

# A deep dive into Direct Lake





# Thank You Sponsors!



**DATA**masterminds

Platinum



**illionX**  
experts in eenvoud

**infoSupport**  
Solid Innovator

**axians**

Gold

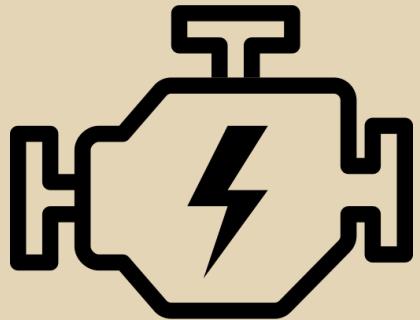
 POWER BI SENTINEL

 **INSPARK**

 **dba.nl**  
database experts

# After this session

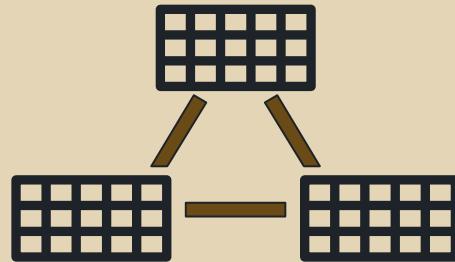
Understanding the engine  
for Direct Lake



Analyzing performance  
across different scenarios



Data modeling benefitting  
from Direct Lake



Go next level with  
advanced patterns



# Assumptions

aka Qualified guessing



# Mathias Halkjær

Principal Architect, Data & Analytics  
Fellowmind Denmark



@halkjaerm

[linkedin.com/in/mhalkjaer](https://linkedin.com/in/mhalkjaer)

[Fluxbi.com](https://Fluxbi.com)

## FAVORITE STUFF:



# Marc Lelijveld

Technical Evangelist | Solution Architect  
Macaw Netherlands



@MarcLelijveld



[linkedin.com/in/MarcLelijveld](https://linkedin.com/in/MarcLelijveld)



[Data-Marc.com](https://Data-Marc.com)

FAVORITE STUFF:





# Storage modes

# Different types of storage modes

## Three familiar storage modes

- **Import** – data cached in the model
- **DirectQuery** – queries are submitted to the back-end data source
- **Dual** – can act in both above storage modes, depending on query context

The screenshot shows the Power BI Desktop interface with a data model visualization on the left and a properties pane on the right.

**Data Model:**

- Product Subcategory:** English Product Subcategory Name, French Product Subcategory Name, ProductCategoryKey, ProductSubcategoryAltName, ProductSubcategoryKey, Spanish Product Subcategory Name.
- Product:** Arabic Description, Chinese Description, Class, Color, Days To Manufacture, Dealer Price.
- Internet Sales - Agg:** Count, Order Calendar Year, ProductSubcategoryKey, Sales Amount.
- Internet Sales:** Carrier Tracking Number, CurrencyKey, Customer PO Number, CustomerKey, Discount Amount, DueDateKey, Extended Amount, Freight Amount, Order Calendar Year.

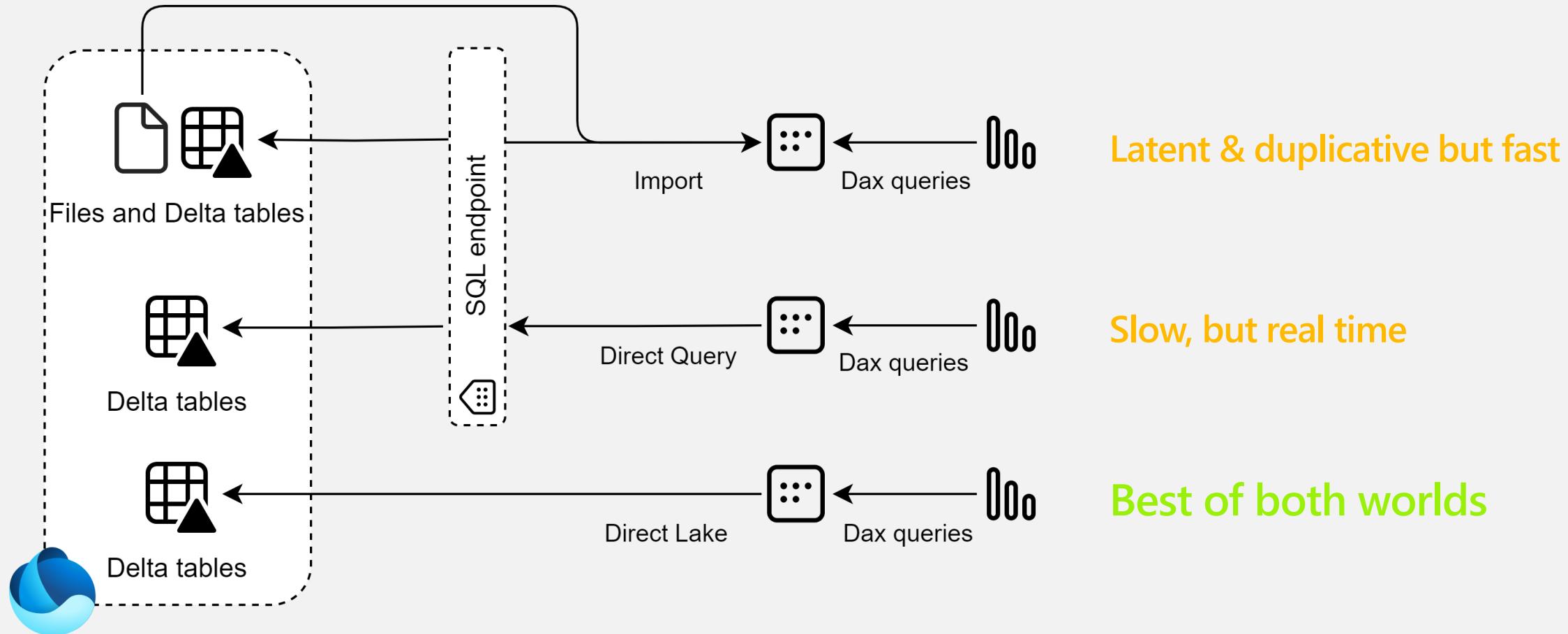
**Relationships:**

- Product Subcategory (1) — Product (\*).
- Product Subcategory (\*) — Internet Sales - Agg (1).
- Product (1) — Internet Sales (\*).

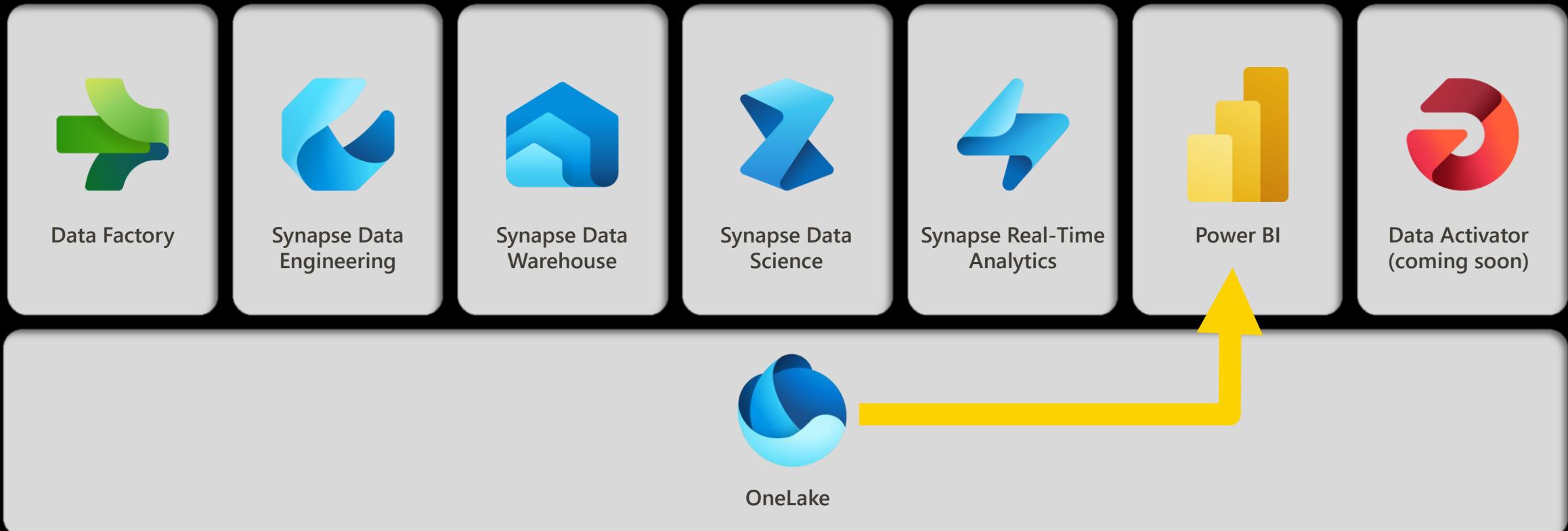
**Properties Pane (Advanced section):**

- Name: Internet Sales
- Description: Enter a description
- Synonyms: Enter a comma-separated list of synonyms for Q&A
- Row label: Select a row label
- Key column: Select a column with unique values
- Is hidden: No
- Is featured table: No
- Storage mode:** DirectQuery (highlighted with a red box)

# Direct Lake



Direct Lake is only applicable to Fabric





**Build your data model benefitting Direct Lake**

# Data transformations



No Power Query or other data transformation capabilities\*

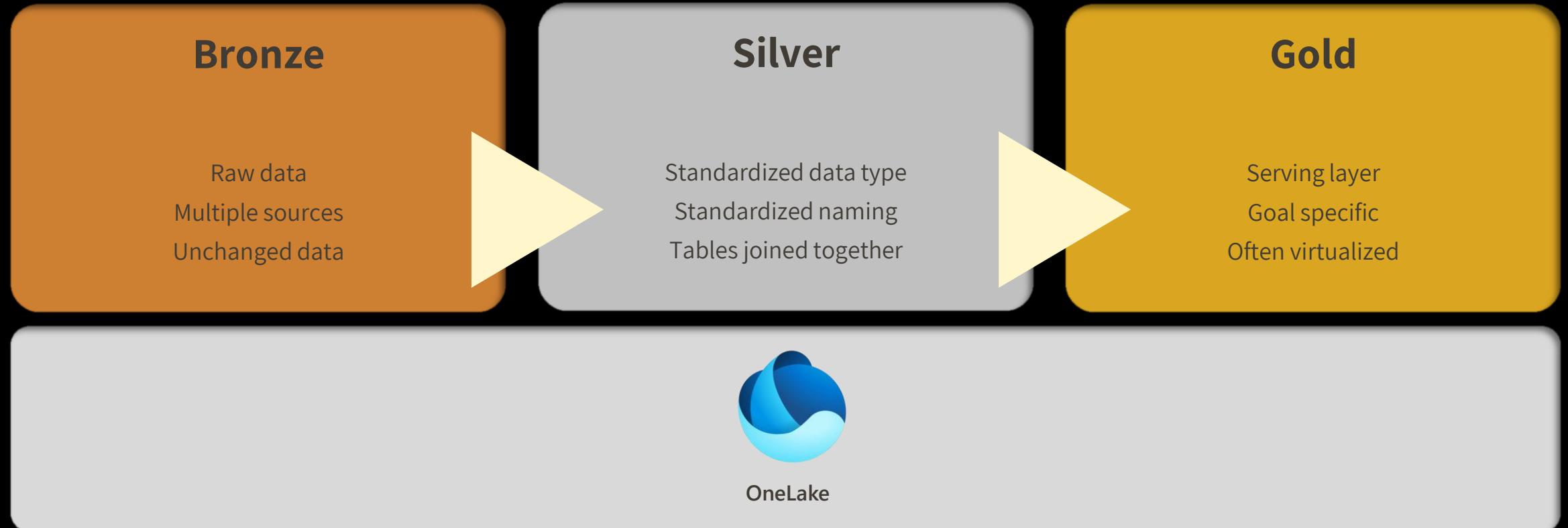


Data transformations should be done as far upstream as possible

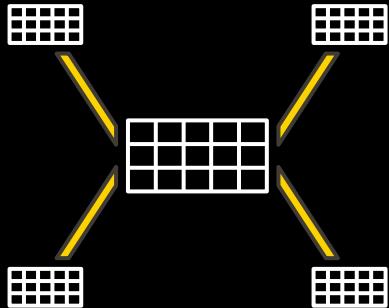


Data transformation directly in the Lakehouse unlocks “new” possibilities

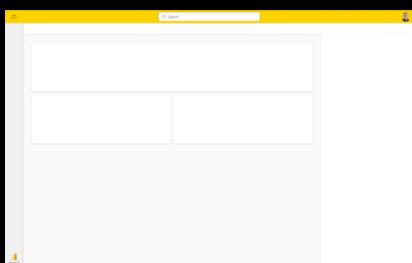
# Medallion architecture



# Data modeling



General best practice to have a star schema still applies



Web (browser) experience only to develop data models

# Data modeling best practices unchanged

- Starschema all the things!
  - Avoid bi-directional or many-to-many relationships
  - Avoid limited relationships
  - Implement role-playing dimensions rather than duplication
  - Minimize redundant measure using calculation groups
  - Avoid ambiguous data models
- ... etcetera

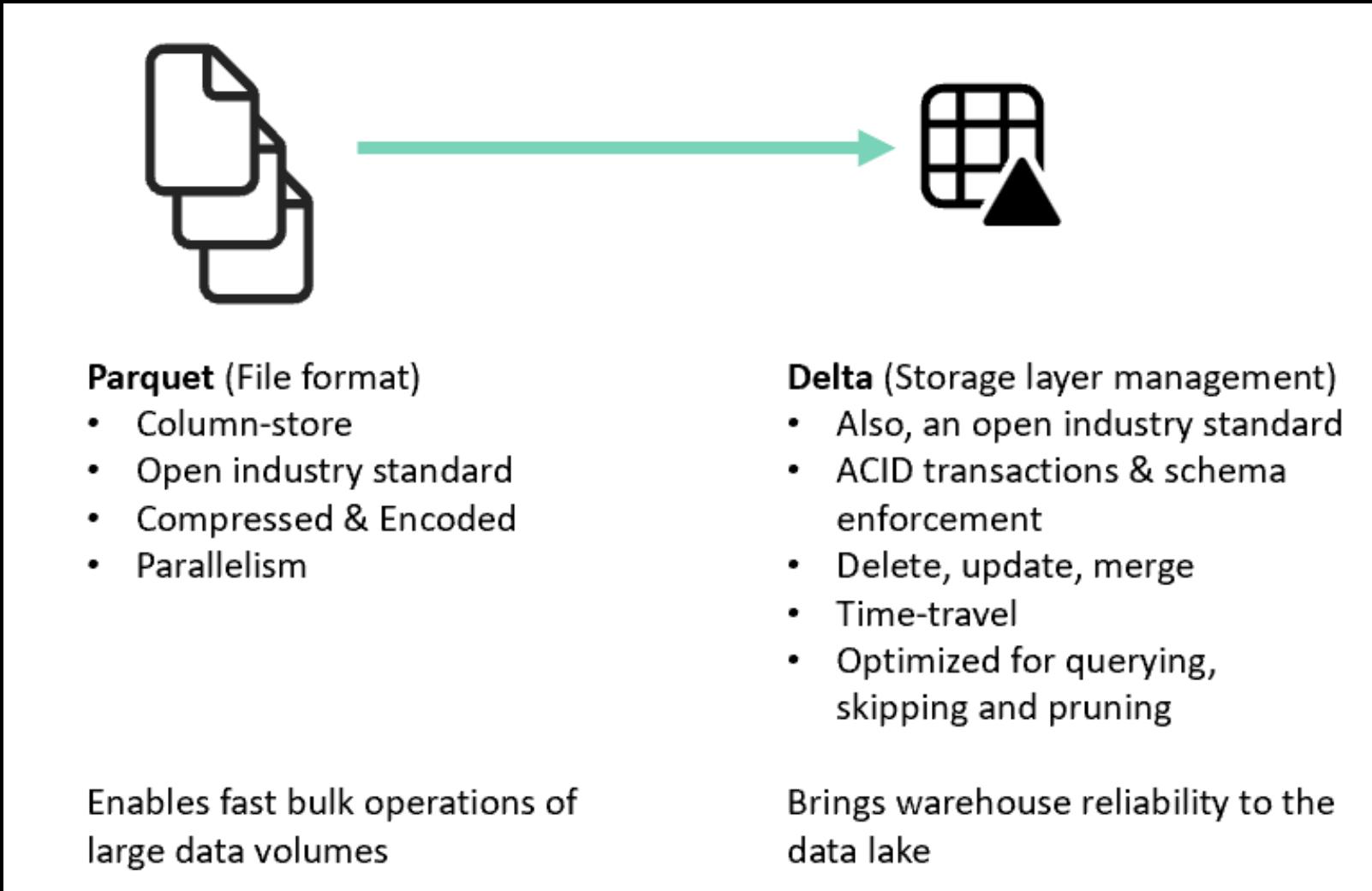
Demo





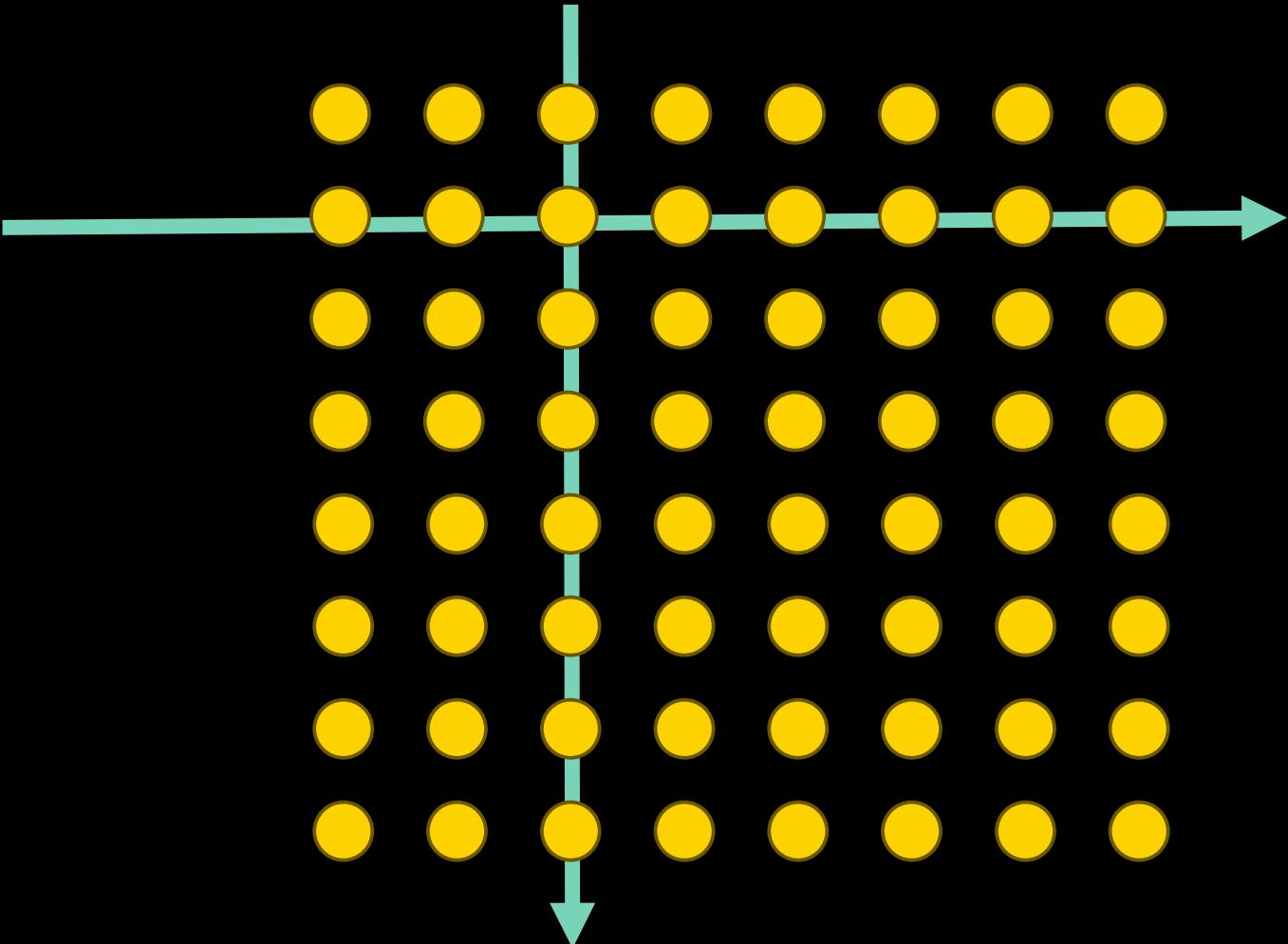
# Internals & performance

# Delta (Parquet)



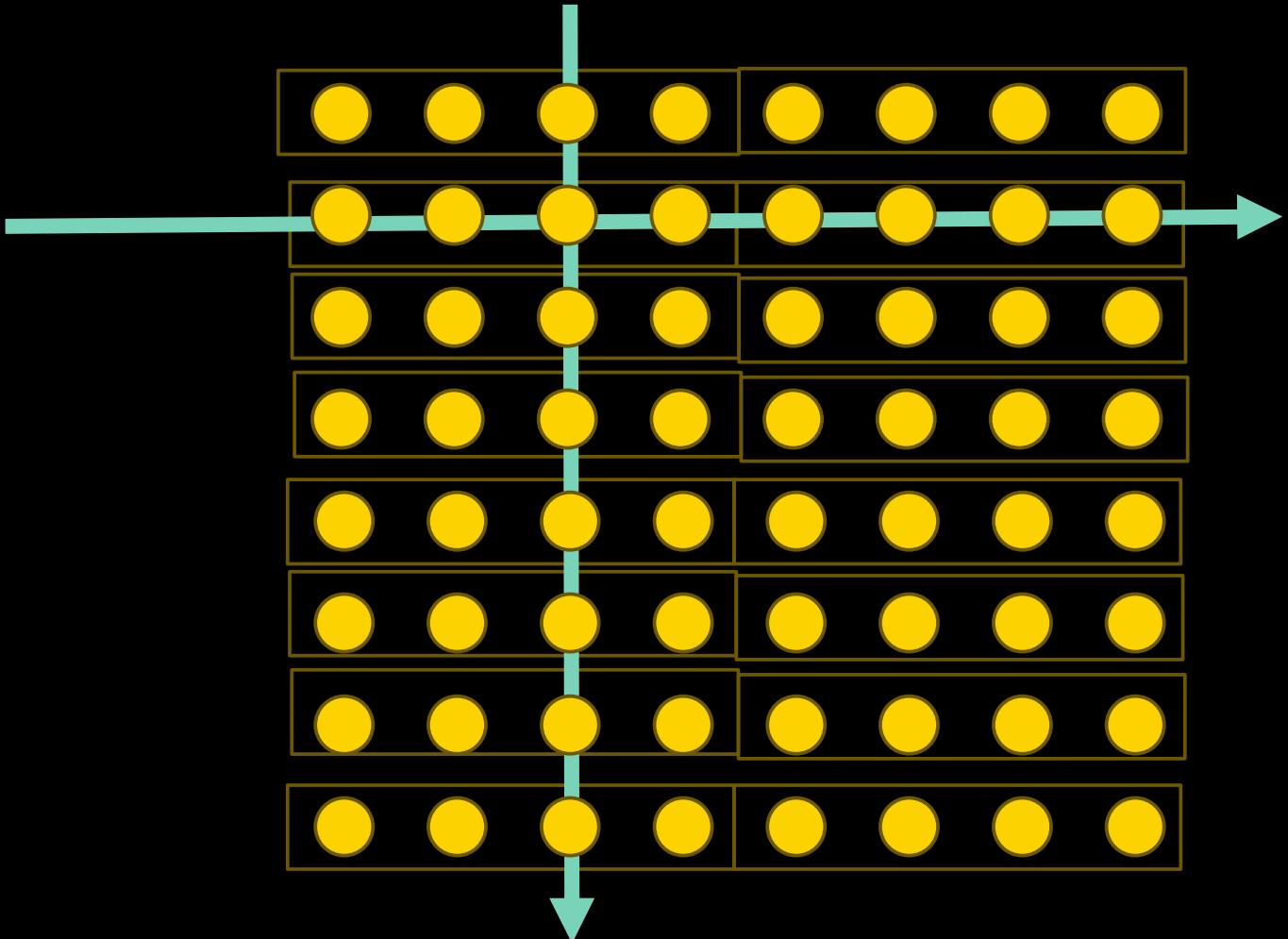
# From Z-order to V-order

SELECT \* FROM points WHERE x=3 or y=2



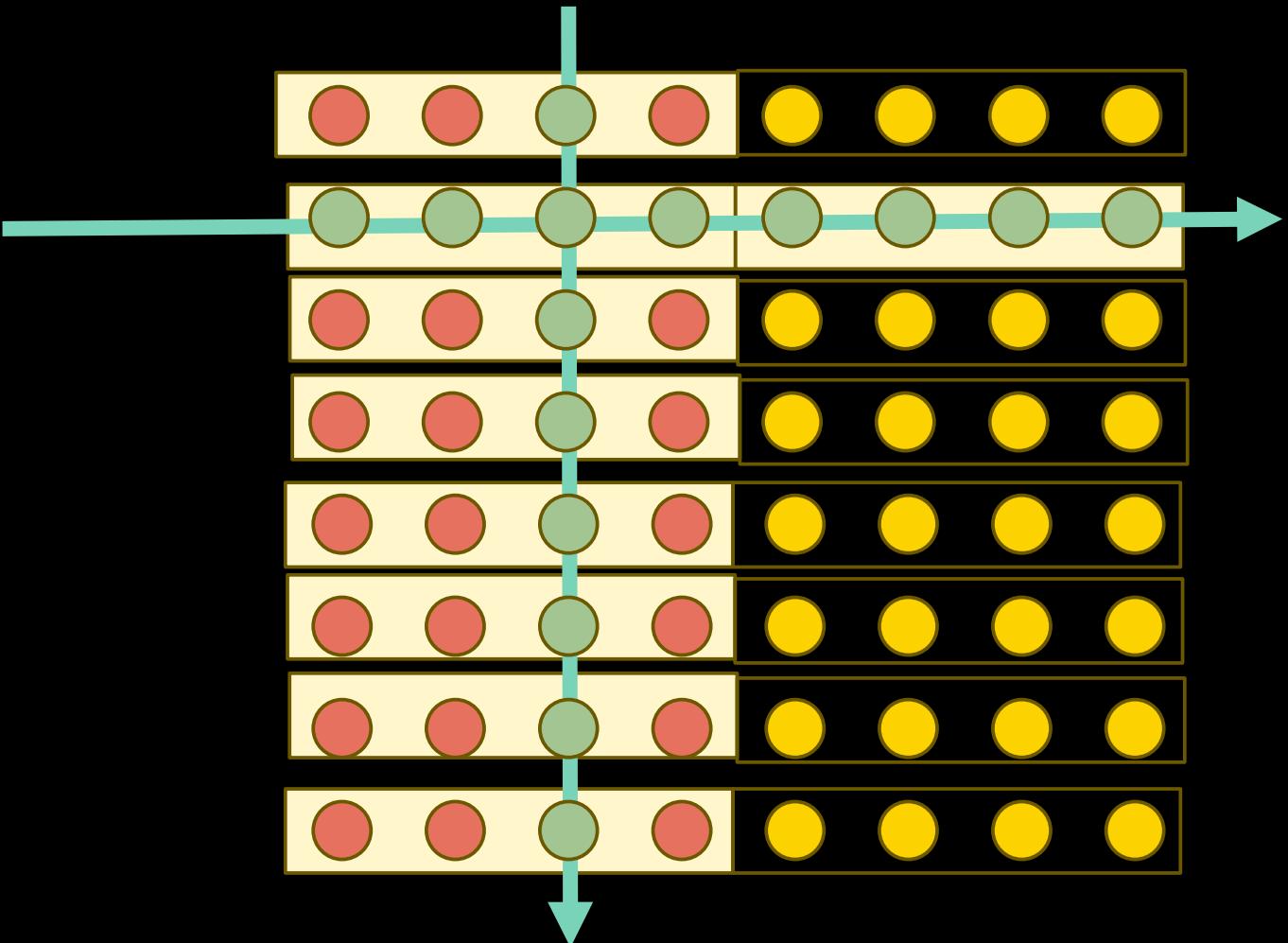
# From Z-order to V-order

SELECT \* FROM points WHERE x=3 or y=2



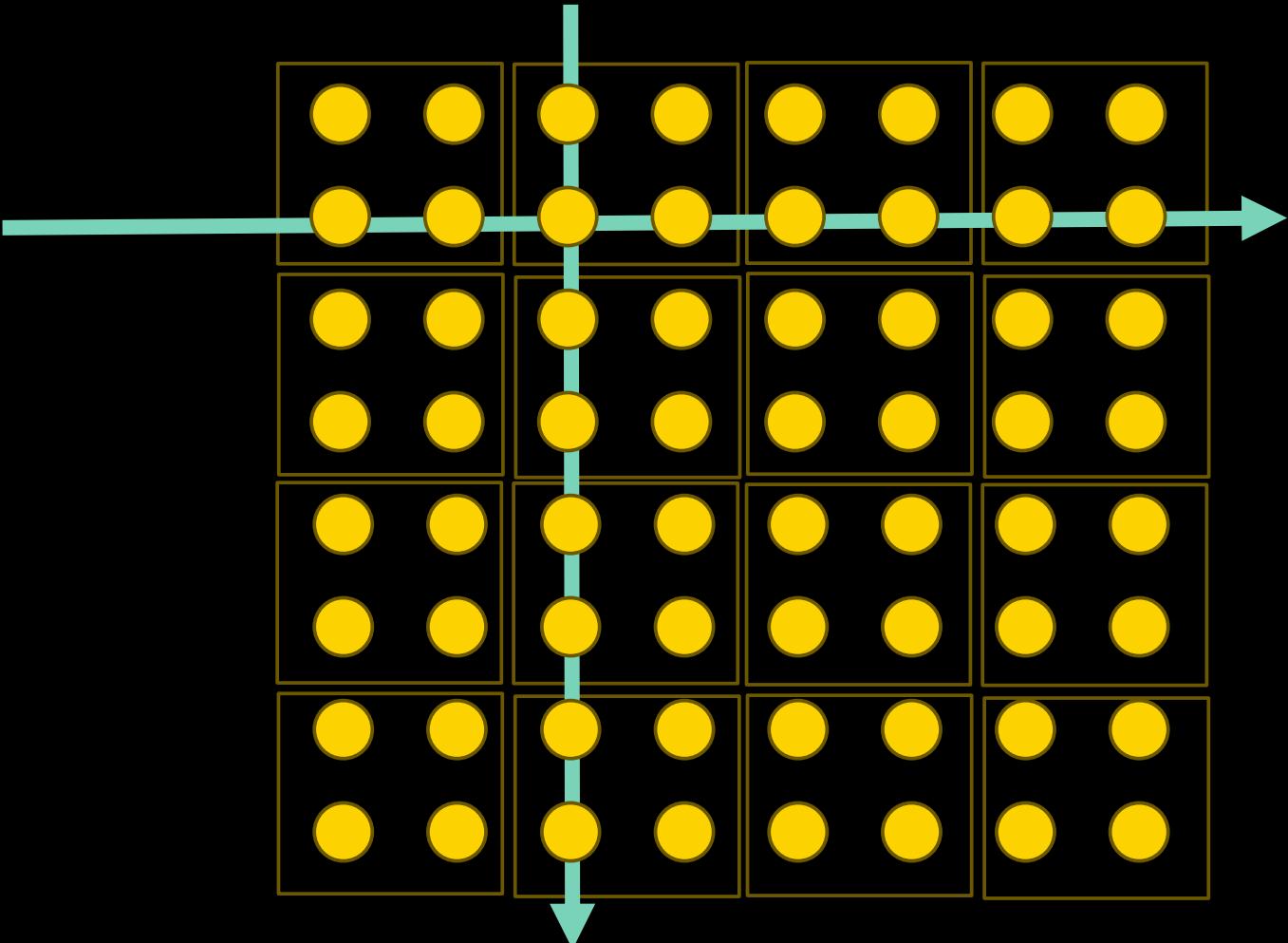
# From Z-order to V-order

SELECT \* FROM points WHERE x=3 or y=2



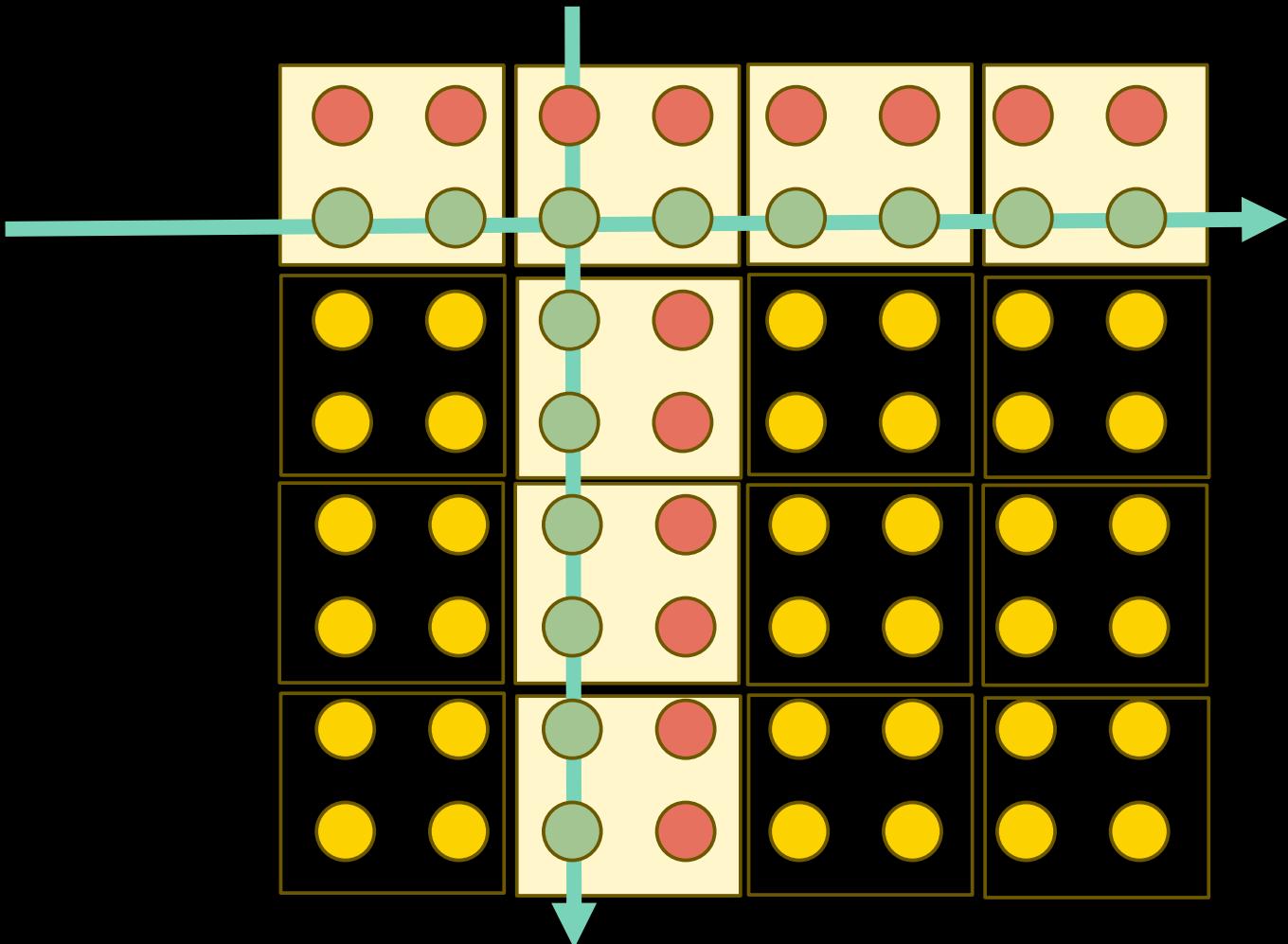
# From Z-order to V-order

SELECT \* FROM points WHERE x=3 or y=2



# From Z-order to V-order

SELECT \* FROM points WHERE x=3 or y=2



# From Z-order to V-order

Yellow taxi (3 Billion rows)



416 GB



164 GB



V-order  
60GB

x3.2  
Less I/O for all ★  
workloads

# Data saving

The Analysis Services column-oriented storage using Delta Lake/Parquet open standard for Direct Lake

## SSAS, AAS, Power BI large models

A screenshot of a Windows File Explorer window. The address bar shows the path: C:\Program Files\Microsoft SQL Server\MSAS15.MSSQLSERVER\OLAP\Data\AdventureWorksTabular.0.db\DimCustomer (10).tbl\238.prt. The main pane displays a list of files:

Name	Date modified	Type	Size
1.DimCustomer (10).AddressLine1 (78).0.idf	1/29/2020 6:36 PM	IDF File	37 KB
1.DimCustomer (10).AddressLine1 (78).0.idfmeta	1/29/2020 6:36 PM	IDFMETA File	1 KB
1.DimCustomer (10).AddressLine2 (79).0.idf	1/29/2020 6:36 PM		
1.DimCustomer (10).AddressLine2 (79).0.idfmeta	1/29/2020 6:36 PM		
1.DimCustomer (10).BirthDate (66).0.idf	1/29/2020 6:36 PM		
1.DimCustomer (10).BirthDate (66).0.idfmeta	1/29/2020 6:36 PM		
1.DimCustomer (10).CommuteDistance (82).0.idf	1/29/2020 6:36 PM		
1.DimCustomer (10).CommuteDistance (82).0.idfmeta	1/29/2020 6:36 PM		

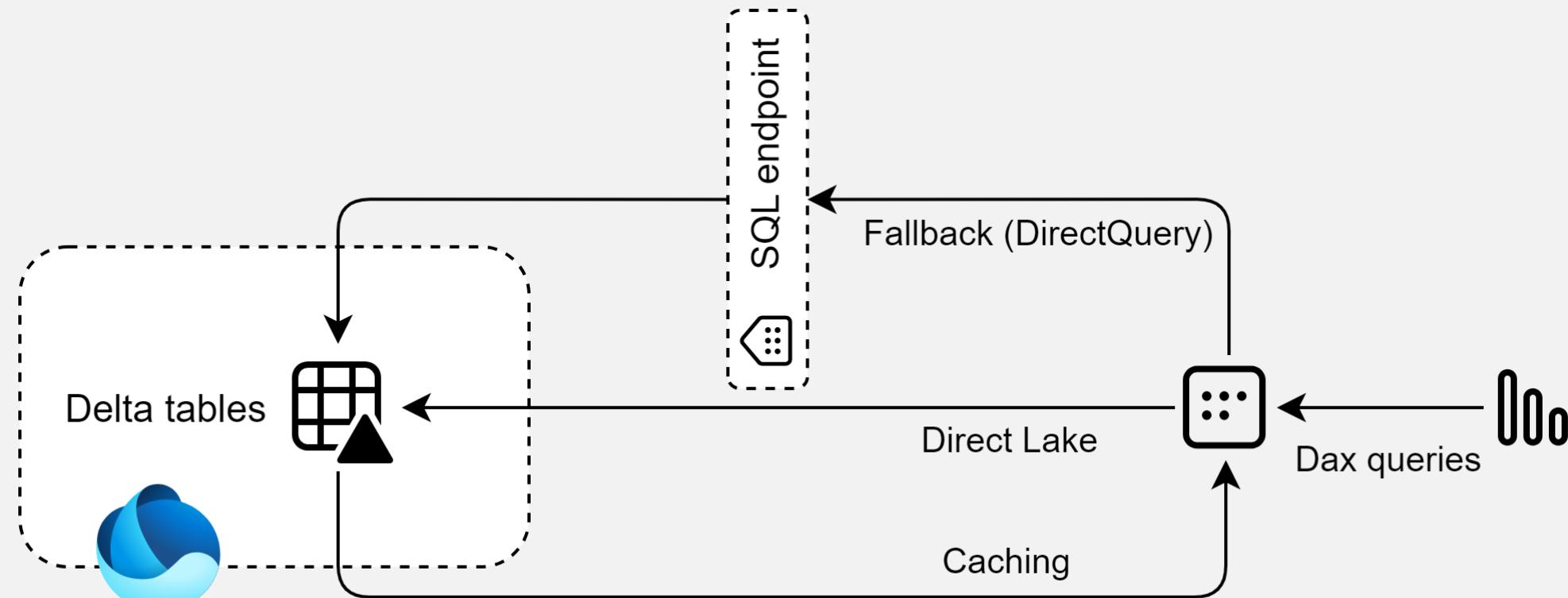
To the right of the file list, there is a navigation pane titled "Finance". It includes a "Lake view" tab (which is selected) and a "Table view" tab. Under "Tables", there is a list of tables: ACR Adjustment, Adjustment Type, Budget, Business, Calendar, Consumed Revenue, Forecast, Forecast Type, Future Flag, and GCMO.

Below the navigation pane, there is a list of files under the "Revenue" folder:

Name	Size	Type
_delta_log	2 items	Folder
part-00000-87858576-90b7-4aff-8c9e-69dcc52db1	8.4 GB	PARQUET
part-00001-631fb085-0591-46b8-a0b5-0fec8f2255	8.4 GB	PARQUET
part-00002-0469bb29-daa5-4ecd-a3ee-bb90331a6	8.4 GB	PARQUET
part-00003-27e6062b-4d55-4469-b285-7cb2a2f32	8.4 GB	PARQUET
part-00004-b12eea8e-f255-41fa-a943-a78def57ce	8.4 GB	PARQUET

A large yellow arrow points from the left side of the slide towards the "Revenue" folder list.

# Fallback & Caching



# Fallback

## When could fallback to DirectQuery happen?

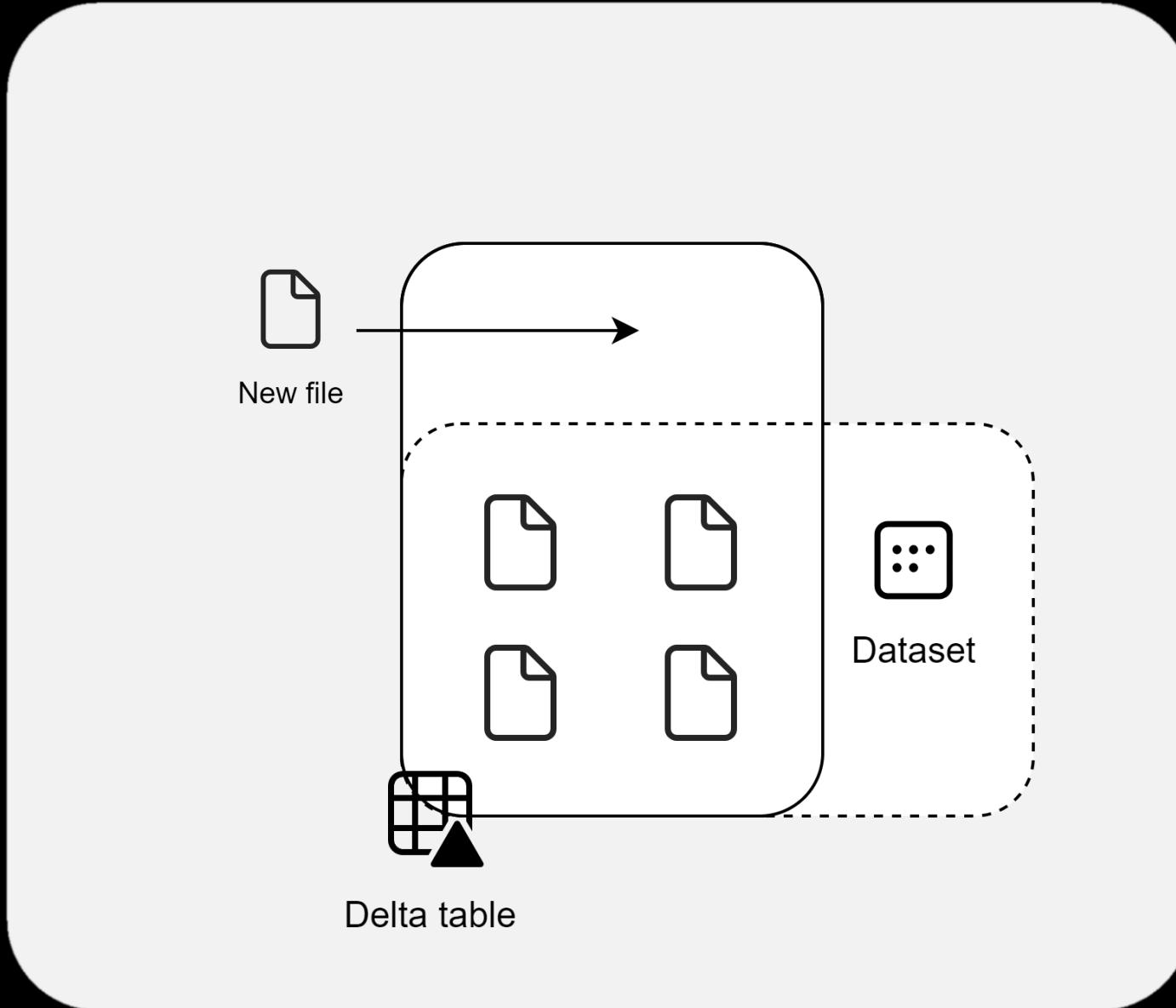
- Special data types
- Large data volumes that does not fit the capacity size
- Composite models
- When you manually configure security  
*Item level on lakehouse*

# Introducing Framing

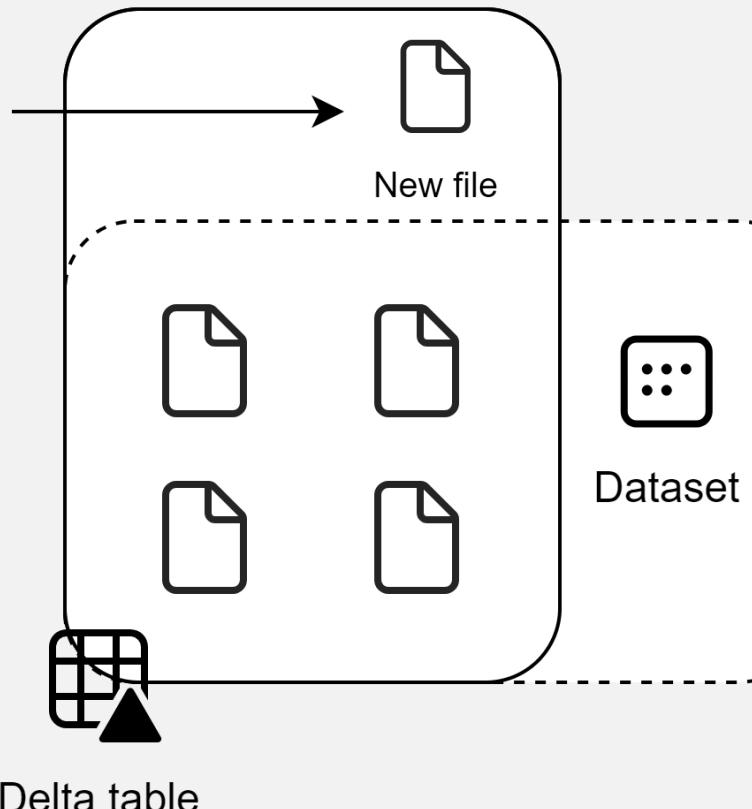
Metadata **refresh** which does not actually load the data, but only the delta table definitions.



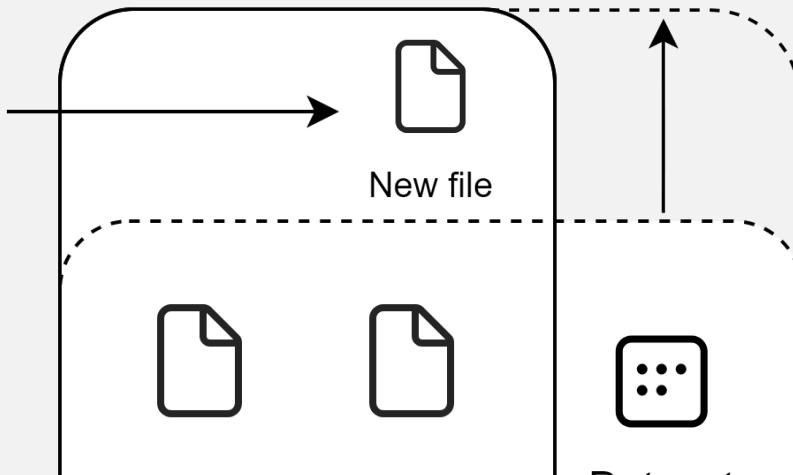
# Framing



# Framing



# Framing



## Refresh

Keep your Direct Lake data up to date

Configure Power BI to detect changes to the data in OneLake and automatically update the Direct Lake tables that are included in this dataset. [Learn more](#)

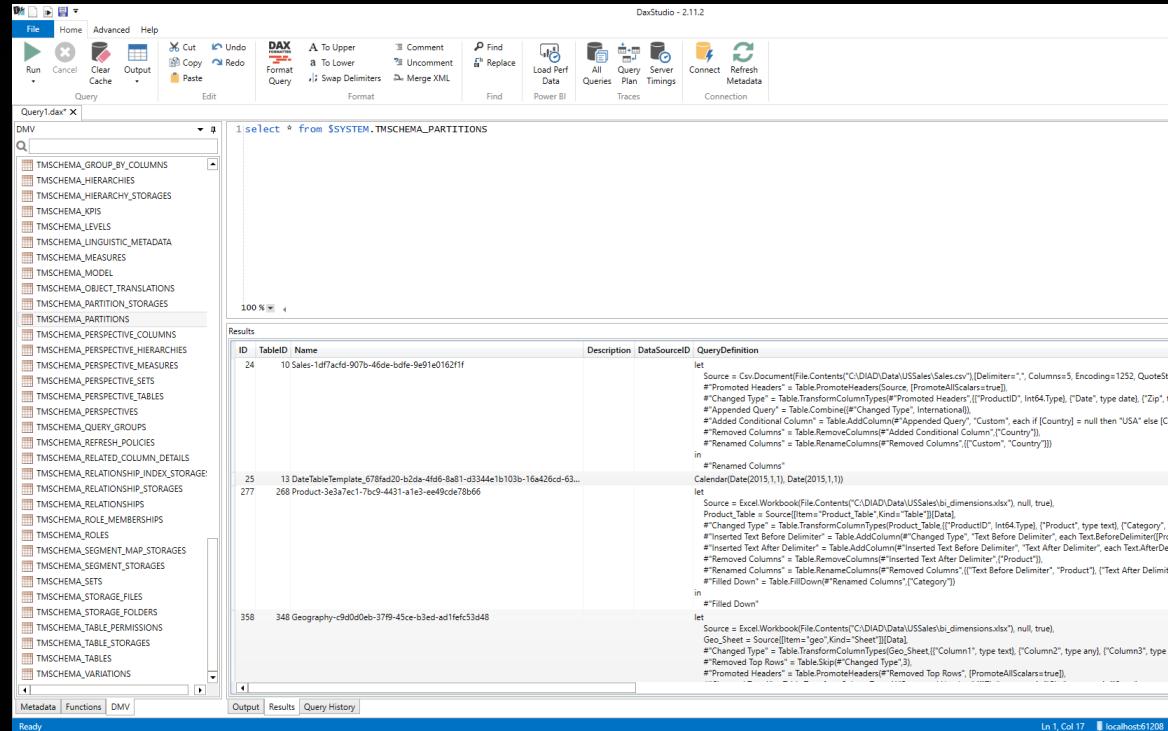


Off

# Dynamic Management Views

Analysis Services Dynamic Management Views (DMVs) are queries that return information about model objects, server operations, and server health.

- DB Schema = Database model
- DISCOVER = Operations & Sessions
- TM Schema = Tabular = Power BI / AAS
- MD Schema = MDX = Multidimensional



The screenshot shows the DaxStudio interface with a query window titled "Query1.dax\*". The query is:

```
1:select * from $SYSTEM.TMSCHEMA_PARTITIONS
```

The results pane displays a table with three rows of data:

ID	TableID	Name	Description	DataSourceID	QueryDefinition
24	10	Sales-1d7acf-907b-46de-bdfc-9e91e0162f1f			let Source = Csv.Document(File.Contents("C:\DIAD\Data\USSales\Sales.csv"),[Delimiters=","], Columns=5, Encoding=1252, QuoteStyle="None");#PivotTable Headers = Table.TransformColumnTypes(Source, [Header=1, MaxColumnWidth=100, MaxRows=100, UseFirstNRowAsSchema=true]);#Changed Row = Table.AddRowColumn(#"PivotTable Headers", "Sales", Int64.Type);#Appended Query = Table.Combine({#"PivotTable Headers", "Sales"});#Added Conditional Column = Table.AddColumn(#"Appended Query", "Country", each if [Country] = null then "USA" else [Country]);#Removed Columns = Table.RemoveColumns(#"Added Conditional Column",["Country"]);#Renamed Columns = Table.RenameColumns(#"Removed Columns", [{"Custom", "Country"}]) in #Renamed Columns#Calender(Date(2015,1,1), Date(2015,1,1))
25	13	DateTableTemplate_678fed20-b2da-4fd6-8a81-d3344e1b103b-16a426cd-63...			let Source = Excel.Workbook(File.Contents("C:\DIAD\Data\USSales\bi_dimensions.xlsx"), null, true);#PivotTable Headers = Table.TransformColumnTypes(Source, [Header=1, MaxColumnWidth=100, MaxRows=100, UseFirstNRowAsSchema=true]);#Changed Row = Table.AddRowColumn(#"PivotTable Headers", "Product", Int64.Type);#Appended Query = Table.Combine({#"PivotTable Headers", "Product"});#Added Conditional Column = Table.AddColumn(#"Appended Query", "Category", each Text.BeforeDelimiter([Product], "-") & Text.AfterDelimiter([Product], "-") & "Category");#Removed Columns = Table.RemoveColumns(#"Added Conditional Column",["Category"]);#Renamed Columns = Table.RenameColumns(#"Removed Columns", [{"Text Before Delimiter", "Product"}, {"Text After Delimiter", "Category"}]) in #Renamed Columns#Filled Down = Table.FillDown(#"Renamed Columns",["Category"])
277	268	Product-3e3a7ec1-7bc9-4431-a1e3-eed49cde78b66			let Source = Excel.Workbook(File.Contents("C:\DIAD\Data\USSales\bi_dimensions.xlsx"), null, true);#PivotTable Headers = Table.TransformColumnTypes(Source, [Header=1, MaxColumnWidth=100, MaxRows=100, UseFirstNRowAsSchema=true]);#Changed Row = Table.AddRowColumn(#"PivotTable Headers", "Product", Int64.Type);#Appended Query = Table.Combine({#"PivotTable Headers", "Product"});#Added Conditional Column = Table.AddColumn(#"Appended Query", "Category", each Text.BeforeDelimiter([Product], "-") & Text.AfterDelimiter([Product], "-") & "Category");#Removed Columns = Table.RemoveColumns(#"Added Conditional Column",["Category"]);#Renamed Columns = Table.RenameColumns(#"Removed Columns", [{"Text Before Delimiter", "Product"}, {"Text After Delimiter", "Category"}]) in #Renamed Columns#Filled Down = Table.FillDown(#"Renamed Columns",["Category"])
358	348	Geography-c9d0d0eb-37ff-45ce-b3ed-ad1fef53d48			let Source = Excel.Workbook(File.Contents("C:\DIAD\Data\USSales\bi_dimensions.xlsx"), null, true);#PivotTable Headers = Table.TransformColumnTypes(Source, [Header=1, MaxColumnWidth=100, MaxRows=100, UseFirstNRowAsSchema=true]);#Changed Row = Table.AddRowColumn(#"PivotTable Headers", "Geo_Sheet", Int64.Type);#Appended Query = Table.Combine({#"PivotTable Headers", "Geo_Sheet"});#Added Conditional Column = Table.AddColumn(#"Appended Query", "Column1", type text, ["Column1", type any, {"Column1", type any}]);#Removed Top Rows = Table.Skip(#"Added Conditional Column", 3);#Promoted Headers = Table.PromoteHeaders(#"Removed Top Rows", [PromoteAllScalars=true])

Demo





# Advanced patterns

# XMLA

Talk with the back-end server, just like an Analysis Services server (read / write)

The screenshot shows the 'Server settings' section of Power BI Desktop. Under 'Connection string', the URL is set to 'Data Source=powerbi://api.powerbi.com/v1.0/myorg/Dive%20into%20'. A green 'Copy' button is visible below the text input. To the right, there's a 'Read Write' dropdown menu with 'Read Write' selected, and 'Apply' and 'Discard' buttons below it.

**⊗ Caution**

At this time, a write operation on a dataset authored in Power BI Desktop will prevent it from being downloaded back as a PBIX file. Be sure to retain your original PBIX file.

# Introducing calculation groups

## Benefits

- Reduce the number of redundant measures and grouping common measure expressions as calculation items
- Avoids duplicating logic in different measures
- Typical use cases are
  - Time-intelligence calculations (YTD / QTD / MTD / ...)
  - Format string change, like currency conversions



## Limitations

- Can only be created from external tools in Power BI (Any tool using the XMLA endpoint such as Tabular Editor) – **but stay tuned....**
- Object level security on Calculation group items is not supported
- Smart narrative visuals in Power BI are not supported with Calculation Groups

# Introducing calculation groups

## Specific DAX expressions for Calculation Groups

- SELECTEDMEASURE()
- SELECTEDMEASURENAME()
- ISSELECTEDMEASURE()
- SELECTEDMEASUREFORMATSTRING()

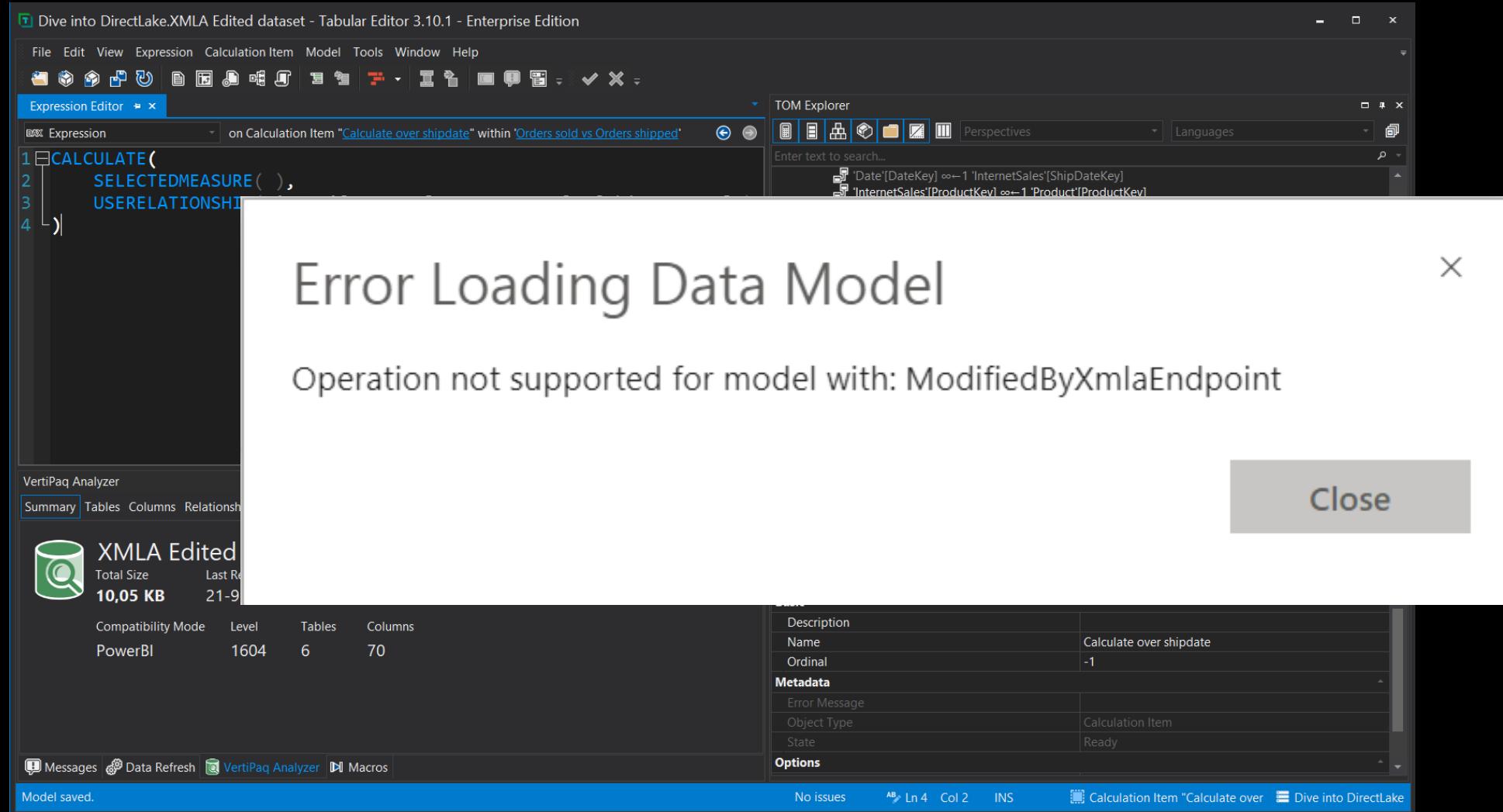
Classic measure:

```
MTD =  
CALCULATE (  
    SUM ( Sales[SalesAmount] ),  
    DATESMTD ( DimDate[Date] )  
)
```

Dynamic measure context MTD with Calculation Group:

```
MTD =  
CALCULATE (  
    SELECTEDMEASURE (),  
    DATESMTD ( DimDate[Date] )  
)
```

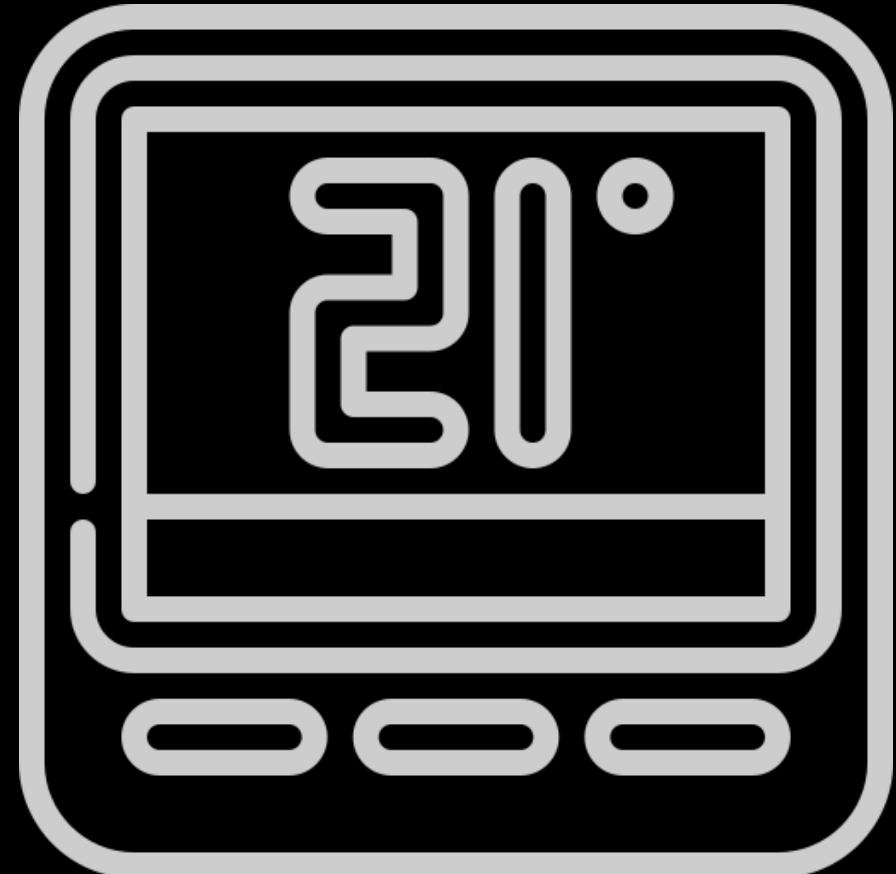
# Creating calculation groups over XMLA



# Temperature management

## Keep it WARM!

Make sure your users are served optimally  
and avoid the capacity memory to be flushed.



# Eviction

Capacity: F64 / P1

Memory: 25 GB

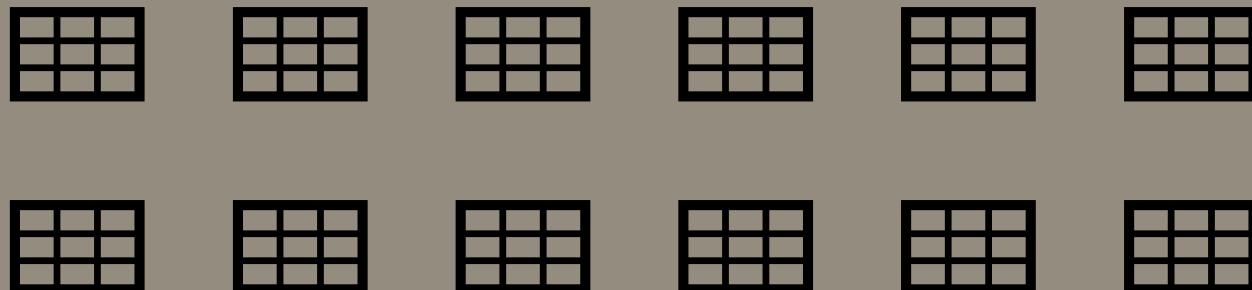
Capacity utilization:

**Cool**

Active memory



Storage



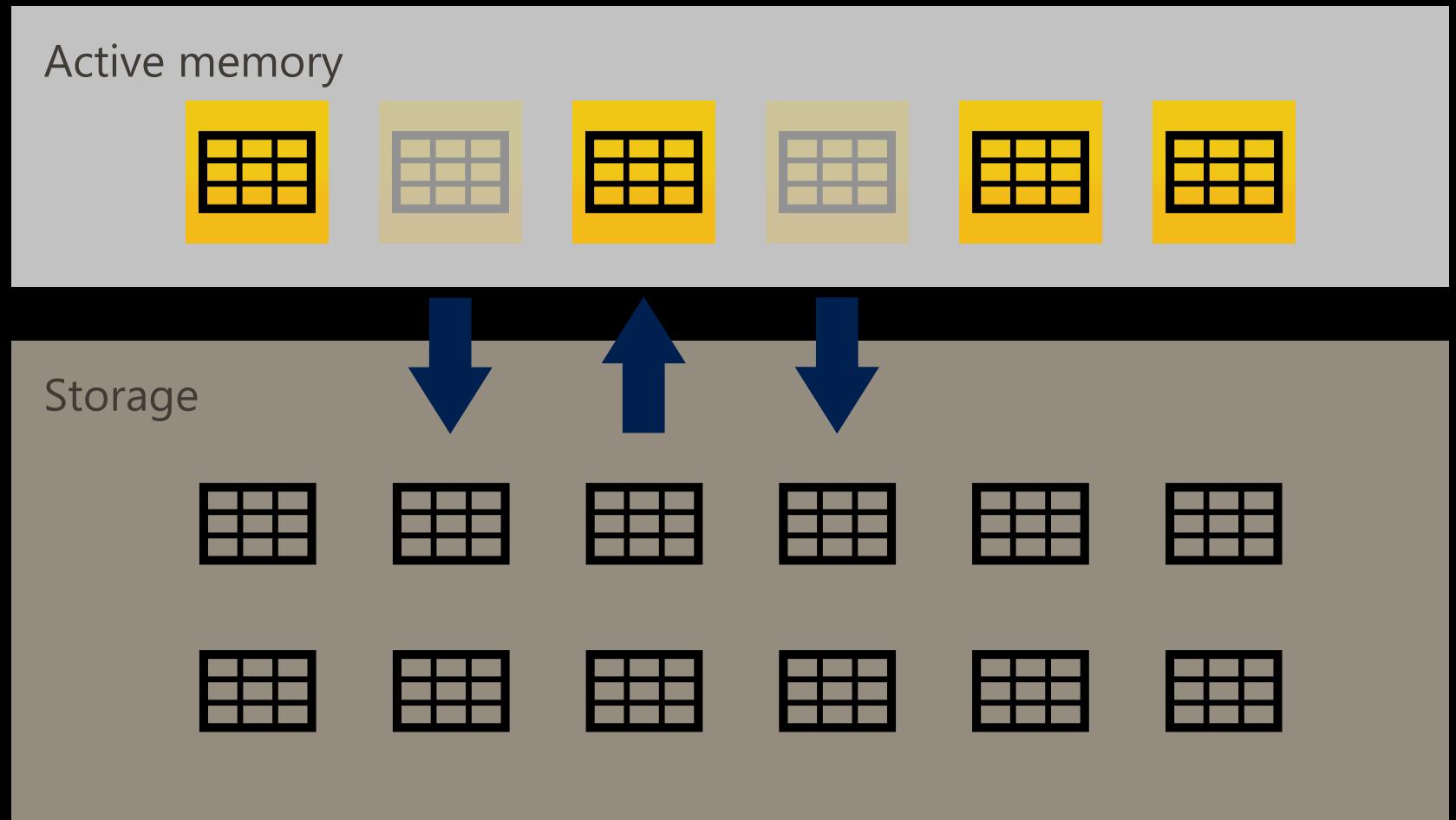
# Eviction

Capacity: F64 / P1

Memory: 25 GB

Capacity utilization:

**Warm**



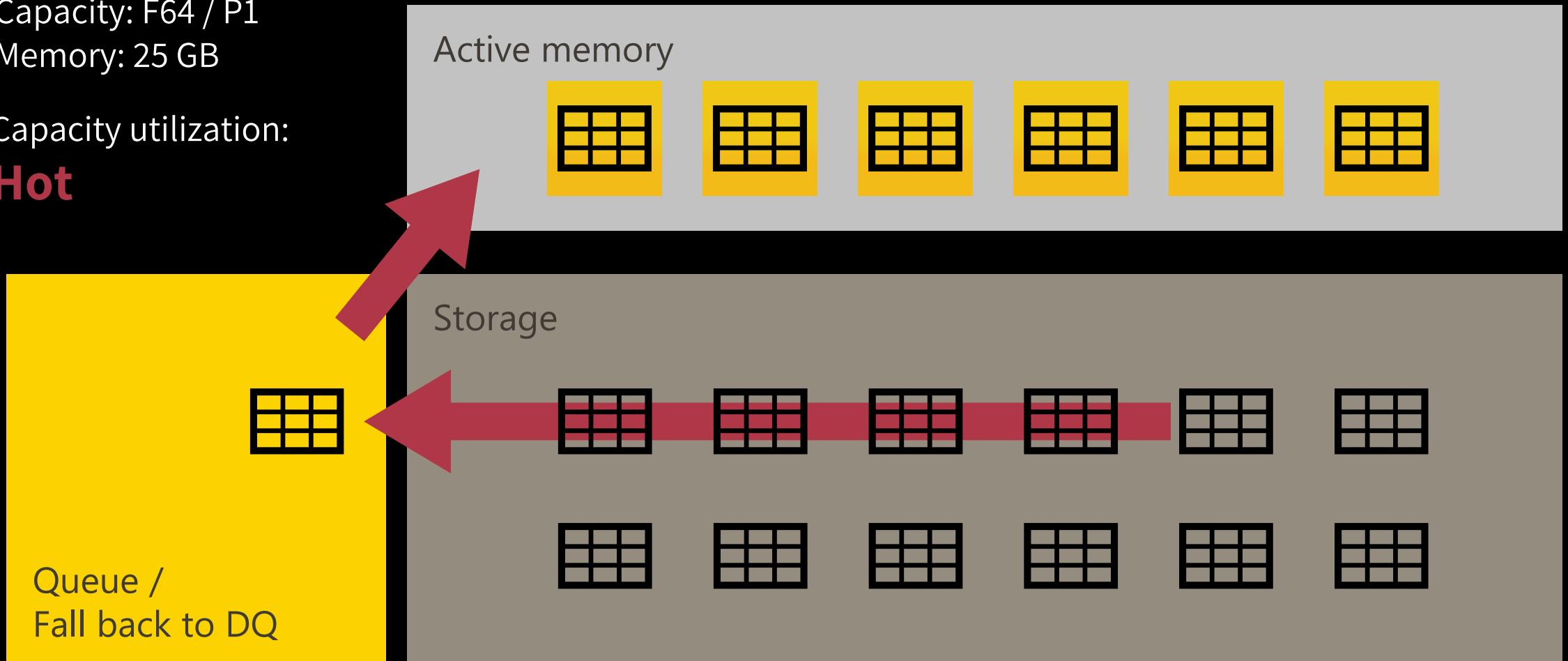
# Eviction - queue / fall-back

Capacity: F64 / P1

Memory: 25 GB

Capacity utilization:

**Hot**



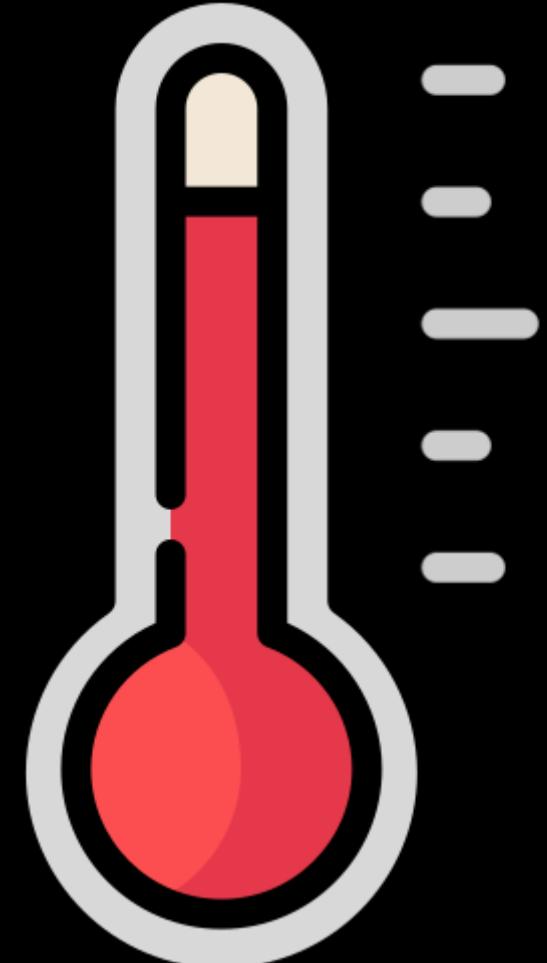
# Temperature management

## What will be evicted?

Basically, your data will be evicted from active memory, that you want to always have available!

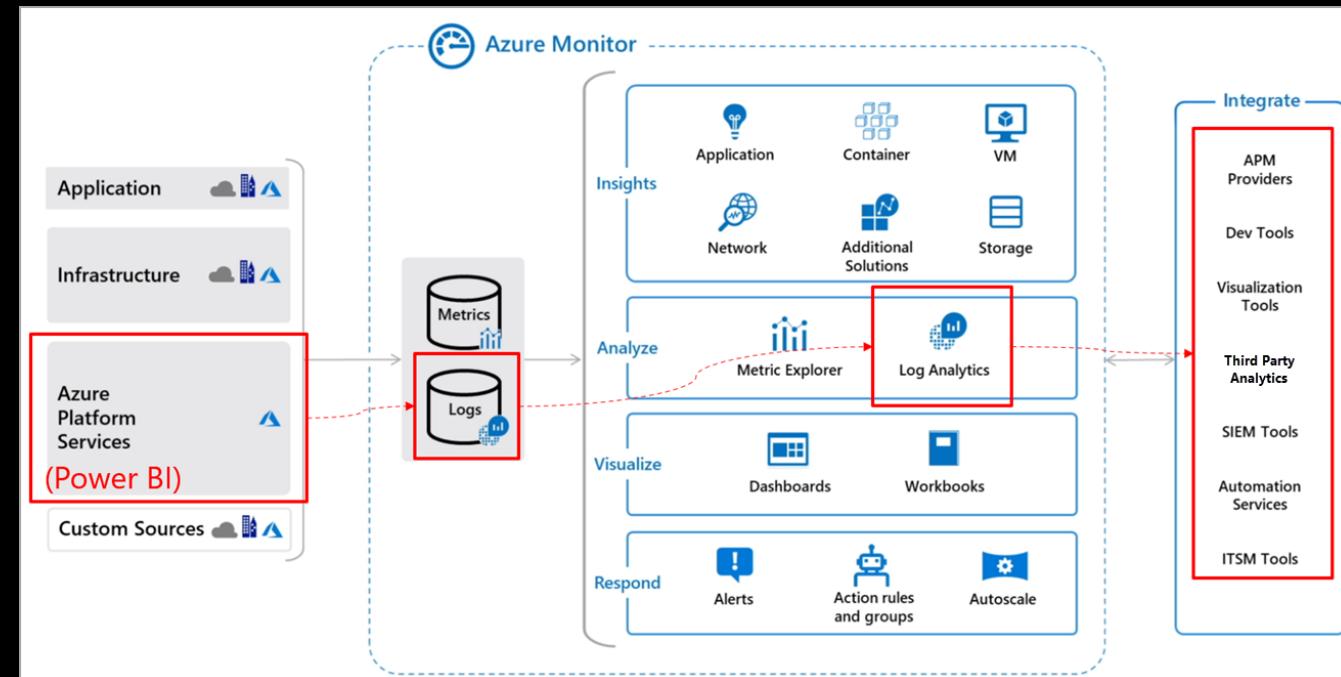
## How can you influence that?

Consider setting up a process (notebook, other automated setup) to pro-actively execute queries to keep certain data **WARM**!



# What should stay in memory?

Azure Monitor delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources they depend on.



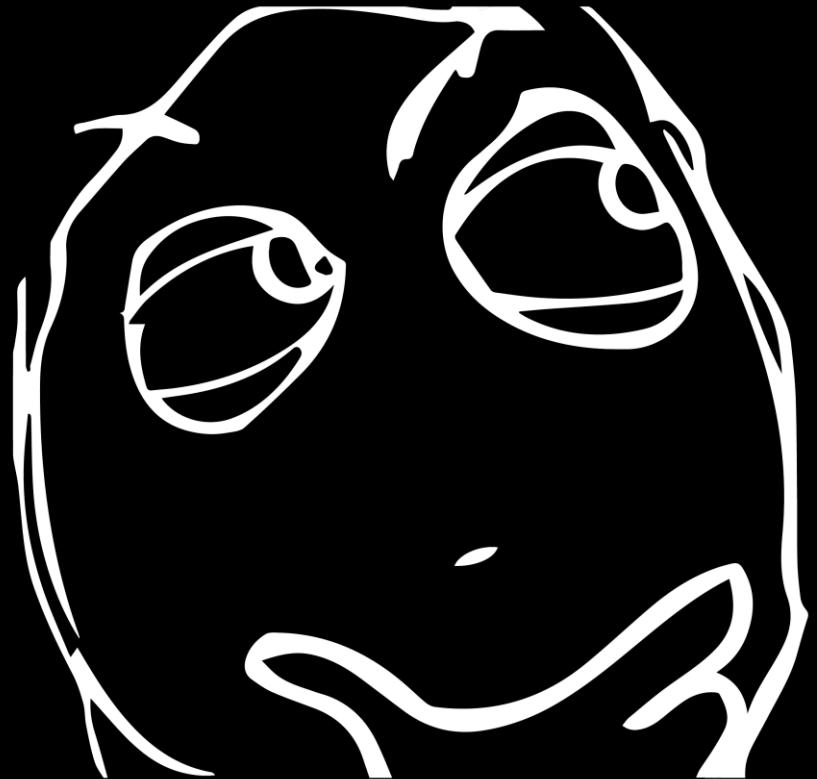
# Wrap up

LET'S  
RECAP...

## Direct Lake...

- Only applicable when using MS Fabric
- No data is imported / copied
- On-demand loading
- Reads data from Lake / Parquet format – Delta is a must
- Performance dependent on capacity size/utilization
- Falls back to DirectQuery when limitations are hit!
- Consider implement advanced patterns for specific use cases

# Considerations



IT DEPENDS

## Should I change all my solutions to start using Direct Lake?

- There is no OneSecurity yet  
*RLS / OLS on dataset level is possible*
- Consider impact on capacities when falling back to DQ
- Performance is better than DQ
- It is in public preview

# Resources

## **Direct Lake generic documentation**

<https://learn.microsoft.com/en-us/power-bi/enterprise/Direct-Lake-overview>

## **Calculation groups for Direct Lake datasets**

<https://powerbi.microsoft.com/en-us/blog/announcing-calculation-groups-for-direct-lake-datasets/>

## **Analyze performance for Direct Lake**

<https://learn.microsoft.com/en-us/power-bi/enterprise/directlake-analyze-qp>

## **On-demand loading of Direct Lake Power BI datasets in Fabric**

<https://blog.crossjoin.co.uk/2023/07/02/on-demand-loading-of-direct-lake-power-bi-datasets-in-fabric/>

## **Direct Lake Frequently Asked Questions**

<https://fabric.guru/power-bi-direct-lake-mode-frequently-asked-questions/>

Big thanks to Benni and Just for helping us and reference materials ☺

# Feedback

