



HOW SNOWFLAKE AUTOMATES PERFORMANCE IN A MODERN CLOUD DATA WAREHOUSE

TABLE OF CONTENTS

- 3** Break with the Old
- 3** Architecture Makes All the Difference
 - 4** Concurrency and workload isolation
 - 4** Automatic micro-partitioning
 - 4** Self-healing
- 5** Maximize Performance Automation
 - 5** Isolate workloads with virtual warehouses
 - 6** Scale automatically in multi-cluster virtual warehouses
 - 7** Create automatic clustering with self-organizing cloud storage
 - 7** How to use the automatic clustering in Snowflake
 - 8** How to use materialized views and automatic maintenance
- 9** Conclusion

BREAK WITH THE OLD

To optimize performance in most database management systems, data administrators must do continual manual maintenance, which increases overhead and leaves little time for actual analytics. Rather than settling for limitations that require manual workarounds, you can use modern cloud data warehousing to provide essentially limitless compute power, complete workload isolation, and the simplicity, flexibility, and scalability needed for fast, built-in performance.

In this white paper, you'll learn how adopting a cloud-built data warehouse provides true workload isolation with instant, infinite elasticity that enables you to deliver significant performance improvements over traditional data warehouses. With a multi-cluster, shared data architecture, it's easy to take advantage of immediately available resources in the cloud, address common performance challenges, and maximize performance to give users the best analytics experience possible.

ARCHITECTURE MAKES ALL THE DIFFERENCE

The overarching performance challenge with traditional data warehouses comes down to one thing: resource contention. Single-cluster systems—both in shared-disk and shared-nothing architectures—require all users in the data warehouse to share a limited set of resources. Too many concurrent users or workloads can lead to significant performance degradation and queueing. No matter how many nodes or resources you add to a single cluster, you will eventually see diminishing returns and bottlenecks.

In contrast, Snowflake provides a patented multi-cluster, shared data architecture and intelligent use of readily available cloud resources to allow essentially limitless concurrency without impacting performance while accessing a single copy of the data (Figure 1). With the cloud's instant and near-infinite elasticity, compute resources can scale out automatically and provide users with consistent performance regardless of the number of queries at any given moment. Compute resources can also scale up instantly for faster performance.

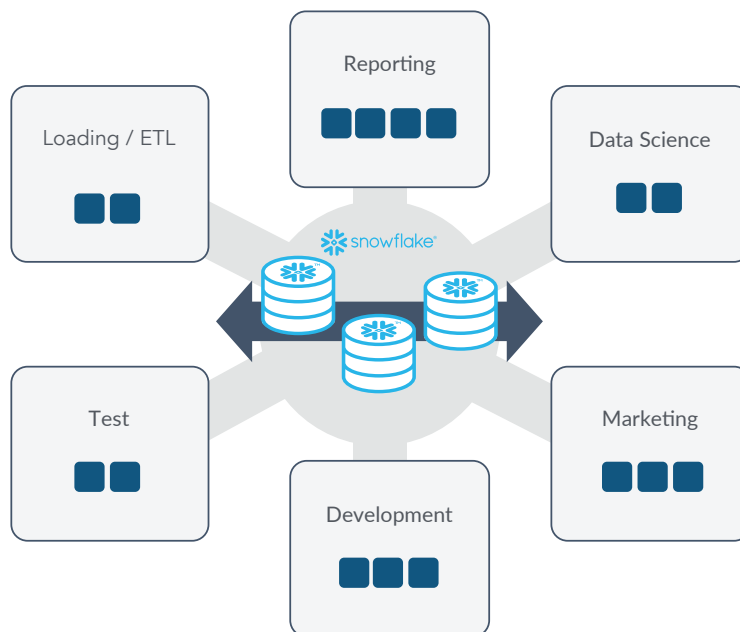


FIGURE 1. SNOWFLAKE'S ARCHITECTURE ENABLES LIMITLESS CONCURRENCY WITHOUT IMPACTING PERFORMANCE
Independent compute clusters are dedicated to each of your workloads or user groups while providing access to the single source of truth. They can each grow, shrink, scale in or out, and turn on or off instantly, as needed.

Concurrency and workload isolation

In traditional data warehouses, administrators can add nodes to clusters to increase scale, but the architecture cannot support infinite scaling. At some point, the concurrency and performance benefits of additional nodes reach a limit as issues such as data skew and CPU context switching become more pronounced. As a result, performance suffers, forcing users to wait for results. Careful planning and a solid understanding of today's workload can alleviate these issues temporarily. However, when query patterns change, new applications and user groups are added to the system, and data volume multiplies, problems can also multiply.

Database administrators typically divide a shared cluster's resources among user groups or workloads. If one workload needs more resources, the administrator must take resources from another workload to compensate. Some users might have to wait until nights or weekends to get the resources they want. Ideally, it wouldn't be necessary to pit different workloads against each other.

Because resources are not limited or fixed in Snowflake, every aspect of the service can be elastically duplicated, expanded, contracted, or even turned off entirely. One of the most important and visible components for users is the virtual warehouse. A virtual warehouse is an MPP cluster of virtual machines that execute SQL queries in a parallel fashion. You can create as many virtual warehouses as you want in your Snowflake account and you can size, resize, or turn on or off each one independently of the others. And because the storage layer is independent of the compute layer, all virtual warehouses can share the entire data set. Any individual, user group, application, or automated workload can take advantage of **dedicated, isolated compute resources** while operating on a **single source of truth** and never worrying about impacting others' query performance.

Automatic micro-partitioning

In traditional databases, good performance requires some combination of indexes, partition keys, shard keys, sort keys, manual query optimization, optimizer hints, and so on. Time that database administrators spend on these activities comes at the expense of

performing higher-value work, such as getting access to new data sets or performing new types of analysis. It is generally difficult to pick an optimal scheme for workloads with different access patterns. Making a wrong decision can be costly. Resizing a cluster takes significant time as administrators must shuffle partitions or repartition data to accommodate the new number of nodes. Planning ahead and deeply understanding your workload is crucial, as changes can be expensive and impact every other process running on your cluster.

In contrast, Snowflake partitions data **automatically based on the natural ingestion order** and stores partitions in a central storage layer that all compute nodes can access. Snowflake automatically collects statistics and updates them with every executed DML statement. Any maintenance on these partitions can be done as an automatic background process **using resources separate from the ones running your workloads**. Snowflake keeps partitions small to enable more granular pruning and extra flexibility during query processing. During a query, Snowflake automatically picks the optimal distribution method for just the partitions needed based on the current size of your virtual warehouse. This makes Snowflake inherently more flexible and adaptive than traditional systems, while reducing the risk of hotspots.

Self-healing

When different pieces of a traditional system start to break down, whether due to software or hardware defects or just an intensive workload, the static nature of these systems prevents them from healing without manual intervention. You can deploy solutions for high availability and disaster recovery, create data marts for offloading, and so on, but that takes more resources and engineering time to build and maintain. As an administrator, you must be prepared to analyze your workloads and tweak any of the hundreds or thousands of controls that might be available to you.

In contrast, Snowflake builds high availability and the ability to self-tune and self-heal into every layer of the system. For example, as data is written to a table in Snowflake, it is synchronously written to highly durable cloud storage in three different data centers. If your compute cluster in one data center starts losing machines or if the entire data center goes

down, Snowflake can instantly provision a cluster in another data center that still has access to all of your data. This capability extends to every layer of the Snowflake service. As queries run on a given cluster, Snowflake caches data locally and shuffles data around if it detects hotspots.

MAXIMIZE PERFORMANCE AUTOMATION

A cloud-built data warehouse such as Snowflake offers a significantly improved framework for performance optimization by taking advantage of the elasticity and available compute resources the cloud provides. Even better, Snowflake enables a level of performance automation never seen before. Because Snowflake can leverage information about actual workloads and query access patterns in near real time, it can perform corresponding optimization strategies transparent to the end user.

Although Snowflake ensures administrators perform very few optimization activities to deliver first-rate performance, knowing how to solve a few common performance challenges lets you deliver the best experience possible for your users.

Isolate workloads with virtual warehouses

Often, too many different use cases or groups accessing the data warehouse prevent consistent performance. For example, even with only a few concurrent queries, one compute-intensive query from the operations team could significantly impact the performance of a query from the marketing team. Dedicated and separate virtual warehouses (compute clusters) for each query and workload in Snowflake solve this contention problem.

This approach enables lower compute usage by allocating each group to the right-sized compute resource. It also eliminates contention. For instance, if the marketing and operations team were sharing a large virtual warehouse (eight credits per hour), it's possible the operations teams could instead serve their needs with a medium virtual warehouse (four credits per hour) while the marketing team could achieve their goals with a small virtual warehouse (two credits per hour). This strategy would save two credits per hour and provide unhindered performance for all queries and workloads.

If you ever need to run a heavy, one-time analysis or want to test a new workload you might not keep running, you can run it on a separate right-sized warehouse so that it doesn't impact other users. For a heavy workload where there is a fixed amount of work, it often makes sense to run it on the biggest warehouse size. It will end up running faster and cost the same as if you did it on a smaller warehouse that would take more time.

BEST PRACTICE:

Enable all virtual warehouses to automatically suspend when idle and automatically resume when queried, so that you don't pay for them while your users aren't running queries. Compare that to traditional data warehouses, which must run at full capacity for 24 hours per day, seven days per week, whether or not you are using them.

Scale automatically in multi-cluster virtual warehouses

In addition to regular virtual warehouses, users can leverage multi-cluster virtual warehouses to automatically scale compute resources for workloads with varying levels of concurrency. For example, the marketing group uses a BI dashboard with metrics that the entire team wants to access on Monday morning. As marketing team members arrive at work and open the dashboard, dozens of concurrent queries would occur.

If all of those queries run on a single cluster, then it is natural to expect a slowdown as they compete for resources. In most systems, queries would start to queue and wait for resources, creating a backlog and delays.

Snowflake has a better solution. When more queries are issued than can execute on a single cluster, the virtual warehouse will automatically start a second, identical cluster and begin automatically load-balancing queries across the two. If two clusters are still not enough for the level of concurrency, the virtual warehouse will start a third cluster, and so on until you reach the maximum number of clusters you configured for the warehouse. As the workload

subsides, one cluster at a time will be shut off so that you pay only for the resources you need in a given moment. This provides **consistent** performance regardless of the number of queries.

BEST PRACTICE:

Make sure a virtual warehouse's size can handle individual queries with good performance. Adjust the warehouse's minimum and maximum number of clusters based on the minimum and maximum concurrent throughput you expect for the workload. Select your warehouse's size, such as small, medium or large, to provide adequate performance for each individual query that runs. Add more clusters to the warehouse to increase the number of queries that can run at once. A given warehouse size can run individual queries twice as fast as the size below it. Each additional cluster allows the warehouse to run more queries in parallel to increase concurrency.

Table: t1

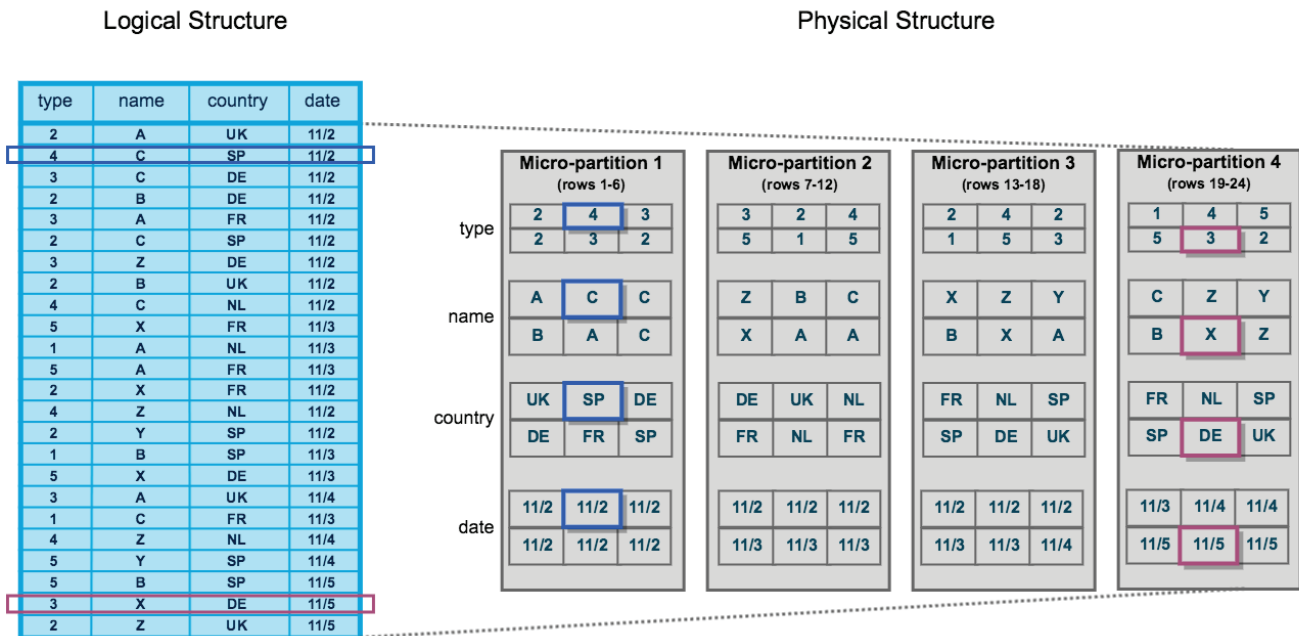


FIGURE 2. CLUSTERED MICRO-PARTITIONS

The table shown consists of 24 rows stored across four naturally clustered micro-partitions, with the rows divided equally between each micro-partition. The data is sorted and stored by column, which enables Snowflake to prune micro-partitions that are not needed for the query, and then prune by column within the remaining micro-partitions.

Create automatic clustering with self-organizing cloud storage

If you still need to improve performance, look at the query profiles for your long-running queries. See where they spend their time and what percentage of your tables they are scanning. Especially for large tables, you can filter the data so that your queries scan only portions of the table. Usually data is loaded into big tables according to date and time it arrives. As a result, any filter on date or time will lend itself to partition pruning on the table and provide good performance.

However, if you can't always filter the table on date or time and have other selective filters in your queries, you can consider adding a cluster key to the table. Cluster keys tell Snowflake if data in a table should be organized in a different order than the natural ingestion order (Figure 2). Although cluster keys are not necessary for most tables, you can add them to your biggest tables to enable pruning based on your different query predicates.

How to use the automatic clustering in Snowflake

Snowflake's *automatic clustering* removes the burden of manually reclustered data. As Snowflake performs continuous and incremental reclustered tasks in the background, a database administrator no longer needs to worry about additional tuning exercises such as choosing the right warehouse size. Automatic clustering is server-less and transparent to the user, and Snowflake manages the needed compute resources entirely. The user can resume, suspend, or modify the keys used by automatic clustering for a table at any time.

While automatic clustering does consume credits, Snowflake bills only for the resources needed to perform the reclustered tasks, which are measured and billed to the second. The billed amount is always displayed as a separate line item in the Snowflake billing UI, and the user has the option to monitor programmatically the amount of

work automatic clustering has performed during a time period.

For general guidance:

- **Adding clustered tables:** If a new clustered table is registered, the user might notice a burst of reclustering activities until a good clustering state has been achieved. The incremental ongoing maintenance will generally require much fewer resources.
- **Workloads churning clustered tables:** Large data loads or DML operations that modify a large amount of data may trigger automated reclustering. Snowflake uses efficient algorithms to determine the clustering quality and performs necessary reclustering and other optimization activities.
- **Changing clustering keys:** Changing the clustering key may result in another spike in reclustering activities as the table data is reorganized.

For more information about cluster key selection, please visit the [Snowflake documentation](#).

How to use materialized views and automatic maintenance

Materialized views improve performance of common and repeated query patterns for resource-intensive queries. Snowflake's materialized views offer the performance benefits of traditional materialized views while providing access to **always current** data.

As data is added or modified in the referenced base table, Snowflake updates materialized views automatically and transparently as a background service. Once created, materialized views require no manual effort to maintain them. Snowflake manages and scales the resources used for maintenance behind the scenes, so your other workloads can continue to run with no contention.

What's more, Snowflake's materialized views deliver an accurate view of the underlying data, even if it hasn't yet been updated by the maintenance service. Snowflake automatically considers any recently added, deleted, or modified data from the base table when it performs a combined query execution across both the dependent materialized view and the base table.

```

1 create table pipeline_pressures (
2     segment_id bigint,
3     pressure_psi float, -- pressure in Pounds per Square Inch
4     measurement_timestamp timestamp
5 );
6
7 create materialized view pipeline_pressures_daily_rollup
8 as select
9     segment_id,
10    measurement_timestamp::date as measurements_date,
11    count(1) as measurements_count,
12    min(pressure_psi) as min_psi,
13    max(pressure_psi) as max_psi
14 from pipeline_pressures
15 group by 1,2;
16
17 alter materialized view pipeline_pressures_daily_rollup
18 cluster by (measurements_date);

```

FIGURE 3. CREATING A CLUSTERED MATERIALIZED VIEW
This example creates a clustered materialized view.

When users combine automatic clustering with materialized views (Figure 3), they can create different representations of the same data by using clustering keys for each materialized view. Both services, automatic clustering and automatic maintenance, keep data in each materialized view optimally clustered while reflecting any changes on the base table. In this way, users optimize the underlying data layout for entirely different query access patterns, without performing any manual tuning activities.

Additionally, users can combine materialized views with functions popular for processing and extracting semi-structured data (such as the flatten function) or with approximate or statistical queries (for example, sampling).

A user can resume and suspend automatic maintenance for each materialized view. However, suspending automatic maintenance for a long period of time results in more work when the materialized view is resumed.

For billing, the same principle applies: Snowflake charges only for the resources used by the automatic maintenance to keep the materialized views up to date and bills usage to the second.

BEST PRACTICE:

Be strategic when creating a materialized view and measure the performance benefits you get from it before deciding whether to keep it. The best candidates for materialized views are particularly expensive aggregations, projections, and selections that must be run frequently. If they are inexpensive to run on the base table or if they are run infrequently, then the cost of materialized view maintenance will not be worth it.

CONCLUSION

With the arrival of the cloud-built data warehouse, performance optimization becomes a challenge of the past. Using Snowflake, everyone benefits from performance automation with very little manual effort or maintenance. Best of all, Snowflake is available to help with performance automation strategies to ensure every user receives all the benefits of exceptional performance. Snowflake will continue to invest in making your life easier when it comes to analyzing your data.

ABOUT SNOWFLAKE

Snowflake is the only data warehouse built for the cloud, enabling the data-driven enterprise with instant elasticity, secure data sharing, and per-second pricing across multiple clouds. Snowflake combines the power of data warehousing, the flexibility of big data platforms, and the elasticity of the cloud at a fraction of the cost of traditional solutions. Snowflake: Your data, no limits. Find out more at [snowflake.com](https://www.snowflake.com)

