

Microsoft SQL Server 2005 Reporting Services: Advanced Report Design (BIN305)

Ciprian Jichici
(<http://www.ciprianjichici.ro>)

Microsoft Regional Director Romania

General Manager
Genisoft (a member of MAXTEL Group)

Based on tight integration to Microsoft Visual Studio, SQL Server 2005 Reporting Services offers one of the most flexible report authoring experiences available in the market today. We're going to take a look at how to leverage all the advanced features available for designing reports supporting multiple data sources, report parameterization and report interactivity as well as how to leverage the report expression language to its fullest. In addition to demonstrating the functionalities available with SQL Server 2005 today, we'll also have a preview of some of the new report engine capabilities that will be available with the next release of Microsoft SQL Server.

Part one – Building Reports with SQL Server 2005 Reporting Services

In this part we are going to talk about some of the more uncommon tasks that can be performed with SQL Server 2005 Reporting Services. First, we take a look at some of the more advanced parameter techniques and then we'll dive into some layout techniques that can make our reports more useful to the information consumers.

Demo 1: SQL Server 2005: Report Parameters

During the demo we are using the Adventure Works sample reports from SQL Server 2005 Reporting Services. The solution containing these sample reports is located in the *%ProgramFiles% \ Microsoft SQL Server \ 90 \ Samples \ Reporting Services \ Report Samples \ AdventureWorks Sample Reports* folder.

These sample reports already show a set of techniques that can be applied in Reporting Services. What we're going to do during the demo is to further extend their functionality.

Technique 1: Add support for "Expand all" in a drilldown report

Let's take a look at the Territory Sales Drilldown report. You can see that it's fairly easy to drilldown moving from level to level (starting with Territory, going through Sales Person and ending with Order

Number). What we want to do here is add some functionality enabling us to expand all the levels at once. One way to do this is using parameters. The steps to be performed are the following:

1. Add a boolean parameter named ShowAll and mark it is as Hidden (the user is not prompted for the value, but the parameter can be provided programmatically). There is also an option named Internal for parameters that are limited to the internal context of the current report.
2. Set the default value of the parameter to be non-queried and False.
3. Add a textbox to the report, set the text to "Show All Levels", and set the navigation properties to jump to the same report.
4. Edit the parameters associated with the navigation action and set the following expression for the ShowAll parameter: "= Not Parameters!ShowAll.Value". This is going to toggle the value of ShowAll each time you click on the textbox.
5. In the table that shows Territory Sales values, go through each grouping line and change the Visibility from True to "= Not Parameters!ShowAll.Value".

One interesting effect of applying this technique is related to the number of pages in the report. If you take a look at the number of pages when you do drilldown you can see that it never changes, meaning that you basically get one big page. Applying the ShowAll technique gets you the expected number of pages when the report is completely expanded.

Technique 2: Enhance support for export in Excel

1. Export the same report (Territory Sales Drilldown) to Excel. Notice that Excel export is doing its best to preserve the layout and the behavior of the original report.
2. Unfortunately, this level of fidelity is also producing some problems. You can see in the exported Excel that sorting is kind of hard to do, there are sometimes merged cells or columns with a very small width, and so on.
3. Under the existing table in the report, add a new one and drag the following fields to it: FirstName, LastName, SalesOrderNumber, and TotalDue. This table is going to be used exclusively for exporting.
4. Add a new parameter to the report named Export. This is going to be Boolean too, with a non-queried default value of False.
5. Set the Hidden property of the new table to be "=Not Parameters!Export.Value".
6. Set the Hidden property of everything else in the report to be "=Parameters!Export.Value".
7. Add a new textbox to the report, set the text to Export, and set the navigation properties to jump to the same report. Edit the parameters and set Export to be true.

You can now click on Export and your report will change to a format that is much easier to export to Excel. You can also choose to export to CSV if you do not need to preserve the layout of the original report. Another interesting option that you can apply is, instead of setting the Export textbox to jump to the same report, you can set it to jump to a link that automatically exports your report to Excel or CSV. I'll leave that as an exercise for you ☺.

Technique 3: Displaying data source fields in headers and footers

1. Open the Product Line Sales report.
2. Sometimes you need data-dependent things in the header and footer of the pages. Unfortunately, Reporting Services does not allow you to drop fields in the header and footer. We'll try to get a workaround based on parameters.
3. Create a new dataset named WebSite that queries for the web site name. For the purpose of this demo, all I did was to create a table in the database that holds the name of the web site and point my dataset to query that particular table.
4. Create a new string parameter called WebSite. Set the parameter to be Hidden. The available values of the parameter are coming from a query, namely from the WebSite dataset. The default values are also coming from a query, via the WebSite dataset.
5. Go to the footer of the report and change the value of the FooterURL textbox to "=Parameters!WebSite.Value". Also set the navigation property to "= "http://" & Parameters!WebSite.Value".

With this technique you can add aggregate values and scalar values to the headers and footers of your reports. What you can't do this way is getting multirow values in those headers and footers.

Technique 4: Localizing reports

1. Right now, there is no localization support built into the product. One way to do it is through a custom assembly. This will work if you replace all your static text with calls to that particular assembly. What we're trying to do is add some localization support using parameters.
2. We'll use the Product Line Sales report again.
3. Create a dataset named Labels that points to a table named Translations with the following columns: Label, Language, and Translation. The query you should use is the following:

```
SELECT Label, Translation
FROM Translations
WHERE Language = @Language
```

4. Go to report parameters and change the Language parameter by setting its available values to be non-queried. Add some values to the list like English with a value of en-US, French with a value of fr-FR. Set the default value to be non-queried and equal to en-US.
5. Add a new parameter named Labels. Set it to be Hidden and Multi-value. The available values come from the Labels dataset. The value field is Label and the label field is Translation. The default values come also from the Labels dataset with the same settings as the ones for available values.
6. Go to Report Properties and switch to the Code tab. Create the following function:

```

Public Function GetLabel(P as Parameter, Label as String) as String
    Dim i as Integer
    For i = 0 to Ubound(P.Value)
        If (P.Value(i) = Label) Then Return P.Label(i)
    Next i
    Return Label
End Function

```

7. Change the value of the TopEmployeesHeader textbox to “=Code.GetLabel(Parameters!Labels, “Top Employees”)”.
8. Change the value of the TopStoresHeader textbox to “=Code.GetLabel(Parameters!Labels, “Top Stores”)”

This technique gets about half way towards getting full localization for your reports. The other half is the actual localization of the data you use in the report. If your data source is Analysis Services that’s quite straightforward. If your data source is a relational database, you need to build that support into your database.

In general, the usage of parameters in Reporting Services is most of the time limited to the classical role of providing values that filter the data used to build the report. In fact, there are lots of other quite interesting uses for parameters which make them a real friend for the Reporting Services report developers. Some of these possible uses are:

- Dynamic queries (you can use parameters to alter the queries you use to get the report data)
- Variable grouping (you can use parameters to change the grouping fields in you reports based on the values of the parameters)
- Fields in page header and footer
- Currency translation (be careful with localization settings in .NET Framework)
- Language translation
- Self-drill reports

Another area where we can do a lot to improve the functionality of our reports is layouts. Sometimes, there are things that we would like to have in our reports that are not supported by the current version of the report items available in Reporting Services. Some of the typical examples include:

- Combination of dynamic and static columns in a table (like dynamic columns holding data for calendar years combined with static columns holding data for demographics).
- Parallel dynamic columns (like having more than one hierarchy share the same axis). Today’s limitation is that you can have only one hierarchy per axis.
- Details with no subtotal (for some elements it doesn’t make sense to have totals).

- Stepped matrix (instead of having a separate column for each level in a hierarchy, we'd like to have a single column where values are stepped according to the level they occupy in the hierarchy).

Demo 2: SQLServer 2005: Report Authoring

Before getting into the details of report layouts, we need to change a little bit the CompanySales report (to make it better suited for the purpose of our discussion). All you need to do is change the query of the dataset to the following:

```

SET DATEFORMAT mdy
SELECT
    PC.Name AS ProdCat,
    PS.Name AS SubCat,
    DATEPART(yy, SOH.OrderDate) AS OrderYear,
    'Q' + DATENAME(qq, SOH.OrderDate) AS OrderQtr,
    SUM(SOD.UnitPrice * SOD.OrderQty / 100) AS Sales,
    Sales.SalesTerritory.Name as Territory,
    Sales.SalesTerritory.[Group],
    SUM(P.StandardCost * SOD.OrderQty / 100) as Cost,
    SUM(SOD.OrderQty) as Qty

FROM
    Production.ProductSubcategory PS
INNER JOIN Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail SOD
    ON SOH.SalesOrderID = SOD.SalesOrderID
INNER JOIN Production.Product P
    ON SOD.ProductID = P.ProductID
    ON PS.ProductSubcategoryID = P.ProductSubcategoryID
INNER JOIN Production.ProductCategory PC
    ON PS.ProductCategoryID = PC.ProductCategoryID
INNER JOIN Sales.SalesTerritory
    ON SOH.TerritoryID = Sales.SalesTerritory.TerritoryID
INNER JOIN Sales.CreditCard
    ON SOH.CreditCardID = Sales.CreditCard.CreditCardID
INNER JOIN Production.ProductModel
    ON P.ProductModelID = Production.ProductModel.ProductModelID

WHERE
    (SOH.OrderDate BETWEEN '1/1/2002' AND '12/31/2003')
GROUP BY
    DATEPART(yy, SOH.OrderDate),
    PC.Name,
    PS.Name,
    'Q' + DATENAME(qq, SOH.OrderDate),
    PS.ProductSubcategoryID,
    Sales.SalesTerritory.Name,
    Sales.SalesTerritory.[Group]

```

Next, create a report named SubCatDrillthrough. This should be a very simple report, with two parameters: ProdCat and SubCat. All it does is displaying the values of the parameters in two text boxes.

Create another report named CatDrillthrough. This too is going to be a simple report, with only one parameter: ProdCat. All it does is display the value of the parameter in a text box.

Now we're ready to move on.

Technique 1: Nesting values in the same column/row

1. Open the CompanySales report.
2. Add a Subtotal at the subcategory level (the SubCategory text box).
3. Change the value of the Subtotal by copying the value from the Category textbox.
4. Notice the green triangle on the upper right corner of the Subtotal textbox.
5. For the Subtotal, set the Backcolor to LightGray and position to Before.
6. For the SubCategory grouping, set the Visibility to be toggled by the Subtotal textbox. Then set the padding to 20pt and remove the borders.
7. Delete the text from the Category text box. Unfortunately, with the current version you cannot delete completely the column, but you can select the cells and set Background to Transparent, then resize the column to make it as narrow as possible (the lowest value is 0.0125 inches).
8. Repeat the same steps for columns. The only difference is that you do not need to set the Visibility property (it will work automatically).

Technique 2: Side-by-side matrixes

1. Duplicate the existing matrix in the CompanySales report and place the copy on the right side of the original.
2. Leave the rows as they are and change the columns as follows:
 - a. Change the OrderQtr to Territory.
 - b. Change the Year to Group

Make sure you change the Visibility settings because they are not reset when you copy a matrix.

3. Set CanGrow to false on the textboxes associated with the grouping columns.
4. Set the visibility to True for the subcategory textboxes.
5. Get rid of the category and subcategory textboxes from the second matrix (using the same techniques as described before: eliminate borders, set background to Transparent, and resize).
6. Adjust the horizontal position of the second matrix to eliminate the gap between the two matrixes. Be careful to check whether the final result looks good in both the preview and the published forms of the report.

Technique 3: Using InScope()

InScope is a quite powerful function that is especially useful in matrixes. It tells you the current grouping context you're in.

1. Change the Navigation properties on the Sales textbox by making it navigate to the SubCatDrillthrough report. Set the ProdCat and SubCat parameters to “=Fields!ProdCat.Value” and “=Fields!SubCat.Value” respectively.
2. With this setting, everything works fine and you can move the the SubCatDrillthrough report from any subcategory total. The problem is that when you click on a category total, you go to the same report, and the subcategory parameter is automatically set to the first subcategory of the selected category. This is probably not what you want to do. Instead, you probably want to go to a different report, maybe CatDrillthrough.
3. Go to the Navigation properties of the Sales text box and change the static for the Jump to Report option to the following expression: “=IF(InScope(“ProductSubcategory”, “SubCatDrillthrough”, “CatDrillthrough”))”.
4. Set the Omit expression for the SubCat parameter to “= Not InScope(“ProductSubcategory”))”.

Let's review a few of the most interesting issues related to report layouts:

- **Parallel groups**

The typical situation where you need parallel groups is when you have a shared axis but want to look at two or more different views.

In this case, you will use side-by-side Tables or Matrices with the following elements:

- Use sorting to ensure proper alignment
- Turn off “can grow” or allow sufficient space for row alignment
- Matrix “Totals” do not take up space, overlap the “Total” area

The most important limitations that apply are the following:

- You cannot use interactive sorting
- Expand/collapse only works on columns
- There will always be a little space before a matrix header (you can use grid size to minimize)

- **Stepped matrix**

- Add subtotal
- Use the “Green triangle” (setting the position to Before)
- Replace the value of “Total” in the subtotal with the outer group value

- Clear the value from the outer group
 - Minimize the outer group column width
 - Set left padding on the inner group
- **Different details from subtotal in matrix**
- Add the desired value to the detail
 - Set the visibility property to false using InScope()

The most important limitations that apply are:

- The space in the Subtotal does not go away, but the value does
- Does not work for the inverse; too much space is wasted
- Not practical for many values

Part two – Building Reports with SQL Server 2008 Reporting Services

Now that we've seen some of the more advanced techniques we can apply to reports in SQL Server 2005, let's take a look to the new things to come in SQL Server 2008. Some of the most interesting among these new things to come in SQL Server 2008 Reporting Services are:

- Tablix (unifies Table and Matrix)
 - The Table has several important advantages: grouping support, headers and footers for each group, and so on. The big problem with the table is that you are stuck with the original set of columns. There is no way for a table to have dynamic columns.
 - The Matrix on the other hand has nice capabilities to grow horizontally. Unfortunately, you are limited to a single hierarchy per axis and you don't have too much control over the grouping and header/footer part.
 - Tablix unifies the two of them keeping the advantages and eliminating most of the limitations.
- New charts and gauges (a complete upgrade from Dundas)
- New designer
 - Data pane (previously just fields)
 - Improved dialogs
 - Snap to lines
 - Zoom
 - Integrated into Visual Studio and standalone

- Grouping pane
- The green triangle is gone!

The Tablix, as I mentioned before, is bringing us the best of the two worlds (table and matrix) plus some more. Some of the most interesting features of the Tablix are:

- Fixed and dynamic columns and rows (think about the Tablix as having a tree on each axis. For each tree, some subtrees can be static and some can be dynamic).
- Arbitrary nesting on each axis (ragged hierarchies).
- Multiple parallel row/column members at each level (you can get rid of those textboxes without having to set them to a very small size).
- Optional omission of row/column headers.

Demo 3: SQL Server 2008 Reporting Services

The demo shows some of the features of the new report items in SQL Server 2008 Reporting Services. The demo script is not included because most probably some things will change in future CTPs (the demo is using CTP4).

Some interesting things to note about the new designer are:

- Smart tags (for example when you use the Table report item).
- Snap lines (allow you align more easily things on the design surface).
- A new grouping pane added to ease the design of groupings.
- The new tokenization support (simplifies the display of the expressions you enter in text boxes).
- Improved handling of images used in a report
- Both the Table and the Matrix items have a Tablix object behind them. The only difference is in the initial setup of the Tablix.

Demo 4: SQL Server 2008 Charting

The demo shows some of the features of the new charts in SQL Server 2008 Reporting Services. The demo script is not included because most probably some things will change in future CTPs (the demo is using CTP4).