DAX as a Query Language

Matt Allington
exceleratorbi.com.au
About me

• 25 year career at Coca-Cola working in both Sales and Information Technology
• Now running a Power Pivot consultancy in Sydney Australia
  • Self Service BI Consulting
  • Power Pivot/Power BI Training
  • Blogger http://xbi.com.au/blog
• Author
• Microsoft MVP

@ExceleratorBI
https://au.linkedin.com/in/mattallington
http://xbi.com.au
Who is in the Audience?

- Who is a SQL Professional or from IT?
- Who is an Excel/Power BI Professional/User?
- What is your DAX Skill?
  - Zero, Intermediate, Middle to Pro?
Agenda

• Introduction to DAX as a Query Language
• Use cases for DAX Queries
• Demos
• More demos

Please ask questions as we go along
Most Common Usage for DAX – As Formulas

• Short for Data Analysis eXpressions
• Great for business/Excel users (Functional Language, similar to Excel)
• Always returns a scalar value, e.g.
  • A number
  • Text
  • Logical TRUE/FALSE

Total Sales = SUMX(Sales, Sales[Qty] * Sales[Unit Price])

Total Sales of Bikes = CALCULATE([Total Sales],
  FILTER(ALL(Products[Category]),
    Products[Category] = “Bikes”
  )
)
DAX as a Query Language

- DAX Queries use the same DAX language
  - Mostly functions are in common with standard DAX
  - Some special “query only” commands (like EVALUATE).
  - Some less common functions like
    - SUMMARIZE
    - ADDCOLUMNS
- The query sends a request for data to a database
- The result is always a table of data
- Queries return 0, 1 or more rows of data
Use Cases for DAX Queries

- Extracting underlying data into a table
  - Detailed
  - Summarised

- Learning to write better DAX measures
  - Debugging formulas
  - Building virtual tables for efficient iteration

- Checking performance of measures (using DAX Studio)
Demo: Intro to DAX Studio and DAX Queries
Key Points: Intro to DAX Studio and DAX Queries

• DAX Studio can connect to different Tabular sources.
• Every query must start with the EVALUATE statement.
• Can use any DAX function that returns a table.
  • FILTER()
  • VALUES()
  • DISTINCT()
  • ALL()
  • CALCULATETABLE()
  • etc
First let’s look at the two Vertipaq engines.
Storage Engine (SE)

- Really Fast
- Compresses data on load
- Can retrieve records without decompressing in the right conditions
- Multi threaded
- Cached

SE is the secret to small and fast workbooks

Formula Engine (FE)

- Really Smart
- Has almost unlimited calculation capability
- Iterates in a row context and hence it can be slow
- Single threaded
- Not cached

FE makes anything possible, but it can come at a cost
Demo: Summarising Tables
Key Points: Summarising Data

• SUMMARIZE function
  • Is a Vertipaq Storage Engine function – very efficient.
  • When using more than 1 table, always start with the data table.
  • Add extra columns from data or any connected lookup table.
  • Can add summarised value columns, but it is inefficient at this task.

• ADDCOLUMNS
  • Use in preference to using SUMMARIZE to add value columns.
  • Works in a row context, so you must force context transition.

• SUMMARIZECOLUMNS
  • Can use this instead of ADDCOLUMNS(SUMMARIZE()).
  • No need to specify the starting table.
  • Most efficient approach.
Demo: Testing Measure Efficiency

Total Sales Value = SUMX(Sales,[Unit Price] * [Total Sales Qty])
Key Points: Testing Measure Efficiency

- Must use the ROW function to convert a measure to a table.

- Can use DEFINE function to override (or create) measures local to the query.

- Clear the cache between tests.
Demo: Debugging filters inside CALCULATE()
Summary: Debugging filters inside CALCULATE()

- DAX queries let you “see” the returned table portion of a measure/calculated column.
- Can also use New Table in Power BI.
- Sort the table out first, then get back to the measure.
Demo: Extracting data to Excel
Key Point: Extracting Data to Excel

- You must launch DAX Studio from within Excel to be able to export directly back to Excel.
Same Stores Growth vs LY
### Growth in Same Sales vs LY

<table>
<thead>
<tr>
<th>Row Labels</th>
<th>Store 1</th>
<th>Store 2</th>
<th>Store 3</th>
<th>Store 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Sales</td>
<td>Total Sales</td>
<td>Total Sales</td>
<td>Total Sales</td>
</tr>
<tr>
<td>2014</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$255</td>
<td>$221</td>
<td>$188</td>
<td>$276</td>
</tr>
<tr>
<td>2</td>
<td>$196</td>
<td>$240</td>
<td>$242</td>
<td>$225</td>
</tr>
<tr>
<td>3</td>
<td>$223</td>
<td>$248</td>
<td>$222</td>
<td>$291</td>
</tr>
<tr>
<td>4</td>
<td>$221</td>
<td>$248</td>
<td>$254</td>
<td>$254</td>
</tr>
<tr>
<td>5</td>
<td>$245</td>
<td>$294</td>
<td>$242</td>
<td>$254</td>
</tr>
<tr>
<td>6</td>
<td>$253</td>
<td>$259</td>
<td>$259</td>
<td>$245</td>
</tr>
<tr>
<td>7</td>
<td>$302</td>
<td>$294</td>
<td>$259</td>
<td>$254</td>
</tr>
<tr>
<td>8</td>
<td>$274</td>
<td>$294</td>
<td>$259</td>
<td>$254</td>
</tr>
<tr>
<td>9</td>
<td>$265</td>
<td>$259</td>
<td>$259</td>
<td>$254</td>
</tr>
<tr>
<td>10</td>
<td>$303</td>
<td>$259</td>
<td>$259</td>
<td>$254</td>
</tr>
<tr>
<td>11</td>
<td>$248</td>
<td>$259</td>
<td>$255</td>
<td>$254</td>
</tr>
<tr>
<td>12</td>
<td>$242</td>
<td>$255</td>
<td>$255</td>
<td>$254</td>
</tr>
<tr>
<td>2014 Total</td>
<td>$4,009</td>
<td>$2,337</td>
<td>$1,667</td>
<td>$1,291</td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$255</td>
<td>$228</td>
<td>$257</td>
<td>$265</td>
</tr>
<tr>
<td>2</td>
<td>$242</td>
<td>$222</td>
<td>$234</td>
<td>$208</td>
</tr>
<tr>
<td>3</td>
<td>$265</td>
<td>$223</td>
<td>$294</td>
<td>$159</td>
</tr>
<tr>
<td>4</td>
<td>$269</td>
<td>$241</td>
<td>$278</td>
<td>$306</td>
</tr>
<tr>
<td>5</td>
<td>$329</td>
<td>$248</td>
<td>$398</td>
<td>$298</td>
</tr>
<tr>
<td>6</td>
<td>$301</td>
<td>$270</td>
<td>$388</td>
<td>$223</td>
</tr>
<tr>
<td>7</td>
<td>$295</td>
<td>$298</td>
<td>$388</td>
<td>$223</td>
</tr>
<tr>
<td>8</td>
<td>$276</td>
<td>$295</td>
<td>$295</td>
<td>$278</td>
</tr>
<tr>
<td>9</td>
<td>$322</td>
<td>$282</td>
<td>$285</td>
<td>$222</td>
</tr>
<tr>
<td>10</td>
<td>$254</td>
<td>$274</td>
<td>$319</td>
<td>$242</td>
</tr>
<tr>
<td>11</td>
<td>$260</td>
<td>$294</td>
<td>$298</td>
<td>$259</td>
</tr>
<tr>
<td>12</td>
<td>$242</td>
<td>$295</td>
<td>$260</td>
<td>$245</td>
</tr>
<tr>
<td>2015 Total</td>
<td>$3,078</td>
<td>$2,678</td>
<td>$3,387</td>
<td>$3,236</td>
</tr>
</tbody>
</table>
Same Sales Growth vs LY

```powerbi
Same Store Sales LY Sumx =
SUMX (stores,
    SUMX (calendar,
        IF ([Total Sales] > 0 && [Total Sales LY] > 0, [Total Sales LY] )
    )
)

Same Store Sales TY Sumx =
SUMX (stores,
    SUMX (calendar,
        IF ([Total Sales] > 0 && [Total Sales LY] > 0, [Total Sales] )
    )
)
```

Excelerator BI
Power BI Solutions for Business
Demo

Same Stores Growth vs LY
Key Points: Growth in Same Sales vs LY

- Can use DAX Queries to create efficient summary tables
- Tables will retain lineage to the data model
- Then iterate efficiently over the summary table

```plaintext
= SUMX ( FILTER ( 
    ADDCOLUMNS ( 
        SUMMARIZE ( sales, 'Calendar'[YYMM], Stores[Store ID] ),
        "Total Sales TY", [Total Sales],
        "Total Sales LY", [Total Sales LY]
    ),
    [Total Sales TY] > 0
    && [Total Sales LY] > 0
), [Total Sales] )
```
Wrap Up and Questions

Questions?

Online Feedback

googl/SrvpmJ

• More Information on DAX Queries

Power Query Training

DAX Training