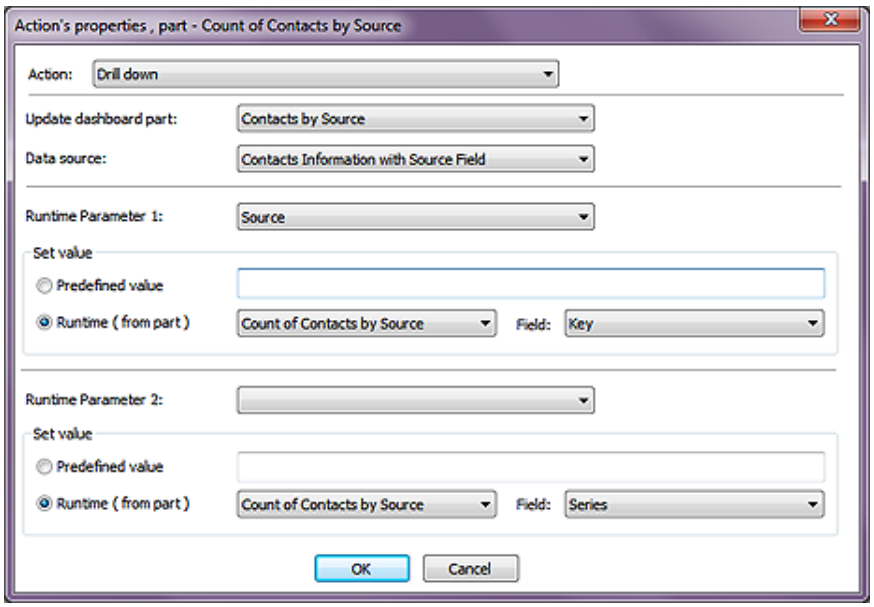

Figure 1-33

I'm thinking that yours is probably working properly at this point as mine certainly is.

Linking Dashboard Records to Contact Records

It's about time to go back and look at the linking capabilities of dashboards. GoldMine uses the **Contact1.AccountNo** field value for most of its own relationships between tables, although in some instances, GoldMine utilizes the **Contact1.RecID** field value. In order to set up a similar relationship in our dashboards, we need to capture (**select**) the **Contact1.AccountNo** field value. Please return to the **Contacts Information with Source Field** data source, and then modify the **Select** tab information by adding the **CONTACT(c1).ACCOUNTNO** field. We are done with the preparatory stage, so let's move on.

We need to make the **Contact1.AccountNo** field value part of our **Data Binding** now, so let's go ahead and **Edit** the **Data Binding** property as shown here in Figure 1-34.

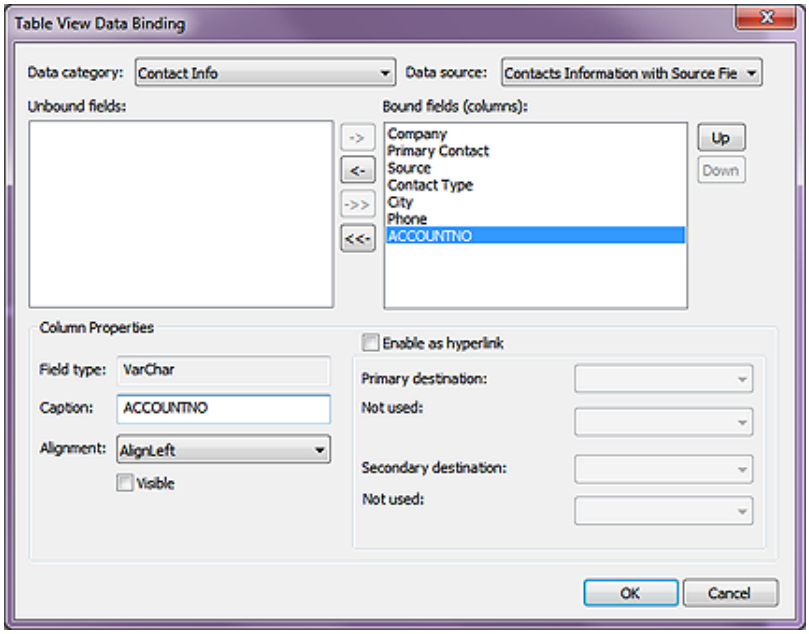

Figure 1-34

You'll notice that I have already pushed the **Contact1.AccountNo** from the **Unbound fields:** column to the last item in the **Bound fields (columns):** list, and as the information is meaningless to the enduser, I have selected to not make the column ☐ **Visible**. I'm not ready to have you leave the **Table View Data Binding** dialog form as yet, and we have more work to do.

Highlight the **Company** column in the **Bound fields (column):** list, and I'll have you work on that dialog form for that column next.

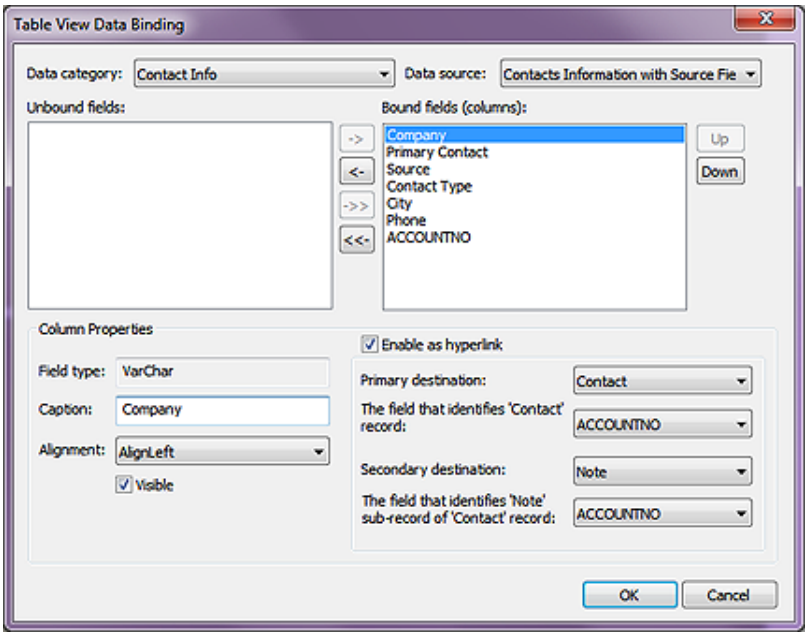

Figure 1-35

As shown here in Figure 1-35, I have already selected **Enable as hyperlink** for the **Company** column. Further, I have select the links **Primary destination:** to be the **Contact** while using the **ACCOUNTNO** as **The field that identifies 'Contact' record:**. Although, you probably don't need a **Secondary destination:** I have selected **Note**, and again **The field that identifies 'Contact' record:** will be the **ACCOUNTNO** field.

Note

If curiosity gets the better of you, and at this point it should, then I suggest that you examine all of the available destinations to which GoldMine is permitting one to link.

I do believe that we have set up all of the prerequisites at this point, hence, if you click on the **OK** button and then on the **View** button, you should see a change in your Table View now whether you click on a Chart unit or not. Does your **Company** column look similar to that which is shown here in Figure 1-36? It should. You should now be able to click on any **Company**, and the hotlink should take you directly to that record in your GoldMine. Additionally, when you arrive at that record, the **Notes** tab should be exposed if you chose the **Secondary destination:** as I had described in the previous paragraph. I do hope that you achieved your goal as well, and if not, you may want to revisit that in the book and try again.

Figure 1-36

Manually Typed Data Sources

To continue following Iains lead, we should discuss the **Manually Typed Data Source** now. The difference between the last data source that we created and the manually typed data source is that the former was produced utilizing a GoldMine Wizard to guide you through the various steps in creating a **New Data Source**. But you should know what goes into a data source at this point, and you should be able to create one manually if only you knew how to get into the proper area. You already know that the data source is nothing more than a bastardized SQL Query so let's continue with that understanding.

We'll mimic Iains videos again, and create a new Dashboard that is simple and that looks at Appointments by UserID over the last 60 days. So go ahead and click on the **New Dashboard...** hotlink, and let's get started. Immediately you'll want to set the **Properties** of **Category** to **Definitive Guide Examples**, and **Name** to **Definitive Guide Example 3**. Click on the **View** button to assure that your Dashboard appears under the correct tree branch, and then back again on the **Design** button to expose the **New Data Source...** hotlink.

Now click on the **New Data Source...** hotlink to bring us into the Wizard that we are so familiar with now, and in the **Type of data** drop list, I would have you select the **Manually typed** option. Let's leave the remainder of this screen alone, and click on the **Next >** button. Now would be a good time to complete the **Name:** value, maybe with something like **Appts last 60 days**. Don't forget to set the **Owner:** value to yourself, and click on the **Next >** button once more. We're not going to set any parameters at this time, hence, you would just click on the **Next >** button.

You now have the ability to set you SQL Query, and I'll have you type in the SQL Query as I've shown you at the top of the next page if you don't mind.

Tip

If you have more than a single database, and you wish to have this query specific to that database then your query should look similar to this one:

```
select UserID,
  count(*) as [Appointments]
from SQLGoldMine.dbo.ContHist
where sRecType = 'A'
  and OnDate >= getdate() - 60
group by UserID
order by Appointments desc
```

```
select UserID,
  count(*) as [Appointments]
from ContHist
where sRecType = 'A'
  and OnDate >= getdate() - 60
group by UserID
order by Appointments desc
```

You will learn more on SQL Queries later on in this book in Chapter 7, **Gathering the Data**. As well I'll show you more about the various table structures in Chapter 8, **The Tables**. You'll want to click on the **Check** button to assure that you enter the query properly, and if you receive the **Query is correct** screen then go ahead and click on the **Preview** button to validate that your query is producing the expected result cursor.

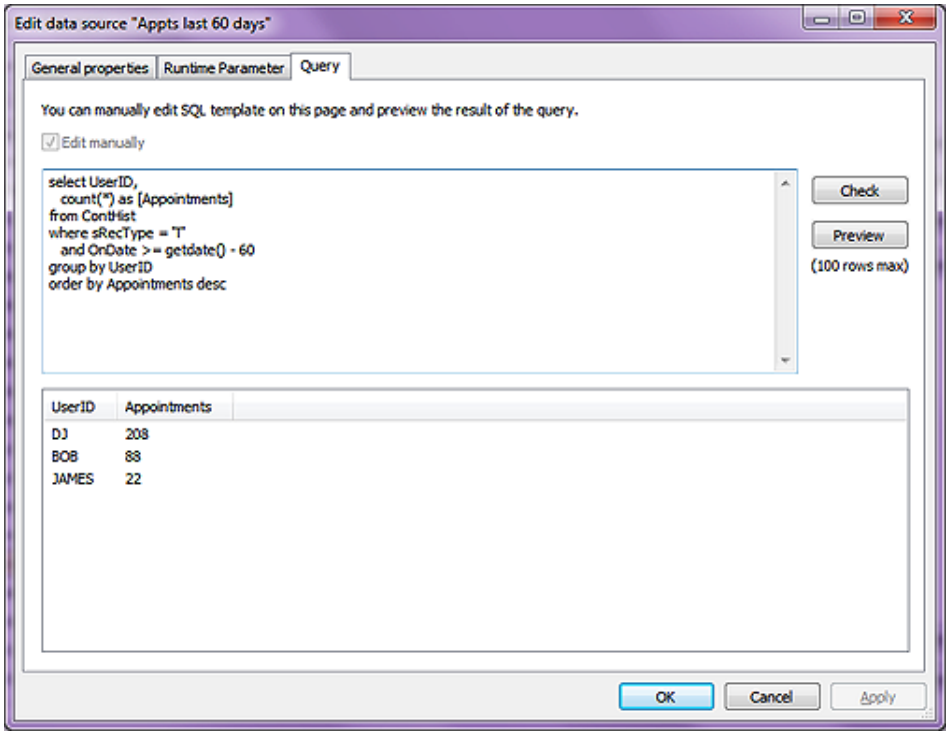

Figure 1-37

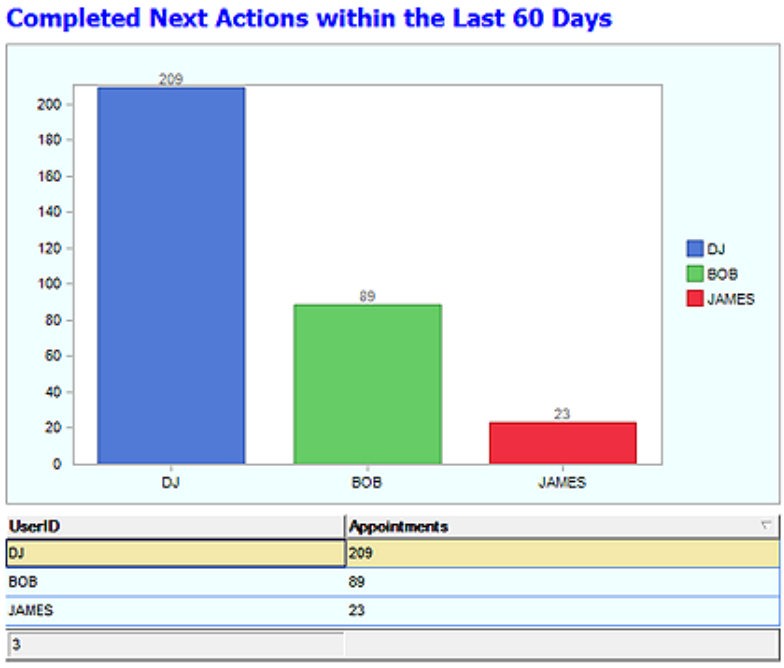

Figure 1-38

Does your Data Source dialog form look anything like mine as shown in Figure 1-37? The observant among you will notice that I have changed my SQL Query slightly as Computerese Inc does not really track Appointments as much as we track our Next Actions which are billable.

Alright then, go ahead and click on the **Finish** button to wrap up the creation of our manually entered data source. At this point you could easily add a Chart and/or a Table View to your dashboard, utilizing the **Appts in 60 Days** as your **Data Source**, and you've created another new Dashboard. You can see what I was able to accomplish using my Data Source in Figure 1-38 on the previous page.

Pivot Table

Iain has a good example of creating a **Pivot Table**, and I think that I want to follow along on this as he also incorporates another component, the **Drop-down List**, and utilizes that component to manipulate his **Pivot Table**.

Let's begin by creating a N**ew Dashboard...** utilizing the hotlink, and please throw this one into the same **Category** of **Definitive Guide Examples** while giving it a **Name** of **Definitive Guide Example 4**. You're an old hand at doing this by now I would think.

Use the hotlink, **New Data Source...**, to create, obviously, a new Data Source. This time we'll have you choose a data type of **Closed Sales**, and just start by selecting to ⦿ **Start with a blank data source**. On to the next dialog form now shall we? Here we will supply a **Name:** of **Closed Sales by Quarter**, and please assign yourself as the **Owner:**. This time, as opposed to everything that we've done previously, we're going to ask you to select the O **Matrix** option. Pivot Table - Matrix, you see the relationship a bit later on.

Note

In the list, the functions are listed at the top, and are proper case while the fields are displayed below the function and are in all caps.

Moving forward to the next dialog form we have to set the **Base field:** value. For each Quarter we want to see the **Amount**, so we're going to ask you to select the **Amount** function from the drop list as we do not want an actual field here, but a calculation. The **Summary method:** naturally will be **SUM**. And now we come to the **Pivot field:** value. We're doing Closed Sales by Quarter so the logical pick has to be a by Quarter selection. We'd like you to pick **Activity fiscal quarter** from the functions in the drop list.

Next, we need to click on the **Add Pivot** button. In the **Add/Edit pivot custom column** dialog form, let's add a **Value:** of **1**, and a **Column name:** of **Q1**. Go ahead and add that, and then click on the **Add Pivot** button once more this time adding the **Value:** of **2**, and a **Column name:** of **Q2**. So you can see that we're adding a column for each of the Quarters in a fiscal year. Go ahead and add the last two Quarter columns on your own.

Now we want to add a **Users** column so we click on the **Add** button, and in the **Add/Edit field** dialog form select a **Field:** of **CONTHIST(h).USERID** which we know is a field as it is all caps. We won't need any **Aggregation:**, and the **Column name:** of **USERID** is popped in there automatically by GoldMine with our field selection. Please click on the **OK** button now.

For the **Predefined Search Criteria**, we'll accept the default as produced by our previous **Category** selection:

```
AND CONTHIST(h).SRECTYPE Equal to S
```

Now this doesn't hold true for the **Runtime Parameters**. We would like you to add a new parameter to the default supplied listing:

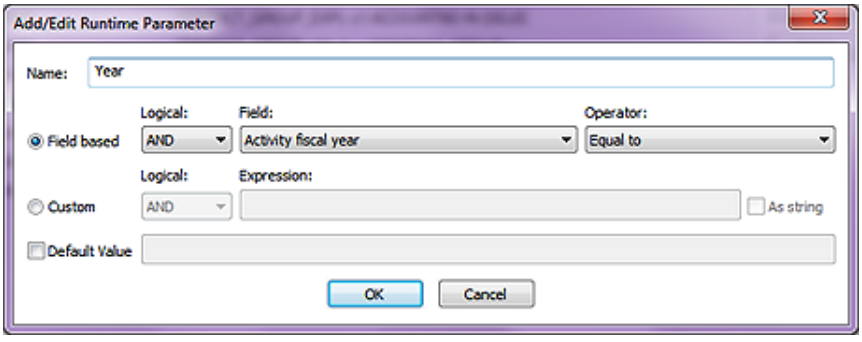

Figure 1-39

This will allow us to select, in the drop list, to affect the Pivot Table.

We'll leave the **Group by / Order by** dialog in its default state as we really have no need to change that for this exercise. You may want to **Check** your **Query** and/or **Preview** it. It should work just fine, and then you'll want to click on the **OK** button to save the **Data Source**.

This is the stage where we must begin designing our actual **Dashboard**. Why not start by dragging a **Table View** on to our palate, and stretching it to appropriate proportions. Not to feaster with it now as you can always adjust it later on. You will, however, want to supply a **Name:**, and I have used

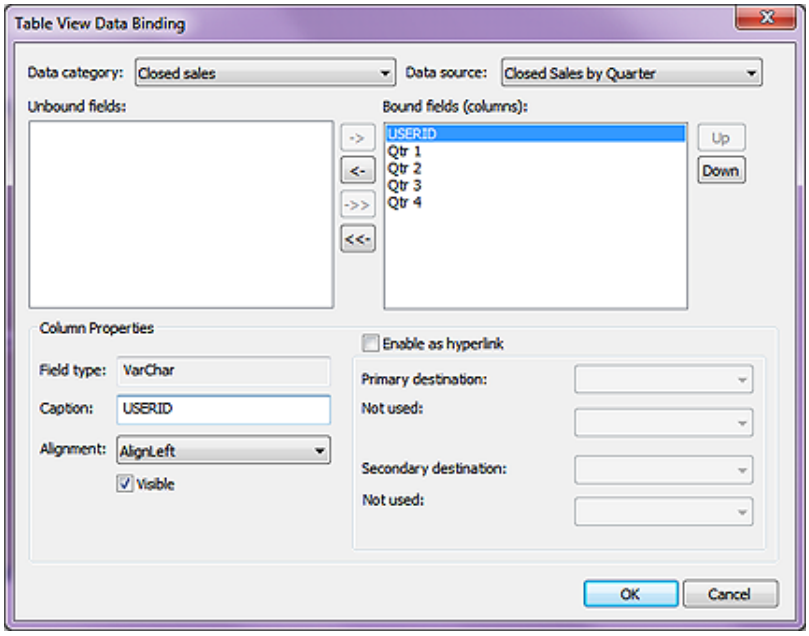

Figure 1-40

Pivot Table View for my test example. Additionally, you will want to do your **Data Binding** to your previously created **Data Source**. You can see above, in Figure 1-40, the **Table View Data Binding** dialog form from my test example. I did not select to **Enable as hyperlink** any of the fields that are bound. About the only other of the **Properties** that I changed was the **Layout Anchors** to **True, True, False, False**. That decision is strictly up to you however. I know that the curious among you, at this point, will want to see what see what you've accomplished so far, so go ahead and click on the **View** button, and when ready, click back again on the **Design** button.

One of the things that you will notice, if you clicked on the **View** button, is that you are viewing all Sales in each Quarter for all years. Now this is what we'll attempt to control via a **Drop-down List**. So go ahead and drag one of those components onto your palate, preferably above the previous created **Table View**. You may also want to add a **Label** component for your **Drop-down List** at this point as well.

We'll just give the **Label** component a **Text** value of **Year**. We'll also want to do a **Date Binding** for our **Drop-down List**. I would suggest that you use the **Manually Typed** for the **Data Category** with the predefined **Data Source** of **Years** which saves you from doing all of the hard work. Let's also change the **Text Field** value to simply **Text**, and the **Value Field** to **Value**. The **Value** will actually supply the current year from the **Drop-down List**. If you were to **View** your design at this point, you would find that everything functions properly, and probably looks great.

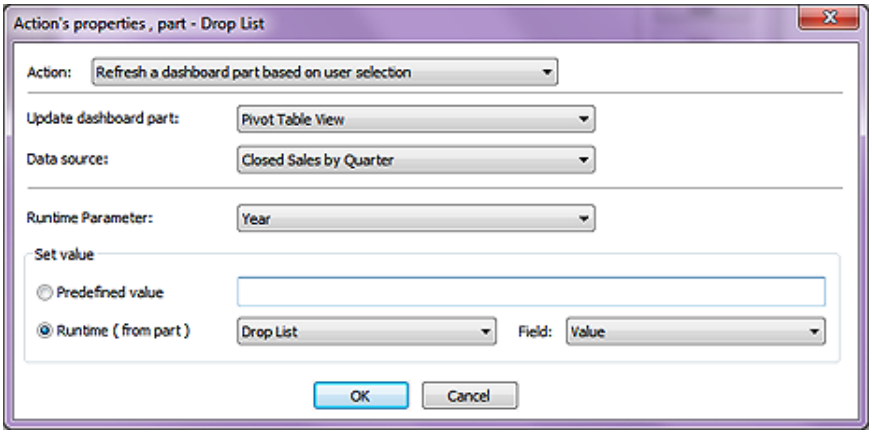

Figure 1-41

Ooops! Selecting a year from the drop list doesn't affect your table in any way now does it? I think that we have a little more work to do here. While in the **Design** mode, let's click on the **Events** button. For our ⦿ **Dashboard part**, we'll want to select your drop list by whatever name you gave it, while the actual **Event:** will be **State changed** as selected from the drop list. Please complete the **Action's properties, part Drop List** dialog form as shown on the previous page in Figure 1-41.

Click on the **OK** button, the **OK** button again, and then the **View** button. What do you think? Does your **Dashboard** function as expected now? I'm thinking that it does. You can spend some more time on this prettying up for the general populace, but for this book I think that we are finished with our **Pivot Table Dashboard**.

To keep this book a reasonable size, I must stop the **Dashboard** chapter here. Iain, however, has a few more videos on Dashboards that accompany this book. You may want to review those in your own time. My time, however, is finished, at least in this chapter.