SQL Day

14 edycja konferencji SQLDay

9-11 maja 2022, WROCŁAW + ONLINE

Data Community

partner złoty

Future Processing

Infinite DATA

KPMG

partner srebrny

Objectivity

summ-it

UBS

dbWatch DATABASE CONTROL

partner brązowy

elitmind think bright

Data Community

Bartek Graczyk, Paweł Potasiński

The Bad and The Ugly - 10 steps to cure your Azure Synapse serverless SQL pool performance and optimize costs

SQL Day

SPEAKERS

Bartek

Work Lead Cloud Solution Architect @ Microsoft

LinkedIn linkedin.com/in/bartlomiejgraczyk/

Twitter @GraczykBartek

Paweł

Work Sr Program Manager @ Microsoft

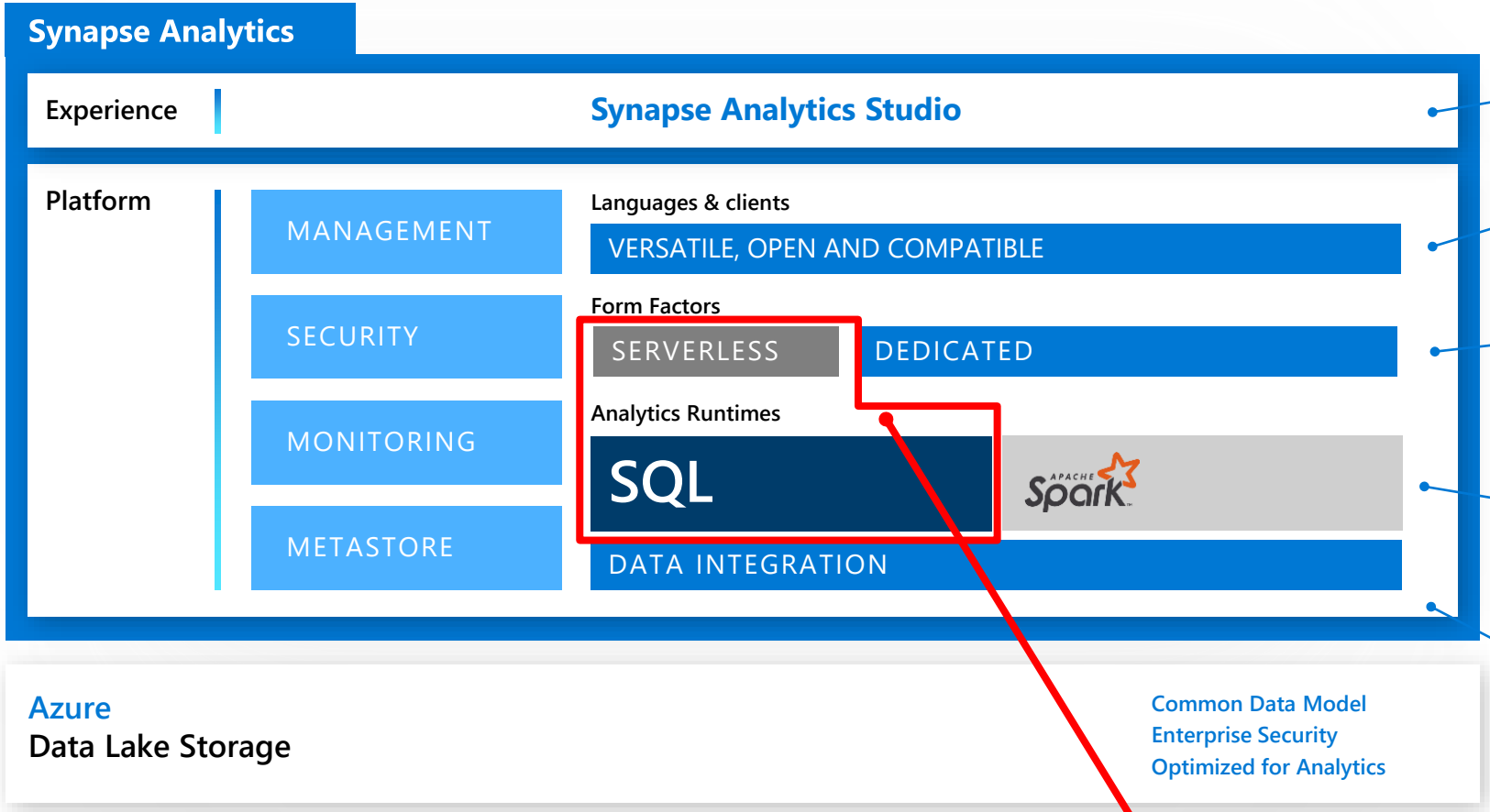
LinkedIn linkedin.com/in/pawelpotasinski/

Twitter @PawelPotasinski

AGENDA

- Introduction to Azure Synapse Analytics
- Scenarios for Azure Synapse serverless SQL
- 10 steps to improve performance and optimize cost
- Some other things to consider
- Summary and resources

Azure Synapse Analytics



SaaS **developer experiences**

Multiple **languages & clients** suited to different workloads

Analytics runtimes available **provisioned** and **serverless on-demand**

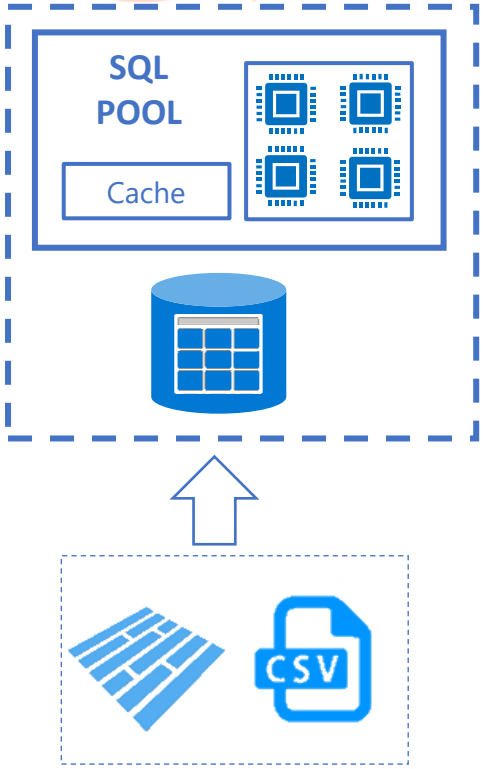
Synapse SQL for T-SQL batch, streaming and interactive processing

Spark for big data processing with Python, Scala, R and .NET

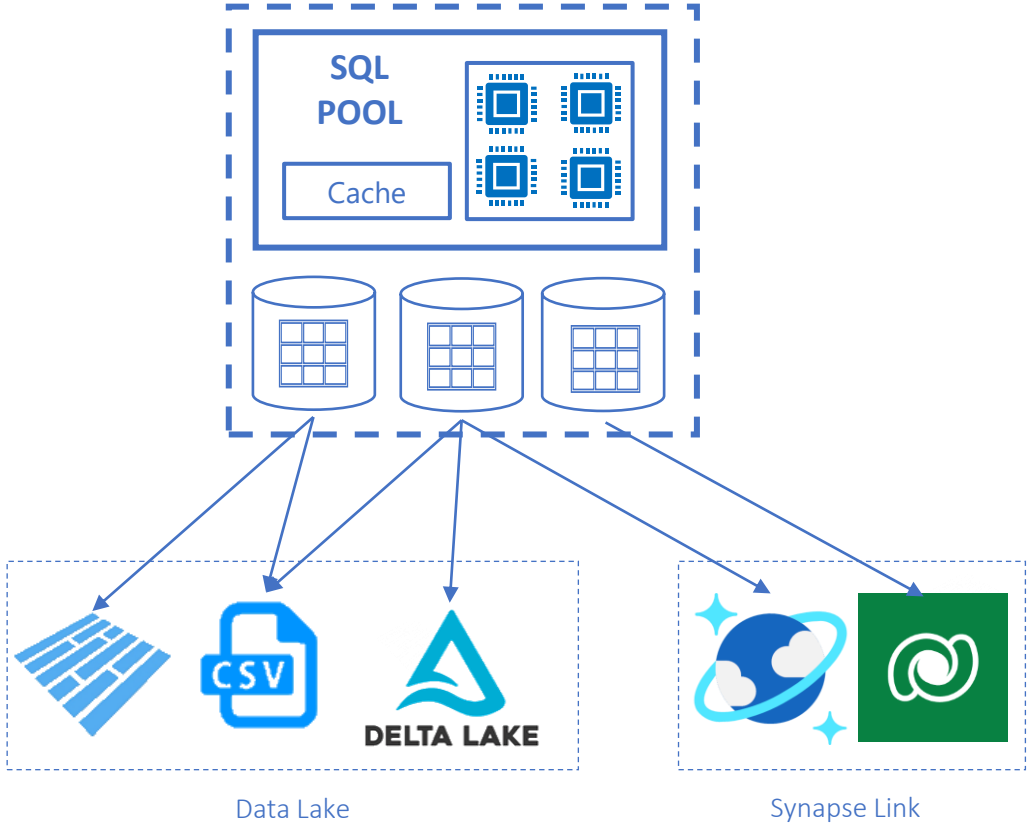
Integrated **platform services** for, management, security, monitoring, and metastore

Focus of this session

Synapse SQL pools



Dedicated SQL pools



Serverless SQL pools

https://aka.ms/synapse-dqp

Serverless SQL pool

Overview

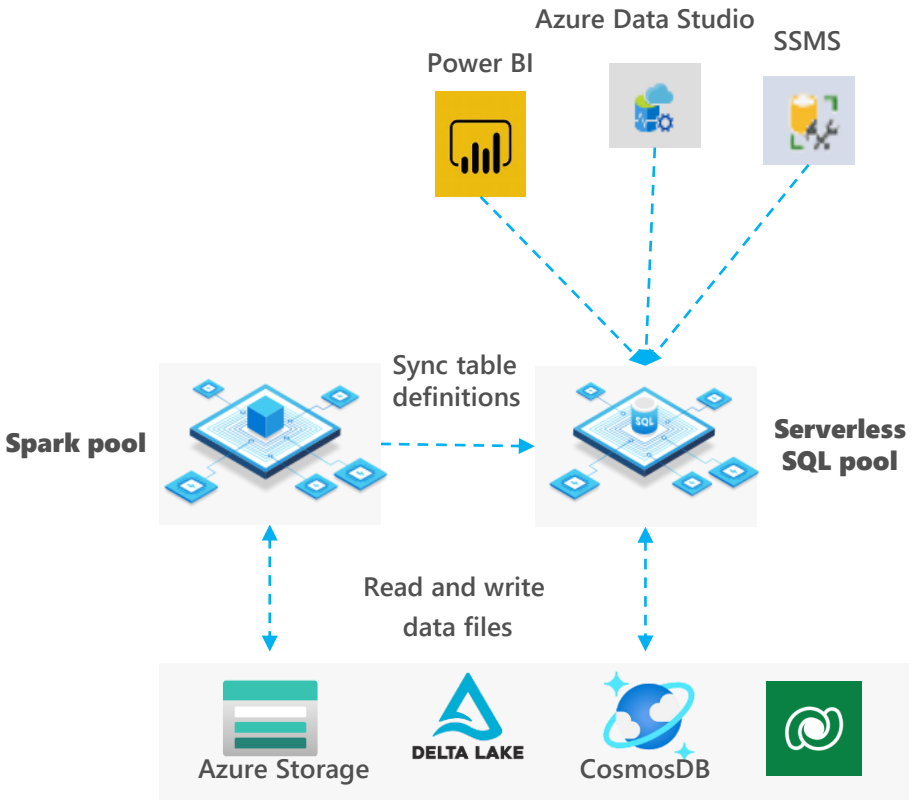
An interactive query service that enables you to use standard T-SQL queries over Data Lake, Cosmos DB, Dataverse, …

Benefits

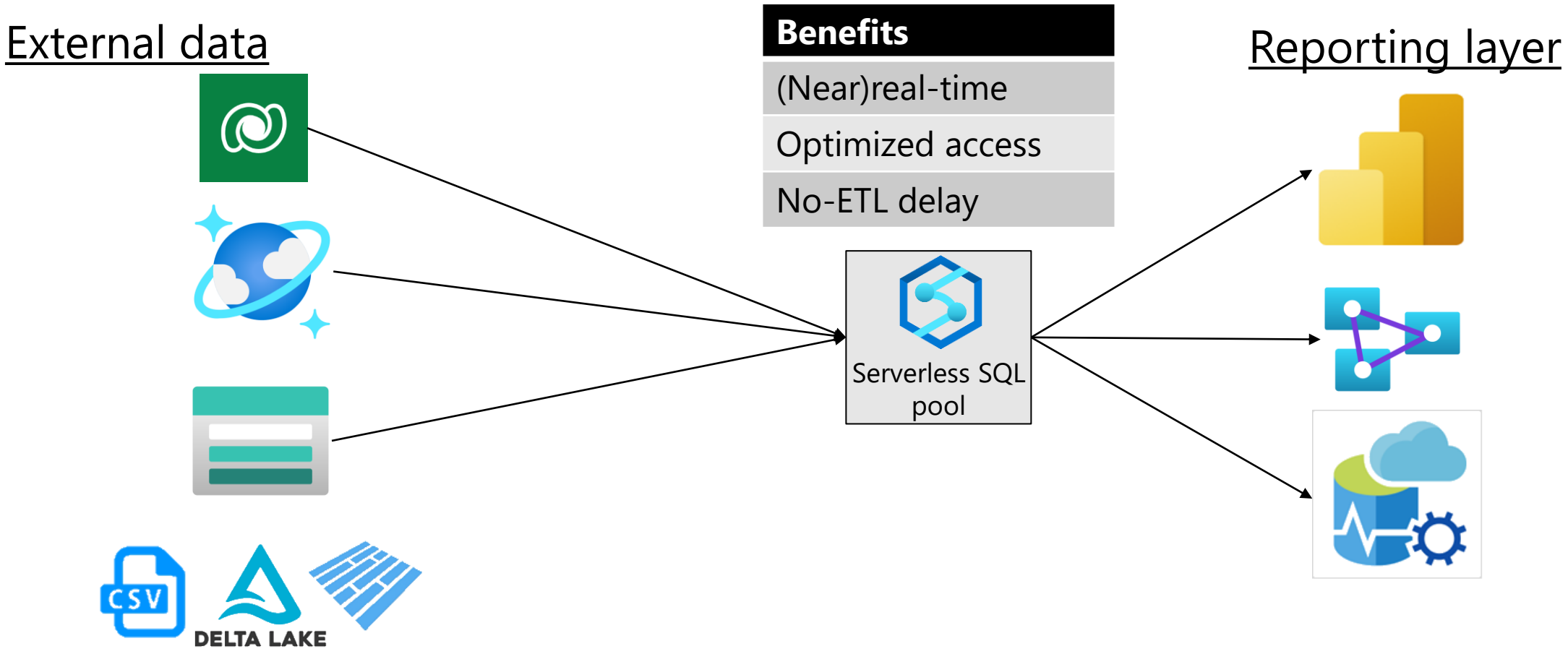
- Use T-SQL language
- Supports any tool or library that uses T-SQL to query data
- Automatically synchronize tables from Spark pool
- Querying multiple storages (Lake, CosmosDB)

Serverless experience

- Auto Scale & Manage
- Pay-per-use model
- Easy to use
- Automatic schema inference

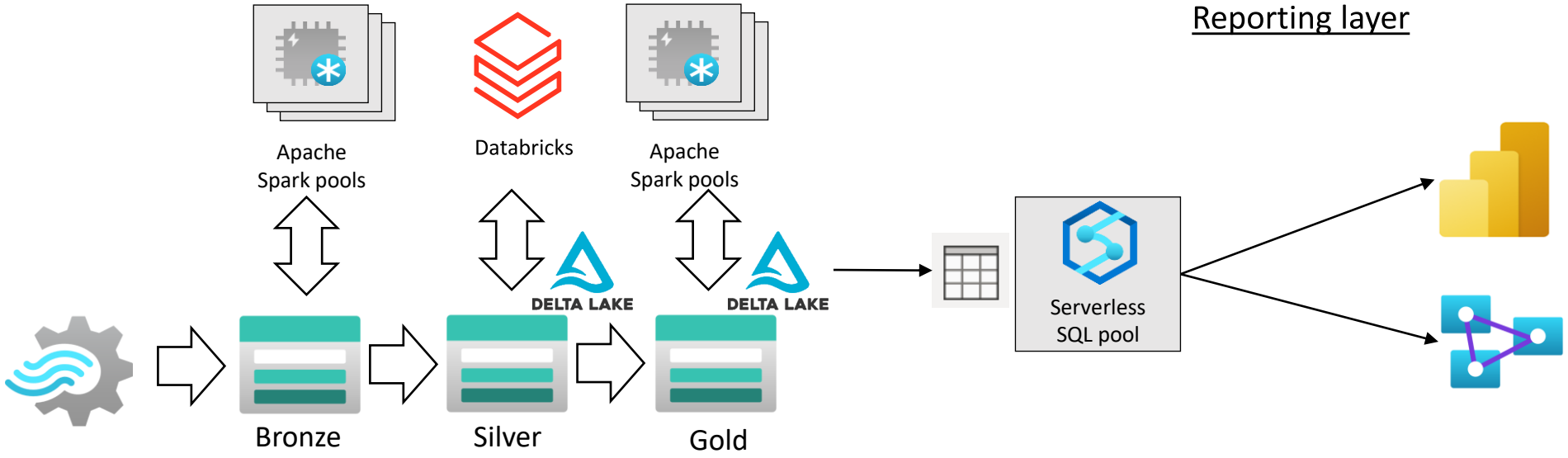


Serverless SQL pool virtualization scenarios



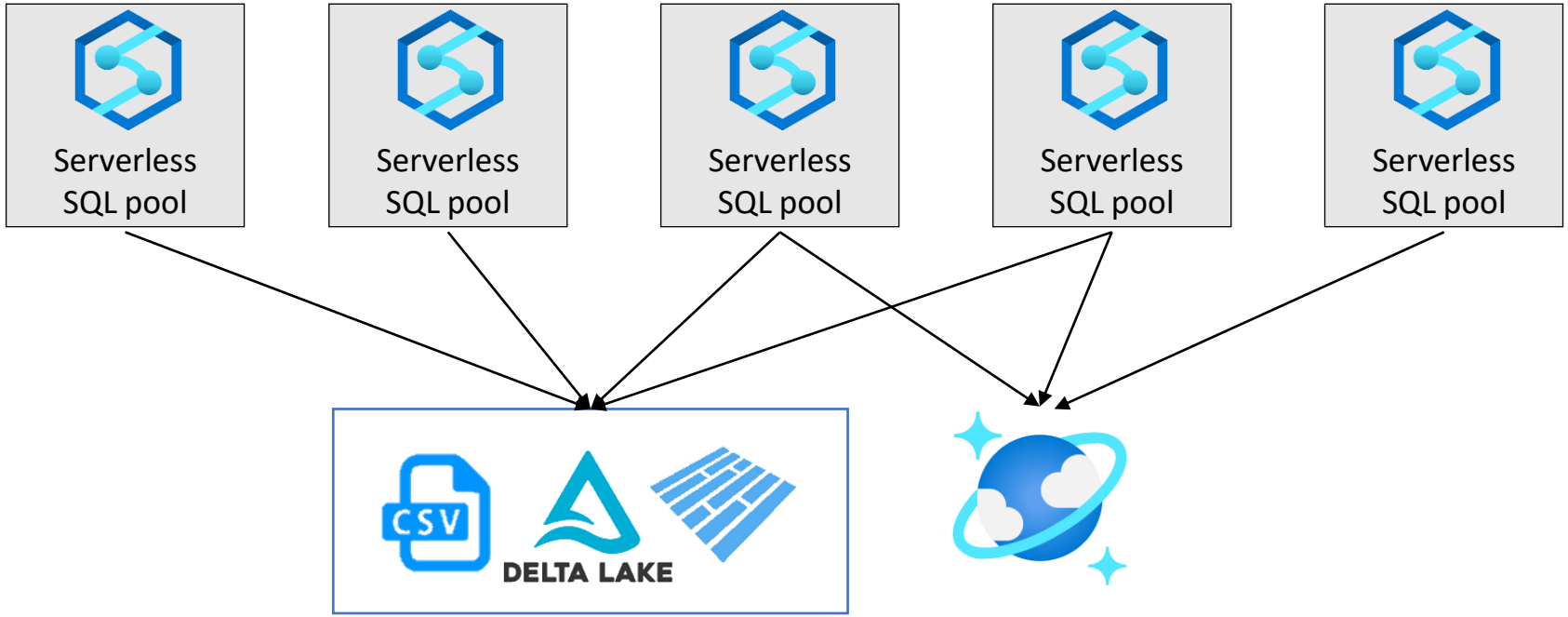
Serving layer for reporting tools

- Data engineers refine shared data using various tools
  - Data is updated through the multiple layers (bronze, silver, gold)
- Data analysts create reports using serverless SQL pools
  - Serverless SQL pools is a bridge between reporting tools and data

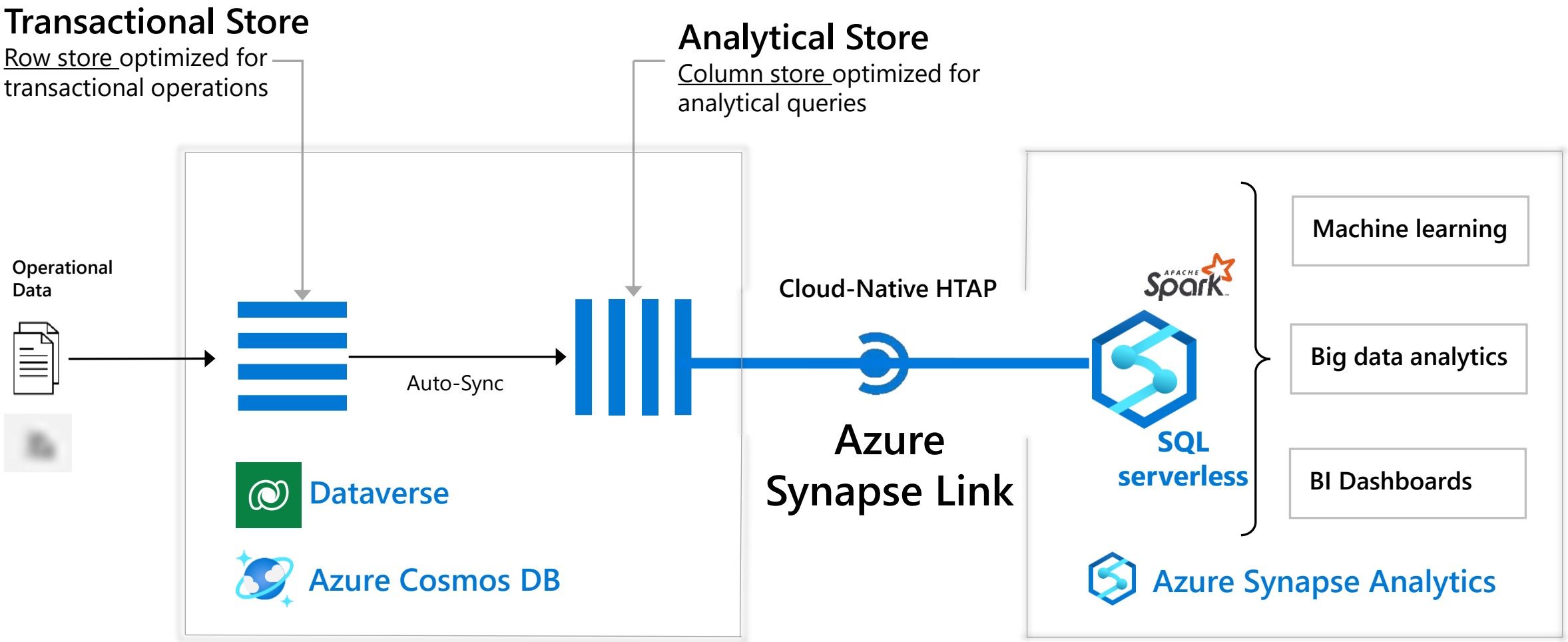


Sharing data layer

- Multiple workspaces sharing the same data
- No data movement/imports



Synapse link – near real-time analytics



ARCHITECTURE: Collocate resources

- Make sure that Azure Synapse **workspace** and **data sources** (ADLS/Cosmos DB) are placed in the **same region** to avoid read latency

- Make sure that Azure Synapse **workspace** is placed in the **same region** as **clients** (VMs/Power BI/AAS), or the region that is close to the clients

ARCHITECTURE: Use caching and optionally multiple workspaces

- Use **caching** (Power BI Import Mode) for the large data sets or to ensure interactive analytics

- Use **multiple workspaces** if you are hitting resource limits or require chargeback mechanism for serverless SQL
  - Serverless compute is limited compute with instant auto pause/resume
  - Drawbacks
    - Need to load-balance workload
    - Need to manually keep schema in Sync

STORAGE: Layout the files

- Keep your **file size** in the range **between 100 MB and 10 GB**
- If a query targets a single large file, you'll benefit from splitting it into **multiple smaller files**
  - Single file is processed with a single node
  - Too many files = slow listing
- It's better to have **equally sized files** for a single OPENROWSET path or an external table LOCATION
- **Partition your data** using folders/paths to allow partition elimination and the use of FILEPATH and FILENAME functions

ACCESS: Consider SAS authentication

- If you need better performance, try using **Shared Access Signature (SAS)** credentials to access the storage

- In general, SAS performs better than Azure Active Directory (Azure AD) Pass-through Authentication

4

SCHEMA: Take care of data types

- Use the **smallest data type** that will accommodate the largest possible value

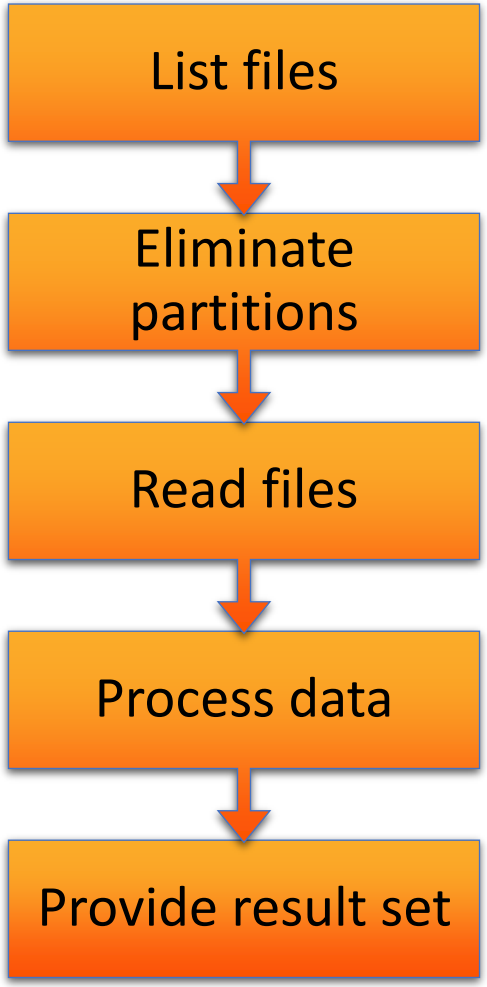
- Some **general rules**:
  - varchar and char instead of nvarchar and nchar
  - varchar of reasonable size instead of varchar(max)
  - char or nchar for fixed length values
  - smallest accommodating type, example: tinyint instead of bigint if largest value fits tinyint

- Use sp_describe_first_resultset stored procedure to get inferred data types

SCHEMA: Take care of data types

- In the partitioned data sets **CAST the FILEPATH()** function to appropriate size to avoid default nvarchar(1024) type
- Use **nvarchar** type if the underlying text is **UTF-16** encoded
- Use **varchar** type with **UTF8** collation if the underlying text is UTF-8 encoded (for example Parquet)
- Use **Latin_General_100_BIN2_UTF8** collation on string partitioning columns or any other column where you will use filters (parquet) [soon to be obsolete]

QUERIES: The process



QUERIES: Understand and monitor data processed

- **Data processed** is the amount of data that the system temporarily stores while a query is run

- Data processed consists of the following **quantities**:
  - Amount of data **read** from storage.
    This amount includes:
    - Data read while reading data
    - Data read while reading metadata (for file formats that contain metadata, like Parquet)
  - Amount of data in **intermediate results**
  - Amount of data **written** to storage

1 TB -> $5

QUERIES: Understand and monitor data processed

EXAMPLE SCENARIOS

- The **population_csv** table is backed by **5 TB of CSV files**. The files are organized in five equally sized columns
- The **population_parquet** table has the same data as the population_csv table. It's backed by **1 TB of Parquet files**. This table is smaller than the previous one because data is compressed in Parquet format
- The **very_small_csv** table is backed by 100 KB of CSV files

QUERIES: Understand and monitor data processed

[5 TB of CSV files]

Query 1: SELECT SUM(population) FROM population_csv

QUERIES: Understand and monitor data processed

[5 TB of CSV files]

Query 1: SELECT SUM(population) FROM population_csv

5 TB (~ 25$)

of data plus a small amount overhead for transferring sums of fragments

QUERIES: Understand and monitor data processed

[1 TB of parquet files]

Query 2: SELECT SUM(population) FROM population_parquet

Query 3: SELECT * FROM population_parquet

QUERIES: Understand and monitor data processed

[1 TB of parquet files]

Query 2: SELECT SUM(population) FROM population_parquet

0.2 TB **(~1$)**

plus a small amount of overhead
for transferring sums of fragments

Query 3: SELECT * FROM population_parquet

If the compression format is 5:1, then the query processes

6 TB because it reads 1 TB and transfers 5 TB of

uncompressed data **(~ 30$)**

QUERIES: Understand and monitor data processed

[100 KB of CSV files]

Query 4: SELECT COUNT(*) FROM very_small_csv

QUERIES: Understand and monitor data processed

[100 KB of CSV files]

Query 4: SELECT COUNT(*) FROM very_small_csv

This query processes
slightly more than 100 KB of data.

The amount of data processed for this query is
rounded up to 1 MB data processed
and charged for 10 MB

QUERIES: Understand and monitor data processed

- Use **Synapse Studio** to monitor queries

SQL script 1

Run | Undo | Publish | Query plan | Connect to: Built-in | Use database: master

```
1  SELECT
2      FORMAT(COUNT(*), '#,#')
3  FROM
4      OPENROWSET(
5          BULK 'https://sqldaydatalake.dfs.core.windows.net/datalake/raw/TPCH_S/LINEITEM/*.parquet',
6          FORMAT = 'PARQUET'
7      ) AS [result];
```

Results | Messages

10:37:43 AM Started executing query at Line 1

Statement ID: {CB3311BF-53AB-4DFF-B3D3-2590263B56D2} | Query hash: 0xA9DA09875DB95EBF | Distributed request ID: {8B76132B-16F1-4540-88FC-D2B98EBFB723}.
Total size of data scanned is 13 megabytes, total size of data moved is 1 megabytes, total size of data written is 0 megabytes.
(1 record affected)

Microsoft Azure | Synapse Analytics ▸ sqlday-synapse

- Home
- Data
- Develop
- Integrate
- Monitor
- Manage

Analytics pools
- SQL pools
- Apache Spark pools
- Data Explorer pools (preview)

Activities
- SQL requests
- KQL requests
- Apache Spark applications
- Data flow debug

SQL requests

Refresh | Edit columns

Filter by keyword | Local time : **Last 7 days** | Status : **All** | Pool : Built-in | Add filter

Showing 1 - 99 of 99 items

Request ID	Request content	Submit time	Duration	Data processed
909114	SELECT FORMAT(C... More	5/2/22, 5:10:52 PM	4 sec	14.00 MB
888781	SELECT l_shipmod... More	5/2/22, 5:02:52 PM	18 sec	10.00 MB
856626	*** Global stats qu... More	5/2/22, 4:43:34 PM	11 sec	44.00 MB
856916	*** Global stats qu... More	5/2/22, 4:43:34 PM	13 sec	44.00 MB

QUERIES: Understand and monitor data processed

- Use **DMVs** to monitor queries
  - `sys.dm_exec_requests` – running queries
  - `sys.dm_exec_requests_history` – historical queries
- Use helper **QPI library**
  - Learn about QPI's capabilies: [article of Jovan Popovic on QPI]
  - Get it from GitHub: [https://github.com/JocaPC/qpi]

Run | Undo | Publish | Query plan | Connect to Built-in | Use database ADLS

```
130 SELECT *
131 FROM qpi.recommendations
```

Results | Messages

View: Table | Chart | Export results

Search

name	score	sch
OPTIMIZE COLUMN TYPE	1	db
OPTIMIZE COLUMN TYPE	1	db
OPTIMIZE COLUMN TYPE	1	db
REMOVE PARTITIONED TABLE	1	db

00:00:01 Query executed successfully.

Run | Undo | Publish | Query plan | Connect to Built-in | Use database ADLS

```
125
126 SELECT *
127 FROM qpi.query_history
128 WHERE query_text_id = 0x59DF9194975DEB9D
```

Results | Messages

View: Table | Chart | Export results

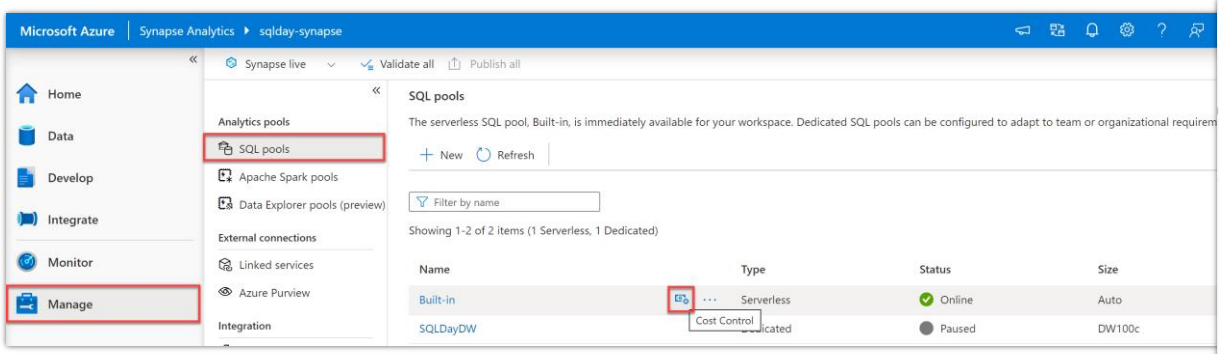
Search

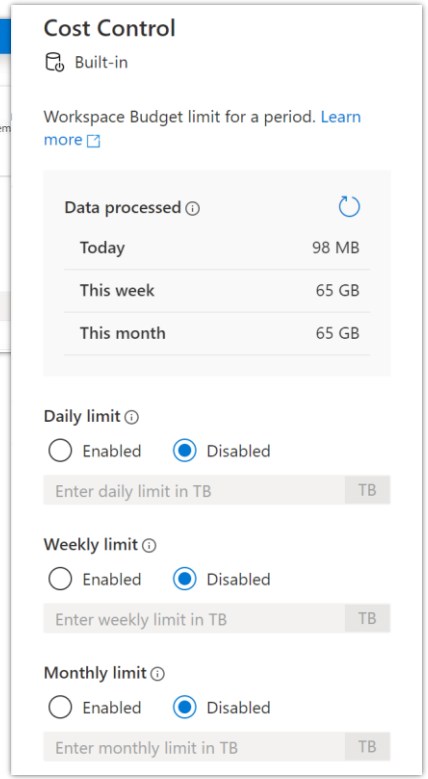
query_text_id	request_id	elapsed_time_s	query_text	data_processe...	start_time	end_time
0x59DF919497...	{83720B11-96A...	0.270000	SELECT TOP 100 * FROM OPENROWSET(BULK 'https://jovanpoptest.dfs.core.wi...	10	2021-11-09T12:...	2021-11-09T
0x59DF919497...	{477C8860-E39...	0.553000	SELECT TOP 100 * FROM OPENROWSET(BULK 'https://jovanpoptest.dfs.core.wi...	10	2021-11-09T12:...	2021-11-09T
0x59DF919497...	{A0E40DE6-105...	0.170000	SELECT TOP 100 * FROM OPENROWSET(BULK 'https://jovanpoptest.dfs.core.wi...	10	2021-11-09T12:...	2021-11-09T

00:00:00 Query executed successfully.

QUERIES: Understand and monitor data processed

- Use **Synapse Studio** to control cost and limits





- Use **system views** to control cost and limits
  - `sys.dm_external_data_processed` – data processed
  - `sys.configurations` – cost control limits

QUERIES: Know how to work with CSV files

- Use PARSER_VERSION 2.0 for querying CSV files
  - Learn the limitations (e.g. no support for LOB)
- Create and maintain statistics for CSV files
  - Optimize DISTINCT, JOIN, WHERE, ORDER BY, and GROUP BY

QUERIES: Use wildcards wisely

- Optimize listing files
  - Move * down to lower levels in the path

```
OPENROWSET(BULK 'https://storageaccount/container/tables/*/*' ...) AS [r]
WHERE r.filepath(1) = 'customers'
```

```
OPENROWSET(BULK 'https://storageaccount/container/tables/customers/*' ...
```

QUERIES: Reduce amount of data for processing

- Use columnar and compressed file format
  - Parquet instead of CSV and JSON
- Use CETAS to prepare frequently used data
  - Example: tables that are often joined
- Read only relevant columns (avoid *)

```
SELECT * FROM
OPENROWSET(BULK 'https://storageaccount/container/customers/*' ...) AS [r]
```

```
SELECT id, name FROM
OPENROWSET(BULK 'https://storageaccount/container/customers/*' ...) AS [r]
```

QUERIES: Eliminate files and folders before reading

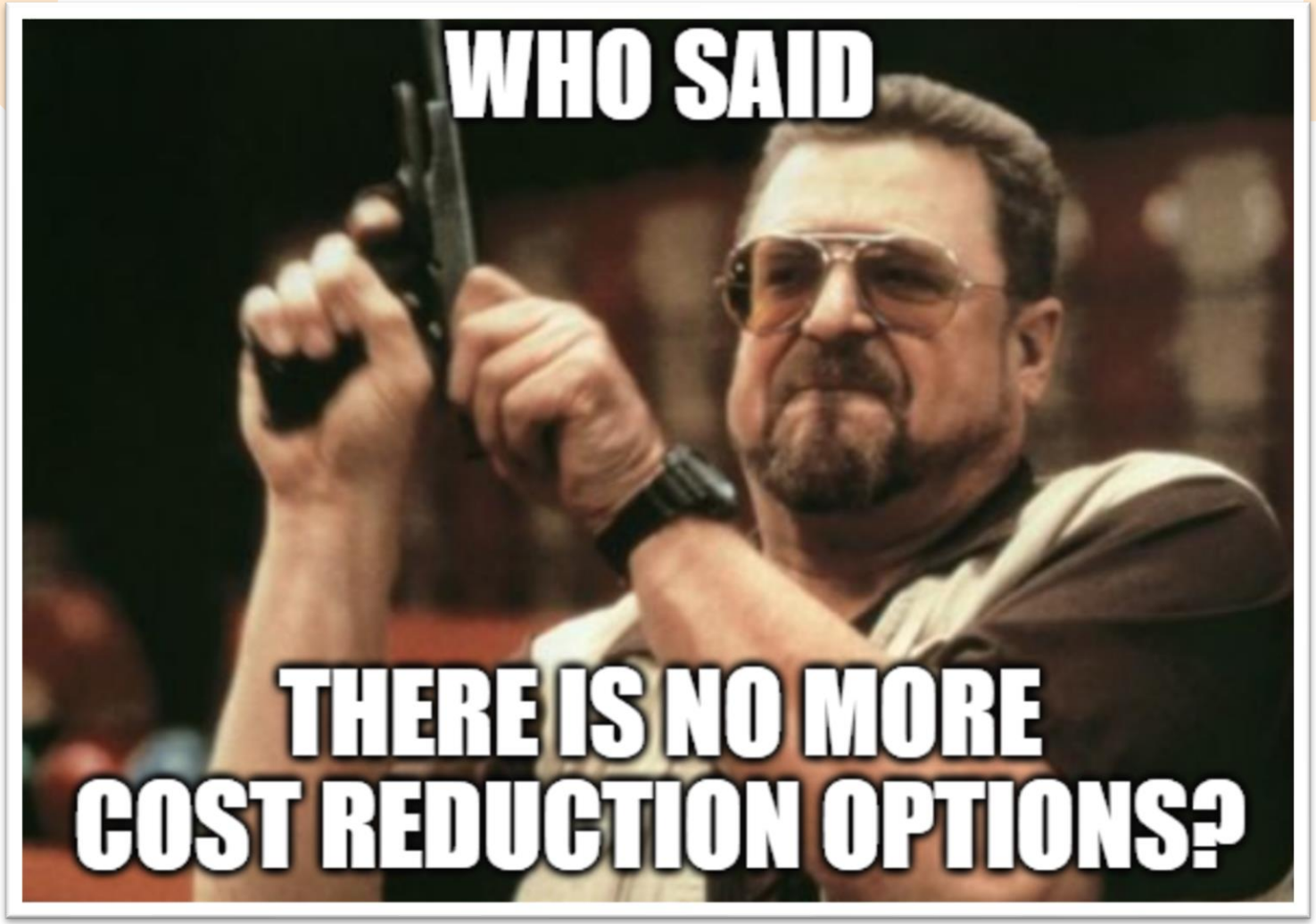
- Utilize partition elimination to target specific files or folders
  - Partitioned external tables support coming soon

```
OPENROWSET(BULK 'https://storageaccount/container/orders/year=*/*' ...) AS [r]
```

```
OPENROWSET(BULK 'https://storageaccount/container/orders/year=*/*' ...) AS [r]
WHERE r.filepath(1) = 2016
```

Some other things to consider

- Maximum result set size ~200 GB
- SSMS and Azure Data Studio are SQL developer’s friends
  - Use when you run multiple queries
  - Use for query troubleshooting
  - Use when experience performance problems in Synapse Studio
- 30 minutes timeout for distributed query execution
- Don't stress the storage account with other workloads


WHO SAID
THERE IS NO MORE
COST REDUCTION OPTIONS?

Azure Synapse Analytics Pre-Purchase Plan (P3)

- A synapse prepurchase applies to all Synapse workloads and tiers
- Pre-Purchase Plan is as a pool of prepaid Synapse commit units
- Usage is deducted from the pool, regardless of the workload or tier
- Unlike VMs, the pre-purchased units don't expire on an hourly basis, and you use them at any time during the term of the purchase
- Any Azure Synapse Analytics use deducts from the pre-purchased SCUs automatically
- Learn more: [Optimize Azure Synapse Analytics costs with a Pre-Purchase Plan | Microsoft Docs]

For example: 5,000 SCUs = 5,000 USD of Synapse compute -> **$ 4,700 (6%)**
24,000 SCUs = 24,000 USD of Synapse compute -> **$21,360 (11%)**

Summary and resources

Summary

1. Collocate resources
2. Use caching and multiple workspaces
3. Layout the files
4. Consider SAS authentication
5. Take care of data types
6. Understand and monitor data processed
7. Know how to work with CSV files
8. Use wildcards wisely
9. Reduce amount of data for processing
10. Eliminate files and folders before reading

Resources

- **https://github.com/DataInsiders/SQLDay2022**
- Best practices for serverless SQL pool (documentation)
- Optimize database schema using QPI library (blog)
- Optimize serverless SQL pool and Synapse link (blog)
- Andy Cutler's serverless SQL pool Utility Scripts (GitHub)

Azure Synapse Resources

Blog
Montly Updates
YouTube
Twitter
Ideas
Documentation
Microsoft Learn
Community Hub

Data Community

Bartek Graczyk, Paweł Potasiński

The Bad and The Ugly - 10 steps to cure your Azure Synapse serverless SQL pool performance and optimize costs

SQL Day

SQL Day

14 edycja konferencji SQLDay

9-11 maja 2022, WROCŁAW + ONLINE

Data Community

partner złoty

Future Processing

Infinite DATA

KPMG

partner srebrny

Objectivity

summ-it

UBS

dbWatch DATABASE CONTROL

partner brązowy

elitmind think bright