# Using Lakehouse data at scale with Power BI. Featuring Power BI Direct Lake mode!

Benni De Jagere

Slides

CAT

Microsoft

# Benni De Jagere

**Principal Program Manager | Fabric Customer Advisory Team ( FabricCAT )**

Fabric CAT

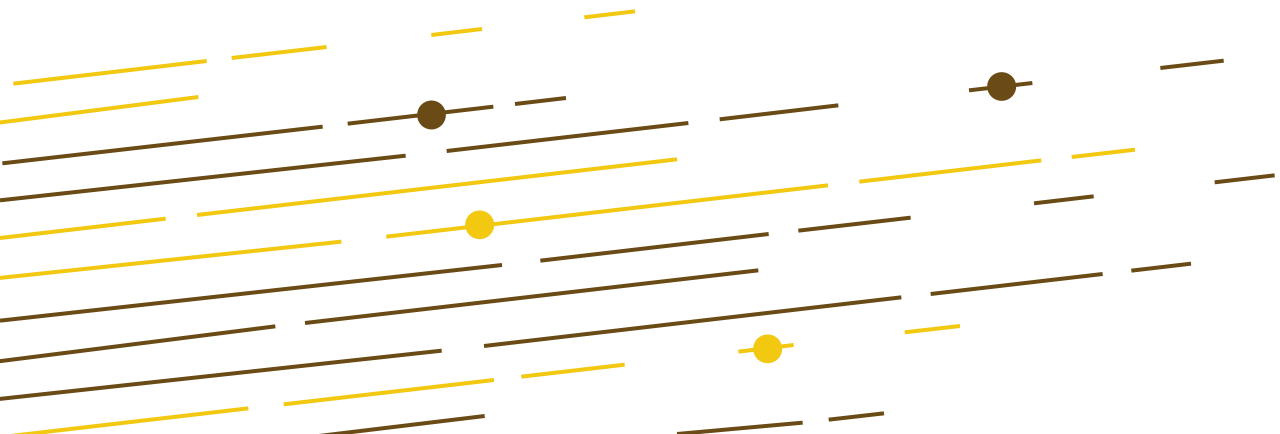
dataMinds .be Member

/bennidejagere

sessionize /bennidejagere

/bennidejagere

# Session Objectives



# Session Objectives

Introduce Fabric and OneLake

Set the scene for Direct Lake

Take it for spin.. ☺

# Introducing Fabric

# Microsoft Fabric

Data Factory

Real-Time Intelligence

Databases

Analytics

Industry Solutions

Power BI

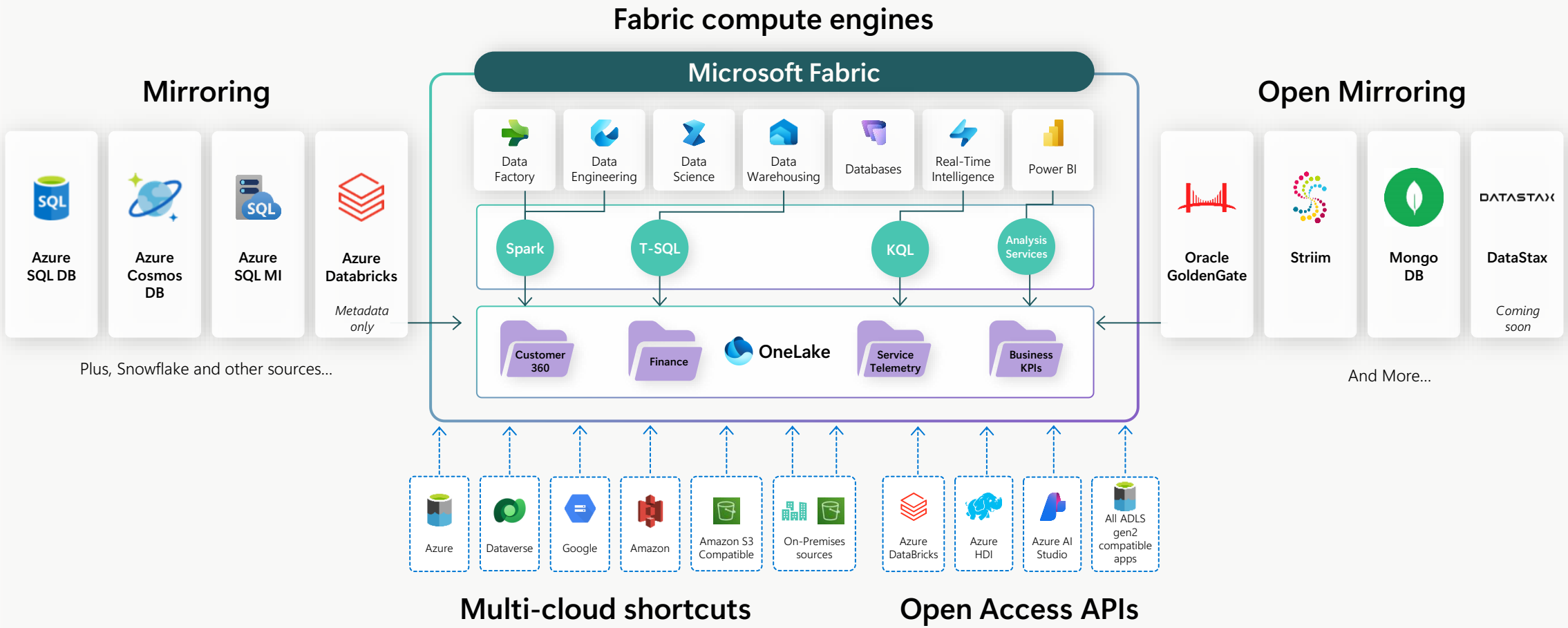
Partner solutions

Copilot in Fabric

OneLake

Microsoft Purview

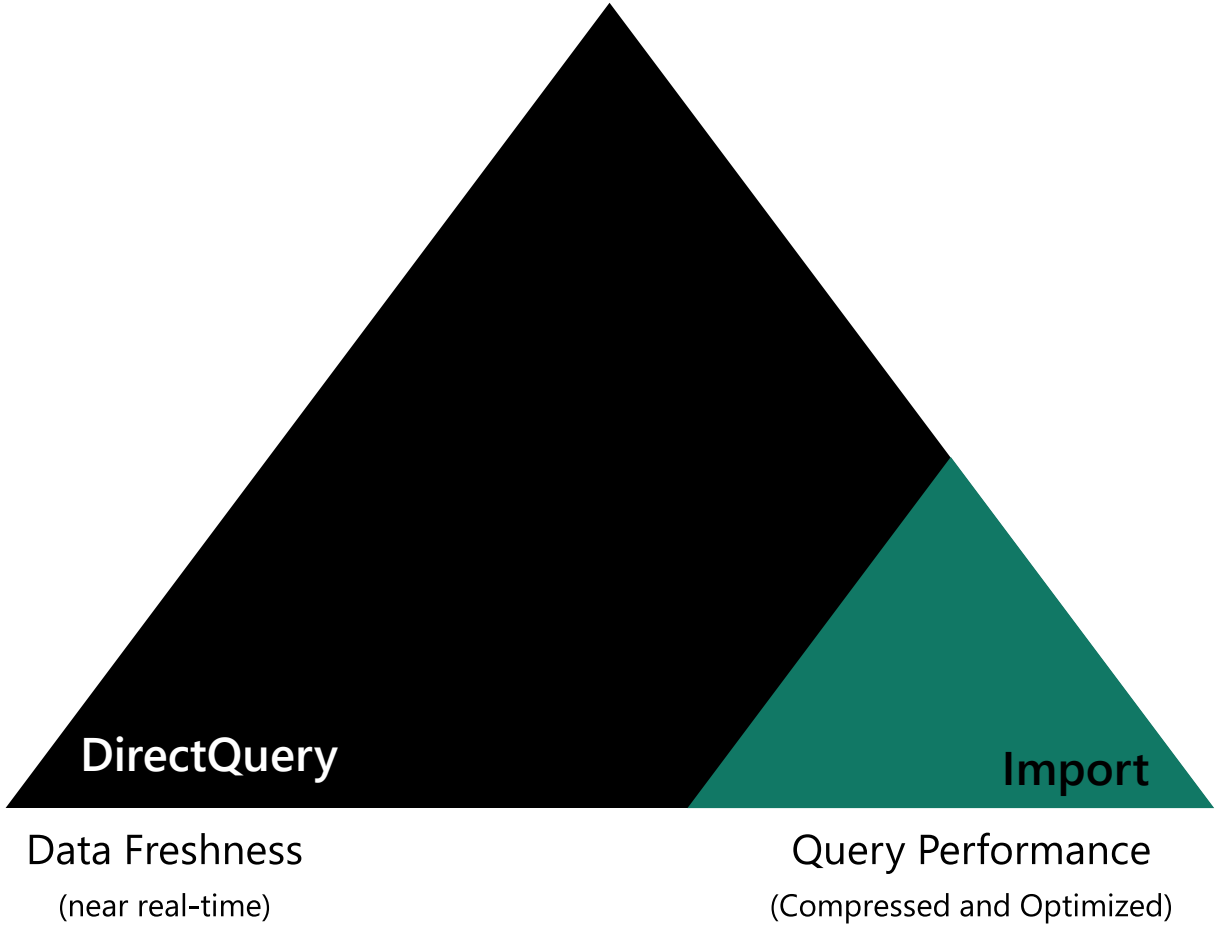
# Bring in data from anywhere



# Storage Modes

**Import:** Caches data into memory to deliver extremely fast performance using the **analysis services** database engine. The default mode when creating a new Power BI Desktop solution along with providing Data Modelers the most design flexibility.

**DirectQuery:** Does not import the data into memory, consists only of the metadata defining the structure. When the model is queried, native queries are used to retrieve data from the underlying data source.




Changing the **Storage mode** of a table to **Import** is an irreversible operation. Once set, this property can't later be changed using Power BI Desktop.

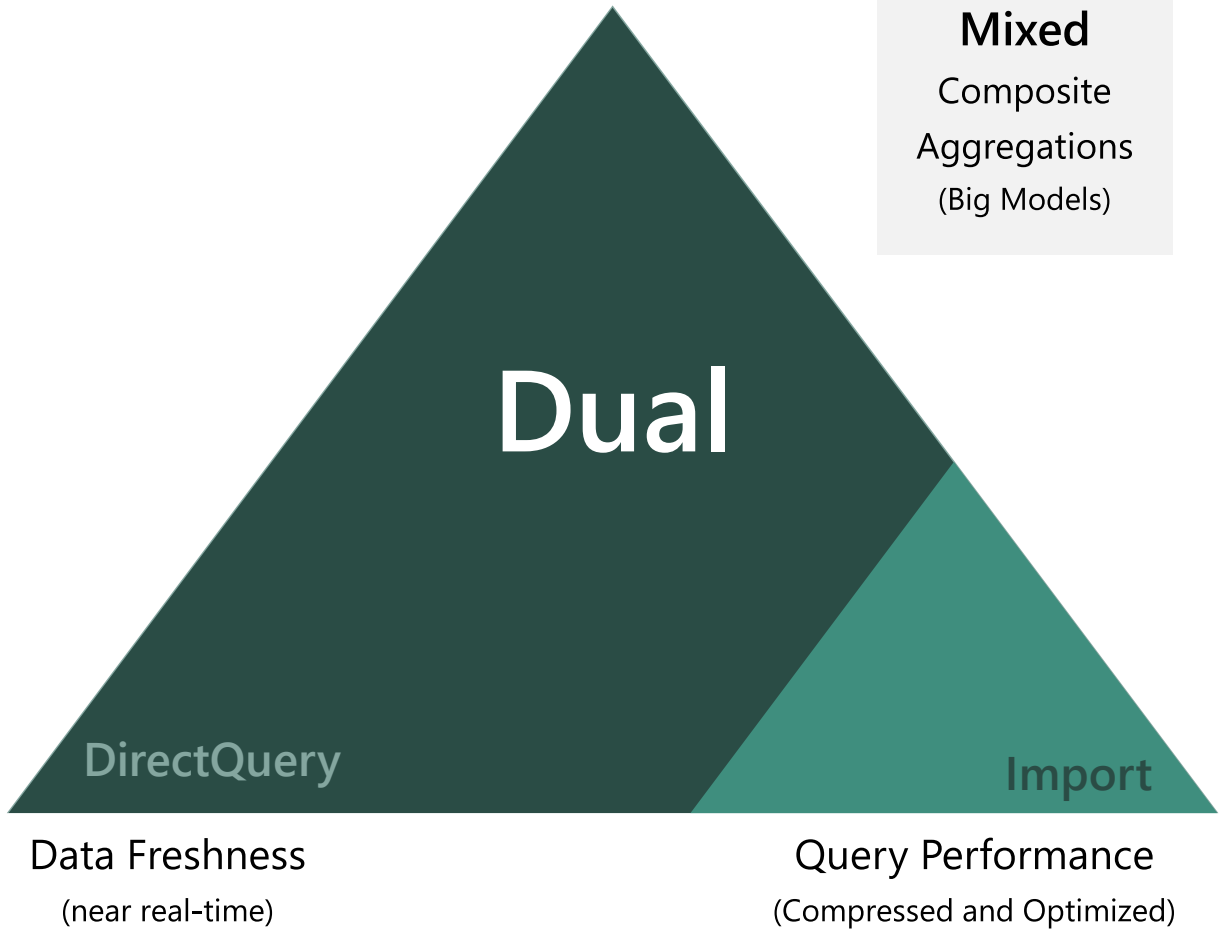
# Storage Modes

**Import:** Caches data into memory to deliver extremely fast performance using the **analysis services** database engine. The default mode when creating a new Power BI Desktop solution along with providing Data Modelers the most design flexibility.

**DirectQuery:** Does not import the data into memory, consists only of the metadata defining the structure. When the model is queried, native queries are used to retrieve data from the underlying data source.

**Dual:** Can act as either cached or not cached, depending on the context of the query that's submitted to the Power BI dataset. In some cases, you fulfill queries from cached data. In other cases, you fulfill queries by executing an on-demand query to the underlying data source.

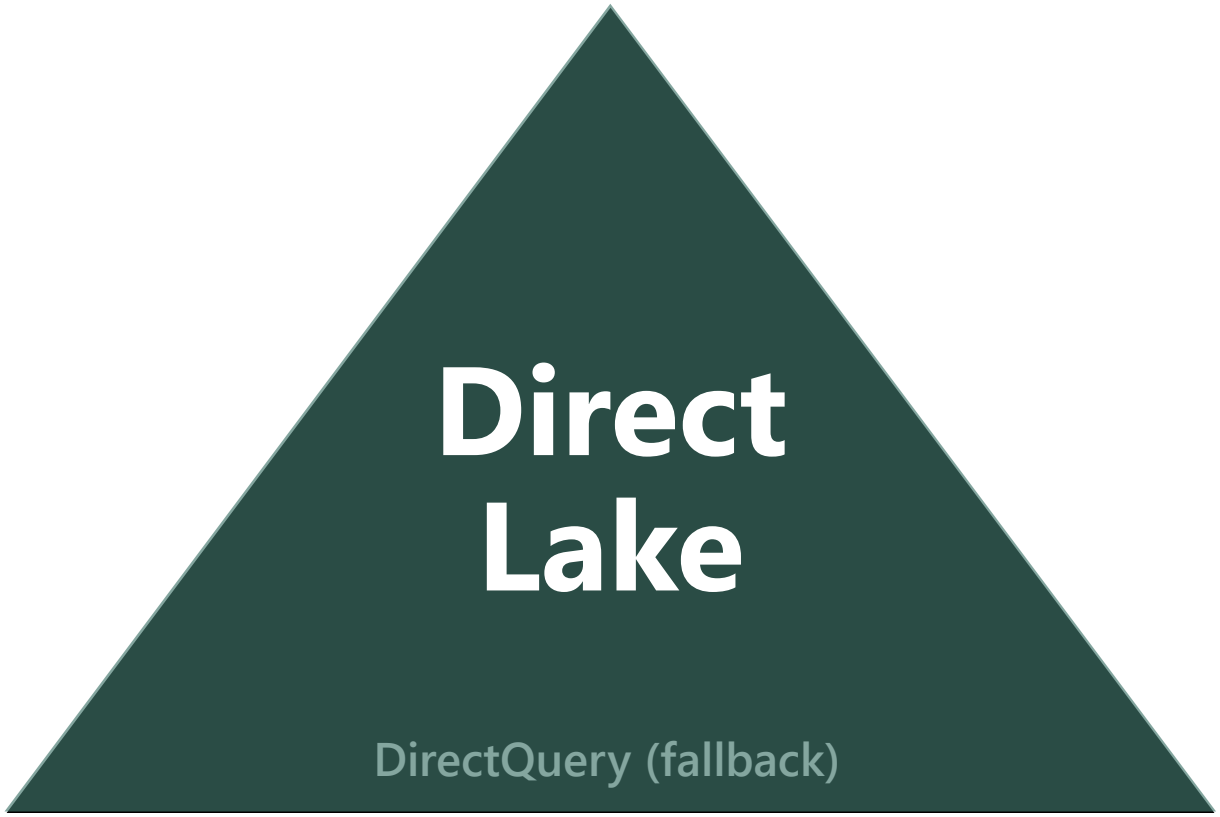
Changing the **Storage mode** of a table to **Import** is an irreversible operation. Once set, this property can't later be changed to either **DirectQuery** or **Dual** using Power BI Desktop.



# Storage Modes

**Direct Lake:** A groundbreaking new dataset capability for analyzing very large data volumes. Based on loading parquet-formatted files directly from a data lake **without having to query a Lakehouse endpoint**, and **without having to import or duplicate data** into a Power BI dataset. Direct Lake is a fast-path to load the data from the lake straight into the Power BI engine, ready for analysis and yielding performance similar to import mode.

**DirectQuery (fallback):** Automatically switches modes- either due to current limitations or based on factors such as available memory in the capacity.

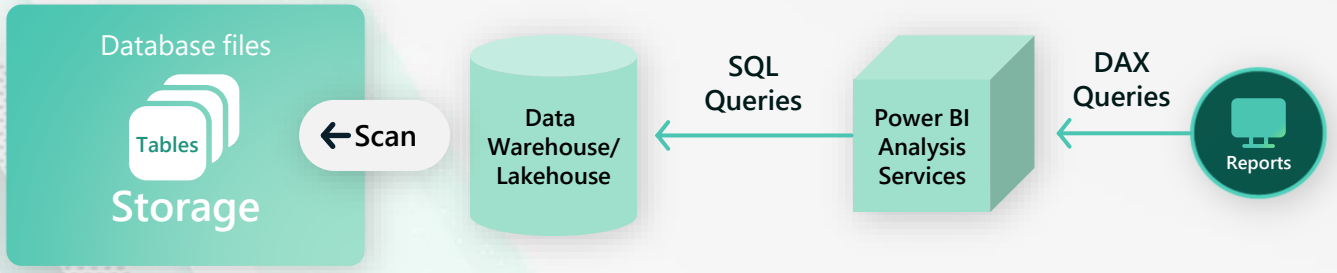


# Power BI | Direct Lake Mode

**Direct Lake** is a fast-path to load the data from the lake straight into the Power BI engine, ready for analysis

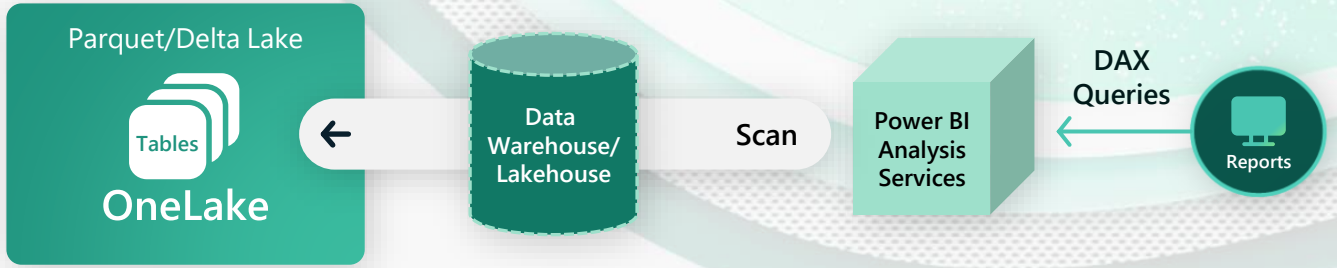
Direct Lake is based on loading parquet-formatted files directly from a data lake without having to query a Lakehouse endpoint, and without having to import or duplicate data into a Power BI dataset

**DirectQuery Mode.** Slower, but real time




**Import Mode.** Fast, but latent and duplicative


**Direct Lake Mode.** **Fast and 'real time'**

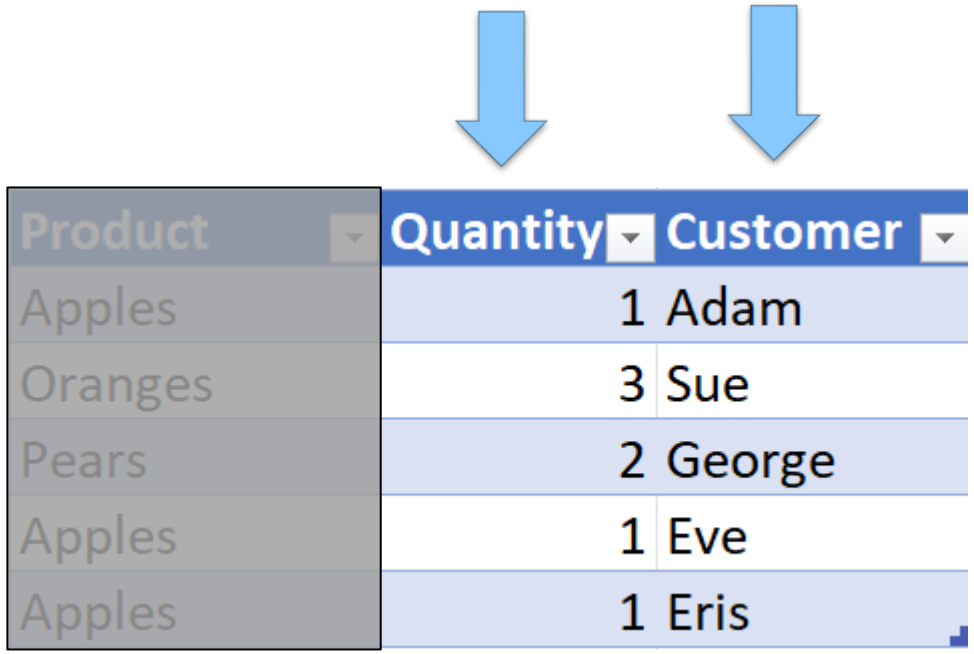


# Why Delta?

# Why Delta (Parquet)?

- Competing with Iceberg for industry standard
- Open Standard for file format
- Column oriented, efficient data storage and retrieval
- Efficient Data Compression and Encoding
- Well suited for pruning ( Column, rowgroup)
- Thrives on bulk operations

# What is Parquet



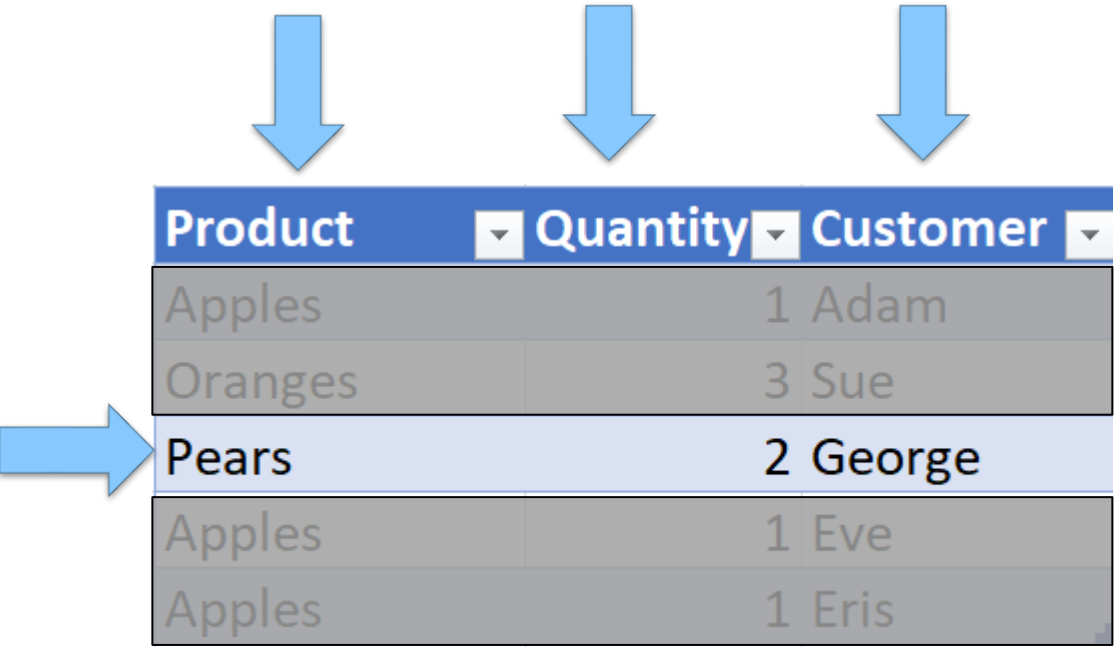
| Product | Quantity | Customer |
|---|---|---|
| Apples | 1 | Adam |
| Oranges | 3 | Sue |
| Pears | 2 | George |
| Apples | 1 | Eve |
| Apples | 1 | Eris |

**A columnar format…**

Great for this

```
SELECT SUM(Quantity)
GROUP BY Customer
```

# What is Parquet



| Product | Quantity | Customer |
|---|---|---|
| Apples | 1 | Adam |
| Oranges | 3 | Sue |
| Pears | 2 | George |
| Apples | 1 | Eve |
| Apples | 1 | Eris |

**A columnar format…**

Not so great for this:

```
UPDATE (Product, Quantity)
WHERE Customer='George'
```

# What is Parquet

| Product | Quantity | Customer |
|---|---|---|
| Apples (6) | 1 | Adam |
| Oranges (7) | 3 | Sue |
| Pears (5) | 2 | George |
| Apples (6) | 1 | Eve |
| Apples (6) | 1 | Eris |

Product Size = 30 characters (bytes)

**…, dictionary encoded, …**

Let

Apples = 1

Oranges = 2

Pears = 3

# What is Parquet

| Product | Quantity | Customer |
|---|---|---|
| 1 | 1 | Adam |
| 2 | 3 | Sue |
| 3 | 2 | George |
| 1 | 1 | Eve |
| 1 | 1 | Eris |

Product Size = 5x4 bytes = 20 bytes

**…, dictionary encoded, …**

Let

Apples = 1

Oranges = 2

Pears = 3

# What is Parquet

| Product | Quantity | Customer |
|---|---|---|
| 0b01 | 1 | Adam |
| 0b10 | 3 | Sue |
| 0b11 | 2 | George |
| 0b01 | 1 | Eve |
| 0b01 | 1 | Eris |

Product Size = 5x2 bits = 10 bits ~ 2 bytes

**That's 15x smaller!**

**…, dictionary encoded, …**

**2 bits are enough for 3 values!!! -- BITPACKING**

Let

Apples = 1 = 0b01

Oranges = 2 = 0b10

Pears = 3 = 0b11

# What is Parquet

| Product | Quantity | Customer |
|---|---|---|
| Apples | 1 | Adam |
| Oranges | 3 | Sue |
| Pears | 2 | George |
| Apples | 1 | Eve |
| Apples | 1 | Eris |

**…, with RLE compression**

**RLE** - Run-length encoding

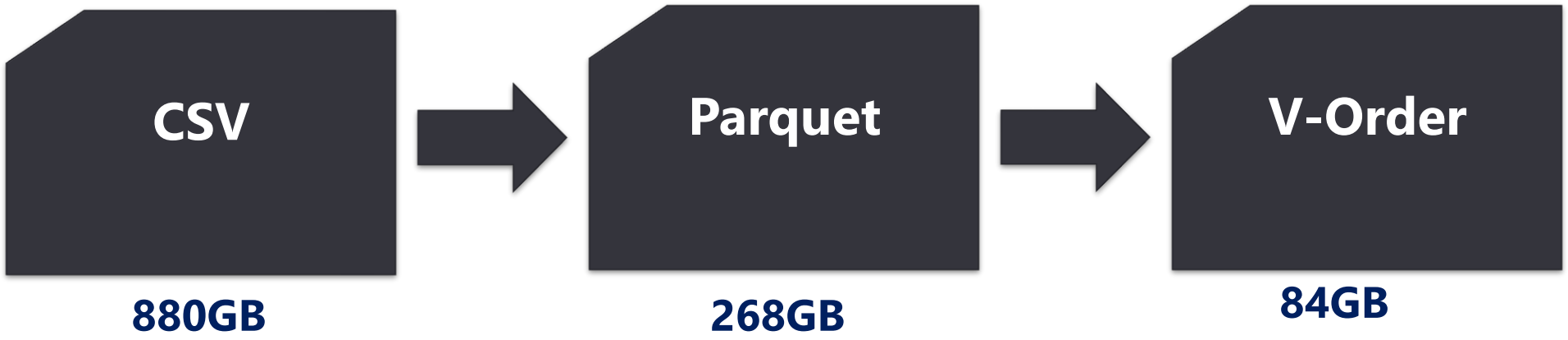
(replace repeated occurrences with the count)

# Introducing V-Ordering

- Write time optimization to parquet files
- Sorting, row group distribution, dictionary encoding, and compression (Shuffling)
- Complies to the open standard
- Z-Order, compaction, vacuum, time travel, etc. are compatible with V-Order

# V-ordering in action

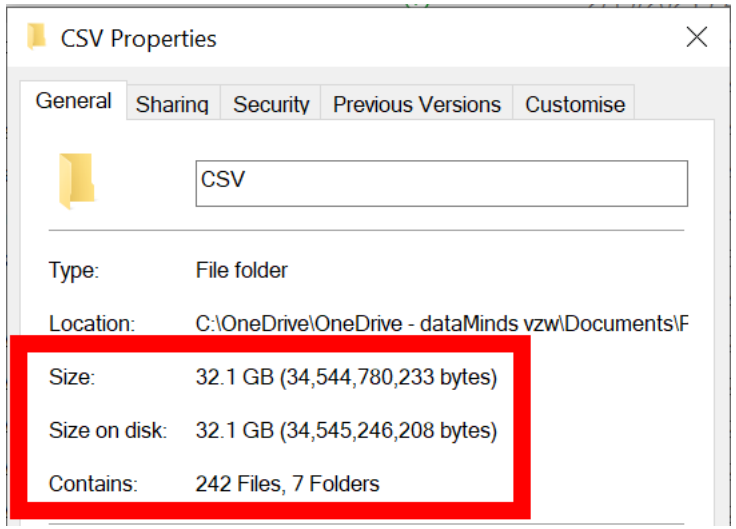
**Microsoft Internal DB (162 tables)**



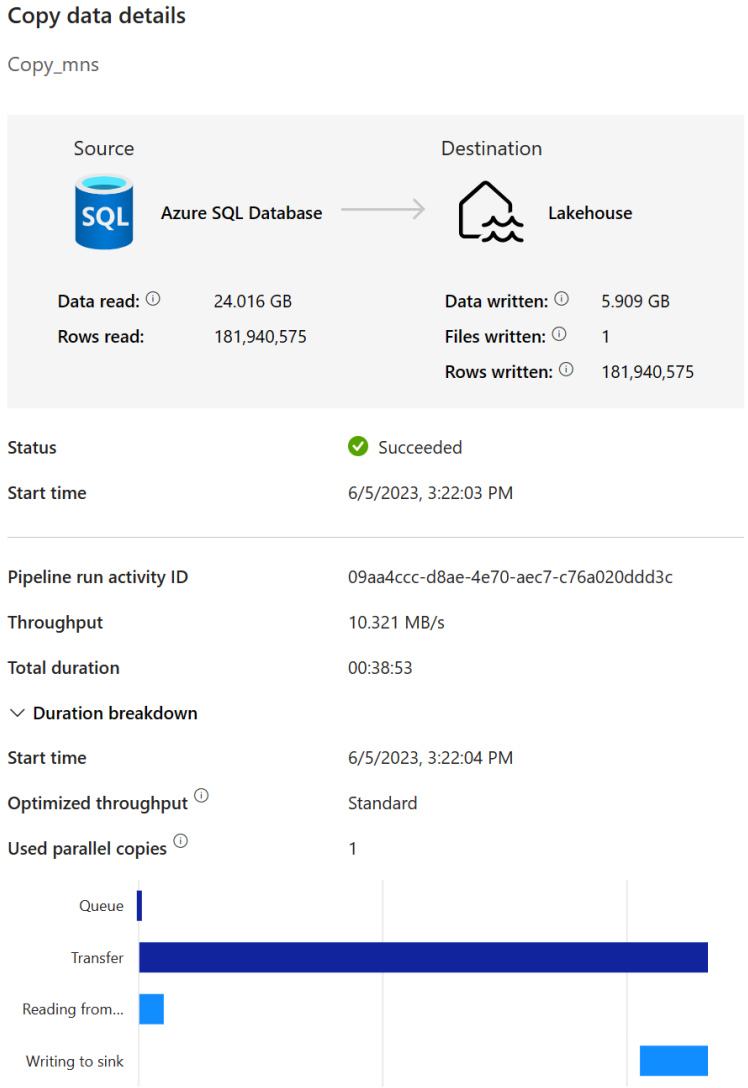

x3.2

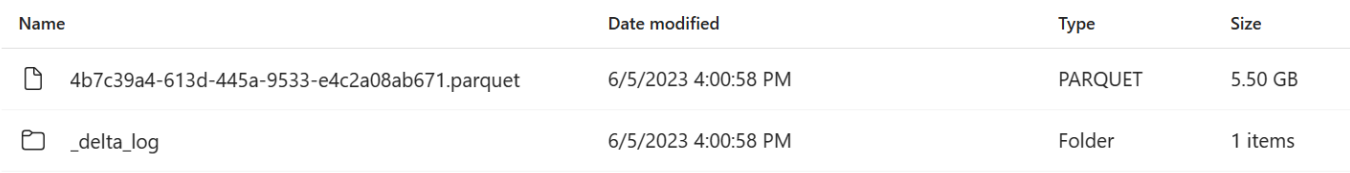
Reduced IO for workloads

# V-ordering in our demo case




| | TABLE_NAME | SCHEMA_NAME | Rows | TotalReservedSpaceMB | UsedDataSpaceMB | FreeUnusedSpaceMB |
|---|---|---|---|---|---|---|
| 1 | Trips_FA | Analytical | 181940575 | 6413 | 6412 | 0 |
| 2 | Time_DI | Analytical | 86400 | 15 | 15 | 15 |
| 3 | Bike_DI | Analytical | 35553 | 1 | 1 | 1 |
| 4 | Date_DI | Analytical | 7304 | 19 | 19 | 19 |
| 5 | Date_DI | Analytical | 7304 | 0 | 0 | 0 |
| 6 | Station_DI | Analytical | 3430 | 0 | 0 | 0 |
| 7 | Gender_DI | Analytical | 59 | 0 | 0 | 0 |
| 8 | Region_DI | Analytical | 8 | 0 | 0 | 0 |
| 9 | RideType_DI | Analytical | 4 | 0 | 0 | 0 |
| 10 | UserType_DI | Analytical | 3 | 0 | 0 | 0 |
| 11 | TripType_DI | Analytical | 3 | 0 | 0 | 0 |
| 12 | MemberType_DI | Analytical | 3 | 0 | 0 | 0 |
| 13 | FileType_DI | Analytical | 3 | 0 | 0 | 0 |



# What is Direct Lake Mode?

# DirectLake Mode

- On start, no data is loaded *in memory*
- Column data is *transcoded* from Parquet files when queried
- Tables can have mix of *resident* and *non-resident* columns
- Column data can get *evicted* over time
- DirectLake *fallback* as an alternative
- "*Framing*" determines what gets loaded from Delta Lake

# STOP! Demo time!

Using Direct Lake mode over a Lakehouse

# Fallback to DirectQuery & Framing

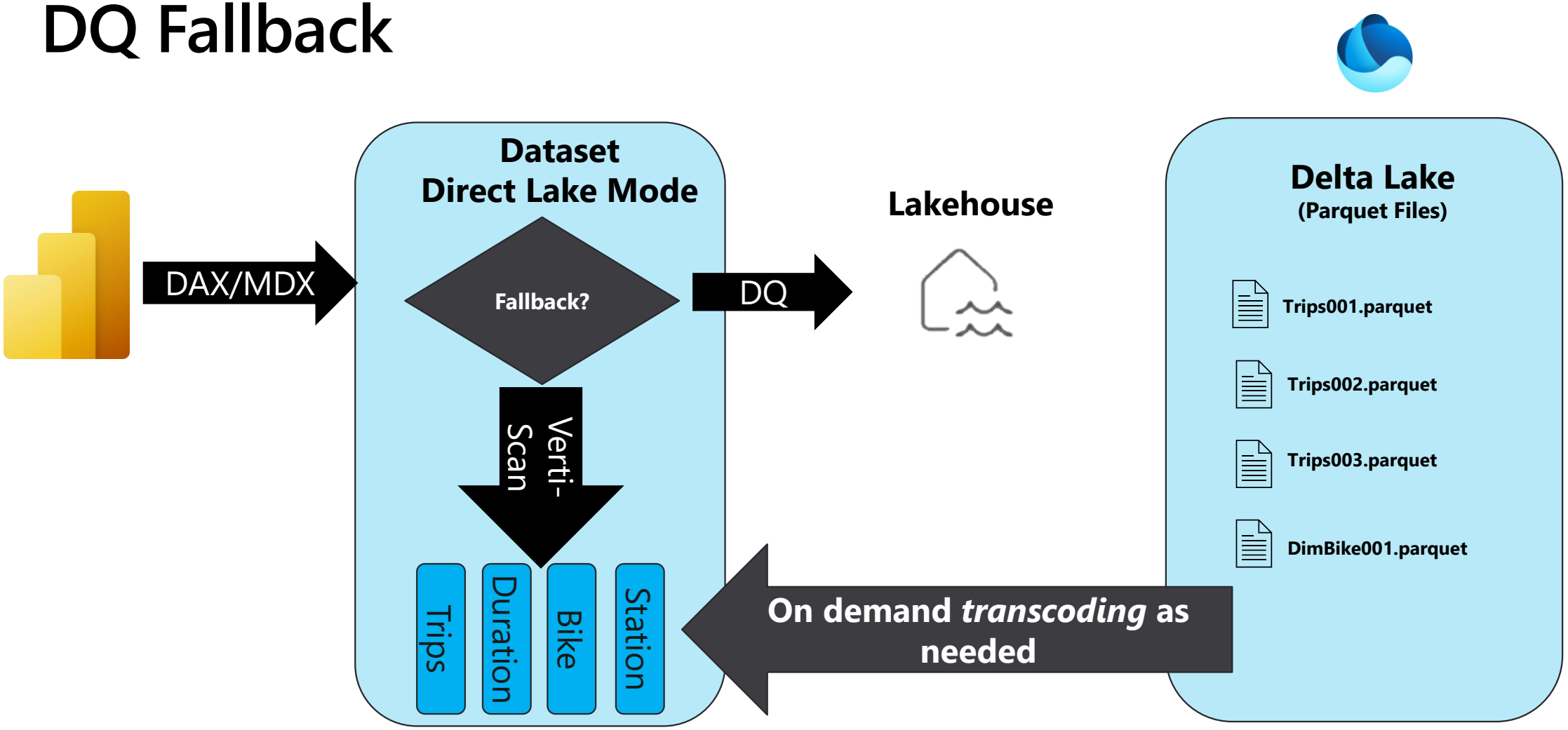
# Framing

- What is framing
  - "point in time" way of tracking what data can be queried by DirectLake
- Framing is near instant and acts like a cursor
  - Determines the set of .parquet files to use/ignore for *transcoding* operations
- Why is this important
  - Delta-lake data is transient for many reasons
- Typical ETL Process
  - Ingest data to delta lake tables
  - Transform as needed using preferred tool
  - When ready, perform *Framing* operation on dataset

# STOP! Demo time!

Let’s look at Framing

# DQ Fallback


DAX/MDX
Dataset
Direct Lake Mode
Fallback?
DQ
Lakehouse
Delta Lake
(Parquet Files)
Trips001.parquet
Trips002.parquet
Trips003.parquet
DimBike001.parquet
Verti-Scan
Trips
Duration
Bike
Station
On demand *transcoding* as needed

# STOP! Demo time!

Let’s look at Fallback

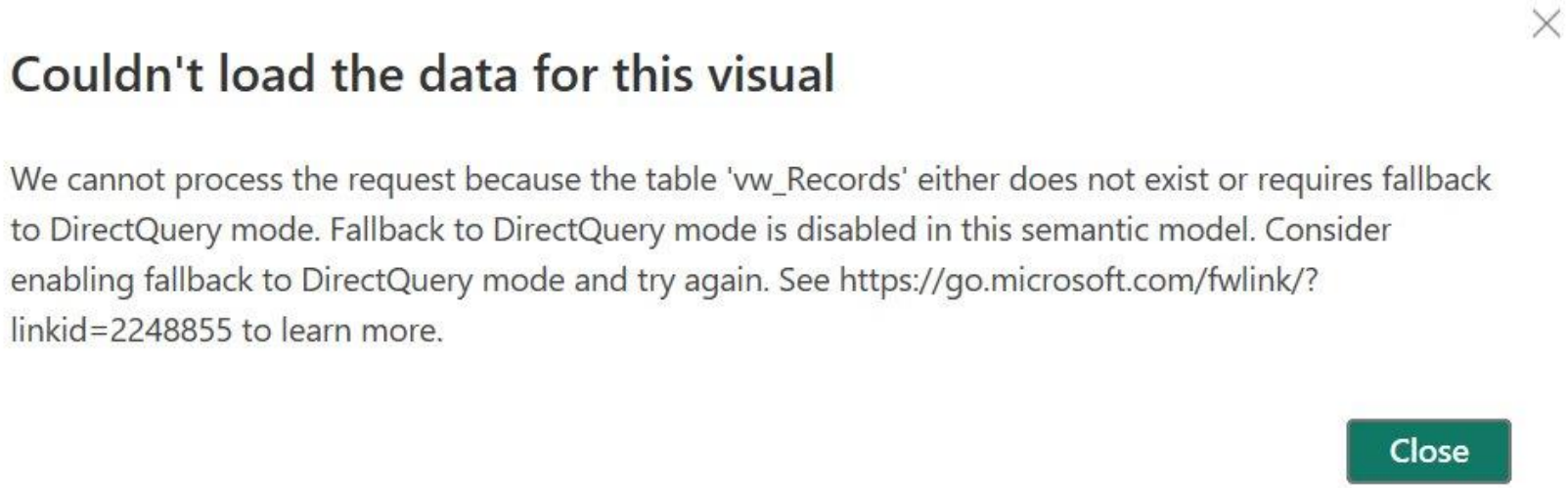
# Guardrails

| Fabric/Power BI SKUs | Parquet files per table | Row groups per table | Rows per table (millions) | Max model size on disk/OneLake1 (GB) | Max memory (GB) |
|---|---|---|---|---|---|
| F2 | 1000 | 1000 | 300 | 10 | 3 |
| F4 | 1000 | 1000 | 300 | 10 | 3 |
| F8 | 1000 | 1000 | 300 | 10 | 3 |
| F16 | 1000 | 1000 | 300 | 20 | 5 |
| F32 | 1000 | 1000 | 300 | 40 | 10 |
| F64/FT1/P1 | 5000 | 5000 | 1500 | Unlimited | 25 |
| F128/P2 | 5000 | 5000 | 3000 | Unlimited | 50 |
| F256/P3 | 5000 | 5000 | 6000 | Unlimited | 100 |
| F512/P4 | 10000 | 10000 | 12000 | Unlimited | 200 |
| F1024/P5 | 10000 | 10000 | 24000 | Unlimited | 400 |
| F2048 | 10000 | 10000 | 24000 | Unlimited | 400 |

https://learn.microsoft.com/en-us/power-bi/enterprise/directlake-overview#fallback

# Identifying Fallback

- You can tell when Fallback happens if ..
  - It’s slower than usual ☺
  - Using the Performance Analyzer, you see a “DirectQuery” category
  - Performing a trace, you see DirectQueryBegin and DirectQueryEnd events
  - Depending on the behaviour, you get an error in the report(s)



# Controlling Fallback Behaviour

- The FallbackBehaviour is set to ‘Automatic’ by default
- Alternative options are:
  - DirectLake only
  - DirectQuery only
- Be careful when making changes to this ..

# Optimizing Delta for Direct Lake mode

# Optimizing Delta for Direct Lake mode

- V-Order makes a big difference, as it's tailored for Verti-Scan
- Direct Lake will work over Shortcuts to external data
  - Expect a performance impact, because reasons ..
- Direct Lake thrives on fewer, larger .parquet files
  - Physical structure will always be crucial
  - OPTIMIZE (bin-compaction) and VACUUM in the Data Engineering process will be key
  - Especially with streaming/small batch architectures, keep this in mind
- Principle of lean models will still apply
  - Only include what's needed for the reports and datasets

# Warming the cache

https://github.com/m-kovalsky/Fabric/blob/main/WarmDirectLakeCache_IsResident.py

# Semantic Link Labs

https://github.com/microsoft/semantic-link-labs

# Limitations

- Relationships based on DateTime types
- Calculated Columns and Calculated Tables
  - Calculation Groups and Field Parameters do work
- User Defined Hierarchies will not work in Excel
- Complex delta table column types (i.e. Binary and GUID)
  - Some other
- T-SQL Based views will **always** fallback to DQ mode
- Composite models are not yet supported

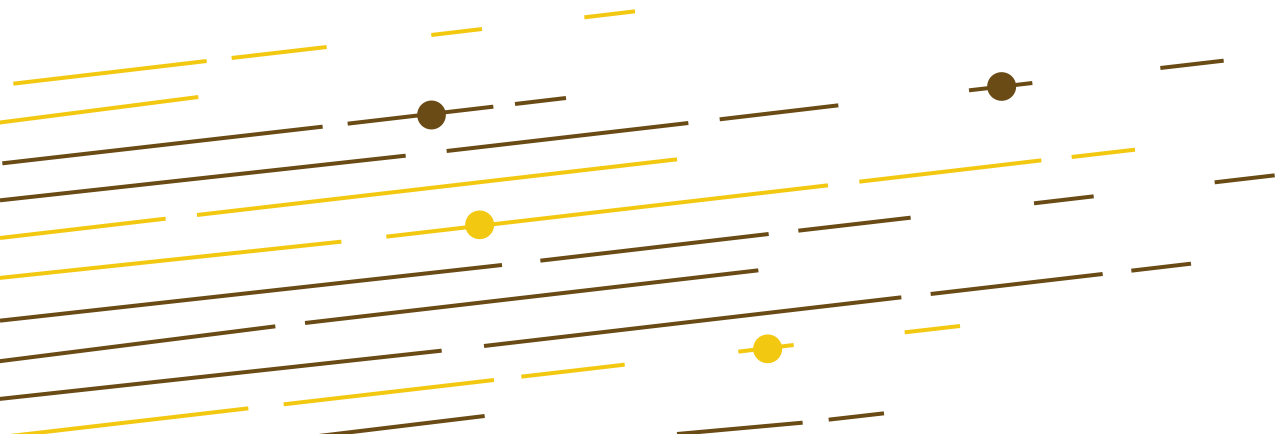
# Wasn’t there this thing called DirectLake over OneLake?

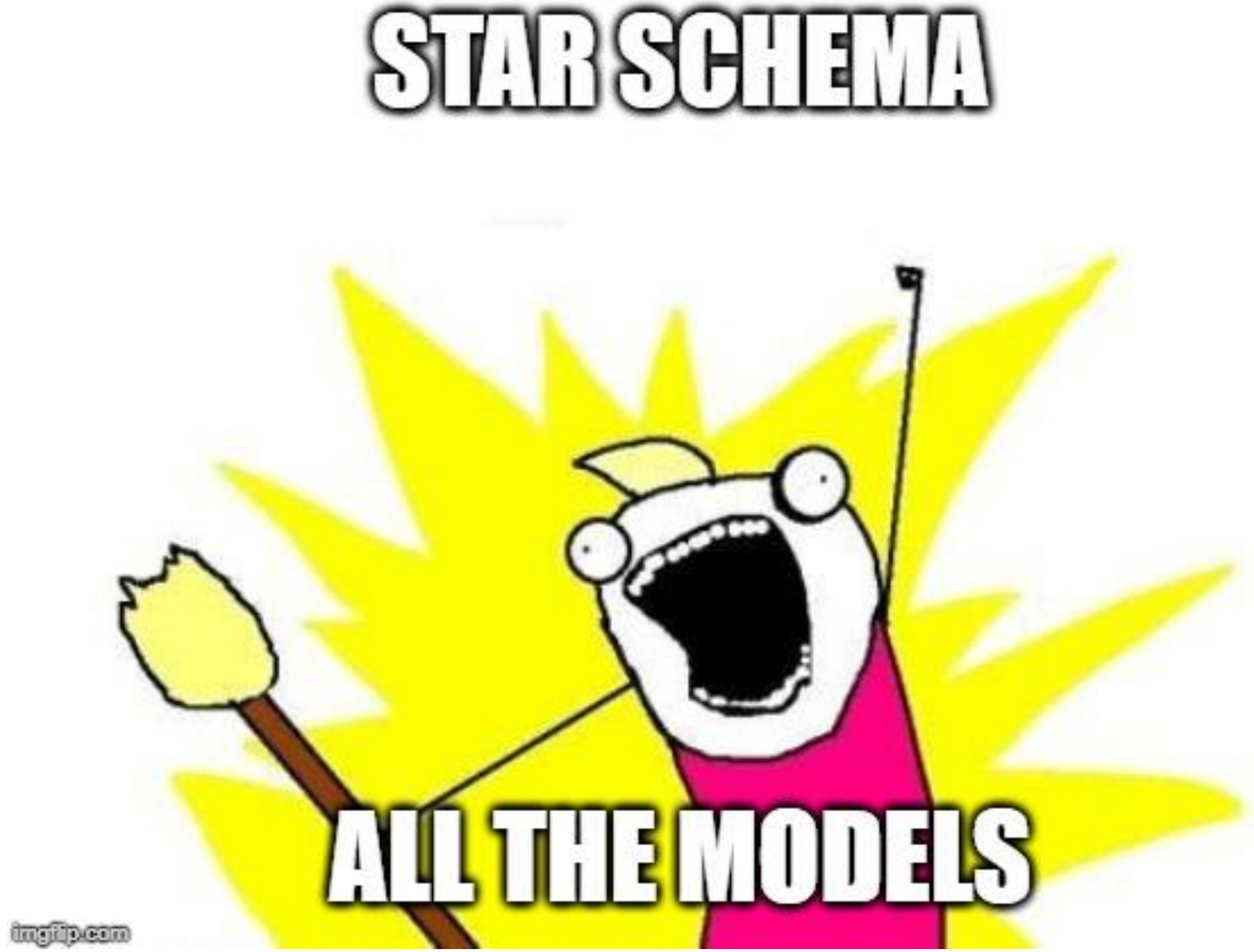
# Common Answers to Common Questions

# “Greatest Hits”

- Delta doesn’t like spaces in object names ☺
- Delta Tables are a hard requirement for Direct Lake mode
  - Dataflows Gen2, Pipelines, Notebooks can create them for you in the lakehouse
- Web modelling is the recommended way to create DirectLake datasets.
  - We do allow to CREATE and EDIT from Power BI Desktop
- XMLA Read/Write is supported, and allows web modelling
- Direct Lake doesn’t have unique DAX limitations
  - DQ does ..

# What does this mean for my data modelling?






Thanks, @KoVer!

*Data should be transformed as far upstream as possible, and as far downstream as necessary.*

Matthew Roche, 2021
(The purple haired sword afficionado)
https://ssbipolar.com/2021/05/31/roches-maxim

# Resources

- https://learn.microsoft.com/en-us/power-bi/enterprise/directlake-overview
- https://learn.microsoft.com/en-us/power-bi/enterprise/directlake-analyze-qp
- https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-pbi-reporting
- https://learn.microsoft.com/en-us/fabric/data-engineering/delta-optimization-and-v-order?tabs=sparksql
- https://fabric.guru/power-bi-direct-lake-mode-frequently-asked-questions
- https://www.fourmoo.com/2023/05/24/using-power-bi-directlake-in-microsoft-fabric/
- https://fabric.guru/controlling-direct-lake-fallback-behavior
- https://github.com/m-kovalsky/Fabric

# Slides

https://github.com/BenniDeJagere/Presentations/{Year}/{YYYYMMDD}_{Event}

# Thank you