

Azure SQL can read Azure Data Lake storage files using Synapse SQL external tables

 devblogs.microsoft.com/azure-sql/read-azure-storage-files-using-synapse-sql-external-tables

December 10, 2020

There are many scenarios where you might need to access external data placed on Azure Data Lake from your Azure SQL database. Some of your data might be permanently stored on the external storage, you might need to load external data into the database tables, etc.

Azure SQL supports the OPENROWSET function that can read CSV files directly from Azure Blob storage. This function can cover many external data access scenarios, but it has some functional limitations.

You might also leverage an interesting alternative – [serverless SQL pools in the Azure Synapse Analytics](#). In this article, I will explain how to leverage a serverless Synapse SQL pool as a bridge between Azure SQL and Azure Data Lake storage.

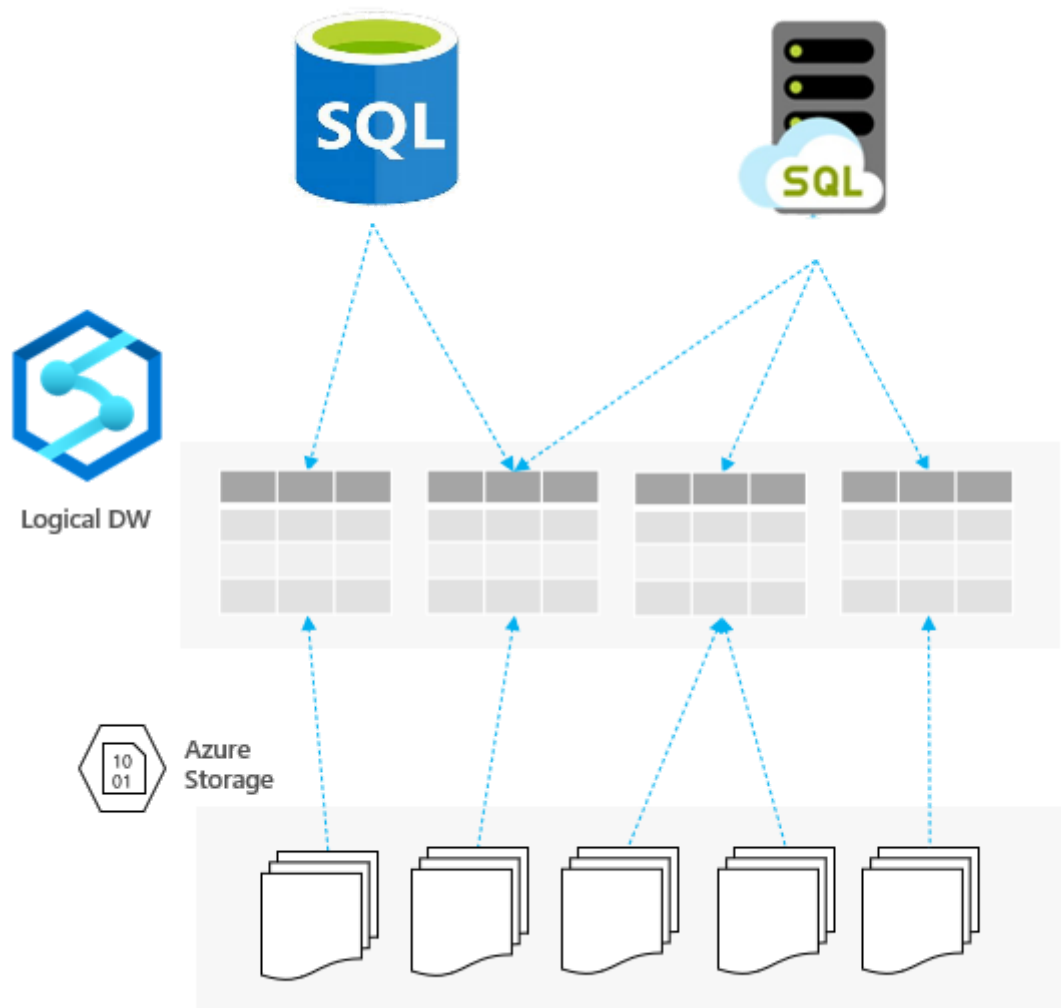
This article applies both on Azure SQL database and Azure SQL managed instance. External tables with type RDBMS can be used in any Azure SQL flavor. Just note that External tables are still in public preview.

Let us first see what is Synapse SQL pool and how can be used from Azure SQL.

What is a serverless Synapse SQL pool?

A [serverless Synapse SQL pool](#) is one of the components of the Azure Synapse Analytics workspace. It is a service that enables you to query files on the Azure storage. You can access the Azure Data Lake files using the T-SQL language that you are using in Azure SQL.

The T-SQL/TDS API that serverless Synapse SQL pools expose is a connector that links any application that can send T-SQL queries with the Azure storage. This way, your applications or databases are interacting with “tables” in so called Logical Data Warehouse, but actually they read the underlying Azure Data Lake storage files.



Serverless Synapse SQL pool exposes underlying CSV, PARQUET, and JSON files as external tables. A variety of applications that cannot directly access the files on storage can query these tables.

You can leverage Synapse SQL compute in Azure SQL by creating proxy external tables on top of remote Synapse SQL external tables. In the previous article, I have explained how to leverage linked servers to run 4-part-name queries over Azure storage, but this technique is applicable only in Azure SQL Managed Instance and SQL Server,

In this article, I will show you how to connect any Azure SQL database (single database or managed instance database) to Synapse SQL endpoint using the external tables that are available in Azure SQL. I will explain the following steps:

- How to configure Synapse workspace that will be used to access Azure storage and create the external table that can access the Azure storage.
- Configure data source in Azure SQL that references a serverless Synapse SQL pool.
- How to create a proxy external table in Azure SQL that references the files on a Data Lake storage via Synapse SQL.

In the following sections will be explained these steps.

Configure Synapse workspace

The prerequisite for this integration is the Synapse Analytics workspace. Creating Synapse Analytics workspace is extremely easy and you need just 5 minutes to create Synapse workspace if you read this article.

The easiest way to create a new workspace is to use this Deploy to Azure button.

This button will show a preconfigured form where you can send your deployment request: You will see a form where you need to enter some basic info like subscription, region, workspace name, and username/password. Probably you will need less than a minute to fill-in and submit the form.



As an alternative, you can use the Azure portal or Azure CLI.

Once you create your Synapse workspace, you will need to:

- Connect to serverless SQL endpoint using some query editor (SSMS, ADS) or using Synapse Studio.
- Create one database (I will call it SampleDB) that represents Logical Data Warehouse (LDW) on top of your ADLS files.
- Create an external table that references Azure storage files.

The first step that you need to do is to connect to your workspace using online Synapse studio, SQL Server Management Studio, or Azure Data Studio, and create a database:

```
CREATE DATABASE SampleDB;
```

Just make sure that you are using the connection string that references a serverless Synapse SQL pool (the endpoint must have **-ondemand** suffix in the domain name).

Now you need to create some external tables in Synapse SQL that reference the files in Azure Data Lake storage. Here is one simple example of Synapse SQL external table:

```
CREATE EXTERNAL TABLE csv.YellowTaxi (  
    pickup_datetime DATETIME2, dropoff_datetime DATETIME2, passenger_count INT, ...  
) WITH ( data_source= MyAdls, location = '/*/*/*.parquet', file_format =  
ParquetFormat);
```

This is very simplified example of external table. You can use this setup script to initialize external tables and views in the Synapse SQL database. As an alternative, you can read this article to understand how to create external tables to analyze COVID Azure open data set.

This is everything that you need to do in serverless Synapse SQL pool. Now you can connect your Azure SQL service with external tables in Synapse SQL.

The activities in the following sections should be done in Azure SQL.

Setup external data source in Azure SQL

Now you need to configure a data source that references the serverless SQL pool that you have configured in the previous step. You can use the following script:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'put some strong password here';
GO
CREATE DATABASE SCOPED CREDENTIAL SynapseSqlCredential
    WITH IDENTITY = '<synapse sql username>', SECRET = '<synapse sql
password>';
GO
CREATE EXTERNAL DATA SOURCE SynapseSqlDataSource
WITH (
    TYPE = RDBMS,
    LOCATION = '<synapse workspace>-ondemand.sql.azuresynapse.net',
    DATABASE_NAME = 'SampleDB',
    CREDENTIAL = SynapseSqlCredential
);
GO
```

You need to create a master key if it doesn't exist. Then create a credential with Synapse SQL user name and password that you can use to access the serverless Synapse SQL pool. Finally, create an EXTERNAL DATA SOURCE that references the database on the serverless Synapse SQL pool using the credential.

Now we are ready to create a proxy table in Azure SQL that references remote external tables in Synapse SQL logical data warehouse in order to access Azure storage files.

Create proxy external table in Azure SQL

If you have used this [setup script](#) to create the external tables in Synapse LDW, you would see the table csv.population, and the views parquet.YellowTaxi, csv.YellowTaxi, and json.Books.

In order to create proxy external table in Azure SQL that references the view named csv.YellowTaxi in serverless Synapse SQL, you could run something like a following script:

```

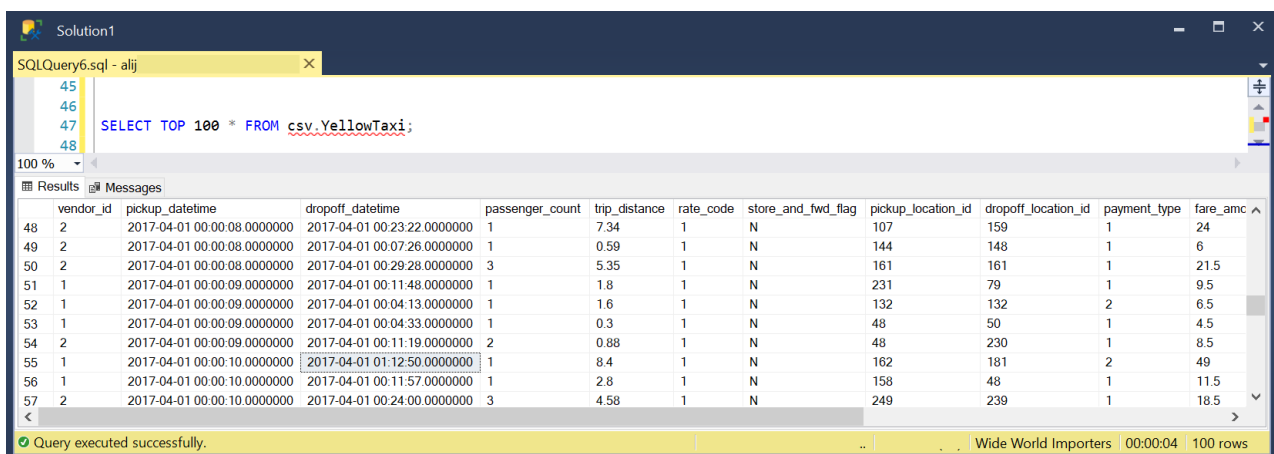
CREATE SCHEMA csv;
GO
CREATE EXTERNAL TABLE csv.YellowTaxi(
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    rate_code INT,
    store_and_fwd_flag VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_location_id INT,
    dropoff_location_id INT,
    payment_type INT,
    fare_amount FLOAT,
    extra FLOAT,
    mta_tax FLOAT,
    tip_amount FLOAT,
    tolls_amount FLOAT,
    improvement_surcharge FLOAT,
    total_amount FLOAT
)
WITH ( DATA_SOURCE = SynapseSqlDataSource );

```

The proxy external table should have the same schema and name as the remote external table or view. In addition, it needs to reference the data source that holds connection info to the remote Synapse SQL pool. This external should also match the schema of a remote table or view.

Query Azure storage files

When you prepare your proxy table, you can simply query your remote external table and the underlying Azure storage files from any tool connected to your Azure SQL database:



The screenshot shows a SQL query window with the following query: `SELECT TOP 100 * FROM csv.YellowTaxi;`. The results pane displays a table with 12 columns: vendor_id, pickup_datetime, dropoff_datetime, passenger_count, trip_distance, rate_code, store_and_fwd_flag, pickup_location_id, dropoff_location_id, payment_type, fare_amount, and extra. The data is sorted by fare_amount in descending order. The status bar at the bottom indicates 'Query executed successfully.' and 'Wide World Importers | 00:00:04 | 100 rows'.

vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	rate_code	store_and_fwd_flag	pickup_location_id	dropoff_location_id	payment_type	fare_amount	extra
48	2	2017-04-01 00:00:08.0000000	2017-04-01 00:23:22.0000000	1	7.34	1	N	107	159	1	24
49	2	2017-04-01 00:00:08.0000000	2017-04-01 00:07:26.0000000	1	0.59	1	N	144	148	1	6
50	2	2017-04-01 00:00:08.0000000	2017-04-01 00:29:28.0000000	3	5.35	1	N	161	161	1	21.5
51	1	2017-04-01 00:00:09.0000000	2017-04-01 00:11:48.0000000	1	1.8	1	N	231	79	1	9.5
52	1	2017-04-01 00:00:09.0000000	2017-04-01 00:04:13.0000000	1	1.6	1	N	132	132	2	6.5
53	1	2017-04-01 00:00:09.0000000	2017-04-01 00:04:33.0000000	1	0.3	1	N	48	50	1	4.5
54	2	2017-04-01 00:00:09.0000000	2017-04-01 00:11:19.0000000	2	0.88	1	N	48	230	1	8.5
55	1	2017-04-01 00:00:10.0000000	2017-04-01 01:12:50.0000000	1	8.4	1	N	162	181	2	49
56	1	2017-04-01 00:00:10.0000000	2017-04-01 00:11:57.0000000	1	2.8	1	N	158	48	1	11.5
57	2	2017-04-01 00:00:10.0000000	2017-04-01 00:24:00.0000000	3	4.58	1	N	249	239	1	18.5

Azure SQL will use this external table to access the matching table in the serverless SQL pool and read the content of the Azure Data Lake files. However, SSMS or any other client applications will not know that the data comes from some Azure Data Lake storage.

Conclusion

With serverless Synapse SQL pools, you can enable your Azure SQL to read the files from the Azure Data Lake storage. This way you can implement scenarios similar to the Polybase use cases.

This method can be used both on the Azure SQL database and Azure SQL managed instance, unlike a similar technique with [linked servers](#) that is available only on Azure SQL managed instance. In both cases, you can expect similar performance because computation is delegated to the remote Synapse SQL pool, and Azure SQL will just accept rows and join them with the local tables if needed.

Just note that the external tables in Azure SQL are still in public preview, and linked servers are generally available. Therefore, you might use linked servers if you are implementing the solution on a managed instance.

If you need native Polybase support in Azure SQL without delegation to Synapse SQL, vote for this feature request on the [Azure feedback site](#).

Even with the native Polybase support in Azure SQL that might come in the future, a proxy connection to your Azure storage via Synapse SQL might still provide a lot of benefits. Synapse endpoint will do heavy computation on a large amount of data that will not affect your Azure SQL resources. Therefore, you don't need to scale-up your Azure SQL in order to assure that you will have enough resources to load and process a large amount of data. This technique will still enable you to leverage the full power of elastic analytics without impacting the resources of your Azure SQL database.

Synapse SQL enables you to query many different formats and extend the possibilities that Polybase technology provides. You can learn more about the rich query capabilities of Synapse that you can leverage in your Azure SQL databases on the Synapse documentation site.

[Learn more about Synapse SQL query capabilities](#)

[Jovan Popovic](#)

