

Demystifying Stages in Snowflake

Introduction

Stages in Snowflake are locations used to store data. If the data that needs to be loaded into Snowflake is stored in other cloud regions like **AWS S3** or **Azure** or **GCP** then these are called External stages whereas if the data is stored inside Snowflake then these are called Internal stages.

Table of Contents

1. Internal Stages
2. Use Case for Internal Stage
3. External Stages
4. Use Case for External Stage

1. Internal Stages

First, I will walk through the Internal Stages. Internal Stages are further divided as below

1. User Stages
2. Table Stages
3. Internal Named stages

Before ingesting data into a table in Snowflake, data has to be loaded into a stage using the **PUT** command, and then it has to be loaded into the table using the **COPY INTO** command. Similarly, if we need to unload the data from a table, it has to be loaded into the stage using the **GET** command, and then it has to be exported using the **COPY INTO** command.

Note that there is a slight change in the syntax based on whether the local system is Unix or windows. I have used a windows based system to store the local files in this article.

As of now, all the tasks are done via the **SNOWSQL** command-line interface. **SNOWSQL** is a client tool where you can install on your local machine and then connect to the snowflake. When you load the data from local to the stage, snowflake automatically compresses the file. If you check the file in the

stage after loading you will see an extension of .gz all the time inside the stage.

1.a.User stages

User stages are tied to a specific user. Every user has a default stage created. We will not be able to either modify or remove these stages.

We can copy the files to these stages to load them further into the table. Once the load is completed we need to ensure to remove these files explicitly otherwise we need to pay for storage. Files in one user stage cannot be accessed by another user. So if you need to load multiple tables from a specific user then this is the best option. We need to refer to the user stages using '@~'

1.b.Table stages

Table stages are tied to a specific user. Whenever a table is created, then automatically the table stage is created. Similarly, the user stage will not be able to either modify or remove the table stage, however, we need to clean up space after the files are loaded. The table stage for a particular table cannot be accessed through another table. So if you need to load one table then you can choose the table stage. We need to refer to the user stages using '@%'

1.c.Internal Named Stages

These stages offer more flexibility compared to user or table stages. These are some of the snowflake objects. So all the operations that can be performed on objects can be performed on Internal named stages as well. We need to create these stages manually and we can also specify the file format options while creating the stage itself which is unlike the table or user stage. We need to refer to the user stages using '~'.

2.Use Case for Internal Stage

2.a.Problem Definition

Load source files from the local system into multiple tables in snowflake and then process the data. Processed data will be available in the target table. Unload the data from the target table into a file in the local system.

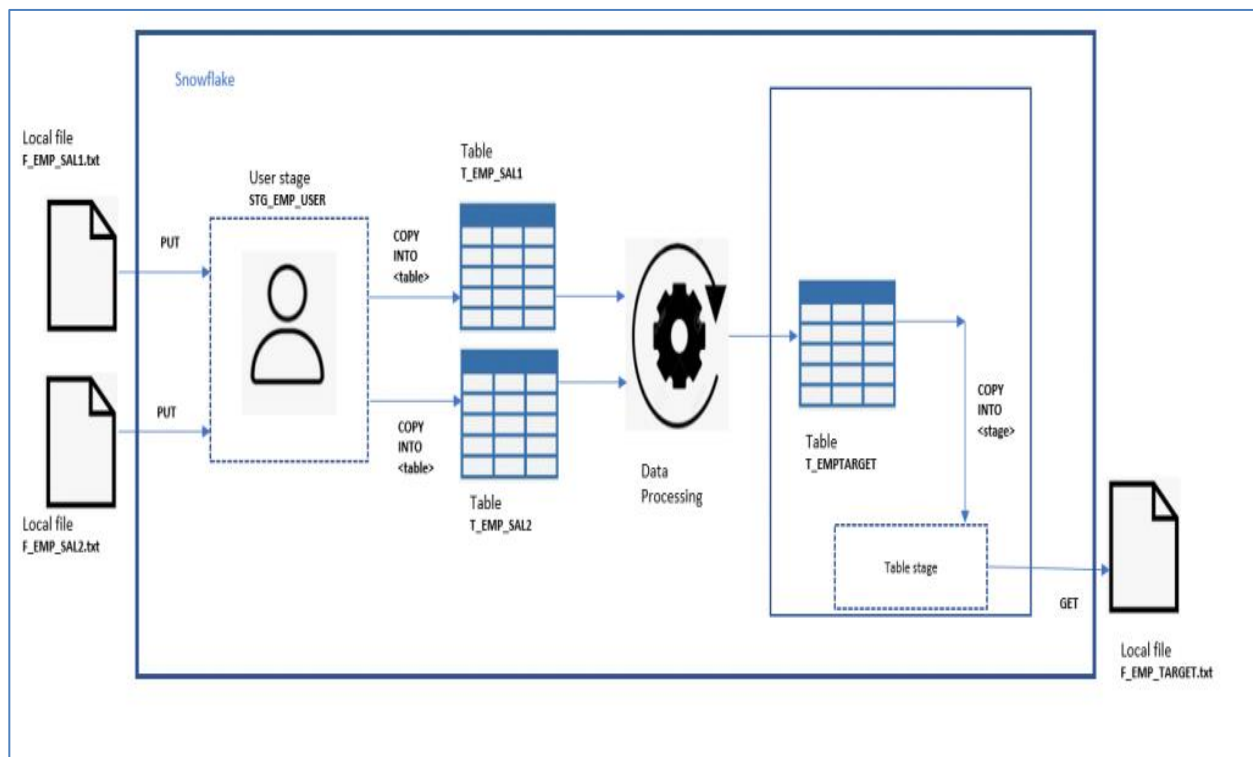
2.b.Solution

Since we need to load multiple tables here, we can either go for the user stage or named stage. Processing the data is out of scope for this article, so I have not included that part. Assume that processed data is available in the target table. Since it is a single target table, we can use either table stage or named stage to unload the data from a table.

2.c.Steps to be Implemented

1. **Connect and log in to SNOWSQL.**
2. **Load all the files from the local disk to the user stage using the PUT command.**
3. **Load the files from the user stage to the source tables in snowflake using the COPY INTO table command.**
4. **Unload the file from the target table into the table stage using COPY into the table stage.**
5. **Unload the stage and copy it into the local system using GET.**
6. **Clean all the files in the stages to avoid billing for storage.**

2.d.The Architecture of the Implementation



2.e.Naming Conventions

To understand easily I will use the naming conventions as below

- Source Files in local disk – **F_EMP_SAL1.txt,F_EMP_SAL2.txt**
- Source Tables :**T_EMP_SAL1,T_EMP_SAL2**
- User stages : **STG_EMP_USER**
- Target table: **T_EMPTARGET**
- Table stages: **STG_EMP_TABLE**
- Target file in local: **F_EMP_TARGET**

2.f.Pre-requisites for Implementation

- SNOWSQL should be installed in the client machine.
- Create a Snowflake free trial account.
- Local txt Files with content is as below:
- **F_EMP_SAL1.txt**

*1,aaron,3000
2,vidhya,4000*

- **F_EMP_SAL2.txt**

*1,Ben,7000
2,vidhya,2000*

- Source and target tables are created in Snowflake with the below script

```
use role accountadmin;
use warehouse compute_wh;
use database DEMO_DB;
use schema PUBLIC;

create table T_EMP_SAL1(emp_id integer,emp_name varchar,empsal
float);

create table T_EMP_SAL2(emp_id integer,emp_name varchar,empsal
float);

create table T_EMP_TARGET(emp_id integer,emp_name varchar,empsal
float);

insert into T_EMP_TARGET values
(1,'Aaron',8000),
(1,'Vidhya',4000),
(1,'Ben',7000);
```

2.g.Implementation

We will see the code used in this section

Code

#Logging into SNOWSQL

```
snowsql -a <>.ap-south-1.aws -u <>
```

#1.Load all the files from the local disk to the user stage using PUT command

```
use role accountadmin;  
use warehouse compute_wh;  
use database DEMO_DB;  
use schema PUBLIC;  
put file://D:SnowflakeContentF_EMP_SAL1.txt @~;  
put file://D:SnowflakeContentF_EMP_SAL2.txt @~;  
select * from T_EMP_SAL1;#This returns empty table  
select * from T_EMP_SAL2;#This returns empty table
```

#2.Load the files from the user stage to the source tables in snowflake using COPY INTO table command

```
copy into T_EMP_SAL1 from @~/F_EMP_SAL1.txt;  
copy into T_EMP_SAL2 from @~/F_EMP_SAL2.txt;  
select * from T_EMP_SAL1;  
select * from T_EMP_SAL2;
```

#3.Copy the file from target table into the table stage

```
copy into @%T_EMP_TARGET from T_EMP_TARGET;  
select * from T_EMP_TARGET;
```

#4.Unload into local system using GET.

```
get @%T_EMP_TARGET file://D:SnowflakeContent
```

#5.Clean all the files in the stages to avoid billing for storage.

```
list @~;  
rm @~ pattern='*.txt.*';  
rm @%T_EMP_TARGET;
```

Let's see the output now for all the above commands

```
1 Row(s) produced. Time Elapsed: 0.364s
LEARNFUNDASH#COMPUTE_WH@DEMO_DB.PUBLIC>put file://0:\Snowflake\Content\F_EMP_SAL1.txt @~/F_EMP_SAL1.txt;
put file://0:\Snowflake\Content\F_EMP_SAL2.txt @~/F_EMP_SAL2.txt;
select * from T_EMP_SAL1;
```

F_EMP_SAL1.txt_c.gz(0.00MB): [#####] 100.00% Done (0.104s, 0.00MB/s).

source	target	source_size	target_size	source_compression	target_compression	status	message
F_EMP_SAL1.txt	F_EMP_SAL1.txt.gz	27	62	NONE	GZIP	UPLOADED	

1 Row(s) produced. Time Elapsed: 1.942s

F_EMP_SAL2.txt_c.gz(0.00MB): [#####] 100.00% Done (0.112s, 0.00MB/s).

source	target	source_size	target_size	source_compression	target_compression	status	message
F_EMP_SAL2.txt	F_EMP_SAL2.txt.gz	25	60	NONE	GZIP	UPLOADED	

1 Row(s) produced. Time Elapsed: 0.993s

```
LEARNFUNDASH#COMPUTE_WH@DEMO_DB.PUBLIC>copy into T_EMP_SAL1 from @~/F_EMP_SAL1.txt;
```

file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column_name
F_EMP_SAL1.txt/F_EMP_SAL1.txt.gz	LOADED	2	2	1	0	NULL	NULL	NULL	NULL

1 Row(s) produced. Time Elapsed: 1.489s

```
LEARNFUNDASH#COMPUTE_WH@DEMO_DB.PUBLIC>copy into T_EMP_SAL2 from @~/F_EMP_SAL2.txt;
```

file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column_name
F_EMP_SAL2.txt/F_EMP_SAL2.txt.gz	LOADED	2	2	1	0	NULL	NULL	NULL	NULL

```

2 Row(s) produced. Time Elapsed: 0.1512s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>select * from T_EMP_SAL1;
+-----+-----+-----+
| EMP_ID | EMP_NAME | EMPSAL |
+-----+-----+-----+
|      1 | aaron    |   3000 |
|      2 | vidhya   |   2000 |
+-----+-----+-----+
2 Row(s) produced. Time Elapsed: 0.183s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>select * from T_EMP_SAL2;
+-----+-----+-----+
| EMP_ID | EMP_NAME | EMPSAL |
+-----+-----+-----+
|      1 | Ben      |   7000 |
|      2 | vidhya   |   2000 |
+-----+-----+-----+
2 Row(s) produced. Time Elapsed: 0.2125s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>copy into @@T_EMP_TARGET from T_EMP_TARGET;
+-----+-----+-----+
| rows_unloaded | input_bytes | output_bytes |
+-----+-----+-----+
|              3 |          38 |           51 |
+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 1.203s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>select * from T_EMP_TARGET;
+-----+-----+-----+
| EMP_ID | EMP_NAME | EMPSAL |
+-----+-----+-----+
|      1 | Aaron    |   8000 |
|      1 | Vidhya   |   4000 |
|      1 | Ben      |   7000 |
+-----+-----+-----+

```




```

5 Row(s) produced. Time Elapsed: 0.148s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>get @%T_EMP_TARGET file://D:\Snowflake\Content\
data_0_0_0.csv.gz(0.00MB): [#####] 100.00% Done (0.129s, 0.00MB/s).
+-----+-----+-----+-----+
| file           | size | status  | message |
+-----+-----+-----+-----+
| data_0_0_0.csv.gz | 51  | DOWNLOADED |         |
+-----+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 2.071s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>list @~;
+-----+-----+-----+-----+
| name                                                    | size | md5                                                    | last_modified |
+-----+-----+-----+-----+
| F_EMP_SAL1.txt/F_EMP_SAL1.txt.gz                       | 64   | 85d4eb12916beace441df9b5e1c01738 | Fri, 16 Jul 2021 13:41:13 GMT |
| F_EMP_SAL2.txt/F_EMP_SAL2.txt.gz                       | 64   | 4669039276b56e5fd5166399711ed370 | Fri, 16 Jul 2021 13:41:14 GMT |
| worksheet_data/01cd779c-8fbb-4555-a5ba-a56d589ef6b4    | 1600 | ae8774863a23084fb26b76e33668ef49 | Mon, 12 Jul 2021 10:34:05 GMT |
| worksheet_data/25f2ff21-421e-404a-99e6-85f79b068e2f    | 816  | f52de973fdc46c9ca34074eee32e8985 | Tue, 13 Jul 2021 12:23:59 GMT |
| worksheet_data/metadata                                | 864  | a0293d5629add78b4d37e542fe8401eb | Tue, 13 Jul 2021 12:23:59 GMT |
+-----+-----+-----+-----+
5 Row(s) produced. Time Elapsed: 0.125s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>rm @~ pattern='*.txt.*';
+-----+-----+
| name                               | result |
+-----+-----+
| F_EMP_SAL2.txt/F_EMP_SAL2.txt.gz | removed |
| F_EMP_SAL1.txt/F_EMP_SAL1.txt.gz | removed |
+-----+-----+
2 Row(s) produced. Time Elapsed: 0.171s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>rm @%T_EMP_TARGET;
+-----+-----+
| name           | result |
+-----+-----+
| data_0_0_0.csv.gz | removed |
+-----+-----+
1 Row(s) produced. Time Elapsed: 0.199s
LEARNFUNDAS#COMPUTE_WH@DEMO_DB.PUBLIC>

```

2.h.Output

Let's see if the file is exported.

This PC > DATA (D:) > Snowflake > Content			
Name ^	Date modified	Type	Size
 data_0_0_0.csv.gz	7/13/2021 4:09 PM	GZ File	1 KB
 F_EMP_SAL1	7/13/2021 3:47 PM	Text Document	1 KB
 F_EMP_SAL2	7/13/2021 3:47 PM	Text Document	1 KB

A	B	C	D
1	Aaron	8000	
1	Vidhya	4000	
1	Ben	7000	

3.External Stage

If the files are located in an external cloud location, for example, if you need to load files from AWS S3 into snowflake then an external stage can be used.

Unlike Internal stages, loading and unloading the data can be directly done using COPY INTO. Get and Put commands are not supported in external stages. The external stage can be created via Web user interface or SNOWSQL as well. We will see how to create using the web user interface.

4.Use Case for External Stage

4.a.Problem Definition

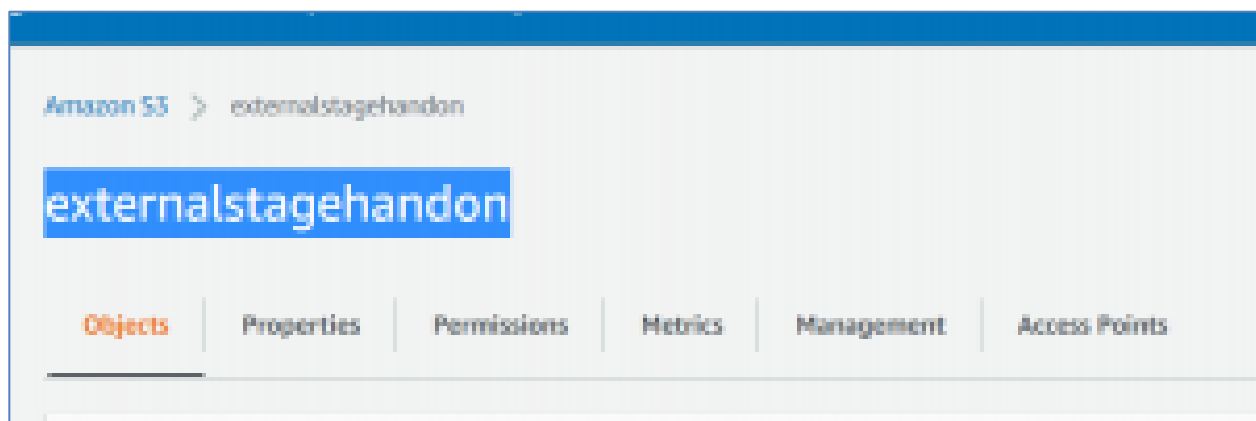
Load files from AWS S3 into a snowflake table using an external stage.

4.b.Prerequisites

1. To perform this demo, you need to have an AWS account.
2. An access key and secret key to connect to an AWS account.
3. Create a table where data needs to be loaded in snowflake with the below script

4.c.Steps for Implementation

1.Create an AWS S3 bucket as shown below



2.Upload files on S3

Amazon S3 > externalstagehandon

externalstagehandon

Objects Properties Permissions Metrics Management Access Points

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	F_EMP_SAL1.txt	txt	July 13, 2021, 17:04:09 (UTC+05:30)	27.0 B	Standard

3. Create an external stage in snowflake using AWS keys

```

24
25 copy into T_EMP_SAL_EXT
26 from s3://externalstagehandon/
27 credentials=(aws_key_id='XXXXXXXX' aws_secret_key='XXXXXX');
28
29

```

Results Data Preview Open History

✓ Query ID SQL 3.85s 1 rows

Filter result... Download Copy Columns ▾

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column
1	s3://externalsta...	LOADED	2	2	1	0	NULL	NULL	NULL	NULL

4.d.Output

```

28
29
30 select * from T_EMP_SAL_EXT

```

Results Data Preview Open History

✓ Query ID SQL 155ms 2 rows

Filter result... Download Copy Columns ▾

Row	EMP_ID	EMP_NAME	EMPSAL
1	1	aaron	3000
2	2	vidhya	4000