Improve report building with parameters

Power BI Days Belgium

April 13th, 2019

macaw

1 Table of contents

2 Lab context

During this workshop you will optimize your report building process. You will do this by applying parameters to your Power Query steps. With these parameters there will be only a small set of data loaded which improves performance on your local machine, but also speed-up the process and makes it easier to validate the numbers.

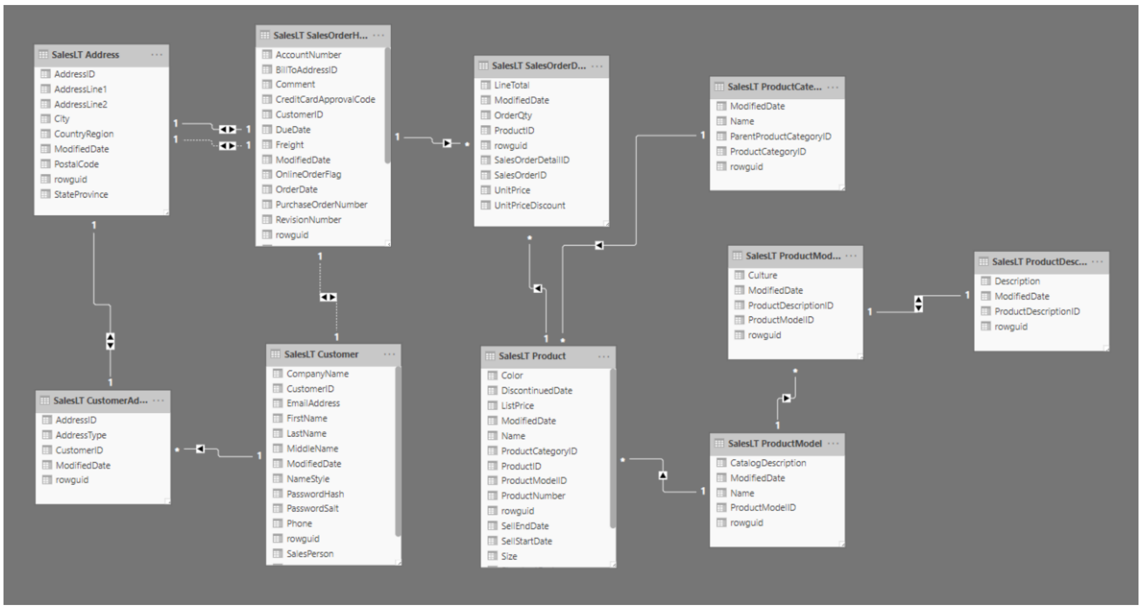
For this case, you will use an Azure SQL Database. This database includes the Adventure Works demo dataset which is provided by Microsoft. For a basic understanding of this data, this dataset is all about sales of product by a reseller. The specific case of the data doesn't matter that much for this process.

Basic understanding of definitions during this lab:
Fact / Fact tables: An Fact tables contain the data corresponding to a particular business process. Each row represents a single event associated with a process and contains the measurement data associated with that event. For example, a retail organization might have fact tables related to customer purchases, customer service telephone calls, and product returns.
Dimension / Dimension tables: Dimensions describe the objects involved in a business intelligence effort. While facts correspond to events, dimensions correspond to people, items, or other objects. In the retail scenario used in the example above, we discussed that purchases, returns, and calls are facts. On the other hand, customers, employees, items, and stores are dimensions and should be contained in dimension tables.
Parameter: Parameters allow users to easily make parts of their reports and data models (such as a query filter, a data source reference, a measure definition, etc.) depend on one or more parameter values. With the new Query Parameters feature, users can now easily define one or multiple parameters to be used in their queries, Data Model and report layers in Power BI Desktop. Users can define new parameters by using the "Manage Parameters" dialog in the Query Editor window.

Content obtained from, and more details available at:
- *https://www.lifewire.com/facts-vs-dimensions-1019646*
- *https://powerbi.microsoft.com/en-us/blog/deep-dive-into-query-parameters-and-power-bi-templates/*

3 Set up a connection to an Azure SQL Database

We kick-off a connection to an Azure SQL Database. This database will have the Adventure Works demo dataset. Below is a schematic view of the datamodel and all included tables in this dataset.



1. To start with the connection, please go to the ribbon in Power BI, and click ***Get Data***.

2. Since the Azure SQL Database option is not listed in the first options, click ***More***.

3. In the screen which pops-up, please search for ***Azure SQL database*** and click ***Connect***.

4. Please use the below shown server and database name to connect with the database.
   Server: powerbidaysdemo.database.windows.net
   Databasename: DemoTopN

   For the data connectivity mode, please use ***Import***.

5. After clicking ***Ok*** you will be prompted to enter credentials. Please ask the workshop instructor to provide you credentials. In this case we will use ***Database*** authentication.

6. A window will appear where you can select the tables you want to load. Please select the below tables to be loaded which are two fact tables and one dimensional table:
   - SalesLT Product
   - SalesLT SalesOrderDetail
   - SalesLT SalesOrderHeader

7. After selecting the above tables, click ***edit*** before you load the data.

4 Adding parameters

We will generate two parameters to improve our report building. First of all we will create boolean parameter which results in a true or false which tells us if the file is in development or not. The second parameter will be a decimal number to define the numbers of rows to be returned from the source.

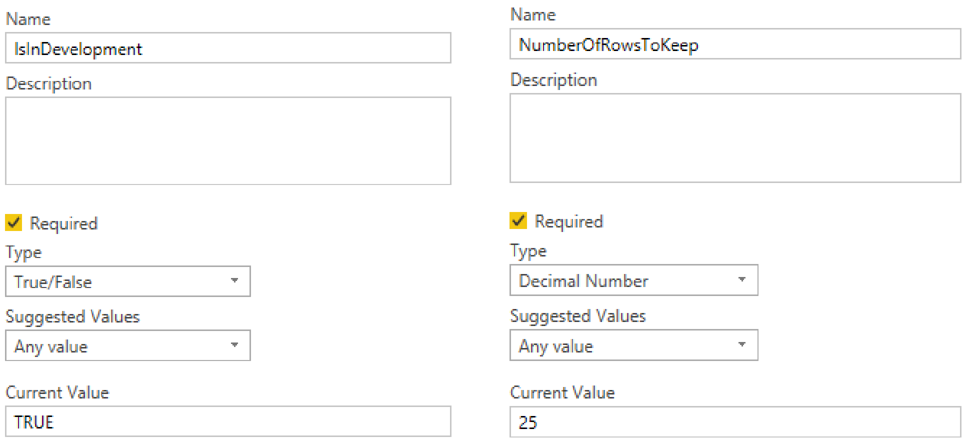
1. After the Query Editor opened up, we will see the three selected tables on the left side in the queries pane. In this section we will start adding parameters. First we will generate groups to separate the fact and dimensional tables.

   Right-click the product table, click ***Move to group***, then ***New group*** and enter a logical name for later reference. For example you can call it dimensions. The Product table is a dimensional table since this table doesn't include any transactional data, but only detailed information about the products which are sold.

2. Next up is doing the same for the fact tables. Please select both other tables, SalesOrderDetail and SalesOrderHeader, right click again and create another group. For example name this group Facts. You can multiple select tables by holding the shift or control + select.

3. In the top ribbon, make sure you are in the ***Home*** tab and click ***Manage Parameters***.

4. Click ***New*** and create the following two parameters:



Name: IsInDevelopment	Name: NumberOfRowsToKeep
Description:	Description:
✓ Required	✓ Required
Type: True/False	Type: Decimal Number
Suggested Values: Any value	Suggested Values: Any value
Current Value: TRUE	Current Value: 25

5. After clicking ***Ok***, select both parameters and create a new group to keep your query editor clean.

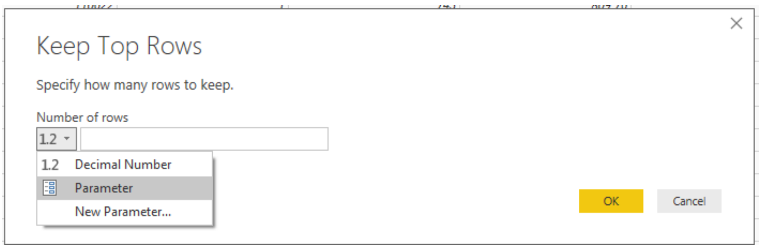
5 Apply Top N rows

Now, the parameters are created and our query editor looks clean, please check one this before you continue on. Please go to ***File > Options and settings > Options***. Under ***Power Query Editor*** make sure that the toggle for ***Display the Formula Bar*** and ***Enable M intellisense in the Formula bar, advance editor, and custom column dialog*** is set on.

1. If you closed it, please go to the query editor and select the ***SalesOrderDetail*** table.

2. In the top ribbon click ***Keep rows*** and choose for ***Keep top rows***.

3. In the window which opens up, please select the small dropdown and choose for ***Parameter***.



Keep Top Rows

Specify how many rows to keep.

Number of rows

4. If the parameter isn't selected yet, please select the earlier created parameter which is named ***NumberOfItemsToKeep*** and click ***Ok***.

5. Since we enabled the Formula bar, we see that the following Power Query step is applied:
   = Table.FirstN(SalesLT_SalesOrderDetail, NumberOfRowsToKeep)

   In this case the parameter is entered in the name without brackets around it because there are no spaces in the parameter name. In Power Query you can call a parameter by putting round brackets around it. So the above Power Query does exactly the same as the below code with the two added brackets in red:
   *= Table.FirstN(SalesLT_SalesOrderDetail, **(**NumberOfRowsToKeep**)**)*

6. So we only have a subset of rows left for our fact table. But actually we only want to have this limited number of rows when the report is in development. So we need to add our other earlier parameter to the code as well. We will do this by using an if statement. We will do that by opening the advanced editor since we need to apply some custom code as well.

   Please go to the top ribbon and in the home section, click ***Advanced Editor***.

7. Our Power Query code in the Advanced Editor will now look like this:

```
let
   Source = Sql.Databases("Powerbidaysdemo.database.windows.net"),
   DemoTopN = Source{[Name="DemoTopN"]}[Data],
   SalesLT_SalesOrderDetail = DemoTopN{[Schema="SalesLT",Item="SalesOrderDetail"]}[Data],
   #"Kept First Rows" = Table.FirstN(SalesLT_SalesOrderDetail, (NumberOfRowsToKeep) )
in
   #"Kept First Rows"
```

What you already might know, each Power Query does have an unique name. The row always starts with this name. Besides that each function always calls the result of the step before by calling the name of this step. See below code where the step names and calling the result in the step afterwards is highlighted with colors.

```
let
   Source = Sql.Databases("Powerbidaysdemo.database.windows.net"),
   DemoTopN = Source{[Name="DemoTopN"]}[Data],
   SalesLT_SalesOrderDetail = DemoTopN{[Schema="SalesLT",Item="SalesOrderDetail"]}[Data],
   #"Kept First Rows" = Table.FirstN(SalesLT_SalesOrderDetail, (NumberOfRowsToKeep) )
in
   #"Kept First Rows"
```

8. Next up, is changing our last line of code to respond to our other created parameter. This parameter was the boolean value which tells us if this report is in development or not. We will do that by adding an if statement to the last line of code. All added code is below highlighted in blue.

```
#"Kept First Rows" = if (IsInDevelopment) = true then Table.FirstN(SalesLT_SalesOrderDetail, (NumberOfRowsToKeep) ) else  SalesLT_SalesOrderDetail
```

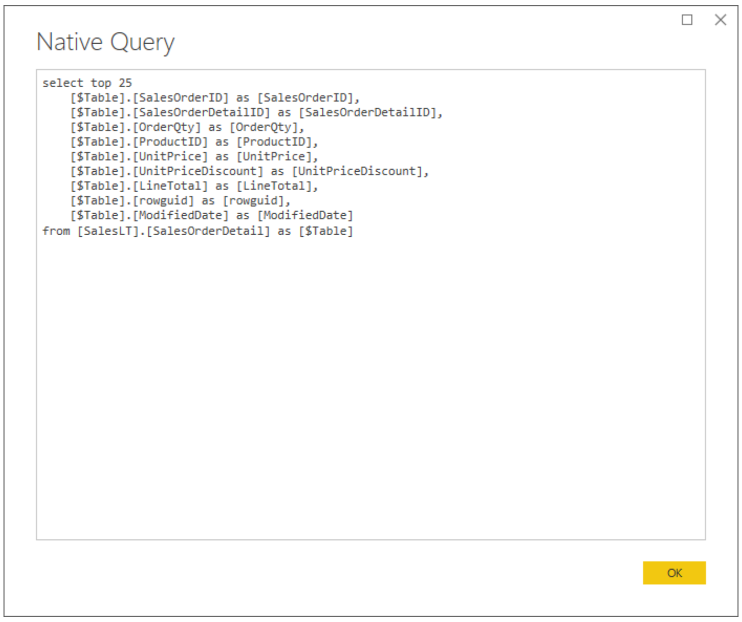
In other words: If the parameter (IsInDevelopment) is set to TRUE, then return a subset of our data corresponding to the number of rows defined in the other parameter (NumberOfRowsToKeep), else return the result of the step before, which still included all rows.

Before you add this step, please be aware of the fact that Power Query is case sensitive!

9. After you have added above explained steps, click ***Done*** and check the results by changing the parameter.

10. Once you have closed the Advanced Editor, please go to your ***Query Settings*** on the right side of your screen. Here you can find all applied steps in your Power Query. Please right click the last step you've taken, which should be ***Kept first rows*** and see that the option ***View Native Query*** is available. Take a look at the difference in the native query by changing the ***IsInDevel-opment*** parameter.

    This functionality is called ***Query Folding*** and will push back the query to the original data-source. That means that the query won't be executed within Power BI, but on the source side. Only the results will be loaded into Power BI. As a result, your report building and perfor-mance increased enormously since your local memory doesn't have to handle the load.



```
select top 25
    [$Table].[SalesOrderID] as [SalesOrderID],
    [$Table].[SalesOrderDetailID] as [SalesOrderDetailID],
    [$Table].[OrderQty] as [OrderQty],
    [$Table].[ProductID] as [ProductID],
    [$Table].[UnitPrice] as [UnitPrice],
    [$Table].[UnitPriceDiscount] as [UnitPriceDiscount],
    [$Table].[LineTotal] as [LineTotal],
    [$Table].[rowguid] as [rowguid],
    [$Table].[ModifiedDate] as [ModifiedDate]
from [SalesLT].[SalesOrderDetail] as [$Table]
```

11. Apply the above steps for each fact table in your datamodel. Don't apply these steps to your dimensional table because they need to be complete to build your report to a realistic da-taset. With applying the parameters to your fact table, you will only shorten the number of or-ders in case of this dataset, but still have all products available in the dimensional table.

12. All done with editing your queries? Click ***Close & Apply*** to load the subset of your data.

6 Change parameters in the Power BI Service

Now we've build a report with a subset of data in Power BI Desktop. The performance of our report building was better than ever before! Now it's time to publish our report to the Power BI service and start sharing the insights with others. Before we do that, we have to make sure that we load the complete set of data, but we don't want to do that on our local machine. Luckily we can change the parameters in the Power BI service.

1. After you are done building your report, click ***Publish*** and select the workspace where you want to share the content.

2. After publishing, go to the workspace and on the right top, click the ***Settings icon*** and click ***Settings***.

3. Go to the ***Datasets tab***.

4. Click the option for the ***Parameters*** and see that you will be able to change the parameter in the Power BI Service.

5. Change the ***IsInDevelopment*** parameter into FALSE. Don't forget to click ***Apply*** in the bottom.

6. If we take a look at the report now, nothing is changed yet. To load the full set of data we have to manually trigger the dataset to refresh or wait till the next scheduled refresh.

macaw

Challenge accepted.