

Databricks meets PowerBI

ORGANIZED BY  databricks



Gerhard Brueckl

Cloud Data Architect @ paiqo.com



@gbrueckl



blog.gbrueckl.at



gerhard@gbrueckl.at



<https://github.com/gbrueckl>



[DatabricksPS](#)



[Databricks VSCode](#)



DELTA LAKE

[PowerBI Connector](#)



www.paiqo.com



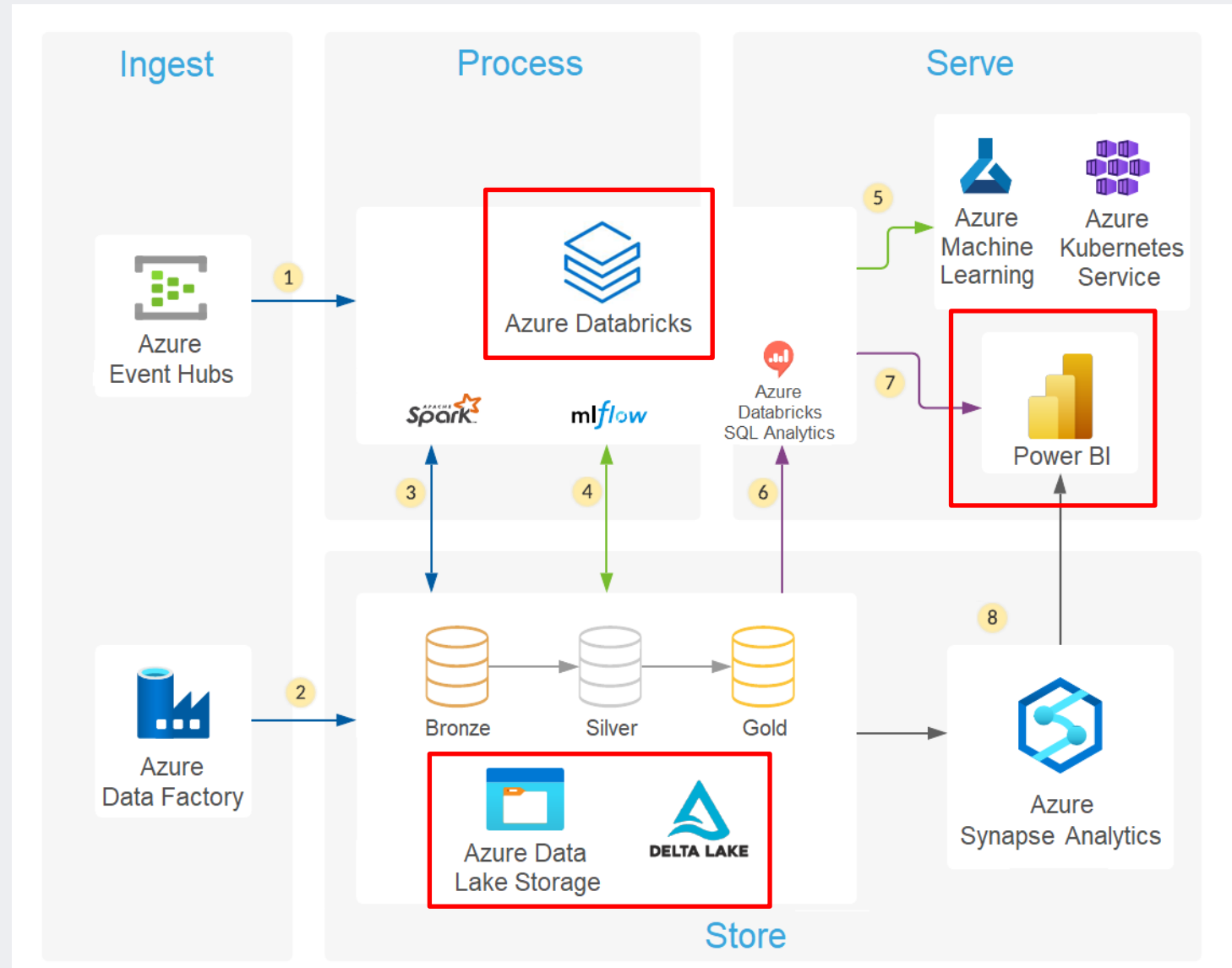
Agenda

- Why this session?
- Querying Databricks from Power BI
 - Connectors
 - Advanced Use-Cases
- Conclusion & Lessons Learned

Why this Session?

Common Architecture

- Results have to be visualized
- Power BI is logical choice for MS/Azure based platforms



Why this session?

- Show different options to connect
- Common pitfalls
 - Performance
 - Costs
- Best-practices / guidelines

Querying Databricks from Power BI

- Connectors -

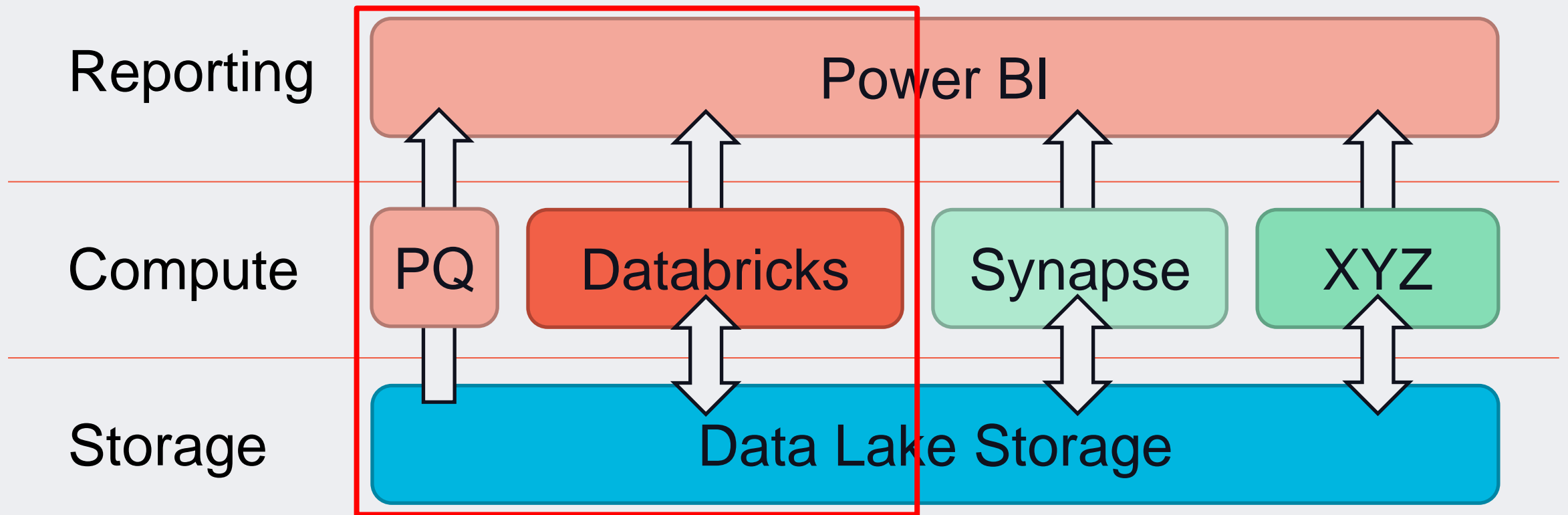
Connecting to Databricks

General

- Separation of Storage & Compute
- Databricks is just the processing engine
- Cluster has to be running to access data
- Data is accessible independently

Connecting to Databricks

General - Power BI and Data Lakes
























Connecting to Databricks






















Import Mode vs. Direct Query

- Import Mode
 - Source (Databricks) is only accessed during processing
 - Further queries are answered by in-memory cache
- Direct Query
 - Every query is sent to Databricks
 - Even a simple Power BI report can create 10s or 100s of queries!

Connecting to Databricks

Connectors

Feature	Spark Connector	Databricks Connector	File / Storage access
Power BI Desktop			
Power BI Service			
Direct Query (Desktop)			
Direct Query (Service)			
Import Mode			
Manual Refresh (Service)			 *
Scheduled Refresh (Service)			 *

Feature	Spark Connector	Databricks Connector	File / Storage access
Azure Active Directory (AAD) Authentication			 *
Personal Access Token Authentication			
Username/Password Authentication			
General Available			
Performance Improvements with Spark 3.x			
Supports On-Premises data gateway			
No running cluster needed			

Connecting to Databricks

Databricks Connector

The image shows a configuration dialog for connecting to Azure Databricks. The dialog is titled "Azure Databricks" and has a close button (X) in the top right corner. It is divided into two main sections: a configuration area on the left and an authentication area on the right.

Configuration Area (Left):

- Server Hostname** (with an information icon):
- HTTP Path** (with an information icon):
- Advanced Options (optional)** (with an information icon):
 - Database (optional)** (with an information icon):
 - Fast Evaluation (optional)** (with an information icon):
- Data Connectivity mode** (with an information icon):
 - Import
 - DirectQuery

Authentication Area (Right):

- Header: **Azure Databricks**
- Text: "You aren't signed in."
- Buttons: "Sign in", "Back", "Connect", "Cancel".

Bottom Buttons: "OK", "Cancel".

Connecting to Databricks

Connection properties

Azure Databricks

Server Hostname ⓘ
adb-72437869625-████████.azuredatabricks.net

HTTP Path ⓘ
sql/protocolv1/o/7243786962536639/0914-112957-████████

Advanced Options (optional)

Database (optional) ⓘ
Example: abc

Fast Evaluation (optional) ⓘ
[Dropdown]

Data Connectivity mode ⓘ
 Import
 DirectQuery

OK

Advanced Options

Azure Data Lake Storage Credential Passthrough ⓘ
 Enable credential passthrough for user-level data access

Spark Tags Logging Init Scripts JDBC/ODBC Permissions

Server Hostname
████████.azuredatabricks.net

Port
443

Protocol
HTTPS

HTTP Path
sql/protocolv1/o/2749489570386029/0213-092211-████████ (unique)
sql/protocolv1/o/2749489570386029/cluster01 (alias, not guaranteed unique)

JDBC URL ⓘ
jdbc:spark://westeurope.azuredatabricks.net:443/default;transportMode=http;ssl=1;httpPath=sql/protocolv1/o/2749489570386029/0213-092211-████████AuthMech=3;UID=token;PWD=<personal-access-token>

Connecting to Databricks

Table Selector

The screenshot displays the Databricks Table Selector interface. On the left, the 'Navigator' pane shows a tree view of folders and tables. The 'dimaccount' table is selected. On the right, the 'dimaccount' table is previewed, showing columns: AccountKey, ParentAccountKey, AccountCodeAlternateKey, and ParentAccountCodeAlternateKey. The data is as follows:

AccountKey	ParentAccountKey	AccountCodeAlternateKey	ParentAccountCodeAlternateKey
1	<i>null</i>		1
2	1		10
3	2		110
4	3		1110
5	3		1120
6	5		1130
7	5		1140
8	3		1150
9	3		1160
10	9		1162
11	9		1164
12	9		1166
13	3		1170
14	3		1180
15	3		1185
17	2		1200
18	17		1210
19	17		1220
20	17		1230
21	17		1240
22	17		1250
23	17		1260
24	2		1300

Connecting to Databricks

Power Query

The screenshot displays the Power Query Editor interface. The ribbon includes tabs for File, Home, Transform, Add Column, View, Tools, and Help. The 'Transform' tab is active, showing options like Sort, Split Column, Group By, and Replace Values. The 'Query' group contains options like Manage Columns, Reduce Rows, and Refresh Preview. The 'Data Type' dropdown is set to 'Whole Number'. The 'Query Settings' pane on the right shows the 'APPLIED STEPS' list, which includes 'Source', 'Navigation', 'Filtered Rows', and 'Calculated Absolute Value'. The main data view shows a table with columns 'ParentAccountKey' and 'AccountCodeAlternateKey'. A context menu is open over the 'AccountCodeAlternateKey' column, listing various actions such as 'Copy', 'Remove', 'Duplicate Column', and 'Between...'. The 'Between...' option is currently selected.

ParentAccountKey	AccountCodeAlternateKey
1	1
2	2
3	3
3	3
5	5
5	5
17	17
17	17
17	17
17	17
17	17
17	17

Power Query

General

- Databricks.Contents (legacy)
- Databricks.Catalogs
 - Supports Databricks Unity Catalog
 - Supports Query Push Down
- Databricks.Query()
 - Custom SQL Query
 - Allows full flexibility and is always pushed down
 - Like running the SQL query in a notebook

Connecting to Databricks

File/Storage Connectors

- Simply connect to storage and read files
 - Any storage supported by Power BI
 - Azure, HDFS, local storage, ...
- Supported file formats by Databricks **AND** PowerBI
 - CSV/JSON/XML
 - Parquet
 - Delta (with connector)

	ABC 123 Content	A ^B C Name	A ^B C Extension
30	Binary	0000000000000000000014.crc	.crc
31	Binary	0000000000000000000014.json	.json
32	Binary	0000000000000000000015.crc	.crc
33	Binary	0000000000000000000015.json	.json
34	Binary	_last_checkpoint	
35	Table	_delta_log	
36	Binary	part-00000-21c03d46-a7a6-4546-8bbd-d5da1361d7d0-c000.snappy.p...	.parquet
37	Binary	part-00000-3777948e-ad1e-4439-99d1-3b7369561e0e-c000.snappy.p...	.parquet
38	Binary	part-00000-39e8a6ec-53a4-42bc-835c-af7227be9d6c-c000.snappy.par...	.parquet
39	Binary	part-00000-473fccda-0826-442a-b0c7-5b077069e89f-c000.snappy.par...	.parquet

Connecting to Databricks

Delta Lake Connector

- Read Delta tables natively in Power BI using Power Query
 - No cluster necessary
 - Supports any storage service supported by Power BI
 - Import Mode only

<https://delta.io/connectors>

Source (release):

<https://github.com/delta-io/connectors/tree/master/powerbi>

Source (pre-release):

<https://github.com/gbrueckl/connectors/tree/master/powerbi>



Power BI Delta Connector

Reading Delta Lake tables natively in PowerBI The provided PowerQuery/M function allows you to read a Delta Lake table directly from any storage supported by PowerBI. Most common storage would be Azure Data Lake Store, Azure Blob Storage, or a local folder or file share.

Power BI

PowerQuery · MIT License

Connecting to Databricks

Delta Lake Connector

The screenshot shows the PowerBI_Delta - Power Query Editor interface. The ribbon includes tabs for File, Home, Transform, Add Column, View, Tools, and Help. The main area displays a query for an Azure Storage Data Lake with the following M code:

```
= AzureStorage.DataLake("https://gbad1s01.dfs.core.windows.net/public/powerbi_delta/DimProduct.delta",
```

The data table below shows the following columns: Content, Name, Extension, and Date accessed. The data is as follows:

Content	Name	Extension	Date accessed
29 Binary	00000000000000000000000013.json	.json	nu
30 Binary	00000000000000000000000014.crc	.crc	nu
31 Binary	00000000000000000000000014.json	.json	nu
32 Binary	00000000000000000000000015.crc	.crc	nu
33 Binary	00000000000000000000000015.json	.json	nu
34 Binary	_last_checkpoint		nu
35 Binary	part-00000-21c03d46-a7a5-4546-8bbd-d5da1361d7d0-c000-cnaayv-n	parquet	nu

At the bottom, it indicates "8 COLUMNS, 50 ROWS" and "Column profiling based on top 1000 rows".

Connecting to Databricks

Delta Lake Connector

The screenshot shows the Power Query Editor interface. The ribbon includes tabs for File, Home, Transform, Add Column, View, Tools, and Help. The Transform ribbon is active, showing various data manipulation options. On the left, the 'Queries' pane shows a list of queries, with 'fn_ReadDeltaTable' highlighted in a red box. The main area displays the configuration for the 'fn_ReadDeltaTable' function. The 'Enter Parameters' section shows a dropdown menu for 'DeltaTableFolderContent' with 'Content_ADLS_DimProduct' selected, also highlighted in a red box. Below the parameters are 'Invoke' and 'Clear' buttons. The function description states: 'Takes the file/folder list of a Delta Lake table and returns the content as a table object in Power Query. An optional 2nd parameter can be used to for special features like Time Travel, Partition Elimination, etc.' The function signature is: `function (DeltaTableFolderContent as table, optional DeltaTableOptions as nullable record) as table`. An 'Example' section shows the usage:

```
let Source = AzureStorage.Blobs("https://gbadls01.blob.core.windows.net/public"), #\"Filtered Rows\" = Table.SelectRows(Source, each Text.StartsWith([Name], \"powerbi_delta/FactInternetSales_part.delta/\")), DeltaTable = fn_ReadDeltaTable(#\"Filtered Rows\", [Version=71])
```

Connecting to Databricks

Delta Lake Connector

The screenshot shows the Power Query Editor interface. The ribbon includes tabs for File, Home, Transform, Add Column, View, Tools, and Help. The main area displays a query named "fn_ReadDeltaTable" with the function name "Content_ADLS_DimProduct, []" highlighted in red. Below the query name is a table with 12 rows and 6 columns: ProductKey, ProductAlternateKey, ProductSubcategoryKey, WeightUnitMeasureCode, and SizeUnitMeasureCode. The "Invoked Function" in the Queries list on the left is also highlighted in red. The bottom status bar indicates "26 COLUMNS, 395 ROWS" and "Column profiling based on top 1000 rows".

	ProductKey	ProductAlternateKey	ProductSubcategoryKey	WeightUnitMeasureCode	SizeUnitMeasureCode
1	212	HL-U509-R	31	null	
2	213	HL-U509-R	31	null	
3	214	HL-U509-R	31	null	
4	215	HL-U509	31	null	
5	216	HL-U509	31	null	
6	217	HL-U509	31	null	
7	218	SO-B909-M	23	null	
8	219	SO-B909-L	23	null	
9	220	HL-U509-B	31	null	
10	221	HL-U509-B	31	null	
11	222	HL-U509-B	31	null	
12	223	CA-1000	10	null	

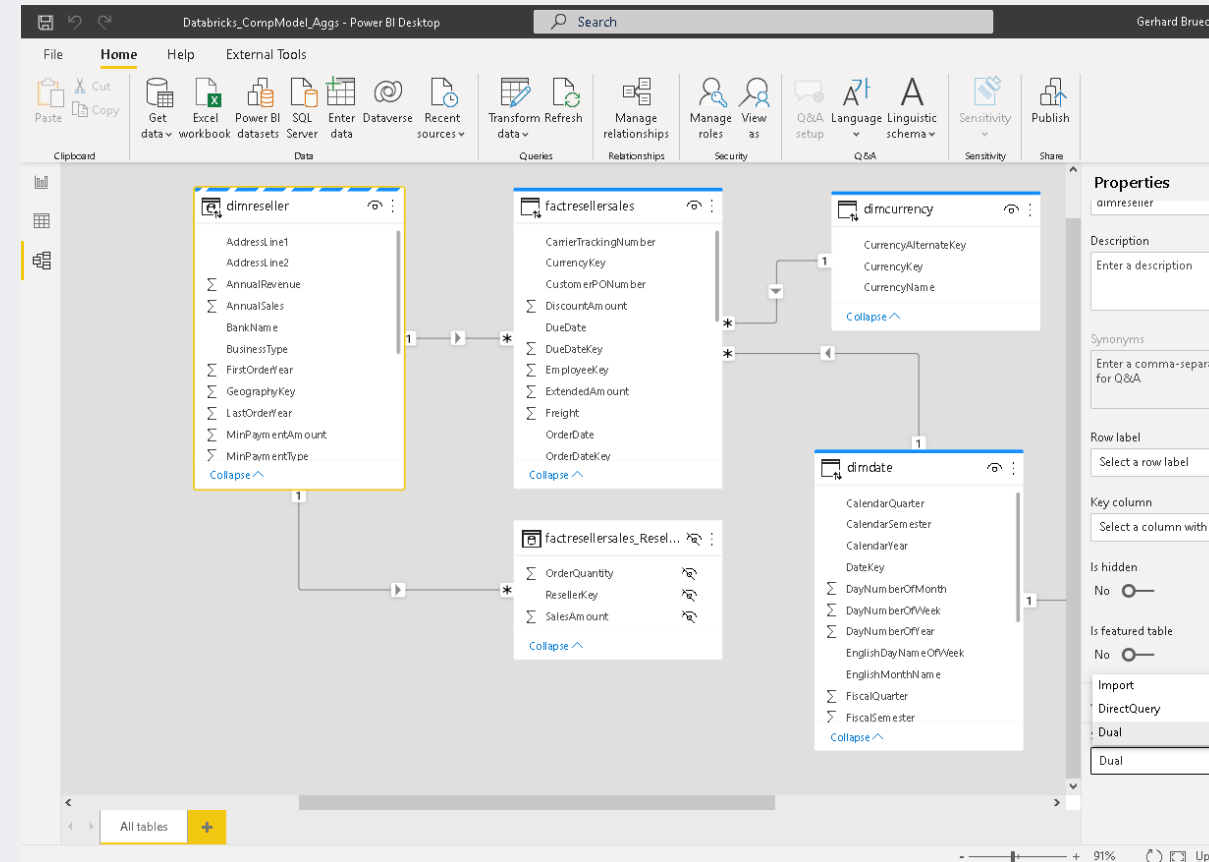
Querying Databricks from Power BI

- Advanced Use-Cases -

Connecting to Databricks

Composite Models and Aggregations

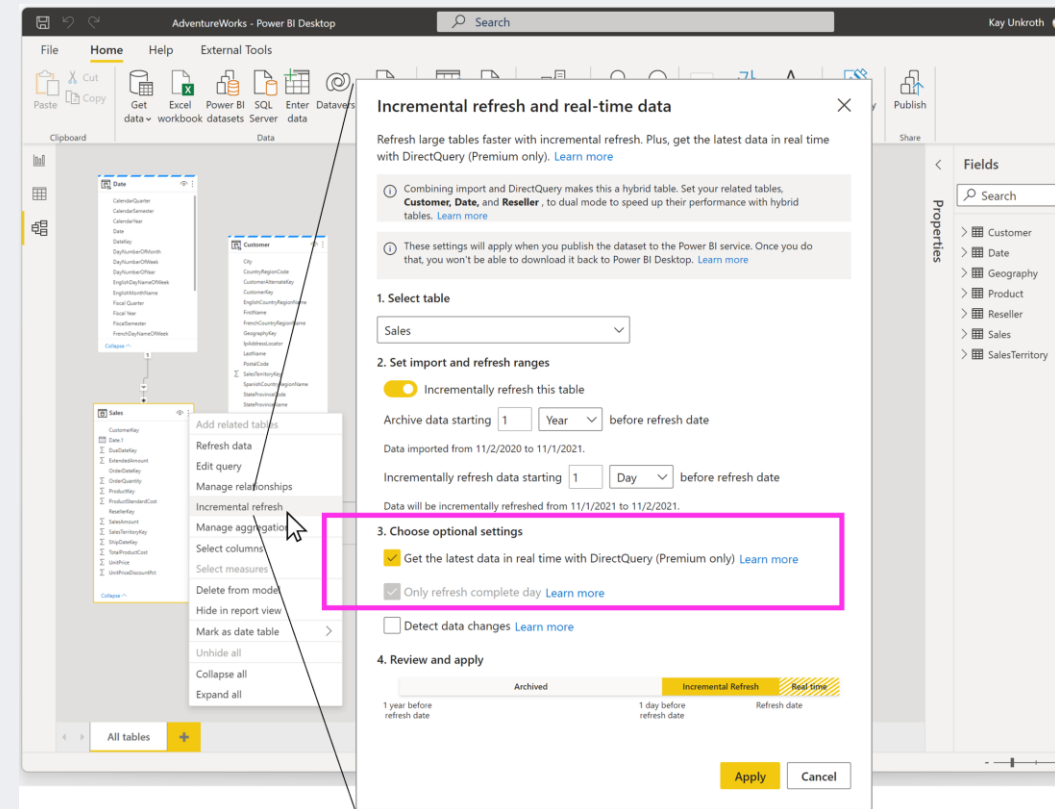
- Composite Models
 - Combine Import Mode and Direct Query
- Aggregations
 - Store pre-aggregated data in separate table for better performance
- Composite Models & Aggregations
 - Import Aggregations and Dimensions
 - Direct Query for Details
 - keep data model small and fast, use backend for details



Connecting to Databricks

Hybrid Tables

- Combine Import mode and Direct Query in one table
 - DQ for Archive and Real-Time
 - Import for latest periods
- Same limitations as in DQ
 - Calculated columns
 - DAX functions
- Public Preview (June 2022)

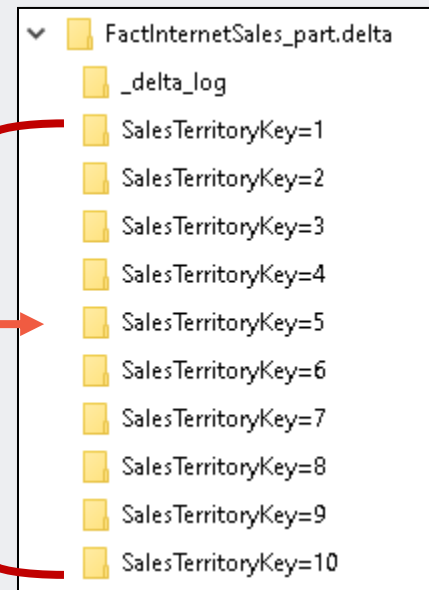


Connecting to Databricks

Partition Pruning/Elimination

```
SELECT *  
FROM FactInternetSales_part  
WHERE SalesTerritoryKey=5
```

```
SELECT *  
FROM FactInternetSales_part  
WHERE OrderDateKey=20210915
```



Conclusion & Lessons Learned

Conclusion & Lessons Learned

Connectors

- Azure Databricks Connector
 - Complex join logics and views
 - Composite Models, Aggregations, Hybrid Tables
- Delta Lake Connector
 - Small amounts of data
 - Import mode only

Conclusion & Lessons Learned

Best Practices & Performance

- Use Views in Databricks (or pre-calculated tables)
 - If data is used multiple times
- Aggregate in Power BI / Power Query
 - If you cannot create views
 - Make sure query is pushed down to Databricks!
- Beware of data volumes!
- Know your partitions!
- Use a proper cluster (size, node-type, ...)

DATA+AI
SUMMIT 2022

Thank you



Gerhard Brueckl

Cloud Data Architect @ paiqo.com