Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2023, 10(7):68-71

Research Article ISSN: 2394 - 658X

Microsoft SQL Server to Snowflake

Naveen Muppa

10494 Red Stone Dr Collierville, Tennessee

ABSTRACT

Microsoft's SQL Server is a database server, and as such, it primarily stores and retrieves data. It is an implementation of both the Relational Database Management System (RDBMS) and the Structured Query Language (SQL). Today, many specialized versions of SQL Server exist catering to a wide spectrum of workloads and demands. For example, there is a data center version tailored to higher levels of application support and scale, and a scaled-down version available as freeware.

Key words: One of the biggest challenges of migrating data to Snowflake is choosing from all the different options available. This blog post covers the detailed steps you need to follow to migrate data from MS SQL Server to Snowflake but first let's take a quick look at these two databases.

INTRODUCTION

Snowflake, on the other hand, is an analytics database built for the Cloud and delivered as a Data Warehouse-as-a-Service (DWaaS). Snowflake runs on AWS, the world's most popular cloud provider. Like other Databases, you can load and query any structured relational data in Snowflake tables using standard SQL data types e.g. NUMBER, BOOLEAN, VARCHAR, TIMESTAMPS, etc.

METHODS TO MOVE DATA FROM MICROSOFT SQL SERVER TO SNOWFLAKE

1. Loading a limited amount of data using the Snowflake Web User Interface
2. Bulk loading a large dataset using Snow pipe
3. Bulk loading a large amount of data using Snow SQL, a Snowflake CLI
4. Using Hevo Data– A Third-Party Data Integration Platform tool

Using the Snowflake Web User Interface is not ideal, there is a limit to the amount of data you can move using this method. Snowflake Software-as-a-Service (SaaS) offering can be hosted on Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure (Azure).

This blog post will focus on how to bulk load a large amount of data from an SQL Server on-site to a Snowflake Data Warehouse on AWS.

The Snowflake database platform on AWS is backed by EC2 instances for computing and S3 buckets for storage, so it makes sense that we will be staging our data in an AWS S3 bucket. We will also look at some of the limitations of using this method.

EXPORT DATA FROM SQL SERVER USING SQL SERVER MANAGEMENT STUDIO

- SQL Server Management Studio is a data management and administration software application that launched with SQL Server.
- We are going to use it to extract data from a SQL database and export it to CSV format. The steps to achieve this are:
- Install SQL Server Management Studio if you don't have it on your local machine.
- Launch the SQL Server Management Studio and connect to your SQL Server.

- From the Object Explorer window, select the database you want to export and right-click on the context menu in the Tasks sub-menu and choose the Export data option to export table data in CSV.
- The SQL Server Import and Export Wizard welcome window will pop up. At this point, you need to select the Data source you want to copy from the drop-down menu.
- After that, you need to select SQL Server Native Client 11.0 as the data source.
- Select an SQL Server instance from the drop-down input box.
- Under Authentication, select “Use Windows Authentication”.
- Just below that, you get a Database drop-down box, and from here you select the database from which data will be copied.
- Once you’re done filling out all the inputs, click on the Next button.
- The next window is the Choose a Destination window. Under the destination drop-down box, select the Flat File Destination for copying data from SQL Server to CSV.
- Under File name, select the CSV file that you want to write to and click on the Next button.
- In the next screen, select Copy data from one or more tables or views and click Next to proceed.
- A “Configure Flat File Destination” screen will appear, and here you are going to select the table from the Source table or view. This action will export the data to the CSV file. Click Next to continue.
- You don’t want to change anything on the Save and Run Package window so just click Next.
- The next window is the Complete Wizard window which shows a list of choices that you have selected during the exporting process. Counter check everything and if everything checks out, click the Finish button to begin exporting your SQL database to CSV.
- The final window shows you whether the exporting process was successful or not. If the exporting process finished successfully, you will see a similar output to what’s shown below.

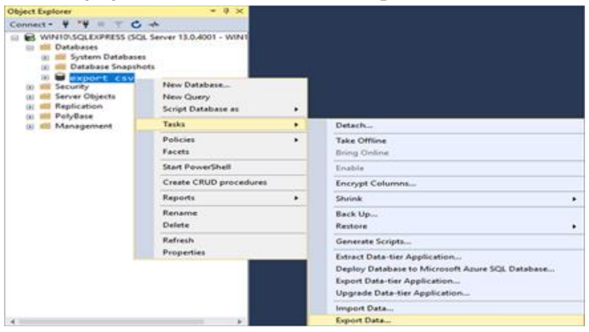

Figure 1: Process

UPLOAD THE CSV FILE TO AN AMAZON S3 BUCKET USING THE WEB CONSOLE

- After completing the exporting process to your local machine, the next step is to transfer the CSV file to Amazon S3.
- How to upload a CSV file to Amazon S3:
- First, we’ll start by creating a storage bucket.
- Click the Create Bucket button and enter a unique name for your bucket on the form.
- Choose the AWS Region where you’d like to store your data.
- Create a new S3 bucket.
- Create the directory that will hold your CSV file.
- In the Buckets pane, click on the name of the bucket that you created.
- Click on the Actions button and select the Create Folder option.
- Enter a unique name for your new folder and click Create.
- Upload the CSV file to your S3 bucket.
- Select the folder you’ve just created in the previous step.
- Select Files wizard and then click on the Add Files button in the upload section.
- Next, a file selection dialog box will open. Here you will select the CSV file you exported earlier and then click Open.

- Click on the Start Upload button and you are done!

UPLOAD DATA TO SNOWFLAKE FROM S3

Since we already have an Amazon Web Services (AWS) account and we are storing our data files in an S3 bucket, we will leverage our existing bucket and folder paths for bulk loading into Snowflake.
To allow Snowflake to read data from and write data to an Amazon S3 bucket, we first need to configure a storage integration object to delegate authentication responsibility for external cloud storage to a Snowflake identity and access management (IAM) entity.
Define Read-Write Access Permissions for the AWS S3 Bucket
Allow the following Actions:
“s3:PutObject”
“s3:GetObject”
“s3:GetObjectVersion”
“s3:DeleteObject”
“s3:DeleteObjectVersion”
“s3:ListBucket”
Create an AWS IAM Role and record your IAM Role ARN value located on the role summary page because we are going to need it later on.
Create a cloud storage integration using the STORAGE INTEGRATION command.

```
CREATE STORAGE INTEGRATION <integration_name>
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN = '<iam_role>'
STORAGE_ALLOWED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/')
[ STORAGE_BLOCKED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/') ]
```

Where:
<integration_name> is the name of the new integration.
<iam_role is> the Amazon Resource Name (ARN) of the role you just created.
<bucket> is the name of an S3 bucket that stores your data files.
<path> is an optional path that can be used to provide granular control over objects in the bucket.
Recover the AWS IAM User for your Snowflake Account
Execute the DESCRIBE INTEGRATION command to retrieve the ARN for the AWS IAM user that was created automatically for your Snowflake account:

```
DESC INTEGRATION <integration_name>;
```

Record the following values:
STORAGE_AWS711/_IAM_USER_ARN — The AWS IAM user created for your Snowflake account.
STORAGE_AWS_EXTERNAL_ID — The external ID that is needed to establish a trust relationship.

GRANT THE IAM USER PERMISSIONS TO ACCESS BUCKET OBJECTS

Log into the AWS Management Console and from the console dashboard, select IAM.
Navigate to the left-hand navigation pane and select Roles and choose your IAM Role.
Select Trust Relationships followed by Edit Trust Relationship.
Modify the policy document with the IAM_USER_ARNand STORAGE_AWS_EXTERNAL_ID output values you recorded in the previous step.
Click the Update Trust Policy button to save the changes.
Step 3.6: Create an External Stage that references the storage integration you created

```
grant create stage on schema public to role <IAM_ROLE>;
grant usage on integration s3_int to role <IAM_ROLE>;
use schema mydb.public;
create stage my_s3_stage
storage_integration = s3_int
```

url = 's3://bucket1/path1'
file_format = my_csv_format;

Step 3.7: Execute COPY INTO <table> SQL command to load data from your staged files into the target table using the Snowflake client, SnowSQL.

Seeing that we have already configured an AWS IAM role with the required policies and permissions to access your external S3 bucket, we have already created an S3 stage. Now that we have a stage built in Snowflake pulling this data into your tables will be extremely simple.

copy into mytable
from s3://mybucket credentials=(aws_key_id='$AWS_ACCESS_KEY_ID' aws_secret_key='$AWS_SECRET_ACCESS_KEY')
file_format = (type = csv field_delimiter = '|' skip_header = 1);

This SQL command loads data from all files in the S3 bucket to your Snowflake Warehouse.

LIMITATIONS AND CHALLENGES OF USING CUSTOM CODE

This method is only intended for files that do not exceed 160GB. Anything above that will require you to use the Amazon S3 REST API.

This method doesn’t support real-time data streaming from SQL Server into your Snowflake DW.

If your organization has a use case for Change Data Capture (CDC), then you could create a data pipeline using Snow pipe.

Also, although this is one of the most popular methods, there are a lot of steps that you need to get right to achieve a seamless migration. Some of you might even go as far as to consider this approach to be cumbersome and error prone.

REFERENCES

[1]. https://aws.amazon.com/premiumsupport/knowledge-center/read-access-objects-s3-bucket/
[2]. https://hevodata.com/?utm_source=sqlserver_to_snowflake&utm_medium=l1_home&utm_campaign=blog_click