OVERNET.

WPC 2023

Faster & Better Power BI

Marco Englaro

Cloud Solution Architect

Microsfot

Who I am

Cloud Solution Architect Data & AI Microsoft

Co Founder

Wallabies FANTAGOAT wallawin ARTIFICIAL INTELLIGENCE FOR BETTING

Marco Englaro

@MarcoEnglaro Marco Englaro | LinkedIn

Improve speed performance of Power BI

Power BI

OVERNET.

Performance Issues

Power BI today It's very slow
The update of dataset lasted more than 4 hours
This report it opens too much time
I filtered the nation and I am waiting

Right Approach

Environment and Architecture

Power BI Capacity

Semantic Model

Connection Type

Data Model

Report

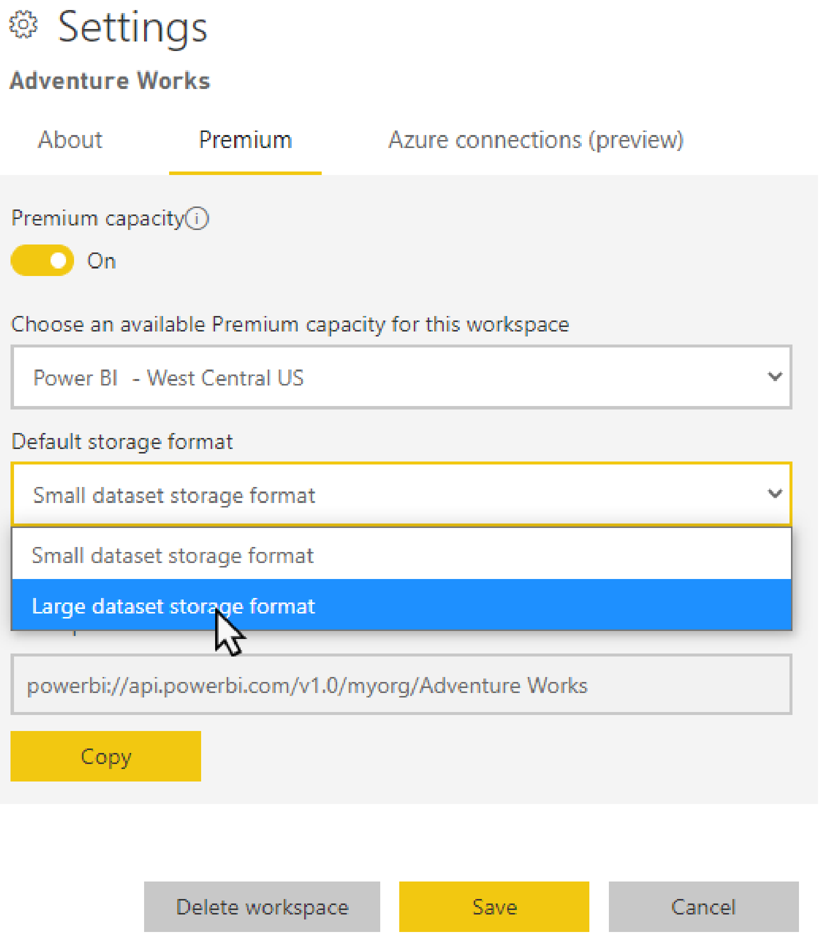
Dax Query

Report Interaction

Environment and Architecture

- Power BI Capacity - Large Dataset

Large semantic model storage format setting has benefits.
When enabled, the large semantic model storage format can improve XMLA write operations performance.
For semantic models using the large semantic model storage format, Power BI automatically sets the default segment size to 8 million rows to strike a good balance between memory requirements and query performance for large tables.



Settings

Adventure Works

Premium capacity: On

Choose an available Premium capacity for this workspace: Power BI - West Central US

Default storage format: Small dataset storage format

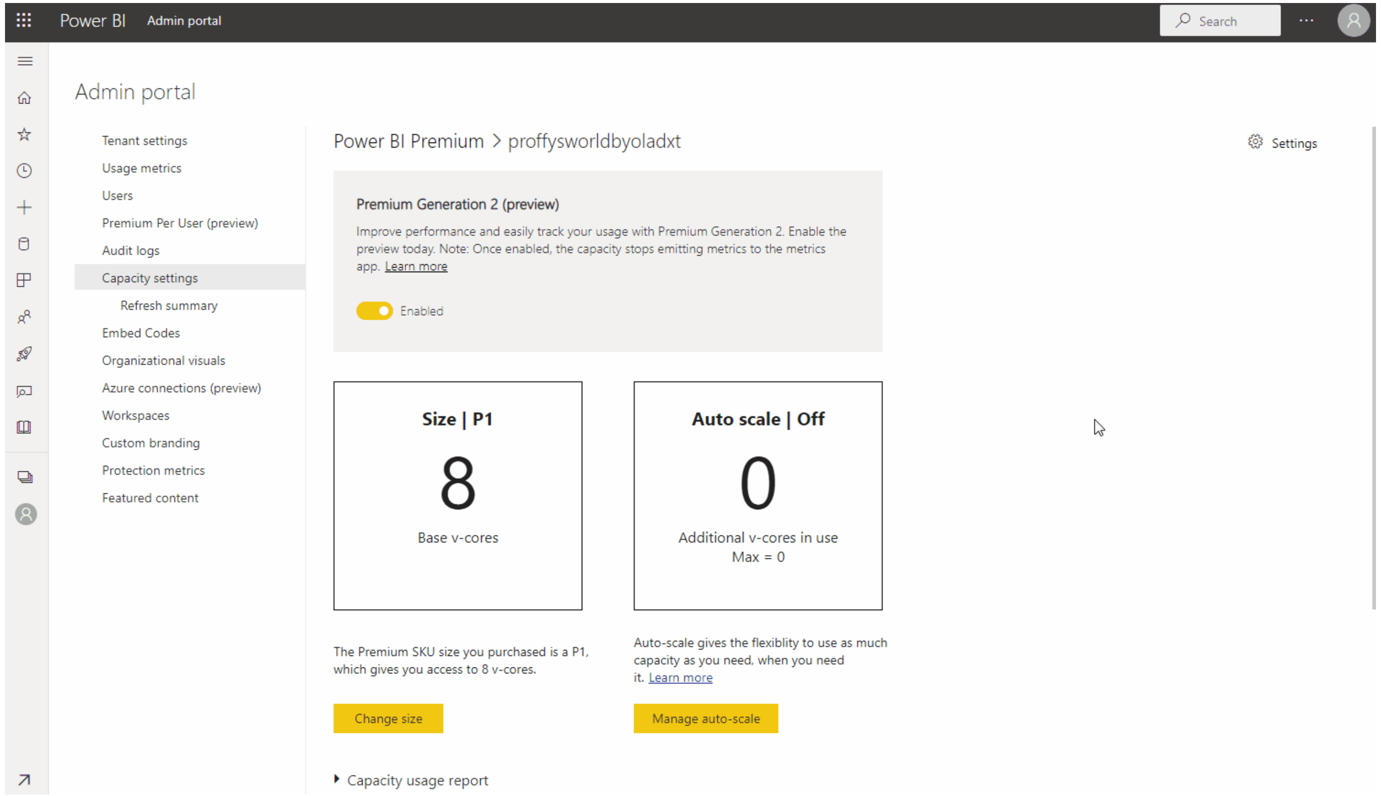
powerbi://api.powerbi.com/v1.0/myorg/Adventure Works

About
Premium
Azure connections (preview)
Small dataset storage format
Large dataset storage format
Copy
Delete workspace
Save
Cancel

Environment and Architecture

- Power BI Capacity - Autoscale

Automatically use more v-cores (virtual CPU cores) when the computing load on your Power BI Premium subscription would otherwise be slowed by its capacity.



Power BI Premium > proffysworldbyoladxt

Premium Generation 2 (preview)

Improve performance and easily track your usage with Premium Generation 2. Enable the preview today. Note: Once enabled, the capacity stops emitting metrics to the metrics app. Learn more

Enabled

Size | P1

8

Base v-cores

Auto scale | Off

0

Additional v-cores in use

Max = 0

The Premium SKU size you purchased is a P1, which gives you access to 8 v-cores.

Auto-scale gives the flexiblity to use as much capacity as you need, when you need it. Learn more

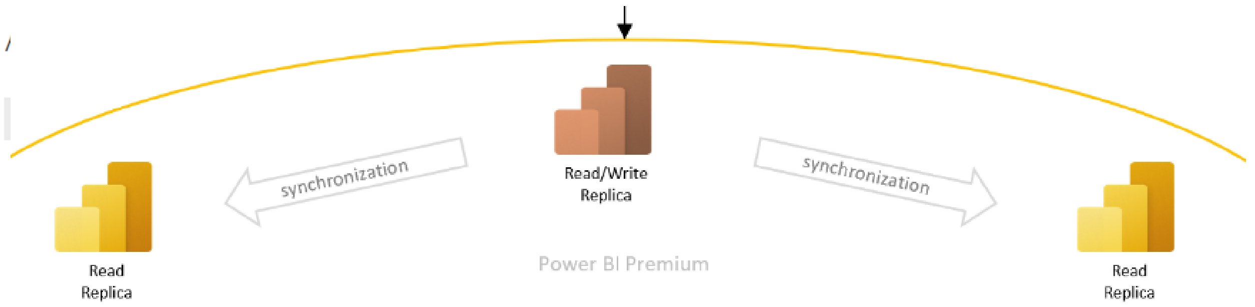
OVERNET.

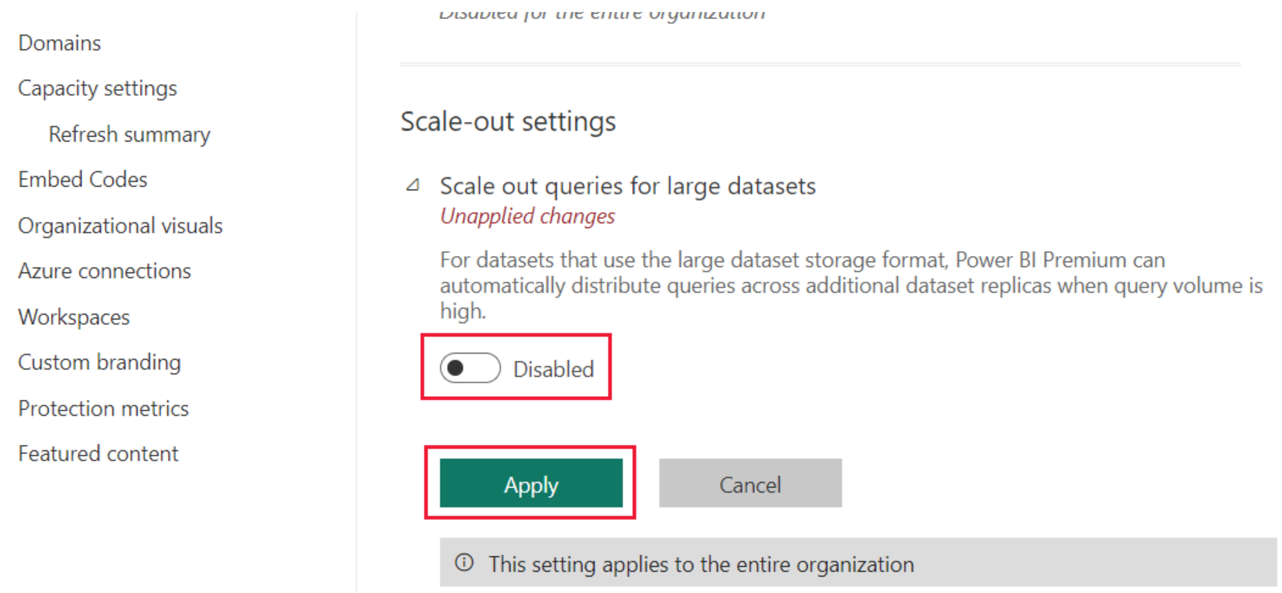
Environment and Architecture

- Power BI Capacity - Scale Out

Semantic model scale-out helps Power BI deliver fast performance while your reports and dashboards are consumed by a large audience.
Semantic model scale-out uses your Premium capacity to host one or more read-only replicas of your primary semantic model.

By increasing throughput, the read-only replicas ensure performance doesn't slow down when multiple users submit queries at the same time.




Disabled for the entire organization

Domains
Capacity settings
Refresh summary
Embed Codes
Organizational visuals
Azure connections
Workspaces
Custom branding
Protection metrics
Featured content

Scale-out settings

◿ Scale out queries for large datasets
Unapplied changes

For datasets that use the large dataset storage format, Power BI Premium can automatically distribute queries across additional dataset replicas when query volume is high.

● Disabled

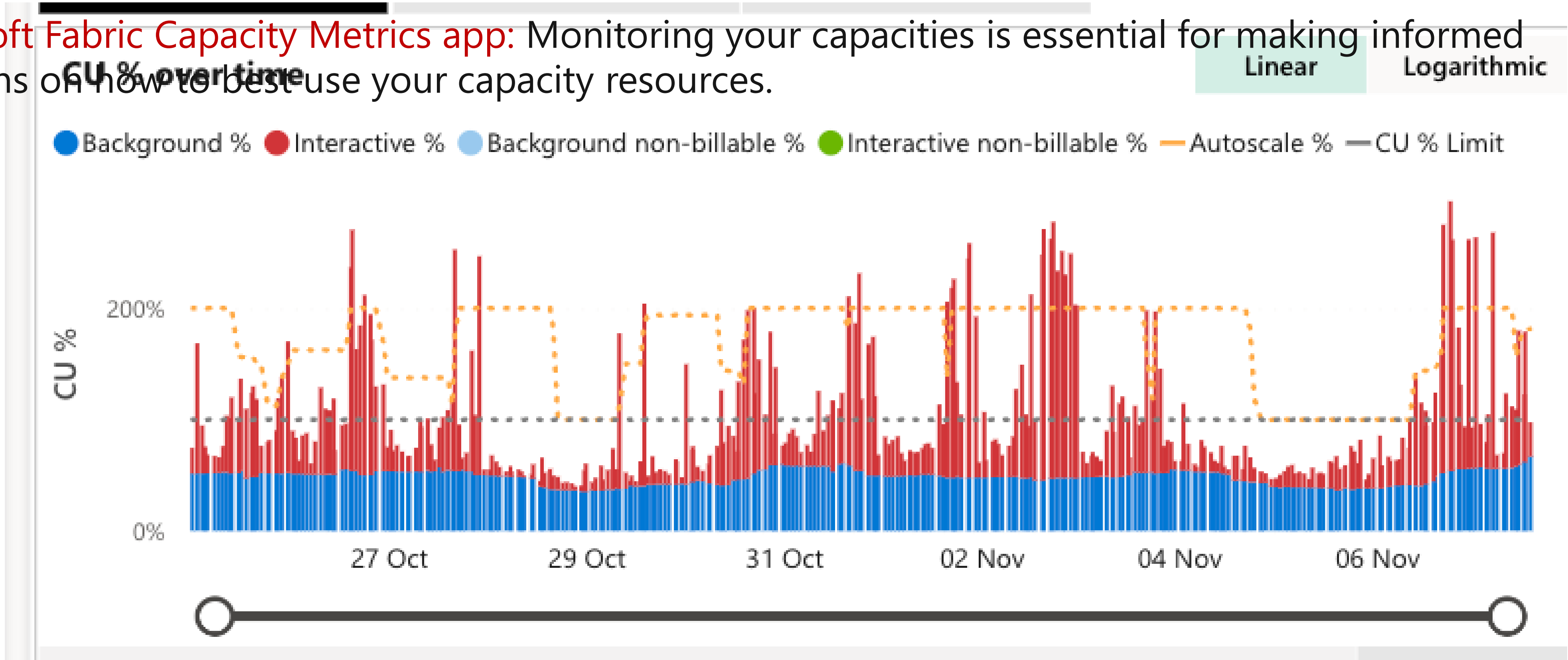
Apply | Cancel

ⓘ This setting applies to the entire organization

Environment and Architecture

- Power BI Capacity - Monitoring

Microsoft Fabric Capacity Metrics app: Monitoring your capacities is essential for making informed decisions on how to best use your capacity resources.



Right Approach

Environment and Architecture

Power BI Capacity

Gateway Configuration

Driver Configuration

Semantic Model

Connection Type

Data Model

Report

Dax Query

Report Interaction

Environment and Architecture

- Gateway Configuration - Connection type

Separating sources prevents the gateway from having thousands of DirectQuery requests queued up at the same time as the morning's scheduled refresh of a large-size data model that's used for the company's main dashboard.

Schedule Refresh

Depending on your query size and the number of refreshes that occur per day, you can choose to stay with the recommended minimum hardware requirements or upgrade to a higher performance machine. If a given query isn't folded, transformations occur on the gateway machine. As a result, **the gateway machine benefits from having more available RAM**.

Direct Query

A query is sent each time any user opens the report or looks at data. If you expect more than 1,000 users to access the data concurrently, make sure your computer has robust and capable hardware components. **More CPU cores result in better throughput for a DirectQuery connection.**

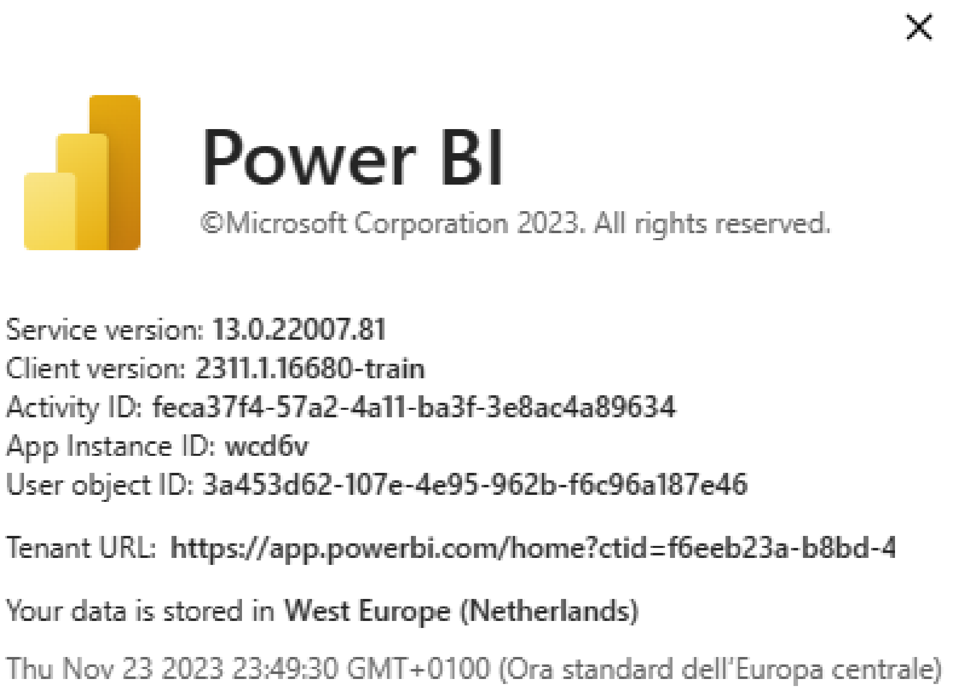
Environment and Architecture

- Gateway Configuration - Location

The location of the gateway installation can have significant effect on your query performance.

Try to make sure that your gateway, data source locations, and the Power BI tenant are as close as possible to each other to minimize network latency.

To determine your Power BI tenant location, in the Power BI service select the question mark (**?**) icon in the upper-right corner. Then select **About Power BI**.



Power BI

©Microsoft Corporation 2023. All rights reserved.

Service version: 13.0.22007.81
Client version: 2311.1.16680-train
Activity ID: feca37f4-57a2-4a11-ba3f-3e8ac4a89634
App Instance ID: wcd6v
User object ID: 3a453d62-107e-4e95-962b-f6c96a187e46

Tenant URL: https://app.powerbi.com/home?ctid=f6eeb23a-b8bd-4

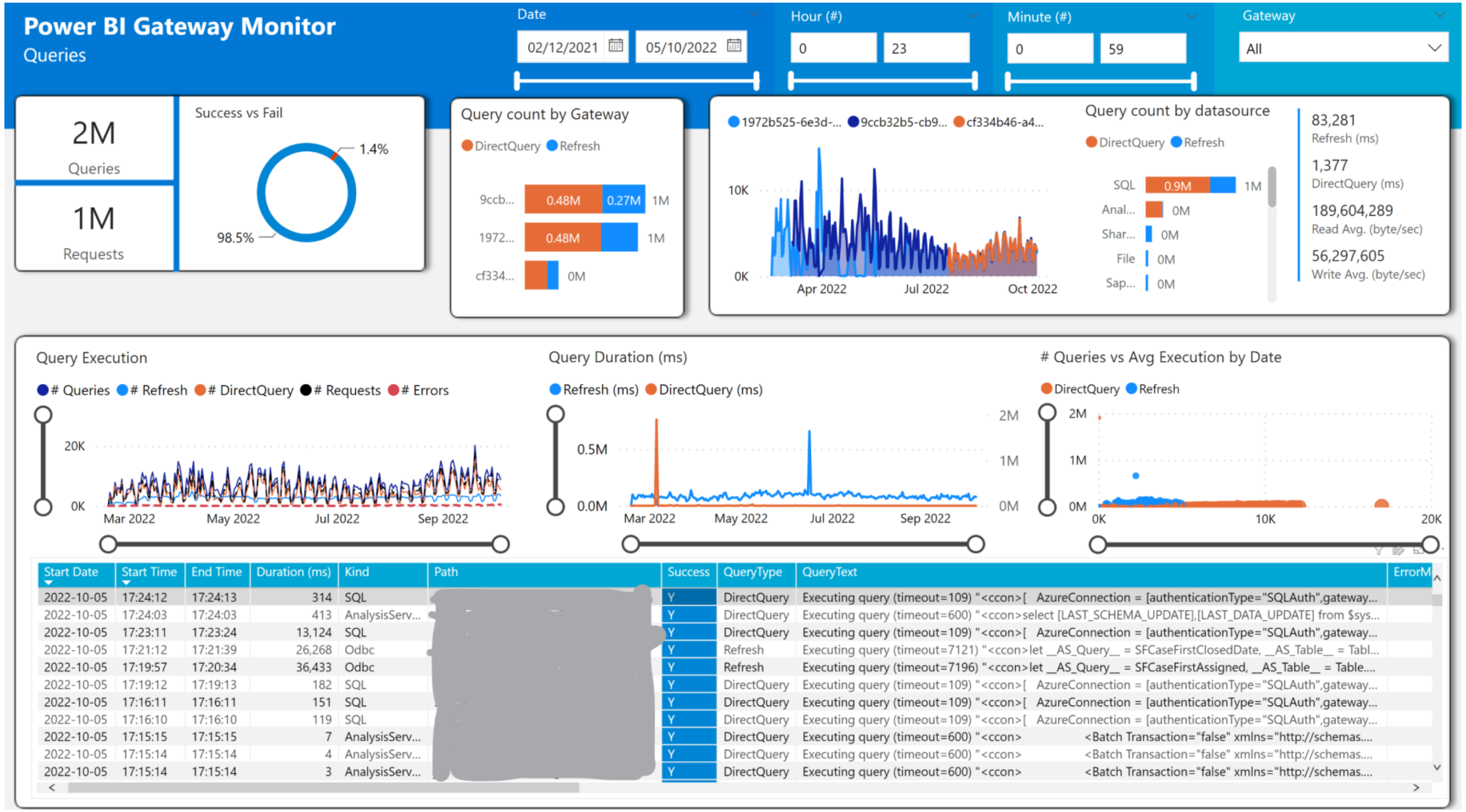
Your data is stored in West Europe (Netherlands)

Thu Nov 23 2023 23:49:30 GMT+0100 (Ora standard dell'Europa centrale)

Environment and Architecture

- Gateway Configuration - Monitoring

Rui Romano
Principal Program Manager



Power BI Gateway Monitor
Queries

Date: 02/12/2021 – 05/10/2022
Hour (#): 0 – 23
Minute (#): 0 – 59
Gateway: All

2M Queries
1M Requests

83,281 Refresh (ms)
1,377 DirectQuery (ms)
189,604,289 Read Avg. (byte/sec)
56,297,605 Write Avg. (byte/sec)

Start Date	Start Time	End Time	Duration (ms)	Kind	Path	Success	QueryType	QueryText	ErrorM
2022-10-05	17:24:12	17:24:13	314	SQL	[redacted]	Y	DirectQuery	Executing query (timeout=109) "<ccon>[AzureConnection = [authenticationType="SQLAuth",gateway...	
2022-10-05	17:24:03	17:24:03	413	AnalysisServ...	[redacted]	Y	DirectQuery	Executing query (timeout=600) "<ccon>select [LAST_SCHEMA_UPDATE],[LAST_DATA_UPDATE] from $sys...	
2022-10-05	17:23:11	17:23:24	13,124	SQL	[redacted]	Y	DirectQuery	Executing query (timeout=109) "<ccon>[AzureConnection = [authenticationType="SQLAuth",gateway...	
2022-10-05	17:21:12	17:21:39	26,268	Odbc	[redacted]	Y	Refresh	Executing query (timeout=7121) "<ccon>let __AS_Query__ = SFCaseFirstClosedDate, __AS_Table__ = Tabl...	
2022-10-05	17:19:57	17:20:34	36,433	Odbc	[redacted]	Y	Refresh	Executing query (timeout=7196) "<ccon>let __AS_Query__ = SFCaseFirstAssigned, __AS_Table__ = Table....	
2022-10-05	17:19:12	17:19:13	182	SQL	[redacted]	Y	DirectQuery	Executing query (timeout=109) "<ccon>[AzureConnection = [authenticationType="SQLAuth",gateway...	
2022-10-05	17:16:11	17:16:11	151	SQL	[redacted]	Y	DirectQuery	Executing query (timeout=109) "<ccon>[AzureConnection = [authenticationType="SQLAuth",gateway...	
2022-10-05	17:16:10	17:16:10	119	SQL	[redacted]	Y	DirectQuery	Executing query (timeout=109) "<ccon>[AzureConnection = [authenticationType="SQLAuth",gateway...	
2022-10-05	17:15:15	17:15:15	7	AnalysisServ...	[redacted]	Y	DirectQuery	Executing query (timeout=600) "<ccon> <Batch Transaction="false" xmlns="http://schemas....	
2022-10-05	17:15:14	17:15:14	4	AnalysisServ...	[redacted]	Y	DirectQuery	Executing query (timeout=600) "<ccon> <Batch Transaction="false" xmlns="http://schemas....	
2022-10-05	17:15:14	17:15:14	3	AnalysisServ...	[redacted]	Y	DirectQuery	Executing query (timeout=600) "<ccon> <Batch Transaction="false" xmlns="http://schemas....	

Environment and Architecture

- Driver Configuration

Managed Driver

Microsoft Documentation

Vendor Documentation

Avoid Generic Drivers

Right Approach

Environment and Architecture

Power BI Capacity

Gateway Configuration

Driver Configuration

Semantic Model

Connection Type

Data Model

Report

Dax Query

Report Interaction

Semantic Model – Connecton Type

- not everything is the same

Import Mode

Direct Query Mode

Composite Mode

Mixed Mode

Direct Lake

Semantic Model – Data Model

1. Import only those columns you really need
2. Reduce the column cardinality!
3. Same as for columns, keep only those rows you need
4. Avoid using calculated columns whenever possible, since they are not being optimally compressed
5. Use proper data types (DateTime to Date)
6. Avoid using floating point data types
7. Disable Auto Date/Time option for data loading
8. Use Star schema instead of Snowflake schema when possible
9. Avoid bi-directional and many-to-many relationships against high cardinality columns
10. Overusing Calculated Columns
11. Overusing expensive relationships
12. Remove Primary Key columns from fact tables

Semantic Model – Data Model

Reduce the column cardinality!

High cardinality columns in PBI models can be particularly expensive. The best practice is to remove them from the model, especially when these columns are not relevant for data analysis, such as a GUID or timestamp of a SQL Server table.

However, whenever the information they contain is required, you can optimize these columns by splitting the value in two or more columns with a smaller number of distinct values.

This will require some more effort when accessing the column value, but the saving can be so high in large tables that it could definitely worth the effort.

Semantic Model – Data Model

Avoid using calculated columns whenever possible, since they are not being optimally compressed

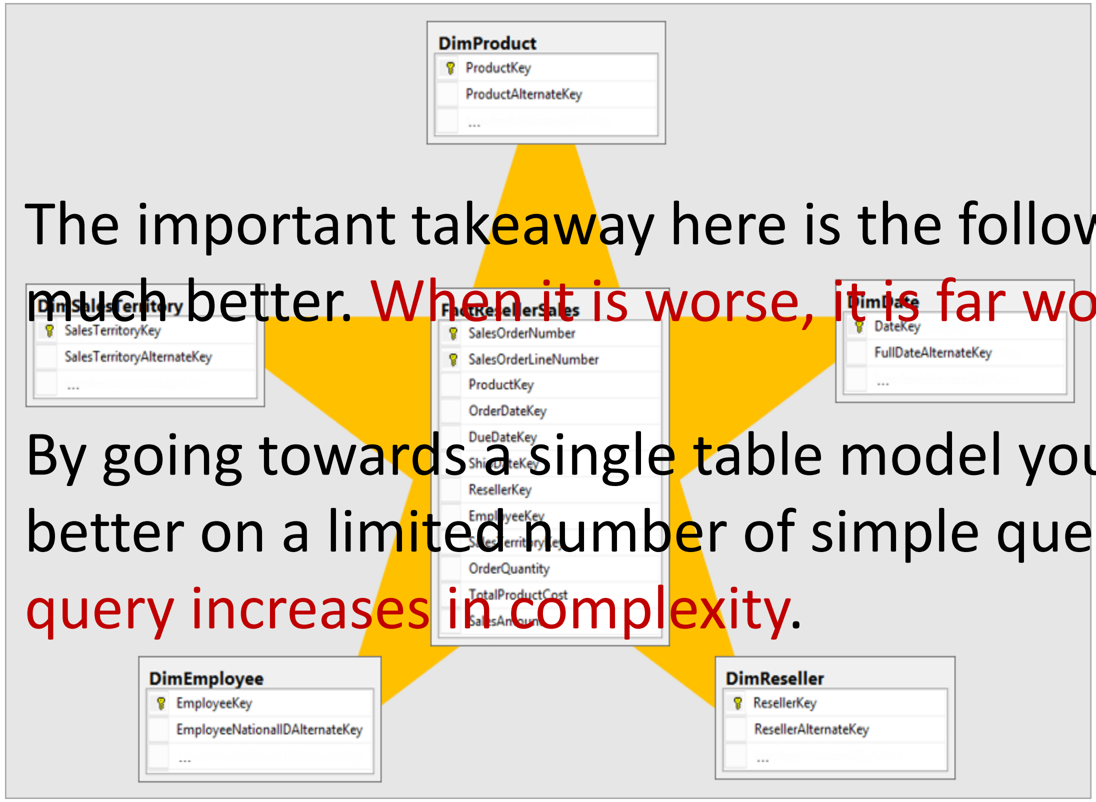
Created at the Data Source

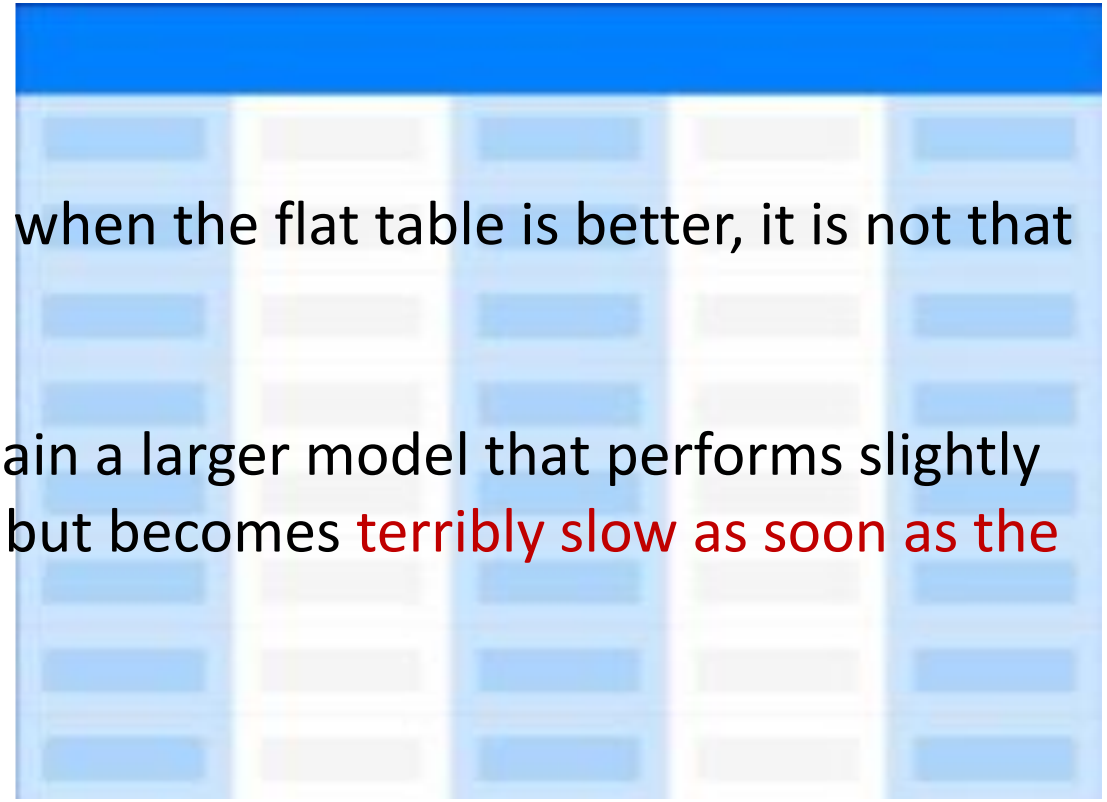
Created with Power Query

Created in DAX

Semantic Model – Data Model

Use Star schema instead of flat table



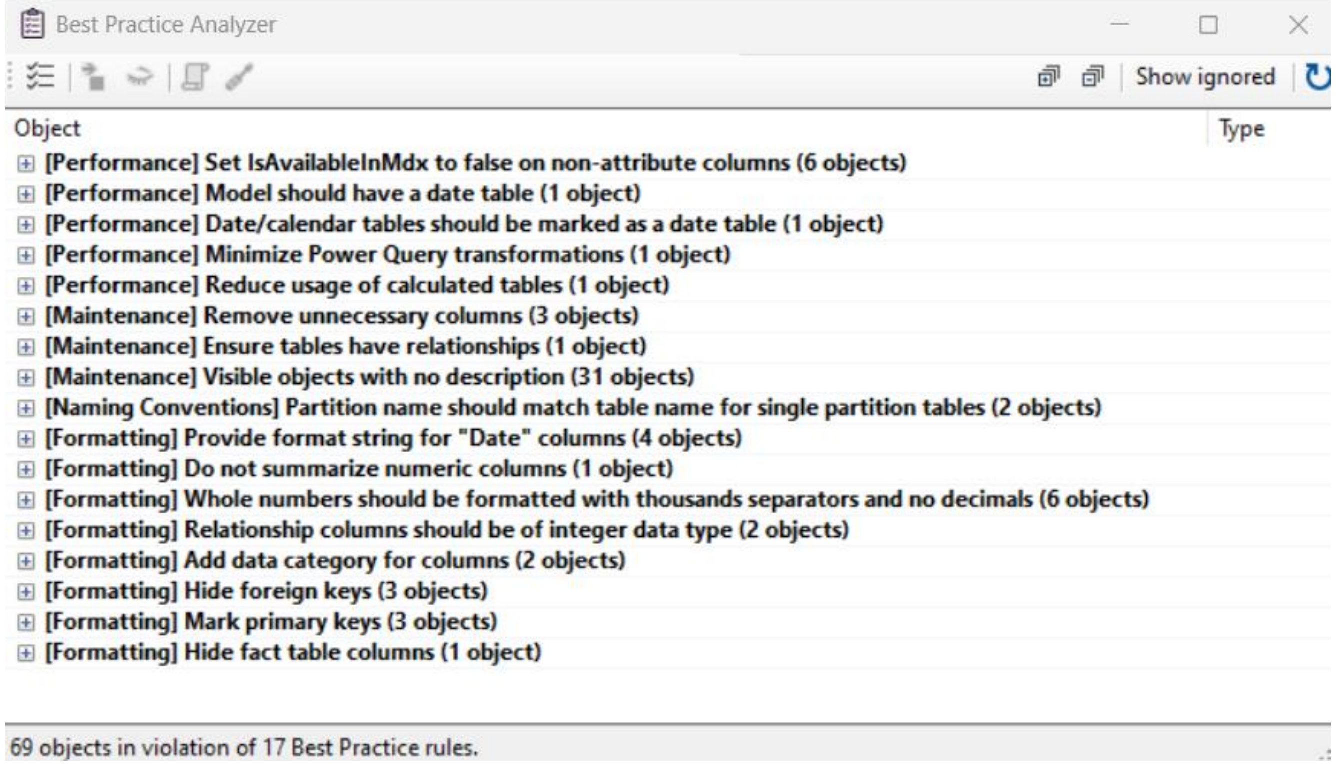


The important takeaway here is the following: when the flat table is better, it is not that much better. When it is worse, it is far worse!

By going towards a single table model you obtain a larger model that performs slightly better on a limited number of simple queries, but becomes terribly slow as soon as the query increases in complexity.

Semantic Model – Data Model

External Tools

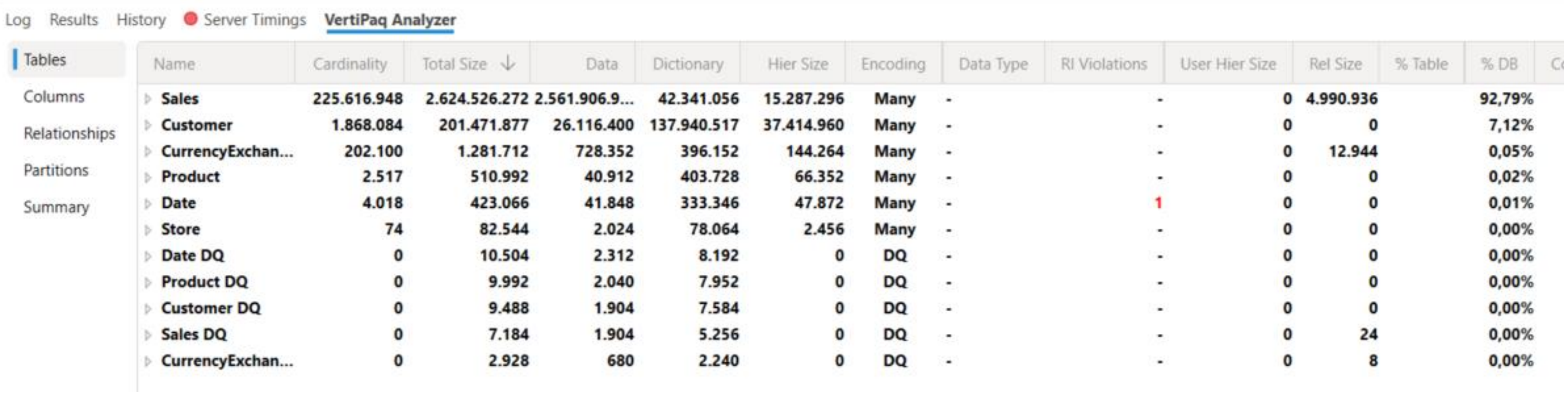


- [Performance] Set IsAvailableInMdx to false on non-attribute columns (6 objects)
- [Performance] Model should have a date table (1 object)
- [Performance] Date/calendar tables should be marked as a date table (1 object)
- [Performance] Minimize Power Query transformations (1 object)
- [Performance] Reduce usage of calculated tables (1 object)
- [Maintenance] Remove unnecessary columns (3 objects)
- [Maintenance] Ensure tables have relationships (1 object)
- [Maintenance] Visible objects with no description (31 objects)
- [Naming Conventions] Partition name should match table name for single partition tables (2 objects)
- [Formatting] Provide format string for "Date" columns (4 objects)
- [Formatting] Do not summarize numeric columns (1 object)
- [Formatting] Whole numbers should be formatted with thousands separators and no decimals (6 objects)
- [Formatting] Relationship columns should be of integer data type (2 objects)
- [Formatting] Add data category for columns (2 objects)
- [Formatting] Hide foreign keys (3 objects)
- [Formatting] Mark primary keys (3 objects)
- [Formatting] Hide fact table columns (1 object)

69 objects in violation of 17 Best Practice rules.

Semantic Model – Data Model

External Tools



Name	Cardinality	Total Size ↓	Data	Dictionary	Hier Size	Encoding	Data Type	RI Violations	User Hier Size	Rel Size	% Table	% DB	C
Sales	225.616.948	2.624.526.272	2.561.906.9...	42.341.056	15.287.296	Many	-	-	0	4.990.936		92,79%	
Customer	1.868.084	201.471.877	26.116.400	137.940.517	37.414.960	Many	-	-	0	0		7,12%	
CurrencyExchan...	202.100	1.281.712	728.352	396.152	144.264	Many	-	-	0	12.944		0,05%	
Product	2.517	510.992	40.912	403.728	66.352	Many	-	-	0	0		0,02%	
Date	4.018	423.066	41.848	333.346	47.872	Many	-	1	0	0		0,01%	
Store	74	82.544	2.024	78.064	2.456	Many	-	-	0	0		0,00%	
Date DQ	0	10.504	2.312	8.192	0	DQ	-	-	0	0		0,00%	
Product DQ	0	9.992	2.040	7.952	0	DQ	-	-	0	0		0,00%	
Customer DQ	0	9.488	1.904	7.584	0	DQ	-	-	0	0		0,00%	
Sales DQ	0	7.184	1.904	5.256	0	DQ	-	-	0	24		0,00%	
CurrencyExchan...	0	2.928	680	2.240	0	DQ	-	-	0	8		0,00%	

DAX STUDIO

OVERNET.

Right Approach

Environment and Architecture

Power BI Capacity

Gateway Configuration

Driver Configuration

Semantic Model

Connection Type

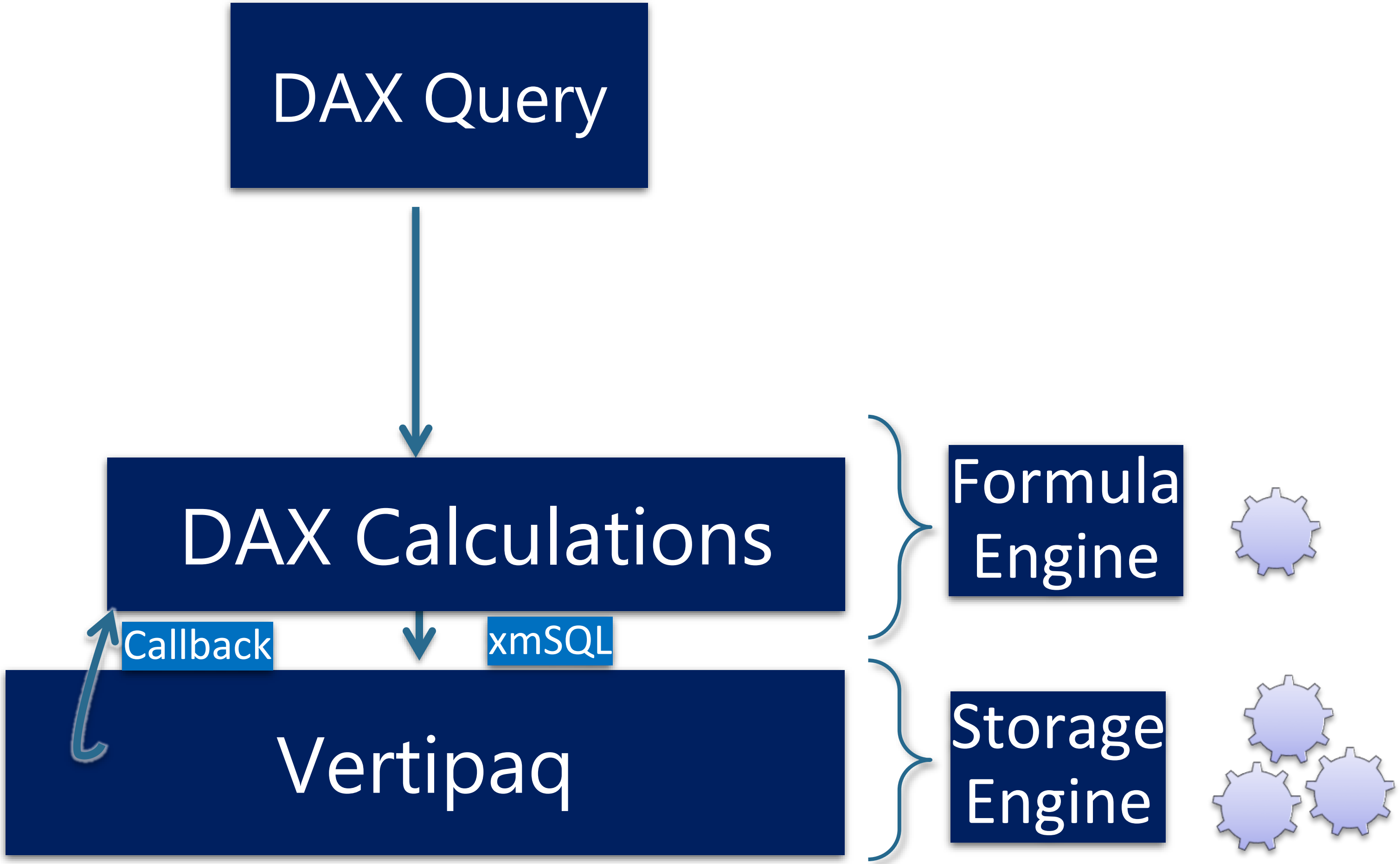
Data Model

Report

Dax Query

Report Interaction

Report – Query Architecture



FORMULA ENGINE (FE)

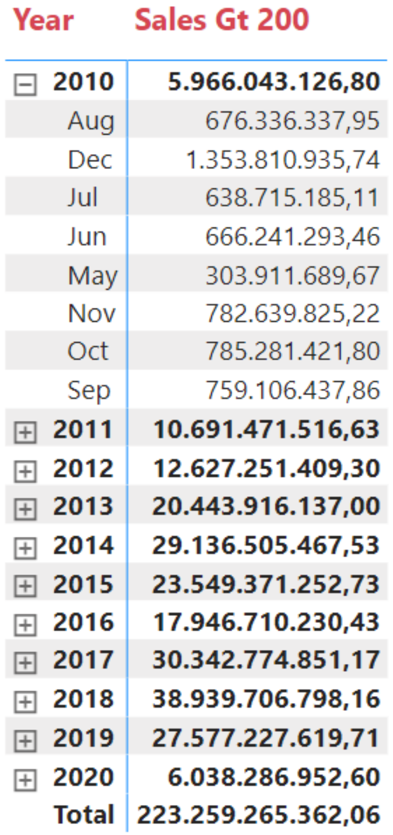
- Monothreaded, ask and fetch SE queries sequentially
- Very Smart !
- Build execution plans
- No cache at FE level for DAX queries

STORAGE ENGINE (SE)

- Multithreaded, scan segments in parallel
- Very Fast !
- Can only handle simple arithmetic calculations (callback can be needed)
- SE query results are cached (if no callback)

Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
⊟ 2010	5.966.043.126,80
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ 2011	10.691.471.516,63
⊞ 2012	12.627.251.409,30
⊞ 2013	20.443.916.137,00
⊞ 2014	29.136.505.467,53
⊞ 2015	23.549.371.252,73
⊞ 2016	17.946.710.230,43
⊞ 2017	30.342.774.851,17
⊞ 2018	38.939.706.798,16
⊞ 2019	27.577.227.619,71
⊞ 2020	6.038.286.952,60
Total	223.259.265.362,06

It all starts with a report that users describe as slow. It contains a simple measure that computes sales for only transactions whose amount is greater than 200 US

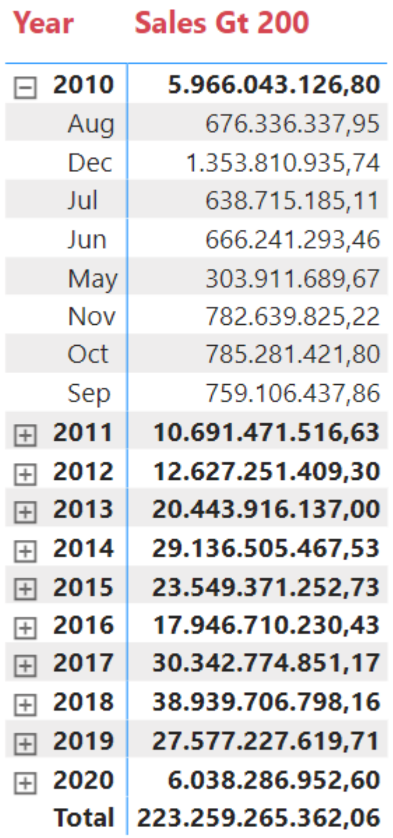
DAX MEASURE

```
Sales Gt 200 = SUMX (Sales, IF (Sales[Quantity] * Sales[Net Price] >= 200, Sales[Quantity] * Sales[Net Price]))
```

```
SUMX (
    Sales,
    IF (
        Sales[Quantity] * Sales[Net Price] >= 200,
        Sales[Quantity] * Sales[Net Price]
    )
)
```

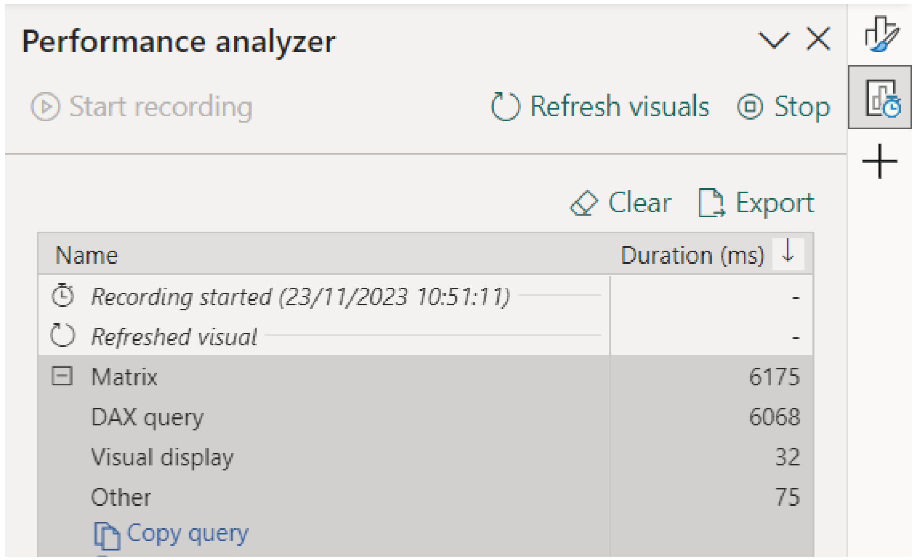
Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
⊟ 2010	5.966.043.126,80
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ 2011	10.691.471.516,63
⊞ 2012	12.627.251.409,30
⊞ 2013	20.443.916.137,00
⊞ 2014	29.136.505.467,53
⊞ 2015	23.549.371.252,73
⊞ 2016	17.946.710.230,43
⊞ 2017	30.342.774.851,17
⊞ 2018	38.939.706.798,16
⊞ 2019	27.577.227.619,71
⊞ 2020	6.038.286.952,60
Total	223.259.265.362,06

We use the Performance Analyzer tool in Power BI Desktop to retrieve the DAX query executed for the visual. The query took 6 seconds to run.

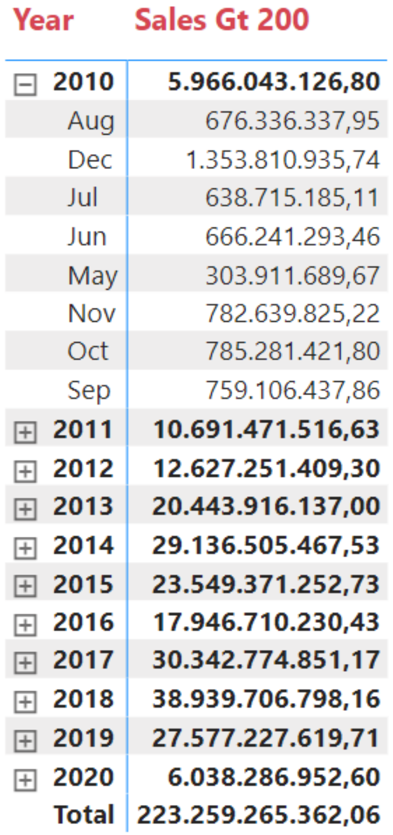


Performance analyzer

Name	Duration (ms)
Recording started (23/11/2023 10:51:11)	-
Refreshed visual	-
⊟ Matrix	6175
DAX query	6068
Visual display	32
Other	75
Copy query	

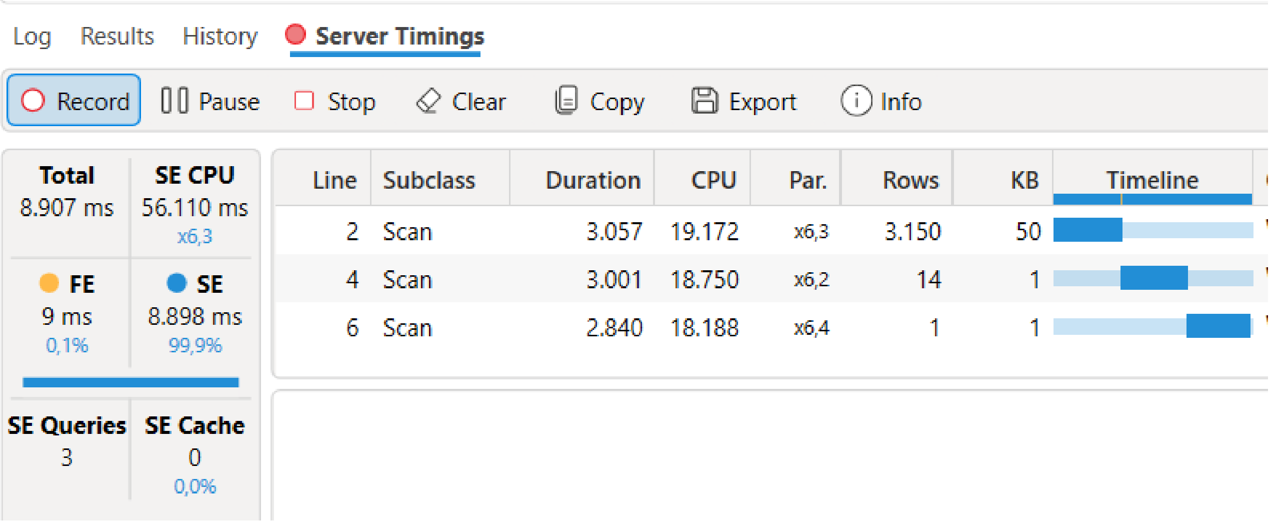
Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
⊟ 2010	**5.966.043.126,80**
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ 2011	**10.691.471.516,63**
⊞ 2012	**12.627.251.409,30**
⊞ 2013	**20.443.916.137,00**
⊞ 2014	**29.136.505.467,53**
⊞ 2015	**23.549.371.252,73**
⊞ 2016	**17.946.710.230,43**
⊞ 2017	**30.342.774.851,17**
⊞ 2018	**38.939.706.798,16**
⊞ 2019	**27.577.227.619,71**
⊞ 2020	**6.038.286.952,60**
Total	**223.259.265.362,06**

The first thing to do is to execute the query in DAX Studio with Server Timings enabled to obtain the first baseline. Later on, we will check that the simplified query did not change the timings in such a way that the issue seems resolved. Here is the DAX Studio timings report



Total 8.907 ms

SE CPU 56.110 ms x6,3

● **FE** 9 ms 0,1%

● **SE** 8.898 ms 99,9%

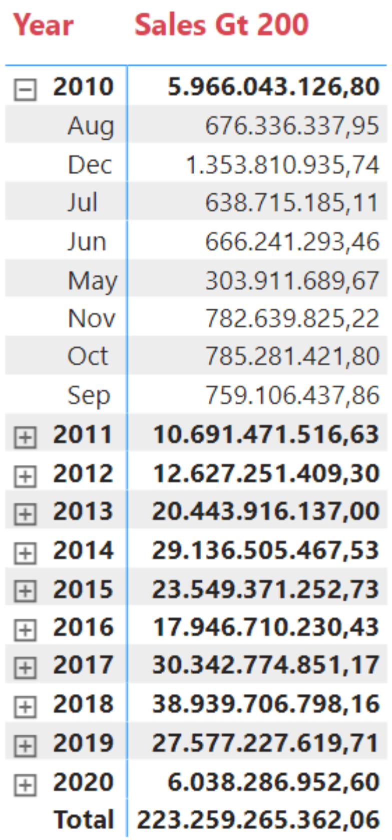
SE Queries 3

SE Cache 0 0,0%

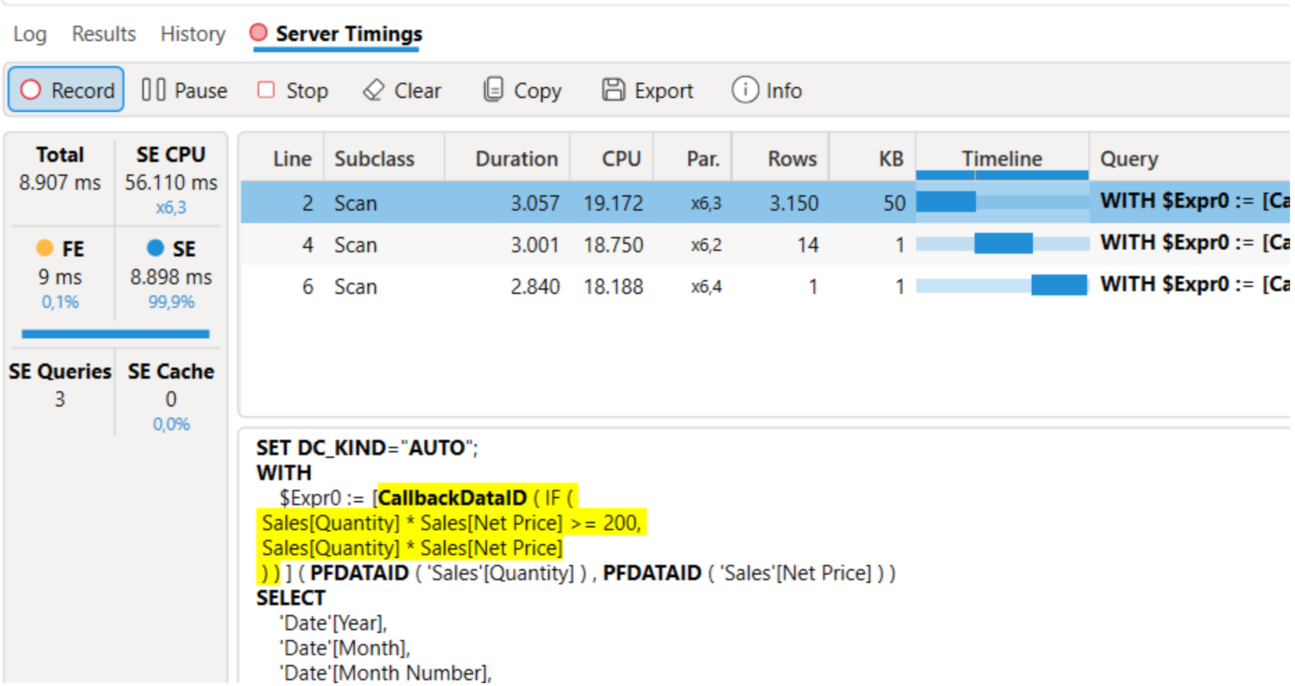
Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline
2	Scan	3.057	19.172	x6,3	3.150	50	
4	Scan	3.001	18.750	x6,2	14	1	
6	Scan	2.840	18.188	x6,4	1	1	

Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
⊟ 2010	5.966.043.126,80
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ 2011	10.691.471.516,63
⊞ 2012	12.627.251.409,30
⊞ 2013	20.443.916.137,00
⊞ 2014	29.136.505.467,53
⊞ 2015	23.549.371.252,73
⊞ 2016	17.946.710.230,43
⊞ 2017	30.342.774.851,17
⊞ 2018	38.939.706.798,16
⊞ 2019	27.577.227.619,71
⊞ 2020	6.038.286.952,60
Total	223.259.265.362,06



Total	SE CPU
8.907 ms	56.110 ms x6,3
● FE 9 ms 0,1%	● SE 8.898 ms 99,9%
SE Queries 3	SE Cache 0 0,0%

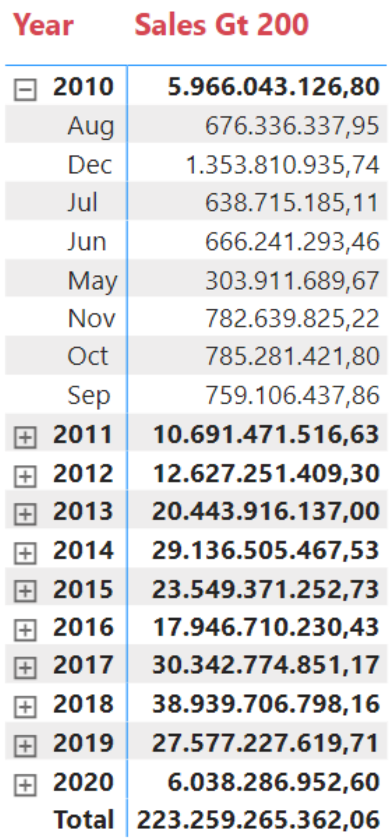
Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline	Query
2	Scan	3.057	19.172	x6,3	3.150	50		WITH $Expr0 := [Ca
4	Scan	3.001	18.750	x6,2	14	1		WITH $Expr0 := [Ca
6	Scan	2.840	18.188	x6,4	1	1		WITH $Expr0 := [Ca

```
SET DC_KIND="AUTO";
WITH
    $Expr0 := [CallbackDataID ( IF (
 Sales[Quantity] * Sales[Net Price] >= 200,
 Sales[Quantity] * Sales[Net Price]
 ) ) ] ( PFDATAID ( 'Sales'[Quantity] ) , PFDATAID ( 'Sales'[Net Price] ) )
SELECT
    'Date'[Year],
    'Date'[Month],
    'Date'[Month Number],
```

Though the entire execution time is reported as storage engine CPU, we can clearly see a CallbackDataID, indicating that the formula engine is required to kick in to compute expressions that cannot be pushed down to the storage engine.

Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
⊟ 2010	**5.966.043.126,80**
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ 2011	**10.691.471.516,63**
⊞ 2012	**12.627.251.409,30**
⊞ 2013	**20.443.916.137,00**
⊞ 2014	**29.136.505.467,53**
⊞ 2015	**23.549.371.252,73**
⊞ 2016	**17.946.710.230,43**
⊞ 2017	**30.342.774.851,17**
⊞ 2018	**38.939.706.798,16**
⊞ 2019	**27.577.227.619,71**
⊞ 2020	**6.038.286.952,60**
Total	**223.259.265.362,06**

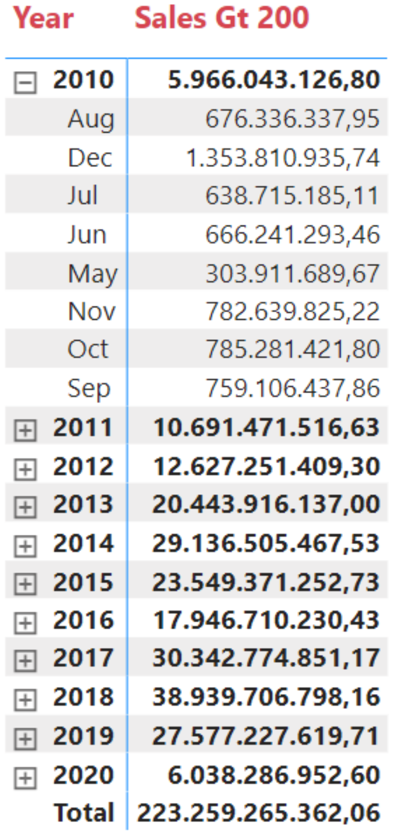
The problem is the IF statement inside the iteration carried on by SUMX, because the VertiPaq storage engine does not support conditional logic. We must rephrase the measure to avoid the IF statement; we replace it with a condition set by CALCULATE to rely on filtering rather than IF.

```
Sales Gt 200 =
SUMX (
    Sales,
    IF (
        Sales[Quantity] * Sales[Net Price] >= 200,
        Sales[Quantity] * Sales[Net Price]
    )
)
```

```
Sales Gt 200 =
CALCULATE (
    SUMX ( Sales, Sales[Quantity] * Sales[Net Price] ),
    FILTER ( Sales, Sales[Quantity] * Sales[Net Price] >= 200 )
)
```

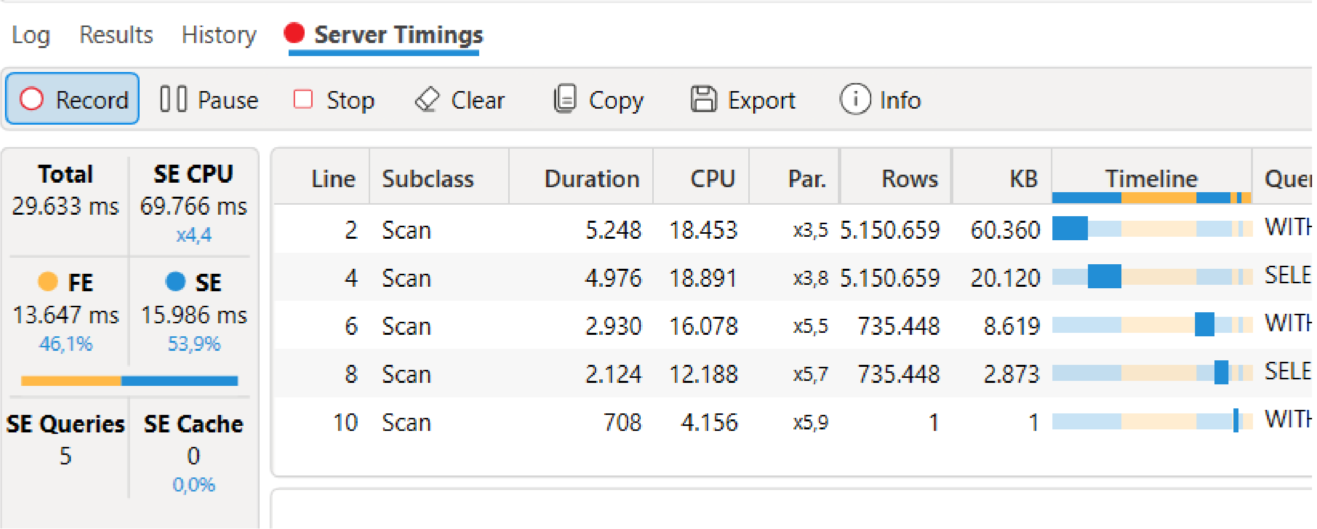
Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
2010	**5.966.043.126,80**
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
2011	**10.691.471.516,63**
2012	**12.627.251.409,30**
2013	**20.443.916.137,00**
2014	**29.136.505.467,53**
2015	**23.549.371.252,73**
2016	**17.946.710.230,43**
2017	**30.342.774.851,17**
2018	**38.939.706.798,16**
2019	**27.577.227.619,71**
2020	**6.038.286.952,60**
Total	**223.259.265.362,06**

The storage engine CPU is a bit lower than the previous version of the measure, but the degree of parallelism is much lower this time. Moreover, the formula engine executes a significant portion of code, making the overall performance much worse than the previous one. Overall, the execution time went from 8 to 29 seconds

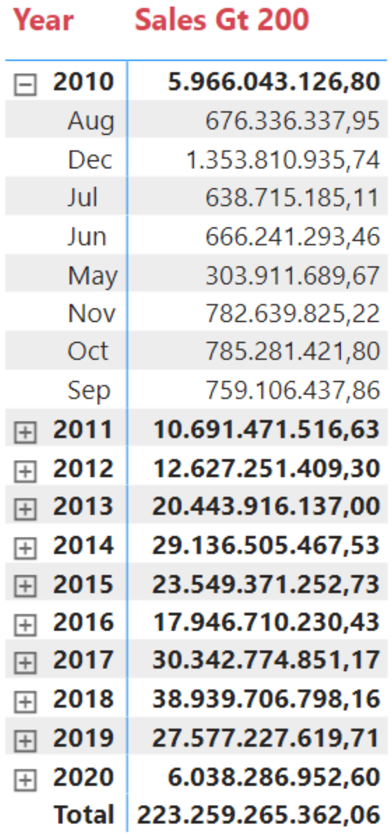


Total	SE CPU
29.633 ms	69.766 ms x4,4
● FE	● SE
13.647 ms 46,1%	15.986 ms 53,9%
SE Queries	SE Cache
5	0 0,0%

Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline	Quer
2	Scan	5.248	18.453	x3,5	5.150.659	60.360		WITH
4	Scan	4.976	18.891	x3,8	5.150.659	20.120		SELE
6	Scan	2.930	16.078	x5,5	735.448	8.619		WITH
8	Scan	2.124	12.188	x5,7	735.448	2.873		SELE
10	Scan	708	4.156	x5,9	1	1		WITH

Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



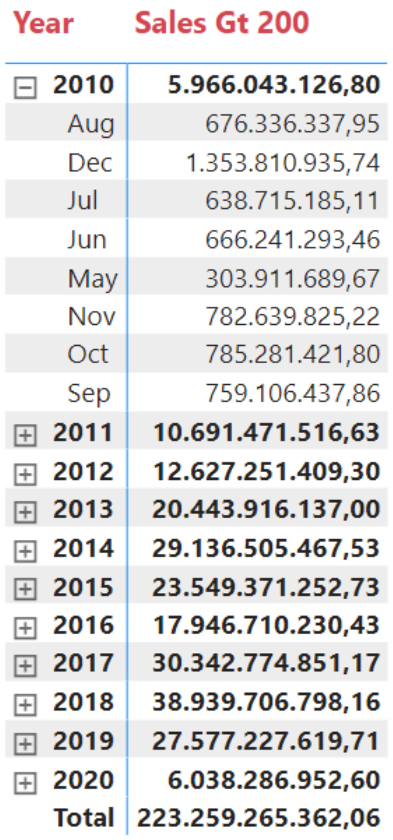
Year	Sales Gt 200
⊟ **2010**	**5.966.043.126,80**
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ **2011**	**10.691.471.516,63**
⊞ **2012**	**12.627.251.409,30**
⊞ **2013**	**20.443.916.137,00**
⊞ **2014**	**29.136.505.467,53**
⊞ **2015**	**23.549.371.252,73**
⊞ **2016**	**17.946.710.230,43**
⊞ **2017**	**30.342.774.851,17**
⊞ **2018**	**38.939.706.798,16**
⊞ **2019**	**27.577.227.619,71**
⊞ **2020**	**6.038.286.952,60**
Total	**223.259.265.362,06**

Indeed, we used a filter over Sales as a filter argument in CALCULATE. A table filter is a very bad practice that newbies oftentimes use. A filter in CALCULATE should work on the minimum number of columns required to obtain its effect

```
Sales Gt 200 =
    CALCULATE (
        SUMX ( Sales, Sales[Quantity] * Sales[Net Price] ),
        FILTER ( Sales, Sales[Quantity] * Sales[Net Price] >= 200 )
    )       ALL (Sales[Quantity] , Sales[Net Price]),
            Sales[Quantity] * Sales[Net Price] >= 200 )
    )
)
```

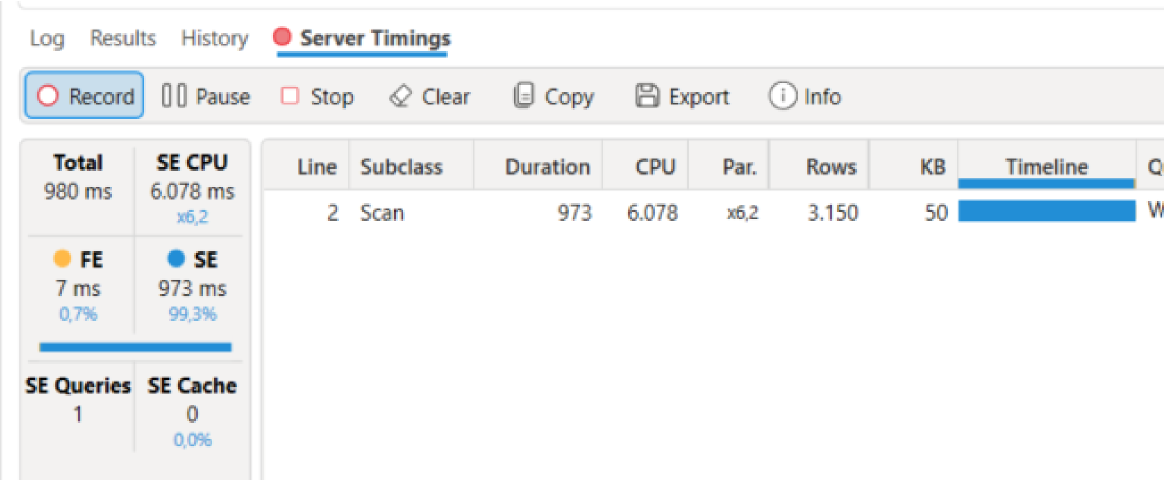
Report – Dax Optimize

Sales Table: 225,616,948 Milion Rows



Year	Sales Gt 200
⊟ **2010**	**5.966.043.126,80**
Aug	676.336.337,95
Dec	1.353.810.935,74
Jul	638.715.185,11
Jun	666.241.293,46
May	303.911.689,67
Nov	782.639.825,22
Oct	785.281.421,80
Sep	759.106.437,86
⊞ **2011**	**10.691.471.516,63**
⊞ **2012**	**12.627.251.409,30**
⊞ **2013**	**20.443.916.137,00**
⊞ **2014**	**29.136.505.467,53**
⊞ **2015**	**23.549.371.252,73**
⊞ **2016**	**17.946.710.230,43**
⊞ **2017**	**30.342.774.851,17**
⊞ **2018**	**38.939.706.798,16**
⊞ **2019**	**27.577.227.619,71**
⊞ **2020**	**6.038.286.952,60**
Total	**223.259.265.362,06**

All the indicators are just perfect. The storage engine CPU is massively reduced, the degree of parallelism is back to being exceptional, there is virtually no formula engine involved in the query and no CallbackDataIDs anywhere.
Materialization is reduced from 5 million rows to only 3,150 rows



Total	SE CPU
980 ms	6.078 ms x6,2
● FE 7 ms 0,7%	● SE 973 ms 99,3%
SE Queries 1	SE Cache 0 0,0%

Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline	Q
2	Scan	973	6.078	x6,2	3.150	50		W

Report – Dax Query

1. Use Variables when you can precompute and reuse some calculations
2. Try to avoid CallBack (ex : complex filter)
3. Avoided this function SEARCH, IFERROR, CONTAINS, and INTERSECT functions
4. The FILTER function is often overused. Its main purpose is for filtering columns based on measure values.
5. Add column and measure references in your DAX expressions
6. Use ISBLANK() instead of =Blank() check
7. Use COUNTROWS instead of COUNT
8. Work upstream, if possible
9. format your code! (Use DAX Formatter)
10. Split your calculations in smaller blocks

Performance Issues

Environment and Architecture

Power BI Capacity

Gateway Configuration

Driver Configuration

Semantic Model

Connection Type

Data Model

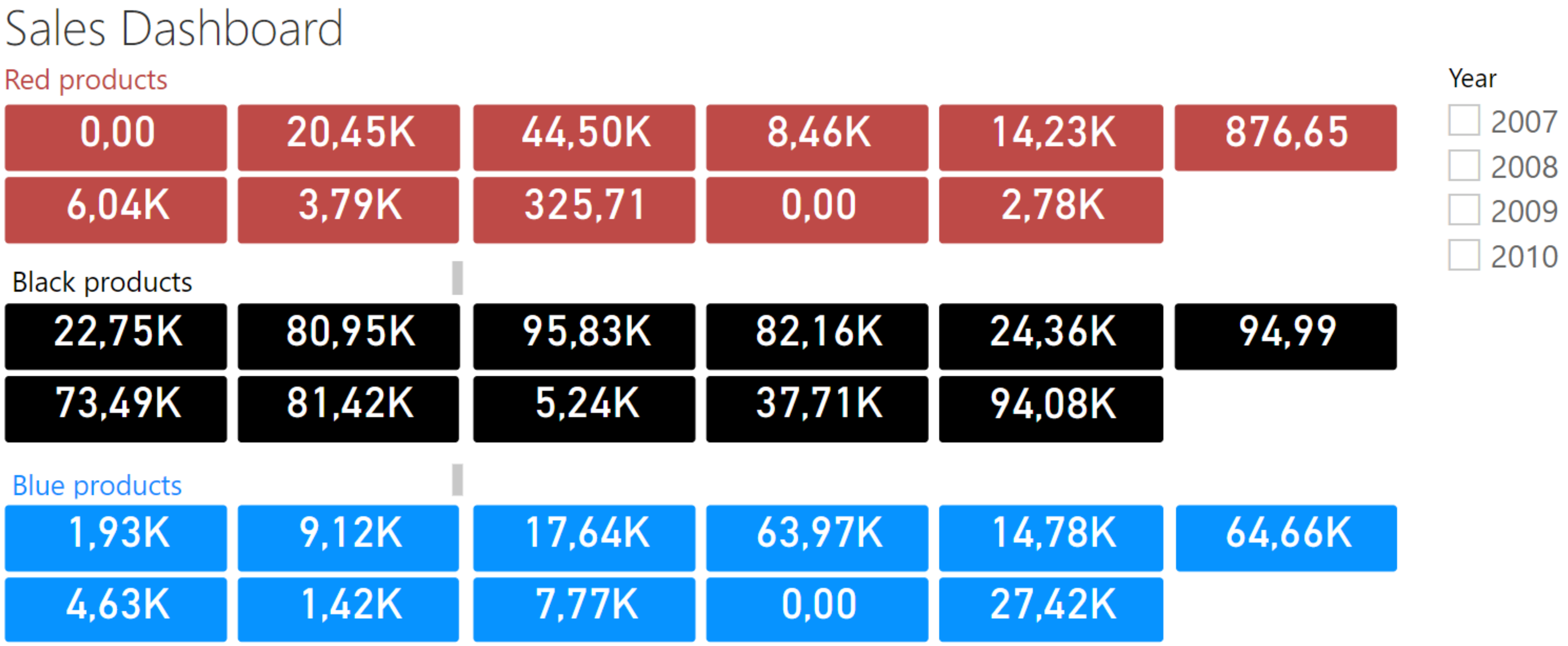
Report

Dax Query

Report Interaction

Report – Report Interaction

The problem behind this report being slow is not to be found anywhere in the DAX code or in the model.
This report is slow because there are too many card visuals. In that scenario, the solution would be to use small multiples rather than focusing on the DAX code of the measures.



Sales Dashboard

Red products

0,00	20,45K	44,50K	8,46K	14,23K	876,65
6,04K	3,79K	325,71	0,00	2,78K	

Black products

22,75K	80,95K	95,83K	82,16K	24,36K	94,99
73,49K	81,42K	5,24K	37,71K	94,08K	

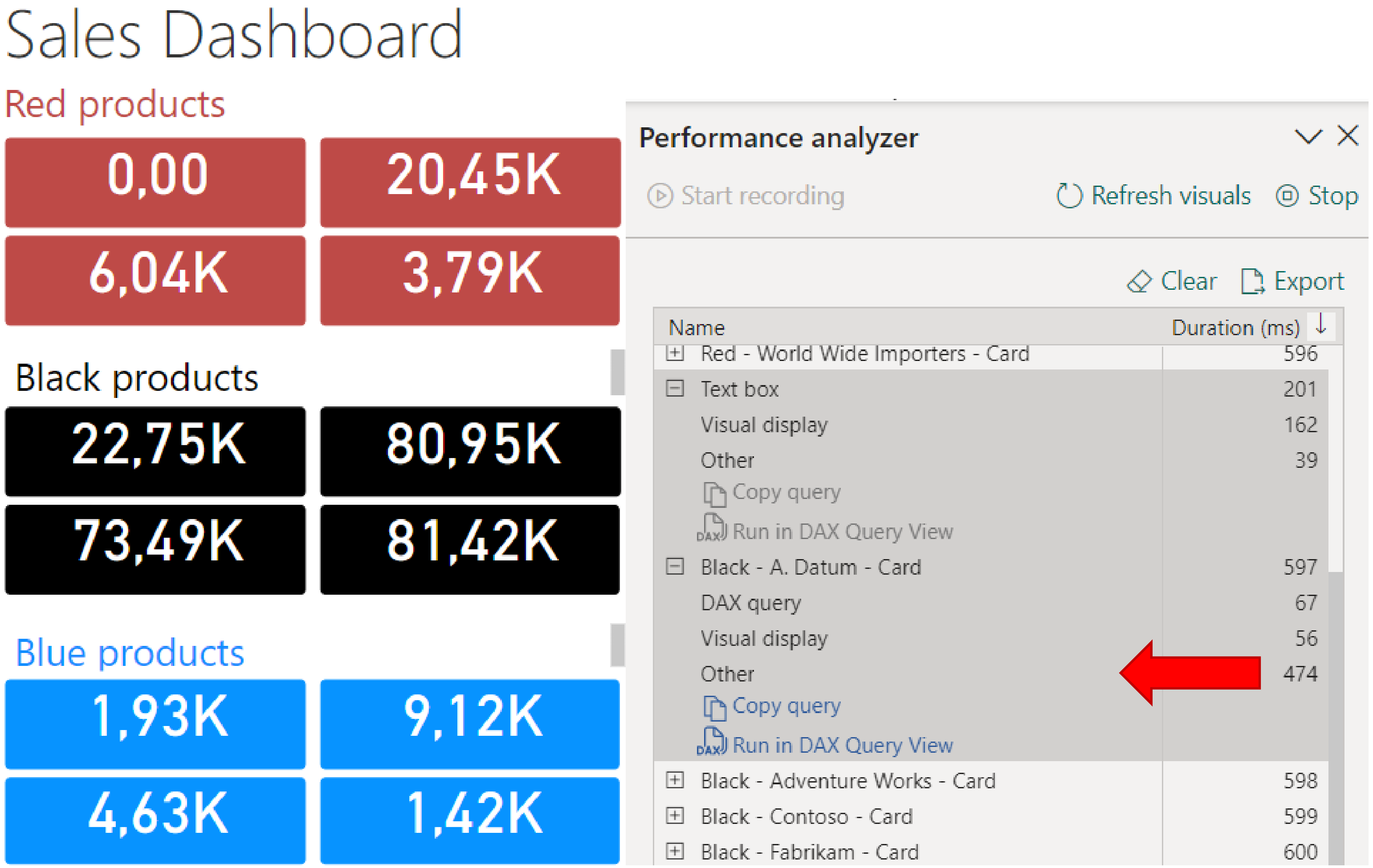
Blue products

1,93K	9,12K	17,64K	63,97K	14,78K	64,66K
4,63K	1,42K	7,77K	0,00	27,42K	

Year

□ 2007
□ 2008
□ 2009
□ 2010

Report – Report Interaction



Sales Dashboard

Red products

0,00 | 20,45K
6,04K | 3,79K

Black products

22,75K | 80,95K
73,49K | 81,42K

Blue products

1,93K | 9,12K
4,63K | 1,42K

Performance analyzer

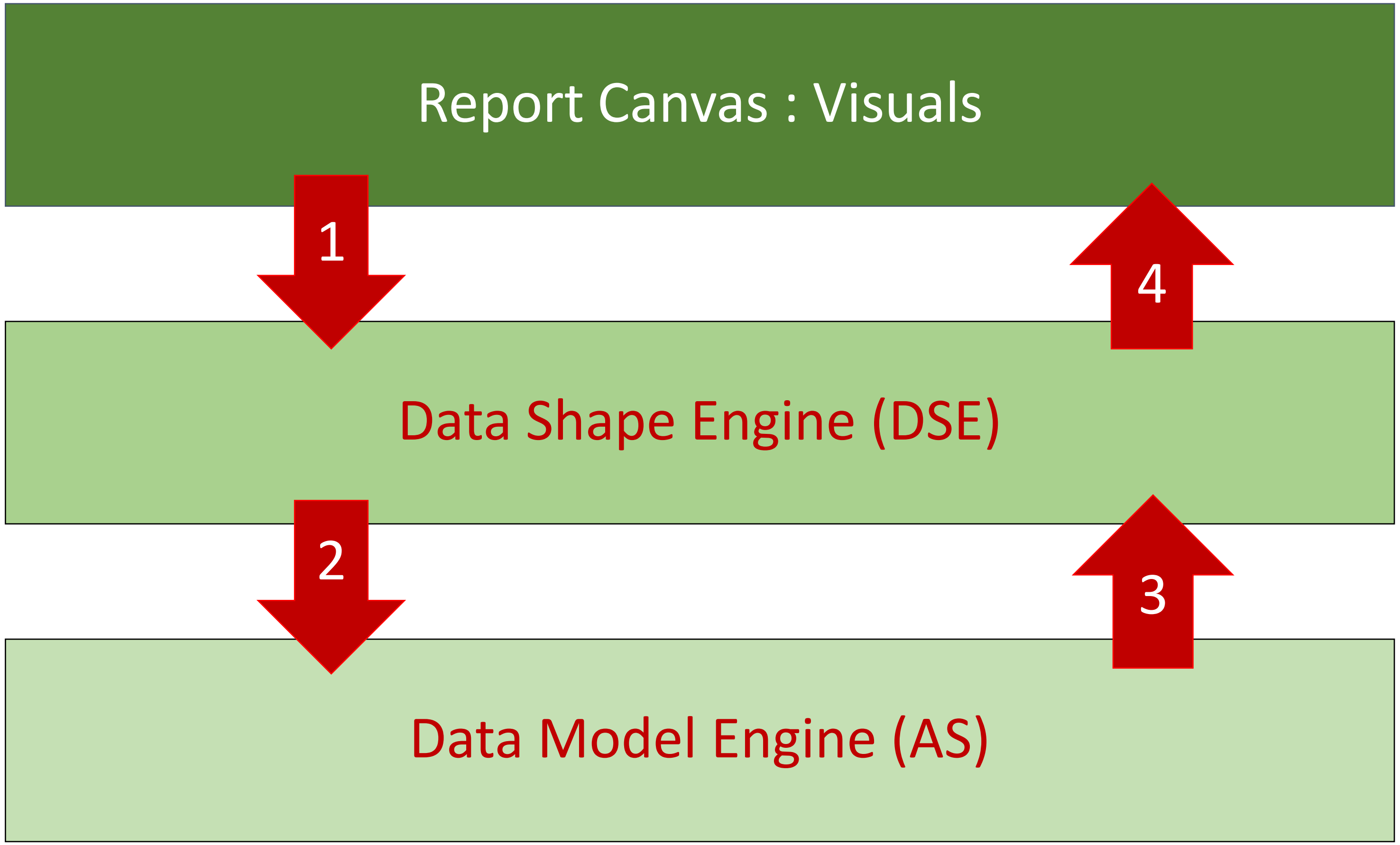
Name	Duration (ms)
⊞ Red - World Wide Importers - Card	596
⊟ Text box	201
Visual display	162
Other	39
Copy query	
Run in DAX Query View	
⊟ Black - A. Datum - Card	597
DAX query	67
Visual display	56
Other	474
Copy query	
Run in DAX Query View	
⊞ Black - Adventure Works - Card	598
⊞ Black - Contoso - Card	599
⊞ Black - Fabrikam - Card	600

DAX query: this is the time required to execute the DAX query.

Visual display: indicates the time required to render the visual. On more complex visuals, the rendering time might be significant.

Other: this is the time the visual had to wait before Power BI Desktop could execute the DAX query to populate the visual

Report – Report Interaction



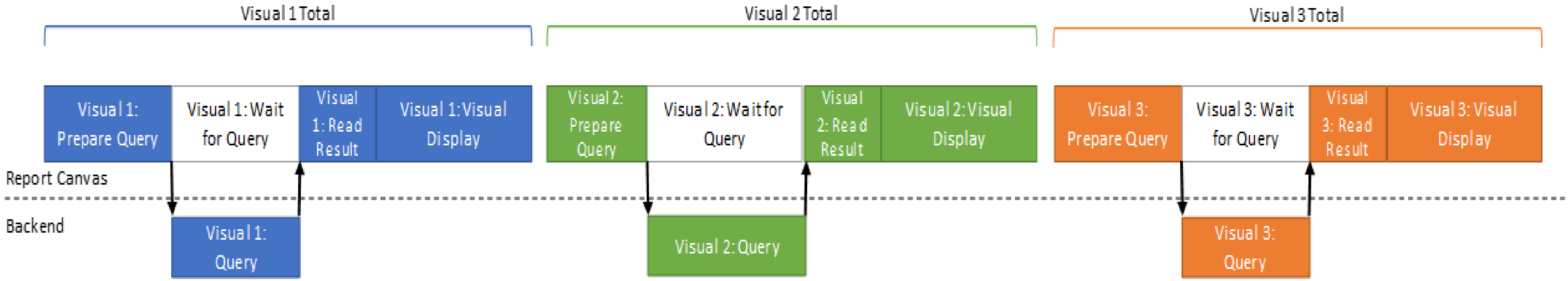
Execution Workflow

Report Canvas retrieves data using an internal, high-level Power BI query language known as Semantic Query.

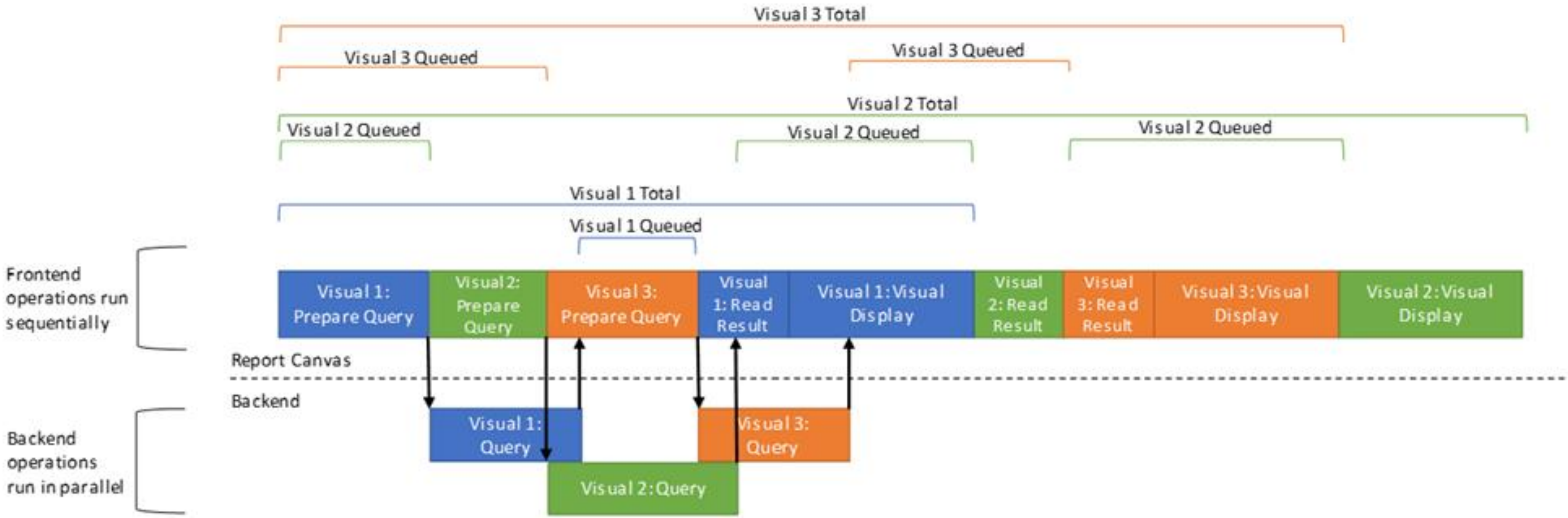
Data Shape Engine (DSE) evaluates semantic queries by generating and executing one or more DAX queries against a semantic model.

Data Model Engine (AS) stores the data model and provides reporting services, such as DAX query evaluation.

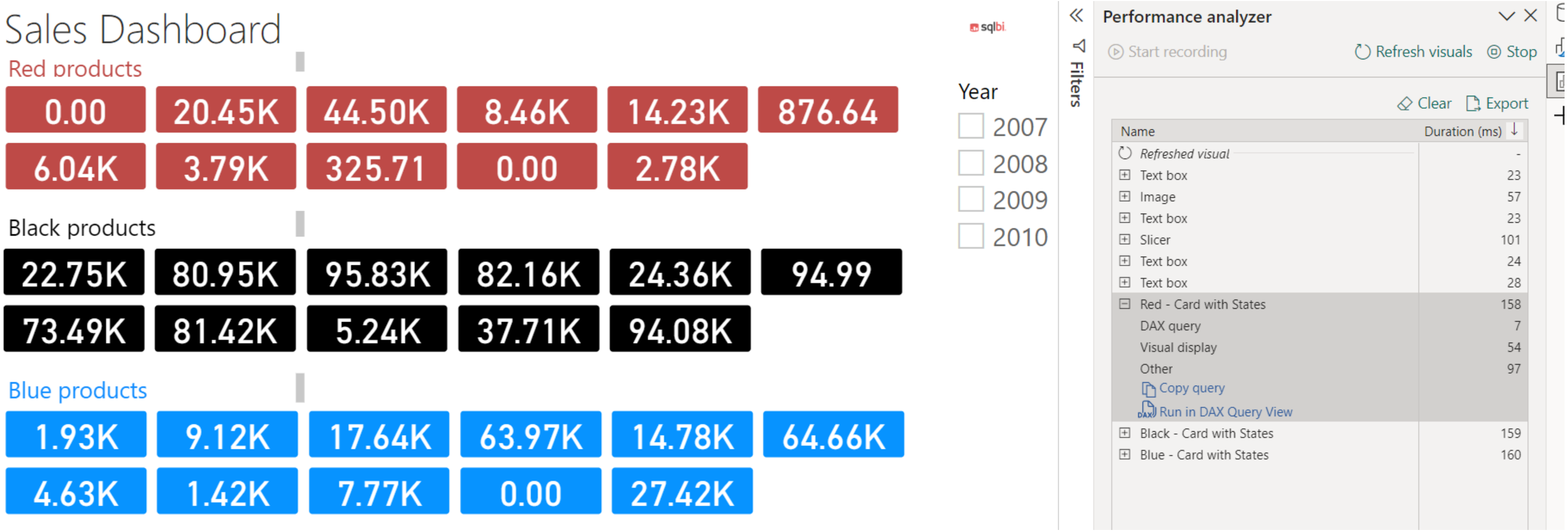
Report – Report Interaction



Report – Report Interaction



Report – Report Interaction



Sales Dashboard

Red products

0.00	20.45K	44.50K	8.46K	14.23K	876.64
6.04K	3.79K	325.71	0.00	2.78K	

Black products

22.75K	80.95K	95.83K	82.16K	24.36K	94.99
73.49K	81.42K	5.24K	37.71K	94.08K	

Blue products

1.93K	9.12K	17.64K	63.97K	14.78K	64.66K
4.63K	1.42K	7.77K	0.00	27.42K	

sqlbi

Year

□ 2007
□ 2008
□ 2009
□ 2010

Performance analyzer

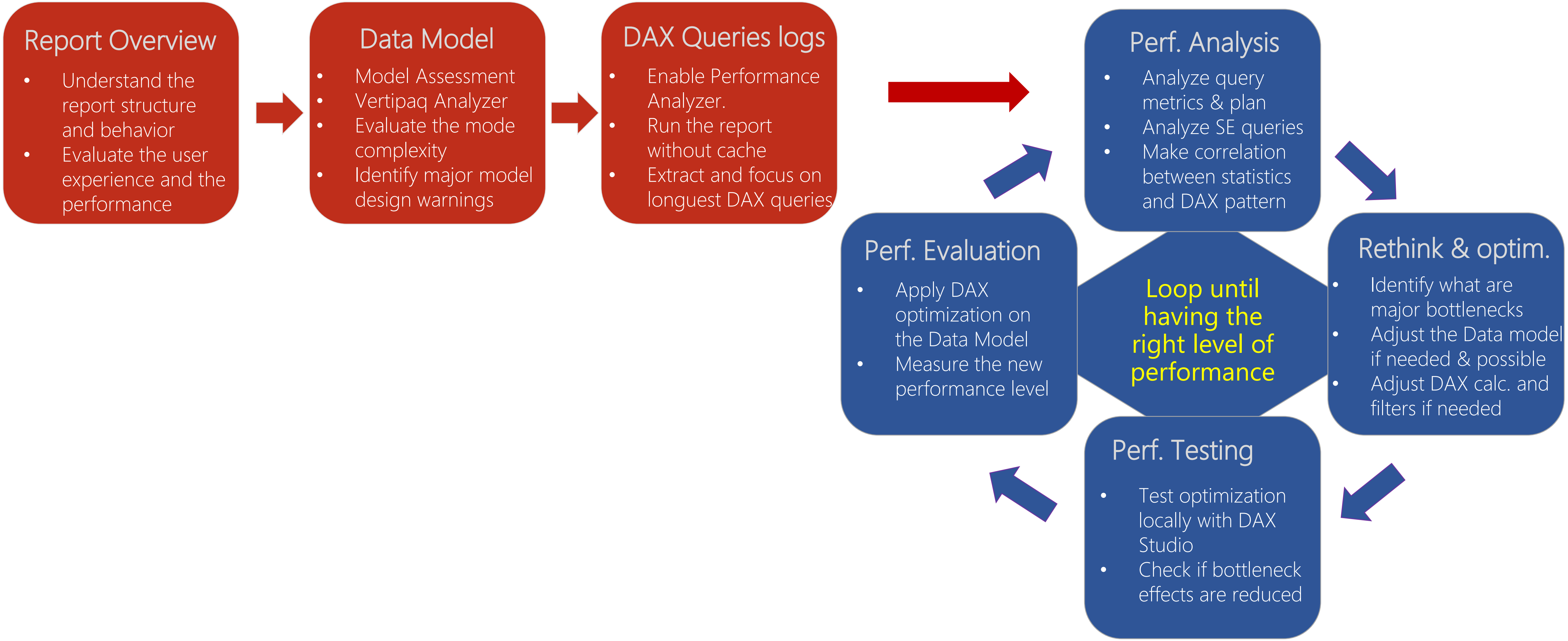
Name	Duration (ms)
Refreshed visual	-
⊞ Text box	23
⊞ Image	57
⊞ Text box	23
⊞ Slicer	101
⊞ Text box	24
⊞ Text box	28
⊟ Red - Card with States	158
DAX query	7
Visual display	54
Other	97
Copy query	
Run in DAX Query View	
⊞ Black - Card with States	159
⊞ Blue - Card with States	160

OVERNET.

Report – Interaction

1. Limit the number of visuals in dashboards and reports
2. Remove unnecessary interactions between visuals
3. Reduce the amount of data loaded on page load
4. Use tooltips to provide more information on visuals and metrics
5. Allow users to personalize visuals in a report
6. Reduce queries
7. Use report backgrounds for static images
8. Avoid scrolls within the visual and on page
9. Test custom visual performance before use
10. Use drillthrough buttons instead of expecting users to right-click on data points

Performance Remediation Process



Report Overview

- Understand the report structure and behavior
- Evaluate the user experience and the performance

Data Model

- Model Assessment
- Vertipaq Analyzer
- Evaluate the mode complexity
- Identify major model design warnings

DAX Queries logs

- Enable Performance Analyzer.
- Run the report without cache
- Extract and focus on longuest DAX queries

Perf. Analysis

- Analyze query metrics & plan
- Analyze SE queries
- Make correlation between statistics and DAX pattern

Rethink & optim.

- Identify what are major bottlenecks
- Adjust the Data model if needed & possible
- Adjust DAX calc. and filters if needed

Perf. Testing

- Test optimization locally with DAX Studio
- Check if bottleneck effects are reduced

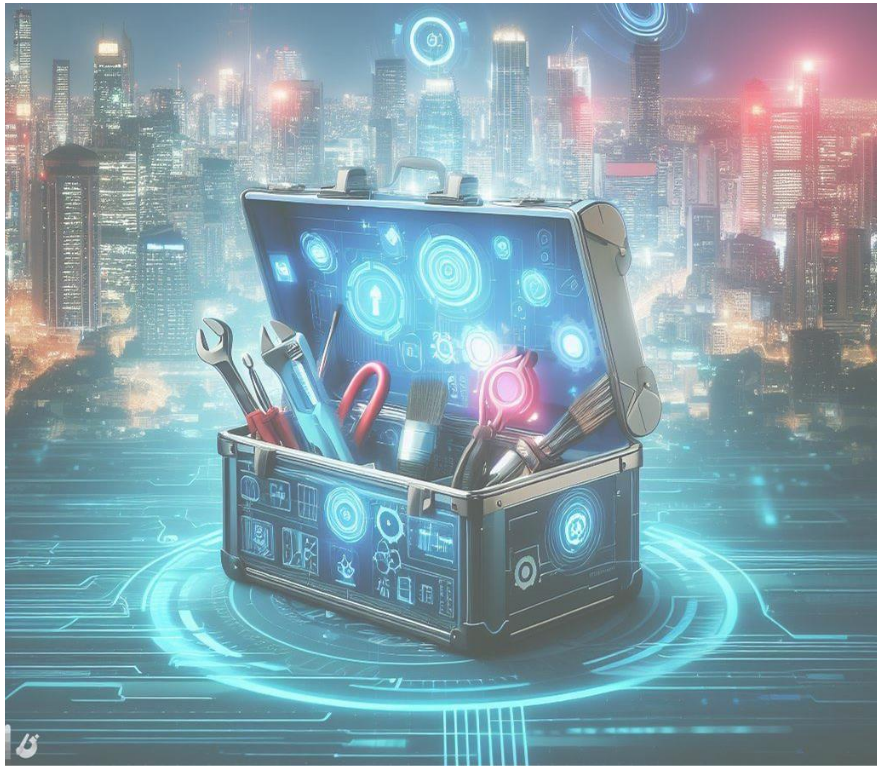
Perf. Evaluation

- Apply DAX optimization on the Data Model
- Measure the new performance level

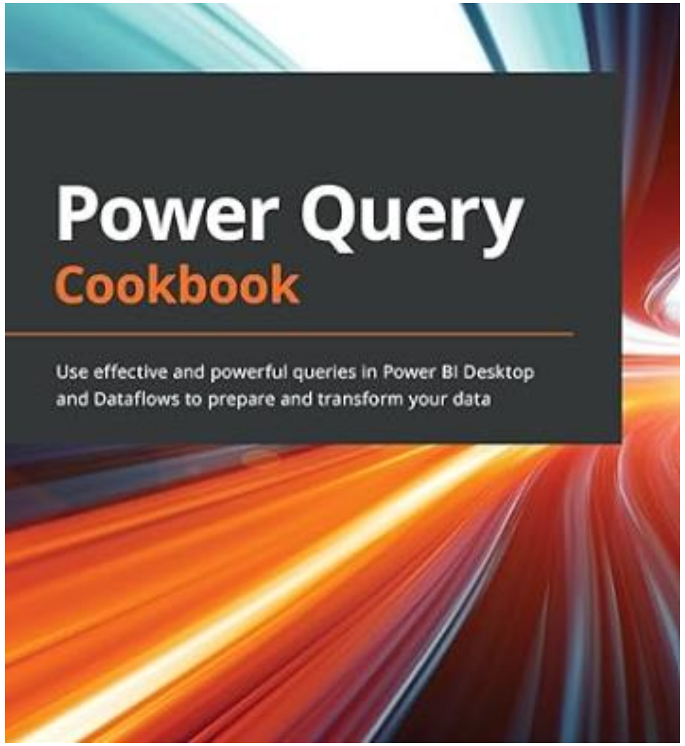
Loop until having the right level of performance

Tools

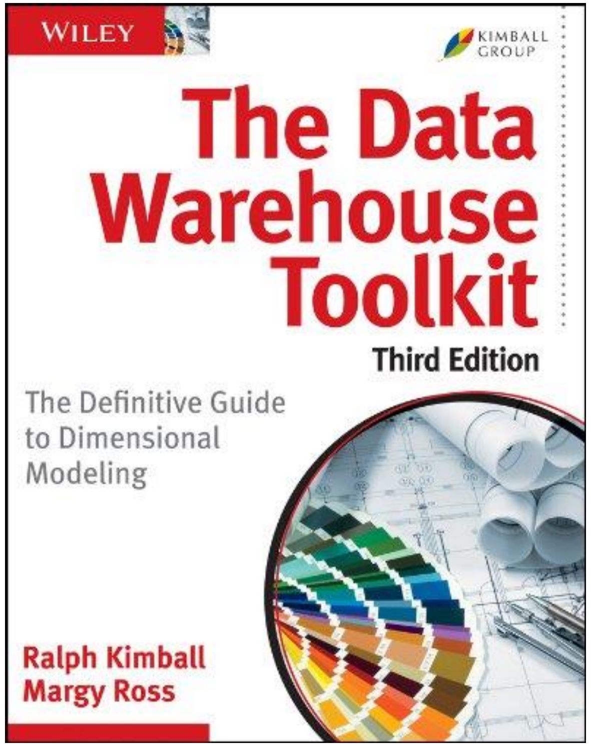
- https://tabulareditor.com/downloads
- https://daxstudio.org/
- https://www.pbiexplorer.com/
- https://www.elegantbi.com/post/reportanalyzer
- https://powerbi.microsoft.com/en-my/blog/best-practice-rules-to-improve-your-models-performance/
- https://www.elegantbi.com/post/bestpracticerulesavings



Book


Power Query
Cookbook

Use effective and powerful queries in Power BI Desktop and Dataflows to prepare and transform your data


WILEY

KIMBALL GROUP

The Data Warehouse Toolkit

Third Edition

The Definitive Guide to Dimensional Modeling

Ralph Kimball
Margy Ross

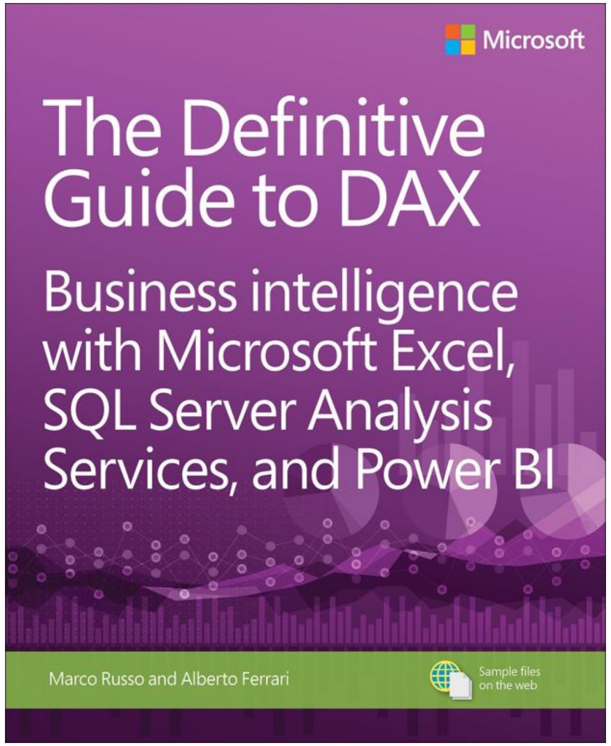
i manuali

Marilena D'Onofrio

DAX

Comprendere e usare il linguaggio delle formule di Microsoft Power BI

tecniche nuove


Microsoft

The Definitive Guide to DAX

Business intelligence with Microsoft Excel, SQL Server Analysis Services, and Power BI

Marco Russo and Alberto Ferrari

Sample files on the web

WPC 2023

28, 29, 30 NOVEMBRE NH MILANO CONGRESS CENTRE

Valuta la sessione

GRAZIE!