***Featherman’s T-SQL Adventures: Two additional comments about SQL sub-queries***

Previously you learned about *data explosion* which can occur when you connect two fact tables in the same JOIN, specifically using a FULL JOIN. Sometimes however the data in a fact table does not require summarization (aggregation). If a fact table is actually fewer lines of data that use measures in a summary format (ie no aggregating of many rows are needed) then you can join two fact tables together in one query (AdventureworksDW2014 FactFinance, and FactSalesQuota are examples). Here is an example to explain.

The second part of this document shows a PowerBI approach to solving the problem of conencting two fact tables, and performs quite well, if you are only needing to build a report and not performing ETL.

First a warm-up: This document shows how to use sub-queries to a) build two columns of metrics, and a third column that subtracts one column from another   
and b) how to use sub-queries to filter the rows. Go ahead and run the query without the WHERE statement to see that you would not filter out bad data

|  |  |
| --- | --- |
| --you can copy this code and run it.  USE [AdventureWorksDW2014];  *--here the level of detail is reseller within states. Two sub-queries are used to create two columns of sales 2011, and 2012*  SELECT [StateProvinceName], [ResellerName]  , (SELECT ISNULL(Sum([SalesAmount]), 0) FROM [dbo].[FactResellerSales] as s  WHERE s.ResellerKey = r.[ResellerKey]  AND YEAR(Orderdate) = 2011) as [2011 sales]  , (SELECT ISNULL(Sum([SalesAmount]), 0) FROM [dbo].[FactResellerSales] as s  WHERE s.ResellerKey = r.[ResellerKey]  AND YEAR(Orderdate) = 2012) as [2012 sales]  *--now we can subtract one column from another*  , (SELECT ISNULL(Sum([SalesAmount]),0) FROM [dbo].[FactResellerSales] as s  WHERE s.ResellerKey = r.[ResellerKey]  AND YEAR(Orderdate) = 2012) -  (SELECT ISNULL(Sum([SalesAmount]),0) FROM [dbo].[FactResellerSales] as s  WHERE s.ResellerKey = r.[ResellerKey] AND YEAR(Orderdate) = 2011) as [Delta]  from [dbo].[DimReseller] as r  JOIN [dbo].[DimGeography] as g ON g.GeographyKey = r.GeographyKey  *-- this WHERE statement filters out the resellers that did not have sales in either year*  WHERE (SELECT ISNULL(Sum([SalesAmount]), 0)  FROM [dbo].[FactResellerSales] as s WHERE s.ResellerKey = r.[ResellerKey]  AND YEAR(Orderdate) IN (2011, 2012)) > 0  ORDER BY Delta | So in this example we need a separate column for 2011 sales and another separate column for 2012 sales. What other way is there to build this dataset? PIVOT?  Before we explain the code logic, if you run this query, perhaps add a slicer using an NTILE(4) command to put the resellers into groups based on increasing or increasing sales. A map report might uncover a geographic effect such as lower sales in the southern regions.  here the level of detail is reseller within states. Line 4 to 6 is one sub-query that totals data. Notice the SQL code in the red box, there is no GROUP BY functionality. Each subquery knows to provide the SUM() aggregation by the granularity of the outer query (Here reseller ).  You can make as many columns of metric that are needed in the report requirements. As you can see the SQL code in lines 12-16 is very tedious. This type of column comparison is much more easily accomplished in Excel, PowerBI, etc. |

***Sometimes you can JOIN one fact table to another - comparing performance to goal*** - How are the salesreps performing, let’s examine the data. The dataset has a table that has the sales quotas. We can see that for the year 2007, employee 272 had 4 different sales quotas, one for each quarter. We would like to compare this quota to the sum of sales that we can calculate from the FactResellerSales table. The problem is that the data is in two different fact tables. The solution to this data management problem was to make the level of granularity similar inside the query. As the sales quotes are quarterly, the sales from the resellerSales table was grouped by year and quarter as well. In this case a sub-query was not needed.

|  |  |
| --- | --- |
| USE AdventureWorksDW2012;  SELECT [EmployeeKey], [CalendarYear],  [CalendarQuarter], [SalesAmountQuota]  FROM [dbo].[FactSalesQuota]  WHERE [CalendarYear]= 2007  ORDER BY [EmployeeKey]  --This query shows the FactSalesQuota table,  notice the sales rep quotas are by quarter. |  |

The data is from a ‘fact’ table, but this fact table is different. True it holds measures, but there are not many rows, and the data does not have to be condensed for us to use it. We can then treat the SalesAmountQuota from the FactSalesQuota table as a dimension and use it in a GROUP BY() query (check the GROUP BY fields below to verify this). This example is shown to mention that most of the time data explosion occurs when you have > 1 fact table which requires you to use sub-queries, but this case the sub-query was not needed. So you try different queries and then accept the one that works;

If you look at the FactSalesQuota table the values are summary one row for each employee and each quarter. The resellersales fact table has many rows of transaction data that need to be summarized by some dimension. So if the data in one of the fact tables DOES NOT require aggregation then perhaps SQL allows 2 fact tables to JOIN to a dimension tabler.

What is tricky is that we are using a measure field (the quotas) as a measure that can be used in a calculation, but we also treat it as a dimension and include it in our GROUP BY statement. We have to GROUP the sales by year and quarter to produce quarterly sales totals that can be used in a calculation.

|  |  |
| --- | --- |
| USE [AdventureWorksDW2012];  SELECT rs.[EmployeeKey], CONCAT([FirstName], ' ', [LastName]) as Name  , DATEPART(year, [OrderDate]) as [Year]  , DATEPART(quarter, [OrderDate]) as [Quarter]  , COUNT([SalesAmount]) as [SalesTA's]  , [SalesAmountQuota]  , SUM([SalesAmount]) as [Sales Total]  , FORMAT(SUM([SalesAmount]) - [SalesAmountQuota], 'N0') as [perf]  FROM [dbo].[DimEmployee] as e  INNER JOIN [dbo].[FactResellerSales] as rs  ON e.[EmployeeKey] = rs.[EmployeeKey]  INNER JOIN [dbo].[FactSalesQuota] as q  ON e.EmployeeKey = q.EmployeeKey  AND q.CalendarYear = DATEPART(year, rs.[OrderDate])  AND q.[CalendarQuarter] = DATEPART(quarter, rs.[OrderDate])  GROUP BY rs.[EmployeeKey], CONCAT([FirstName], ' ', [LastName]), [SalesAmountQuota]  , DATEPART(year, [OrderDate])  , DATEPART(quarter, [OrderDate])  ORDER BY rs.[EmployeeKey], [Year], [Quarter] | *Sales by quarter by salesrep are calculated by the GROUP BY to match the year/qtr/employeeID granularity – this makes the granularity of the JOINED data and the fact table for sales quotas match. This is the secret sauce of this query. Sales quotas by quarter for all the sales reps for all the years are joined in, and finally the employee table is joined to show the employee name.*  Notice the GROUP BY uses the salesAmountQuotaField. Hmm no sales rep is meeting their targets? What would you do future business leader of America? |

A downfall of subqueries that create calculated measures, is that they are cumbersome to use (too much typing) when performing column comparisons (so far subtracting one column from another or dividing one column by another). What if we wanted to produce a % of total calculation for example to answer the question, “what % of gloves are sold online”?

So the advice is to build the columns of base measures in sub-queries when needed, then performed the rest of the calculations in the report writing software that is easier to use. In the end you will learn many SQL best practices, but never forget you have options. If the final report is made using a report writing software such as PowerBI, then it is easier to perform some functionality there.

Graphical user interface, application, table

Description automatically generated

First we create the three calculated fields, for example:   
YearQtrT = (Query1[Year] \*100) + Query1[Quarter]

Chart, radar chart

Description automatically generated

This is a gauge chart, which are useful to compare actual performance to goal. Two other charting techniques are shown.

Graphical user interface, chart, application

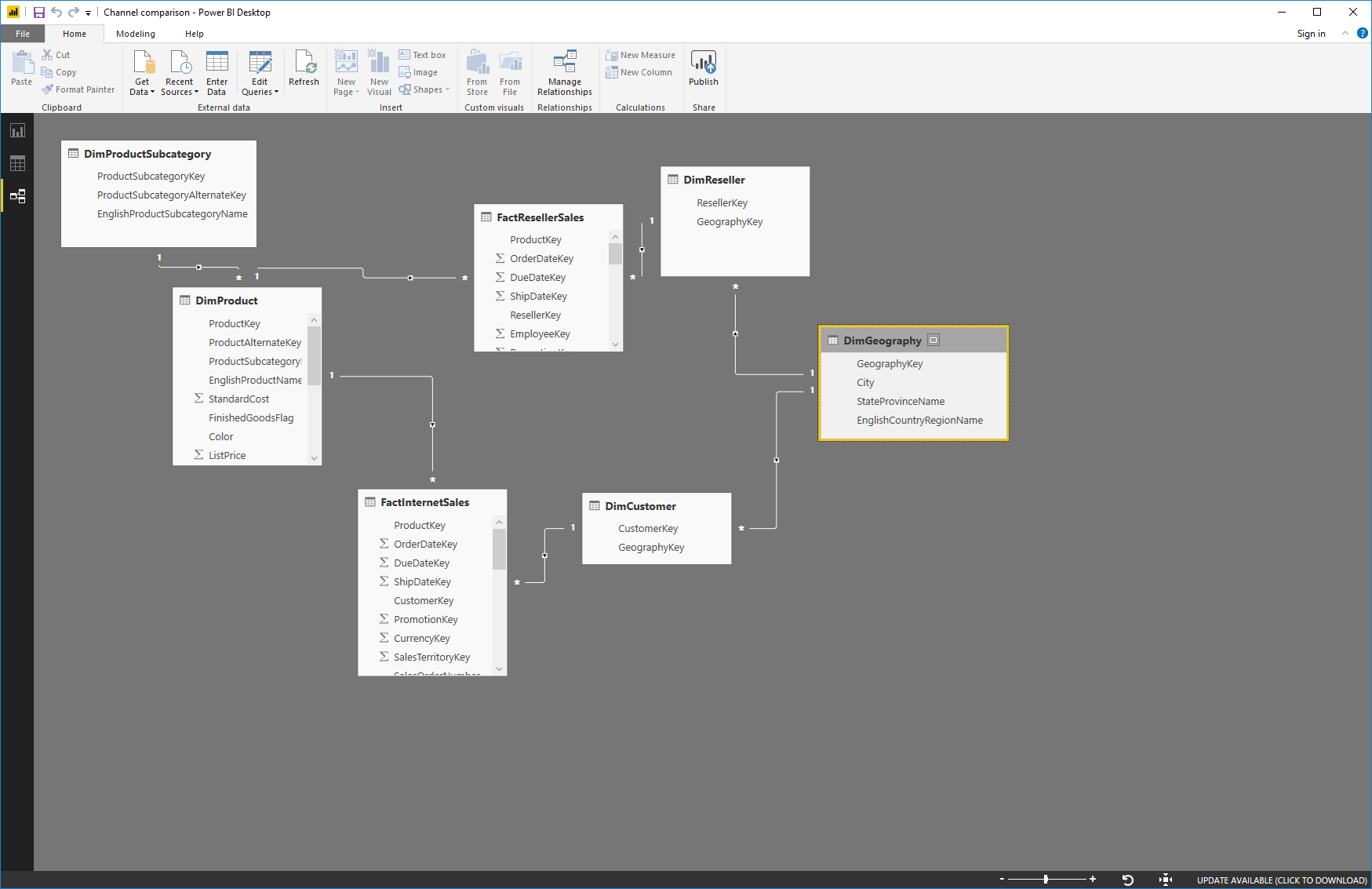
Description automatically generated

Chart, bar chart

Description automatically generated

Similar functionality can be produced in PowerBI using its DAX programming langiage.  
If we pull in the tables and fields into a POWER BI project – then we can make the fields below which leverage   
a) PowerBI’s use of relational modeling specifically with the RELATEDTABLE function and   
b) the advent that you can refer to a calculated column by its name and you do not have to repeat the SQL code. In PowerBI click Modeling tab and select new column and add the table. Next use the code below to create 5 columns. Later we will produce a table and some graphs.

You will be adding columns of calculated measures to a dimension table. While many dimension tables hold just master data such as customer addresses, some dimension tables are hybrid in that they keep running totals or other summary measures. The problem with this approach is that the columns in the dimension table do not get updated automatically, 3 months after you produce your report, another unsuspecting analyst will think the numbers are correct, and get laughed at. PowerBI does have a refresh button, but you would have to know to press it to have the formulas recalculate.



|  |  |
| --- | --- |
| WebUnits = sumx(RELATEDTABLE(FactInternetSales),FactInternetSales[OrderQuantity]) | You can make a new column in the products table that looks at a related table (here the fact table that has the transaction data).  The SUMX function is an iterator, and will loop through each and every row in the specified related table and here sum the values in the specified column (orderquantity).  There are other useful iterators such as COUNTX, |
| ResellerUnits = sumx(RELATEDTABLE(FactResellerSales),FactResellerSales[OrderQuantity]) | Similar to above except a different table is specified. |
| TotalUnits = DimProduct[WebUnits] + DimProduct[ResellerUnits] | So nice to be able to reference a calculated column in another calculation. Thank you PowerBI !! |
| Web%Total = DimProduct[WebUnits]/DimProduct[TotalUnits] | Also PowerBI does not generate any divide by 0 errors when there are no values in the numerator or denominator. |
| Reseller%Total = DimProduct[ResellerUnits]/DimProduct[TotalUnits] | Nice! |

The following image has productID 300 as the first rowGraphical user interface, table

Description automatically generated

Chart, waterfall chart

Description automatically generated

The above chart and table can benefit from a geography slicer.

Well this concludes this special case and analysis.