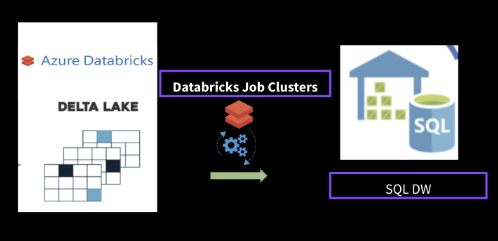
Databricks Delta to Azure Synapse Analytics (SQL DW) Synchronization (CDC) Tool

Tool to manage CDC/Synch tables from Databricks Delta tables to Azure Synapse Analytics using configuration. Designed to scale to hundreds of tables with minimal configuration and easy defaults for most options. Tool is designed to handle skipping columns and transformations on columns. Tool will be able to manage inserts, updates and deletes (CDC) on source databricks delta tables and merge those changes to Synapse Analytics target tables. Tool can also support append only to target tables in Synapse Analytics. Tool provides autoscaling to meet client SLA.



# Environment:

* Azure Databricks
* Azure Synapse Analytics
* ADLS Gen2 Storage
* Databricks REST API Client Jar
* Azure SQL Service

# Features/Benefits:

* Configuration driven from SQL Tables
* Low latency with microbatch of Structured Streaming
* Uses structured streaming for efficiency in streaming changes and schema enforcement
* Scales to any number of tables
* Uses cluster pools for startup of clusters and scaling clusters. This will allow for fast startup of cluster and fast scaleup of cluster to meet SLA.
* Transformations on columns can be configured (Best practice is to complete these transformation operations on the previous layer in Delta. Looking at other vendors/solutions in the industry, Retain the same data in source and target for data quality checks)
* Skip columns can be configured
* Primary keys can be configured to be used for Upsert on Synapse Analytics
* Multiple tables CDC can be done by one Job on One Cluster to save on resource utilization. Example: 50 CDC tables jobs on each cluster with 20 clusters on a workspace can handle 1000 table CDC.
* Logging feature
* Auto scaling cluster based on SLA. Each bundle batch with any number of tables can have a different SLA.
* Upsert in target table in Synapse from Delta
* Append on target table in Synapse
* Log table for tracking jobs creation and run
* Uses automated clusters for lower cost compared to interactive clusters
* Provides custom auto scaling of nodes using consensus algorithm
* If the target table does not exist, the tool will create the target table to match the schema being written. This applies only for upsert tables.
* Uses checkpointing for tracking offsets. On job failure it will restart the job using offsets stored in checkpoint location. Each table has its own checkpoint location
* Log table for viewing heart beats from foreachbatch
* Log table to keep track of latency on Synapse Analytics for upsert
* Log table for microbatch details from Streaming Query Listener
* Consensus based scale up and scale down of cluster, when cluster has multiple queries running. Implemented algorithm will look at the average batch interval in the last 30 minutes for all micro batches in that bundle. It will allow the streaming query with the highest batch interval to take the decision of scale up and scale down. Default is 5 minutes SLA. Scale up or Scale down is configured to trigger if consecutive 2 batches are slower than SLA or consecutive 2 batches are faster than SLA.

# Important:

* Source delta table can only have soft deletes
* Actual deletes on delta table will cause the stream to fail (<https://docs.databricks.com/delta/delta-streaming.html>)

# Questions/Suggestions:

Srinivas Nelakuditi

Email: [srinivas.nelakuditi@databricks.com](mailto:srinivas.nelakuditi@databricks.com)

# Configuration Tables

Configuration tables are being maintained in SQL DW.

CREATE TABLE dbo.synch\_delta\_sqldw\_tasks

(

source\_table\_name varchar(250) NOT NULL,

target\_table\_name varchar(250) NOT NULL,

skip\_columns varchar(2000),

key\_columns varchar(2000),

active\_flag INT NOT NULL,

bundle\_batch\_id varchar(1000) NOT NULL,

upsert\_in\_target\_flag INT NOT NULL

);

* This table will have the tables that need to stay synched.
* Skip\_columns is to allow for skipping some columns when data is synched from source to destination
* Key\_columns will provide the primary\_keys on the target table in SQL DW
* Active\_flag will indicate if synch should be enabled on this table
* Sla\_minutes\_per\_batch is used for auto scaling cluster to meet sla for table
* Bundle\_batch\_id is a unique\_id and all the tables with the same id will be grouped together with a single job on a single cluster.
* Control the behaviour of CDC as append to target table or upsert in target table using upsert\_in\_target\_flag column

CREATE TABLE dbo.synch\_delta\_sqldw\_bundle\_batch\_sla

(

sla\_minutes INT NOT NULL,

bundle\_batch\_id varchar(1000) NOT NULL,

active\_flag INT NOT NULL

);

* Table to manage required SLA per Bundle Batch
* All tables in the bundle batch will have the same SLA

CREATE TABLE dbo.synch\_delta\_sqldw\_transformation\_master

(

transformation\_name varchar(500) NOT NULL,

description varchar(1000) NOT NULL,

active\_flag INT NOT NULL

);

* This table will have all the transformations that can be applied on the stream before writing to Synapse Analytics

CREATE TABLE dbo.synch\_delta\_sqldw\_table\_transformations\_apply

(

source\_table\_name varchar(250) NOT NULL,

column\_name varchar(500) NOT NULL,

transformation\_name varchar(500) NOT NULL,

active\_flag INT NOT NULL

);

* This table will have transformations to be applied per table. Functionality of this feature is being implemented.

# Log Tables

CREATE TABLE dbo.synch\_delta\_sqldw\_jobs\_created

(

job\_id varchar(500) NOT NULL,

run\_id varchar(500) NOT NULL,

cluster\_id varchar(500) NOT NULL,

bundle\_batch\_id varchar(500) NOT NULL,

created\_at\_datetime datetime NOT NULL

);

* This log table will be is used by Master to keep track of all jobs and run creations

CREATE TABLE dbo.synch\_delta\_sqldw\_heartbeat

(

job\_id varchar(500) NOT NULL,

run\_id varchar(500) NOT NULL,

source\_table\_name varchar(500) NOT NULL,

bundle\_batch\_id varchar(500) NOT NULL,

records\_processed\_in\_batch bigint NOT NULL,

batch\_interval bigint NOT NULL,

number\_of\_nodes bigint NOT NULL,

created\_at\_datetime datetime NOT NULL

);

* This log table is used by streaming queries from each microbatch as part of the foreachbatch process for all tables in the bundle batch.

CREATE TABLE dbo.synch\_delta\_streaming\_query\_log

(

job\_id varchar(500) NOT NULL,

cluster\_id varchar(500) NOT NULL,

number\_of\_nodes bigint NOT NULL,

target\_table\_name varchar(500) NOT NULL,

records\_processed\_in\_batch bigint,

bundle\_batch\_id varchar(500) NOT NULL,

streaming\_query\_name varchar(1000) ,

streaming\_query\_id varchar(1000) ,

batch\_interval bigint,

created\_at\_datetime datetime2 NOT NULL

);

* This table is logged from streaming query listener

# Notebooks:

### Synch\_Delta\_SQLDW\_Master (scala notebook)

* Master notebook to look at configuration tables and uses Rest API to create Jobs and Job Runs. This notebook can be configured to execute at regular intervals looking for any CDC jobs to be launched or to restart jobs on failure.

### Synch\_Delta\_SQLDW\_Job\_Run (scala notebook)

* Slave notebook triggered as job run by Master to conduct CDC/Synch for all the tables in a bundle batch id.
* Functionality to provide column skipping, transformations before writing, auto scaling to meet SLA.

Synapse Analytics Merge:

You can replace merge statements, at least in part, by using CTAS. You can combine the INSERT and the UPDATE into a single statement. Any deleted records should be restricted from the SELECT statement to omit from the results.

**Note**: MSFT SME suggested a two step process of insert and update from the staging table. If the target table has high volume, you need to have a limitation on how long into history to look at 6 months or 1 year?

Target table in Synapse will need partitions for finding which records are inserts and which records are updates.

**CREATE TABLE dbo.[DimProduct\_upsert]**

**WITH**

**( DISTRIBUTION = HASH([ProductKey])**

**, CLUSTERED INDEX ([ProductKey])**

**)**

**AS**

**-- New rows and new versions of rows**

**SELECT s.[ProductKey]**

**, s.[EnglishProductName]**

**, s.[Color]**

**FROM dbo.[stg\_DimProduct] AS s**

**UNION ALL**

**-- Keep rows that are not being touched**

**SELECT p.[ProductKey]**

**, p.[EnglishProductName]**

**, p.[Color]**

**FROM dbo.[DimProduct] AS p**

**WHERE NOT EXISTS**

**( SELECT \***

**FROM [dbo].[stg\_DimProduct] s**

**WHERE s.[ProductKey] = p.[ProductKey]**

**);**

**RENAME OBJECT dbo.[DimProduct] TO [DimProduct\_old];**

**RENAME OBJECT dbo.[DimProduct\_upsert] TO [DimProduct];**

# Data Quality Checks:

* Identify what kinds of data quality checks are required and build notebooks.

# Testing:

* Integration test completed with 10 tables (done)
* Autoscaling with consensus algorithm tested to meet user provided SLA (done)
* Test soft delete synchronization on target Synapse Analytics
* Test upsert functionality (done)
* Test append only tables (done)
* Test single table on bundle batch with scaling (done)
* Test when user does not provide SLA and use the default SLA (done)

# Load testing:

* Need client environment access.

# Open questions:

1. Logging telemetry to app insights
2. Apply a max limit on cluster size while scaling up. This should be configured at bundle batch level. What should be the default max size? As nodes or of different types how do we pick the max for different node types.

# Open tasks/improvements by priority:

1. Move configuration/log tables to Azure SQL in the next iteration

# Low priority future items:

1. Functionality to deal with multiple key columns (confirm if this needs to be supported)
2. Log time taken by upsert in Synapse Analytics

# Reported/Resolved Issues:

* MSFT suggested to have a two step process of insert and update than using the CTAS approach on their documentation. (03/05 meeting)

# Deployment steps:

* Create Dev, QA, Prod and PreProd Databricks Subscriptions
* Create a Synapse Analytics environment for each of the above subscriptions
* Create Azure SQL accounts for config/log tables
* Run the config and log table creation scripts on all Azure SQL subscriptions.
* Test the configuration in Dev/QA and on success promote to PreProd and Prod in sequence
* Use secret scopes for all configuration passwords and keys
* Test for stability at product data rate on preprod environment for 48 hours before promoting to Production
* Any job config cannot be directly configured on Prod or Preprod. It has to be promoted only after success in QA to pre prod and prod
* Use CI/CD process to promote notebooks to higher environments
* If logic in the notebooks is changed please allow for complete integration and load testing
* Use Azure Data Factory to trigger the Master job at regular frequency