Optimizing Snowflake

A real-world guide

MATILLION | snowflake®



Contents

2 Introduction
3 About this book
4 Snowflake Architecture
5 Snowflake Virtual Warehousing
7 Accessing Snowflake
9 Loading Data
16 Exporting Data
18 Storage and Compute Costs
20 Best Practices
21 Conclusion

Introduction

If you are reading this eBook, you are probably considering or have already selected Snowflake—the data warehouse built for the cloud—as your modern cloud data warehouse. That is an excellent choice. You are ahead of the curve (and your competitors, with their outdated on-premise databases).
Now what?

Whether you're a data warehouse developer, a data architect or manager, a business intelligence specialist, an analytics professional, or a tech-savvy marketer, you now need to make the most of the Snowflake platform to get the most out of your data. That's where this eBook comes in handy. It introduces the Snowflake platform and helps you understand the various options available in the platform to ingest, manipulate, and export data that is performant and cost-efficient.

This eBook is brought to you by Matillion. Matillion is an industry-leading data transformation solution for cloud data warehouses. Delivering a true end-to-end data transformation solution (not just data prep or movement from one location to another), Matillion provides an instant-on experience to get you up and running in just a few clicks, a pay-as-you-go billing model to cut out lengthy procurement processes, and an intuitive user interface to minimize technical pain and speed up time to results. Matillion is available globally for Snowflake on AWS Marketplace, Microsoft Azure Marketplace and Google Cloud Platform.

More information is available at **www.matillion.com**

About this book

Snowflake Architecture

In this chapter, you'll "look under the hood" to understand Snowflake's unique underlying architecture. The primary differences between Snowflake and other cloud offerings is Snowflake's "multi-cluster shared data" approach, which separates your compute and storage capabilities to improve concurrency. Data is accessible as your users need it, speeding up data processing and analysis.

Snowflake Virtual Warehousing

Snowflake is essentially a cluster of compute resource, which makes Snowflake a scalable and powerful solution. Knowing how to size your resources can help you take advantage of Snowflake's highly scalable, low-cost, and automatically managed enterprise data warehouse. Ultimately, this makes your business more performant and cost-efficient.

Accessing Snowflake

This chapter looks Snowflake access options. This may be different from what you are used to since you will have more options. Once you understand the inner workings of Snowflake and can access it, you can start making the most of it. Query your data using SnowSQL, which is specifically designed for Snowflake, or use a number of other methods, giving you the flexibility you need to get the data you need.

Loading Data

This chapter walks you through ways to bring data into Snowflake. This information is useful if your business is just starting out on Snowflake and you need to populate the data warehouse with historical data or external data from many different sources. It's also useful if you want to improve your data loading processes.

Exporting Data

This chapter describes differences between exporting data to an internal or external stage or to a local file system or other target. What you want to do with the data may affect which option is best for your use case.

Storage and Compute Costs

Costs for storage and compute resources couldn't be simpler due to Snowflake's pay-as-you-go pricing model. Although Snowflake is a cost-efficient solution, this chapter provides some tips that can help you keep costs down and under control.

Best Practices

The final chapter ties together all the information in this eBook into best practices that reduce costs, increase performance, and help you make the most out of your data.

Snowflake Architecture

Snowflake is a fully relational, analytical, SQL-based, virtual data warehouse built from the ground up to take advantage of the full elasticity and scalability of the cloud. Snowflake delivers a database-as-a-service (DBaaS) platform to relieve users of the complexity and administrative burdens that plague traditional architectures. Furthermore, Snowflake was built with a new, unique architecture called "multi-cluster shared data."

Snowflake can give your company the flexibility and agility to meet changing data needs. Flexible cloud storage allows you to store nearly unlimited amounts of structured and semi-structured data in a single location, consolidating your disparate data sources. Compute nodes, called "virtual warehouses" execute queries and perform transformations on your data. These compute nodes can be created at any time using either SQL commands or the Snowflake UI. This means your virtual warehouse is scalable, allowing your business to respond to growing data needs, without additional procurement or managerial overhead.

If you have different workloads with different needs, you can size your warehouses to fit your current need. You can even create "multi-cluster warehouses" that scale automatically to accommodate the number of concurrent queries. Best of all, you can turn off any of the virtual warehouses at any time, so you pay only for what you use.

On top of its scalable storage and compute platform, Snowflake handles optimization, security, and availability for your business and provides many other overhead-reducing benefits. This makes it a fully managed solution. Lastly, you can move real-time data, at nearly unlimited scales. With Snowflake Data Sharing, data can be simply shared with another Snowflake account.

Snowflake Virtual Warehousing

In Snowflake, a virtual warehouse, often referred to simply as a "warehouse," is a cluster of compute resources. A warehouse provides necessary resources—such as CPU, memory, and temporary storage—to perform operations in a Snowflake session:

Perform the following actions in Snowflake:

- Executing SQL SELECT statements that require compute resources (for example, retrieving rows from tables and views)
- Loading data into tables (COPY INTO table)
- Performing DML operations (DELETE/INSERT/UPDATE)
- Unloading data from tables (COPY INTO location)



Virtual Warehousing Sizing

The size of a warehouse can impact the amount of time required to execute queries submitted to the warehouse, particularly for larger, more complex queries. In general, query performance scales linearly with warehouse size, because additional compute resources are provisioned with each size increase.

Virtual warehouses come in various sizes: X-Small, Small, Medium, Large, X-Large, 2X-Large, and so on. The size specifies the number of servers that comprise each cluster in a warehouse.

Unlike traditional systems, you are not locked into a specific size. You have the flexibility to change the size of your virtual warehouse at any time: manually via the Snowflake web interface or by using the ALTER WAREHOUSE command.

Also, bigger is not always better. While a larger warehouse places a lot of compute resources at your disposal, depending on the size and distribution of your data, a medium-sized warehouse may suffice. Try different sizing options to see what best suits your requirements.

See Warehouse Considerations in the Snowflake documentation for more advice about sizing.

continued...

Single vs. Multiple Warehouses

Besides scaling a warehouse (up/down), you may also consider creating virtual warehouses of different sizes and then using whichever one is more appropriate for the task at hand.

For example, you may be loading data from two sets of files: one that is just a few thousand rows and another with millions of rows. Instead of resizing a single, small warehouse to handle the larger files, you may create a separate large warehouse and use that. This would free up your small warehouse to do other relevant tasks, and it would also provide the benefit of having rightly sized warehouses handling your loads in parallel.

This can be especially useful when you load data, and it enables faster parallel loads.

Also, you can use the Auto-Suspend and Auto-Resume features to save costs.

Further Reading:

- Overview of Virtual Warehouses
- Warehouse Considerations
- Understanding Snowflake Credit and Storage Usage
- Multi-cluster Warehouses

Accessing Snowflake

Now that you've had a glimpse under the hood of Snowflake, let's look at the different ways you can access and start using Snowflake. Snowflake allows users to interact with it in a number of ways. This chapter explores some of the options available for connecting to your Snowflake account.

How to access Snowflake:

See Snowflake documentation for a list of all available options and further detail on each.

Snowflake Web Interface

Snowflake's web interface is a powerful and easy-to-use platform for accessing your Snowflake account and the data you've loaded there. This is the default method for accessing Snowflake. Use the following URL in any supported browser to access your account using the web interface.

For more information on logging in, see Logging

Once you login, you will be able to manage your databases and warehouses, execute DDL/DML statements using worksheets, review query history, and much more. The screenshot shows the Worksheet page.

Check out Snowflake's the web interface tour for more details on working with Snowflake.

Continued...

SnowSQL: Snowflake's Command-Line Interface (CLI)

If you are a script junkie, you'll love SnowSQL. SnowSQL is a modern CLI that allows users to execute SQL queries, perform all DDL and DML operations—including loading and unloading data into and out of Snowflake—and perform many other tasks. SnowSQL may be used to to access your Snowflake database from the command line quickly and, if required, automate operations by running scripts.

See Installing SnowSQL for instructions on downloading and installing the

SnowSQL (that is, the snowsql executable) can be run as an interactive shell or in batch mode. Here's an example of running a simple query where results are printed to stdout (the console). Note that the login credentials represented in the example command by myaccount are stored in the [connections] section of a configuration file.

HELPFUL HINT: Use SnowSQL for writing and running scripts in Snowflake or for automating your data load and other processes.

Here's another example that executes a script stored in a local file (input_script.sql) and stores the results in a local file. Here, a username and password are used instead of stored credentials.

```
snowsql -a abc123 -u jsmith
-f //tmp/input_script.sql -o
output_file=//tmp/output.csv
-o quiet=true -o friendly=false
-o header=false -o output_
format=csv
```

You may use the PUT command to upload local files into an internal stage and the GET command to download files from an internal stage to local disk/system. Read more about internal and external stages in the Ch 4: Loading Data.

For more details, see the SnowSQL user guide.

OTHER DATA QUERYING AND MANAGEMENT OPTIONS

You may want to query Snowflake from your favorite SQL client or from a reporting or dashboarding tool. If you are a software developer, you may want to access or manage your data in Snowflake using your favorite programming language.
Snowflake provides various drivers or programmatic interfaces, some of which are listed below:

- Snowflake connector for Python
- Snowflake connector for Apache Spark
- JDBC driver
- Node.js driver
- Go Snowflake driver (which provides interface for developing applications using the Go programming language)
- .NET driver
- ODBC driver

Loading Data

To start fully benefiting from the power of Snowflake, you need to load your data into a database. There are several options for loading data into Snowflake tables:

- Bulk load data
- Use Snowpipe to continuously load data
- Load data using the web interface
- Use custom/vendor applications

These options are reviewed in detail below. Choose a method that suits your use case.

Bulk Loading Data

All data platforms provide some means by which to load data in bulk, and files have typically been
the medium of choice. Bulk loading data is a fast and cost-effective method for populating your Snowflake warehouse.

You can bulk load data from files into tables in Snowflake using the COPY INTO table command. You may execute a COPY command from the Snowflake web interface, from a SnowSQL prompt, or from your favorite programming language by using the appropriate driver, as discussed earlier in the Other Data Querying and Management Options section.

1. Check that the files you intend to load are of a supported format (see below).
2. Compress files for faster loading.
3. Check that your target table already exists (use CREATE TABLE to create a table).
4. Ensure that the files are already staged in an internal or external stage.

Review and (where possible) adhere to best practices.

File types supported by the bulk loader:

- Any flat, delimited plain-text format (comma-separated values, tab-separated values, etc.)
- Semi-structured data in JSON, Avro, ORC, Parquet, or XML format (XML is currently supported as a preview feature)

With **Matillion ETL for Snowflake** you can also take advantage of components developed to streamline efforts, such as the built-in scheduler to ensure your Snowflake tables are all updated consistently at a convenient time interval.

Continued...

A stage in Snowflake is a (named) location where you store data files before loading them. Use CREATE STAGE command to create one. Snowflake supports internal and external stages, which in turn, can be permanent or temporary. A temporary stage gets dropped automatically at the end of a session.

Internal Stage

An internal stage stores data within your Snowflake environment. Files can be uploaded into an internal stage using SnowSQL and the PUT command, or they can be loaded programmatically using one of the drivers. Loading data via an internal stage:

Step 1: Upload one or more data files to a Snowflake stage (a named internal stage, a stage for a specified table, or a stage for the current user) using the PUT command.

Step 2: Use the COPY INTO table command to load the contents of the staged file(s) into a Snowflake database table.

When should you use an internal stage?

An internal stage is intended for temporarily holding data files before loading them into tables or downloading to a local system. You will incur standard data storage costs. You should monitor the data files and remove them from the stages once you have loaded the data and you no longer need the files.

External Stage

An external stage stages the data in a location outside of Snowflake. This is usually within Amazon Simple Storage Service (S3), Microsoft Azure Blob Storage or and Google Cloud Storage Containers. External stages can be named entities in Snowflake or references to a location or bucket in the relevant service (for example, s3://bucketname/path-to-file).

Accessing an external stage requires credentials. These can be passed via the CREDENTIALS parameter of the COPY INTO table command. Alternatively, the CREATE STAGE command allows you to specify the credentials required to access an external stage. Snowflake would automatically use the stored-credentials if they are not passed into the COPY INTO command when loading from this stage. However, many organisations frequently rotate keys for added security which may invalidate the stored key and may introduce additional overhead when storing keys within Snowflake.

Continued...

Loading data via an external stage:

Step 1: Use the upload interfaces or utilities provided by Amazon or Microsoft to stage your files into the external stage.

Step 2: Use the COPY INTO table command to load the contents of the staged file(s) into a Snowflake database table.

When should you use an external stage?
External stages may also double as an additional backup for your data or as a location from where other processes or consumers may consume data in the data files.

You may already have existing or established processes for moving data files into an S3 bucket or a Microsoft Azure Container.
Costs related to using an external stage can be found on the bill from the respective vendor. You may use any tools at your disposal to upload files to or download files from an external stage.

Data Staging Examples
Let's look at some examples of using the COPY INTO command to load data from files to tables in Snowflake.

Loading from an internal stage

Assume that you uploaded some files to an internal stage using SnowSQL and the PUT command. Here are some examples of loading a file called 1.csv from various internal stages—a named stage (@mystage), a table stage (%mytable), and a user stage (~):

```
copy into mytable from '@mystage/path 1/file 1.csv';
copy into mytable from '@%mytable/path 1/file 1.csv';
copy into mytable from '@~/path 1/file 1.csv';
```

Loading from an external stage (Amazon S3)

The following example loads data from a file in Amazon S3. Note that credentials are passed as part of the command.

```
copy into mytable
    from 's3://mybucket 1/prefix 1/file 1.csv'
    credentials = (aws_key_id='xxxx' aws_secret_key='xxxxx'
aws_token='xxxxxx');
```

Continued...

Loading data from a file in a named external stage

Here's an example of loading data from a file in a named external stage:

```
copy into mytable
from '@myextstage/some folder/file 1.csv';
```

Loading from an external stage (Microsoft Azure)

The following example loads data from a file in Microsoft Azure.

```
copy into mytable
from 'azure://myaccount.blob.core.windows.net/my
load/encrypted_files/file 1.csv';
```

Further Reading:

- Bulk Loading from a Local File System Using COPY
- Bulk Loading from Amazon S3 Using COPY
- Bulk Loading from Microsoft Azure Using COPY

DID YOU KNOW...

Snowflake allows you to query the data in files in an internal or external stage? See following for more information:

- Querying Staged Data
- Querying Metadata for Staged Files

Using Snowpipe to Load Data

Snowpipe is a service from Snowflake that can be used to load data from files as soon as they are available in a Stage (internal or named-external). The service provides Snowflake-managed compute resources and exposes a set of REST endpoints that can be invoked to initiate loads (COPY).

Snowflake also provides Java and Python APIs that simplify working with the Snowpipe REST API. This is a great option if you are a programmer building tools or implementing workflows for ingesting data files into Snowflake.

Users can build tools that initiate loads by invoking a REST endpoint; without managing a virtual warehouse or manually running a COPY command every time a file needs to be loaded. The service is managed by Snowflake and it automatically scales up or down based on the load on the Snowpipe service.

Snowpipe Billing

Snowpipe is billed based on compute credits used per second. Snowflake tracks the resource consumption of loads for all pipes in an account, with per-second/per-core granularity, as Snowpipe actively queues and processes data files. "Per-core" refers to the physical CPU cores in a compute server. You will see a new line item (SNOWPIPE) on your Snowflake bill. Go to the "Billings and Usage" page in the Snowflake web interface to get a detailed breakdown of Snowpipe usage. You can break usage information down to a specific date and hour as well as

Continued...

Central to Snowpipe is the concept of a “pipe”. A Snowpipe pipe is a wrapper around the COPY command that is used to load a file into the target table in Snowflake. A few of the options from the COPY command are not supported. See CREATE PIPE for more information.

Snowpipe Benefits

Snowpipe provides a serverless data loading option that manages compute capacity on your behalf. You can also take advantage of the per-second/per-core billing to save on compute costs and pay only for the exact compute resources you use.

- Instant insights. Snowpipe immediately provides fresh data to all your business users without contention.
- Cost-effectiveness. You pay only for the per-second compute used to load data rather than the costs for running a warehouse continuously or by the hour.
- Ease-of-use. You can point Snowpipe at an S3 bucket from within the Snowflake UI and data will automatically load asynchronously as it arrives.
- Flexibility. Technical resources can interface directly with the programmatic REST API, using Java and Python SDKs to enable highly customized loading use cases.
- Zero management. Snowpipe automatically provisions the correct capacity for the data being loaded. There are no servers or management to worry about.
- Cost
- Performance

Read more about how you can streamline data loading with Snowpipe.

LOAD FILES BY INVOKING A REST API

Let’s look at how to explicitly invoke Snowflake via the REST API to load files using the Snowpipe service. In this example, we will copy a file to an S3 staging area represented by a named external stage in snowflake and then invoke the Snowpipe REST endpoint to ingest the file. We are ingesting a JSON file which will be loaded into a Variant column in a table.

Continued...

Configuring Snowpipe

You may notice that Steps 1, 2, and 3 below are exactly the same as in the previous section except that the pipe is created without auto_ingest=true.

Step 1: Create a named stage in Snowflake.

```
create or replace stage mydb.public.snowstage
 url='s3://snowpipe-demo/'
 credentials = (AWS_KEY_ID = '...' AWS_SECRET_KEY = '...' );
```

Step 2: Create a table.

```
create table mydb.public.mydatatable(jsontext variant);
```

Step 3: Create a pipe.
Create a new pipe in the system for defining the "COPY INTO <table>" statement used by Snowpipe to load data from an ingestion queue into tables. For more information, see CREATE PIPE.

```
create or replace pipe mydb.public.mysnowpipe as
   copy into mydb.public.mydatatable
    from @mydb.public.snowstage
    file_format = (type = 'JSON');
```

Step 4: Configure security (per user).

Users cannot authenticate with the REST API using their Snowflake login credentials. Generate a public-private key pair for making calls to the Snowpipe REST endpoints. In addition, grant sufficient privileges on the objects for the data load, for example, the target database, schema, and table; the stage object; and the pipe.

For more information on generating a compliant key pair and associating it with the relevant user, see Configure Security in the Snowflake documentation. Also refer to the relevant SDK documentation on key-based authentication.

Continued...

Further Reading:

- Loading Continuously Using Snowpipe
- Understanding Billing for Snowpipe Usage
- How Snowpipe Streamlines Your Continuous Data Loading and Your Business
- Video: Automatically Ingesting Streaming Data with Snowpipe
- Video: Load Data Fast, Analyze Even Faster

Loading Data Using the Web Interface

You may use the Snowflake web interface to issue DML commands such as INSERT and UPDATE or to load data files up to 50MB in size. Both of these options are good for ad hoc data loads and when you are working with small data sets.

The Snowflake web interface provides a simple wizard for loading files into tables. The wizard uploads files to an internal stage (via PUT), and then it uses a COPY command to load data into the table. See the documentation for more information on using the web interface to upload files and load data.

Using Custom/Vendor Applications

You can use your favorite programming language to build your own applications to access and leverage the Snowflake platform. Snowflake provides connectors and drivers for many popular languages that can be used to build custom applications.
SnowSQL is a good example of this. It uses the Python connector provided by Snowflake to provide an effective CLI.

Exporting Data

There are several options for exporting (also called unloading) data:

- Bulk exporting data from a table, a view, or the result of a SELECT statement into files in an internal or external stage
- Using custom code or a client to query a table or a view and then writing the results to one or more files in a local file system or any other target

These options are reviewed in detail below. Choose a method that suits your use case.

Exporting Data to a Stage

Use the COPY INTO location command to perform a bulk export of data (also called data unloading) from a table, a view, or the result of a SELECT statement into files in an internal or external stage (Amazon S3, Microsoft Azure or Google Cloud Storage).

Files exported to an internal stage may be downloaded to a local file system using SnowSQL and the GET command. Files exported to an external stage may be accessed/downloaded via interfaces provided by the respective platform (Amazon S3, Microsoft Azure or Google Cloud Storage).

At the time of this writing, Snowflake can export data as single-character delimited files (CSV, TSV, etc.) or as JSON. Exports can be optionally compressed and are always transparently encrypted when written to internal stages. Exports to external stages can be optionally encrypted as well.

Exporting Data to a Stage Examples

Sign up for a demonstration of Matillion ETL for Snowflake and an opportunity to speak to a Solution Architect about your unique use case.

Continued...

Exporting to an external stage (Amazon S3)

The following command is an example of exporting the result of a SELECT statement to an internal stage named my_stage, to files that reside in a folder named result and whose names are prefixed with data_, with a file format object named vsv, and using gzip compression. You can then load data from these files into other tables or download the files to your local disk using the SnowSQL GET command.

```
copy into @my_stage/result/data_ from (select * from orders)
file_format=(format_name='vsv' compression='gzip');
```

Exporting to an external stage (Microsoft Azure)

The following command is an example of exporting from a table named mytable to CSV-formatted files in Amazon S3 by specifying the credentials* for the desired S3 bucket.

```
copy into s3://mybucket/unload/ from mytable
 credentials = (aws_key_id='xxxx' aws_secret_key='xxxxx' aws_token='xxxxxx')
 file_format = csv;
```

The following command is an example of exporting from a table named mytable to CSV-formatted files in Microsoft Azure. The command specifies credentials for the targeted Blob storage bucket.

```
copy into azure://myaccount.blob.core.windows.net/unload/
 from mytable
 credentials = (aws_key_id='xxxx' aws_secret_key='xxxxx' aws_token='xxxxxx')
 file_format = csv;
```

NOTE
You can avoid specifying credentials by creating named external stages in advance using the CREATE STAGE command. To do this, you specify the external stage location and optionally the credentials required to access this location.

Exporting Data to a Local File System or Other Target

You may also use your favorite programming language or client to query a table or a view and then write the results to one or more files in a local file system or any other target. This approach may be slower than using the "COPY INTO <location>" command, because data needs to travel to the local machine running your code or client, which then writes to the file system. This approach may not noticeably affect exports for small tables but it will affect exports for larger tables. COPY INTO also benefits from using compression techniques when data is exported, which results in reduced network traffic and faster data movement.

You can also issue a "COPY INTO <location>" command using programming interfaces, download the files from the appropriate stage, and then access the data. This may be appropriate if you intend to download large data sets to local files.

Further Reading: Data Unloading

Storage and Compute Costs

This eBook has described methods and best practices for optimizing your usage of Snowflake to control costs. This chapter discusses how billing works, recaps methods for cost optimization, and describes how you can track and control costs. The following information is based on Snowflake's pricing guide.

Snowflake's unique architecture allows for a clear separation between your storage and compute resources. This allows Snowflake's pricing model to be much simpler and to include only two items:

- The cost of storage used
- The cost of compute resources (implemented as virtual warehouses) consumed

Snowflake credits are used to pay for the processing time used by each virtual warehouse.

Storage Costs

All customers are charged a monthly fee for the data they store in Snowflake. Storage cost is measured using the average amount of storage used per month for all customer data consumed or stored in Snowflake, after compression.

Note that features such as Time Travel and Fail-safe may increase costs associated with storage, because data is not immediately deleted but instead is held in reserve to support these features. Files in internal stages and features such as Snowflake Data Sharing, also known as the Data Sharehouse™, and cloning will also affect your storage costs.

Virtual Warehouse (Compute) Costs

Snowflake charges you only when your warehouse is in a "started" state. There is no charge when it is in a "suspended" state. This allows you to create multiple warehouse definitions and suspend them to prevent you from being billed for them. You must issue an ALTER WAREHOUSE RESUME command before you intend to use a suspended virtual warehouse.

Continued...

There is a linear relationship between the number of servers in a warehouse cluster and the number of credits the cluster consumes. Snowflake uses per-second billing (with a 60-second minimum each time the warehouse starts), so warehouses are billed only for the credits they actually consume. For more information, see Understanding Snowflake Credit and Storage.

You can profile a warehouse to understand its usage and credits spent. To profile a warehouse, use the WAREHOUSE_LOAD_HISTORY and WAREHOUSE_METERING_HISTORY functions. The information provided by these functions can tell you if scaling up a warehouse would benefit any existing loads. Conversely, you may also be able to identify underutilized warehouses and consolidate them, if appropriate. Read more about profiling here.

Once you understand how best to use your virtual warehouse for your data needs you can implement best practices for performance and cost efficiency.

OTHER FACTORS

The following factors influence the unit costs for the credits you use and the data storage you use:

- Whether you have a Snowflake On Demand account or a Capacity account
- The Region in which you create your Snowflake account
- The Snowflake Edition that your organization chooses

Most Snowflake customers use Snowflake On Demand initially to develop and test the application workload in order to gain real-world experience that enables them to estimate their monthly costs. When the application workload is understood, customers can then purchase an appropriately sized capacity.

NOTE: The Snowflake Edition your business chooses will impact billing. On Demand: Usage-based pricing with no long-term licensing requirements. Capacity: Discounted pricing based on an upfront capacity commitment.

Further Reading:

- Snowflake's Pricing Guide
- How Usage-Based Pricing Delivers a Budget-Friendly Cloud Data Warehouse

Best Practices

This chapter lists best practices for optimizing Snowflake. These best practices fit many use cases, but there are some circumstances where they may not apply. Your understanding of Snowflake's architecture should help you determine what is best for your particular use case.

IMPROVING LOAD PERFORMANCE

- Use bulk loading to get the data into tables in Snowflake.
- Consider splitting large data files so the load can be efficiently distributed across servers in a cluster.
- Delete from internal stages files that are no longer needed. This may improve performance in addition to saving on costs.
- Isolate load and transform jobs from queries to prevent resource contention. Dedicate separate warehouses for loading and querying operations to optimize performance for each.
- Leverage the scalable compute layer to do the bulk of the data processing.
- Consider using Snowpipe in micro-batching scenarios.
- Cost

IMPROVING QUERY PERFORMANCE

- Consider implementing clustering keys for large tables.
- Try to execute relatively homogeneous queries (size, complexity, data sets, etc.) on the same warehouse. Your query may benefit from cached results from a previous execution.
- Use separate warehouses for your queries and load tasks. This will facilitate targeted provisioning of warehouses and avoid any resource contention between dissimilar operations.
- Cost
- Performance

MANAGING A VIRTUAL WAREHOUSE

- Experiment with different warehouse sizes before deciding on the size that suits your requirements. Remember, you are not tied into a particular size.
- Use auto-suspend and auto-resume to save costs. Depending on your workloads, these may save costs as far as loads are concerned. This may not be as good for queries, though; see the next item below.
- Understand the impact of caching on queries. Warehouses that use auto-suspend and auto-resume may not benefit from caching. Consider the trade-off between saving credits by suspending a warehouse versus maintaining the cache of data from previous queries for quicker response times.
- Warehouse suspension and resumption takes time, which is noticeable for larger warehouses. Keep the warehouse running if you need an immediate response.
- Cost
- Performance

Conclusion

We hope you enjoyed this eBook and that you have found some helpful tips on how to make the most of your Snowflake database. Implementing the best practices and optimizations described in this eBook should help you enhance big data analytics performance and reduce your Snowflake costs.

With Snowflake, you can spend fewer resources on managing database overhead and focus on what's really important: answering your organization's most pressing business questions.

About Matillion

Matillion is an industry-leading data transformation solution for cloud data warehouses. Delivering a true end-to-end data transformation solution (not just data prep or movement from one location to another), Matillion provides an instant-on experience to get you up and running in just a few clicks, a pay-as-you-go billing model to cut out lengthy procurement processes, and an intuitive user interface to minimize technical pain and speed up time to results. Matillion is available globally for Snowflake on AWS Marketplace, Microsoft Azure Marketplace and Google Cloud Platform.

Find out more at **www.matillion.com**

© 2019 Matillion. All rights reserved


MATILLION
snowflake®