

Power BI Deep Dive

Remigiusz Suszkiewicz (B.A.)

ppedv AG

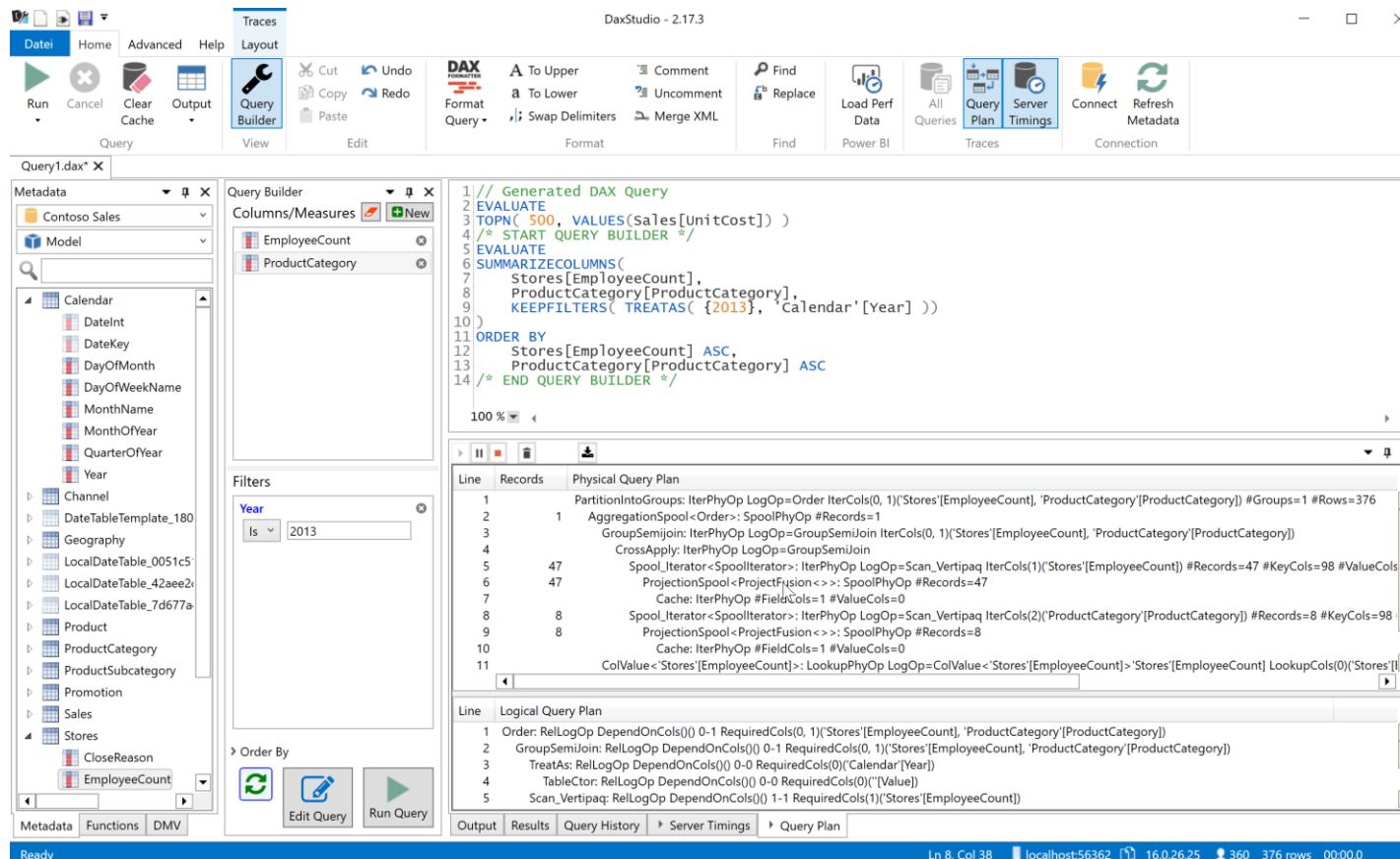
Datum: 01.06.2022

Power BI DAX - Was steckt dahinter?

- DAX wird zum Schreiben von Formeln verwendet
- Jede DAX-Formel ist Teil einer DAX-Abfrage, die von zwei Engines verarbeitet wird:
 - Formel-Engine
 - Speicher-Engine
- Power BI sendet die DAX-Abfrage an die Analysis Services-Engine
 - lokal für Power BI Desktop oder ein externer Dienst sein
 - lokal (SQL Server Analysis Services) oder in der Cloud (Azure Analysis Services)
- Analysis Services empfängt die Abfrage
 - Abfrageplan generiert und ausgeführt von der Formel-Engine und der Speicher-Engine.
- Abfrage mit DAX Studio analysieren
 - Der Abfrageplan ist eine dichte Liste von Vorgängen.

Analyse Tool DaxStudio

- DaxStudio – kostenfreies lizenzfreies Tool für DAX Performance Analysen



The screenshot displays the DaxStudio 2.17.3 application window. The interface includes a menu bar (Datei, Home, Advanced, Help), a ribbon with various tool groups (Query, View, Edit, Format, Find, Power BI, Traces, Connection), and a main workspace divided into several panes.

Query Builder: Shows the 'Columns/Measures' pane with 'EmployeeCount' and 'ProductCategory' selected. The 'Filters' pane shows 'Year' with a value of '2013'. The 'Order By' pane is empty.

Generated DAX Query:

```
1 // Generated DAX Query
2 EVALUATE
3 TOPN( 500, VALUES(Sales[UnitCost]) )
4 /* START QUERY BUILDER */
5 EVALUATE
6 SUMMARIZECOLUMNS(
7     Stores[EmployeeCount],
8     ProductCategory[ProductCategory],
9     KEEPFILTERS( TREATAS( {2013}, 'Calendar'[Year] ) )
10 )
11 ORDER BY
12     Stores[EmployeeCount] ASC,
13     ProductCategory[ProductCategory] ASC
14 /* END QUERY BUILDER */
```

Physical Query Plan:

Line	Records	Physical Query Plan
1		PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1)('Stores'[EmployeeCount], 'ProductCategory'[ProductCategory]) #Groups=1 #Rows=376
2	1	AggregationSpool<Order>: SpoolPhyOp #Records=1
3		GroupSemiJoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1)('Stores'[EmployeeCount], 'ProductCategory'[ProductCategory])
4		CrossApply: IterPhyOp LogOp=GroupSemiJoin
5	47	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=Scan_Vertipaq IterCols(1)('Stores'[EmployeeCount]) #Records=47 #KeyCols=98 #ValueCols=1
6	47	ProjectionSpool<ProjectFusion<>>: SpoolPhyOp #Records=47
7		Cache: IterPhyOp #FieldCols=1 #ValueCols=0
8	8	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=Scan_Vertipaq IterCols(2)('ProductCategory'[ProductCategory]) #Records=8 #KeyCols=98
9	8	ProjectionSpool<ProjectFusion<>>: SpoolPhyOp #Records=8
10		Cache: IterPhyOp #FieldCols=1 #ValueCols=0
11		ColValue<'Stores'[EmployeeCount]>: LookupPhyOp LogOp=ColValue<'Stores'[EmployeeCount]>'Stores'[EmployeeCount] LookupCols(0)('Stores'[EmployeeCount])

Logical Query Plan:

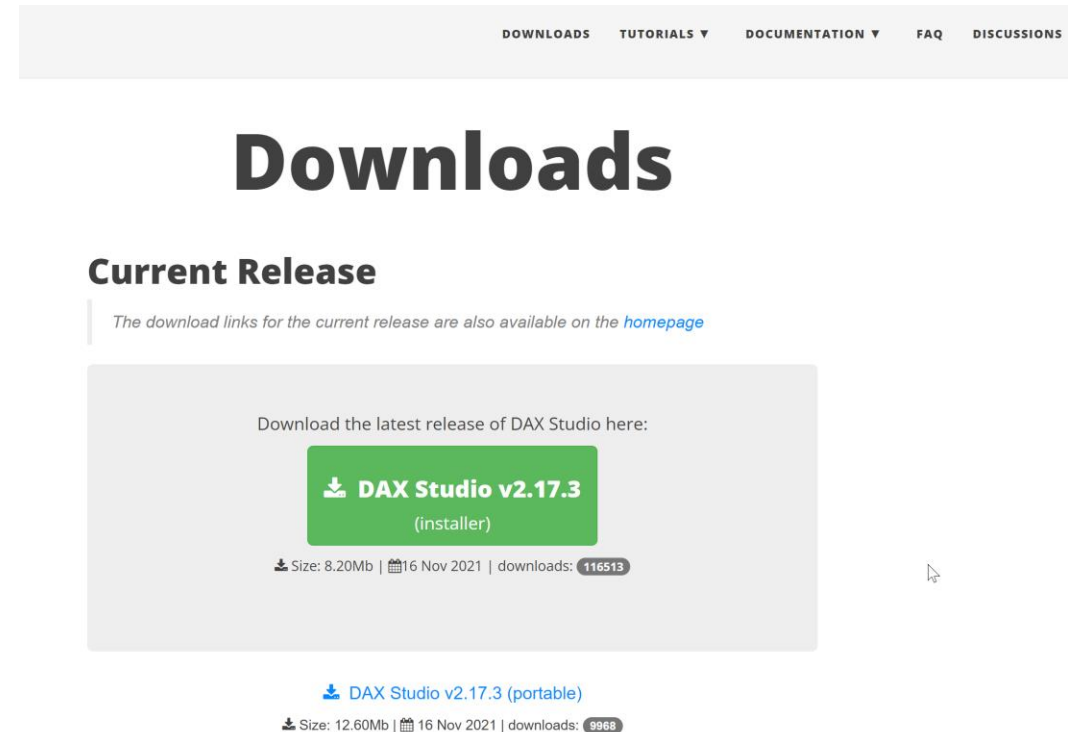
Line	Logical Query Plan
1	Order: RelLogOp DependOnCols(0, 1)('Stores'[EmployeeCount], 'ProductCategory'[ProductCategory])
2	GroupSemiJoin: RelLogOp DependOnCols(0, 1)('Stores'[EmployeeCount], 'ProductCategory'[ProductCategory])
3	TreatAs: RelLogOp DependOnCols(0, 0-0 RequiredCols(0)('Calendar'[Year]))
4	TableCtor: RelLogOp DependOnCols(0, 0-0 RequiredCols(0)('Value'))
5	Scan_Vertipaq: RelLogOp DependOnCols(0, 1-1 RequiredCols(1)('Stores'[EmployeeCount]))

The status bar at the bottom indicates 'Ready', 'Ln 8, Col 38', 'localhost:56362', '16.0.26.25', '360', '376 rows', and '00:00.0'.

DaxStudio Download und Installation

- <https://daxstudio.org/downloads/>
- Weitere von Microsoft empfohlene Tools:

<https://docs.microsoft.com/de-de/power-bi/transform-model/desktop-external-tools>



The screenshot shows the 'Downloads' section of the DAX Studio website. At the top, there is a navigation bar with links for 'DOWNLOADS', 'TUTORIALS', 'DOCUMENTATION', 'FAQ', and 'DISCUSSIONS'. The main heading is 'Downloads'. Below it, the 'Current Release' section is highlighted. A note states: 'The download links for the current release are also available on the homepage'. The main content area features a large green button labeled 'DAX Studio v2.17.3 (installer)'. Below this button, it specifies 'Size: 8.20Mb | 16 Nov 2021 | downloads: 116513'. At the bottom, there is a link for 'DAX Studio v2.17.3 (portable)' with details 'Size: 12.60Mb | 16 Nov 2021 | downloads: 9968'.

DOWNLOADS TUTORIALS ▼ DOCUMENTATION ▼ FAQ DISCUSSIONS

Downloads

Current Release

The download links for the current release are also available on the [homepage](#)

Download the latest release of DAX Studio here:

DAX Studio v2.17.3
(installer)

Size: 8.20Mb | 16 Nov 2021 | downloads: 116513

[DAX Studio v2.17.3 \(portable\)](#)

Size: 12.60Mb | 16 Nov 2021 | downloads: 9968

Arbeitsweisen der Engines (Formel und Speicher)

- Im Abfrageplan verarbeitet jeder Knoten das Ergebnis eines oder mehrerer untergeordneter Knoten
- Jeder Knoten entspricht einer Operation, die von der Formel-Engine ausgeführt wird
- Knoten der die Anfrage an die Speicher-Engine übergibt
- Die Formel-Engine kann Daten verarbeiten, aber keine Daten aus den Tabellen im Datenmodell abrufen
- Die Speicher-Engine ist für den Datenabruf zuständig.

In-Memory Engine VertiPaq

- Die In-Memory-Speicher-Engine heißt VertiPaq.
- Beim Import wird der Inhalt der Tabelle nur einmal aus der Datenquelle gelesen und komprimiert im Arbeitsspeicher gespeichert. Die Abfragen an die VertiPaq-Speicher-Engine werden in einem xSQL Format beschrieben
- Dies ist die einzelne xSQL-Abfrage, die vom vorherigen Abfrageplan ausgeführt wird:

```

13 EVALUATE
14 SUMMARIZECOLUMNS (
15     ProductCategory[ProductCategory],
16     'Product'[ProductName],
17     'Product'[UnitPrice],
18     KEEPFILTERS ( TREATAS ( { 2013 }, 'Calendar'[Year] ) ),
19     "Sales YTD", [Sales YTD]
20 )
21 ORDER BY
22     ProductCategory[ProductCategory] ASC,
23     'Product'[ProductName] ASC,
24     'Product'[UnitPrice] ASC
25 /* END QUERY BUILDER */

```

Total 53 ms
SE CPU 78 ms

FE 23 ms
43,4%
SE 30 ms
56,6%

SE Queries 3
SE Cache 0
0,0%

Line	Subclass	Duration	CPU	Par.	Rows	KB	Query
2	Scan	1	0		5.116	40	SELECT 'Calendar'[DateKey] FROM
4	Scan	0	0		5.116	40	SELECT 'Calendar'[DateKey] FROM
6	Scan	29	78	x2,7	1.497	24	SELECT 'ProductCategory'[Produc

Abfrage in DaxStudio analysieren

- Abfrage mittels Query Builder und selbst erstelltes Measure ausführen und analysieren

Query Builder

Columns/Measures

- ProductCategory
- ProductName
- UnitPrice
- Sales YTD

Filters

Year

Is 2013

Order By

Logical Query Plan

```

12 EVALUATE
13 SUMMARIZECOLUMNS (
14     ProductCategory[ProductCategory],
15     'Product'[ProductName],
16     'Product'[UnitPrice],
17     KEEPFILTERS ( TREATAS ( { 2013 }, 'calendar'[Year] ) ),
18     "Sales YTD", [Sales YTD]
19 )
20 )
21 ORDER BY
22     ProductCategory[ProductCategory] ASC,
23     'Product'[ProductName] ASC,
24     'Product'[UnitPrice] ASC
25 /* END QUERY BUILDER */

```

Physical Query Plan

Line	Records	Physical Query Plan
1		PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1, 2, 3)(ProductCategory[ProductCategory], 'Product'[ProductName], 'Product'[UnitPrice], "[Sales YTD]"
2	1	AggregationSpool<Order>: SpoolPhyOp #Records=1
3		GroupSemijoin: IterPhyOp LogOp=GroupSemijoin IterCols(0, 1, 2, 3)(ProductCategory[ProductCategory], 'Product'[ProductName], 'Product'[UnitPrice], "[Sales YTD]"
4	1.497	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=Sum_Vertipaq IterCols(1, 2, 3)(ProductCategory[ProductCategory], 'Product'[ProductName], 'Product'[UnitPrice], "[Sales YTD]"
5	1.497	ProjectionSpool<ProjectFusion<Copy>>: SpoolPhyOp #Records=1497
6		Cache: IterPhyOp #FieldCols=3 #ValueCols=1
7	366	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=DateAdd<Year> IterCols(4)(Calendar[DateKey]) #Records=366 #KeyCols=1 #ValueCols=1
8	366	AggregationSpool<GroupBy>: SpoolPhyOp #Records=366
9		DateAdd<Year>: IterPhyOp LogOp=DateAdd<Year> IterCols(4)(Calendar[DateKey])
10		SingletonTable: IterPhyOp LogOp=TableCtor IterCols(0)(Calendar[DateKey])
11	365	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=Scan_Vertipaq IterCols(0)(Calendar[DateKey])
12	365	ProjectionSpool<ProjectFusion<>>: SpoolPhyOp #Records=365
13		Cache: IterPhyOp #FieldCols=1 #ValueCols=0
14	1	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=TreatAs IterCols(0)(Calendar[Year])
15	1	AggregationSpool<GroupBy>: SpoolPhyOp #Records=1
16		TreatAs: IterPhyOp LogOp=TreatAs IterCols(0)(Calendar[Year])
17		TableCtor: IterPhyOp LogOp=TableCtor IterCols(0)(Calendar[Year])

SE CPU

SE

FE

SE Queries 3 SE Cache 0 0,0%

Line	Subclass	Duration	CPU	Par.	Rows	KB	Query
2	Scan	0	0		5.116	40	SELECT 'Calendar'[DateKey] FROM
4	Scan	0	0		5.116	40	SELECT 'Calendar'[DateKey] FROM
6	Scan	36	109	x3,0	1.497	24	SELECT 'ProductCategory'[ProductCategory]

Unterschiede der beiden Engines

- Die **Formel-Engine** verarbeitet die Anfrage, generiert und führt einen Abfrageplan aus
- Die **Speicher-Engine** ruft Daten aus dem tabellarischen Modell ab, um die Anforderungen der Formel-Engine zu beantworten.
Die Speicher-Engine hat zwei Implementierungen:
 - **VertiPaq** hostet eine Kopie der Daten im Arbeitsspeicher, die regelmäßig von der Datenquelle aktualisiert wird.
 - **DirectQuery** leitet Anfragen für jede Anfrage direkt an die ursprüngliche Datenquelle weiter.
- Die Formel-Engine ist die übergeordnete Ausführungseinheit der Abfrage-Engine. Es kann alle von DAX angeforderten Operationen verarbeiten.