Advanced Time Intelligence in DAX

Alberto Ferrari
SQLBI
www.sqlbi.com
Alberto Ferrari

• BI Expert and Consultant
• Founder of www.sqlbi.com
  – Problem Solving
  – Complex Project Assistance
  – Data Warehouse Assessments and Development
  – Courses, Trainings and Workshops
• Book Writer
• Microsoft Gold Business Intelligence Partner
• SSAS Maestro – MVP – MCP
What is Time Intelligence?

• Many different topics in one name
  – Year To Date
  – Quarter To Date
  – Running Total
  – Same period previous year
  – Working days computation
  – Fiscal Year

• In short: anything related with time
  – Handled in an intelligent way
Date Table

- Time intelligence needs a date table
  - Built in DAX
  - Or in a SQL Table

- Date table properties
  - All dates should be present
  - From 1° of January, to 31° of December
  - No holes
  - Otherwise time intelligence will not work
Create a Date Table

• Stored in a SQL table inside the DWH
• Optional
  – Create a holidays table
  – Useful for working days calculations
• Set all the necessary attributes
  – Do not create them in DAX
  – SQL is the best place for all attributes
• Building a view is always the best option
Returns a table with a single column named “Date” containing a contiguous set of dates in the given range, inclusive.

```
CALENDAR (  
  DATE ( 2005, 1, 1 ),  
  DATE ( 2015, 12, 31 )  
)

CALENDAR (  
  MIN ( Sales[OrderDate] ),  
  MAX ( Sales[OrderDate] )  
)
```
If you have multiple fact tables, you need to compute the correct values

=CALENDAR ( 
    MIN ( 
        MIN ( Sales[OrderDate] ),
        MIN ( Purchases[PurchaseDate] )
    ),
    MAX ( 
        MAX ( Sales[OrderDate] ),
        MAX ( Purchases[PurchaseDate] )
    )
)
Automatically creates a calendar table, including the dates including the fiscal years (both start and end)

```plaintext
--
-- The parameter is the starting month
-- of the fiscal year
--
= CALENDARAUTO (6)
```

Beware: CALENDARAUTO uses all the dates in your model, excluding only calculated columns and tables.
Mark as Date Table

- Need to mark the calendar as date table
- Set the column containing the date
- Needed to make time intelligence works
- Used by client tools as metadata information
  - Power View
  - Q&A
  - Excel
- Multiple tables can be marked as date table
Set Sorting Options

• Month names do not sort alphabetically
  – April is not the first month of the year
• Use Sort By Column
• Set all sorting options in the proper way
• Beware of sorting granularity
  – 1:1 between names and sort keys
Handling DateTime

• If time is a useful information
• Separate Date from Time
  – Date part → Calendar Table
  – Time part → Time Table
• Reduces distinct values
• Makes analysis much easier
• Time can often be normalized in time ranges of 5, 10 or 30 minutes
Multiple Dates

• Date is often a role dimension
  – Many roles for a date
  – Many date tables

• How many date tables?
  – Try to use only one table
  – Use many, only if needed by the model
  – Many date tables lead to confusion
    • And issues when slicing

• Use proper naming convention
Counting Working Days

• How many working days in a date range?
  – Easily solved with Calendar table
  – Define a new column «WorkingDays»
  – Aggregate with SUM
• Handles any date range
• Works on periods with «holes»
• No separation between fact tables and dimensions in Tabular
Aggregations Over Time

• Many useful aggregations
  – YTD: Year To Date
  – QTD: Quarter To Date
  – MTD: Month To Date

• They all need a Calendar Table

• And some understanding of CALCULATE
Year To Date: the easy way

TOTALYTD: the “DAX for dummies” version

SalesAmountYTD :=

TOTALYTD ( SUM ( Sales[SalesAmount] ), 'Date'[Date] )
Handling Fiscal Year

The last, optional, parameter is the end of the fiscal year
Default: 12-31 (or 31/12 - locale dependent)

SalesAmountYTD :=

TOTALYTD (  
    SUM ( Sales[SalesAmount] ),  
    'Date'[Date],  
    "06-30"
  )
SalesAmountYTD :=

CALCULATE (  
    SUM ( Sales[SalesAmount] ),  
    DATESYTD ( 'Date'[Date] )  
)
Same Period Last Year

Same period in previous year. CALCULATE is needed

Specialized version of DATEADD

Sales_SPLY :=

CALCULATE (  
    SUM ( Sales[SalesAmount] ),  
    SAMEPERIODLASTYEAR ( 'Date'[Date] )  
)
Mixing Time Intelligence Functions

YTD on the previous year. In DAX, it is very simple, just mix the functions to obtain the result

Sales_YTDLY :=

CALCULATE (  
    SUM ( Sales[SalesAmount] ),
    DATESYTD (  
        SAMEPERIODLASTYEAR ( 'Date'[Date] )
    )
)
DATEADD

Similar as SAMEPERIODLASTYEAR, used to calculate different periods: YEAR, MONTH, DAY ...

Does not sum dates, it shifts periods over time

Sales_SPLY :=

CALCULATE ( 
    SUM( Sales[SalesAmount] ),
    DATEADD ( 'Date'[Date], -1, YEAR )
)
PARALLELPERIOD

Returns a set of dates (a table) shifted in time

The whole period is returned, regardless dates in the first parameter

Sales_PPLY :=

CALCULATE (  
    SUM ( Sales[SalesAmount] ),  
    PARALLELPERIOD ( 'Date'[Date] , -1, YEAR  )  
)
Running total, as most of the more complex time intelligence aggregations, needs the CALCULATE version, because there is no syntax sugar here.

SalesAmountRT :=

CALCULATE (  
    SUM ( Sales[SalesAmount] ),  
    FILTER (  
        ALL ( 'Date' ),  
        'Date'[Date] <= MAX ( 'Date'[Date] )  
    ) 
)
Moving Annual Total

Moving window from the current date back one year

```sql
CALCULATE ( 
    SUM ( Sales[SalesAmount] ),
    DATESBETWEEN ( 
        'Date'[Date],
        NEXTDAY ( 
            SAMEPERIODLASTYEAR ( 
                LASTDATE ( 'Date'[Date] )
            )
        )
    ),
    LASTDATE ( 'Date'[Date] )
)
```
There are many week scenarios, depending on what you mean by «week»... CALCULATE is your best friend here

**CALCULATIONS OVER WEEKS**
Custom Calendars

• Time Intelligence functions
  – One day belong to the same quarter every day
  – Not true if you use week calculations
• 4-4-5, 4-5-4, 5-4-4 ?????
  – One quarter is made of three months
  – Two of 4 weeks
  – One of 5 weeks
  – Difference only in position of the 5 weeks month
• No support in DAX for these calendars
Create Week Numbers

- Usually stored in the database
- If not, use Excel to compute them
- Or, as a better solution, use Power Query
  - M code is included in the attendee’s pack!
  - Just copy and paste in Power Query
TOTALYTD is internally transformed into a CALCULATE statement

Cal YTD :=
TOTALYTD (  
    SUM ( Sales[Sales Amount] ),
    Date[Date]
  )
You can compute YTD even if DATESYTD was not available, it is harder but it works the very same way

Cal YTD: :=
IF ( 
    HASONEVALUE ( Date[Year] ), 
    CALCULATE ( 
        SUM ( Sales[Sales Amount] ), 
        FILTER ( 
            ALL ( Date ), 
            Date[Date] <= MAX ( Date[Date] ) 
        ), 
        VALUES ( Date[Year] ) 
    )
)
Week Calculations

• Fine, so how to work with weeks?
• Easier to look at a demo 😊
Semi Additive Measures

• Additive Measure
  – SUM over all dimensions

• Semi Additive Measure
  – SUM over some dimensions
  – Different function over other dimensions
  – Time is the standard exception for aggregations
  – Examples
    • Warehouse stocking
    • Current account balance
Current Account Balance

- Month level correct
- Quarter level wrong
- Year level wrong
Semi additive measures

- Aggregation depends on the filter
  - LastChild over time
  - SUM for the other dimensions

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Luis Bonifaz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maurizio Macagno</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Katie Jordan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Occupation</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Farmer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IT Consultant</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Balance Average</th>
<th>Column Labels</th>
<th>IT Consultant Total</th>
<th>Grand Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Labels</td>
<td></td>
<td>IT Consultant</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Luis Bonifaz</td>
<td>Maurizio Macagno</td>
</tr>
<tr>
<td>Q1/2010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/1/2010</td>
<td>2,450.00</td>
<td>2,500.00</td>
<td>2,475.00</td>
</tr>
<tr>
<td>2/1/2010</td>
<td>1,470.00</td>
<td>1,500.00</td>
<td></td>
</tr>
<tr>
<td>3/1/2010</td>
<td>2,450.00</td>
<td>2,500.00</td>
<td></td>
</tr>
<tr>
<td>Q2/2010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/1/2010</td>
<td>2,025.33</td>
<td>2,066.67</td>
<td>2,046.00</td>
</tr>
<tr>
<td>5/1/2010</td>
<td>1,960.00</td>
<td>2,000.00</td>
<td></td>
</tr>
<tr>
<td>6/1/2010</td>
<td>1,764.00</td>
<td>1,800.00</td>
<td></td>
</tr>
<tr>
<td>Q3/2010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/1/2010</td>
<td>3,332.00</td>
<td>3,366.00</td>
<td>3,366.00</td>
</tr>
<tr>
<td>8/1/2010</td>
<td>3,136.00</td>
<td>3,200.00</td>
<td></td>
</tr>
<tr>
<td>9/1/2010</td>
<td>4,410.00</td>
<td>4,500.00</td>
<td></td>
</tr>
<tr>
<td>Q4/2010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/1/2010</td>
<td>1,992.67</td>
<td>2,033.33</td>
<td>2,013.00</td>
</tr>
<tr>
<td>11/1/2010</td>
<td>1,960.00</td>
<td>2,000.00</td>
<td></td>
</tr>
<tr>
<td>12/1/2010</td>
<td>1,813.00</td>
<td>1,850.00</td>
<td></td>
</tr>
<tr>
<td>Grand Total</td>
<td>2,450.00</td>
<td>2,500.00</td>
<td>2,475.00</td>
</tr>
</tbody>
</table>
SemiAdditive Measures

CALCULATE: to set the filter
LASTDATE: to find the last child

LastBalance :=

CALCULATE (  
    SUM ( Balances[Balance] ),  
    LASTDATE ( Date[Date] )  
)
Last Non Blank

Searches in the fact table for the last non empty date. Remember: LASTNONBLANK is an iterator

```
LastBalanceNonBlank :=

CALCULATE (  
  SUM ( Balances[Balance] ),
  LASTNONBLANK (  
    BalanceDate[Date],  
    CALCULATE ( COUNTROWS ( Balances ) )  
  )
)
```
Opening and Closing Balance

ClosingBalanceMonth =
CLOSINGBALANCEMONTH ( SUM ( Balances[Balance] ), BalanceDate[Date] )

ClosingBalanceQuarter =
CLOSINGBALANCEQUARTER ( SUM ( Balances[Balance] ), BalanceDate[Date] )

ClosingBalanceYear =
CLOSINGBALANCEYEAR ( SUM ( Balances[Balance] ), BalanceDate[Date] )

<table>
<thead>
<tr>
<th>Row Labels</th>
<th>LastBalance</th>
<th>ClosingBalanceMonth</th>
<th>ClosingBalanceQuarter</th>
<th>ClosingBalanceYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>6,686.00</td>
<td>6,686.00</td>
<td>6,686.00</td>
<td>6,686.00</td>
</tr>
<tr>
<td>Q1</td>
<td>10,667.00</td>
<td>10,667.00</td>
<td>10,667.00</td>
<td>10,667.00</td>
</tr>
<tr>
<td>01 - January</td>
<td>4,657.00</td>
<td>4,657.00</td>
<td>4,657.00</td>
<td>4,657.00</td>
</tr>
<tr>
<td>02 - February</td>
<td>7,762.00</td>
<td>7,762.00</td>
<td>7,762.00</td>
<td>7,762.00</td>
</tr>
<tr>
<td>03 - March</td>
<td>10,667.00</td>
<td>10,667.00</td>
<td>10,667.00</td>
<td>10,667.00</td>
</tr>
<tr>
<td>Q2</td>
<td>7,452.00</td>
<td>7,452.00</td>
<td>7,452.00</td>
<td>7,452.00</td>
</tr>
<tr>
<td>04 - April</td>
<td>6,210.00</td>
<td>6,210.00</td>
<td>6,210.00</td>
<td>6,210.00</td>
</tr>
<tr>
<td>05 - May</td>
<td>5,589.00</td>
<td>5,589.00</td>
<td>5,589.00</td>
<td>5,589.00</td>
</tr>
<tr>
<td>06 - June</td>
<td>7,452.00</td>
<td>7,452.00</td>
<td>7,452.00</td>
<td>7,452.00</td>
</tr>
<tr>
<td>Q3</td>
<td>7,762.00</td>
<td>7,762.00</td>
<td>7,762.00</td>
<td>7,762.00</td>
</tr>
<tr>
<td>07 - July</td>
<td>9,936.00</td>
<td>9,936.00</td>
<td>9,936.00</td>
<td>9,936.00</td>
</tr>
<tr>
<td>08 - August</td>
<td>13,972.00</td>
<td>13,972.00</td>
<td>13,972.00</td>
<td>13,972.00</td>
</tr>
<tr>
<td>09 - September</td>
<td>7,762.00</td>
<td>7,762.00</td>
<td>7,762.00</td>
<td>7,762.00</td>
</tr>
<tr>
<td>Q4</td>
<td>6,686.00</td>
<td>6,686.00</td>
<td>6,686.00</td>
<td>6,686.00</td>
</tr>
<tr>
<td>10 - October</td>
<td>6,210.00</td>
<td>6,210.00</td>
<td>6,210.00</td>
<td>6,210.00</td>
</tr>
<tr>
<td>11 - November</td>
<td>5,744.00</td>
<td>5,744.00</td>
<td>5,744.00</td>
<td>5,744.00</td>
</tr>
<tr>
<td>12 - December</td>
<td>6,686.00</td>
<td>6,686.00</td>
<td>6,686.00</td>
<td>6,686.00</td>
</tr>
</tbody>
</table>

Grand Total 6,686.00 6,686.00 6,686.00 6,686.00
Opening and Closing Balance

<table>
<thead>
<tr>
<th>DAX Function</th>
<th>Equivalent date filter in CALCULATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPENINGBALANCEMONTH</td>
<td>STARTOFMONTH</td>
</tr>
<tr>
<td>OPENINGBALANCEQUARTER</td>
<td>STARTOFQUARTER</td>
</tr>
<tr>
<td>OPENINGBALANCEYEAR</td>
<td>STARTOFYEAR</td>
</tr>
<tr>
<td>CLOSINGBALANCEMONTH</td>
<td>ENDOFMONTH</td>
</tr>
<tr>
<td>CLOSINGBALANCEQUARTER</td>
<td>ENDOFQUARTER</td>
</tr>
<tr>
<td>CLOSINGBALANCEYEAR</td>
<td>ENDOFYEAR</td>
</tr>
</tbody>
</table>
Time Intelligence: Conclusions

• Based on evaluation contexts
  – Replace filter on date
  – Many predefined functions
  – You can author your own functions
• Basic Time Intelligence
• Creating more complex aggregations
• Working with ISO weeks