Loading Data Into Snowflake

To be able to access data inside Snowflake there are a few methods available to us. Let's look at the different data loading methods available to us in Snowflake:

1. Bulk/Batch Data Load

2. Continuous Data Load

Bulk Data Load

Let's start our discussion with Bulk Data Load, which is the most frequently used method through which data is loaded into Data Warehouse or other data platforms.

The data that is loaded through a batch process are usually extracted from the sources at a regular interval of time. The batch data usually consists of bulk rows that need to be loaded all at once as opposed to event/streaming data.

For loading, the bulk data Snowflake provides the COPY command. The COPY command can load data from cloud storage locations such as Amazon S3, Azure Blob Storage and Google Cloud Storage. The COPY command may also be used to load data which is available in the local file system.

It is important to note that the COPY command uses the compute resources of the Snowflake cluster which is loading data into. It will make use of a virtual warehouse instance to load the data and it is up to the architect and developers of the system to adequately scale the virtual warehouse up or down according to the loading needs.

The COPY command also provides some basic capabilities such as reordering columns, pruning columns, casting columns and the ability to truncate strings to a specific size.

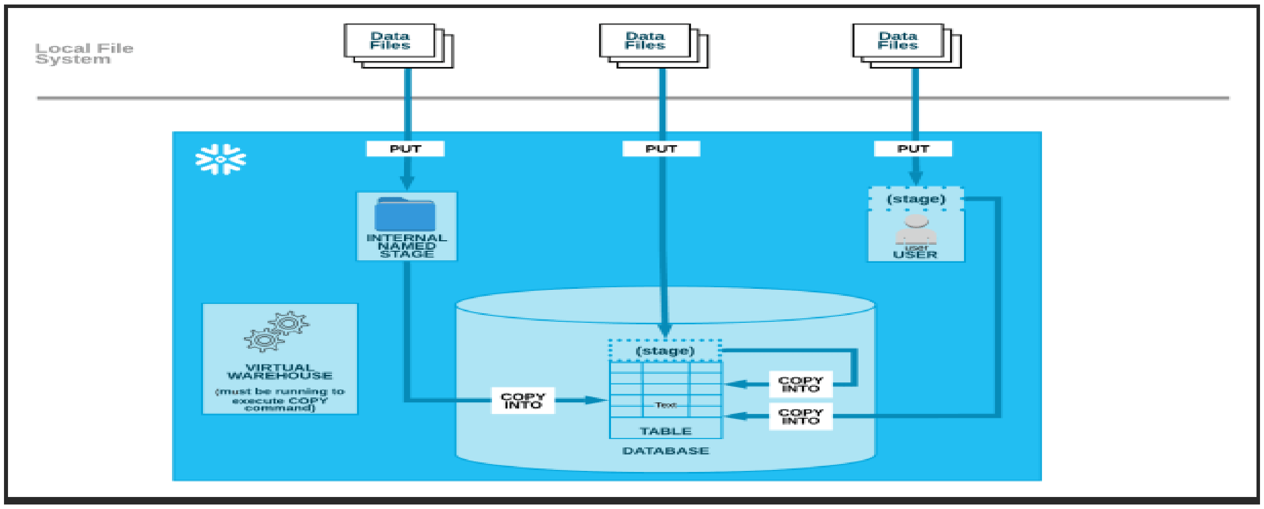


Fig: Bulk Data Load from Local File System

Continuous Load

What if we have streaming data? Or data in micro-batches that we want to load more frequently into Snowflake. SNOWPIPE is used to load streaming data into Snowflake in a continuous manner.

Snowpipe uses a serverless architecture so it is managed directly by Snowflake. We don't need to manage the resources of Snowpipe ourselves and we don't need to scale it up or down ourselves either i.e., it will scale up or down as per the need.

A key thing to note is that because of its serverless architecture it does not use the virtual warehouse compute resources. The cost for the compute used by Snowpipe is determined separately from the virtual warehouse compute.

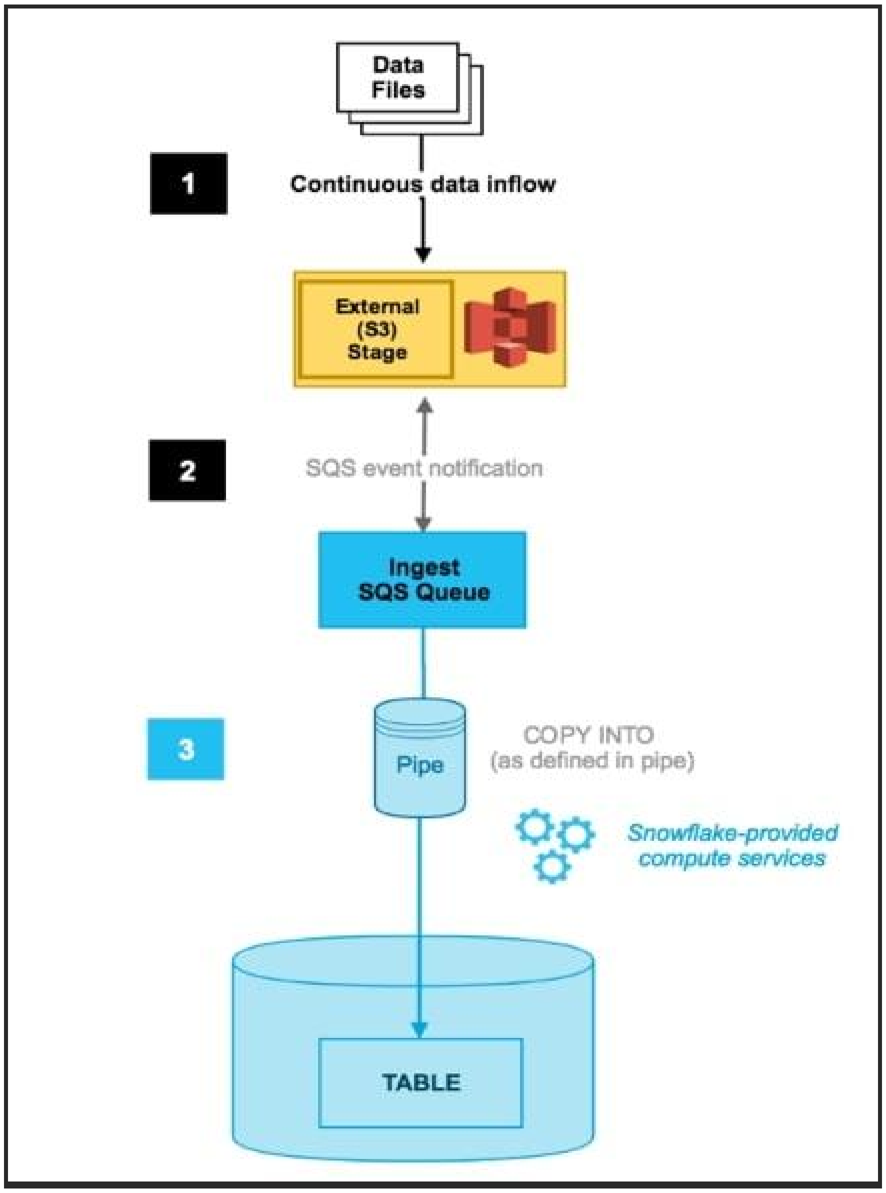
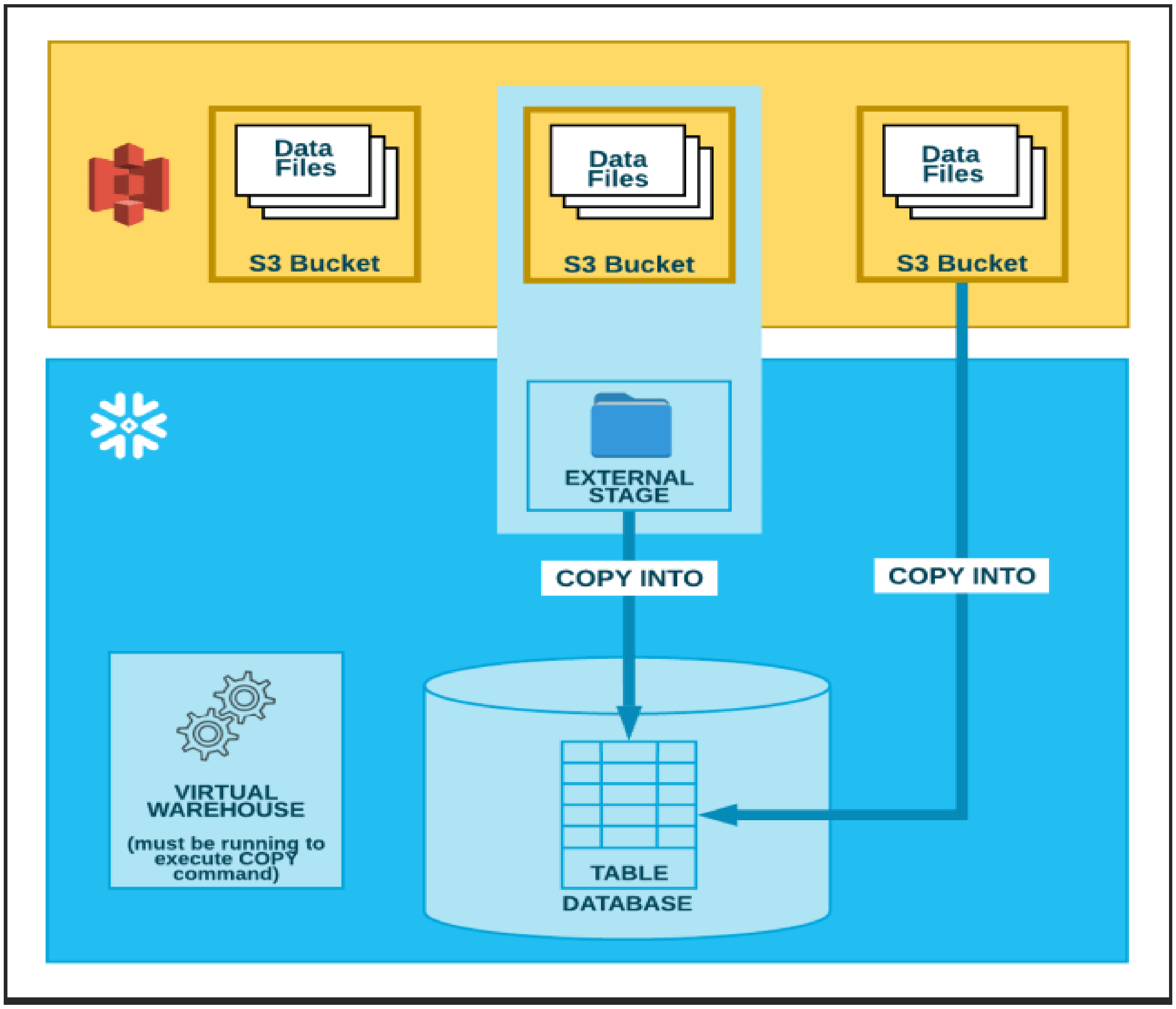


Fig: Continuous Data Loading using Snowpipe

We will discuss Snowpipe in an upcoming blog in detail.

Steps in loading bulk data from Cloud or Local storage:

1. Prepare the files – Before loading the data into Snowflake, we need to prepare the files into format and size which is optimal in terms of processing and cost.
2. Stage the data – We would stage our files into S3 bucket or Azure Blob Storage or a local staging area.
3. Execute COPY command – Once the data is staged, we run the COPY command to load the staged data into the table.
4. Manage regular loads – If we need to perform regular loads, then we must organize our files accordingly and schedule our loads.



Bulk Data Load from S3 Bucket

Let's now look into the details of each step except the 4th step (this is beyond the scope of this blog).

1. Prepare the files

Before we stage our files, we need to prepare our files so that they are optimal in term of size and format. Some of the key things around the format and the size are as follows:

a. Delimited Files

1. The field should be delimited by a single character. A consistent character like the pipe (|), comma (,), caret (^), tilde (~).
2. Rows delimited by a different character. Usually, a new line character is a common choice.
3. The number of columns in each row should be consistent.
4. If a field contains a delimiter character, we should enclose the whole field value in between quotes.

b. Optimize file size

1. To take advantage of parallelism, the optimal file size should be around 10 MB to 100 MB.
2. If the file size is very large, then we should split the large file into small multiple chunks ranging in the size of 10 MB to 100 MB.
3. If we have a number of small files, then we must merge them into a single file to achieve optimal sizes.

c. Data types

1. Numeric data types should not have embedded characters. Eg. 12,345 should be simply 12345. Otherwise, it will be treated as two different fields.
2. Date Time data type should be consistent and according to a specific format.

2. Stage the data

Staging the data in the preparation of eventual load is the key concept in many data warehouses.

Now the question is why do we need a Stage?

The stage is an area which is external to the database. But that area is accessible by the database. So this is how Snowflake gets access to the data that is outside of its bounds.

The most common area for staging data for Snowflake is in an S3 bucket or Azure Blob storage. It can also be done in the local file system.

The syntax to create a stage is:

```
CREATE OR REPLACE STAGE <name_of_stage> URL = <'path_to_staging_area'>
STORAGE_INTEGRATION = <name_of_integration_object>
```

You can find the sample code at the bottom of the blog.

3. Execute COPY command

The COPY command is the most common mechanism for loading bulk data into Snowflake in a batch mode. The command supports several options to specify the files to load. We can specify the stage from which we want to load, we can specify the exact file names which we want to load or we can use a pattern matching to load specific files.

<u>Syntax of COPY command is:</u>

COPY INTO <table_name>

FROM @<stage_name>

PATTERN = '.*.csv'

FILE_FORMAT = (type = csv field_delimiter = '|' skip_header = 1)

Sample Code: (Assuming you have CSV files placed in an S3 bucket)

```
CREATE OR REPLACE DATABASE my_db;
USE DATABASE my_db;
CREATE TABLE public.sales (
OrderID NUMBER,
CustomerID NUMBER,
CustomerName STRING,
TransactionDate DATE
);
CREATE OR REPLACE SCHEMA external_stages;
CREATE OR REPLACE SCHEMA file_formats;
--AWS S3 Configuration (ACCOUNTADMIN has the privilege)
CREATE OR REPLACE STORAGE INTEGRATION s3_int
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN =
'arn:aws:iam::111222333444:role/snowflake_role'
STORAGE_ALLOWED_LOCATIONS =
('s3://snowflake-private/knoldus/load_data/');
--Create file format
CREATE OR REPLACE FILE FORMAT my_db.file_formats.my_csv_format
TYPE = CSV FIELD_DELIMITER = '|' SKIP_HEADER = 1 NULL_IF = ('NULL',
'null') EMPTY_FIELD_AS_NULL = TRUE;
--Create external stage
CREATE OR REPLACE STAGE my_db.external_stages.my_s3_ext_stage
STORAGE_INTEGRATION = s3_int
URL = 's3://snowflake-private/knoldus/load_data/'
--Copy command
COPY INTO my_db.public.sales FROM
@my_db.external_stages.my_s3_ext_stage
FILE_FORMAT = (FORMAT_NAME = 'my_csv_format')
ON_ERROR = 'skip_file';
SELECT * FROM my_db.public.sales LIMIT 20;
```