Fabric
I love you,
I love you not

Riccardo Perico
BI & Power BI Engineer @Lucient Italia

Sponsor

Saturday
innova
community

altitudo

beanTech
IT moves your business

OVERNET.
upgrade your digital skills

[stesi]
Powered by Innovation

Bio

Riccardo Perico
BI & Power BI Engineer

riccardo-perico

R1k91

R1k91

riccardo-perico

riccardo-perico

In this session I'll try **not to "simply explain features"**,
main goal is to **give a feedback** on them,
based on my personal experience with Fabric
doing tests and POCs.

DISCLAIMER

What to expect from this session?

- **Discover** things
- **Enforce** knowledge
- If you enter that door with doubts… will you get an **answer**?

  Maybe

  Maybe not…

- But I hope you'll get the **right doubts**

D E A D

DISCLAIMER

Maybe I’m wrong, or you’ve a different point of view…

Share your experience while we go, if you please

DISCLAIMER

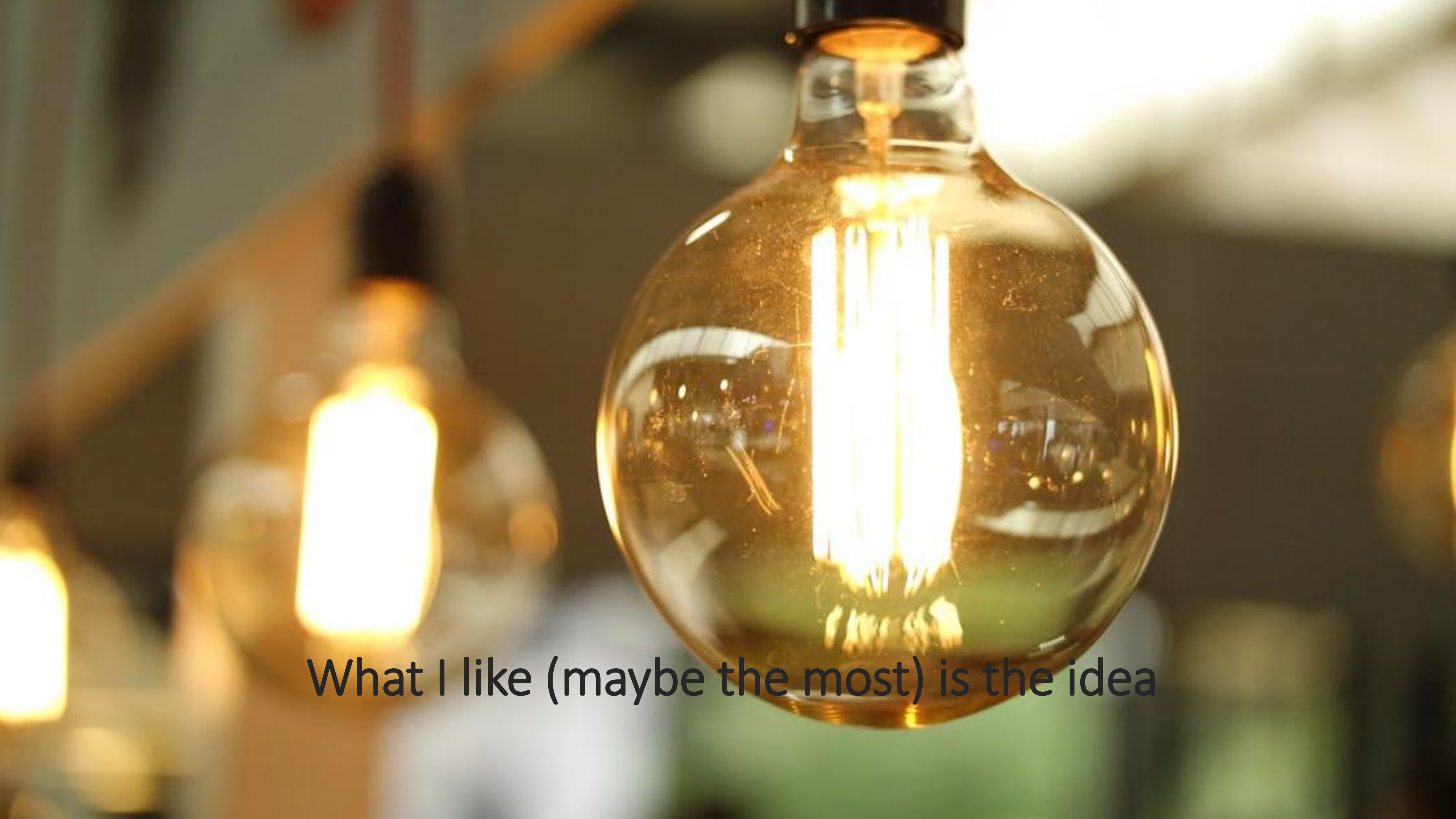
I tried to keep my session **as aligned as possible with all the recent announcements**.

Please forgive me if I lost something.

I feel this session very ambitious,

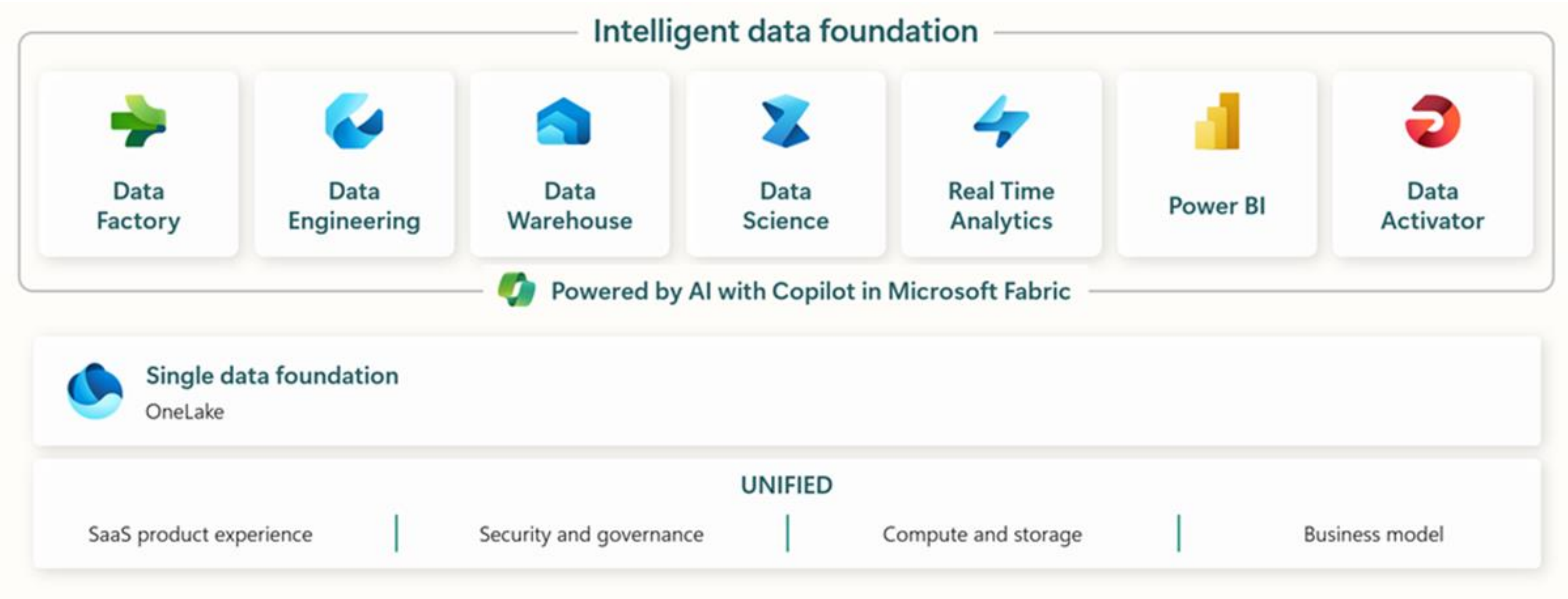
I hope to met your expectations.

DISCLAIMER

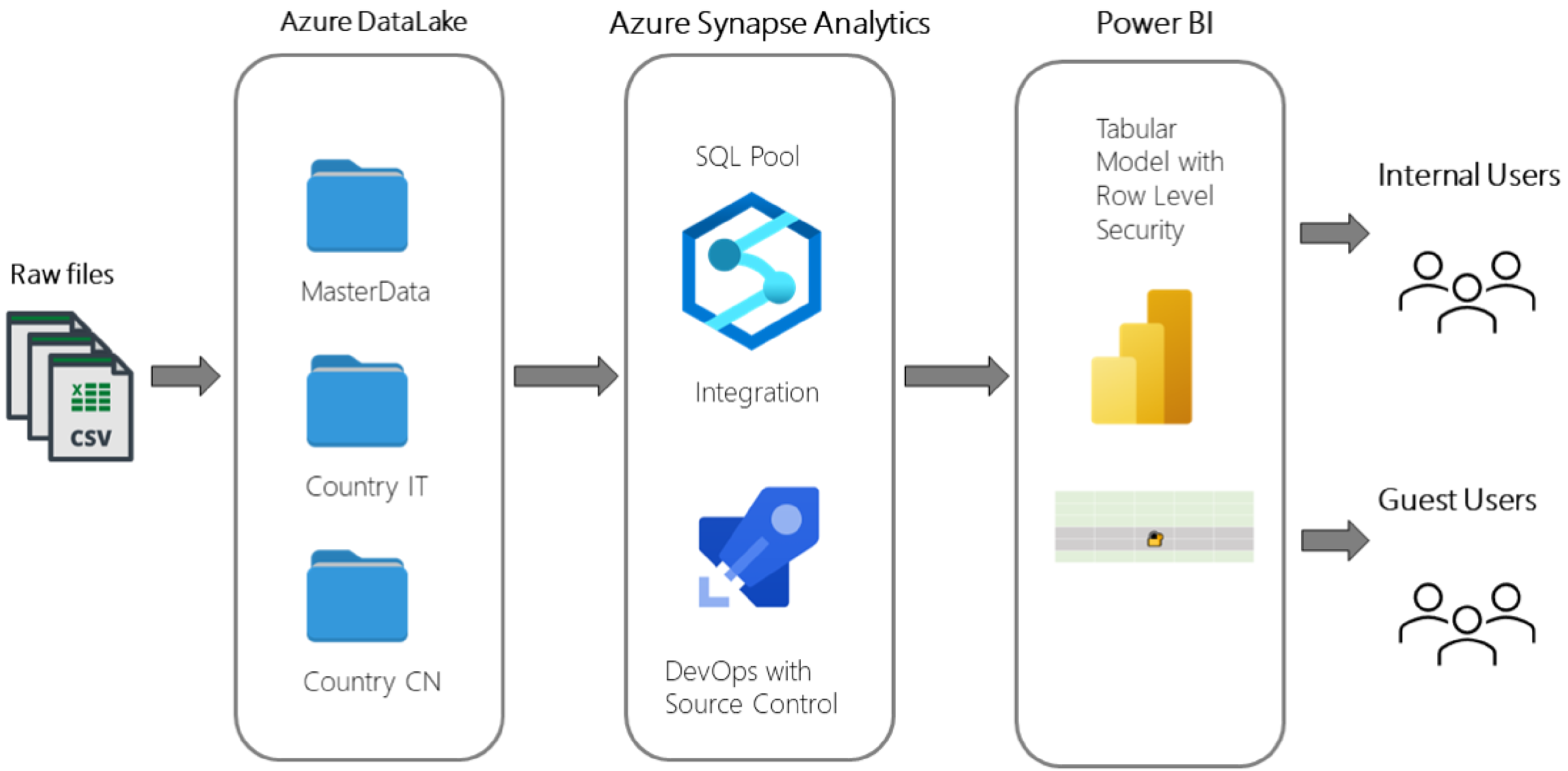


What I like (maybe the most) is the idea

The great unifier

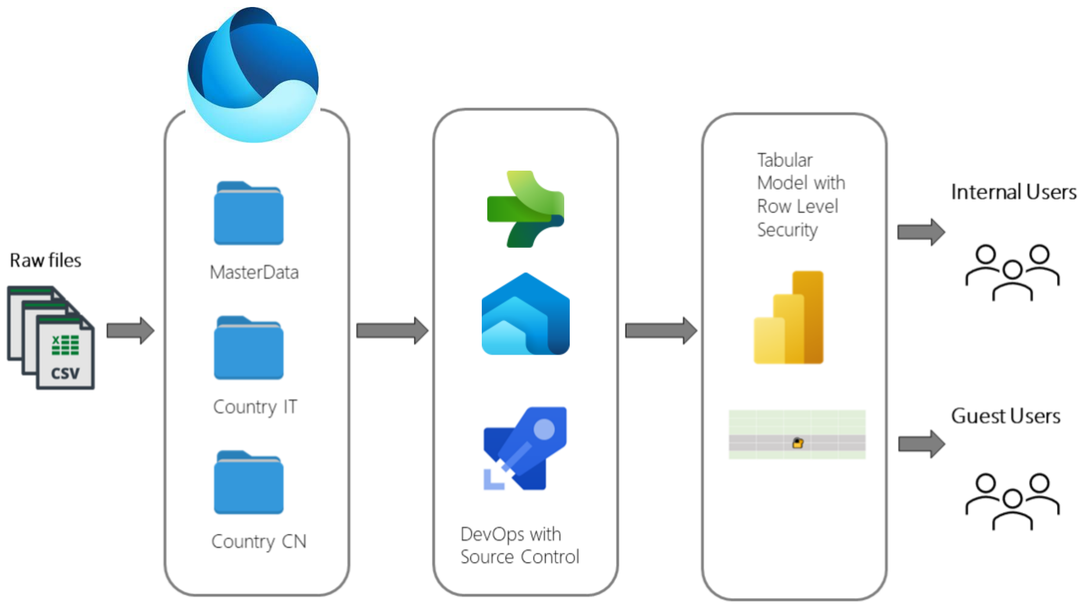


As it is


Raw files
CSV
Azure DataLake
MasterData
Country IT
Country CN
Azure Synapse Analytics
SQL Pool
Integration
DevOps with
Source Control
Power BI
Tabular
Model with
Row Level
Security
Internal Users
Guest Users

To be

- From **ADLS** to **OneLake folders**
- Users can use **Explorer Add-in**
- From **Synapse Pipelines** to **Data Pipeline**
- From **Dedicated SQL Pool** to **Warehouse**
- From **Power BI** to **Power BI**
- Therefore… all-in one… Workspace

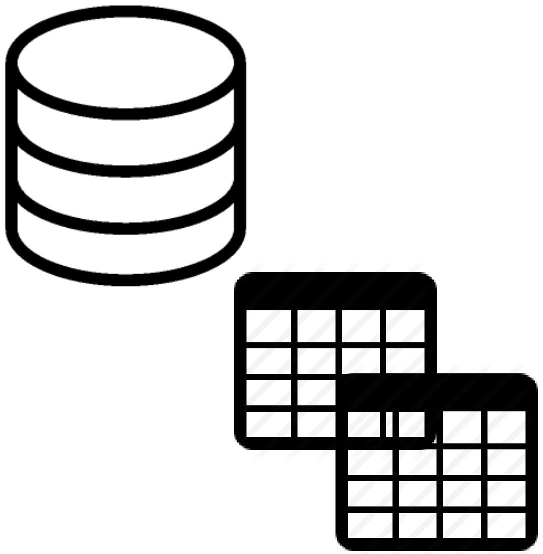


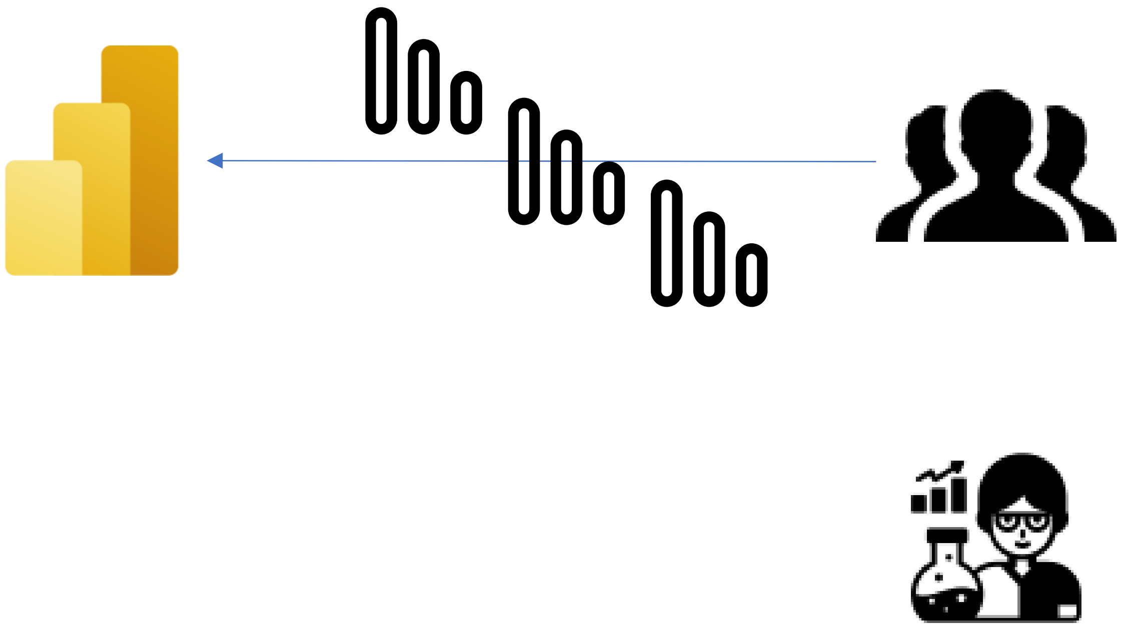
OneLake idea is great

- If I'm the CDO, I can create a **single repository**
  - Structured
  - Unstrctured
- Tell **all my platforms** to read and write there
  - Fabric workload
  - ADF pipelines
  - Databricks
  - Snowflake
- **Standard file format**

OneLake support for Import Mode Semantic Models

The usecase

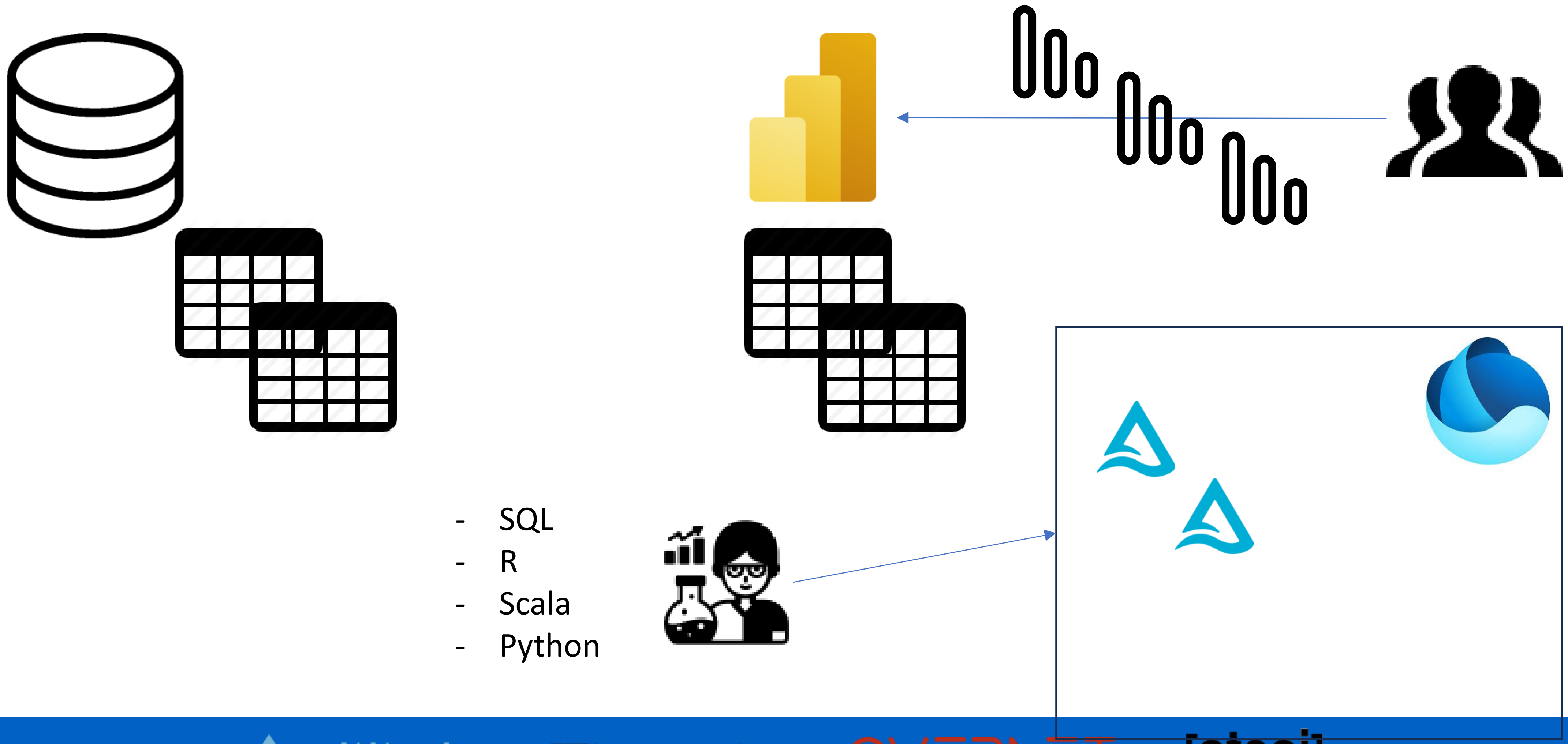




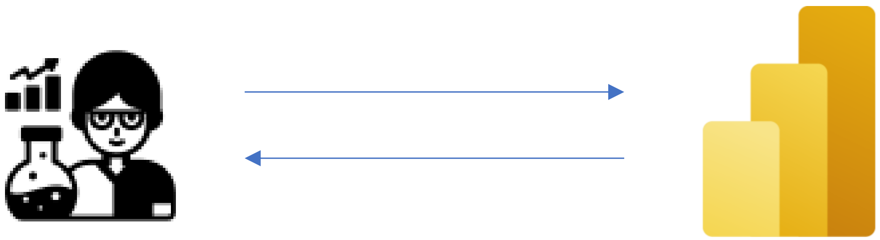
Business Layer / Semantic Layer
+ Cleansing
+ Naming Convention
+ Relationships
+ Calculated Columns
+ Calculated Tables

- Ask for CSV extract
- Use executeQueries API
- Go back to the source

A possible solution



Sempy to leverage the “Golden Model”



Business Layer / Semantic Layer
+ Cleansing
+ Naming Convention
+ Relationships
+ Calculated Columns
+ Calculated Tables
+ Measures

```
%pip install semantic-link

import sempy.fabric as fabric

dataset = "Retail Analysis Sample PBIX"

fabric.evaluate_measure
(dataset, measure="Average Selling Area Size")
```

Use cases for Sempy

- Data Analysts / Data Scientists searching for **data already prepared**
- **Golden Model enrichment**
- Improved **data governance**
- **Strict data pipeline**
- Programmatic interface for **administrative tasks**
    - Models and partitions **refresh**
    - Dataset migration to **Direct Lake**
    - **Monitoring**
    - **Tenant management**
    - **API calls** in Python (alternative to PowerShell)
    - …

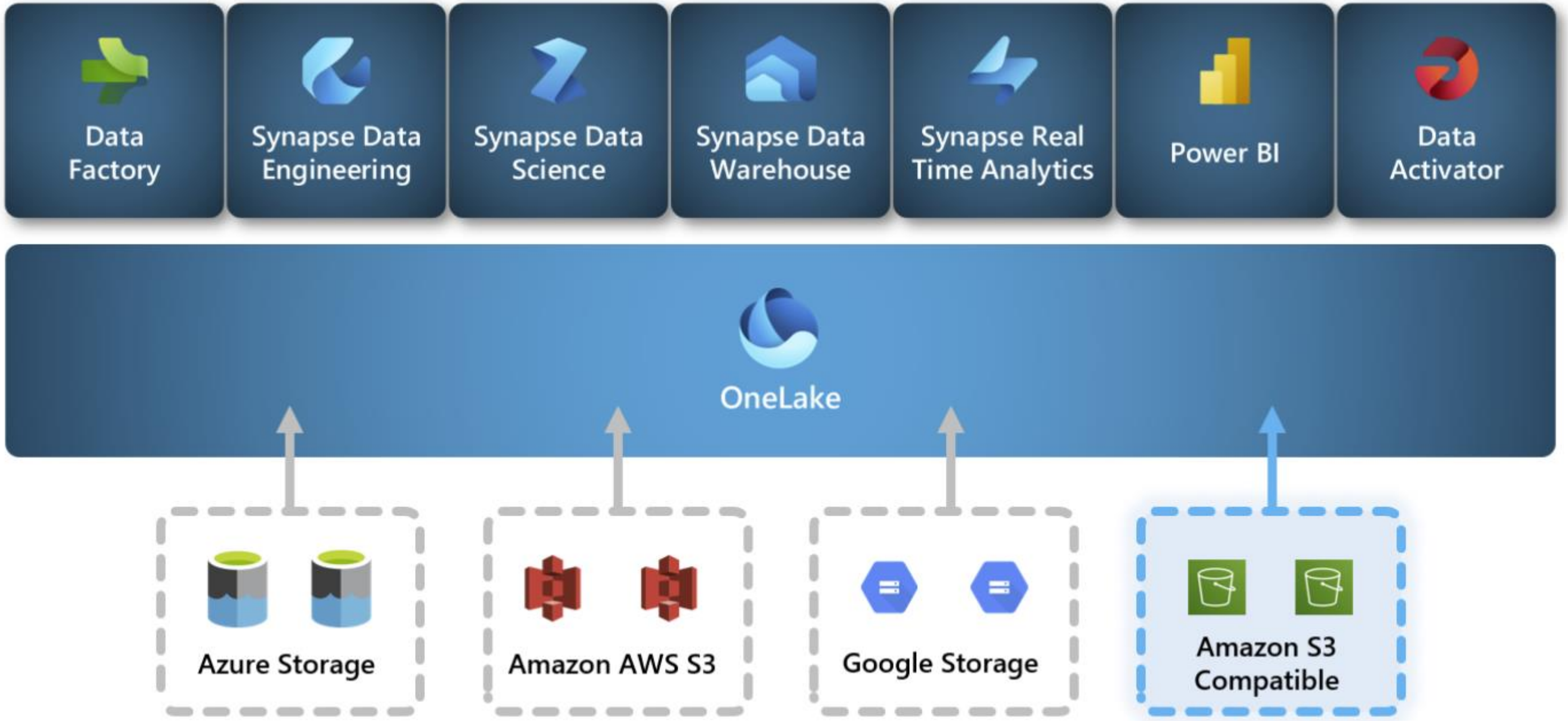
OneLake loves other platforms

It’s not a closed environment

- OneLake support for **Iceberg**
- OneLake **Shortcuts over Snowflake**
- Snowflake can **store data in OneLake**
- Snowflake can **read data from OneLake**

- “Mirrored” (federated) **Azure Databricks Unity Catalog**
- Federated Access **OneLake from Databricks**

One storage to rule them all


Data Factory
Synapse Data Engineering
Synapse Data Science
Synapse Data Warehouse
Synapse Real Time Analytics
Power BI
Data Activator
OneLake
Azure Storage
Amazon AWS S3
Google Storage
Amazon S3 Compatible

T-SQL is alive and well

Clone

- **Metadata only** copy (zero copy)
- **Time travel** back up to 30 days
- Good for **development and testing**
- Good for **backup and restore**
- **Archiving** system

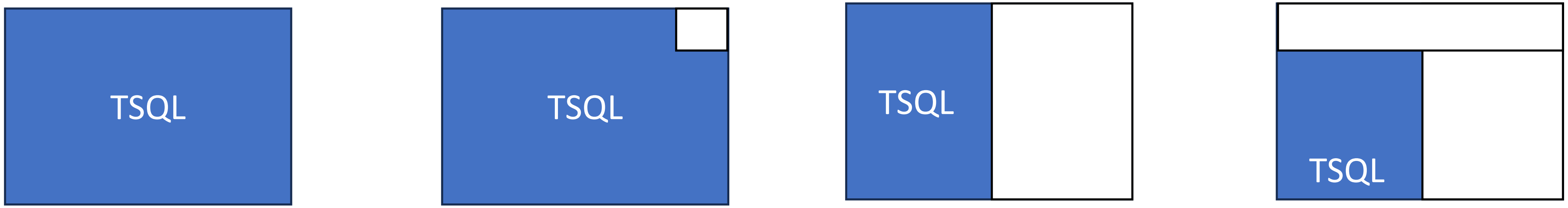
```
CREATE TABLE
	dbo.nyctaxi_clone
AS CLONE OF
	dbo.nyctaxi;
OPTION
	(FOR TIMESTAMP AS OF
	'yyyy-mm-ddTHH:MM:SS.SSS');
```

Time Travel (preview)

```sql
SELECT *
FROM dbo.Top10CustomersView
OPTION (
	FOR TIMESTAMP AS OF '2024-04-24T20:59:06.097'
);
```

Plan migration carefully

Migration is not straight forward



Not Supported

- SET XACT_ABORT ON
- **MERGE**
- **PRIMARY KEY**
- UNIQUE
- **IDENTITY**
- SET IDENTITY_INSERT
- Scalar Function
- ~~**TRUNCATE TABLE**~~

Latin1_General_100_BIN2_UTF8

```
SELECT *
FROM dbo.MyTable;
```

<>

```
SELECT *
FROM dbo.mytable;
```

```
SELECT COUNT(*)
FROM dbo.MyTable
WHERE Code = ‘abc’;
```

Id	Code
1	abc
2	abc
3	abc

Id	Code
1	Abc
2	abc
3	ABC

NVARCHAR

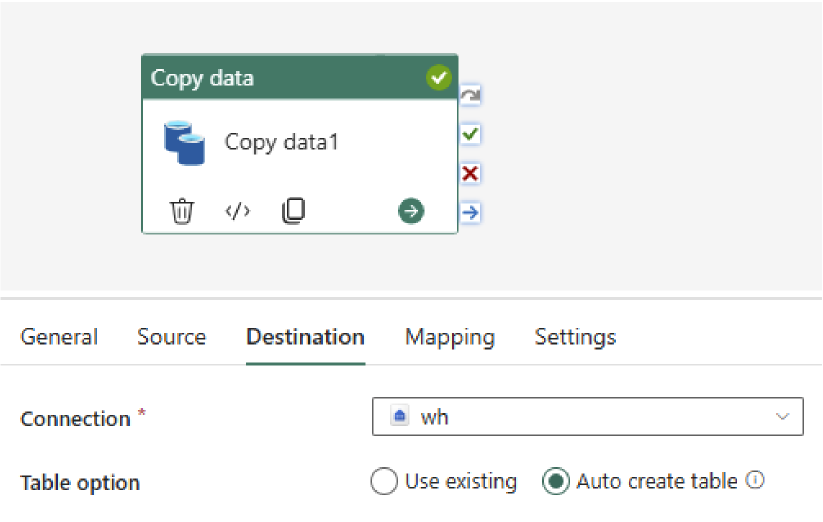
As of today, not supported.

Not a big deal since **Parquet manage automatically encoding** and collation is UTF-8.

View-table pipeline looses types

```
CREATE VIEW dbo.MyView AS
  SELECT
      AnIntField = CONVERT(int, t.F1)
      ,AString = CONVERT(varchar(20), t.F2)
      ,AndADateField = CONVERT(date, t.F3)
   FROM dbo.MyTable;
GO
```

```
CREATE TABLE dbo.MyNormalizedTable
(
		AnIntField int  NULL,
		Astring varchar (8000)  NULL,
		AndADateField date  NULL
)
GO
```



Workaround

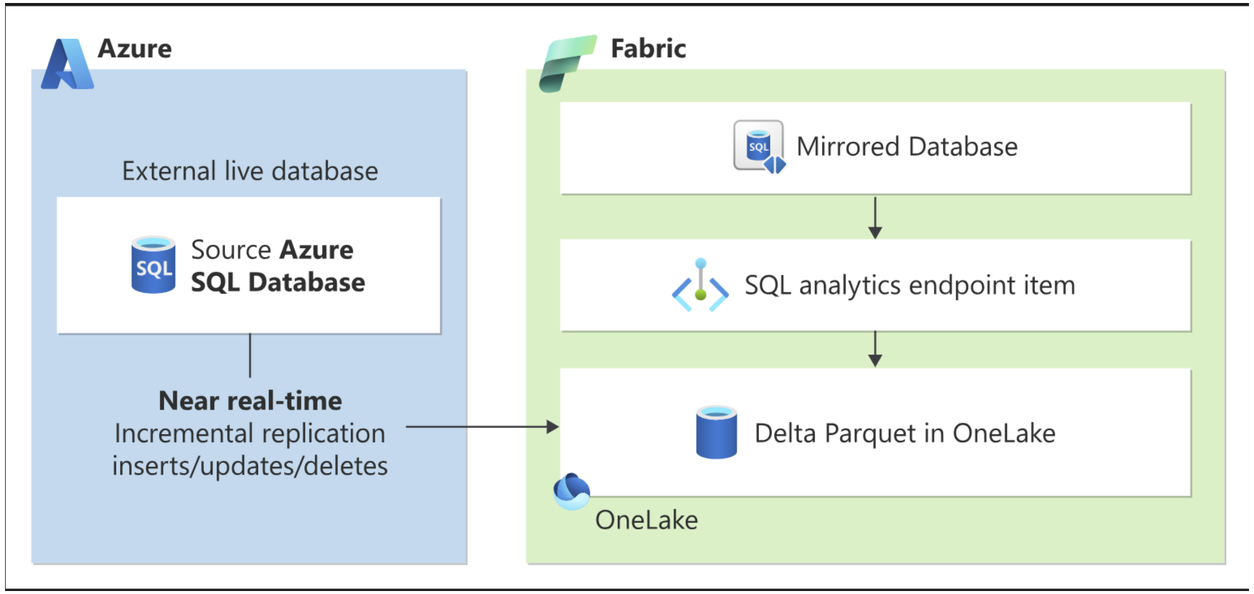
```
SELECT *
INTO dbo.MyNormalizedTable
FROM dbo.MyView;
```

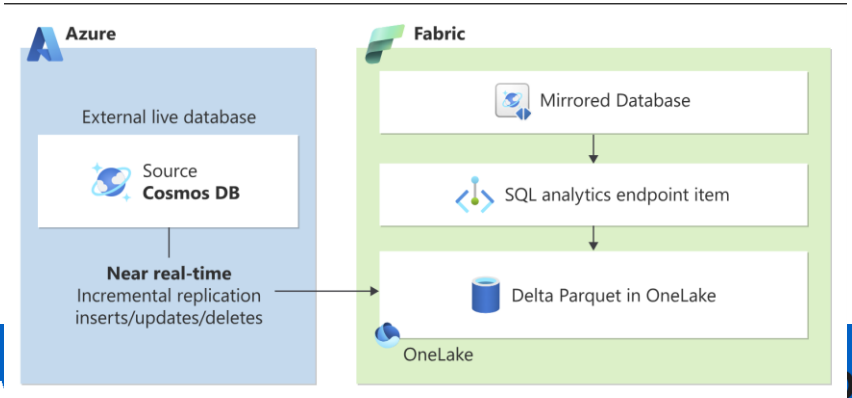
Or

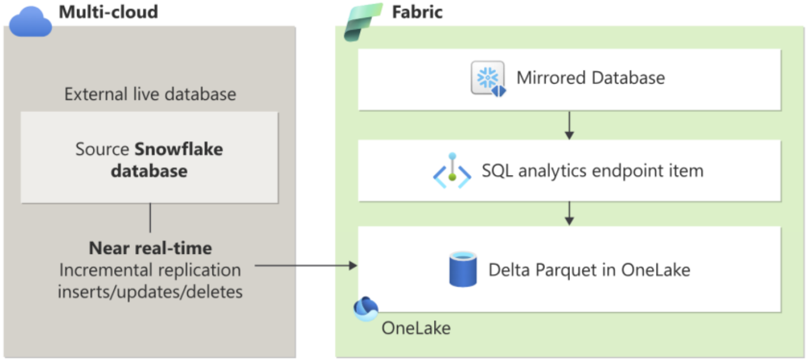
```
CREATE TABLE dbo.MyNormalizedTable AS
SELECT *
FROM dbo.MyView;
```

Pump data before playing

Mirroring to Fabric







Pros & Cons (On the SQL side)

- Managed ELT
- Near real-time replication
- Multi platform
- No extra cost 1TB per CU

- Preview feature
- No On-Prem SQL Server
- Public access only supported
- Cross tenant not supported
- … it's not magic

Cons… continue

- A table cannot be mirrored if it does not have a **primary key rowstore clustered index**
- Following DDL operations aren't allowed on source tables
  - Switch/Split/Merge partition
  - Alter primary key
  - ~~**Drop table**~~
  - **Truncate table**
  - ~~Rename table~~
  - ~~Alter column~~
- When there is **DDL change, a complete data snapshot** is restarted for the changed table, and data is reseeded

Snowflake has less limitations

Another definitive Great Unifier…

Simply unmatched, truly limitless: Announcing Azure Synapse Analytics

By Rohan Kumar, Corporate Vice President, Azure Data

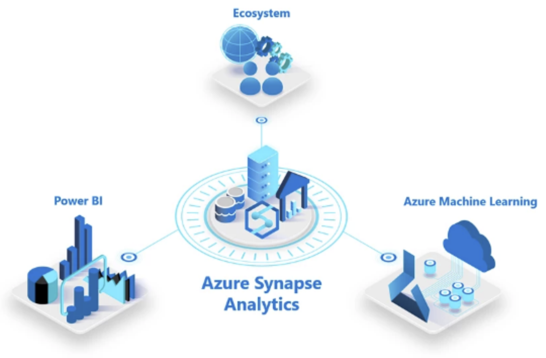
Posted on November 4, 2019
3 min read

Big Data

Today, businesses are forced to maintain two types of analytical systems, data warehouses and data lakes. Data warehouses provide critical insights on business health. Data lakes can uncover important signals on customers, products, employees, and processes. Both are critical, yet operate independently of one another, which can lead to uninformed decisions. At the same time, businesses need to unlock insights from *all* their data to stay competitive and fuel innovation with purpose. Can a single cloud analytics service bridge this gap and enable the agility that businesses demand?

Azure Synapse Analytics

Today, we are announcing Azure Synapse Analytics, a limitless analytics service, that brings together enterprise data warehousing and Big Data analytics. It gives you the freedom to query data on your terms, using either serverless on-demand or provisioned resources, at scale. Azure Synapse brings these two worlds together with a unified experience to ingest, prepare, manage, and serve data for immediate business intelligence and machine learning needs.



Explore

Let us know what you think of Azure and what you would like to see in the future.

Provide feedback

Build your cloud computing and Azure skills with free courses by Microsoft Learn.

Explore Azure learning

It seems a building site yet

- **Tons of features** every months
- Many features stay in **preview for a very long time** (even core)
  - GIT integration
  - Invoke Pipeline
  - Folders
  - Data Activator
  - Mirroring (only Snowflake GA)
  - PBIP and TMDL, New Card, Field Parameters…
- Flagship features only announced i.e. **OneSecurity**
- ~~**Security features premium+** (+F64)~~

Database project support

DACPACs

Lakehouse or not Lakehouse

What experts say

Jovan Popovic

Principal Program Manager at Microsoft, working on Microsoft Fabric Warehouse. Worked on Azure Synapse, Azure SQL Azure SQL Managed Instance, and SQL Server.

Published Aug 10, 2023

[Choosing between Lakehouse and Warehouse in Microsoft Fabric (linkedin.com)]

- **Language preference for data management** - The choice between Lakehouse and Warehouse depends on the preferred programming languages of your team. This includes both **syntax** and **functionalities** that are available in the language that you choose. If your team leans towards PySpark or Scala for data management, Lakehouse is the natural choice. On the other hand, Warehouse caters to those who favor T-SQL. This distinction may be less relevant for visual, low-code, or citizen developers. However, it's important to consider the features associated with different languages. For instance, if you need ML/regular expressions, Lakehouse is the choice for you because these features are available in Python/Scala. On the other hand, if you need multi-table transactions, some enterprise data management features, or fine-grained permissions on every object in the database with GRANT/DENY, RLS, data masking you should choose T-SQL language in the Warehouse.
- **Data format requirements** - If your data exists primarily in Delta format with relational structure, Warehouse seamlessly handles your needs. However, if you work with diverse formats like CSV, Parquet, or JSON, or you are using non-structured data, Lakehouse proves to be the more versatile solution.
- **Migration scenarios** - If your existing data solutions are implemented on SQL Server, Azure SQL, Synapse warehouse or other RDBMS systems, or involve a significant T-SQL code base that you wish to retain while transitioning to Fabric, the Warehouse is the preferable option. If you're migrating from a Spark/Databricks and have already implemented significant data processing logic in PySpark, Scala or SparkSQL notebooks, Lakehouse provides an easier path to migration within the Fabric ecosystem.

Even in official docs

Microsoft Fabric decision guide: choose a data store

Article • 06/05/2024 • 10 contributors

Feedback

In this article

Use this reference guide and the example scenarios to help you choose a data store for your Microsoft Fabric workloads.

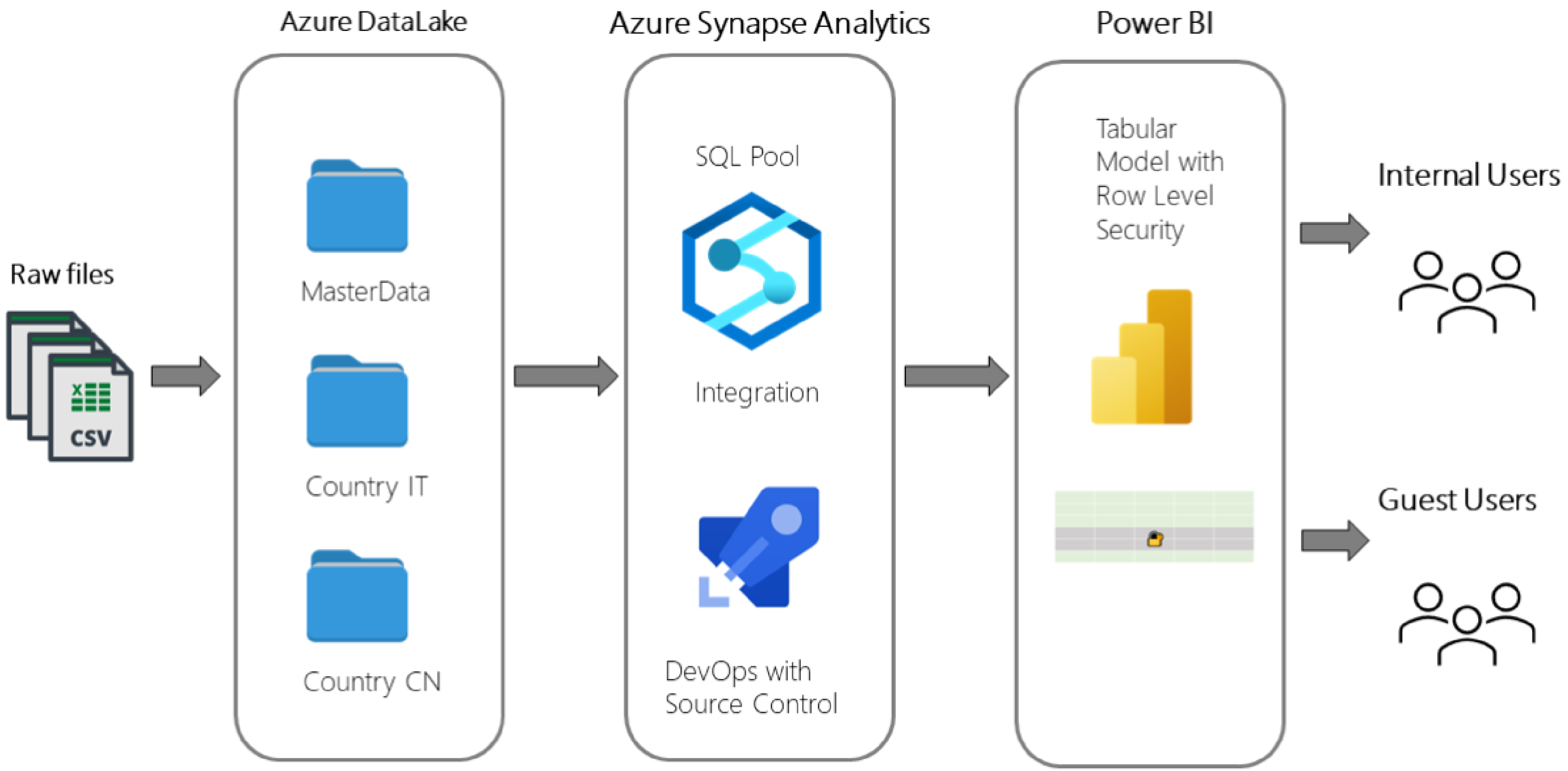
Data store properties

Expand table

	Warehouse	Lakehouse	Power BI Datamart	Eventhouse
Data volume	Unlimited	Unlimited	Up to 100 GB	Unlimited
Type of data	Structured	Unstructured, semi-structured, structured	Structured	Unstructured, semi-structured, structured
Primary developer persona	Data warehouse developer, SQL engineer	Data engineer, data scientist	Citizen developer	Citizen Data scientist, Data engineer, Data scientist, SQL engineer
Primary developer skill set	SQL	Spark(Scala, PySpark, Spark SQL, R)	No code, SQL	No code, KQL, SQL
Data organized by	Databases, schemas, and tables	Folders and files, databases, and tables	Database, tables, queries	Databases, schemas, and tables
Read operations	T-SQL, Spark (supports reading from tables using shortcuts, doesn't yet support accessing views, stored procedures, functions etc.)	Spark, T-SQL	Spark, T-SQL, Power BI	KQL, T-SQL, Spark, Power BI
Write operations	T-SQL	Spark(Scala, PySpark, Spark SQL, R)	Dataflows, T-SQL	KQL, Spark, connector ecosystem
Multi-table transactions	Yes	No	No	Yes, for multi-table ingestion. See update policy.
Primary development interface	SQL scripts	Spark notebooks,Spark job definitions	Power BI	KQL Queryset, KQL Database
Security	Object level (table, view, function, stored procedure, etc.), column level, row level, DDL/DML, dynamic data masking	Row level, table level (when using T-SQL), none for Spark	Built-in RLS editor	Row-level Security

[Fabric decision guide - choose a data store – Microsoft Fabric | Microsoft Learn]()

Is it really a choice?



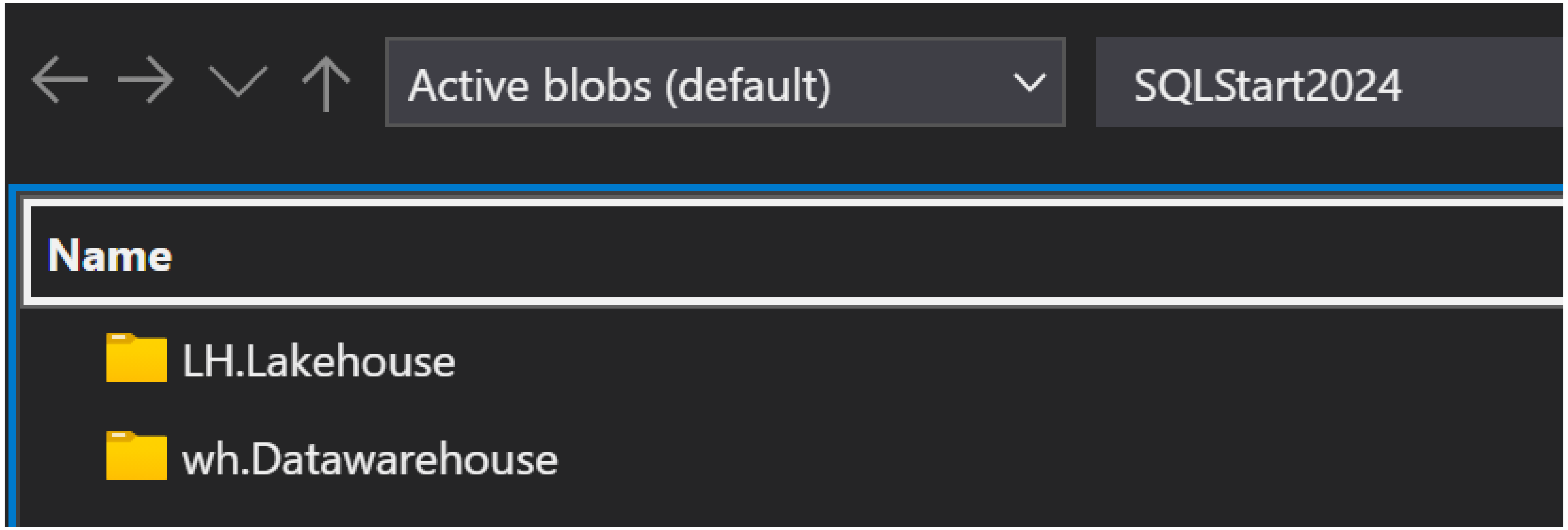
You’ll likely need both…

- No Mirroring
- No cross-table transaction
- No T-SQL C~~R~~UD

- No Shortcut
- No OneLake interoperability
- No security features
    - ADLS firewalled access

Dealing with Delta in a different way

Under the hood



Lakehouse style

```python
df = spark.sql("SELECT 1 AS col")

for i in range(1, 11):
    df.write.format("delta").mode("append").
saveAsTable("TestVacumLH")
```

_delta_log	6/9/2024 5:07:14 PM	Folder	11 item
part-00000-29e9efd5-5c0f-4bff-884b-06ee4a4bb9ee-c000.snappy.parquet	6/9/2024 5:07:26 PM	parquet	781 B
part-00000-3e33182e-74ad-4d19-91b1-8b2e6a5b74e2-c000.snappy.parquet	6/9/2024 5:07:28 PM	parquet	781 B
part-00000-47b3d85f-2df0-409b-8594-0ae8fb4ebdef-c000.snappy.parquet	6/9/2024 5:07:30 PM	parquet	781 B
part-00000-4d680093-de40-4e73-9218-bb7b55f690ad-c000.snappy.parquet	6/9/2024 5:07:15 PM	parquet	781 B
part-00000-5ec6b95a-7c36-4a86-a13a-33780505117b-c000.snappy.parquet	6/9/2024 5:07:32 PM	parquet	781 B
part-00000-5f6d19a6-4a95-4581-8184-b53d5e69558a-c000.snappy.parquet	6/9/2024 5:07:22 PM	parquet	781 B
part-00000-7438f8bc-418e-436a-bdc2-b4fd22cdffc3-c000.snappy.parquet	6/9/2024 5:07:18 PM	parquet	781 B
part-00000-8a391aab-84cc-4516-9ad7-ebcfcdf7ee16-c000.snappy.parquet	6/9/2024 5:07:24 PM	parquet	781 B
part-00000-b0f803f5-7e1f-4302-94a3-271cbbc7b3b3-c000.snappy.parquet	6/9/2024 5:07:34 PM	parquet	781 B
part-00000-e43b0ed5-9c09-4b99-b230-588fc0b30b68-c000.snappy.parquet	6/9/2024 5:07:20 PM	parquet	781 B

testvacumlh (file view) > **_delta_log**

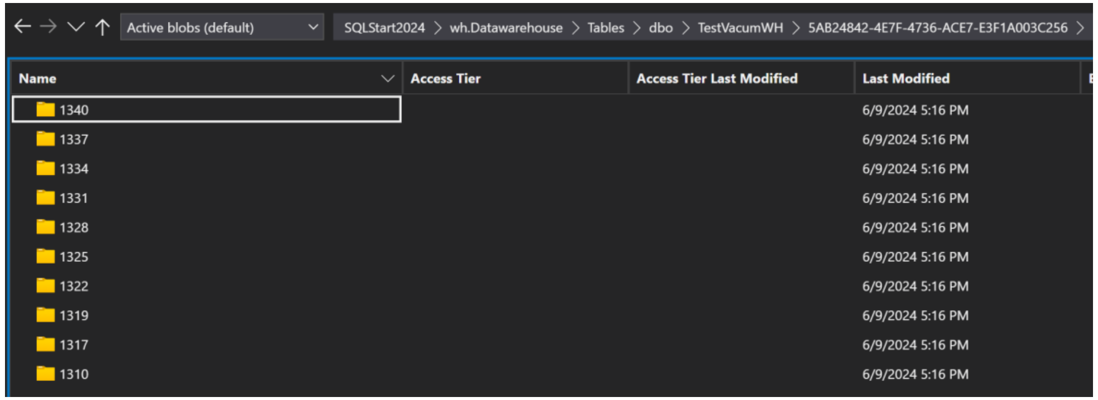
Name	Date modified	Type	Size
00000000000000000000.json	6/9/2024 5:07:15 PM	json	1 KB
00000000000000000001.json	6/9/2024 5:07:18 PM	json	713 B
00000000000000000002.json	6/9/2024 5:07:20 PM	json	713 B
00000000000000000003.json	6/9/2024 5:07:22 PM	json	713 B
00000000000000000004.json	6/9/2024 5:07:24 PM	json	713 B
00000000000000000005.json	6/9/2024 5:07:26 PM	json	713 B
00000000000000000006.json	6/9/2024 5:07:28 PM	json	713 B
00000000000000000007.json	6/9/2024 5:07:30 PM	json	713 B
00000000000000000008.json	6/9/2024 5:07:32 PM	json	713 B
00000000000000000009.json	6/9/2024 5:07:34 PM	json	713 B
_temporary	6/9/2024 5:07:15 PM	Folder	0 item

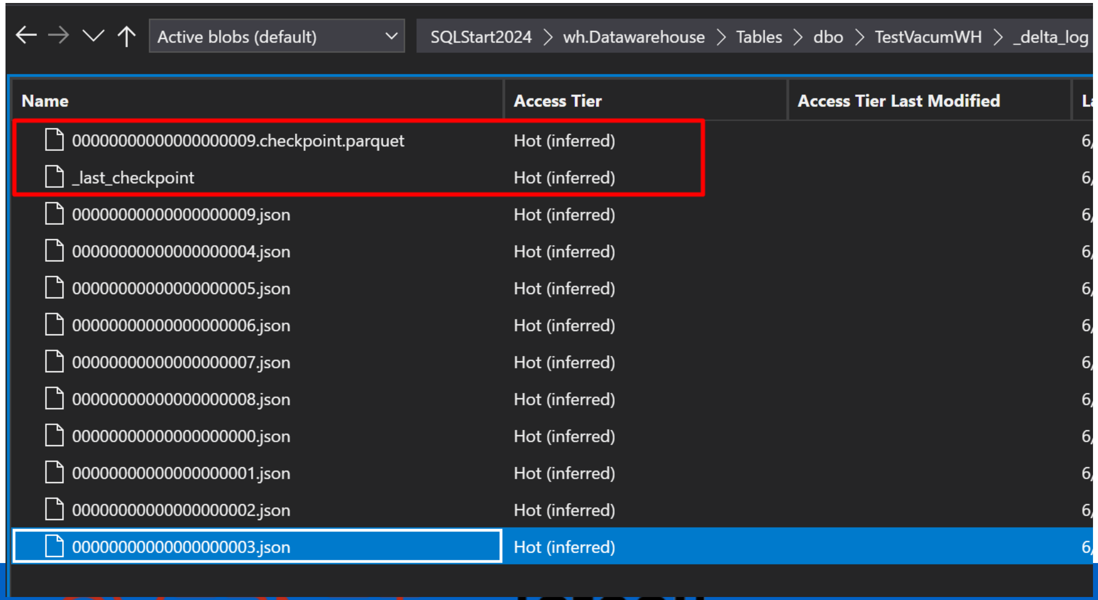
Warehouse style

```sql
SELECT 1 AS col
INTO dbo.TestVacumWH;
GO

DECLARE @i INT = 1;

WHILE @i < 10
BEGIN
    INSERT INTO
      dbo.TestVacumWH
    VALUES(1);
    SET @i = @i + 1;
END;
```





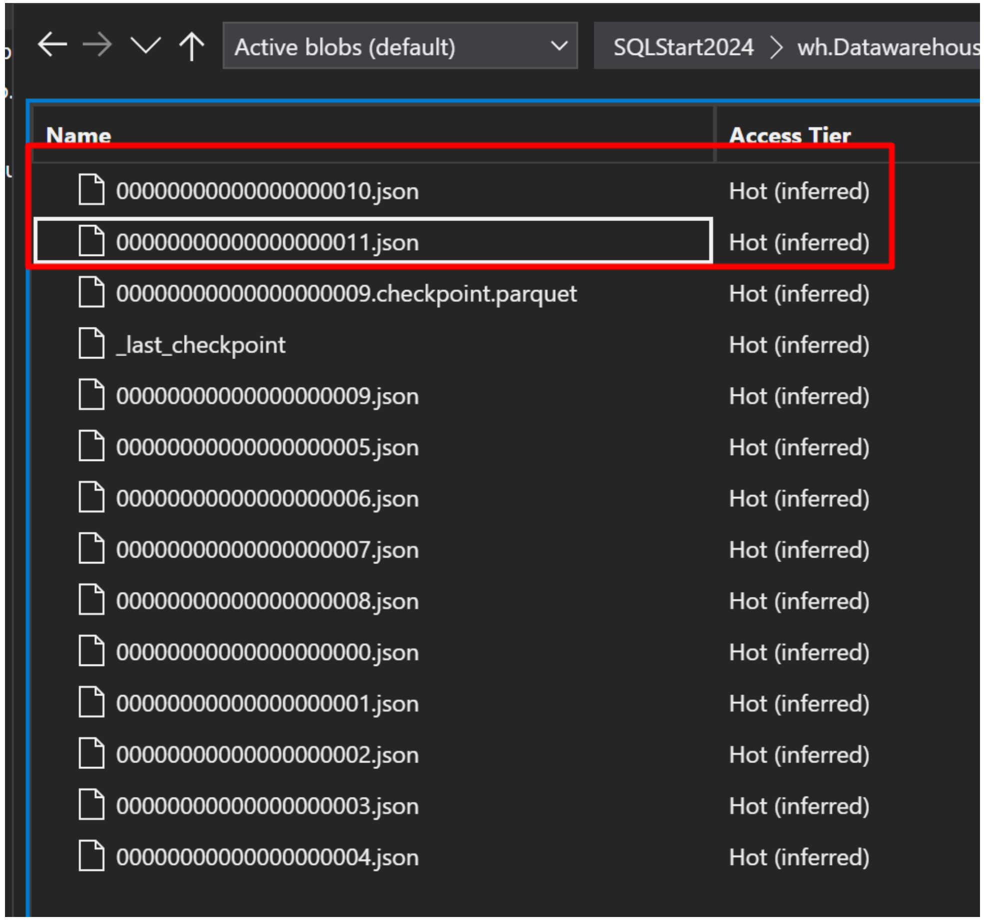
Last checkpoint

```json
{
    "version": 11,
    "size": 15,
    "sizeInBytes": 13909,
    "numOfAddedFiles": 10
}
```

Checkpoint.parquet


```
{"txn":{"appId":"424c3456-c3dc-e5f8-530f-135019cd5974","version":1530,"lastUpdated":null},"protocol":null,"metaData":null,"add":null,"remove":null}
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1530/3893A81A-6437-43CB-B1FE-CB1D4BE98D84.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1524/60547DA2-DBFC-44CD-91E6-3E13517ACC05.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1521/9A4F0D97-5F62-469A-A454-6F8DD2571F4A.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1518/EA152942-9558-433D-BD7D-BE66C8907563.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1515/6A4A7BC2-CCA8-48A5-827E-256D8082190B.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1512/C34BE407-2192-4C72-920C-8FA21C85EC09.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1509/CF0614B4-DD85-43E8-B53D-AD95F991A481.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1506/7E96F95B-AF37-441E-92F7-DE95561EFAAC.parquet",
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1504/32D7FE8B-3CDD-49FF-A0AE-FC2280D11C2E.parquet",
{"txn":null,"protocol":{"minReaderVersion":1,"minWriterVersion":1,"readerFeatures":null,"writerFeatures":null},"metaData":null,"add":null,"remove":nu
{"txn":null,"protocol":null,"metaData":{"id":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3","name":"TestVacumWH","description":null,"format":{"provider":"par
{"txn":null,"protocol":null,"metaData":null,"add":{"path":"E7285DBA-A770-473F-8ED7-D7D849AF3BB3/0/1497/421E29B3-8296-4E5D-91CF-9302B295CD71.parquet",
```

Later on...



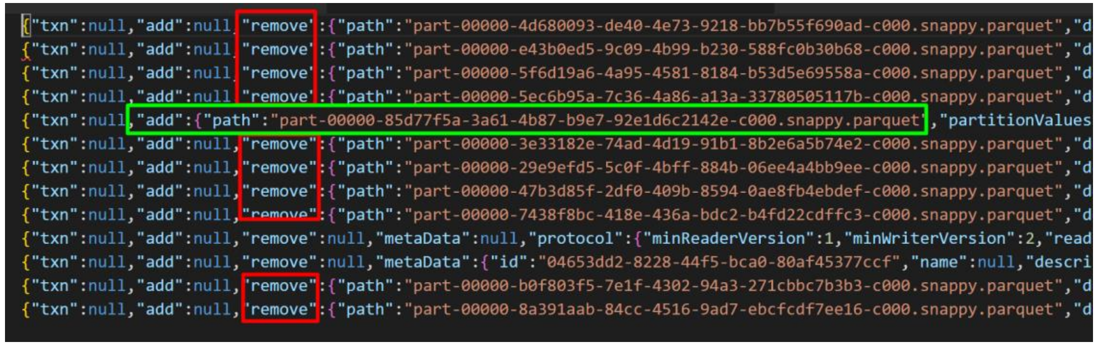
Active blobs (default)
SQLStart2024
wh.Datawarehous
Name
Access Tier
00000000000000000010.json
Hot (inferred)
00000000000000000011.json
Hot (inferred)
00000000000000000009.checkpoint.parquet
Hot (inferred)
_last_checkpoint
Hot (inferred)
00000000000000000009.json
Hot (inferred)
00000000000000000005.json
Hot (inferred)
00000000000000000006.json
Hot (inferred)
00000000000000000007.json
Hot (inferred)
00000000000000000008.json
Hot (inferred)
00000000000000000000.json
Hot (inferred)
00000000000000000001.json
Hot (inferred)
00000000000000000002.json
Hot (inferred)
00000000000000000003.json
Hot (inferred)
00000000000000000004.json
Hot (inferred)

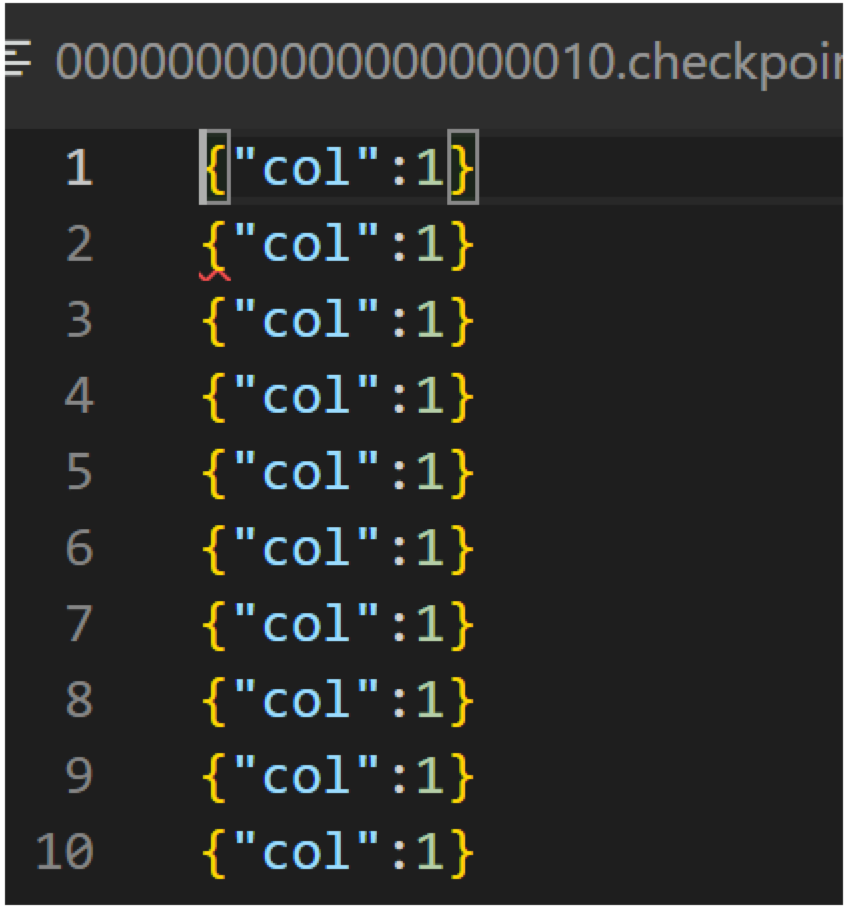
WH - Automatic Data Compaction

Name	Access Tier	Access Tier Last Modified	Last Modified	Blob Type	Content Type	Size
25259834-67A2-4751-8862-4651D0E84220.parc	Hot (inferred)		06/02/2024 09:52	Block Blob	application/octet-stream	40,01 MiB
126AAE70-7D44-4408-8BF6-BD7AE00B2097.par	Hot (inferred)		06/02/2024 09:52	Block Blob	application/octet-stream	46,59 MiB
1E043E48-202E-4A81-872B-D7FD57061F2C.parc	Hot (inferred)		06/02/2024 09:52	Block Blob	application/octet-stream	41,27 MiB
4A40B569-B4B4-4A30-9BD1-9BA62AF3DBB5.pa	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	14,61 MiB
5AEA304A-8AFA-43FE-B970-BDF6DAAF63CD.pa	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	14,34 MiB
956D0660-AC72-48A0-889C-6E85C412E0B2.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	16,56 MiB
DA7641A3-7D9E-4DF8-8125-8A24B7954808.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	11,39 MiB
028692F8-DC16-4417-B865-41C6B2E88252.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	6,24 MiB
087C739E-0B07-4E8F-95F2-D0A5B5983415.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	6,05 MiB
36786AFA-C453-43E2-9E23-1C54794D6C87.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	6,14 MiB
3988471F-7687-44A8-B089-2850AC277D1B.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	7,07 MiB
695E3C89-0F4A-4284-81FB-85315EA64B4C.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	4,74 MiB
6D7E0EC9-BA19-4C2F-9B8E-01A94E715C2D.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	12,14 MiB
7345937E-0954-445B-81FB-45219F02058B.parq	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	5,85 MiB
736CD976-865C-4840-A31B-12FC936B2679.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	5,74 MiB
747682B2-8448-409C-9A09-8EA4AF8A5F02.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	8,48 MiB
778CF6B1-36E6-4E23-B1F3-7A4D2EE9348B.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	5,84 MiB
7A018363-724B-49D3-AC46-1410E6C6BEE7.par	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	8,55 MiB
A7598F85-9714-4600-9149-B1E4A57A38DE.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	5,37 MiB
0A5B1A55-280B-48D6-A0AC-B3682F02CECB.pa	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	50,04 MiB
28300206-4B5A-4F28-8645-5F909D3AB809.parc	Hot (inferred)		06/02/2024 09:38	Block Blob	application/octet-stream	49,59 MiB

Showing 1 to 67 of 67 cached items

%%sql
OPTIMIZE <table|fileOrFolderPath>;



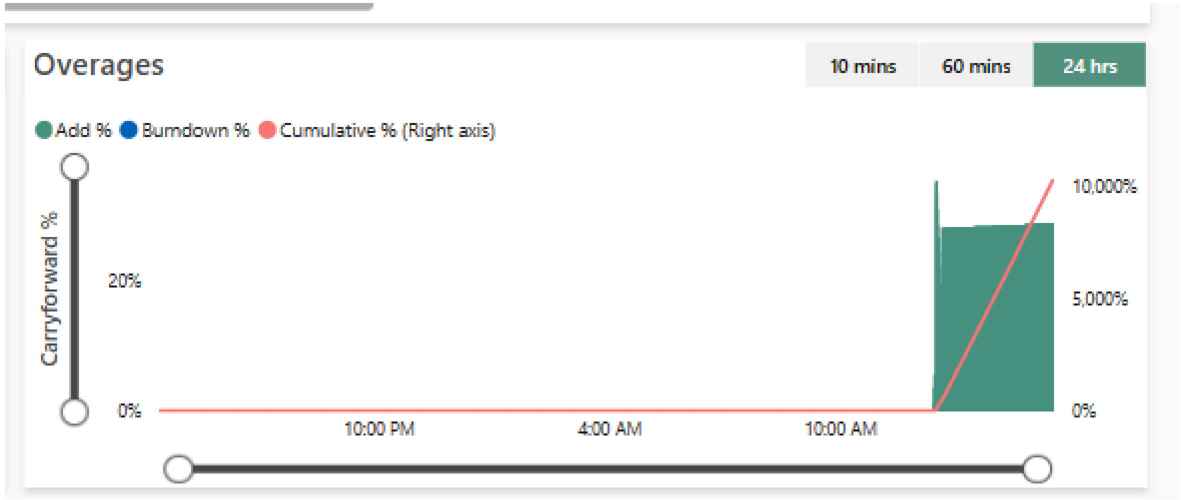


With great power comes great responsibility

Burns as much as it can

Fabric SKU	Equivalent Premium SKU	Baseline Capacity Units (CU)	Burstable Scale Factor
F2		2	1x - 32x
F4		4	1x - 16x
F8		8	1x - 12x
F16		16	1x - 12x
F32		32	1x - 12x
F64	P1	64	1x - 12x
F128	P2	128	1x - 12x
F256	P3	256	1x - 12x
F512	P4	512	1x - 12x
F1024	P5	1024	1x - 12x
F2048		2048	1x - 12x

And then...


Overages
10 mins
60 mins
24 hrs
Add %
Burndown %
Cumulative % (Right axis)
Carryforward %
20%
0%
10:00 PM
4:00 AM
10:00 AM
10,000%
5,000%
0%

Useful Links

- [Access Fabric data locally with OneLake file explorer]
- [Transitioning from ADLS to OneLake]
- [Datasets - Execute Queries - REST API (Power BI Power BI REST APIs)]
- [Exporting Power BI Reports And Sharing With Granular Access Control In Fabric]
- [Limitations for Fabric mirrored databases]
- [Create shortcuts to on-premises data]
- [Microsoft Idea Limit Bursting]
- [Apache XTable™ (Incubating)]

Questions?

Thank You!