ALL ABOUT SNOWFLAKE STAGES

Snowflake Stages :

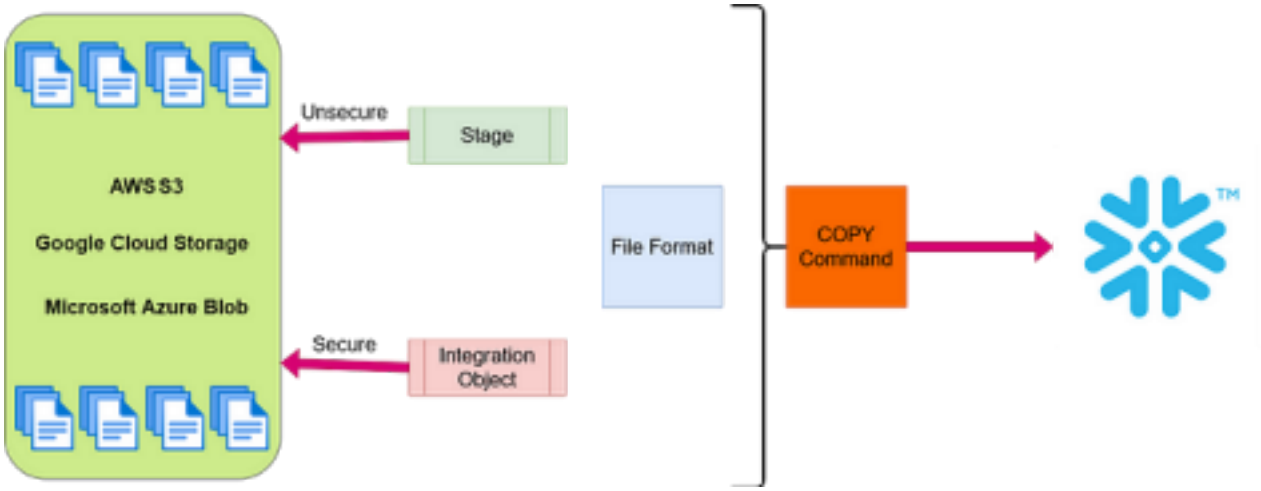
The staging area in snowflake is a blob storage area where you load all your raw files before loading them into the snowflake database. Snowflake stage is not a data warehouse stage.

Snowflake provides an **internal stage** area as well as an **external stage area**. The external stage area includes **Microsoft Azure Blob storage, Amazon AWS S3,** and **Google Cloud Storage.**

External Stage Area Available in Snowflake

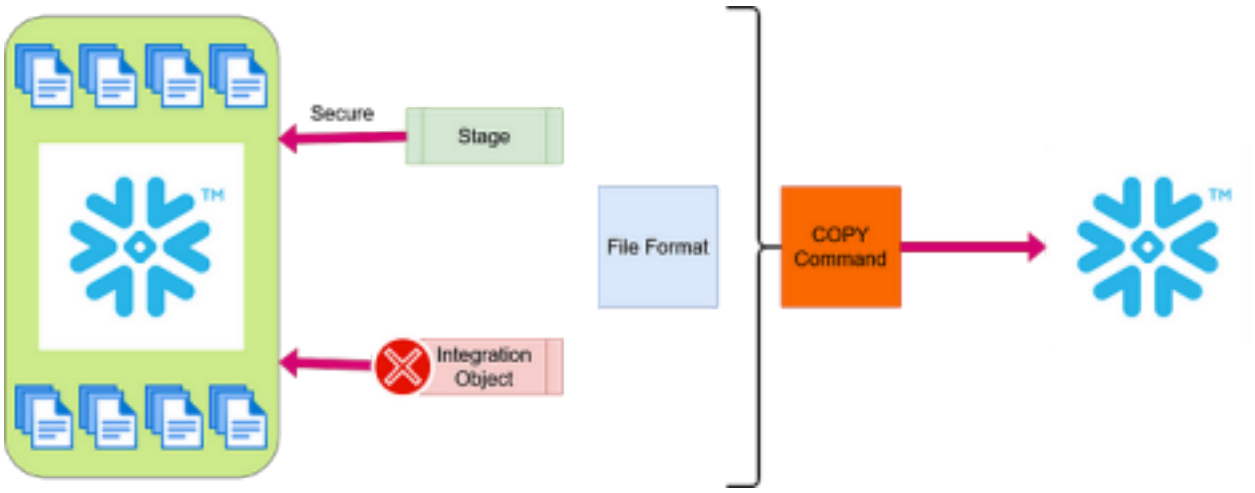
In the external stage area, we store our raw files in the blob storage provided by Microsoft, Google, and Amazon. In order to make a connection between the external staging area and the snowflake, we use a stage object. But this stage object will not be secure since the **Stage object** describes how to load data from the external staging area and in order to make it secure, we use the **Integration object**.

After this, we also require a file format which describes the file properties. All these objects are used by the COPY command to load data.



Data Loading Process from External Stage

In the internal stage area, the process will remain almost the same, the only difference will be that the integration object will not be required. This is because the staging area will be provided by Snowflake and the stage object will be sufficient to provide the secure connection.



Data Loading Process From External Stage

Types of Internal Stages :

We have 3 types of internal stages in the Snowflake.

1. User Stage
2. Table Stage
3. Named Stage

User Stages — Each User is allocated to them by default for storing the files. This is the best suited option if your files will be accessed by ***only one user and are required in multiple tables.*** The user would require access to load the data from its stage to the tables.

Table Stages — Each Table in the Snowflake is allocated with a default file storing stage. This is the best-suited if your files need to be accessed by ***multiple users and are required/copied into one table.*** Unlike the Named stages, table stages can't be altered or dropped. It does not support creating the file format inside this stage, so we need to explicitly define the File Format in the COPY INTO <table> Command.

Table stages also do not support the transforming data while loading it.

Named Stages — The Named stages provide the greatest degree of flexibility for data loading. Because they are database objects, the security/ access rules that apply to the database, will apply the same to its objects as well. ***Users with appropriate privileges on the Named stage can be load data into any table within the database.***

Data Loading with Snowflake Stage :

1. Internal Stages

Internal stages are storage locations managed by Snowflake within your Snowflake account. These stages simplify the data ingestion process and are commonly used for temporary storage during data loading or unloading.

Types of Internal Stages:

- **User Stage**:
  Each Snowflake user automatically has a personal stage. It's tied to the user account and is useful for loading or unloading small datasets.
  - **Access via @~.**
- **Table Stage**:
  Every table in Snowflake has a dedicated stage for storing files temporarily during bulk operations. Files in a table stage are accessible only to the table they are associated with.
- **Named Internal Stage**:
  These are explicitly created and named by users. They can be shared across sessions and objects, making them suitable for reusable storage.

Use Cases:

- Quick and secure data loading/unloading.
- Temporary storage for intermediate processing.

With Internal Stage :

Using the internal table stage, we will understand how to load the data into Snowflake.

For that, we would require to have a prerequisite installation of the **SnowSQL** for which you can refer documentation :
https://github.com/Guruvendra47/Snowflake/blob/main/L_45_Data_Unloading_With_Int_Ext_Stage/SNOWSQL_CLI.pdf

We can make use of the COPY command to load the data from the table to its stage.

Once we unload the data in the stage, we can download it into our local machine using the **GET** command.

-- create a file format

```
create or replace file format csv_file_format
 type = 'csv'
 compression = 'none'
 field_delimiter = ','
 field_optionally_enclosed_by = 'none'
 skip_header = 1 ;
```

-- creating an internal stage

```
CREATE OR REPLACE STAGE my_internal_stage
file_format = csv_file_format;
```

-- Example for Table 1

COPY INTO @my_internal_stage/customer_data.csv → **give complete file name else default data.csv will be given**
FROM (SELECT * FROM customer_data); → **choose any table available in your database**

-- Example for Table 2

```
COPY INTO @my_internal_stage/sales_region_data.csv
FROM (SELECT * FROM sales_region_data)
OVERWRITE = TRUE ;
```

-- - We can use OVERWRITE = TRUE in the COPY Statement in order to overwrite a file which is already existing in the table stage.

--List down all the stages in snowflake

```
SHOW STAGES;
```

--list the contents available in stage(will show above 2 files)

```
LIST @my_internal_stage;
```

GET @my_internal_stage file://D:\Snowflake_Output; → **target repo path in your local**
-- run this in SNOWSQL using CLI not in SNOWFLAKE UI as from snowflake ui it will throw

unsupported feature

2. External Stages

External stages point to storage locations outside of Snowflake, such as cloud storage services (AWS S3, Azure Blob Storage, or Google Cloud Storage). External stages provide a seamless way to integrate Snowflake with external storage solutions.

Features:

- Data is accessed directly from the external storage without the need to copy it into Snowflake-managed storage.
- Snowflake supports secure integration with external services via credentials or storage integrations.

Use Cases:

- Loading data directly from external storage into Snowflake.
- Offloading data from Snowflake to external storage for archiving or sharing.

With External Stage :

--create a storage integration for linking snowflake with aws using specified role and policies

```
CREATE OR REPLACE STORAGE integration s3_int
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN ='arn:aws:iam::661806635168:role/banking_role'
STORAGE_ALLOWED_LOCATIONS =('s3://czec-banking/DOWNLOAD_CSV/');
```

--Describe the integration

```
DESC integration s3_int;
```

-- create a file format

```
create or replace file format csv_file_format
 type = 'csv'
 compression = 'none'
 field_delimiter = ','
 field_optionally_enclosed_by = 'none'
 skip_header = 1 ;
```

-- Define an external stage to access the AWS S3 bucket.

-- Ensure the necessary IAM role and S3 bucket policies are set for Snowflake to read from the bucket.

```
CREATE OR REPLACE STAGE s3_stage
URL ='s3://czec-banking/DOWNLOAD_CSV/'
file_format = csv_file_format
storage_integration = s3_int;
```

-- to show all stages

```
SHOW STAGES;
```

-- To show files processsed in stage

```
LIST @s3_stage;
```

-- Example for Table 1

```
COPY INTO @s3_stage/customer_data.csv
FROM (SELECT * FROM customer_data);
```

-- Example for Table 2

```
COPY INTO @s3_stage/sales_region_data.csv
FROM (SELECT * FROM sales_region_data);
```

ALTERNATIVE APPROACH :

-- if staging details are not availble such as no roles and policies have been created abnd client have given aws access key then execute below steps

```
CREATE OR REPLACE STAGE my_external_stage
URL = 's3://czec-banking/DOWNLOAD_CSV/' -- path to S3 bucket
folder --STORAGE_INTEGRATION = my_s3_integration
CREDENTIALS = (
 AWS_KEY_ID = '***************' -- aws access key
 AWS_SECRET_KEY = '************' -- aws secret access key
)
FILE_FORMAT = csv_file_format;


COPY INTO @my_external_stage/customer_data.csv
FROM (SELECT * FROM customer_data)
OVERWRITE=TRUE ;


COPY INTO @my_external_stage/sales_region_data.csv
FROM (SELECT * FROM sales_region_data)
OVERWRITE=TRUE ;
```

-- if file already exists with the same name it will overwrite it else if you have not use this property and trying to write to the same file you get an error

-- copying data into external stage give proper file name with extension else it will create with data.csv(filename)

3. Temporary Stages

Temporary stages are implicitly created by Snowflake when data loading or unloading commands are executed without specifying a stage. These stages are not persistent and exist only for the duration of the session or command execution.

Features:

- Automatically created for specific operations.
- No explicit management required by users.

Use Cases:

- Simplified workflows for ad hoc data loading/unloading.

Comparison of Stages

Feature	Internal Stages	External Stages	Temporary Stages
Management	Fully managed by Snowflake	Managed by the user (external storage service)	Implicitly managed by Snowflake
Use Case	Temporary or reusable data storage	Direct access to external storage systems	Ad hoc, session-specific operations
Security	Secured within Snowflake	Requires credentials or storage integrations	Secured within Snowflake
Persistence	Persistent (except for session-based usage)	Depends on external service	Non-persistent

Choosing the Right Stage

- **Internal Stages**: Use when you need simple and secure data loading/unloading and don't want to manage external storage.
- **External Stages**: Use when your data resides in external storage like AWS S3, Azure Blob Storage, or GCS.
- **Temporary Stages**: Use for one-time operations or when you don't need to persist staging configurations.