Querying external data in Azure Blob/Data Lake Storage

Azure SQL Managed Instance Private Preview Guide

Contents

Intro

This document describes the functionality, onboarding steps, and the limitations of querying external data in Azure Blob Storage and Azure Data Lake Storage from Azure SQL Managed Instance during the private preview of the feature. The document is conceived as sufficient for successful onboarding and usage, and the references to the online articles are given throughout the document only for the wider context, or as further readings.

Supplemental Terms of Use for Microsoft Azure Previews apply to the preview of this feature.

The document as a whole or its parts are not intended for publishing or sharing outside of your organization.

About the feature

This feature enables your Azure SQL managed instance to execute T-SQL queries that read data from files in Parquet and CSV format, stored in Azure Data Lake Storage v2 or Azure Blob Storage, and to combine it in queries with locally stored relational data.

In terms of the syntax, the feature resembles SQL Server's PolyBase capabilities, currently limited to read-only access to files. Additionally, it supports OPENROWSET syntax, for quick exploratory querying without creating external tables in the database. As an extra capability, multiple files and entire folders can be read by using wildcards, as long as all files accessed have the same structure.

What to expect

The purpose of the private preview is to validate that functionality meets your needs. The feature has been thoroughly tested using generic tests and data sets to make sure the quality satisfies requirements for exposing it to the customers. Still, nothing can replace real-world cases and during validation you can still run across errors or query timeouts. We would appreciate your feedback in case you notice anything unexpected.

During the preview, new capabilities will be added gradually. We will inform you as soon as they are available and share the details and instructions via email and/or updated version of this document.

Capabilities

There are two different ways of querying external files using this feature, optimized for different scenarios:

- OPENROWSET syntax – optimized for ad-hoc querying of files. Typically used to quickly explore the content and the structure of a new set of files.
- External tables – optimized for repetitive querying of files using identical syntax as if data were stored locally in the database. Requires few more preparation steps compared to the first option, but it allows more control over data access and it's generally more performant since it supports full statistics. Typically used for analytical workloads and reporting.

Both ways will be described in detail in the following sections.

OPENROWSET

The OPENROWSET approach is optimized for quick exploratory querying of files, without hassle of creating almost any objects, simply to get an insight into the content.

When accessing public files, like some of the data sets in the Azure Open Datasets catalog, all you need to start using OPENROWSET is to know file location and format, like in this example with querying a single parquet file stored in the COVID-19 data lake:

```
SELECT * FROM
        OPENROWSET(
        BULK 'https://pandemicdatalake.blob.core.windows.net/public/curated/covid-
19/ecdc_cases/latest/ecdc_cases.parquet',
        FORMAT='PARQUET'
    ) AS [CovidCaseExplorer]
```

For querying files located in private repositories, only SQL authentication is currently supported, there is no support for OPENROWSET with AAD authentication yet. Also, there must be a Shared Access Signature (SAS) key that you can use to access the file location. Make sure that you are using a valid SAS key in terms of its validity period and read access rights included. To avoid potential issues with validity period due to specific time zone set on your instance, make sure to put validity start date in the past. Step-by-step instructions for creating SAS token in form of the PowerShell script can be found in the section 1 of the tutorial originally written for SQL Server. Also, make sure to remove "?" character at the beginning of the SAS key.

There are again two ways to provide SAS key:

Using instance-scoped credential

With corresponding instance-scoped credential created, OPENROWSET command can access files in a private location without explicitly referencing SAS key:

```
--Step1: Create instance-scoped credential with name corresponding to the URL of data storage:
CREATE CREDENTIAL [https://sqlondemandstorage.blob.core.windows.net/parquet]
WITH IDENTITY='SHARED ACCESS SIGNATURE',
SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=rl&st=2019-10-14T12%3A10%3A25Z&se=2061-12-
31T12%3A10%3A00Z&sig=KlSU2ullCscyTS0An0nozEpo4tO5JAgGBvw%2FJX2lguw%3D'
GO

--Step2: Run OPENROWSET referencing file and specifying file format:
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=2016/month=10/part-01649-
tid-3416720079774751848-41f947ae-75dc-402d-bbc8-bffb3e250a02-3491.c000.snappy.parquet',
    FORMAT = 'parquet') AS filerows
```

The name of the credential must match path to the storage account and container in the following format: https://*<storage_account>*.blob.core.windows.net/*<storage_path>*. In the example above, the path used was: https://*sqlondemandstorage*.blob.core.windows.net/*parquet*.

Using external data source

Instead of specifying file path directly within OPENROWSET and using instance-level certificate, you can encapsulate part of the file path or entire file path and authentication parameters within External Data Source (EDS) object and then reference it in the DATA_SOURCE parameter of OPENROWSET. In this case DATA_SOURCE parameter will be automatically prepended to the BULK parameter to form full path to the file(s).

External Data Source is intended for easier management of file locations across multiple queries and for more powerful authentication options that will be added gradually in the future. At this moment it supports SQL authentication and SAS key only, just like instance-scoped credential. Authentication parameters are stored within database-scoped credential.

The following script creates a credential that is used to access files on storage using SAS token. The script will create a sample external data source that uses this SAS token to access storage:

```
-- Step0 (optional): Create master key if it doesn't exist in the database:
-- CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<Very Strong Password>'
GO

--Step1: Create database-scoped credential (requires database master key to exist):
CREATE DATABASE SCOPED CREDENTIAL [DemoDBCredential]
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
     SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=rl&st=2019-10-14T12%3A10%3A25Z&se=2061-12-
31T12%3A10%3A00Z&sig=KlSU2ullCscyTS0An0nozEpo4tO5JAgGBvw%2FJX2lguw%3D';
GO

--Step2: Create external data source pointing to the root of the catalog, and referencing database-
scoped credential:
CREATE EXTERNAL DATA SOURCE DemoDataSource
WITH (    LOCATION   = 'https://sqlondemandstorage.blob.core.windows.net/parquet',
          CREDENTIAL = [DemoDBCredential]
)
GO

--Step3: Run query specifying data source and the file name:
SELECT TOP 10  *
FROM OPENROWSET(
    BULK 'taxi/year=2016/month=10/part-01649-tid-3416720079774751848-41f947ae-75dc-402d-bbc8-
bffb3e250a02-3491.c000.snappy.parquet',
        DATA_SOURCE = 'DemoDataSource',
    FORMAT = 'parquet') AS filerows
```

The following OPENROWSET capabilities are available both with instance-scoped credential and data source, but sample scripts will use the former one:

Querying multiple files and folders

While in the previous examples OPENROWSET command queried a single file, it can also query multiple files or folders by using wildcards:

```
--Query all files with .parquet extension in folders matching name pattern:
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=*/month=*/*.parquet',
    FORMAT = 'parquet') AS filerows
```

Important: When querying multiple files or folders, all files accessed with the single OPENROWSET must have the same structure, i.e. number of columns and their data types.

Folders currently cannot be traversed recursively.

Filename and Filepath functions

If you are querying multiple files or folders, you can leverage Filepath and Filename functions to get the part of the path or full path and name of the file that the row originates from:

```
--Query all files and project file path and file name information for each row:
SELECT TOP 10 filerows.filepath(1) as [Year_Folder], filerows.filepath(2) as [Month_Folder],
filerows.filename() as [File_name], filerows.filepath() as [Full_Path], *
FROM OPENROWSET(
    BULK 'https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=*/month=*/*.parquet',
    FORMAT = 'parquet') AS filerows

--List all paths:
SELECT DISTINCT filerows.filepath(1) as [Year_Folder], filerows.filepath(2) as [Month_Folder]
FROM OPENROWSET(
    BULK 'https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=*/month=*/*.parquet',
    FORMAT = 'parquet') AS filerows
```

Schema inference

The schema inference works with file in the *Parquet* format and helps you quickly write queries and explore data without knowing file schemas, as seen in previous sample scripts. The cost of this convenience is that inferred data types may be larger than the actual data types, affecting the performance of queries. This happens when there isn't enough information in the source files to make sure the appropriate data type is used. For example, Parquet files don't contain metadata about maximum character column length, so instance infers it as varchar(8000).

You can use *sp_describe_first_results_set* to check the resulting data types of your query:

```
EXEC sp_describe_first_result_set N'
        SELECT
        vendor_id, pickup_datetime, passenger_count
        FROM
                OPENROWSET(
        BULK ''https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/*/*/*'',
                FORMAT=''PARQUET''
        ) AS nyc';
```

After you know the inferred data types for the query, you can specify appropriate data types using WITH clause, to improve performance:

```
SELECT TOP 100
    vendor_id, pickup_datetime, passenger_count
FROM
        OPENROWSET(
                BULK 'https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/*/*/*',
                FORMAT='PARQUET'
    )
        WITH (
                vendor_id varchar(4), -- we used length of 4 instead of the inferred 8000
```

```
                pickup_datetime datetime2,
                passenger_count int
        ) AS nyc;
```

For CSV files though, the schema cannot be automatically determined, and you need to explicitly specify columns using WITH clause:

```
--Step1: Create instance-scoped credential with name corresponding to the URL of data storage:
CREATE CREDENTIAL [https://sqlondemandstorage.blob.core.windows.net/csv]
WITH IDENTITY='SHARED ACCESS SIGNATURE',
SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=rl&st=2019-10-14T12%3A10%3A25Z&se=2061-12-
31T12%3A10%3A00Z&sig=KlSU2ullCscyTS0An0nozEpo4tO5JAgGBvw%2FJX2lguw%3D'
GOa

--Step2: Run OPENROWSET specifying columns:
SELECT TOP 10  *
FROM OPENROWSET(
BULK 'https://sqlondemandstorage.blob.core.windows.net/csv/population/population.csv',
        FORMAT = 'CSV')
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS filerows
```

Creating view on top of OPENROWSET

You can create and use views to wrap OPENROWSET for easy reusing of underlying query:

```
CREATE VIEW TaxiRides AS
SELECT *
FROM OPENROWSET(
    BULK 'https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=*/month=*/*.parquet',
    FORMAT = 'parquet') AS filerows
```

Views also enable reporting and analytic tools like PowerBI to consume results of OPENROWSET.

External tables

External tables encapsulate access to the files making the querying experience almost identical to querying local relational data stored in user tables. Creation of an external table requires external data source and external file format:

```
-- Step0 (optional): Create master key if it doesn't exist in the database:
-- CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<Very Strong Password>'
GO

--Step1: Create database-scoped credential (requires database master key to exist):
CREATE DATABASE SCOPED CREDENTIAL [DemoDBCredential]
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
     SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=rl&st=2019-10-14T12%3A10%3A25Z&se=2061-12-
31T12%3A10%3A00Z&sig=KlSU2ullCscyTS0An0nozEpo4tO5JAgGBvw%2FJX2lguw%3D';
GO

--Step2: Create external data source pointing to the root of the catalog, and referencing database-
scoped credential:
```

```
CREATE EXTERNAL DATA SOURCE DemoDataSource
WITH (    LOCATION   = 'https://sqlondemandstorage.blob.core.windows.net/parquet',
          CREDENTIAL = [DemoDBCredential]
)
GO

--Step3: Create external file format
CREATE EXTERNAL FILE FORMAT DemoFileFormat
WITH (
    FORMAT_TYPE=PARQUET
)
GO

--Step4: Create external table:
CREATE EXTERNAL TABLE tbl_TaxiRides(
     vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
     pickup_datetime DATETIME2,
     dropoff_datetime DATETIME2,
     passenger_count INT,
     trip_distance FLOAT,
     fare_amount FLOAT,
     extra FLOAT,
     mta_tax FLOAT,
     tip_amount FLOAT,
     tolls_amount FLOAT,
     improvement_surcharge FLOAT,
     total_amount FLOAT
)
WITH (
    LOCATION = 'taxi/year=*/month=*/*.parquet',
    DATA_SOURCE = DemoDataSource,
    FILE_FORMAT = DemoFileFormat
);
GO

--Step5: Query external table
SELECT TOP 10 *
FROM tbl_TaxiRides
```

Just like OPENROWSET, external tables allow querying multiple files and folders by using wildcards. However, schema inference and filepath/filename functions are not supported.

Performance

There is no hard limit in terms of number of files or amount of data that can be queried, but performance will of course depend on the amount of data, data format, and complexity of queries and joins. You are strongly encouraged to try it out with real-world queries and on the amount of data you currently have or expect to have in next several years. Please share with us any good or bad surprises you experience with performance of queries, so we can act accordingly.

Statistics

Collecting statistics on your data is one of the most important things you can do for query optimization – the more instance knows about your data, the faster it can execute queries. Automatic creation of statistics is currently not supported, but you can and should create statistics manually.

OPENROWSET stats

Single-column statistics for OPENROWSET path can be created using *sp_create_openrowset_statistics* stored procedure, by passing the select query with a single column as a parameter:

```
EXEC sys.sp_create_openrowset_statistics N'
SELECT pickup_datetime
FROM OPENROWSET(
    BULK ''https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=*/month=*/*.parquet'',
    FORMAT = ''parquet'') AS filerows
'
```

By default, instance uses 100% of the data provided in the dataset for creating statistics. You can optionally specify sample size as a percentage using TABLESAMPLE options.
To create single-column statistics for multiple columns, you should execute stored procedure for each of columns. You cannot create multi-column statistics for OPENROWSET path.

Important: Statistics for OPENROWSET path are currently not persisted across failovers. You should create them again after instance failover, typically caused by system-initiated maintenance events or user-initiated instance resize operations. If you notice performance drop for OPENROWSET paths that you previously created statistics for, it may indicate that failover happened and that you should recreate statistics.

To update existing statistics, you should drop them first using *sp_drop_openrowset_statistics* stored procedure, and then recreate them:

```
EXEC sys.sp_drop_openrowset_statistics N'
SELECT pickup_datetime
FROM OPENROWSET(
    BULK ''https://sqlondemandstorage.blob.core.windows.net/parquet/taxi/year=*/month=*/*.parquet'',
    FORMAT = ''parquet'') AS filerows
'
```

External table stats

Syntax for creating stats on external tables resembles the one used for ordinary user tables. To create statistics on a column, provide a name for the statistics object and the name of the column:

```
CREATE STATISTICS sVendor
ON tbl_TaxiRides (vendor_id)
WITH FULLSCAN, NORECOMPUTE
```

WITH options are mandatory, and for the sample size allowed options are *FULLSCAN* and *SAMPLE n percent*.

To create single-column statistics for multiple columns, you should execute stored procedure for each of columns. You cannot create multi-column statistics.

Unlike stats for OPENROWSET path, stats for external tables are preserved across failovers.

Limitations

Some of the functional limitations listed will be removed during the private and public preview. Your feedback is appreciated and may influence prioritization:

- When querying using wildcards, folders cannot be traversed recursively.
- Manual statistics for OPENROWSET are not persisted across failovers.
- Automatic creation of statistics is not supported yet.

There are also limitations that may stay in place after GA:

Support for additional types of data sources, like Hadoop, SQL Server, Oracle, Teradata, and MongoDB is not in the immediate plans, but may be added later or through other features. Your feedback also may influence prioritization.

Onboarding

If you are interested and ready to invest some time in trying out the functionalities and sharing your feedback and questions, send us the name of the instance where you would like to have feature available for testing, using email address azuresqlpolybasefdbk@microsoft.com.

Once we receive this information, we will complete the onboarding process within three working days and confirm by responding to your email.

Important: During the private preview phase, we don't recommend testing the feature on your production instances. Please use a dev/test instance for this purpose.

Contacts during the private preview

For any questions, inquiries, or issues pertaining to the feature or preview, please contact the engineering team directly, using the email address azuresqlpolybasefdbk@microsoft.com. Azure Customer Support does not cover the feature during the private preview.