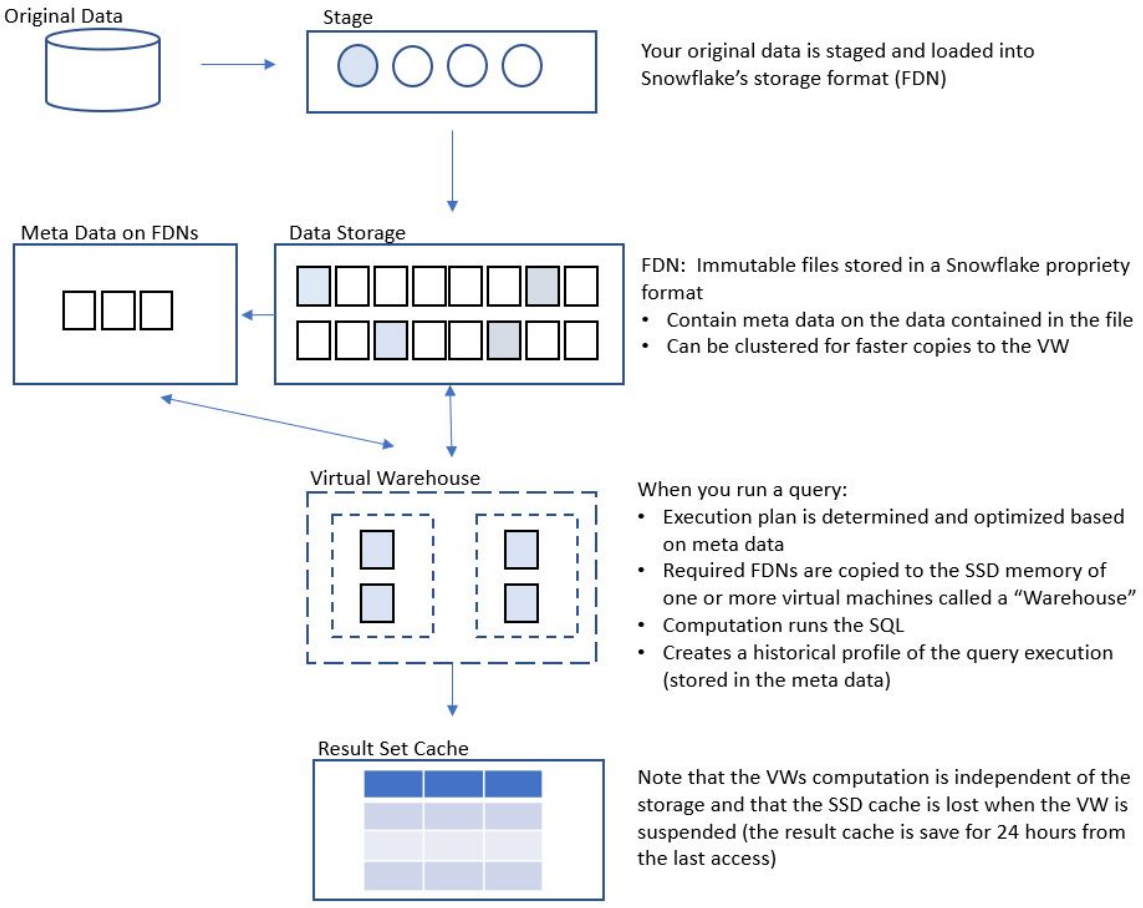
Tips for Snowflake Performance Tuning and Cost Savings

One of the benefits to Aptitive's partnership with Snowflake is access to advanced trainings and certifications. Combined with our Snowflake project work, these trainings not only help us to be Snowflake experts, but also to find new opportunities to help Snowflake's solution work better for our clients. The most recent training for data engineers, helped us to identify some important tactics for solving one of client's biggest concerns: *How do I optimize for cost?* Here is a list of actions that you should take to make sure you are not overspending on your Snowflake computation or storage.

First, for context, here is an extremely simplified diagram of how Snowflake is functioning in the background:



Since the most expensive part of any Snowflake deployment is compute, we have identified a few dozen tactics to store data strategically for efficient reads, write supercharged SQL scripts, and balance your performance vs cost.

Loading

Although very different than storing data on traditional disk, there are many benefits to loading Snowflake data strategically.

1. Sort on ingestion: Data is automatically partitioned in SF on natural ingestion order.
    - Sorting an S3 bucket (using something like syncsort) before bulk load via copy could be way faster than insert with an order by
2. CSV (Gzipped) is the best format for loading to SF (2-3x faster than Parquet or ORC)
3. Use COPY INTO instead of INSERT because it utilizes the more efficient bulk loading processes.

Sizing

Take advantage of the native cloud ability to scale, create, and optimize your compute resources.

4. Scale up or out appropriately.
    - As seen above, when you run a query, Snowflake will:
        - Find required FDN files
        - Pull files down into SSD VMs (Note: if >160 GB for AWS or >400 GB for Azure, will spill over to remote IO)
        - Performs compute
        - Files will stay on VM until DW is suspended
    - SO, 1 big query = increase size of Data Warehouse
    - Lots of small queries = queries are queuing = increase # of DWs or # of clusters (if enterprise you can enable multi-cluster)
5. Turn your VW on and off for certain workloads
    - Turn on for batch/ then immediately turn off (no reason to wait for auto-suspend)
    - Use Auto resume when it makes sense
6. Control query processing and concurrency with parameters
    - Max_concurrency_level
    - Statement queued timeout in seconds
    - Statement timeout in seconds
7. Use warehouse monitoring to size and limit cost PER WORKLOAD (not per DB → this is a shift from the on-prem mentality)
    - If your workload is queuing then add more clusters
    - If your workload is slow with no queuing then size up

Data Modeling

Often overlooked, organizing your information into a mature data model will allow for high-performance SQL scripting better caching potential. Shameless plug, this is Aptitive's bread and butter. Please reach out for a free working session to discuss data modeling for your company.

8. Do a data model for analytics
    - Star Schema, 3NF, Data vault are optimal for SF
    - Snowflake is NOT ideal for OLTP workloads
9. Bake your constraints into design because SF DOES NOT enforce them
    - Build queries to check for violations
10. Build a process to alert you of loading issues (use an ETL framework)
    - Information_Schema.load_history
    - Aptitive ETL Toolkit

Tracking Usage

Snowflake preserves a massive amount of usage data for analysis. At the very least, it allows you to see which workflows are the most expensive:

11. Account Usage Views (eg warehouse_metering_history) for tracking history and performance and cost
12. Don't use AccountAdmin or Public (except for looking at costs)…create securable objects with the correct role and integrate new roles into the existing hierarchy
    - Create roles by Business functions
13. Use Resource monitors
    - Cut off DW when you hit credit amount
        i. One resource monitor per DW
        ii. Enable notifications

Performance tuning

The history profiler is the primary tool for observe poorly written queries and make the appropriate changes.

14. Use history profiler to optimize queries
    - Goal is to put the most expensive node in the bottom right hand corner of profiler diagram
    - SYSTEM$CLUSTERING_DEPTH shows how effective the partitions are, the smaller the average depth, the better clustered the table is with regards to the specified columns
        i. tip: You can add an new automatic reclustering service, but I don't think it is worth the money right now
15. Analyze bytes Scanned: Remote VS Cache (red/green)
    - Make your Bytes Scanned column Green most of the time else create cluster key
16. Make the ratio of partitions scanned/ partition use as small as possible by pruning

SQL coding

The number one issue driving costs in a Snowflake deployment is poorly written code! Resist the tendency to just increase the power (and therefore the cost) and focus some time improving your SQL scripts.

17. Drop temporary and transient tables when done using
18. Don't use "CREATE TABLE AS", SF hates trunc and reloads for time travel issues. Instead, use "CREATE OR REPLACE"
    - Again, Use COPY INTO *not* INSERT INTO
    - Use staging tables to manage transformation of imported data
    - Validate the data BEFORE loading into SF target tables
19. Use ANSI Joins because they are better for the optimizer
    - Use "JOIN ON a.id = b.id" format
    - NOT the "WHERE a.id=b.id"
20. Use "WITH" clauses for windowing instead of temp tables or sub-selects
21. Don't use ORDER BY. Sorting is very expensive!
    - Use integers over strings if you must order
22. Don't handle duplicate data using DISTINCT or GROUP BY

Storing

Finally, setup the Snowflake deployment to work well in your entire data ecosystem.

23. Locate your S3 buckets in the same geographic region
24. Set up the buckets to match how the files are coming across (ie by date or application)
25. Keep files between 60-100 MB to take advantage of parallelism
26. Don't use materialized views except in specific use cases (eg pre aggregating)

Snowflake is shifting the paradigm when it comes to data warehousing in the cloud. However, by fundamentally processing data differently than other solutions, Snowflake has a whole new set of challenges for implementation.