Microsoft

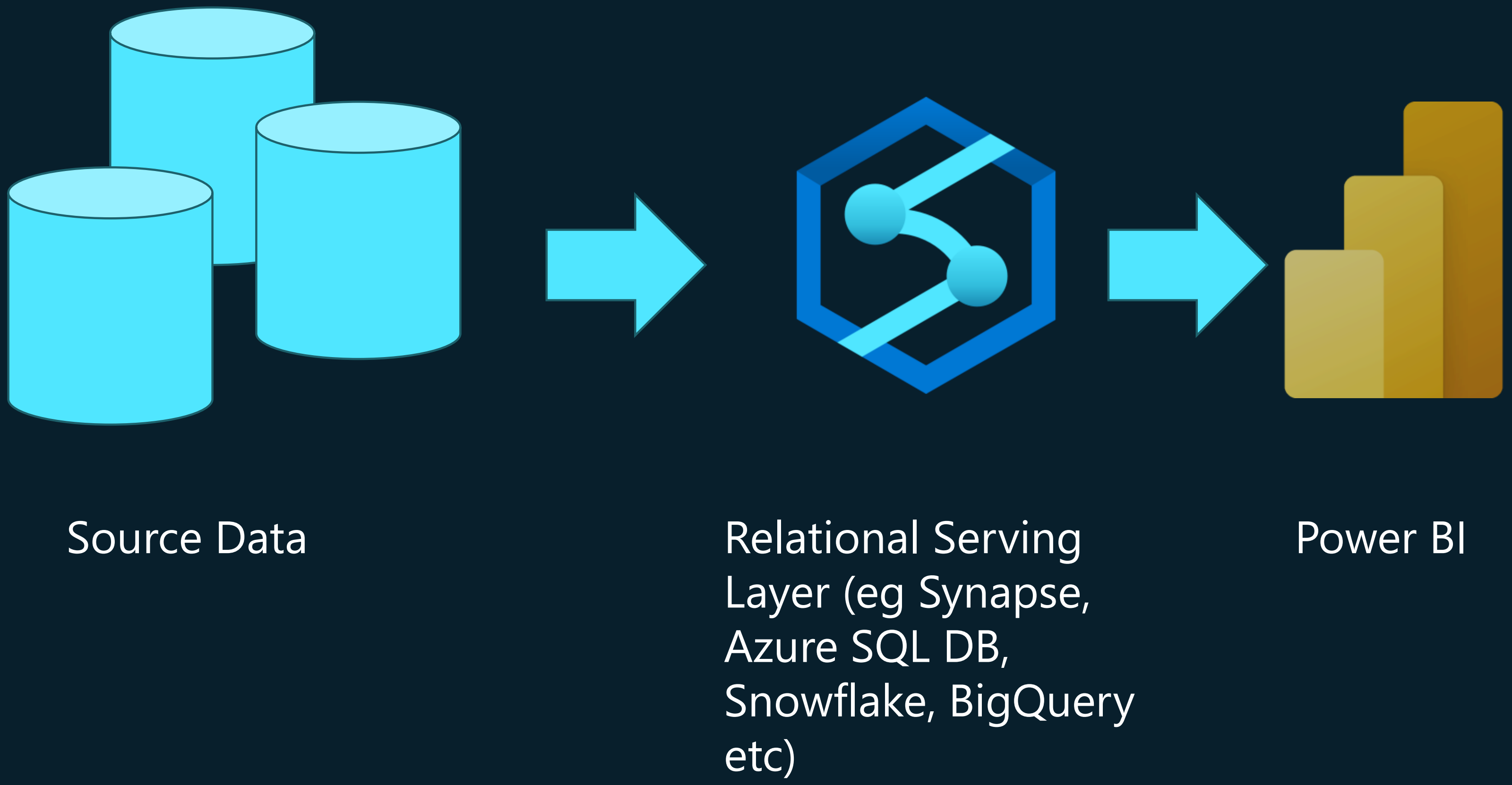
# Direct Lake

Andrea Benedetti

in /in/abenedetti X @anBenedetti https://github.com/anbened

# What Is Direct Lake?

# Traditional Microsoft BI Architecture



Source Data

Relational Serving Layer (eg Synapse, Azure SQL DB, Snowflake, BigQuery etc)

Power BI

# Storage Modes (today)

| SMALLER MODELS | Time to Import Data | Model Size | Query Speed |
|---|---|---|---|
| Direct Query | - | - | ? |
| Import | ☺ | ☺ | ☺ |

| LARGE MODELS | Time to Import Data | Model Size | Query Speed |
|---|---|---|---|
| Direct Query | - | - | ? |
| Import | 😐 | 😐 | ☺ |

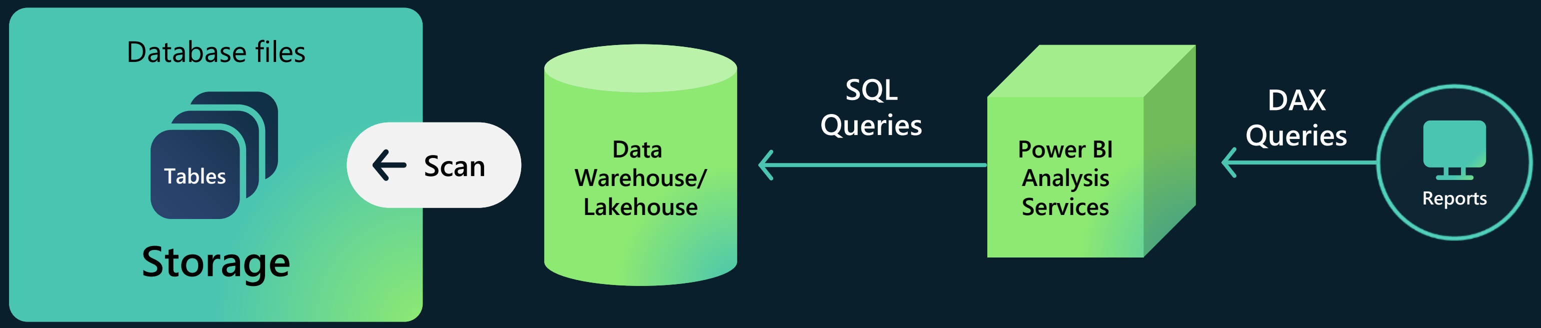
# Storage Modes

| SMALLER MODELS | Time to Import Data | Model Size | Query Speed |
|---|---|---|---|
| Direct Query | - | - | ? |
| Import | ☺ | ☺ | ☺ |
| Direct Lake | ☺ | ☺ | ☺ |

| LARGE MODELS | Time to Import Data | Model Size | Query Speed |
|---|---|---|---|
| Direct Query | - | - | ? |
| Import | 😐 | 😐 | ☺ |
| Direct Lake | ☺ | ☺ | ☺ |

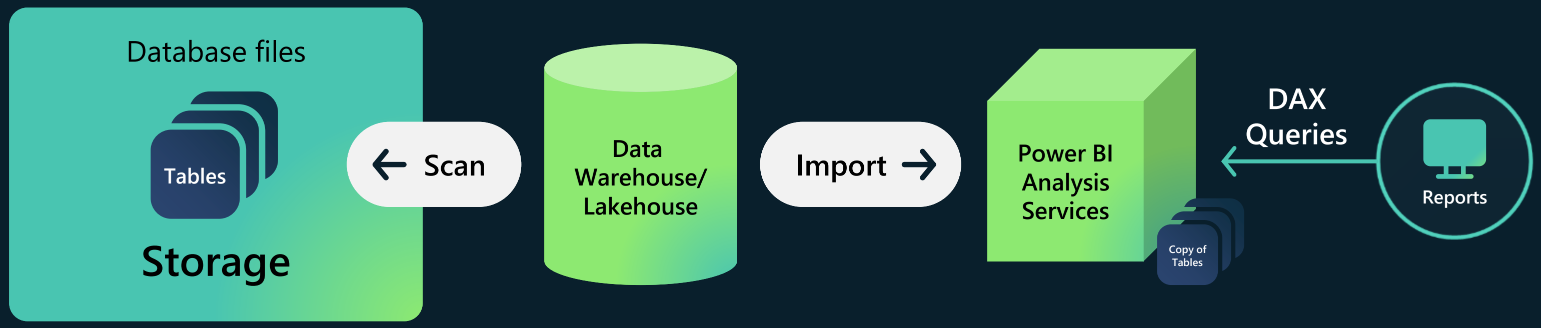
## “Direct Query Mode”

Slow, but real time



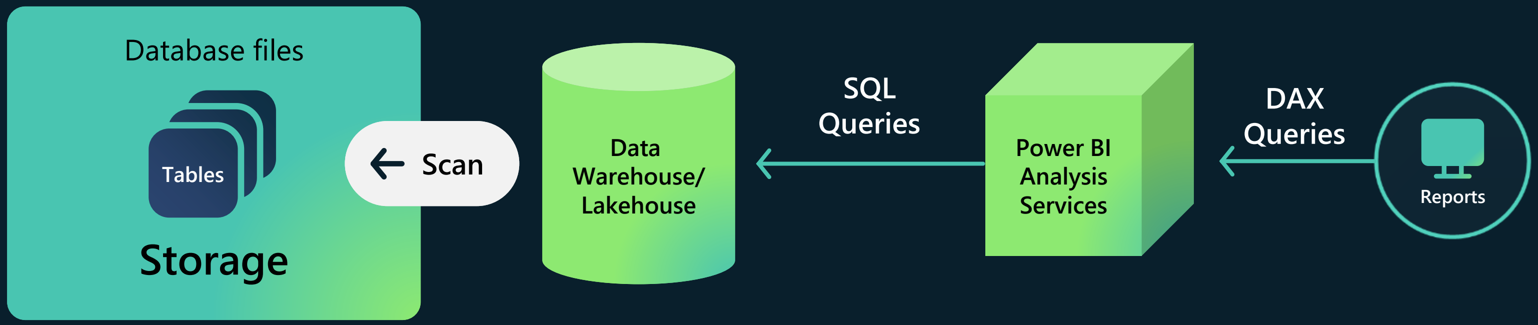
## “Import Mode”

Latent & duplicative but fast



## “Direct Query Mode”

Slow, but real time



Database files

Tables

Storage

Scan

Data Warehouse/ Lakehouse

SQL Queries

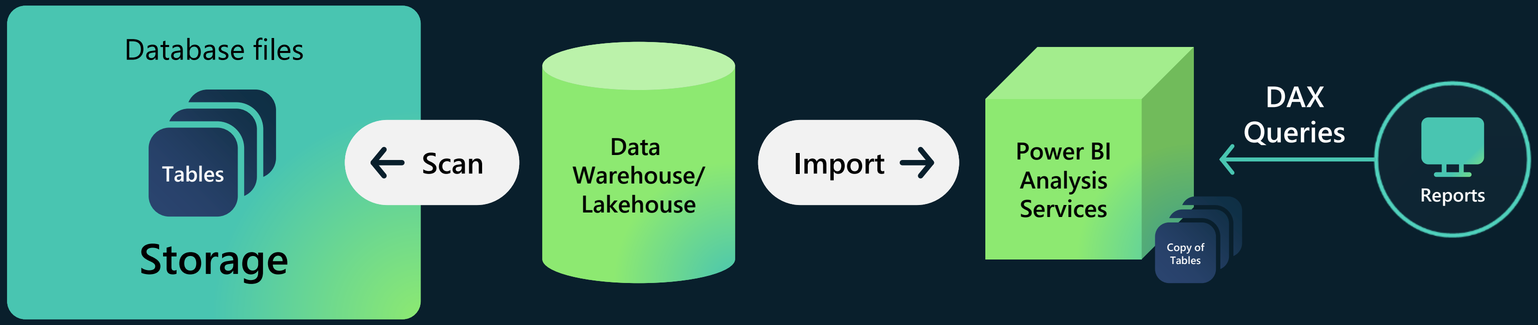
Power BI Analysis Services

DAX Queries

Reports

## “Import Mode”

Latent & duplicative but fast



Database files

Tables

Storage

Scan

Data Warehouse/ Lakehouse

Import

Power BI Analysis Services

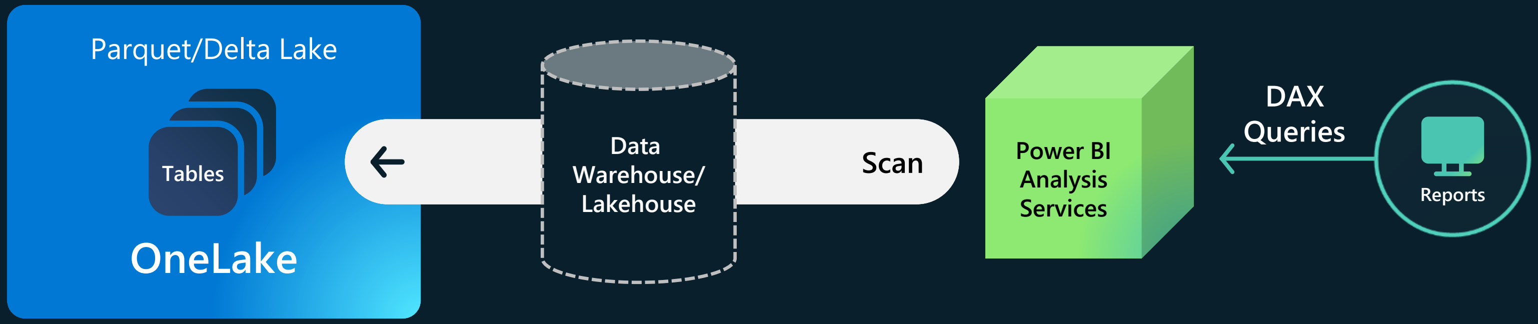
Copy of Tables

DAX Queries

Reports

## “Direct Lake Mode”

Perfect!



Parquet/Delta Lake

Tables

OneLake

Data Warehouse/ Lakehouse

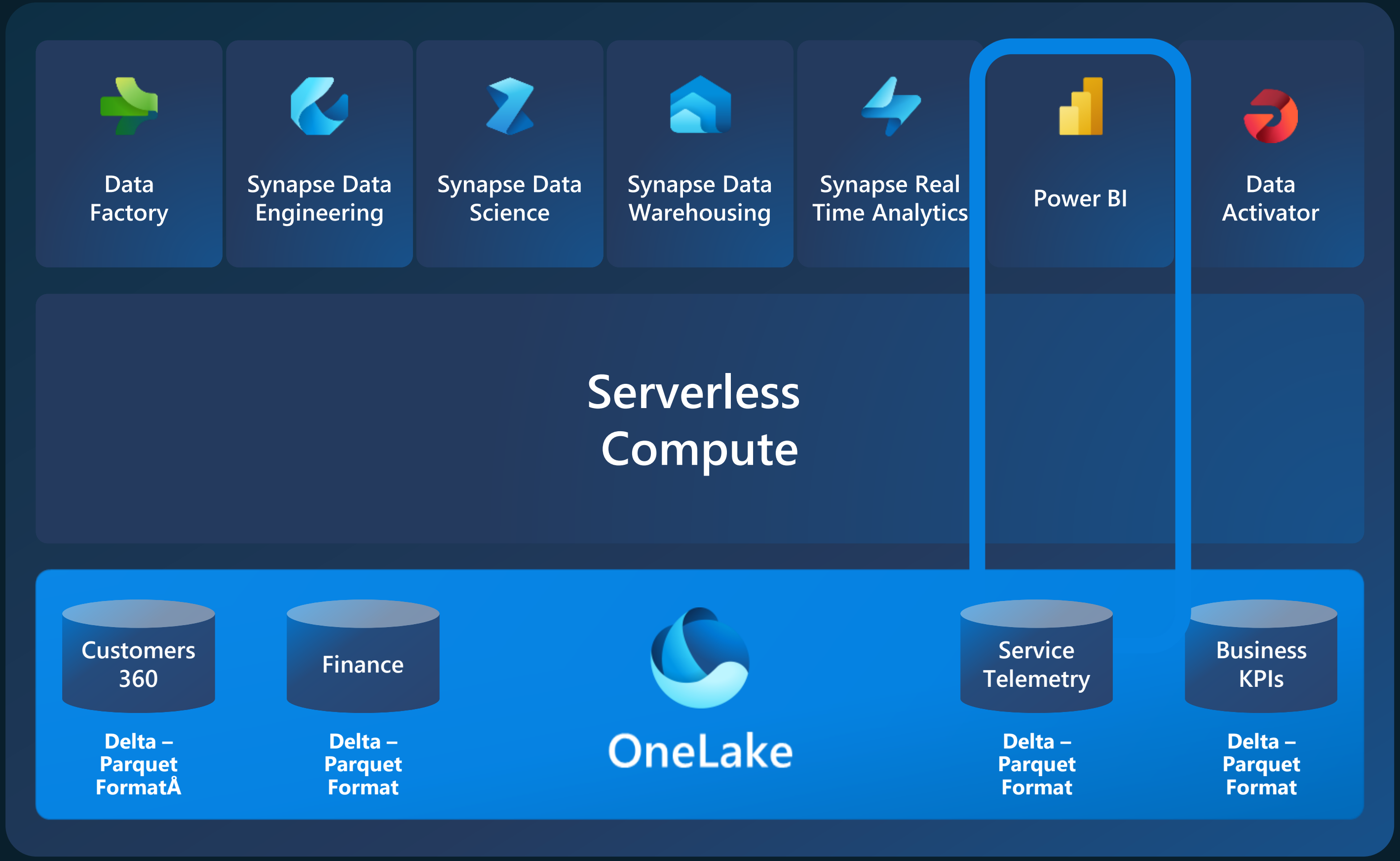
Scan

Power BI Analysis Services

DAX Queries

Reports

# One Copy – Direct Lake



All the compute engines store their data automatically in OneLake

The data is stored in a single common format

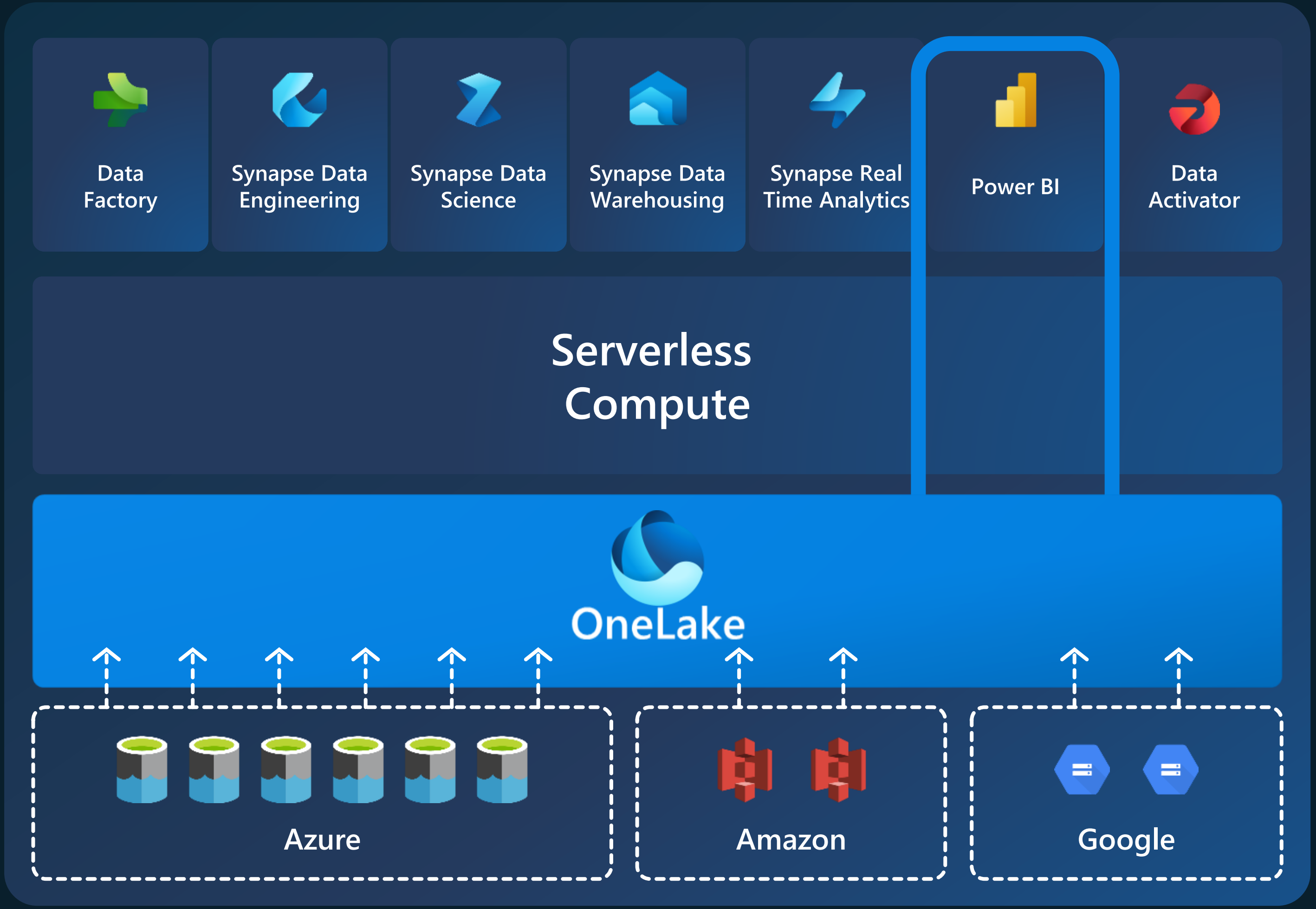
Delta – Parquet, an open standards format, is the storage format for all tabular data in Analytics vNext

Once data is stored in the lake, it is directly accessible by all the engines without needing any import/export

All the compute engines have been fully optimized to work with Delta Parquet as their native format

Shared universal security model is enforced across all the engines

# One Copy – Direct Lake



Sharing data in OneLake is as easy as sharing files in OneDrive, removing the needs for data duplication

With shortcuts, data throughout OneLake can be composed together without any data movement

Shortcuts also allow instant linking of data already existing in Azure and in other clouds, without any data duplication and movement, making OneLake the first multi-cloud data lake

With support for industry standard APIs, OneLake data can be directly accessed by any application or service

# Fundamentals

- Only one data source can be used with Direct Lake
- Direct Lake semantic model starts life with no data in memory
- Data is *paged* into semantic model on demand triggered by query
- Tables can have resident and non-resident columns
- Column data can get evicted for multiple reasons
- Direct Lake fallback to SQL Server for *suitable* sub-queries
- “Framing” for data consistency in Power BI reports

# Direct Lake limitations (for now)

- No calculated columns or calculated tables
- No composite models
  - Although calculation groups and field parameters are now allowed
- Can only be used with tables, not views
- Can only be used with security defined in the semantic model
- Web authoring experience (or 3rd party tool)
- Not all data types supported
  - No structured data types, binary or GUID columns
  - DateTime relationships not supported
  - String length limited to 4000 characters

# Direct Lake prerequisites

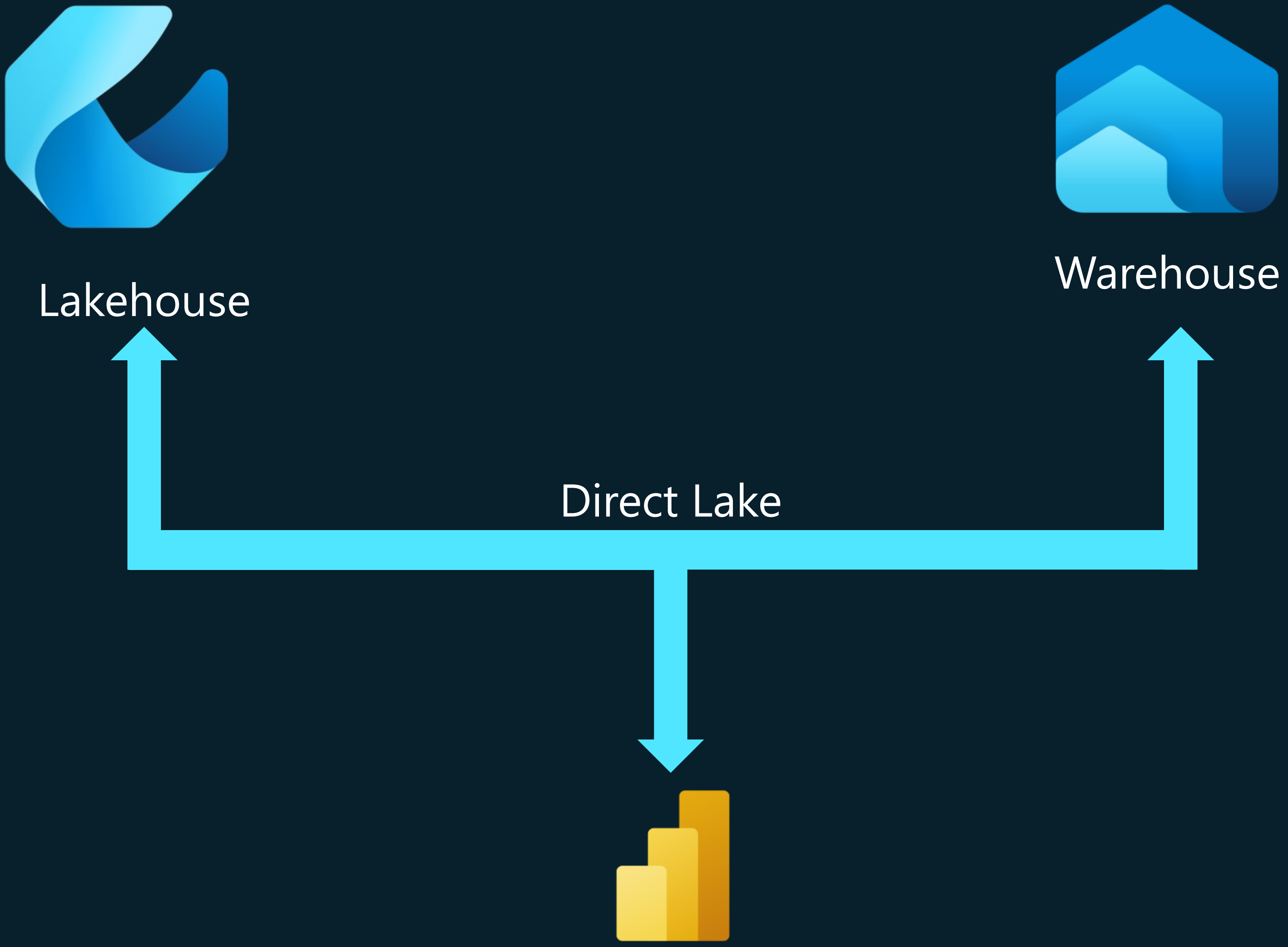
SKU Requirements

- Power BI Premium P
- Microsoft Fabric F SKUs only, including Trial

Not supported on:

  - Power BI Pro
  - Premium Per User
  - Power BI Embedded A/EM Skus

# Direct Lake prerequisites



Lakehouse

Warehouse

Direct Lake

# Anatomy of Parquet

# Why Parquet?

- Open source/open data format
- Column-oriented format is optimized for data storage and retrieval
- Efficient data compression and encoding especially data in bulk
- Is lingua franca for data storage format
    - Databricks, Microsoft - delta lake and parquet
    - Snowflake - iceberg and parquet/orc

# Anatomy of a Parquet File

**CSV**, XML, JSON…… Parquet

```
StoreID , DateTime   , ProductID , Value
StoreA  , 2023-01-01 , SKU001    , 10
StoreA  , 2023-01-02 , SKU001    , 15
StoreA  , 2023-01-03 , SKU001    , 12
```

# Anatomy of a Parquet File

CSV, **XML**, JSON...... Parquet

```
<sale>
     <StoreID>StoreA</StoreID>
     <DateTime>2023-01-01</DateTime>
     <ProductID>SKU001</ProductID>
     <Value>10</Value>
</sales>
<sale> … </sale>
```

# Anatomy of a Parquet File

CSV, XML, **JSON**…… Parquet

```
{sales[
    {
    StoreID: “StoreA” ,
    DateTime: “2023-01-01” ,
    ProductID: “SKU001” ,
    Value:10
    },
    {…}
]}
```

# Anatomy of a Parquet File

CSV, XML, JSON...... **Parquet**

```
Header:
RowGroup1:
     StoreID   :    StoreA, StoreA, StoreA
     DateTime  :    2023-01-01, 2023-01-02, 2023-01-03
     ProductID:     SKU001, SKU001, SKU001
     Value     :    10, 15, 12
RowGroup2:
….
Footer:
```

# Anatomy of a Parquet File – Dictionary IDs

CSV, XML, JSON…… **Parquet**

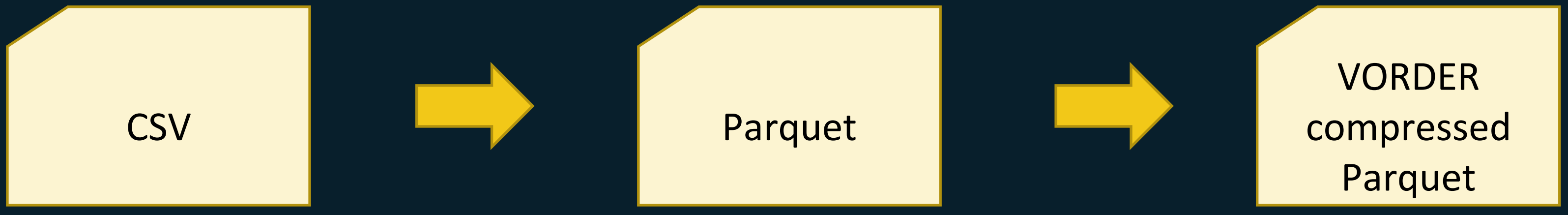
```
    Header:
    RowGroup1:
         StoreID   :   1, 1, 1
         DateTime  :   1, 2, 3
         ProductID :   1, 1, 1
         Value     :   1, 2, 3
    RowGroup2:
    ….
    Footer:
```

# V-Order

# V-Order

- V-Order is a Microsoft-proprietary optimisation for writing data in parquet files (as used in Delta tables)
- V-Order uses the same algorithms used by Power BI Import mode semantic models to compress data
- V-Ordered Delta tables can be read by any tool that can read Delta
- Direct Lake will perform better on V-Ordered Delta tables
- Direct Lake will work on all Delta tables, even without V-Order

# VORDER compressed Parquet



CSV

Parquet

VORDER compressed Parquet

880GB

268GB

84GB

x3.2

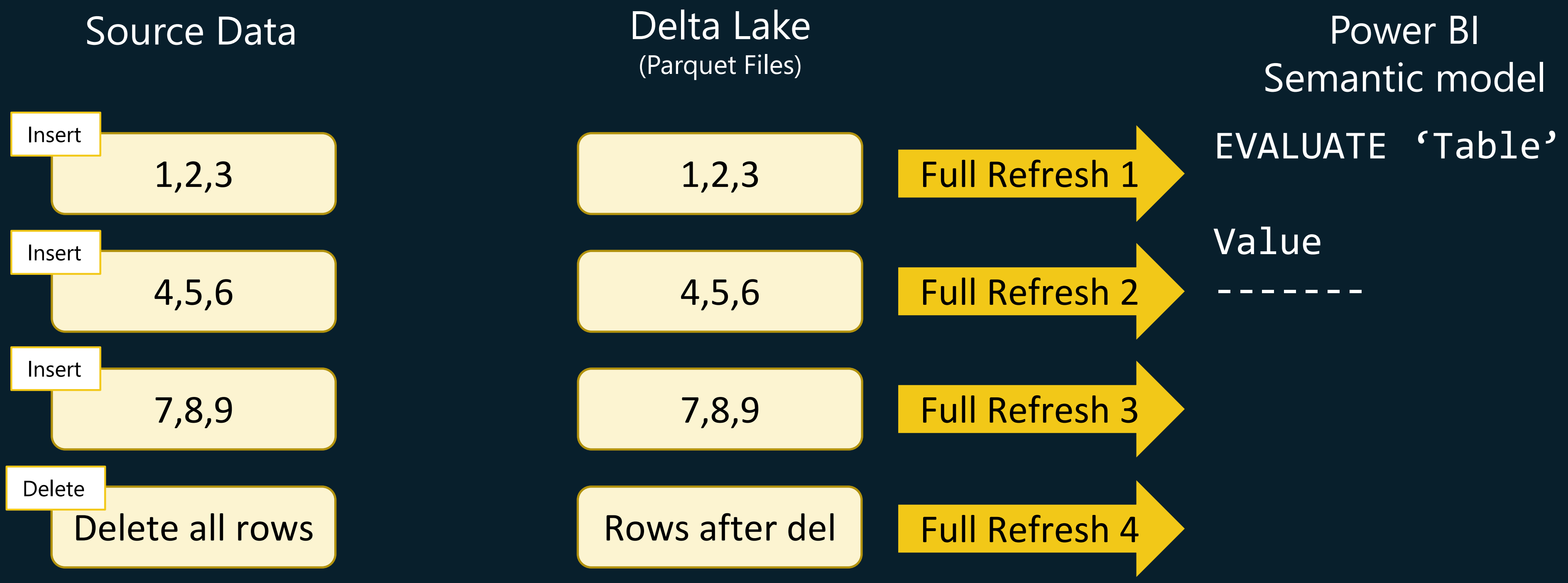
# Demo

# Framing

# Framing

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

# Framing



Source Data

Insert 1,2,3

Insert 4,5,6

Insert 7,8,9

Delete Delete all rows

Delta Lake
(Parquet Files)

1,2,3

4,5,6

7,8,9

Rows after del

Full Refresh 1

Full Refresh 2

Full Refresh 3

Full Refresh 4

Power BI
Semantic model

```
EVALUATE ‘Table’

Value
-------
```

# Framing - Options

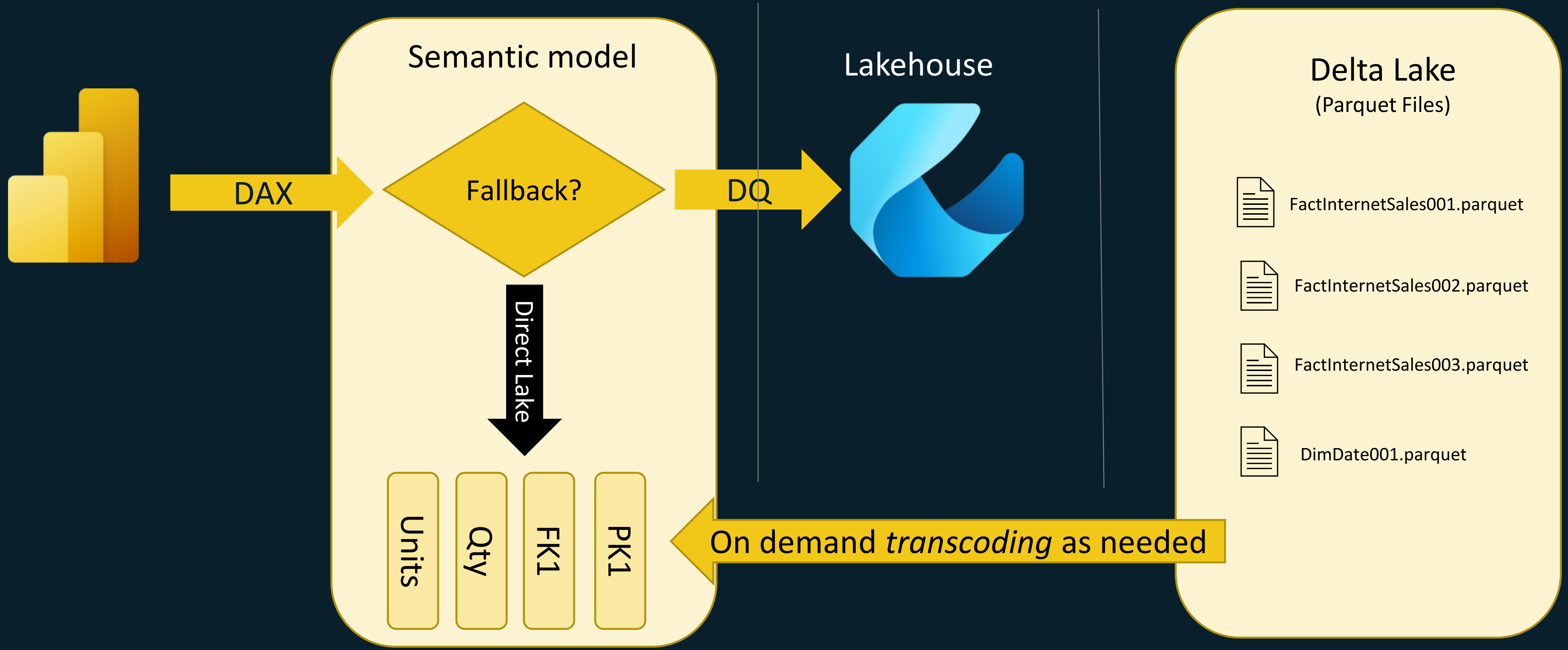
- Automatic
  - Default - can be turned off
  - Triggered each time Delta table gets modified
- Via Fabric Service
  - Manually by refreshing the semantic model
  - Configure a schedule
- Via Notebook
  - Use Semantic-link to call reframe using native method
  - Execute_tmsl for fine grain reframing
  - Consider cache-warming as option

# Framing - Options

- SSMS (TMSL)
- Rest API
- Pipeline
- Notebooks (sempy)
- Power Automate etc.

# DirectQuery Fallback

# Fallback to DirectQuery


Semantic model
DAX
Fallback?
DQ
Lakehouse
Direct Lake
Units
Qty
FK1
PK1
Delta Lake
(Parquet Files)
FactInternetSales001.parquet
FactInternetSales002.parquet
FactInternetSales003.parquet
DimDate001.parquet
On demand *transcoding* as needed

# Fallback to DirectQuery

- You are using features that prevent Direct Lake

- Views are not allowed because they don’t have corresponding tables stored in a Lakehouse

- RLS or OLS is defined in a Warehouse
  - Security rules take high priority when defined

# Fallback to DirectQuery – data volumes

- There are limits on how much data can be used for Direct Lake
- These limits vary by capacity SKU size
- If you exceed these limits, Direct Lake will use DirectQuery
  - Query performance may be noticeably worse
- Fabric checks limits during reframing process
- Can be turned On/Off using Direct Lake Behaviour property

# Fallback guardrails

| Fabric/Power BI SKUs | Parquet files per table | Row groups per table | Rows per table (millions) | Max model size on disk/OneLake (GB) | Max memory (GB) |
|---|---|---|---|---|---|
| F2 | 1,000 | 1,000 | 300 | 10 | 3 |
| F4 | 1,000 | 1,000 | 300 | 10 | 3 |
| F8 | 1,000 | 1,000 | 300 | 10 | 3 |
| F16 | 1,000 | 1,000 | 300 | 20 | 5 |
| F32 | 1,000 | 1,000 | 300 | 40 | 10 |
| F64/FT1/P1 | 5,000 | 5,000 | 1,500 | Unlimited | 25 |
| F128/P2 | 5,000 | 5,000 | 3,000 | Unlimited | 50 |
| F256/P3 | 5,000 | 5,000 | 6,000 | Unlimited | 100 |
| F512/P4 | 10,000 | 10,000 | 12,000 | Unlimited | 200 |
| F1024/P5 | 10,000 | 10,000 | 24,000 | Unlimited | 400 |
| F2048 | 10,000 | 10,000 | 24,000 | Unlimited | 400 |

Learn about Direct Lake in Power BI and Microsoft Fabric

# Detecting fallback to DirectQuery

- Performance Analyzer, Profiler traces and/or Log Analytics will show what happens for individual queries

- Limits on data volumes can be checked with Python notebooks (Delta Analyzer) and in some cases DMVs

# Controlling fallback to DirectQuery

- The **DirectLakeBehavior** property sets fallback behaviour

- Automatic (default): allows fallback to DirectQuery if data can’t be loaded into memory

- DirectLakeOnly: allows use of DirectLake but prevents fallback and returns an error instead of using DirectQuery

- DirectQueryOnly: forces all queries to use DirectQuery mode

# Demo

## Direct Lake Behavior

# Performance

# Performance considerations

- Reframing
- Cold Cache
- Warm Cache
- SQL Fallback

# Performance considerations – Reframing

- Time to evict columns and load certain objects

- Loads Delta metadata and some metadata from parquet files

# Performance considerations – Cold Cache

- The time needed to page data into memory from One Lake
- Number/layout of data across Parquet files
- Cache warming tricks

# Performance considerations – Warm Cache

- Query Plans
  - Direct Lake Behaviour property
  - Other optimisations

- Encoding
  - All data is HASH encoded – no option to use VALUE encoding

- Segment data profile
  - Number and layout of data within segments can impact scan performance
  - Depends greatly on filters used per query

# Performance numbers – sample model

- With V-Order
- No V-Order
- Column partitioned by Date
- V-Order and Z-Order

# Performance  - some numbers

| | V-Order | No V-Order | Partitioned by DateKey | V-Order & Z-Order |
|---|---:|---:|---:|---:|
| Rows | 1,000,000,000 | 1,000,000,000 | 1,000,000,000 | 1,000,000,000 |
| Columns | 10 | 10 | 10 | 10 |
| V-Order | TRUE | | TRUE | TRUE |
| Z-Order | | | | DateKey |
| Parquet Size | 7.1GB | 11.6GB | 8.4GB | 6.9GB |
| Files | 14 | 200 | 807 | 6 |
| Row Groups | 26 | 200 | 807 | 24 |
| | | | | |
| Model Size | | | | |
| Data | 7.1GB | 14.9GB | 6.6GB | 6.9GB |
| Total (memory) | 9.6GB | 17.5GB | 9.1GB | 9.4GB |

# Performance numbers – Cold cache

| | V-Order | No V-Order | Partitioned by DateKey | V-Order & Z-Order |
|---|---:|---:|---:|---:|
| Test 1 | 2m 24s | 7m 35s | 7m 16s | 2m 18s |
| Test 2 | 2m 26s | 8m 34s | 7m 30s | 2m 17s |
| Test 3 | 2m 27s | 7m 46s | 7m 27s | 2m 18s |

| | | | | |
|---|---:|---:|---:|---:|
| Parquet Size | 7.1GB | 11.6GB | 8.4GB | 6.9GB |
| Files | 14 | 200 | 807 | 6 |
| Row Groups | 26 | 200 | 807 | 24 |
| | | | | |
| Model Size | | | | |
| Data | 7.1GB | 14.9GB | 6.6GB | 6.9GB |
| Total | 9.6GB | 17.5GB | 9.1GB | 9.4GB |

# Performance numbers – Warm Cache

Group by Weekday
Filter on Category
and Month
Sum Quantity

Group by Month
Filter on Category
Compare using
PREVIOUSMONTH
Sum Quantity

Filter by Month
All countries ranked
by distinct User ID

| | V-Order | No V-Order | Partitioned by DateKey | V-Order & Z-Order |
|---|---|---|---|---|
| Query 1 | | | | |
| | 223 | 863 | 47 | 203 |
| | 1,449 | 12,840 | 550 | 1,148 |
| | | | | |
| Query 2 | | | | |
| | 1,594 | 2,891 | 94 | 1,379 |
| | 11,890 | 39,125 | 1,070 | 10,754 |
| | | | | |
| Query 3 | | | | |
| Total Time | 4,817 | 4,129 | 5,851 | 10,845 |
| FE CPU | 29,937 | 43,933 | 42,523 | 18,867 |

# Summary

- Will my reports run faster with Direct Lake?
- Do I have to use Direct Lake with Fabric?
- Incremental Refresh?
- Aggregations?

Thanks