

SQL KONFERENZ 2026

SQL KONFERENZ 2026



Fabric meets Databricks

Tuesday, 3 Mar 2026 11:15 am - 12:15 pm (60 minutes)

Gerhard Brueckl



@gbrueckl

blog.gbrueckl.at

in/gbrueckl

github.com/gbrueckl

Power BI Studio

Fabric Studio

Databricks Power Tools

SQL KONFERENZ 2026

paiqo

www.paiqo.com

cloudflight

www.cloudflight.io

Blend Mexhuani



SQL KONFERENZ 2026

in/blendmexhuani

github.com/blendmexhuani

paiqo

www.paiqo.com

cloudflight

www.cloudflight.io

Agenda

- General Introduction
- Databricks
- Fabric
- Security & Governance
- [Power] BI

SQL
KONFERENZ
2026



Fabric & Databricks

And what both have in common

Common Technologies



- Distributed computing engine for large-scale data processing
- In-memory execution for faster performance at
- Unified API for batch, streaming, SQL, and ML
- Native integration with Delta Lake, Hive, and cloud storage
- **Invented by Databricks**



- Open-source storage layer built on Parquet
- ACID transactions for reliability and consistency
- Time travel enables querying historical data snapshots
- Schema enforcement and evolution built in
- **Self-contained folder structure for data and metadata**
- **Invented by Databricks**

Delta Lake

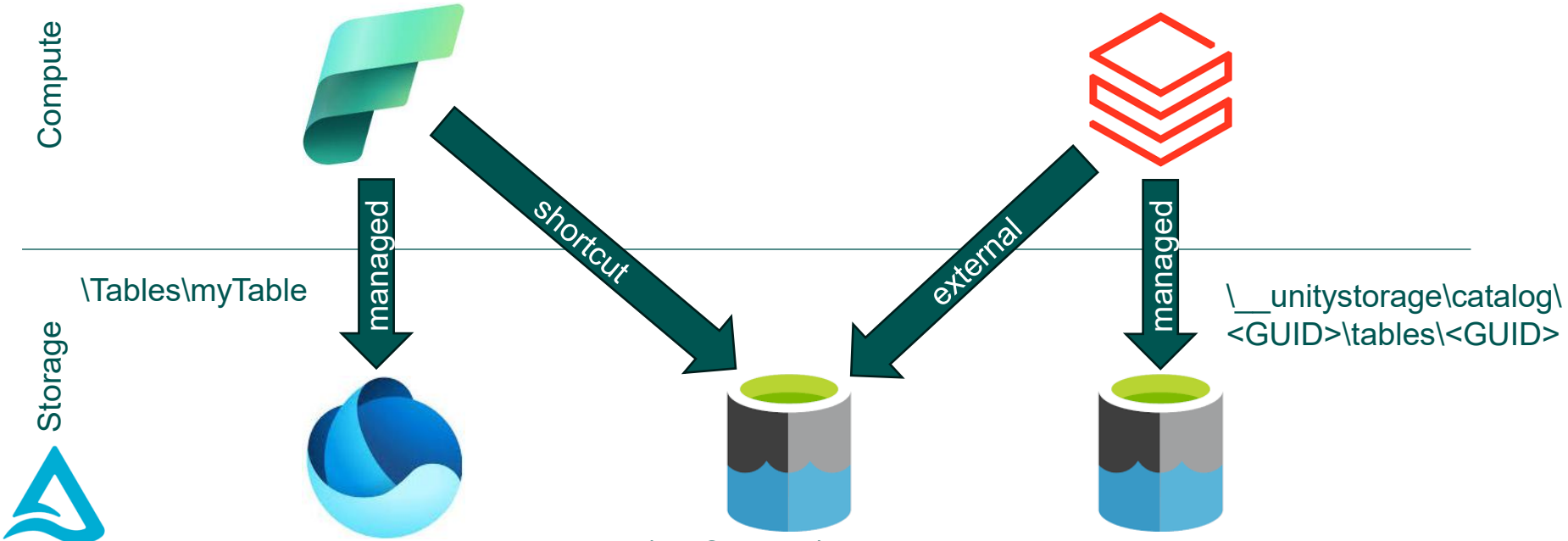




- For Lakehouses
- Every workload that stores data
- Data Exchange between Fabric Engines
- Direct Lake for Power BI

- Default Table Format for everything
- Data engineering / BI / Machine Learning / …

Separation of Storage and Compute



DELTA LAKE

The data catalog
Bringing together Storage and Compute

- Connects Compute to Storage
- Central point to …
  - Browse/search for
  - Access
  - Govern

  … data
- **Delta Lake:** reference to storage location



Unity Catalog



OneLake Catalog

SQL
KONFERENZ
2026

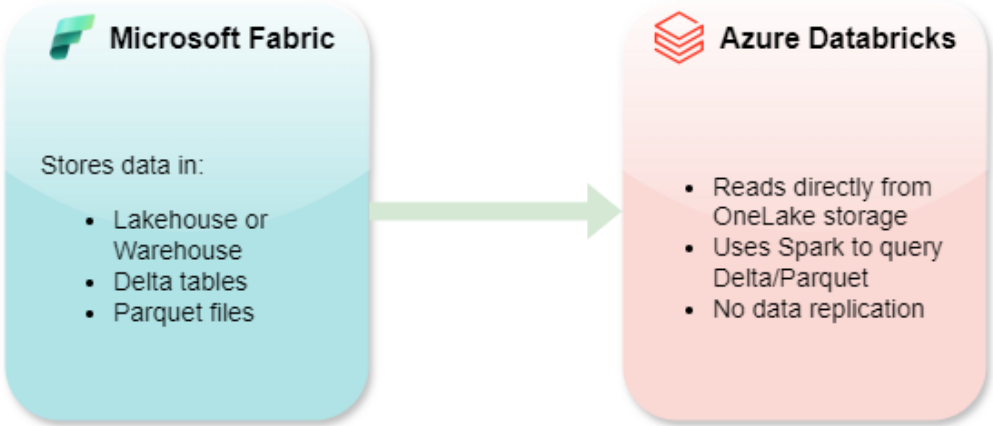


Databricks

Integrating Onelake into Databricks

Objective

- Enable Databricks to read data stored in Microsoft Fabric/OneLake without copying or moving the data



SQL KONFERENZ 2026

Integrating Onelake into Databricks



1. Cluster setting: Enable Credential Passthrough (DBR <15.0)
2. Service-Principal authentication using access-token (as DeltaTable)
3. Mount the Lakehouse path with a Service-Principal authentication
4. Onelake as foreign-catalog in Unity Catalog (recommended) → Public Preview

Integrating Onelake into Databricks

Service-Principal authentication using access-token (as DeltaTable)

```
app = ConfidentialClientApplication(
    "<client-id>",
    authority="https://login.microsoftonline.com/<tenant-id>",
    client_credential="<client-secret>"
)
result = app.acquire_token_for_client(scopes=["https://onelake.fabric.microsoft.com/.default"])
if "access_token" in result:
    access_token = result['access_token']
    print("Access token acquired.")
else:
    print("Access token could not be acquired. Please check your credentials.")
    dbutils.notebook.exit(0)

workspace_id = "<insert-fabric-workspace-id>"
lakehouse_id = "<insert-fabric-lakehouse-id>"
table_to_read = "<insert-lakehouse-table-name>"
onelake_uri = f"abfss://{workspace_id}@onelake.dfs.fabric.microsoft.com/{lakehouse_id}.lakehouse/Tables/{table_to_read}"

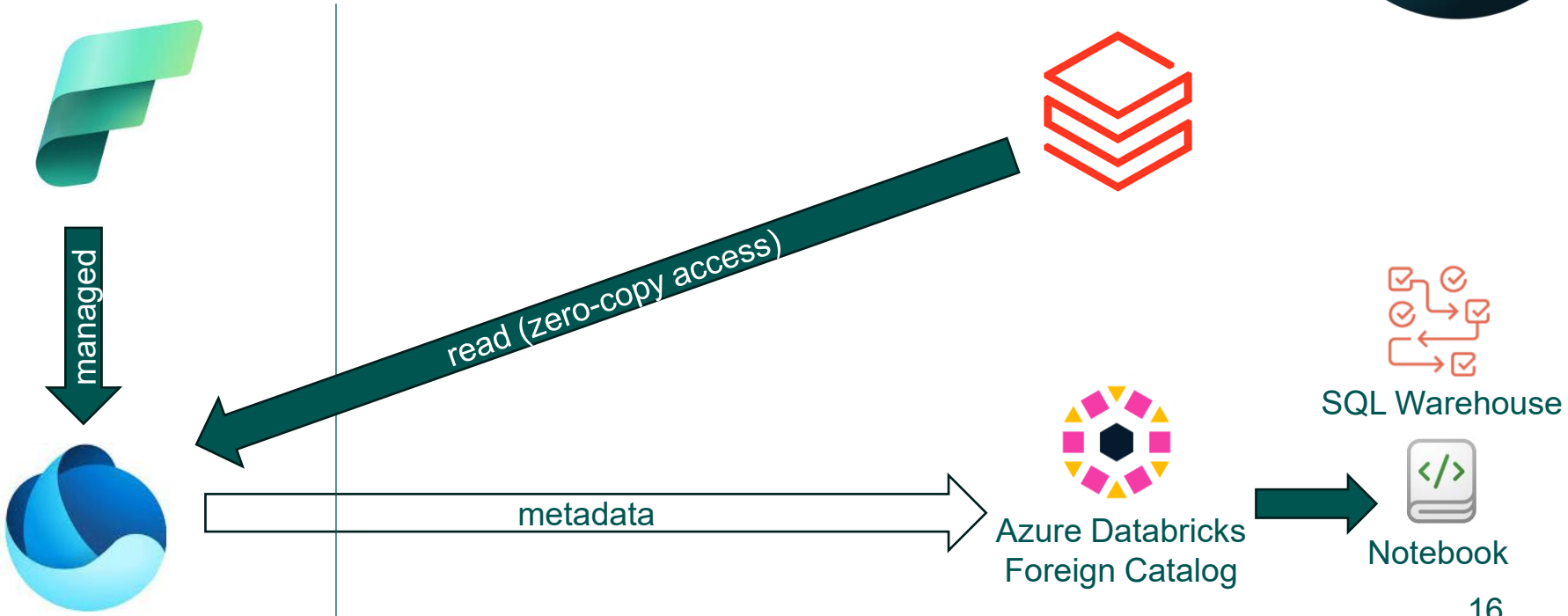
dt = DeltaTable(onelake_uri, storage_options={"bearer_token": f"{access_token}", "use_fabric_endpoint": "true"})
df = dt.to_pandas()
df.head()
```

Mount the Lakehouse path with a Service-Principal authentication

```
source = "abfss://<insert-workspace-id>@onelake.dfs.fabric.microsoft.com/<insert-lakehouse-id>/"
configs = {
    "fs.azure.account.auth.type": "OAuth",
    "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
    "fs.azure.account.oauth2.client.id": client_id,
    "fs.azure.account.oauth2.client.secret": client_secret,
    "fs.azure.account.oauth2.client.endpoint": f"https://login.microsoftonline.com/{tenant_id}/oauth2/token",
}
dbutils.fs.mount(source=source, mount_point="/mnt/onelake", extra_configs=configs)

sdf = spark.read.format("delta").load("/mnt/onelake/Tables/sampletable")
display(sdf)
```

Onelake as foreign-catalog



Onelake as foreign-catalog



1. Create **credential** object in Unity Catalog
    - Managed Identity (via Databricks Access Connector) and Service-Principal are the supported authentication methods

2. Create **Onelake connection** object in Unity Catalog
    - Use the credential created in step 1
    - Need to provide the Fabric Workspace ID during the creation process

3. Create **foreign catalog** object in Unity Catalog
    - Use the connection created in step 2
    - Need to provide the Fabric Lakehouse ID or Warehouse ID during the creation process
    - Compute to access the data needs **DBR 18+**
    - **Read-only** access to Onelake
    - Only Fabric Lakehouse and Warehouse items are supported

https://learn.microsoft.com/en-us/azure/databricks/query-federation/onelake

SQL
KONFERENZ
2026



Microsoft Fabric

Accessing Databricks data in Fabric (on the storage level)



Mirrored Azure Databricks Catalog

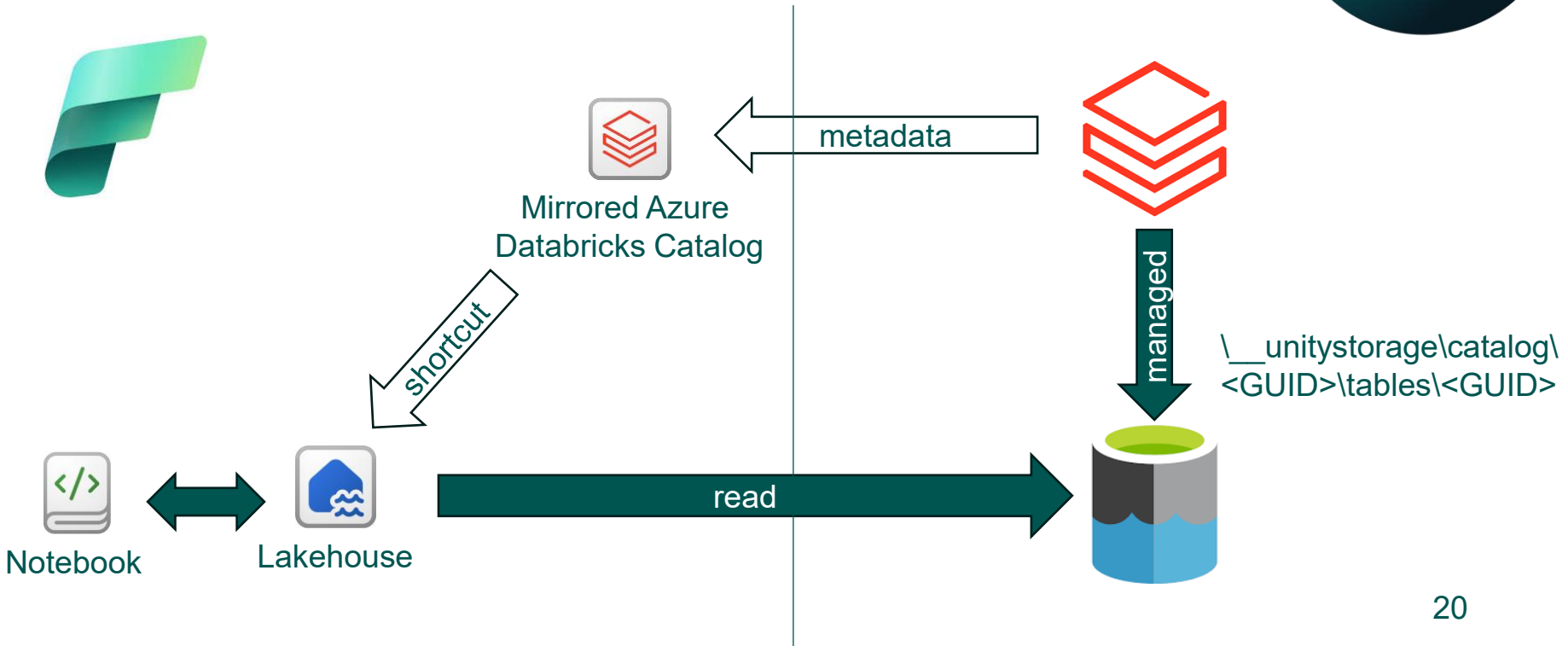
- Separate Item Type/Workload
- Syncs **metadata** from UnityCatalog
- For Databricks managed tables
- **It's a ShortCut, not a Copy/Mirror!**



Shortcut

- Feature of e.g. Lakehouses
- Reference to physical ADLS used by Databricks
- For Databricks external tables
- (technically also for Managed Tables – GUIDs!)

Mirrored Azure Databricks Catalog



Mirrored Azure Databricks Catalog



Prerequisites Databricks

- External Data Access (Preview!)
- Credential with
  - EXTERNAL USE SCHEMA privilege
  - Permissions to schema/tables

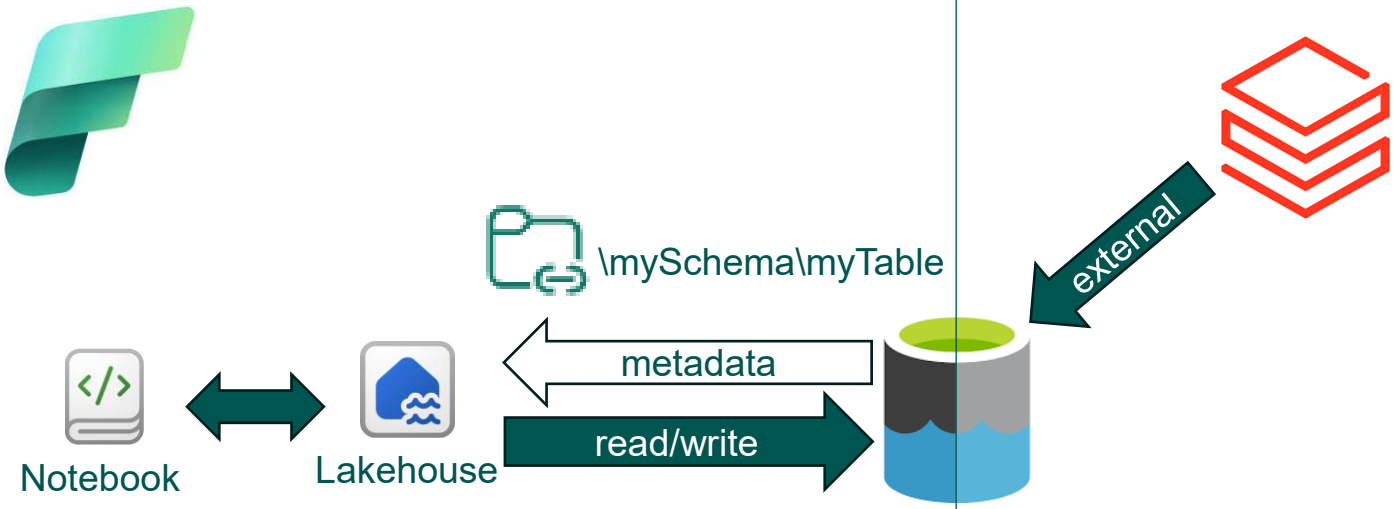
Prerequisites Fabric

- Connection (with Credential)

Gotchas / Findings

- **Read-only**
- All tables must use same ADLS
- Not for tables with RLS/CLS
- No Views, etc.
- Different Delta Lake versions!
- One catalog per mirror
- needs to be linked to a Lakehouse to be used with Notebooks

Shortcut to External Table



Shortcut to External Table



Prerequisites Databricks

- Use of EXTERNAL TABLES on ADLS

Prerequisites Fabric

- Connection with Permissions on ADLS
- Lakehouse

Gotchas / Findings

- No automatic sync
- Add new tables manually
- **Read-write**
- For tables with RLS/CLS (not applied in Fabric!)
- No Views, etc.
- Different Delta Lake versions!

SQL
KONFERENZ
2026

Security & Governance



Security & Governance



Onelake as a **foreign catalog** → Read-only

Direct access to ADLS → Read and Write



Databricks as a **mirrored catalog** → Read-only

Direct access to ADLS (**shortcut**) → Read and Write

Key Limitation

- Security setup are not natively federated between platforms
  - Security based on the Managed-Identity / Service-Principal credentials
- Policy-Weaver

Where RLS/CLS Actually Applies

	Databricks	Microsoft Fabric
• Define RLS/CLS on mirrored/federated catalog?	✓	✓
• Other platforms respect that RLS/CLS?	✗	✗
• External tables (ADLS / Shortcut) enforces RLS/CLS?	✗ (ignored)	✗ (ignored)

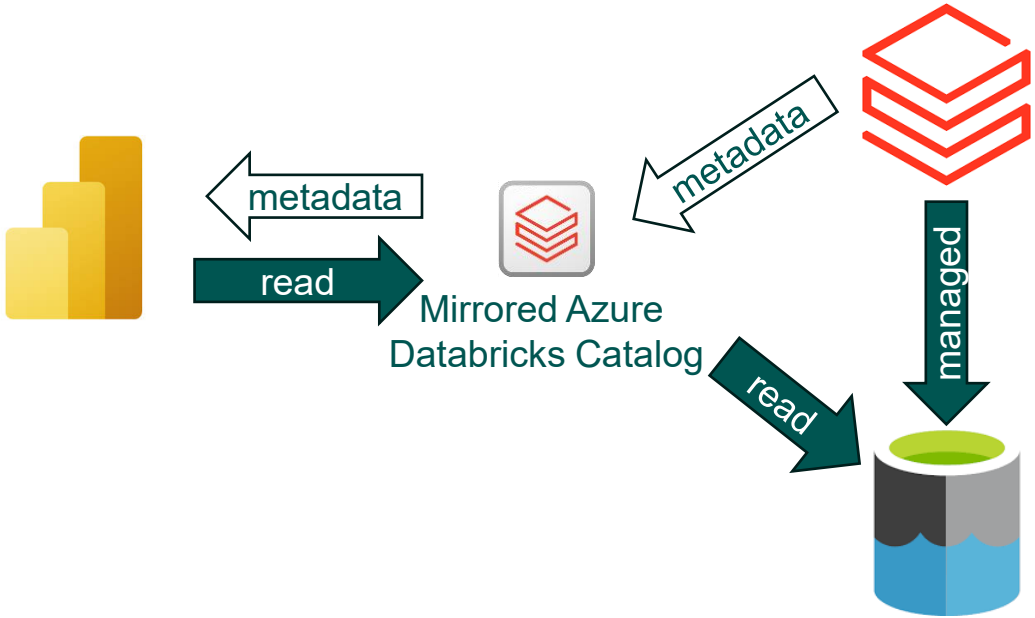
SQL
KONFERENZ
2026

What about [Power] BI



Power BI on Databricks data

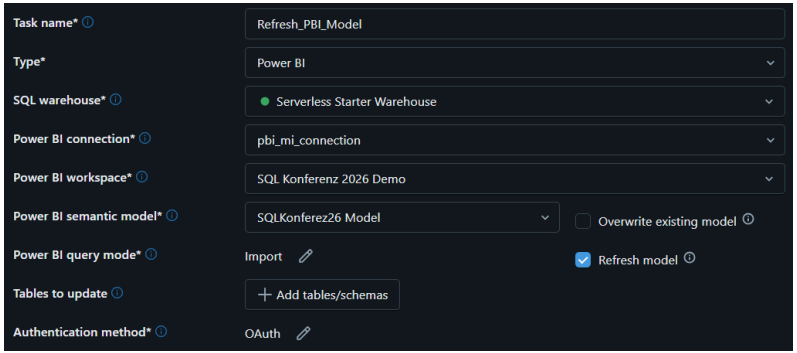
- **Direct Lake**
- Import Mode
  - From Fabric Mirror SQL Endpoint
  - From Databricks SQL WH
- Direct Query
  - On Fabric Mirror SQL Endpoint
  - On Databricks SQL WH



Databricks Power BI Connection

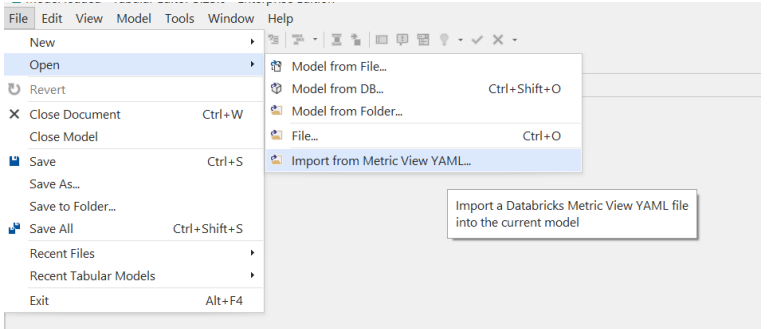
- **Create/replace** a Semantic Model
  - Permissions in Power BI
  - Databricks SQL Warehouse
  - Import or Direct Query – no Direct Lake
  - No DAX or measures
  - Relationship from Unity Catalog

- **Refresh** an existing Semantic Model



Databricks Metric Views

- **Metric views**
  - Define business-ready KPIs directly in Databricks
  - Provide a governed metric layer
- **Power BI** cannot simply consume them → requires materialized views or tables
- Workaround?
  - **Semantic Bridge by Tabular Editor**



https://docs.tabulareditor.com/features/semantic-bridge.html

What we have seen



- Delta Lake is the foundation for sharing data
- Both, Databricks and Fabric can read from each other
- Read-Write is possible but not native
- Governance is not solved yet
- Power BI integrates well

SQL
KONFERENZ
2026



Session Feedback