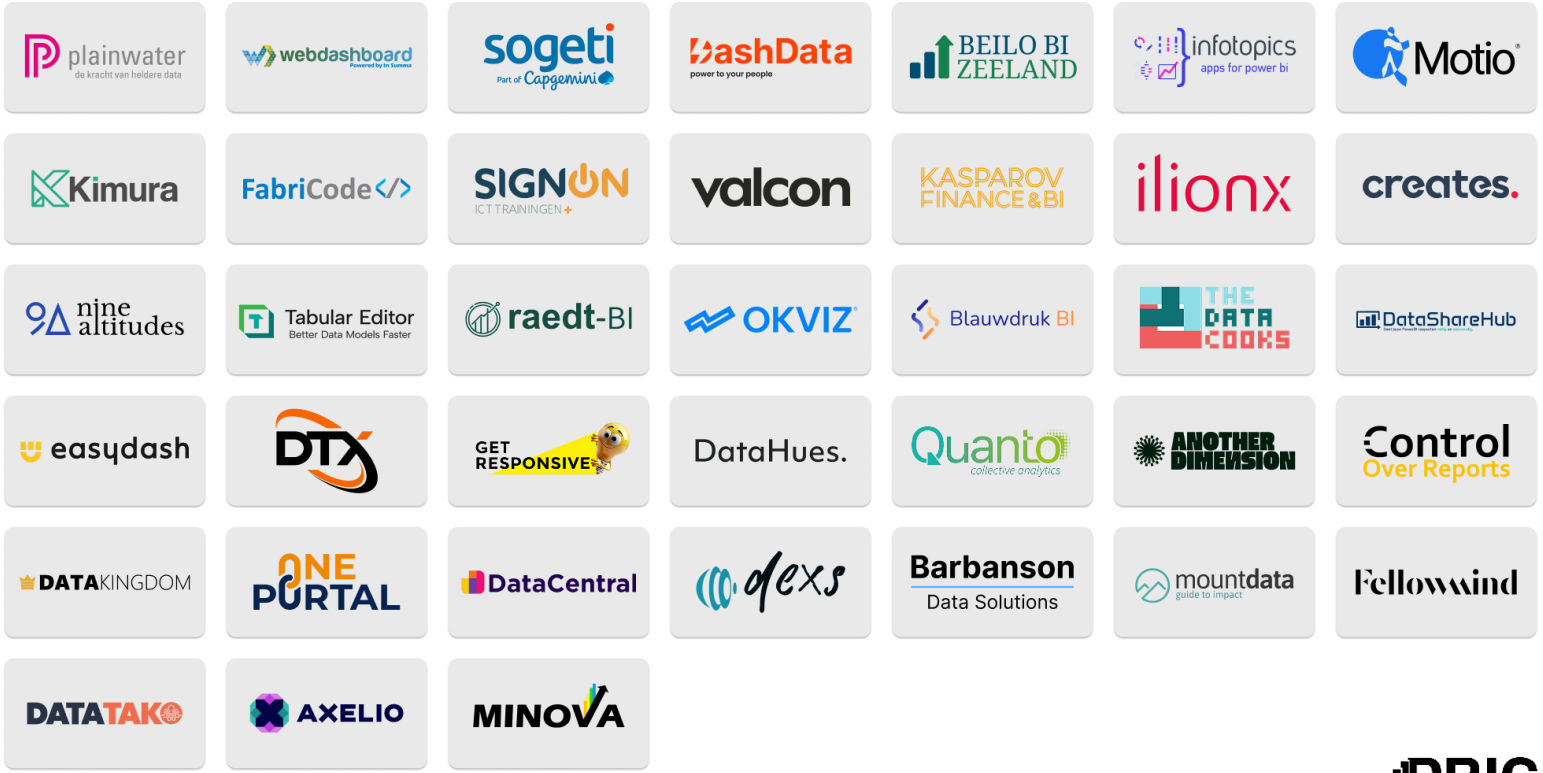
50 Shades of Direct Lake

Nikola Ilic

Data Mozart, Microsoft Data Platform MVP

A big thank you to our partners



PBIG
POWER BI GEBRUIKERSGROEP

Nikola Ilic

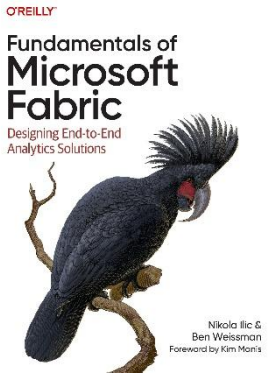
Principal Architect

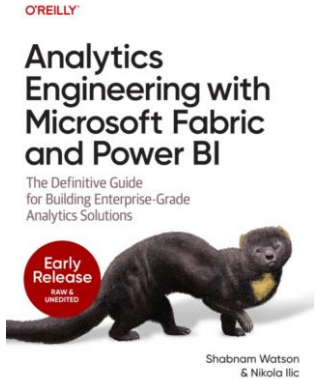
iLink Digital

data-mozart.com

learn.data-mozart.com

- *I'm making music from the data*!
- Power BI and Fabric addict, blogger, speaker...
- Father of 2, Barca & Leo Messi fan...





@DataMozart

Agenda

01 Power BI storage modes

02 Parquet/Delta format

03 Key concepts in Direct Lake

04 Two “flavors” of Direct Lake

05 Benefits & Limitations

06 When to use Direct Lake?

07 Security

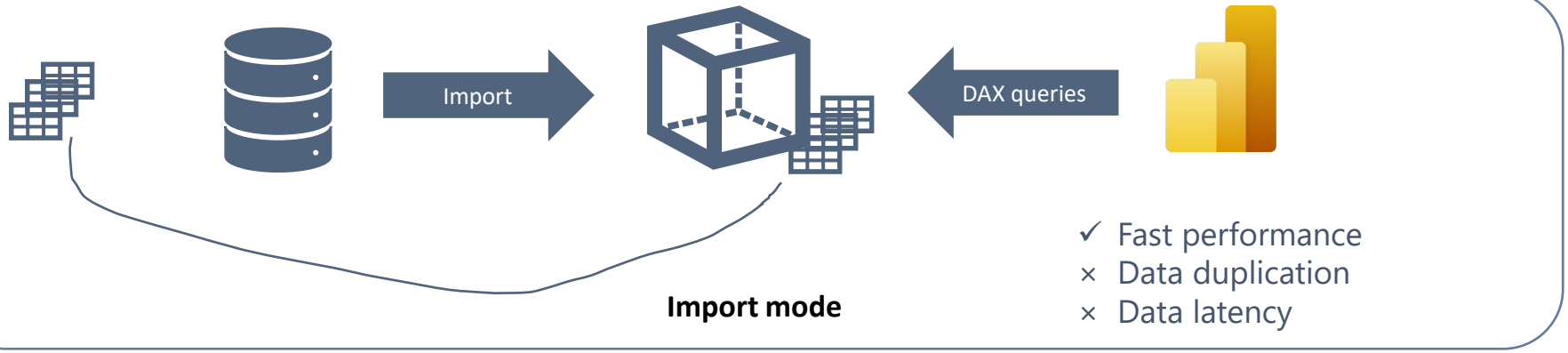
TODAY()

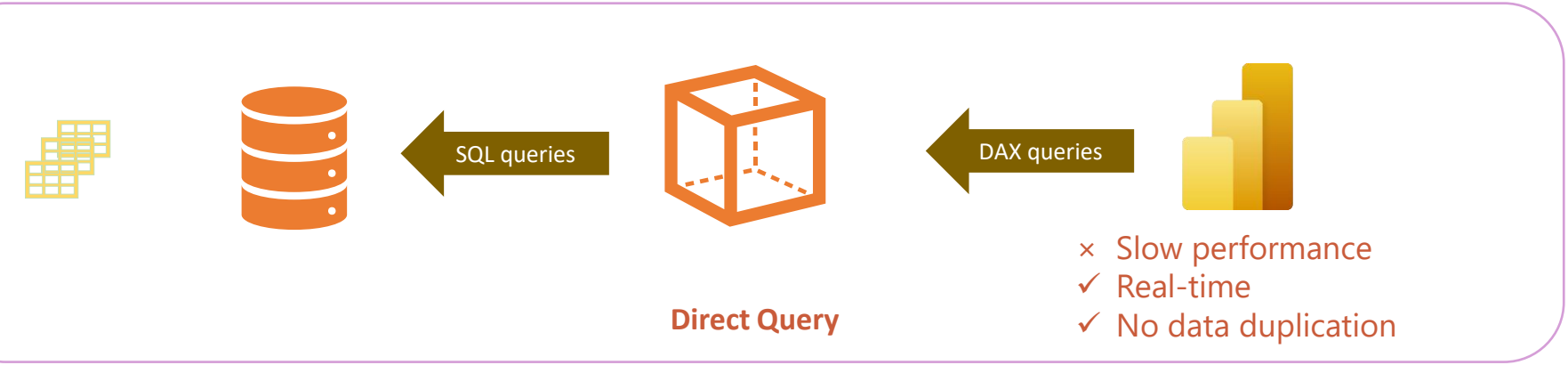
Everything is subject to change!

I’m a Power BI Professional…

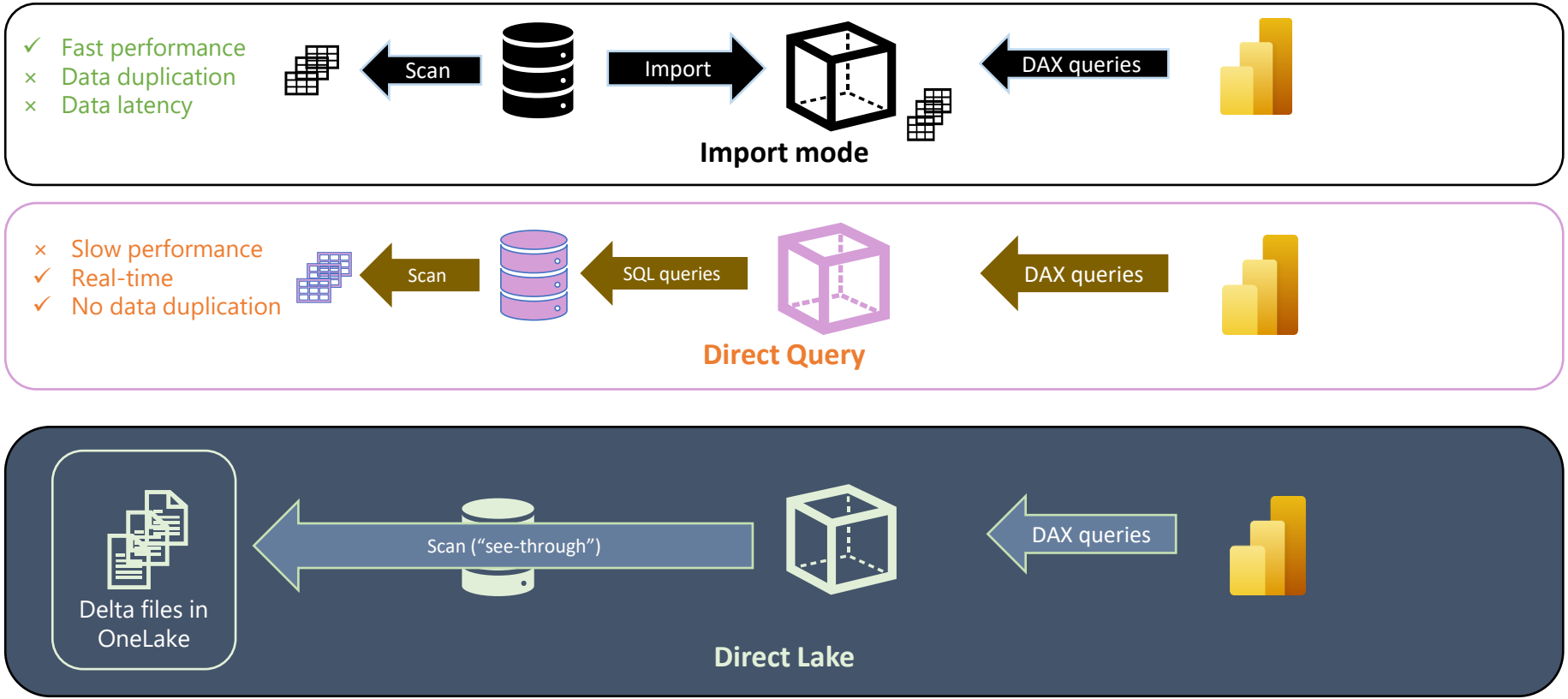
What should I do now?!

Power BI Architecture – Pre-Fabric





Power BI Architecture - Fabric



Direct Lake Prerequisites

- ✓ Fabric F Capacity/Power BI Premium
- ✓ Lakehouse/Warehouse
- ✓ Delta tables
- ✓ V-Ordering*

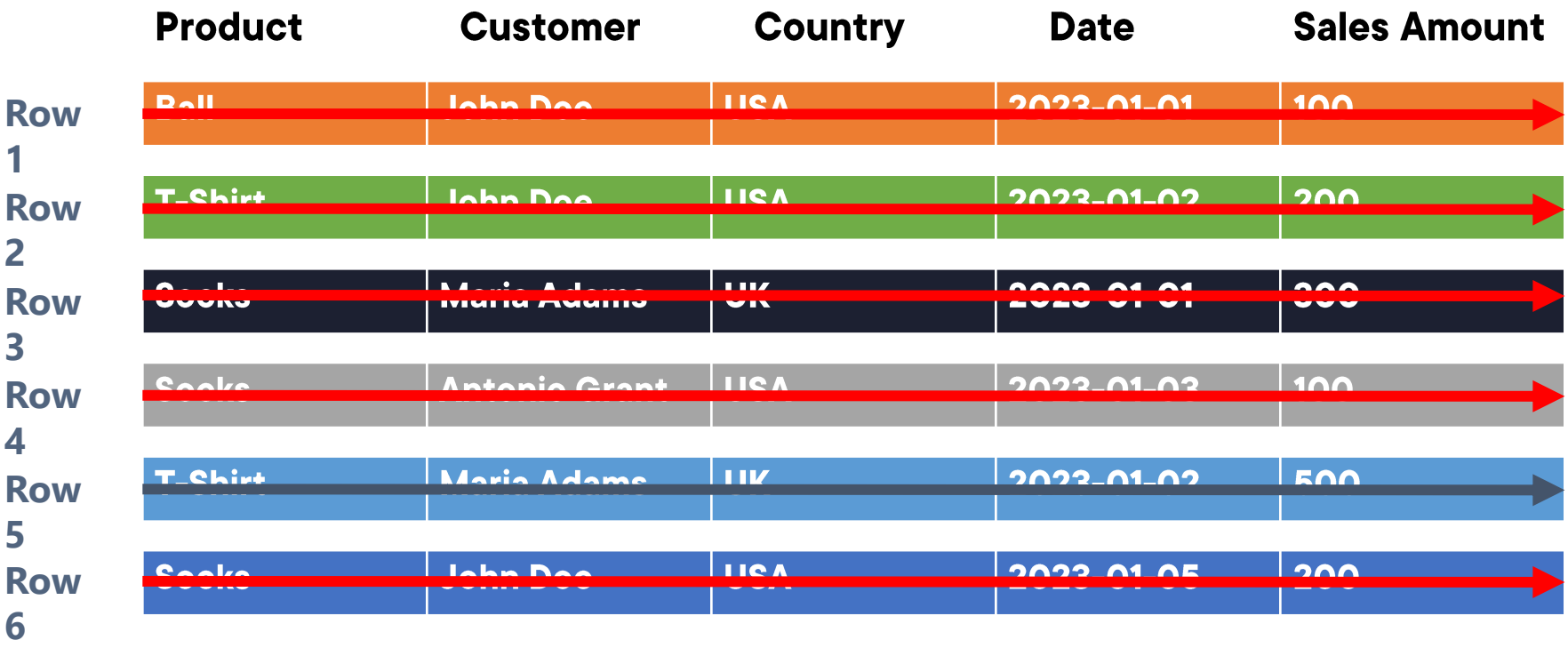
* V-Ordering
Fabric-specific way of additionally optimizing Parquet files when writing data

Dive directly into a lake full of Parquet & Delta files!

Row-Based Storage

	Product	Customer	Country	Date	Sales Amount
Row 1	Ball	John Doe	USA	2023-01-01	100
Row 2	T-Shirt	John Doe	USA	2023-01-02	200
Row 3	Socks	Maria Adams	UK	2023-01-01	300
Row 4	Socks	Antonio Grant	USA	2023-01-03	100
Row 5	T-Shirt	Maria Adams	UK	2023-01-02	500
Row 6	Socks	John Doe	USA	2023-01-05	200

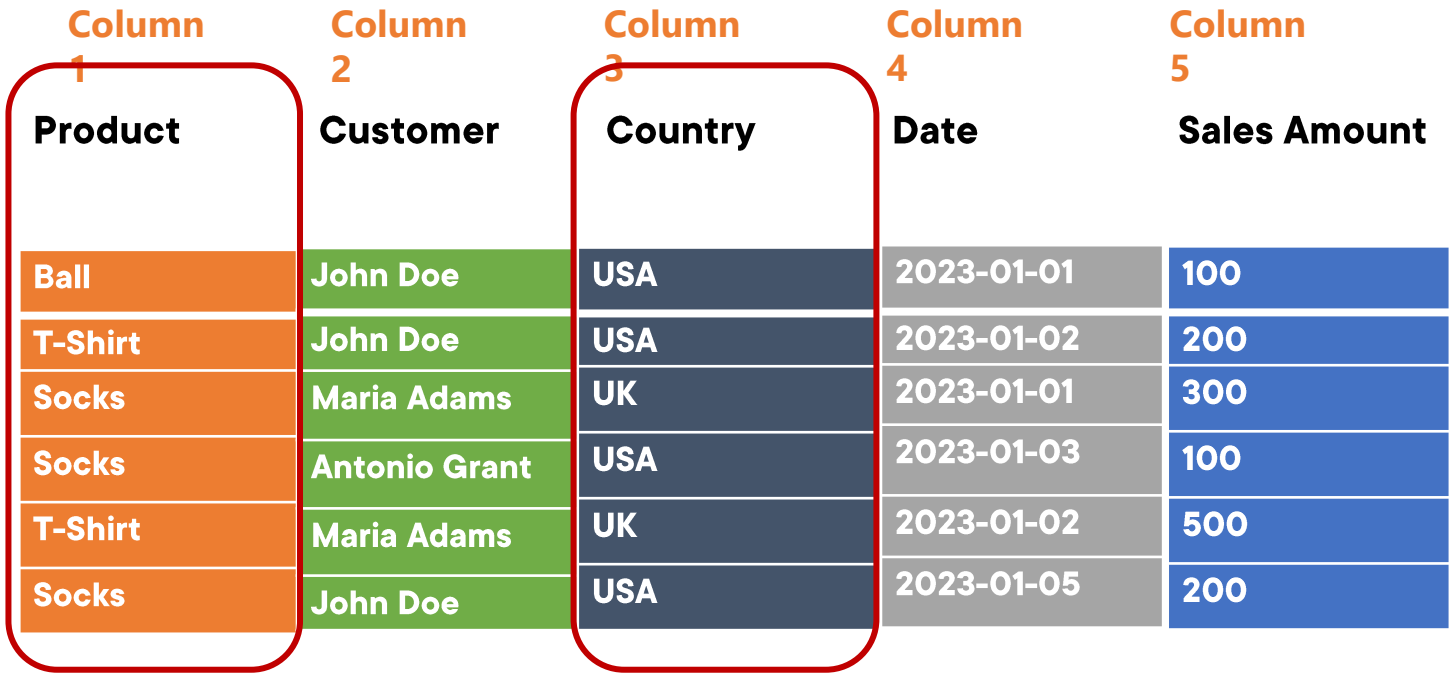
Row-Based Storage



Row-Based Storage

	Product	Customer	Country	Date	Sales Amount
Row 1	Ball	John Doe	USA	2023-01-01	100
Row 2	T-Shirt	John Doe	USA	2023-01-02	200
Row 3	Socks	Maria Adams	UK	2023-01-01	300
Row 4	Socks	Antonio Grant	USA	2023-01-03	100
Row 5	T-Shirt	Maria Adams	UK	2023-01-02	500
Row 6	Socks	John Doe	USA	2023-01-05	200

Column-Based Storage



Parquet Storage

	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	Country	Date	Sales Amount
Row group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row group 2	Socks	Maria Adams	UK	2023-01-01	300
	Socks	Antonio Grant	USA	2023-01-03	100
Row group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200

Projection and Predicate(s)

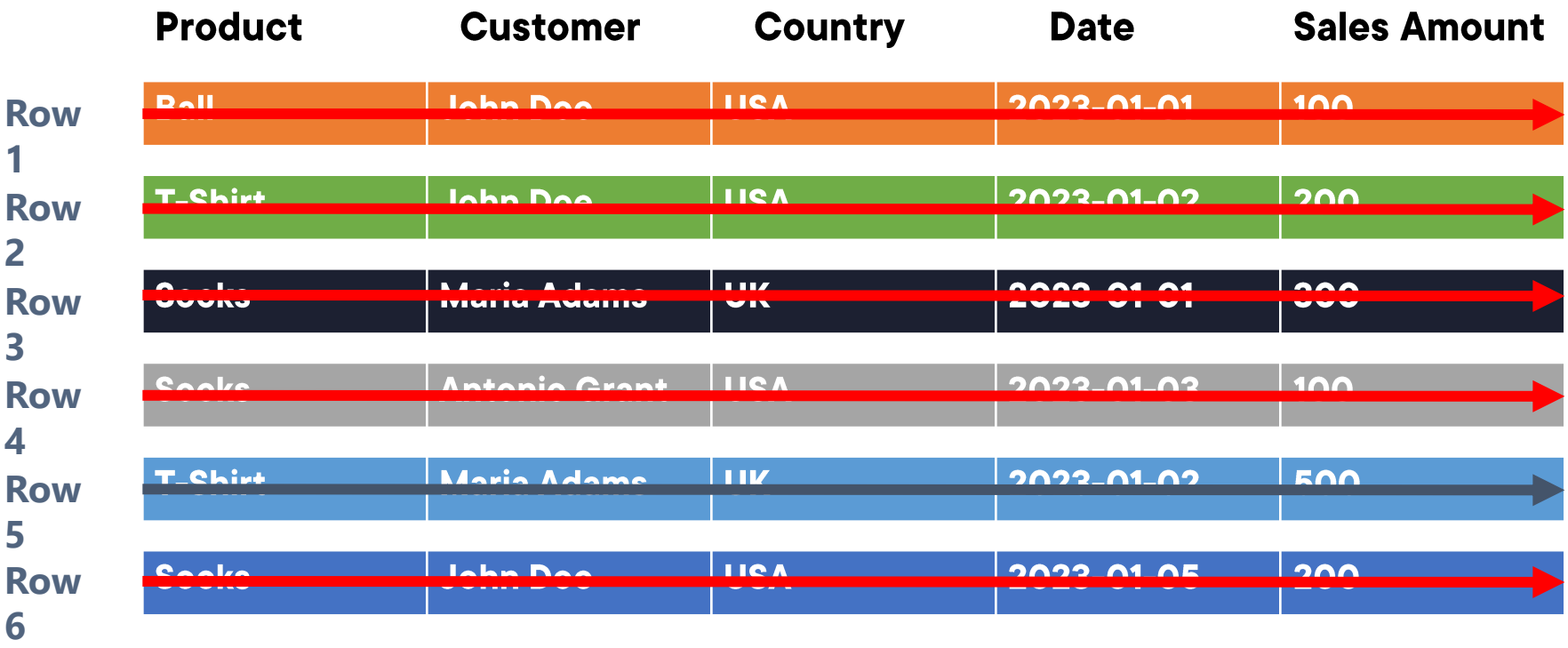
Projection = SELECT

Predicate(s) = WHERE

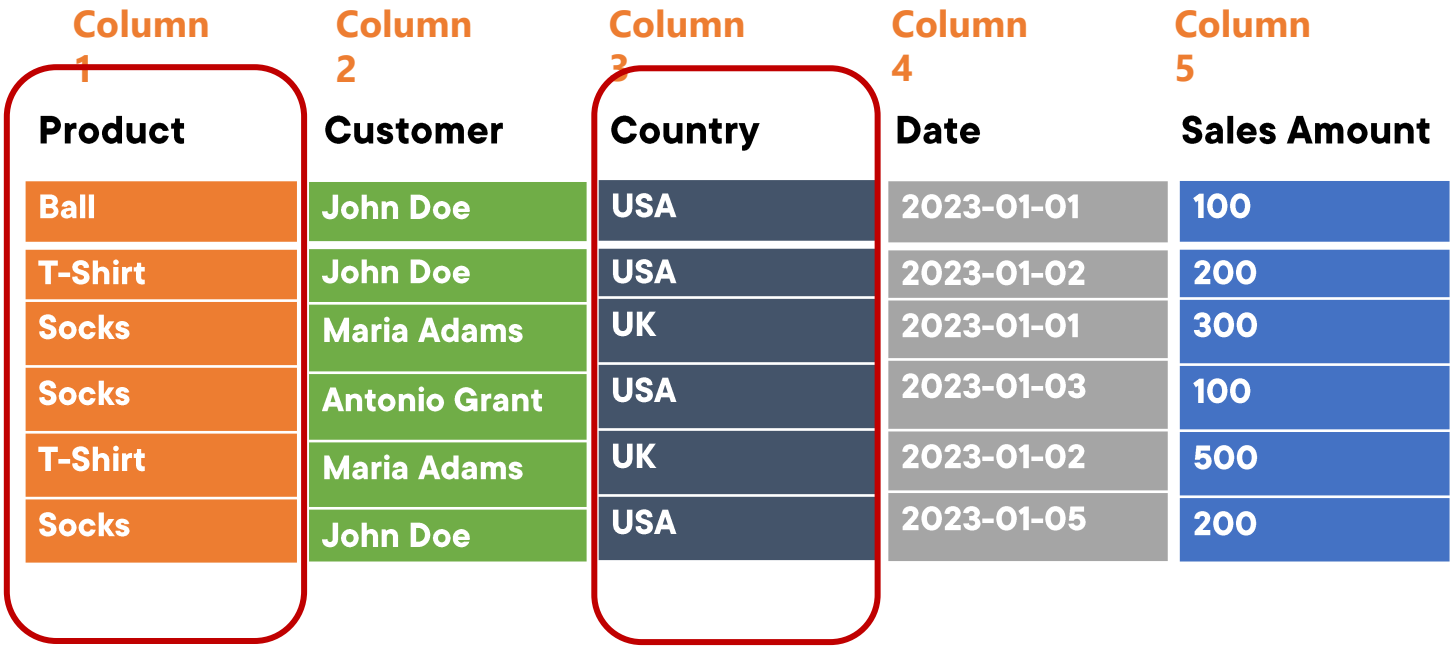
	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	**Country**	Date	Sales Amount
Row group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row group 2	Socks	Maria Adams	UK	2023-01-01	300
	Socks	Maria Adams	USA	2023-01-02	150
Row group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200

The engine will skip scanning these records!

Row-Based Storage: 5 Columns + 6 Rows



Column-Based Storage: 2 Columns + 6 Rows



Column Storage With Row Groups: 2 Columns + 4 Rows

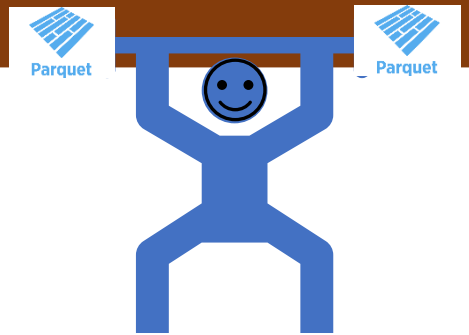
Column 1	Column 2	Column 3	Column 4	Column 5
Product	Customer	Country	Date	Sales Amount
Ball	John Doe	USA	2023-01-01	100
T-Shirt	John Doe	USA	2023-01-02	200
Socks	Maria Adams	UK	2023-01-01	300
Socks	Maria Adams	USA	2023-01-02	400
T-Shirt	Maria Adams	UK	2023-01-02	500
Socks	John Doe	USA	2023-01-05	200

The engine will skip scanning these records!

Can it be better than THIS?!

Delta format

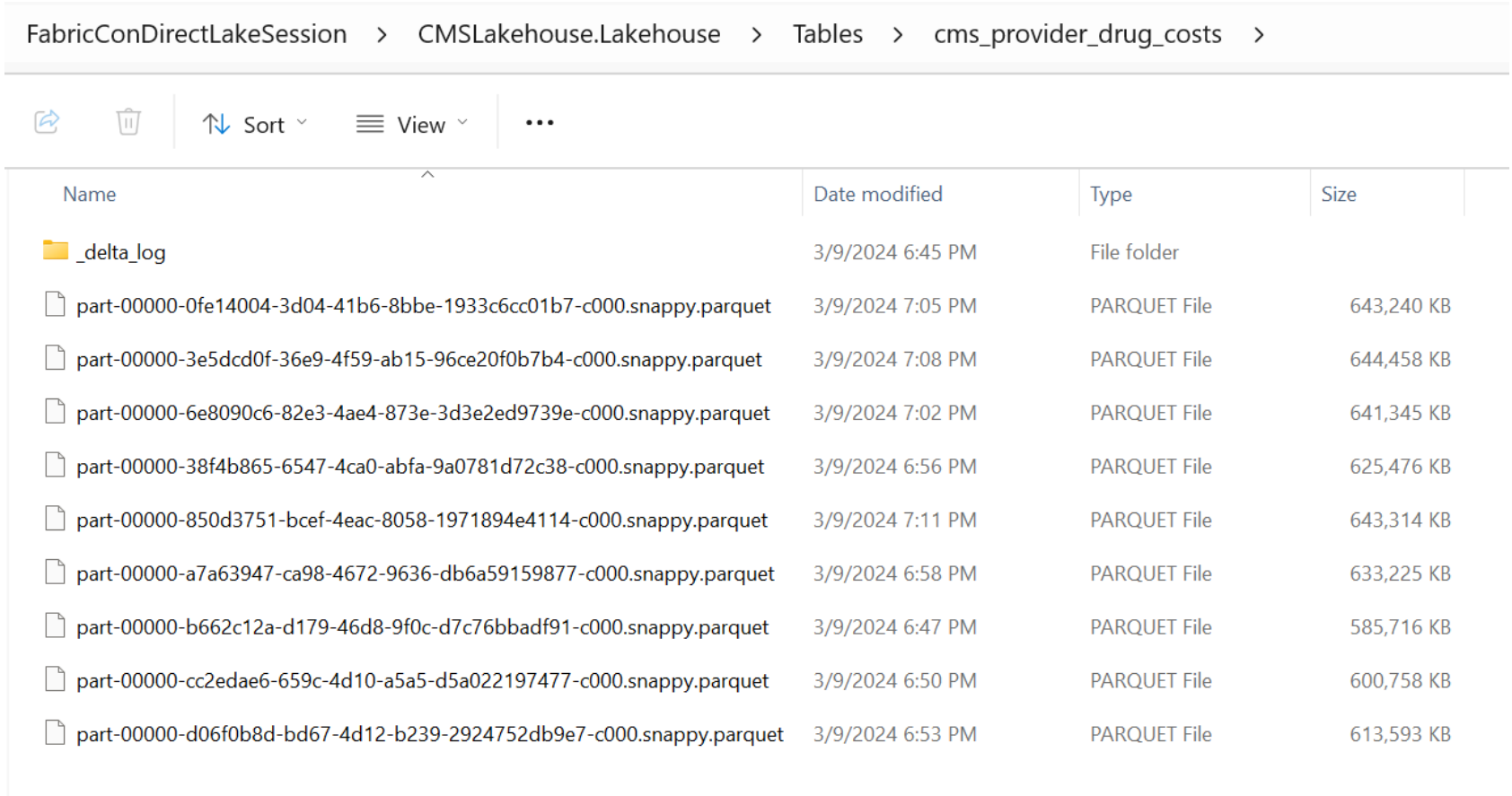
Parquet format on steroids!



- ✓ Versioning of Parquet files
- ✓ Stores transaction log
- ✓ Tracks all changes

DELTA LAKE

Delta File Format



_delta_log folder

FabricConDirectLakeSession > CMSLakehouse.Lakehouse > Tables > cms_provider_drug_costs > _delta_log >

Sort View ...

Name

- _temporary
- 00000000000000000000
- 00000000000000000001
- 00000000000000000002
- 00000000000000000003
- 00000000000000000004
- 00000000000000000005
- 00000000000000000006
- 00000000000000000007
- 00000000000000000008

```
1  %%sql
2
3  DESCRIBE HISTORY cms_provider_drug_costs
```

✓ 2 sec -Command executed in 2 sec 372 ms by shabnam watson on 7:18:58 PM, 3/09/24 Spark S

Spark jobs (1 of 1 succeeded) Log

Table Chart Search

	version	timestamp	userId	userName	operation
1	8	2024-03-10T00:11:01Z	NULL	NULL	WRITE
2	7	2024-03-10T00:08:03Z	NULL	NULL	WRITE
3	6	2024-03-10T00:05:03Z	NULL	NULL	WRITE
4	5	2024-03-10T00:02:01Z	NULL	NULL	WRITE
5	4	2024-03-09T23:59:00Z	NULL	NULL	WRITE
6	3	2024-03-09T23:56:02Z	NULL	NULL	WRITE
7	2	2024-03-09T23:53:06Z	NULL	NULL	WRITE
8	1	2024-03-09T23:50:13Z	NULL	NULL	WRITE
9	0	2024-03-09T23:47:25Z	NULL	NULL	WRITE

Back to ~~the future with~~ Microsoft Fabric!

How does this ~~architecture~~ magic work?

Parquet is very similar to how Power BI stores data in import mode (in VertiPaq)

Power BI engine reads from Delta tables directly

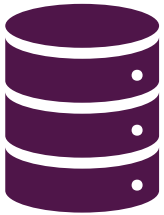
Semantic model is initially empty

Loads data into memory on-demand

Two Types of Direct Lake

OneLake

- Multi-source
- No DirectQuery fallback
- Created in Power BI Desktop



SQL

- Single-source
- DirectQuery fallback available
- Created in Fabric Web UI

How to Create a Direct Lake Model?

Power BI Desktop

- DL on OneLake
- You can ***edit*** the DL on SQL models
- The model changes are applied in real time to the model “published” in the Service

Fabric Web UI

DL on OneLake	DL on SQL
• Create -> OneLake catalog • From the Lakehouse explorer -> New semantic model • OneLake catalog from web modeling • Workspace -> New item -> Semantic model	• From the SQL EP of the Lakehouse or from the Warehouse -> New semantic model

How to Check the Direct Lake "flavor"?

1 TMDL View

```
468   expression 'DirectLake - CMS_LH' =
469       let
470           Source = AzureStorage.DataLake("h
                  ",
              [HierarchicalNavigation=true])
471       in
472           Source
473   lineageTag: 3dc120bc-ec3e-4745-8c94-34d8b80166a5
474
```

DL on OneLake

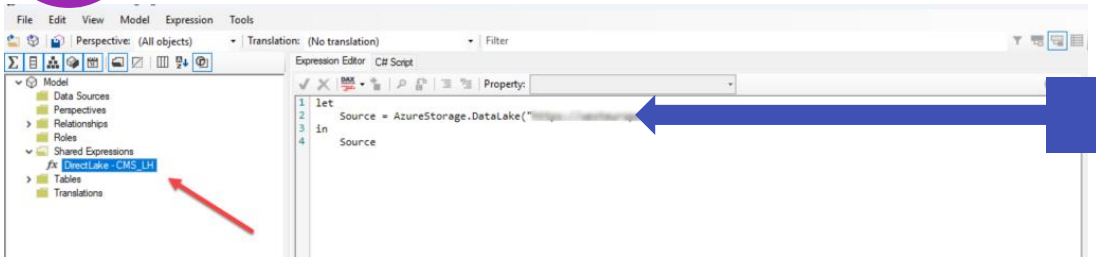
Apply | Preview

```
1   createOrReplace
3       model Model
541         cultureInfo en-US
543         linguisticMetadata = {"Version":"1.0.0","Language":"en-US"}
544             contentType: json
545
546         expression DatabaseQuery =
547             let
548                 database = Sql.Database("
                    fabric.microsoft.com", "                    ")
549             in
550                 database
```

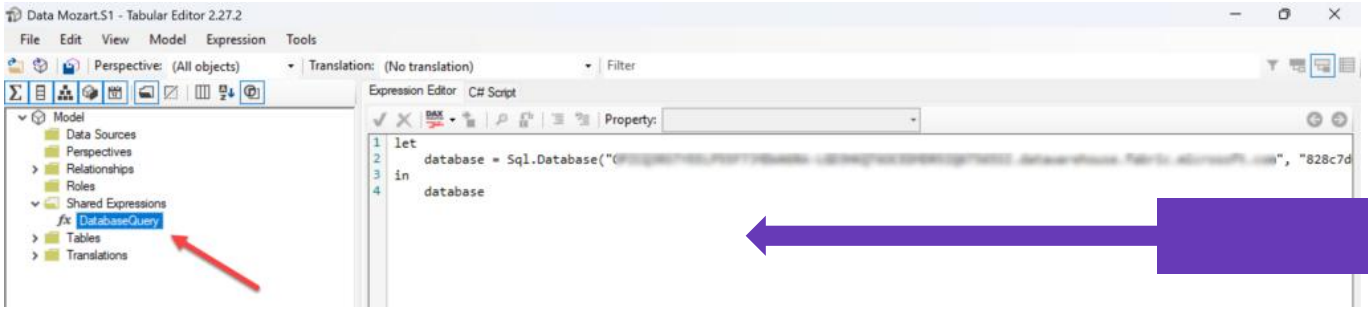
DL on SQL

How to Check the Direct Lake "flavor"?

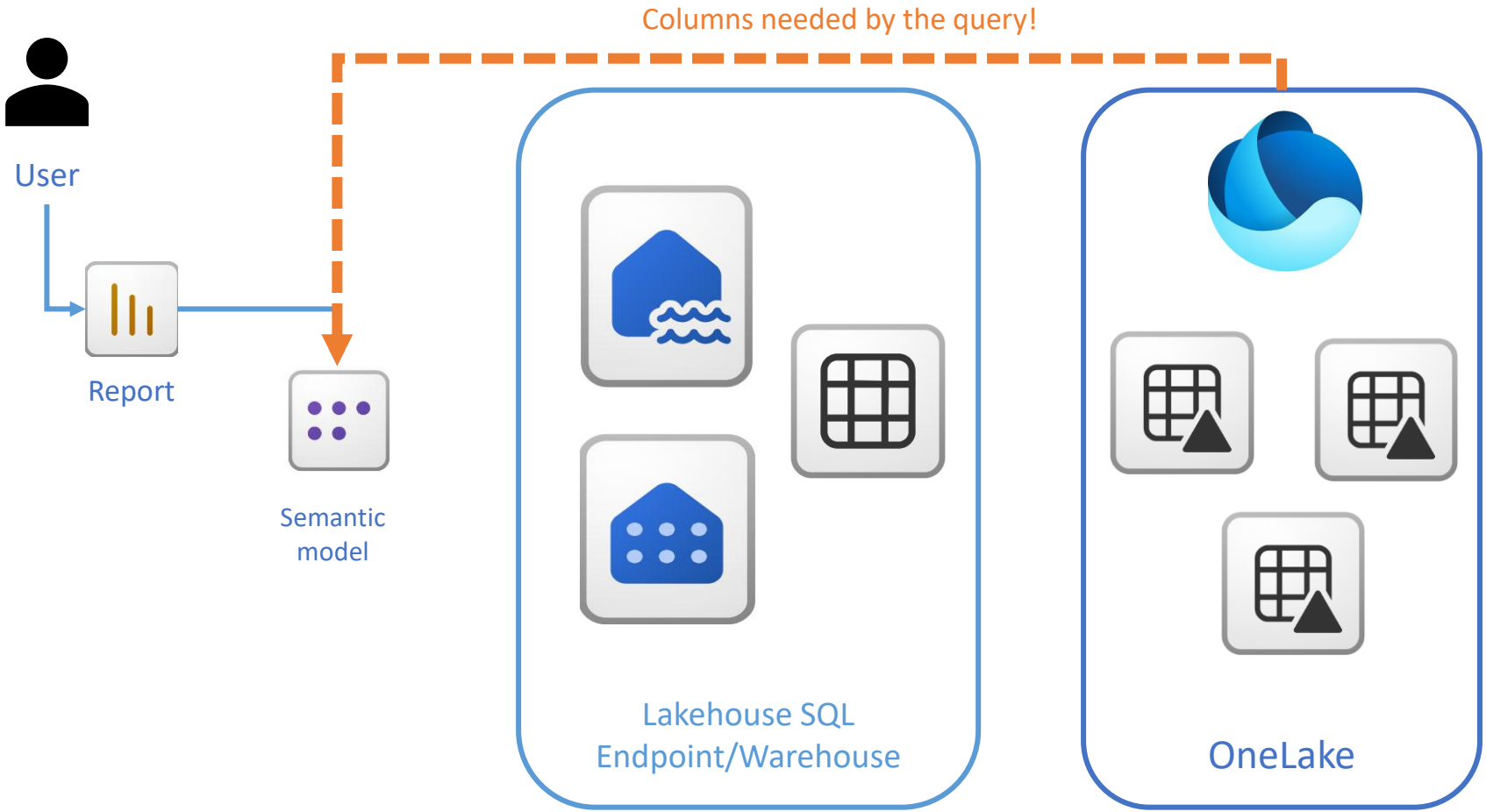
2 Tabular Editor



DL on OneLake



DL on SQL


Columns needed by the query!
User
Report
Semantic model
Lakehouse SQL Endpoint/Warehouse
OneLake

Columns needed by the query!

User

Report

Semantic model

FirstName	Amount
John	100
Jane	500

Customer

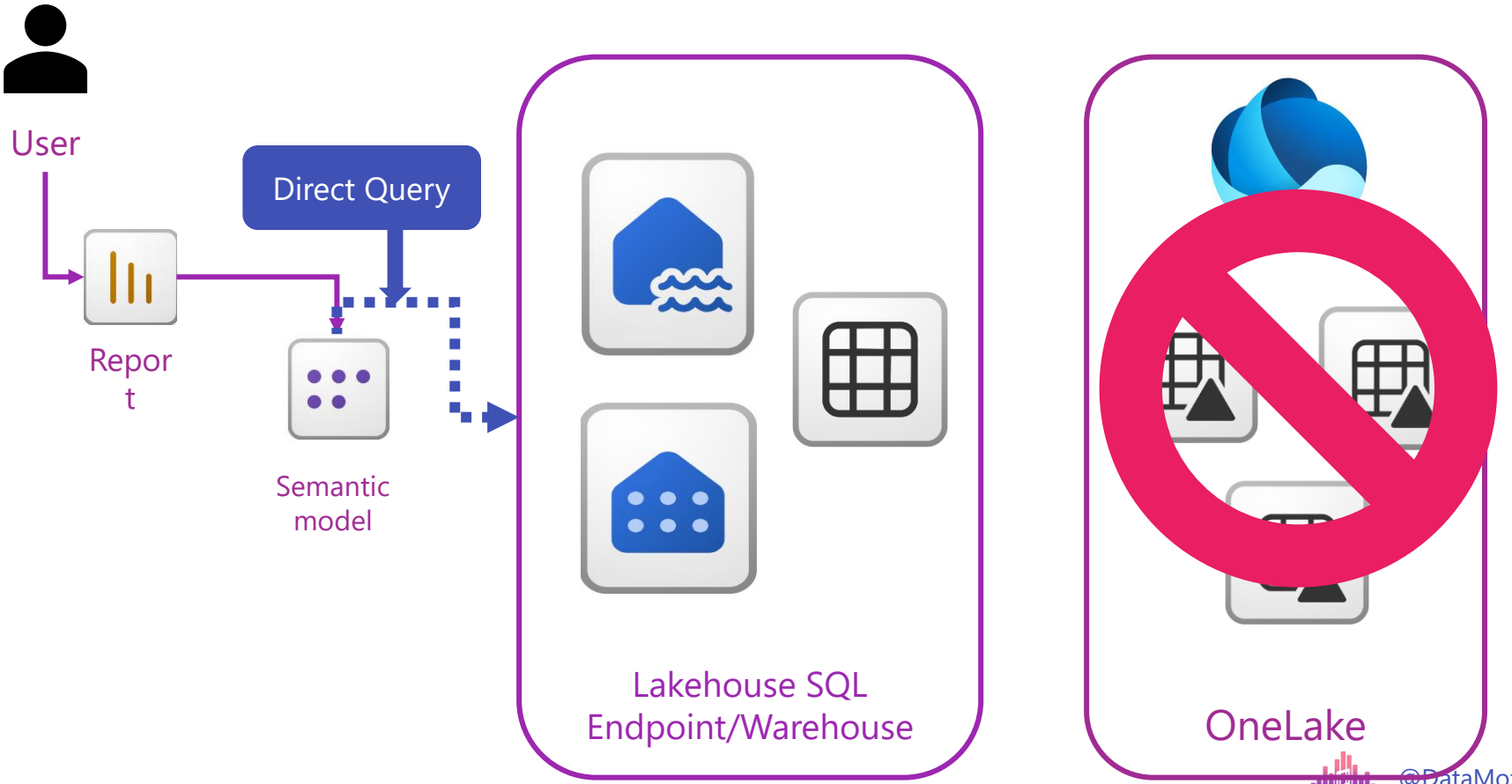
CustomerKey	FirstName	LastName	Email	Address
1	John	Doe	jd@email.com	Street 1
2	Jane	Doe	Jad@email.com	Street 2

Sales

SalesKey	CustomerKey	ProductKey	Date	Amount
1	1	123	2024-01-01	100
2	2	456	2024-02-01	200
3	2	789	2024-03-01	300

The entire column is loaded into memory!!!

What if Direct Lake doesn't work?



Direct Lake Capacity Requirements

Fabric SKU	Parquet files per table	Row groups per table	Rows per table (millions)	Max model size on disk/OneLake (GB)	Max memory (GB) [1]
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

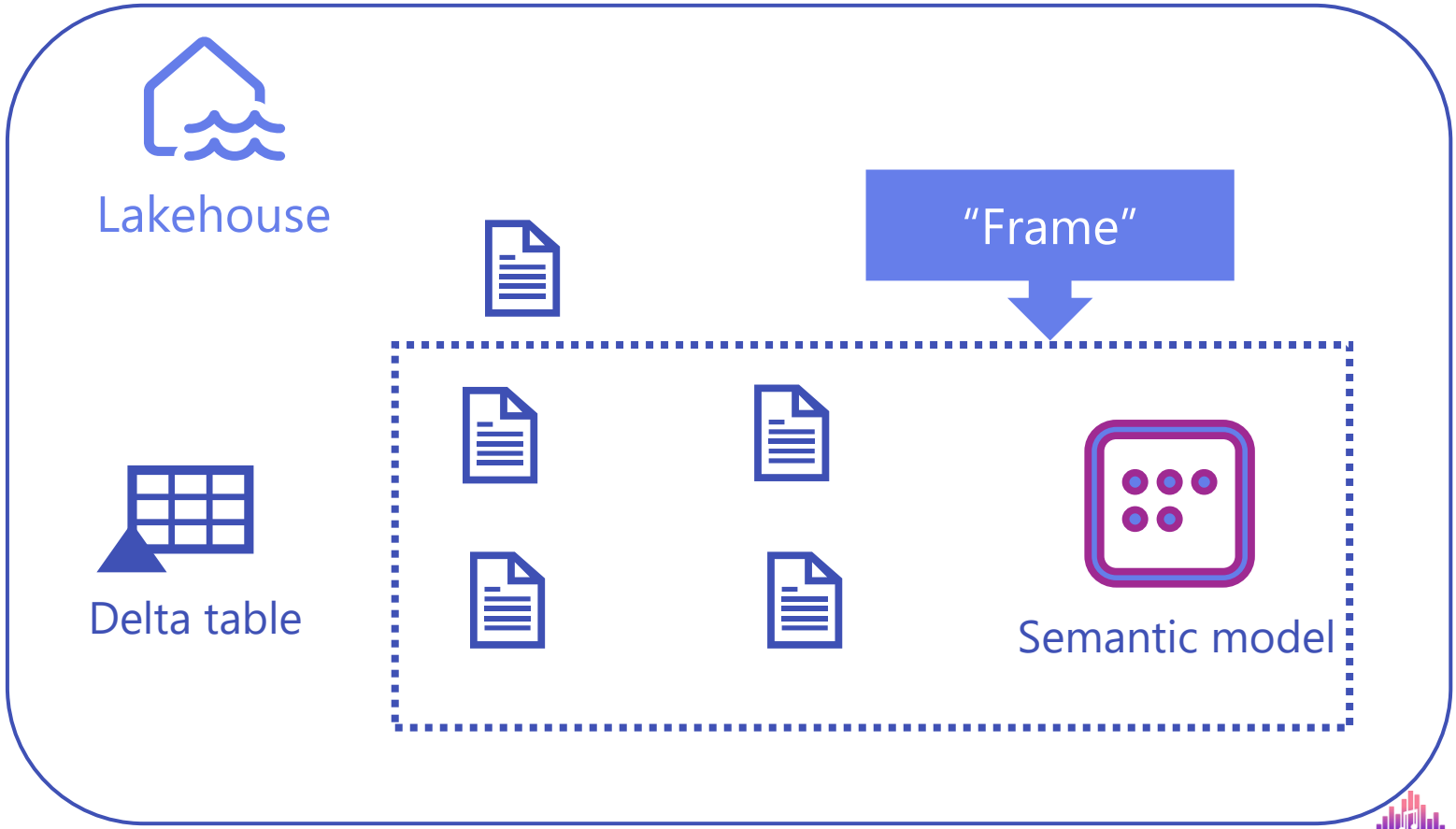
- ✓ Max Memory = memory resource limit for how much data can be paged in
- ✓ No fallback to DirectQuery mode
- ✓ Can have a performance impact if the amount of data is large enough to cause excessive paging in and out of the model data from the OneLake data.

Learn about Direct Lake in Power BI and Microsoft Fabric - Power BI | Microsoft Learn

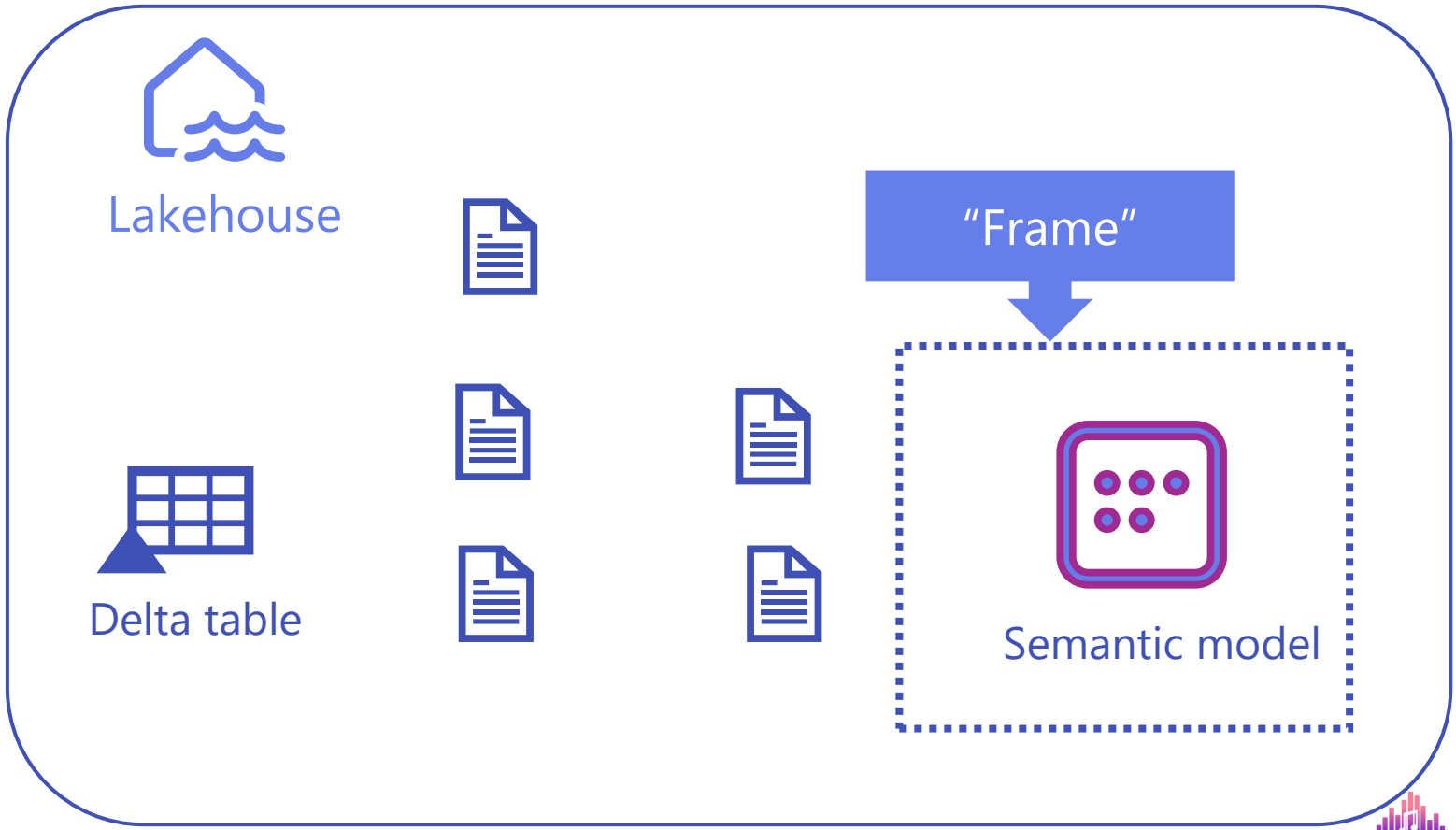
Direct Lake – Data Refresh

- Called internally as “framing”
- A metadata operation to read the latest Parquet schema
- It doesn’t copy (or refresh) any actual data!

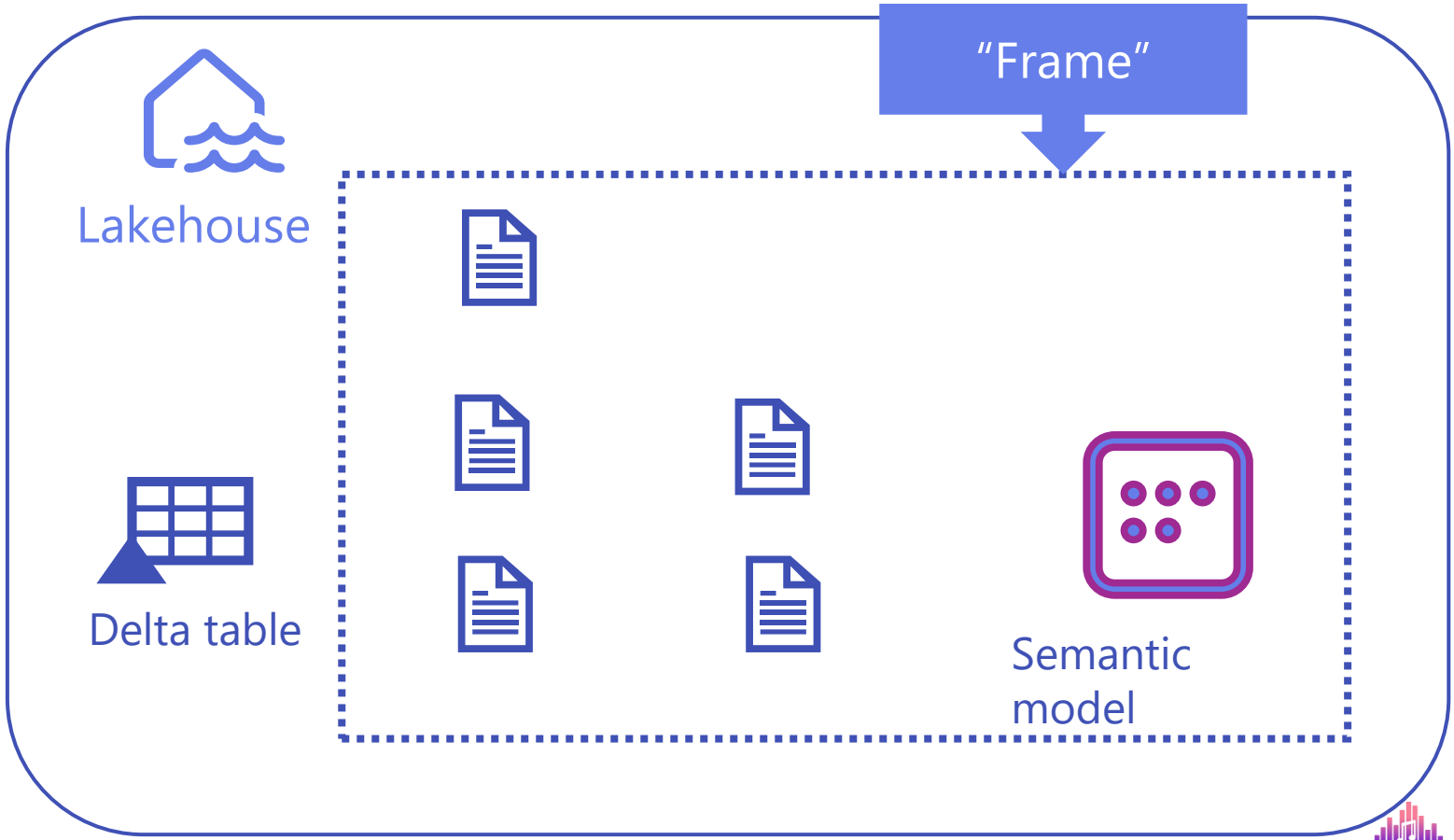
Direct Lake Refresh (AKA “Framing”)



Direct Lake Refresh (AKA “Framing”)

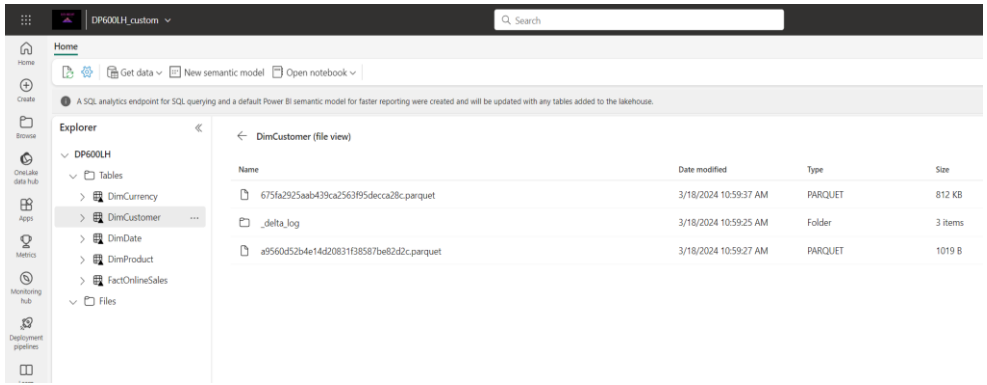


Direct Lake Refresh (AKA “Framing”)

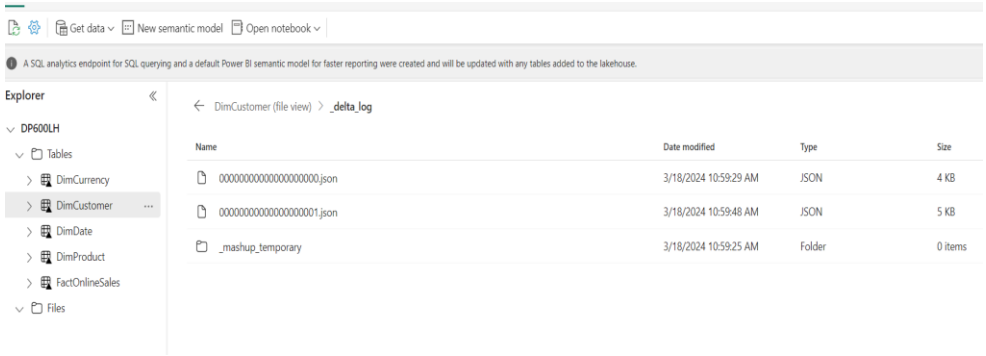


Direct Lake Refresh (AKA “Framing”)

Lakehouse

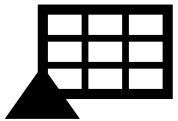


DimCustomer

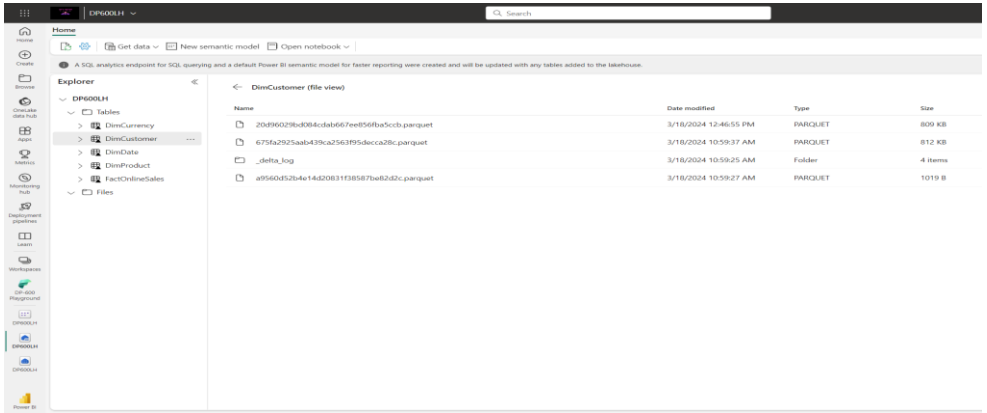


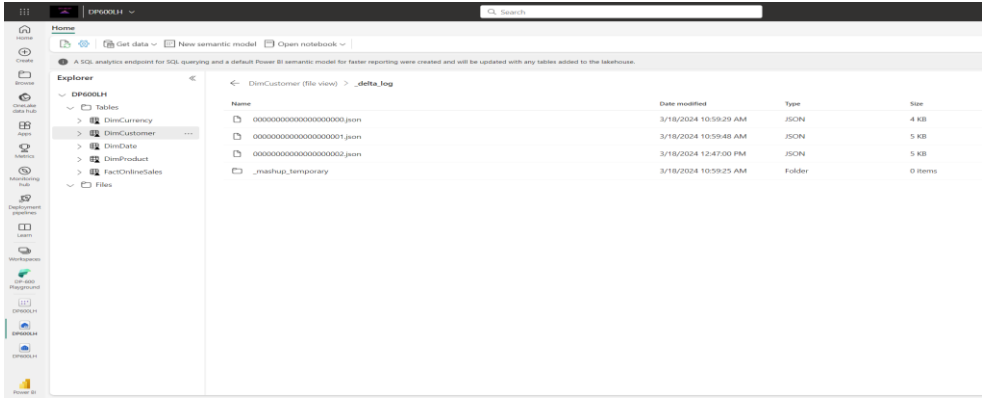
Direct Lake Refresh (AKA “Framing”)

Lakehouse

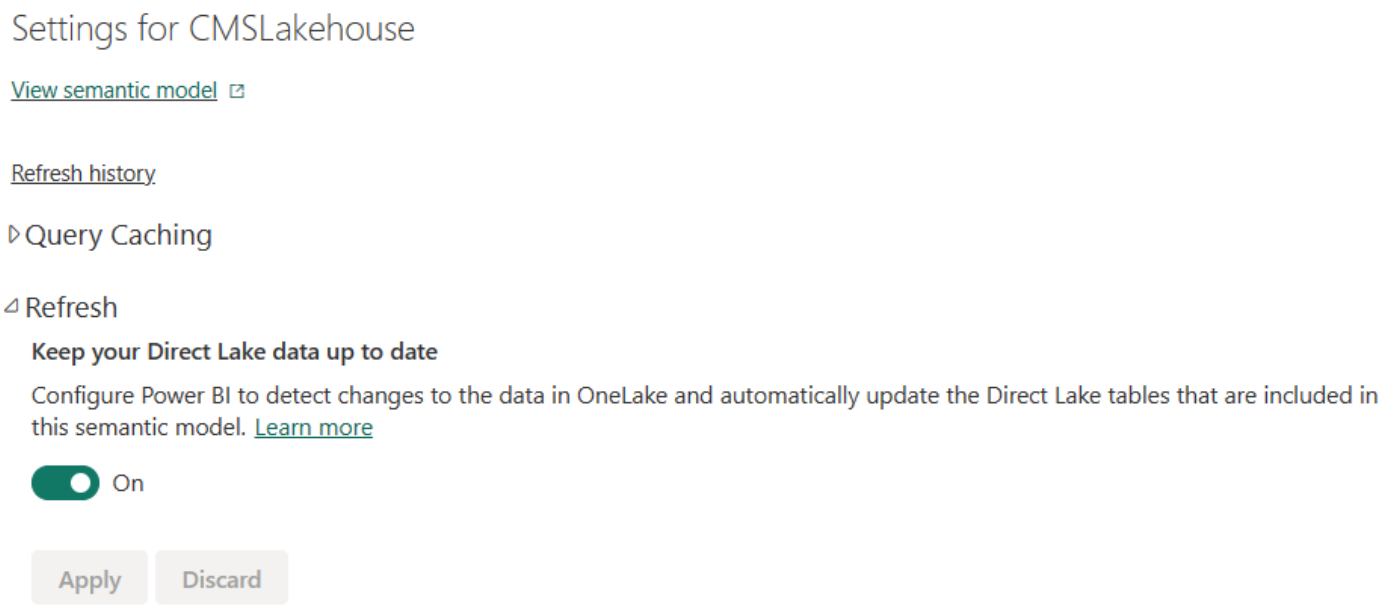


DimCustomer





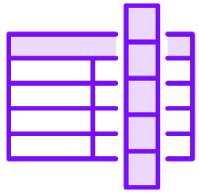
Refresh option for semantic models



Framing

Adding the info about the latest "version" of the data to a semantic model

Transcoding



Loading columns needed by query in cache memory

Temperature

Keep frequently used columns in cache memory

Performance

What Affects Performance?

- ✓ V-Order
- ✓ Data types
- ✓ Segment size and count
- ✓ Column cardinality
- ✓ Indexes and aggregations
- ✓ DirectQuery fallback
- ✓ Degree of memory residency
- ✓ Memory pressure

Degree of Memory Residency

- **Cold** – all data must be loaded from Delta tables
- **Semiwarm** – DL only drops column segments during framing that belonged to removed row groups -> only updated/newly added data loaded
- **Warm** – column data already fully loaded into memory
- **Hot** – data loaded into memory, VertiScan caches populated -> queries hit the caches

Memory Residency

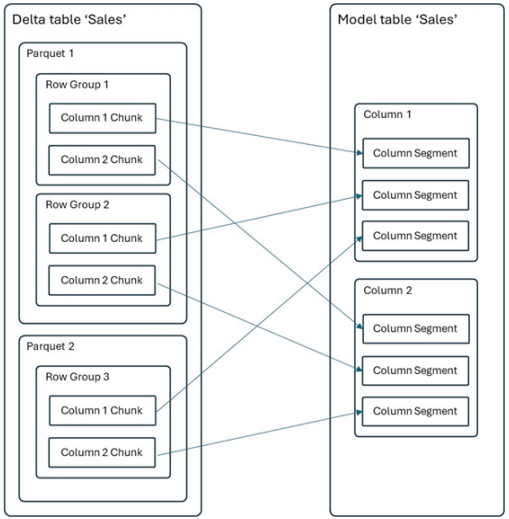
1. VertiPaq dictionary transcoding
2. Loading Parquet column parts into column segments
3. Generating join indexes
4. Applying Delta deletion vectors

Incremental Framing

- DL analyzes the Delta log of each Delta table -> drops loaded segments and join indexes **only when the underlying data has changed**
- **Dictionaries** retained -> new values simply added to existing dictionaries
- Reduces the reload burden and helps cold query performance

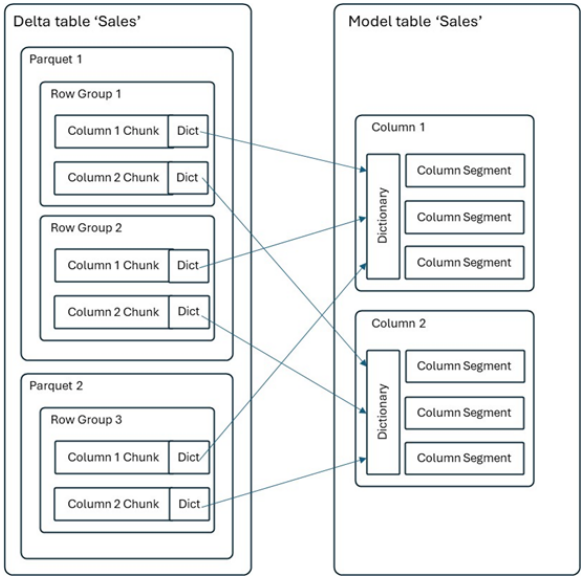
Delta Table Layout

- Direct relationship between row group number in the Delta table and the segment number for each semantic model column



Delta Table Layout

- Local dictionaries vs. global dictionary



Delta Table Update Patterns

- Overwrite – truncates the Delta log -> **NO INCREMENTAL FRAMING!!!**
- Batch processing without vs. with **partitioning**
- **Incremental loading**
- **Stream processing ->** many small files/row groups -> worse performance

Delta Table Maintenance

- **VACUUM** - removes unreferenced Parquet files -> **doesn't affect DL performance!**
- **OPTIMIZE** – merges small files into larger ones -> **affects cold DL performance!**
  - Optimize infrequently
  - Optimize more frequently if small Parquet files accumulate quickly
- Ideal row group size = **1 – 16 million rows** -> **don't change the default size!**

Append-only pattern + OPTIMIZE

Performance Optimization Checklist

- Avoid destructive Delta table updates to preserve incremental framing
- Target 1-16 million rows row group size -> DL likes large column segments
- Avoid partitioning high-cardinality columns
- DL doesn't use Delta/Parquet statistics for row group/file skipping when loading the data

Limitations of Direct Lake

- T-SQL Views are not supported (will fall back to DirectQuery)
- DAX queries exceeding limits or using unsupported features fall back to DirectQuery mode
- No DAX calculated columns yet (except Calculation groups and Field parameters)

Always check the list of current limitations!

Framing
=
Refreshes METADATA ONLY!

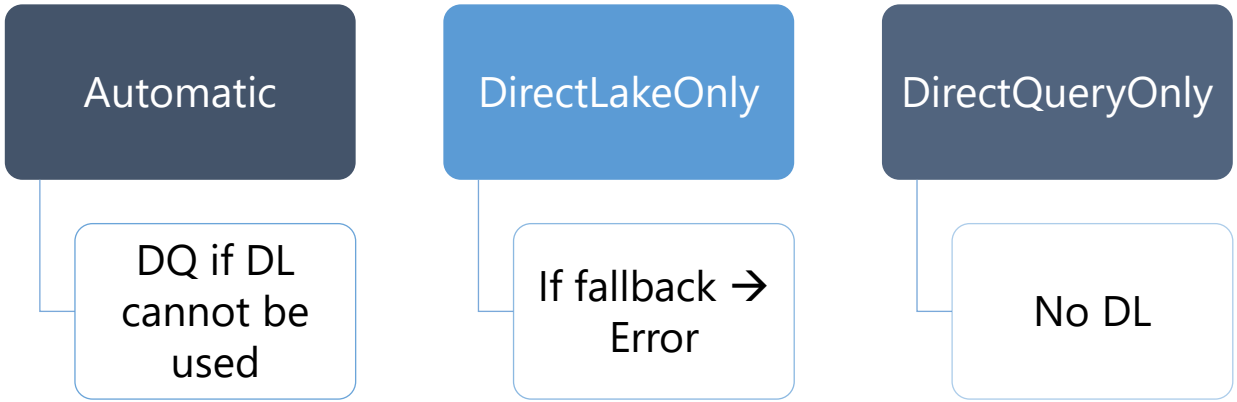
DEMO

- Import vs. Direct Lake
- **Hot ‘n’ cold**

Fallback to Direct Query*

**Only for Direct Lake on SQL models*

DirectLakeBehavior Property



Security

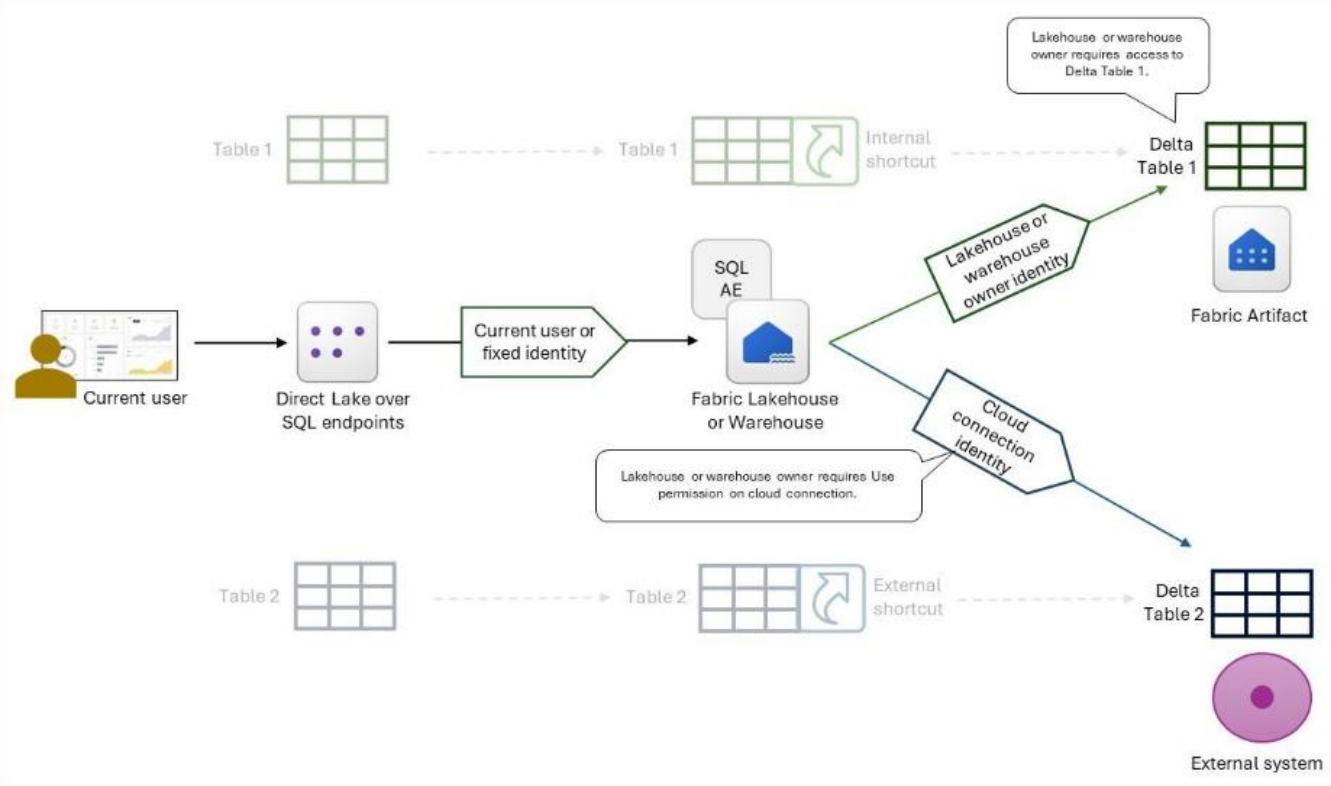
Workspace Access

- Contributors, Members, and Admins can automatically read OneLake data
- Viewers get access via combination of item permissions, compute permissions, or OneLake security roles

Things to Consider

- **DL on SQL** checks permissions via the SQL endpoint
  - User doesn't need permission to read Delta table directly in OneLake
  - It's enough to have read access to the Fabric item (LH), and SELECT permission on a table via SQL endpoint
- **DL on OneLake** uses OneLake security or requires Read and ReadAll permissions on the lakehouse
- **DL owners** – model owner must have read permission on the source tables

Using Shortcuts



OLS/RLS

- Can be defined both at the Delta table level (OneLake security) and/or semantic model level -> different behavior between DL on OneLake vs DL on SQL!
- Different scopes
  - OneLake security affects all Fabric engines
  - OLS/RLS on the model affects the model only
  - OneLake security has a precedence -> when both OneLake Security OLS/RLS and Direct Lake OLS/RLS used, users who have OneLake access can still retrieve and work with the data, even if Direct Lake model rules further restrict data

Security Checklist

- Assign workspace viewers OneLake security roles that grant *read* access to source Fabric items
- If a source item has shortcuts to another Fabric item, the user also needs *read* access to each shortcut's target Fabric item
- Use fixed identity to isolate users from a source Fabric item
- Bidirectional relationships are not supported if the source Fabric item relies on OneLake RLS
- Currently, only static RLS on a single table is supported
- Consolidate OneLake security RLS and OLS permissions into one role per user instead of assigning multiple roles

Final thoughts…

I'm a Power Query Guru...

What should I do now?!

No Power Query in Direct Lake models!

Dataflows Gen2

What about RLS?!

When defined on the semantic model

When enforced on a table queried via SQL Endpoint

Limitations of Direct Lake

- T-SQL Views are not supported (will fall back to DirectQuery with DL on SQL)
- DAX queries exceeding limits or using unsupported features
- No DAX calculated columns yet (except Calculation groups and Field parameters)

Always check the list of current limitations!

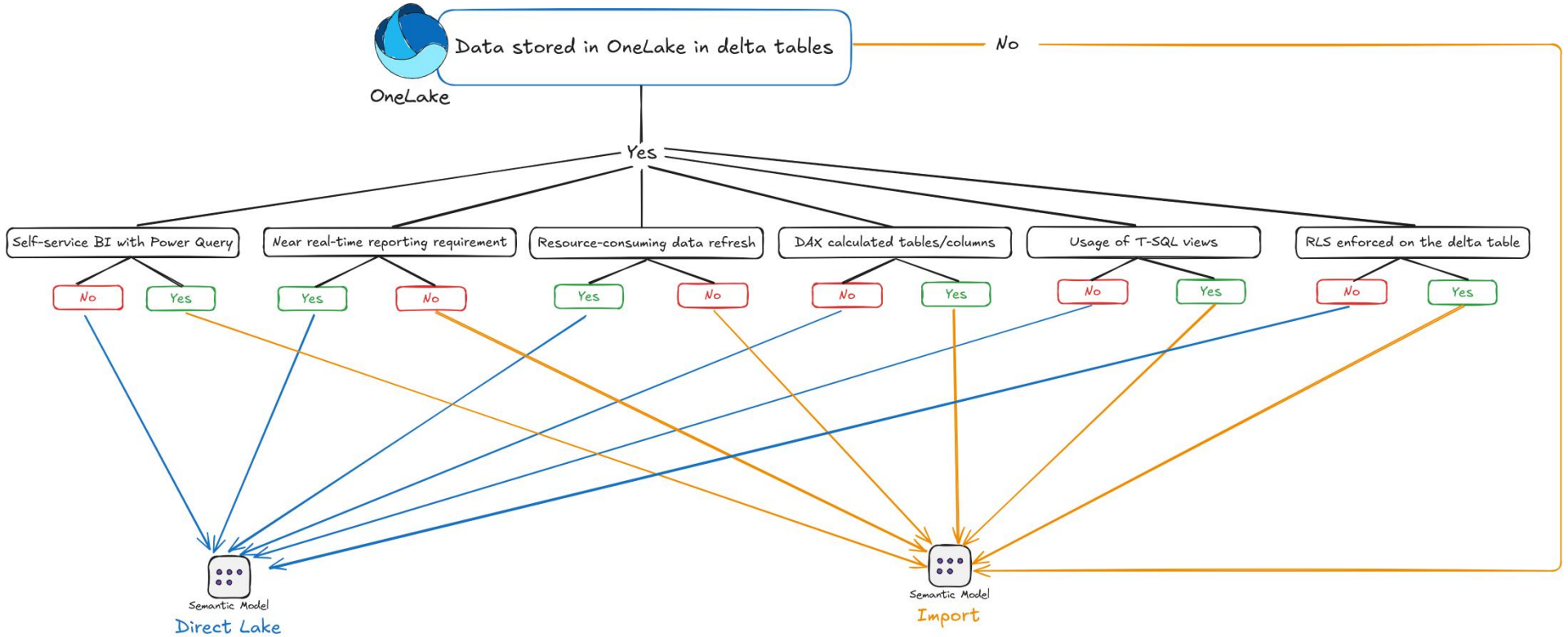
Direct Lake Considerations

Consideration	Direct Lake on OneLake	Direct Lake on SQL
RLS enforced via SQL endpoint	Supported	Supported -> DirectQuery fallback
Use T-SQL views	Not supported	Supported -> DirectQuery fallback
Composite models	Supported	Not supported
Deployment Pipeline rules (data source)	Not supported directly -> create a parameter expression to use in the connection string	Supported

Benefits of Direct Lake

- ✓ Performance comparable to Import mode
- ✓ Eliminating the serving layer (Azure SQL DB, Azure Synapse…) saves costs
- ✓ Refreshes in Import mode may use a lot of CUs

Import vs. Direct Lake



To Wrap Up…

- ✓Direct Lake **IS** a fantastic feature!
- ✓Direct Lake **IS NOT** a one solution to “rule-them-all”!
- ✓If you’re happy with your existing Import models – don’t switch them to Direct Lake (yet)!
- ✓Primary choice for “Greenfield” lake-centric solutions?

We are ALL still learning when to use Direct Lake☺

Loved it? Learned something? Tell us!
Share your feedback in just 1 minute

50 Shades of Direct Lake

PBIG
POWER BI GEBRUIKERSGROEP




Thank you

Nikola Ilic

nikola@data-mozart.com

www.data-mozart.com