

Harnessing Azure Synapse Analytics Serverless SQL Pools with Power BI Dataflows



Andy Cutler
March 2021

www.datahai.co.uk

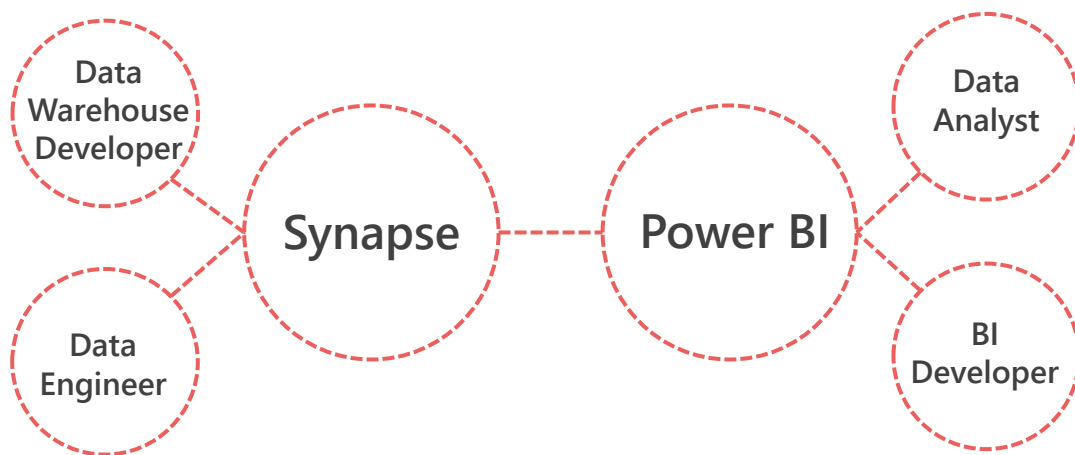
Contents

3. Who is this for?
4. Why are we connecting Power BI to Synapse?
5. What are Power BI Dataflows?
6. What is Azure Synapse Analytics Serverless SQL Pools?
7. Why use Serverless SQL Pools with Dataflows?
8. Serverless SQL Pools and Power BI Dataflows Together
9. Azure Synapse Analytics Ecosystem
10. Connecting to the Serverless SQL endpoint
11. Power BI Query Folding
12. Supported File Formats
13. Which SQL objects can Power BI connect to?
14. Dataflow Incremental Refresh
15. Data Loading Performance Analysis
16. Considerations
17. Serverless SQL Pools monitoring
18. Further Reading

Who is this for?

Azure Synapse Analytics and Power BI have different role profiles. However, there is cross-over in terms of features available in Synapse such as Power BI integration.

Depending on the organisation size and data team maturity, the following roles may be carried out by dedicated teams or role-played amongst individuals.



Data Warehouse Developer

- Building Data Warehouse solutions using Synapse Analytics

Data Engineer

- Extracting, Loading and Transforming data from many sources

Business Intelligence Developer

- Creating centralised data sets and data models with metrics

Data Analyst

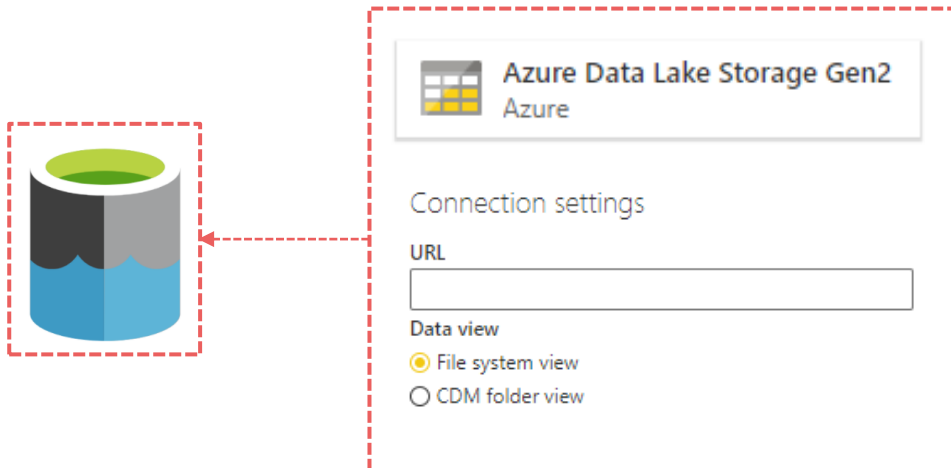
- Creating analyses including reports and dashboards

Why are we connecting Power BI to Synapse?

Power BI already has a native Azure Data Lake Gen2 connector...why would we want to connect to Synapse Analytics and use the Serverless SQL engine?

Azure Storage

Power BI



- “Out of the box” connector readily available.
- Power BI can connect to any level in the Azure Data Lake Gen2 folder hierarchy.
- Power BI can recursively load all file data located within the folder hierarchy.
- Can perform better than Serverless for small source datasets < 1GB.

However...

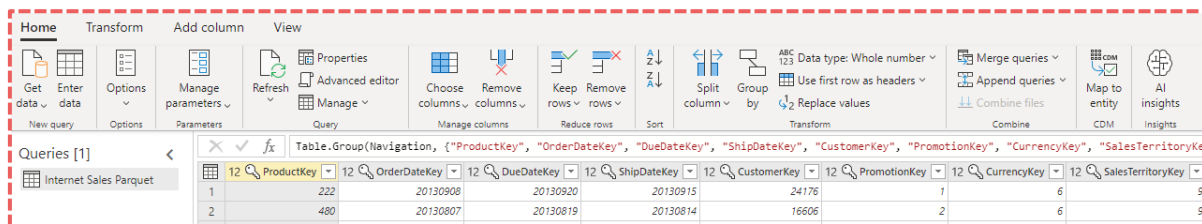
- If the size of the data grows or required transformations affect performance...Serverless SQL can take the heavy-lifting away from Power BI. The **best gains will be aggregation transformations** in which the data is being reduced in size from the source into Power BI.

Power BI Dataflows

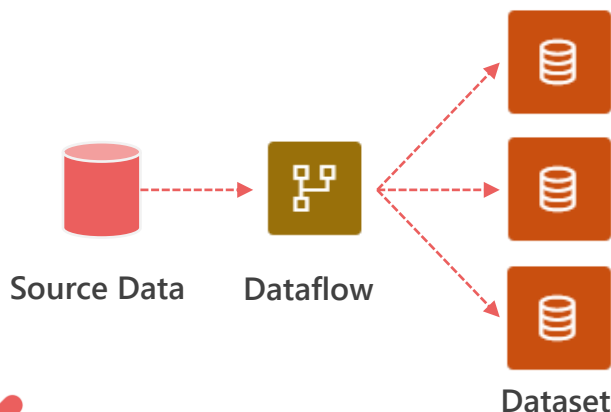


A **Power Query** based feature in the **Power BI Service** which enables:

- Connecting to a variety of data sources including SQL Databases and Data Lake Storage
- Cleansing and **Transforming** the data to suit requirements
- Mapping to common business entities using the **Common Data Model**
- Creating a **centralised repository** of data ready for using in Power BI Datasets



Dataflows reduce the number of times a data source is queried for data and reduces duplication of transformation logic.



In this example, a single Dataflow connects to and loads data from a data source, reducing the number of times the data source is queried. The Datasets then use the single Dataflow.

Azure Synapse Analytics Serverless SQL Pools



Azure Synapse Analytics Serverless SQL is a cloud analytics platform to **read and write data** in Azure Storage and Azure Data Lake Gen2.

Serverless SQL is built around the ability to **query file data "in place"** using T-SQL without copying data to internal storage.

```
SELECT * FROM
OPENROWSET
(
    BULK 'conformed/factsalesorderheader/**/*.*',
    DATA_SOURCE = 'ExternalDataSourceDataWarehouse',
    FORMAT = 'parquet'
) as fctsl
WHERE fctsl.filepath(1) = 2020
    AND fctsl.filepath(2) = 7
    AND fctsl.filepath(3) = 6
```

In the example above, T-SQL is used to SELECT all the data from every file that exists in the specified folder structure. The **filepath()** function is used to specify which folders to read data from, therefore eliminating any folders not required.

Serverless SQL includes the ability to **write the results of a SELECT query** to external data lake storage for later retrieval or use in a data solution..

Serverless SQL exposes a standard SQL endpoint which allows **SQL Client tools**, **Business Intelligence software** and many other tools to connect and pass-through SQL commands and browse supported objects.

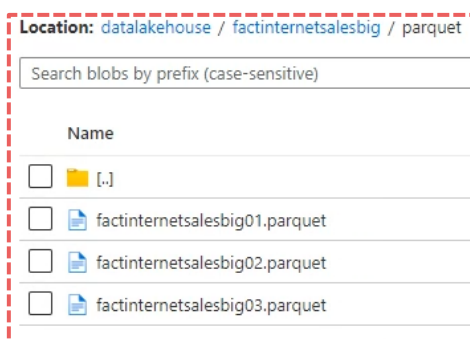
Serverless SQL Pools and Dataflows together

Power BI can connect to the Serverless SQL endpoint just like any SQL database. Connections can be made from Power BI Desktop and Power BI Service.

Dedicated SQL endpoint : synapsedemodh.sql.azure.synapse.net
 Serverless SQL endpoint : synapsedemodh-ondemand.sql.azure.synapse.net
 Development endpoint : https://synapsedemodh.dev.azure.synapse.net

SQL endpoint visible via the Azure Portal

Source data in Storage / Data Lake



Create table to read data

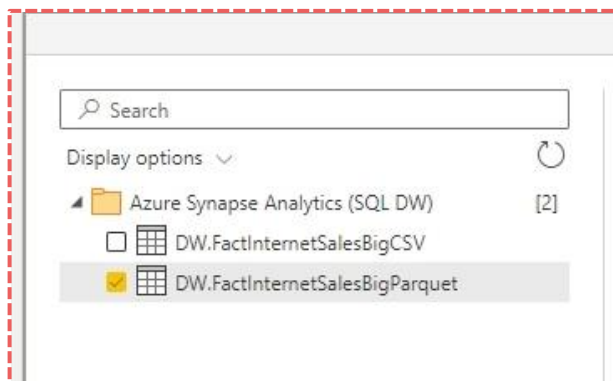
```
CREATE EXTERNAL TABLE DW.FactInternetSalesBigParquet (
  ProductKey INT,
  OrderDateKey INT,
  DueDateKey INT,
  ShipDateKey INT,
  CustomerKey INT,
  PromotionKey INT,
  CurrencyKey INT,
  SalesTerritoryKey INT,
  ....(all other columns)
)
WITH (
  LOCATION = '/factinternetsalesbig/parquet',
  DATA_SOURCE = ExternalDataSourceDataLake,
  FILE_FORMAT = SynapseParquetFormat
);
```

If you have **CSV, Parquet** or **JSON** files in Storage, SQL Serverless can connect and perform data processing.

You can create a Power BI Dataflow and use **Serverless SQL** as a data source.

SQL is pushed down to SQL Serverless due to Power Query's **Query Folding** feature.

Connect to table in Power BI Dataflow



Azure Synapse Analytics

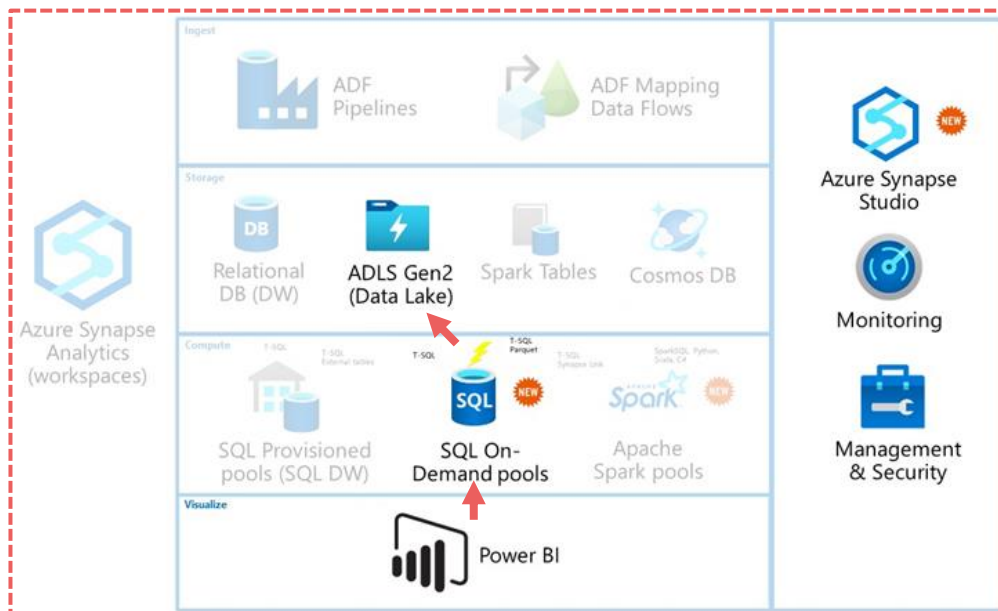


In this scenario Serverless SQL is being used as an intermediate service by Power BI Dataflows for data processing and transformation.

When an Azure Synapse Analytics resource is created, a **Serverless SQL Pool** is created automatically and is available immediately for use.

Connect to ✔ Built-in

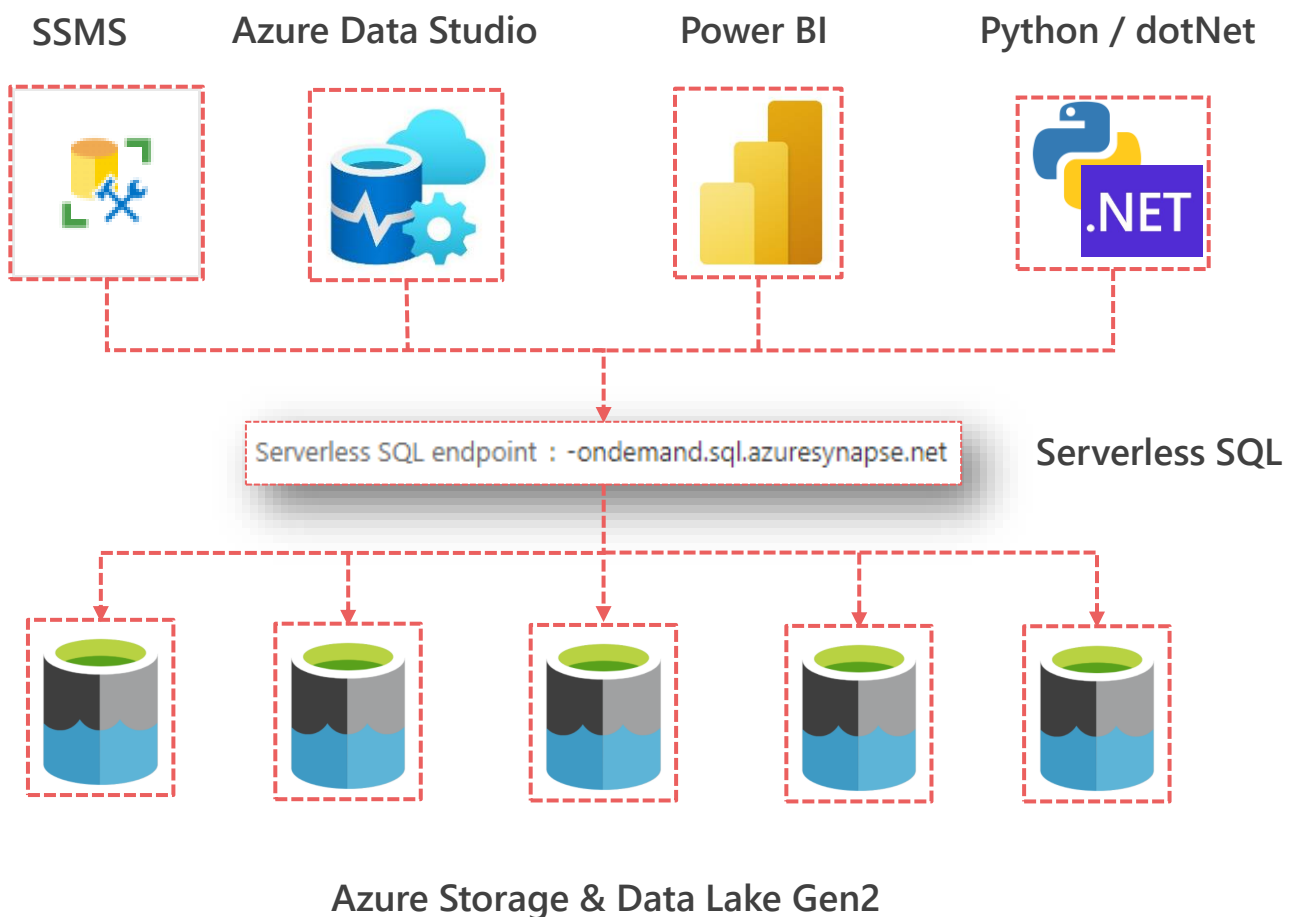
“Built-in” Serverless SQL pool available as soon as Azure Synapse Analytics resource is created.



- We **do not need** to provision a Dedicated SQL Pool
- We **do not need** to provision Spark pools
- We **do not need** to integrate Power BI workspaces
- We **do not need** to create pipelines

Connecting to the Serverless SQL endpoint

As Serverless SQL exposes a **SQL endpoint**, there are many options in terms of connecting from various software.



In the context of using with Power BI, a Data Analyst/BI Developer could connect to Serverless SQL via a SQL client tool to perform ad-hoc analysis before creating a Power BI artifact.

Power BI Query Folding

When transformations are applied to the dataset and can be “folded”, the logic to perform the transformations is passed to the data source. In this scenario, **Serverless SQL will receive a SQL statement.**

In this example, the **GROUP BY** transformation within the Dataflow is being used to aggregate and therefore reduce the incoming dataset size.

The SQL generated by the transformation will be sent to Serverless SQL.

Group by
Specify the column to group by and the desired output.

☐ Basic ☒ Advanced

Group by

ProductKey

OrderDateKey

DueDateKey

ShipDateKey

CustomerKey

PromotionKey

CurrencyKey

SalesTerritoryKey

Add grouping

New column name Operation Column ...

OrderQuantity Sum ProductKey

Count rows

Add aggregation

☐ Use fuzzy grouping

> Fuzzy group options

OK Cancel

Table.Group(Navigation, {"ProductKey", "OrderDateKey", "DueDateKey", "ShipDateKey", "CustomerKey", "PromotionKey", "CurrencyKey", "SalesTerritoryKey"}, {"{"Order...

	12	OrderDa...	12	DueDa...	12	ShipDa...	12	Custom...	12	Promoti...	12	Curren...	12	SalesTerrito...	12	OrderQuantity	1,2	SalesAmount
1	182	20121228	20130109	20130104	12132					1	100			7		3		104.97
2	182	20121228	20130109	20130104	16313					1	100			8		3		104.97
3	182	20121229	20130110	20130105	11241					1	100			7		3		104.97
4	182																	104.97
5	182																	104.97
6	182																	104.97
7	182																	104.97
8	182																	104.97
9	182																	104.97
10	182																	104.97
11	182																	104.97
12	182																	104.97
13	182																	104.97
14	182																	104.97
15	182																	104.97
16	182																	104.97
17	182																	104.97
18	182																	104.97
19	182																	104.97
20	182																	104.97
21	182																	104.97
22	182																	104.97
23	182																	104.97
24	182	20130112	20130124	20130119	18218					1	100			1		3		104.97
25	182	20130112	20130124	20130119	18217					1	6			9		3		104.97
26	182	20120113	20130124	20130119	11498					1	100			4		3		104.97

Data source query

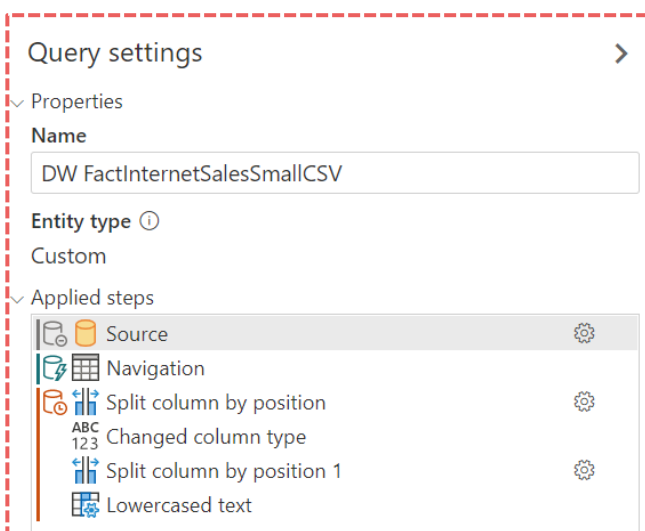
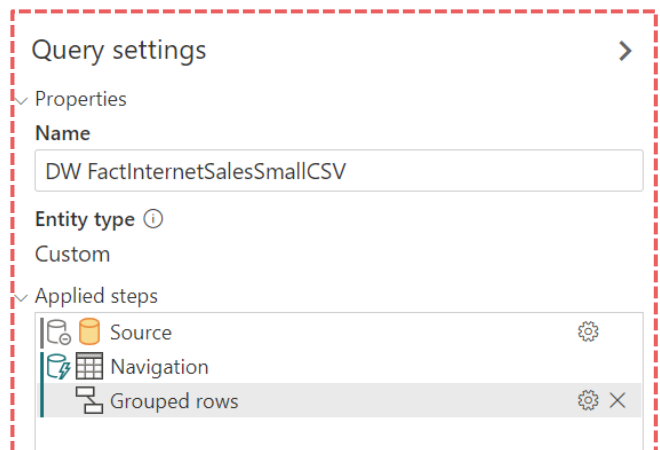
```
select [rows].[ProductKey] as [ProductKey],
[rows].[OrderDateKey] as [OrderDateKey],
[rows].[DueDateKey] as [DueDateKey],
[rows].[ShipDateKey] as [ShipDateKey],
[rows].[CustomerKey] as [CustomerKey],
[rows].[PromotionKey] as [PromotionKey],
[rows].[CurrencyKey] as [CurrencyKey],
[rows].[SalesTerritoryKey] as [SalesTerritoryKey],
sum([rows].[OrderQuantity]) as [OrderQuantity],
sum([rows].[SalesAmount]) as [SalesAmount],
sum([rows].[TaxAmt]) as [TaxAmount],
sum([rows].[Freight]) as [FreightAmount]
from
(
select [ProductKey],
[OrderDateKey],
[DueDateKey]
```

OK

Power BI Query Folding

However, not all transformations with the Dataflow can be “folded” and therefore not pushed down to the source data engine. In this instance, a transformation within the Power BI Dataflow can cause the query to no longer “fold”.

Within the Dataflow, there is **an indicator to show whether the query is being folded**. In this example the indicator is green, which shows that the query is being folded.



In this example a **split column by delimiter** transformation has been applied to the data resulting in an orange indicator from that specific step. This shows that this new transformation is **not being folded** to Serverless SQL.

It's important to note that the steps preceding this new step are still being folded to Serverless SQL. It's important to **move any foldable transformations before any non-foldable transformations**.

Supported File Formats

CSV, **Parquet** and **JSON** file formats stored in Azure Storage and Azure Data Lake Gen2 are supported by Serverless SQL. The focus for the remainder of this guide will be **CSV** and **Parquet**.

CSV

A CSV (**Comma-Separated File**) is a text format file which is readable by a wide-range of text editors and data transformation tools.

Serverless SQL will **read the entire** CSV file when a SQL query is issued regardless of any filter present. The **filepath()** and **filename()** can be used to include or exclude certain folders and CSV files to reduce data processed.

```
SELECT cust.*
FROM OPENROWSET(
    BULK 'sourcedata/*.csv',
    DATA_SOURCE = 'ExternalDataSourceDataWarehouse',
    FORMAT = 'CSV',
    PARSER_VERSION = '2.0',
    HEADER_ROW = TRUE,
    FIELDTERMINATOR = '|'
) WITH
(
    ProductKey INT 1,
    CustomerKey INT 5
) AS cust
```

Within the **SELECT** statement, a **WITH** keyword can be used to select specific columns from the CSV using the ordinal position of the column.

Ordinal Position 1

Ordinal Position 2

ProductKey	OrderDateKey	DueDateKey	ShipDateKey	CustomerKey
214	20121228	20130109	20130104	12132
214	20121228	20130109	20130104	16313
214	20121229	20130110	20130105	11241
214	20121229	20130110	20130105	12390
214	20121230	20130111	20130106	11338
214	20121230	20130111	20130106	24604
214	20121231	20130112	20130107	11061

Supported File Formats

Parquet

A Parquet file is a **columnar** (columnstore) based **compressed** file format which contains the data itself plus the schema definition and statistics about the data.

Serverless SQL will **only read the columns and data necessary** to support the SQL query issues as Parquet files support predicate push-down.

The **filepath()** and **filename()** can be used to include or exclude certain folders and Parquet files to reduce data processed.

```
SELECT *
FROM
  OPENROWSET(
    BULK '/sourcedata/*.parquet',
    DATA_SOURCE = 'ExternalDataSourceDataWarehouse',
    FORMAT = 'PARQUET'
  ) WITH
  (
    ProductKey INT,
    CustomerKey INT
  )
AS fact
```

Within the **SELECT** statement, a **WITH** keyword can be used to select specific columns from the CSV using the name of the column in the Parquet file.

Named Column



Named Column

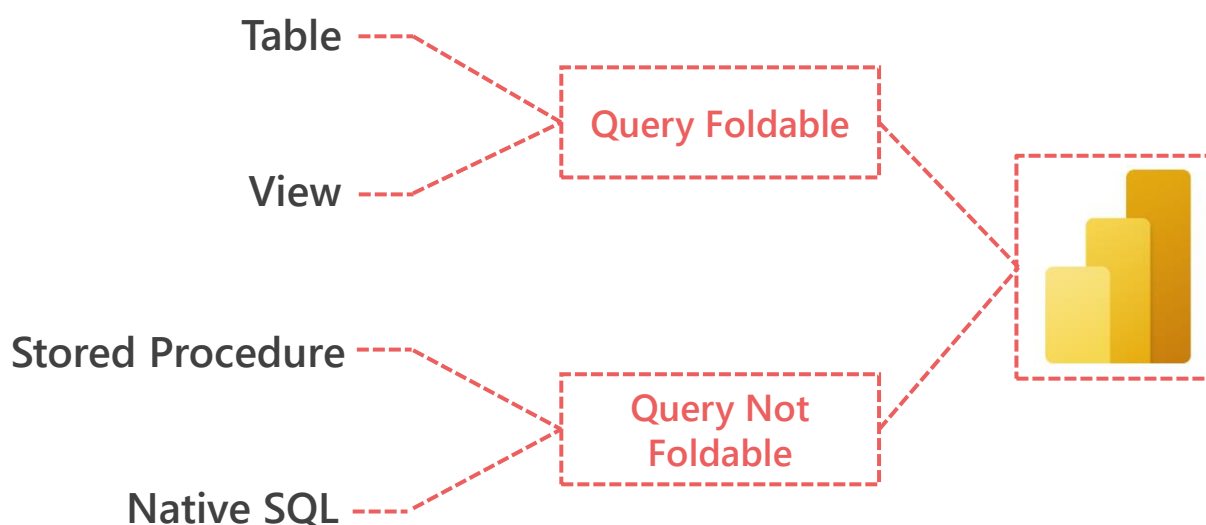


ProductKey	OrderDateKey	DueDateKey	ShipDateKey	CustomerKey
214	20121228	20130109	20130104	12132
214	20121228	20130109	20130104	16313
214	20121229	20130110	20130105	11241
214	20121229	20130110	20130105	12390
214	20121230	20130111	20130106	11338
214	20121230	20130111	20130106	24604
214	20121231	20130112	20130107	11061

Which SQL objects can Power BI connect to

The Serverless SQL engine exposes a SQL endpoint which will allow Power BI to connect and interact with the following supported objects:

Object	Notes
Table	An External Table which points to a folder location within Azure Storage / Data Lake Gen 2
View	A View which can point to specific folders locations within a Data Lake and also allow "partition pruning"
Stored Procedure	Conditional logic support but has limitations around "query folding"
Native SQL	A SQL statement can be written and used to retrieve data but has similar limitation to a stored procedure with regards to "query folding"



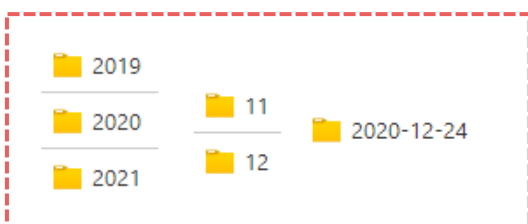
Dataflow Incremental Refresh

If there is a **DateTime** column within the data then we are able to enable the **Incremental Refresh** (Premium feature in Dataflows) and Power BI Desktop (Pro feature).

External Tables: If using an external table as the data source, the refresh will scan all folders, sub-folders, and files referenced by the external table location.

Views: If using a View as the data source, we have the option of **partition pruning (excluding folders not required)** using the **filepath()** function within Serverless SQL.

The SELECT statement will pass the date through to the filepath function within the View and **only read folders in the Data Lake that match the filter.**



Incremental refresh settings

Internet Sales CSV

Incremental refresh updates only data that's changed, to speed refresh, reduce capacity usage, and store historic data. [Learn more](#)

☒ On

Choose a DateTime field to filter by *

Choose a field ▼

Store rows from the past *

Choose a time period ▼

Refresh rows from the past *

Choose a time period ▼

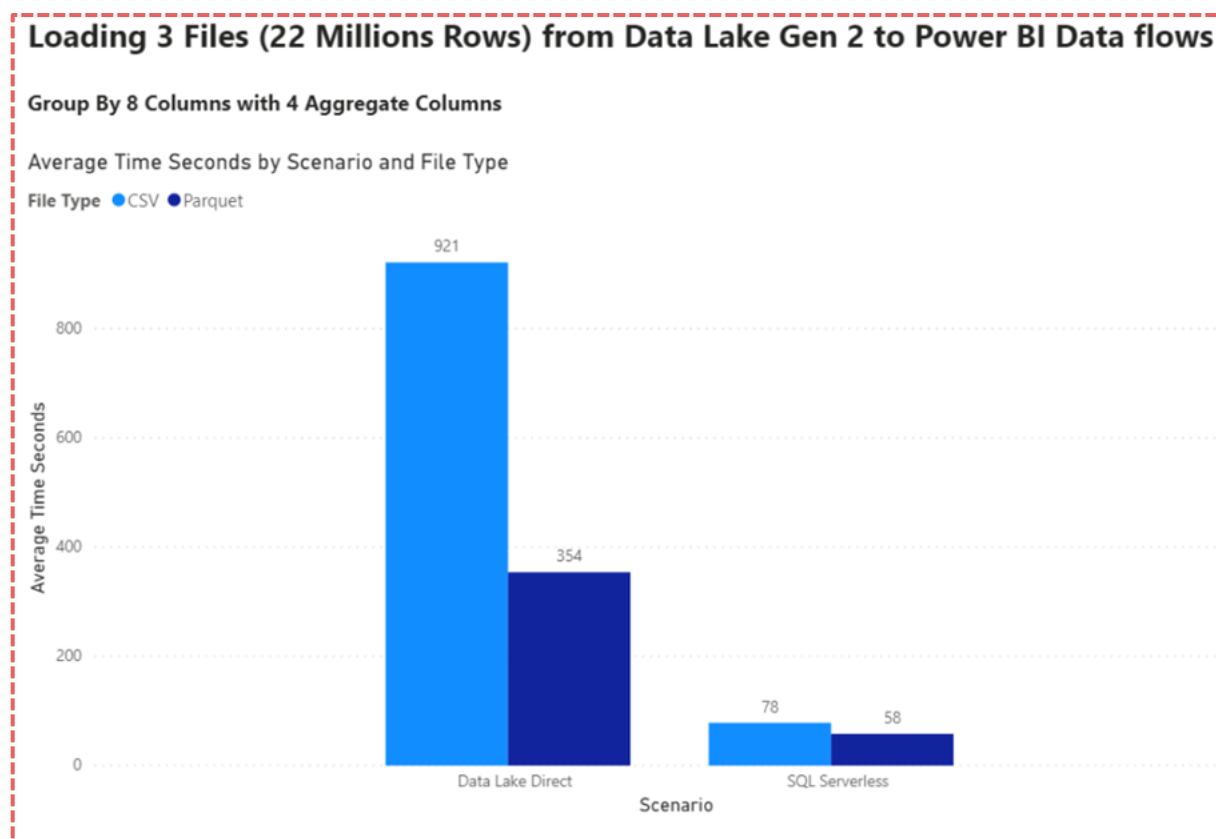
```
CREATE VIEW DW.vwFactInternetSales
AS
SELECT CAST(fct.filepath(1) AS DATETIME)
      AS FilePathDate,
ProductKey,
COUNT(*) AS TotalProductCount
FROM
OPENROWSET...
```

```
SELECT ProductKey, TotalProductCount
FROM DW.vwFactInternetSales
WHERE FilePathDate = '2020-12-24'
```

We create a folder structure in the Data Lake than can be read by the filepath() function.

Data Loading Performance Analysis

When comparing the performance of Serverless SQL and the Power BI native Data Lake connector, we can see an improvement in loading performance.



The data loading tests were carried out on **CSV** and **Parquet** data.

Total CSV file size: **4.5 GB**

Total Parquet file size: **1.5 GB**

Pushing foldable transformations down to the Serverless SQL Pool can reduce the time taken to load Power BI Dataflows.



Considerations

Cost



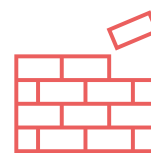
SQL Serverless does cost money!

£4.659 per 1 Terabyte (1TB) of Data Processed

When developing/testing, use a smaller file or set of smaller files

Data at rest does not necessarily translate directly into data processed

Infrastructure



Adds **another service into a data architecture** which will need managing

However, you can use Synapse Analytics SQL Serverless as a **processing engine without any data warehousing**



Serverless SQL Pools Monitoring

The SQL generated by Power BI can be seen in the **Monitor** area of Synapse Analytics Studio.

The screenshot displays the Microsoft Azure Synapse Analytics Studio interface. The left sidebar shows the navigation menu with 'Monitor' selected. The main area is divided into three panels: Integration, SQL requests, and Request content. The SQL requests panel shows a list of 101-187 items, with the selected item (Request ID 19460679) displayed in the Request content panel. The Request content panel shows the SQL query generated by Power BI.

Request ID	Request content
6042378	select ROW_NUMBER() OVER (
6039351	select * from Staging.Customers
19704878	select top 100 [rows].[ProductKey]
19477969	select [rows].[ProductKey] as [Pr
19468401	select [rows].[ProductKey] as [Pr
19460679	select [rows].[ProductKey] as [Pr
19450045	-
19450502	-
19451514	-
19451909	-
19452316	-
19452695	-
19453457	-
19454336	-

```

select [rows].[ProductKey] as [ProductKey],
[rows].[OrderDateKey] as [OrderDateKey],
[rows].[DueDateKey] as [DueDateKey],
[rows].[ShipDateKey] as [ShipDateKey],
[rows].[CustomerKey] as [CustomerKey],
[rows].[PromotionKey] as [PromotionKey],
[rows].[CurrencyKey] as [CurrencyKey],
[rows].[SalesTerritoryKey] as [SalesTerritoryKey],
sum([rows].[OrderQuantity]) as [OrderQuantity],
sum([rows].[SalesAmount]) as [SalesAmount],
sum([rows].[TaxAmt]) as [TaxAmount],
sum([rows].[Freight]) as [FreightAmount]
from
(
select [ProductKey],
[OrderDateKey],
[DueDateKey],
[ShipDateKey],
[CustomerKey],
[PromotionKey],
[CurrencyKey],
[SalesTerritoryKey],
[OrderQuantity],
[SalesAmount],
[TaxAmt],
[Freight]
from [DW].[FactInternetSales8] as [Table]
) as [rows]
group by [ProductKey].
  
```

The statistics include the query time and also the **Data Processed** amount, which is a vital statistic as this is how Serverless SQL's cost model is based on. Keep track of this metric to ensure costs are tracked.

Synapse Studio includes the capability to limit the amount of data processed per day, week, and month by using the **Cost Management** feature available in the Manage area.

Further Reading

The following links will take you through the technical implementation of Synapse Analytics and connecting with Power BI.

[Getting Started with Azure Synapse Analytics SQL Serverless](#)

[Harnessing Azure Synapse Analytics SQL Serverless in Power BI Dataflows](#)



The screenshot shows the Datahai website with the email address andy.cutler@datahai.co.uk in the top left. The navigation bar includes links for Home, About, Blog, Learn, and Contact, along with a search icon. The main heading of the blog post is 'Harnessing Azure Synapse Analytics SQL Serverless in Power BI Dataflows'. Below the heading, it indicates a reading time of 17 minutes. The text of the blog post discusses Power BI Dataflows and Azure Synapse Analytics SQL Serverless.

andycutler@datahai.co.uk

atahai | Azure Data Platform Solutions

Home About Blog Learn Contact

Harnessing Azure Synapse Analytics SQL Serverless in Power BI Dataflows

Reading Time 17 mins

Power BI Dataflows are a Power BI Service component that allow the creation of centralised, cleansed, transformed and certified tabular data ready to be consumed by report developers in datasets and models. Dataflows reduce the need to connect to and load from the same data sources repeatedly and help reduce duplication of data cleansing and transformation logic across a Power BI estate. There are many data source connectors available in Dataflows, with the ability to connect to and load data from an **Azure Data Lake Gen2** account as a native connector.

Azure Synapse Analytics SQL Serverless is a new engine available within Azure that allows data to be read from and written directly to an Azure Storage (Data lake Gen2) account using familiar SQL commands. CSV, JSON and Parquet data ingested into a Data Lake can be connected to and manipulated via the new powerful **SQL Serverless** engine.

In this blog post we'll be creating a **Dataflow** which loads data from an **Azure Data Lake Gen2** account using the **Synapse Analytics SQL Serverless** engine. We'll then perform a basic loading performance comparison when using the native Data Lake Gen2 connector versus the SQL Serverless engine. Creating a Dataflow using the native Data Lake Gen2 connector is out of the scope of this blog post.