T-SQL: Bulk Insert Azure CSV Blob into Azure SQL Database

Introduction

We already knew that we can use Bulk Insert from performing a bulk insert from a file into SQL Server table. See this good article by pituach for a good understanding of how to implementing the bulk inserts on a on-premises file i.e, you already have the file in your file system(local disk). But when coming to the cloud, **especially in Azure,** all the structure and unstructured data will be stored inside a blob container (In Azure Storage Account) as a **blob.** In this article, we are going to see how we are going to import (or) bulk insert a CSV file from a blob container into Azure SQL Database Table using a Stored Procedure.

Prerequisite

1. **Azure Subscription**- We need to have a valid Azure Subscription in order to create any Azure resources like Logic Apps, Azure SQL Database. You can create a free Azure Account if you don't have any.
2. **Azure Storage Account** - Once we have a valid Azure Subscription, we need to create an Azure storage account where we store all our blobs in a blob container
3. **Azure SQL Database** - We need to have an Azure SQL Database, where our Stored Procedure will reside.

Step-By-Step

Creating master key encryption by password

We need to create a database master key if one does not already exist, using your own password. This key is used to encrypt the credential secret in all the further step.

Example:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S0me!nfo' ;
```

Creating database scoped credential

We need to create database scoped credential first in order to record the authentication information that is required to connect to a resource outside SQL Server. It will create a database credential. A database credential is not mapped to a server login or database user. The credential is used by the database to access to the external location anytime the database is performing an operation that requires access.

Syntax:

```
CREATE DATABASE SCOPED CREDENTIAL credential_name WITH IDENTITY = 'identity_name'  [ , SECRET = 'secret' ]
```

Example:

For our case, we need to get access the storage blob using SAS token, so we are going to create a database scope credentials with the SAS token

```
CREATE DATABASE SCOPED CREDENTIAL MyCredentials WITH IDENTITY = 'SHARED ACCESS SIGNATURE' ,SECRET = 'QLYMgmSXMklt%2FI1U6DcVrQixnlU5Sgbtk1qDRakUBGs%3D' ;
```

Note: *The SAS key value might begin with a '?' (question mark). When you use the SAS key, you must remove the leading '?'. Otherwise, your efforts might be blocked.*

Creating external data source

Creating an external data source helps us to refer our Azure blob storage container, specify the Azure blob storage URI and a database scoped credential that contains your Azure storage account key.

Syntax:

```
CREATE EXTERNAL DATA SOURCE data_source_name

WITH (

TYPE = BLOB_STORAGE,

LOCATION =
'https://storage_account_name.blob.core.windows.net/container_name_'
[, CREDENTIAL = credential_name ]

)
```

Example:

```
CREATE EXTERNAL DATA SOURCE MyAzureStorage  WITH (

TYPE = BLOB_STORAGE,

LOCATION =
'https://myaccount.blob.core.windows.net/testingcontainer' ,
CREDENTIAL = MyCredentials

);
```

Bulk Insert from Azure Blob Container

This Bulk Insert command helps us to bulk insert our CSV file from Blob container to SQL Table

Syntax:

```
BULK INSERT
[ database_name . [ schema_name ] . | schema_name . ] [ table_name | view_name ]
FROM 'data_file'
[ WITH
(
[ [ , ] BATCHSIZE = batch_size ]
[ [ , ] CHECK_CONSTRAINTS ]
[ [ , ] CODEPAGE = { 'ACP' | 'OEM' | 'RAW' | 'code_page' } ]
[ [ , ] DATAFILETYPE =
{ 'char' | 'native' | 'widechar' | 'widenative' } ]
[ [ , ] DATASOURCE = 'data_source_name' ]
[ [ , ] ERRORFILE = 'file_name' ]
[ [ , ] ERRORFILE_DATA_SOURCE = 'data_source_name' ]
[ [ , ] FIRSTROW = first_row ]
[ [ , ] FIRE_TRIGGERS ]
[ [ , ] FORMATFILE_DATASOURCE = 'data_source_name' ]
[ [ , ] KEEPIDENTITY ]
[ [ , ] KEEPNULLS ]
[ [ , ] KILOBYTES_PER_BATCH = kilobytes_per_batch ]
[ [ , ] LASTROW = last_row ]
[ [ , ] MAXERRORS = max_errors ]
[ [ , ] ORDER ( { column [ ASC | DESC ] } [ ,...n ] ) ]
[ [ , ] ROWS_PER_BATCH = rows_per_batch ]
[ [ , ] ROWTERMINATOR = 'row_terminator' ]
```

```
[ [ , ] TABLOCK ]


-- input file format options
[ [ , ] FORMAT =  'CSV' ]
[ [ , ] FIELDQUOTE =  'quote_characters' ]
[ [ , ] FORMATFILE =  'format_file_path' ]
[ [ , ] FIELDTERMINATOR =  'field_terminator' ]
[ [ , ] ROWTERMINATOR =  'row_terminator' ]
)]
```

Example:

```
BULK  INSERT  Sales.Invoices
FROM  'inv-2017-12-08.csv'
WITH  (DATA_SOURCE =  'MyAzureStorage' );
```

CSV File

For the demo purpose, we will use a simple CSV file named inputblob.csv, which has 3 columns.

```
1,JAYENDRAN,24
2,Testing,25
```

Stored Procedure

Now we are going to implement all the above-mentioned step in a single **Store Procedure.**

First, we will create a table corresponding to the CSV File Schema

```
CREATE  table  userdetails (id  int , name  varchar ( max ),age  int  )
```

Now we will see the Stored Procedure code,

```
CREATE  PROCEDURE  dbo.BulkInsertBlob
(
@Delimited_FilePath  VARCHAR (300),
```

```
@SAS_Token   VARCHAR ( MAX ),
@Location   VARCHAR ( MAX )
)
AS
BEGIN

BEGIN TRY

--Create new External Data Source & Credential for the Blob, custom for the current upload
DECLARE @CrtDSSQL NVARCHAR( MAX ), @DrpDSSQL NVARCHAR( MAX ), @ExtlDS SYSNAME, @DBCred SYSNAME, @BulkInsSQL NVARCHAR( MAX ) ;

SELECT @ExtlDS =  'MyAzureBlobStorage'
SELECT @DBCred =  'MyAzureBlobStorageCredential'

SET @DrpDSSQL = N '
IF EXISTS ( SELECT 1 FROM sys.external_data_sources WHERE Name = ' '' + @ExtlDS +  '' ' )
BEGIN
DROP EXTERNAL DATA SOURCE ' + @ExtlDS +  ' ;
END;

IF EXISTS ( SELECT 1 FROM sys.database_scoped_credentials WHERE Name = ' '' + @DBCred +  '' ' )
BEGIN
DROP DATABASE SCOPED CREDENTIAL ' + @DBCred +  ';
END;
' ;
```

```
SET @CrtDSSQL = @DrpDSSQL + N '
CREATE DATABASE SCOPED CREDENTIAL ' + @DBCred + '
WITH IDENTITY = ' 'SHARED ACCESS SIGNATURE' ',
SECRET = ' '' + @SAS_Token + '' ';

CREATE EXTERNAL DATA SOURCE ' + @ExtlDS + '
WITH (
TYPE = BLOB_STORAGE,
LOCATION = ' '' + @Location + '' ' ,
CREDENTIAL = ' + @DBCred + '
);
' ;

--PRINT @CrtDSSQL
EXEC (@CrtDSSQL);


--Bulk Insert the data from CSV file into interim table
SET @BulkInsSQL = N '
BULK INSERT userdetails
FROM ' '' + @Delimited_FilePath + '' '
WITH ( DATA_SOURCE = ' '' + @ExtlDS + '' ',
Format=' 'CSV' ',
FIELDTERMINATOR = ' ',' ',
--ROWTERMINATOR = ' '\n' '
ROWTERMINATOR = ' '0x0a' '
```

```
);
' ;

--PRINT @BulkInsSQL
EXEC (@BulkInsSQL);

END TRY
BEGIN CATCH

PRINT @@ERROR
END CATCH
END ;
```

Here the Stored procedure have 3 parameters

1. **Delimited_FilePath** - The name of the CSV File that already in the blob container
2. **SAS_Token** - For accessing any blob within the container (Private container) we need a SAS token. (You can refer here on how to generate the SAS token)
3. **Location** - The URL of the Azure Storage Account along with the container

Executing the Stored Procedure

Now we will see how we are going to execute our Stored procedure

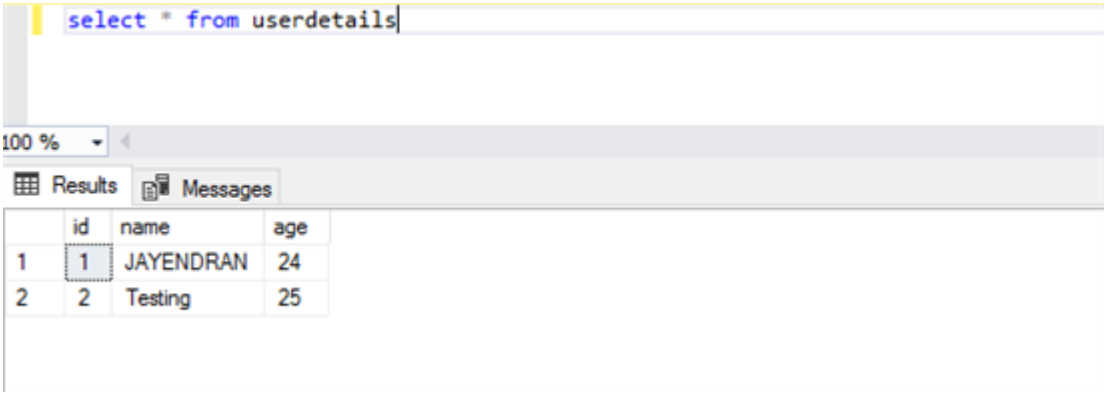
```
exec BulkInsertBlob 'inputblob.csv' , 'st=2018-10-21T14%3A32%3A16Z&se=2018-10-22T14%3A32%3A16Z&sp=rl&sv=2018-03-28&sr=b&sig=5YCuPCTVTt826ilyVsLBrKarPNg5sWUyrN7bMQ5fIhc%3D' , 'https://testingstorageaccount.blob.core.windows.net/testing_'
```

The parameters we are using are:

- Storage Account Name: https://testingstorageaccount.blob.core.windows.net/
- Container Name: testing
- SAS Token: st=2018-10-21T14%3A32%3A16Z&se=2018-10-22T14%3A32%3A16Z&sp=rl&sv=2018-03-28&sr=b&sig=5YCuPCTVTt826ilyVsLBrKarPNg5sWUyrN7bMQ5fIhc%3D
- FileName: inputblob.csv

After successful executing we will get the the message: (2 rows affected)

Let's check the userdetails table



You can find the script along with the Stored Procedure at the gallery here

Summary

Using the database scoped credentials and external data source we can easily bulk insert any types of blobs into Azure SQL Table. For simiplicty in this, artilce we take a simple CSV file with 3 fields. But we can do this more complexity as well as different field/Row delimiters as well.

See Also