



Loading Azure SQL Data Warehouse Dynamically using Azure Data Factory

By: [Ron L'Esteve \(/sqlserverauthor/329/ron-lesteve/\)](https://sqlserverauthor/329/ron-lesteve/) | Updated: 2020-04-16 | [Comments](#) | Related: [More \(/sql-server-developer-resources/\)](#) > [Azure Data Factory \(/sql-server-tip-category/245/azure-data-factory/\)](#)

Problem

In my last article, [Load Data Lake files into Azure Synapse DW Using Azure Data Factory \(/sqlservertip/6350/load-data-lake-files-into-azure-synapse-analytics-using-azure-data-factory/\)](https://sqlservertip/6350/load-data-lake-files-into-azure-synapse-analytics-using-azure-data-factory/), I discussed how to load ADLS Gen2 files into Azure SQL DW using the COPY INTO command as one option. Now that I have designed and developed a dynamic process to 'Auto Create' and load my 'etl' schema tables into SQL DW with snappy compressed parquet files, I now want to explore options for creating and loading tables into a 'curated' schema, where I can dynamically define my schemas and distribution types at run-time to create my 'curated' schema tables. How can this be achieved dynamically using Azure Data Factory?

Solution

In my previous article, [Azure Data Factory Pipeline to fully Load all SQL Server Objects to ADLS Gen2 \(/sqlservertip/6350/load-data-lake-files-into-azure-synapse-analytics-using-azure-data-factory/\)](https://sqlservertip/6350/load-data-lake-files-into-azure-synapse-analytics-using-azure-data-factory/), I introduced the concept of a pipeline parameter table to track and control all SQL server tables, server, schemas and more. Essentially, this pipeline parameter table is set up to drive the Azure Data Factory orchestration process. To solve for dynamically being able to define my distribution types along with curated schemas, I will introduce a few new columns to this pipeline parameter table: [distribution_type], [dst_schema], and [dst_name]. I will then use these new columns within my Data Factory pipeline to dynamically create and load my curated tables from my 'etl' schema.

Pre-requisites

As a pre-requisite, I would recommend getting familiar with some of my previous articles related to this topic, which eventually lead to this process:

1. [Azure Data Factory Pipeline to fully Load all SQL Server Objects to ADLS Gen2 \(/sqlservertip/6302/azure-data-factory-pipeline-to-fully-load-all-sql-server-objects-to-adls-gen2/\)](https://sqlservertip/6302/azure-data-factory-pipeline-to-fully-load-all-sql-server-objects-to-adls-gen2/)
2. [Logging Azure Data Factory Pipeline Audit Data \(/sqlservertip/6320/logging-azure-data-factory-pipeline-audit-data/\)](https://sqlservertip/6320/logging-azure-data-factory-pipeline-audit-data/)
3. [Using COPY INTO Azure Synapse Analytics from Azure Data Lake Storage gen2 \(/sqlservertip/6336/using-copy-into-command-to-load-azure-synapse-analytics-from-azure-data-lake-storage-gen2/\)](https://sqlservertip/6336/using-copy-into-command-to-load-azure-synapse-analytics-from-azure-data-lake-storage-gen2/)
4. [Load Data Lake files into Azure Synapse DW Using Azure Data Factory \(/sqlservertip/6350/load-data-lake-files-into-azure-synapse-analytics-using-azure-data-factory/\)](https://sqlservertip/6350/load-data-lake-files-into-azure-synapse-analytics-using-azure-data-factory/)

Dynamically Create and Load New Tables Using ADF Pre-Copy Script

As always, the process will begin with a look-up activity to the pipeline parameter table using a query where I can specify my flags and filters appropriately.

(<https://www.mssqltips.com/>)

Lookup

Get-Tables

ForEach

Copy-Each-Table

Activities

3 activities

General Settings User properties

Source dataset * DS_ASQMDB_PIPELINE_PARAMETER Open New Preview data

Use query ☐ Table ☒ Query ☐ Stored Procedure

Query *
SELECT [ID]
,[server_name]
,[src_type]
,[src_schema]
,[src_db]

For example, in this source lookup query, the following filters are applied. The pipeline_status = 'success' allows me to track if the files successfully made it to the lake and this is done via a SQL stored procedure. Also, it is important to note that I have quite a few columns in this pipeline parameter that help mw track steps throughout the end to end process. For the purpose of this demo, I am interested in columns [dst_schema], [dst_schema] and [distribution_type].



```
,[server_name]
,[src_type]
,[src_schema]
,[src_db]
,[src_name]
,[dst_type]
,[dst_name]
,[include_pipeline_flag]
,[partition_field]
,[process_type]
,[priority_lane]
,[pipeline_date]
,[pipeline_status]
,[load_synapse]
,[load_frequency]
,[dst_folder]
,[file_type]
,[lake_dst_folder]
,[spark_flag]
,[data_sources_id]
,[dst_schema]
,[distribution_type]
,[load_sqldw_etl_pipeline_date]
,[load_sqldw_etl_pipeline_status]
,[load_sqldw_curated_pipeline_date]
,[load_sqldw_curated_pipeline_status]
,[load_delta_pipeline_date]
,[load_delta_pipeline_status]
FROM [dbo].[pipeline_parameter]
WHERE load_synapse = 1
AND pipeline_status = 'success'
AND include_pipeline_flag = 1
AND process_type = 'full'
AND load_frequency = 'daily'
```

As an example, the dst_schema and distribution_type in the pipeline parameter table may look like this:

dst_schema	distribution_type
ref	replicate
ref	replicate
ref	replicate
ref	replicate
ref	replicate
svc	round robin
svc	round robin
svc	round robin
ref	replicate
svc	round robin

As we move on to the ForEach activity, I will ensure that the Items within the 'settings' is filled in correctly to get the output of the lookup activity.

(<https://www.mssqltips.com/>)

The screenshot shows the SSIS package designer interface. At the top, a 'Lookup' container holds a 'Get-Tables' activity. An arrow connects the output of 'Get-Tables' to the 'Copy-Each-Table' activity, which is part of a 'ForEach' loop. The 'ForEach' loop's 'Activities' list shows '3 activities'. Below the designer, the 'Settings' tab is active, showing the 'Batch count' set to 50 and the 'Items' property set to '@activity("Get-Tables").output.value'.

Upon drilling into the ForEach activities, there are three activities: Copy-Table, SUCCESS-Stored Procedure and Create_ASQldb_Log. My previous articles discussed logging so this article will focus on the details of the Copy-Table activity.

The screenshot shows the SSIS package designer interface with the 'Copy-Each-Table' activity selected. The 'Copy data' container shows 'Copy-Table1'. Two arrows connect the output of 'Copy-Table1' to the 'SUCCESS-Stored Procedure1' and 'Create_ASQldb_Log1' activities. Below the designer, the 'Source' tab is active, showing the 'Source dataset' as 'DS_ASQldw_ETL_SRC'. The 'Dataset properties' table is displayed below.

NAME	VALUE	TYPE
dst_name	@(item()).dst_name	string
src_schema	@(item()).src_schema	string
distribution_type	@(item()).distribution_type	string
load_sqldw_etl_pipelin...	@(item()).load_sqldw_etl_pipeline_date	string
load_sqldw_etl_pipelin...	@(item()).load_sqldw_etl_pipeline_status	string
load_sqldw_curated_pi...	@(item()).load_sqldw_curated_pipeline_date	string
load_sqldw_curated_pi...	@(item()).load_sqldw_curated_pipeline_status	string
dst_schema	@(item()).dst_schema	string

My source dataset is the SQLDW ETL schema.



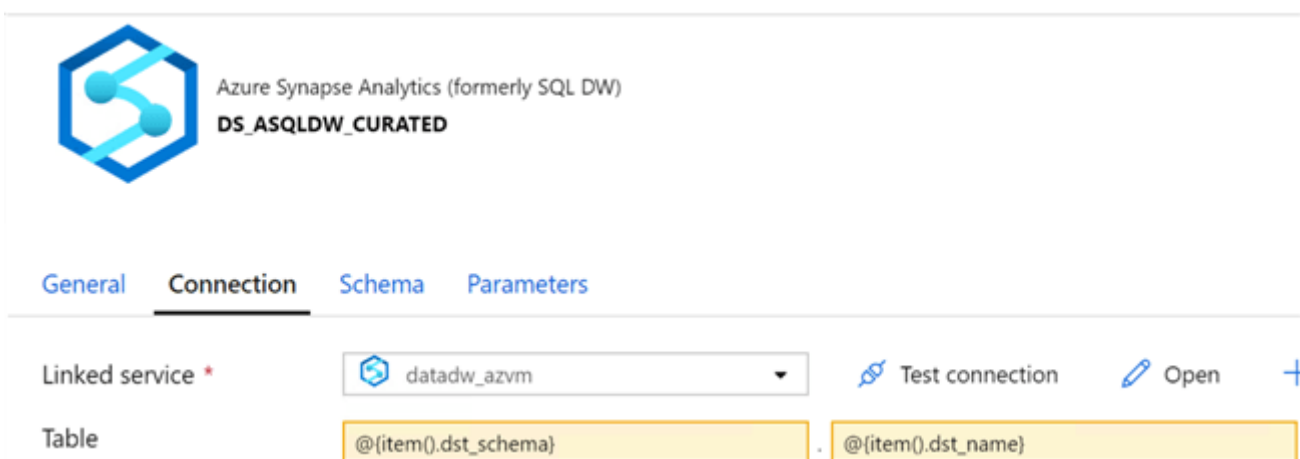
Azure Synapse Analytics (formerly SQL DW)
DS_ASQLDW_ETL_SRC

General **Connection** Schema Parameters

Linked service * datadw_azvm Test connection Open +

Table etl @{item().dst_name}

And my sink dataset is defined as my 'curated' schema where I can parametrize the destination schema and name. Note that the source dataset contains parameters that will be needed from the source schema, however the sink dataset does not contain any parameters.



Azure Synapse Analytics (formerly SQL DW)
DS_ASQLDW_CURATED

General **Connection** Schema Parameters

Linked service * datadw_azvm Test connection Open +

Table @{item().dst_schema} @{item().dst_name}

After creating my datasets, I can take a closer look at the pre-copy script. Note that I am using Bulk insert as the copy method since the data currently exists in the 'etl' schema on SQL DW and must be loaded to a curated schema.

I'll also set the table option to 'none' since I'll be creating tables using the following pre-copy script, which is basically a dynamic Create Table As Select Syntax that referenced my destination schema and name along with the distribution type. I am using a Select Top (0) because I only want to create the tables using this step and load them using the ADF Copy Activity.

```
CREATE TABLE @{item().dst_schema} @{item().dst_name}
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = @{item().distribution_type}
)
AS SELECT TOP (0) * FROM etl @{item().dst_name}
OPTION (LABEL = 'CTAS : @{item().dst_name}');
```

Copy data

Copy-Table1

General Source Sink Mapping Settings User properties

Sink dataset *

DS_ASQLDW_CURATED

Open

New

Copy method

☐ PolyBase

☐ Copy command (Preview)

☒ Bulk insert

Table option

☒ None

☐ Auto create table ⓘ

Pre-copy script

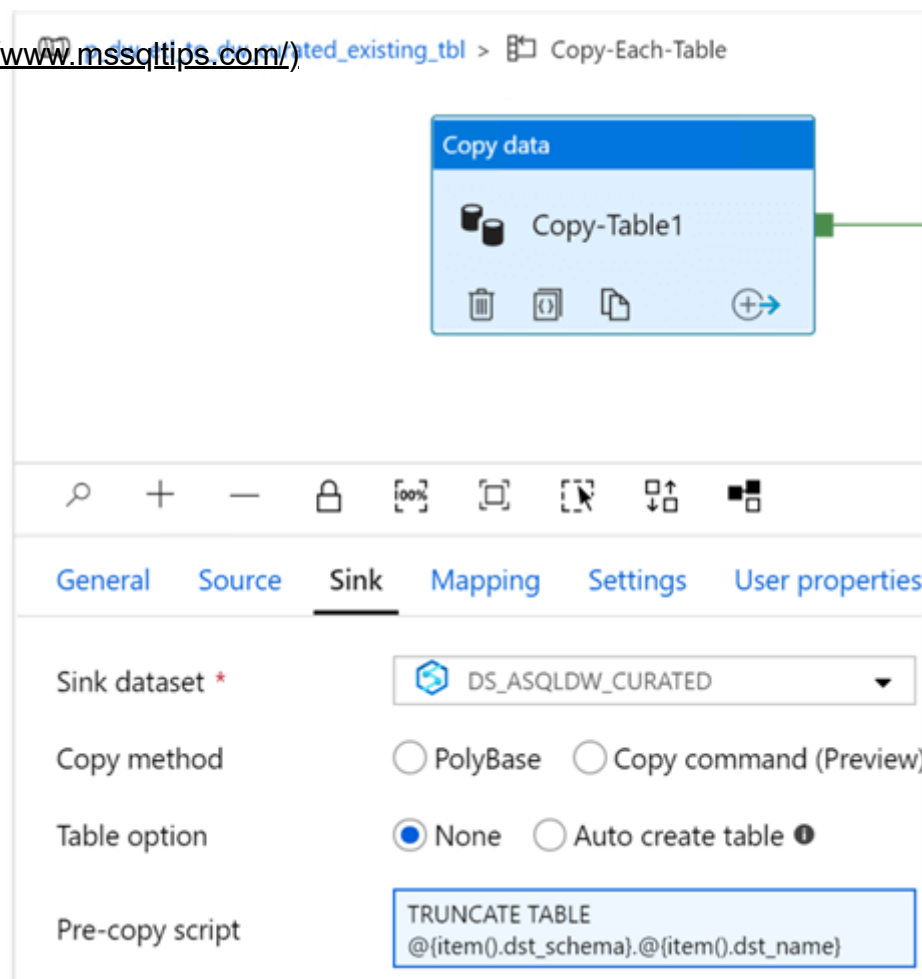
```
CREATE TABLE
@({item().dst_schema}).@({item().dst_name})
WITH ( CLUSTERED COLUMNSTORE INDEX,
DISTRIBUTION =
@({item().distribution_type}) ) AS SELECT
TOP (0) * FROM etl.@({item().dst_name})
OPTION (LABEL = 'CTAS :
@({item().dst_name})');
```

ⓘ

After running the pipeline, all the curated tables will be created in SQL DW with the appropriate destination schema, name, and distribution type.

Dynamically Truncate & Load Existing Tables Using ADF Pre-Copy Script

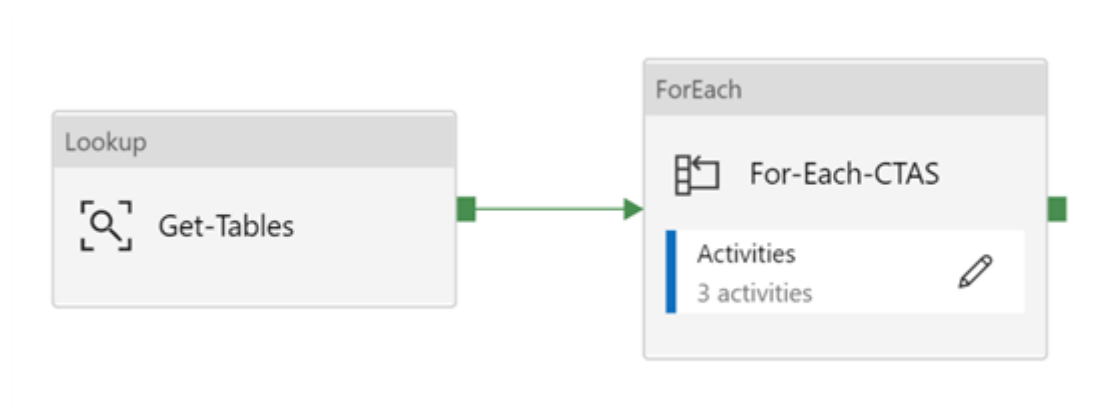
In a scenario where I need to dynamically truncate and load existing tables rather than recreate the tables, I would approach it by simply truncating the destination table. This would be the only notable change from the previous pipeline.



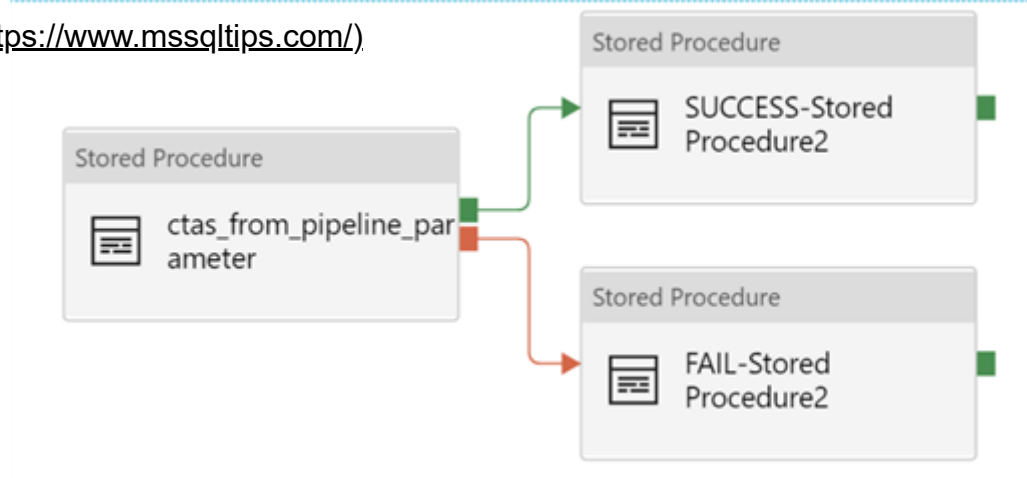
Dynamically Drop, Create and Load Tables Using SQL DW Stored Procedure

Lastly, I am interested in exploring an option that will allow me to use store procedures on the SQL DW to drop and create my curated tables.

The pipeline design will be very similar to my previous pipelines by starting with a lookup and then flowing into a ForEach activity.



Within the ForEach, I have 3 stored procedure activities: 1) CTAS from pipeline parameter, 2) SUCCESS, and 3) FAIL. The success and fail stored procedures simply log the status of the pipeline run in the pipeline parameter table. For the purpose of this article, I will focus on the CTAS stored procedure.



The selected stored procedure has been created within the SQL DW. Additionally, the destination name and schema have been defined as stored procedure parameters and are being passed to the stored procedure.

p_lake_to_synapse_new_tbl_adhoc > For-Each-CTAS

Stored procedure name * [etl].[ctas_from_pipeline_parameter] ☒ Edit ⓘ

Stored procedure parameters ⓘ

+ New |

NAME	TYPE	VALUE
name	String	@(item().dst_name)
schema	String	@(item().dst_schema)
distribution_type	String	@(item().distribution_type)

When I head over to SSMS and script the stored procedure, I can see that the script is doing a few things:

1. Declaring and setting the distribution_types along with etl, curated and schema/table names dynamically
2. Dropping the curated_stage table if it exists
3. Setting the SQL Syntax to Create the stage table as select * from etl.table with the distribution type dynamically set.
4. Dropping the actual/original curated table



```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROC [etl].[ctas_from_pipeline_parameter] @schema [varchar](255),@name [varchar](255),@distribution_tpy
BEGIN

declare @table varchar(255)
declare @table_stage varchar(255)
declare @table_etl varchar(255)
declare @sql varchar(max)

set @table = @schema + '.' + @name
set @table_stage = @table + '_stage'
set @table_etl = 'etl.' + @name

set @sql =
'if object_id ('''+ @table_stage + ''',''U'') is not null drop table ' + @table_stage + ';

CREATE TABLE ' + @table_stage + '
WITH
(
DISTRIBUTION = ' + @distribution_type + '
,CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT *
FROM ' + @table_etl + ';

if object_id ('''+ @table + ''',''U'') is not null drop table ' + @table + ';

RENAME OBJECT ' + @table_stage + ' TO ' + @name + ';

exec(@sql)

END
GO
```

In a scenario where I am interested in renaming the original curated table, rather than dropping the original curated table, I would use this script:



```
SET QUOTED_IDENTIFIER ON
GO

CREATE PROC [etl].[ctas_from_pipeline_parameter] @schema [varchar](255),@name [varchar](255),@distribution_type [varchar](255)
BEGIN

declare @table varchar(255)
declare @table_stage varchar(255)
declare @table_drop varchar(255)
declare @table_etl varchar(255)
declare @schematable_drop varchar(255)
declare @sql varchar(max)

set @table = @schema + '.' + @name
set @table_stage = @table + '_stage'
set @table_drop = @name + '_drop'
set @table_etl = 'etl.' + @name
set @schematable_drop = @table + '_drop'

set @sql =
'if object_id ('''+ @table_stage + ''',''U'') is not null drop table ' + @table_stage + ';

CREATE TABLE ' + @table_stage + '
WITH
(
DISTRIBUTION = ' + @distribution_type + '
,CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT *
FROM ' + @table_etl + ';

if object_id ('''+ @table + ''',''U'') is not null rename object ' + @table + ' TO ' + @table_drop + ';

RENAME OBJECT ' + @table_stage + ' TO ' + @name + ';

if object_id ('''+ @schematable_drop + ''',''U'') is not null drop table ' + @schematable_drop + ';

exec(@sql)

END
GO
```

Lastly, it is important to note that within SQL DW, if I am attempting to drop or rename a table that has dependencies linked to Materialized View that has been created, then the drop and rename script will fail.

In this article, I outlined steps to Dynamically Create & Load New Tables Using ADF Pre-Copy Script, Dynamically Truncate & Load Existing Tables Using ADF Pre-Copy Script, and finally Dynamically Drop, Create, & Load Tables Using SQL DW Stored Procedure.

Next Steps

- For more detail on designing distributed tables in SQL DW, see [Guidance for designing distributed tables in SQL Analytics](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-tables-distribute) (<https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-tables-distribute>)



dynamically defining distribution types, explore options for dynamically defining indexes, partitions and more in the [\(https://www.mssqltips.com/\)](https://www.mssqltips.com/) **MENU**

parameter table and dynamic SQL within Azure Data Factory.

- See my article [Design and Manage Azure SQL Data Warehouse \(/sqlservertip/4889/design-and-manage-azure-sql-data-warehouse/\)](https://sqlservertip/4889/design-and-manage-azure-sql-data-warehouse/) for more detail on working with SQL DW.
- Explore and research options for dropping and/or renaming tables that have dependencies linked to Materialized Views.

Last Updated: 2020-04-16

About the author



[\(/sqlserverauthor/329/ron-lesteve/\)](https://sqlserverauthor/329/ron-lesteve/) Ron L'Esteve is a seasoned Data Architect who holds an MBA and MSF. Ron has over 15 years of consulting experience with Microsoft Business Intelligence, data engineering, emerging cloud and big data technologies.

[View all my tips \(/sqlserverauthor/329/ron-lesteve/\)](https://sqlserverauthor/329/ron-lesteve/)

Related Resources

- [More Database Developer Tips... \(/sql-server-developer-resources/\)](https://sql-server-developer-resources/)