



# A Detailed View Inside Snowflake

THE DATA WAREHOUSE BUILT FOR THE CLOUD

# Contents

IMAGINING A FRESH APPROACH TO DATA WAREHOUSING >> 4

THE LIMITS OF TRADITIONAL DATA WAREHOUSES AND NOSQL ALTERNATIVES >> 5

SNOWFLAKE: DATA WAREHOUSE BUILT FOR THE CLOUD >> 7

HOW SNOWFLAKE DELIVERS ON THE PROMISE OF THE CLOUD DATA WAREHOUSE >> 9

ENABLE EASE OF USE >> 12

PAY AS YOU GO >> 13

## THE NEED FOR CHANGE

Legacy data warehouses are based on technology that is, at its core, decades old. They were designed in a time when data was simpler, and the number of people in an organization with the need or desire to access the database were few. As analytics has become a company-wide practice, and a larger volume of more diverse data is collected, the data warehouse has become the biggest roadblock that people are facing in their path to insight. To meet the demands and opportunities of today and tomorrow, data warehouses will need to fundamentally change.

»» **Data is becoming more diverse.** It used to be that data came primarily from internal sources (e.g. transactional, ERP, and CRM systems) in structured forms at a predictable rate and volume. Today, in addition to traditional sources, data is being generated by a diverse and rapidly changing set of sources, including application logs, web interactions, mobile devices, and more. That data frequently arrives in flexible semi-structured formats such as JSON or Avro, at highly variable rates and volumes.

»» **Data is being used differently.** Data used to flow through complex ETL pipelines into a data warehouse, where reporting queries ran periodically to update fixed dashboards and reports. That process often took days. Today, a wide array of analysts need to explore and experiment with data as quickly as possible, without knowing in advance where they might find value in it. A growing number of applications need immediate access to data in order to support new and existing business processes.

»» **Technology has evolved.** There are technologies available today, like the cloud, that were not even conceived of when conventional data warehouses were designed. As such, they weren't designed to take advantage of the unlimited scale and convenience of the cloud.

»» **Purchasing has evolved.** With the diverse and ever changing workload of the modern data warehouse, many organizations would prefer to pay for their data infrastructure and software as a subscription, instead of a permanent (and large) one-time capital outlay.

---

“Today’s data warehouses are based on technology that is decades old. To meet the demands and opportunities of today, data warehouses have to fundamentally change.”

— Jeff Shukis, VP Engineering and Tech Ops, VoiceBase

---

Traditional data warehouses have been adequate for years, but their architectural baggage is becoming more evident as they fail to evolve to changing needs. They are often quite expensive, as well.

At the same time, newer “big data” offerings and noSQL systems such as Hadoop are failing to provide a better alternative. They can be useful tools for data transformation and data science, but they weren’t designed for data warehousing. They require difficult-to-find skillsets, are not fully compatible with the existing ecosystem of SQL-based tools, and fail to deliver interactive performance. What’s more, to deliver the capabilities required to be a data warehouse even partially, they need to be paired with other compute and processing tools.

## IMAGINING A FRESH APPROACH TO DATA WAREHOUSING

These limitations can’t be fixed with haphazard feature updates; they are fundamental to the inadequate architecture of traditional data warehouses and big data solutions. To address their shortcomings, a complete redesign and reimagining of data warehouse architecture and technology is necessary.

If we were to start over, unencumbered by the accumulated baggage of data warehousing history, what would we build? The ideal data warehouse would combine the strengths of data warehousing—performance, security, and a broad ecosystem—with the flexibility and scalability of “big data” systems.

The screenshot displays the Snowflake web interface. At the top, there are navigation tabs for Databases, Warehouses, Worksheet (active), and History. The user is identified as ROSS SYSADMIN. Below the navigation, there's a 'Worksheet 1' header. The main area is split into two panes: a SQL editor on the left and a results table on the right.

**SQL Editor Content:**

```

1 // 1. Create a table to host the JSON files in my
2 CREATE OR REPLACE TABLE TEMPORARY_LOADING
3 (V variant);
4
5 COPY INTO TEMPORARY_LOADING from @WEATHERSTAGE;
6
7 select * from TEMPORARY_LOADING;
8
9 //2. Create a relational table, then flatten and
10 CREATE OR REPLACE TABLE DAILY_WEATHER_HISTORY
11 ( CITY_ID INT,
12 T timestamp,
13 MAIN variant,
14 WEATHER variant);
15
16 INSERT INTO DAILY_WEATHER_HISTORY
17 SELECT t.V:city_id,
18 h.VALUE:dt::timestamp,
19 h.VALUE:main,
20 w.VALUE
21 FROM TEMPORARY_LOADING t,
22 LATERAL FLATTEN(input => t.V:list) h,
23 LATERAL FLATTEN(input => h.VALUE:weather) w;
24
25 select * from DAILY_WEATHER_HISTORY;
26
27 //3. Create a view that will parse out the high s
28 CREATE OR REPLACE VIEW DAILY_SUMMARY AS
29 (SELECT
30 DATE_TRUNC(DAY,T) DAY,
31 MAX(WEATHER:main) TYPE,

```

**Query Execution Log:**

Status	Query ID	SQL Text	Warehouse	Clust...	Size	Start Time	End Time	Total Duration
✓	14d0c7b0-d...	select * from DAILY_W...	TABLEAU_WH	1	X-Large	11:56:41 AM	11:56:47 AM	5.6s
✓	517a02f2-3...	INSERT INTO DAILY_...	TABLEAU_WH	1	X-Large	11:56:33 AM	11:56:40 AM	6.5s
✓	48f16be1-c...	//2. Create a relational ...	TABLEAU_WH			11:56:31 AM	11:56:31 AM	79ms
✓	c6d955d8-4...	select * from TEMPOR...	TABLEAU_WH	1	X-Large	11:55:48 AM	11:55:52 AM	4.2s
✓	c2c9b3ad-8...	COPY INTO TEMPOR...	TABLEAU_WH	1	X-Large	11:55:14 AM	11:55:32 AM	17s
✓	ea893d2c-f...	// 1. Create a table to h...	TABLEAU_WH			11:55:12 AM	11:55:12 AM	88ms
✗	769e5ec6-c...	select * from TEMPOR...	TABLEAU_WH			11:54:54 AM	11:54:54 AM	35ms
✓	bfb2e0f0-cc...	//3. Create a view that ...	TABLEAU_WH			10:48:22 AM	10:48:22 AM	517ms

**Results Table:**

row#	CITY_ID	T	MAIN	WEATHER
1	1258993	2016-01-01 00:34:10.000	{"grnd_level": 1006.07, "humidi...	{"description": "Sky is Clear", "l...
2	1258993	2016-01-01 02:30:24.000	{"grnd_level": 1006, "humidity":...	{"description": "sky is clear", "ic...
3	1258993	2016-01-01 03:05:19.000	{"grnd_level": 1006, "humidity":...	{"description": "sky is clear", "ic...
4	1258993	2016-01-01 04:31:13.000	{"grnd_level": 1006, "humidity":...	{"description": "sky is clear", "ic...
5	1258993	2016-01-01 05:05:11.000	{"grnd_level": 1006, "humidity":...	{"description": "sky is clear", "ic...

Fig. 1: Snowflake can store any scale of diverse data at a low cost.

Such a data warehouse would be:

- **Able to store any type of business data:** Natively handle diverse types of data without requiring complex transformations before loading that data into the data warehouse.
- **Instantly scalable for flexible performance and concurrency:** Able to infinitely scale up and instantly scale down at any time without disruption. It would also be able to scale out to as many different use cases as needed without disruption. It goes without saying that complete elasticity is difficult to accomplish without an unlimited compute resource like the cloud affords.
- **A true service:** Management and infrastructure would be automatically managed by the warehouse so that users could focus on getting value from their data.
- **A seamless fit with existing skills and tools:** The data community has been myopically focused on supporting tools for a small number of data scientists, without addressing the huge community of people and tools that understand standard SQL. Full support for standard SQL makes it possible to offer a better engine for those users without the need for new expertise, programming paradigms, and training.
- **A flexible subscription and service:** Businesses should be able to pay for all of their services and infrastructure as a service, and data warehouses are no different. The flexibility of the subscription model allows for the ebb and flow of business needs, and more elegantly supports the rapid growth and capital expenditure models of modern organizations.

- **Able to facilitate seamless data sharing:** With organizations looking to share data both inside and outside of their walls, the data warehouse of the future would enable support for seamless data sharing.

Unfortunately, traditional data warehouses and the noSQL systems that are frequently promoted as their complement or even replacement - are fundamentally unable to fulfill all these requirements.

## THE LIMITS OF TRADITIONAL DATA WAREHOUSES AND NOSQL ALTERNATIVES

Traditional data warehouses are fundamentally unable to deliver this vision. Data warehouse appliances with fixed configurations are certainly the most limited, but even software-only products cannot be truly elastic. Those limitations are driven by fundamental flaws in the two dominant scalable database architectures in traditional databases: shared-disk and shared-nothing.

### The Shared-Disk Architecture and its Limitations

The shared-disk architecture was the first approach to emerge for scaling beyond the single-node SMP architectures of early systems. It is designed to scale processing beyond a single server while keeping data in a central location. In a shared-disk system, all of the data is stored on a storage device that is accessible from all of the nodes in the database cluster. Any change in the data is updated and reflected in the single storage location. Shared-disk architectures are attractive for their simplicity of data management: all processing nodes in the database cluster have direct access to all data, and that data is consistent because all modifications to the data are written to the shared disk. However, the scalability of this architecture is severely limited because

even a modest number of concurrent queries will overwhelm the storage device and the network to it, forcing processing to slow down for data to be returned from the shared disk. Additional compute nodes only exacerbate the overloaded shared disk. Further, complicated on-disk locking mechanisms are needed to ensure data consistency across the cluster.

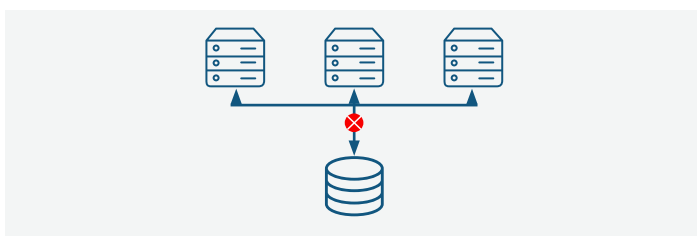


Fig. 1: Shared disk architecture is limited by the performance of the disk

## Shared-Nothing Architecture and its Limitations

Shared-nothing databases arose as a solution to the bottlenecks of the shared-disk architecture. The shared-nothing architecture scales processing and compute together by distributing different subsets of data across all of the processing nodes in the system, eliminating the bottleneck of communication with a shared disk. Designed in an era where bandwidth and network latency to storage was a key bottleneck, the shared-nothing architecture took advantage of inexpensive local disk, moving data storage close to compute.

However, the shared-nothing architecture has its own limitations, which have become increasingly apparent as technology and data analytics have advanced.

For one, the shared-nothing architecture has performance bottlenecks of its own. As the cluster is scaled to more and more nodes, the fact that data is distributed across the cluster requires shuffling data between nodes. That shuffling adds overhead that reduces performance and makes performance heavily dependent on how data is distributed across the nodes in the system.

The challenges of optimizing data distribution in a shared-nothing system have only grown as workloads have become more dynamic and varied. Distribution of data across compute nodes is typically done through static assignment--data is distributed at the time it is loaded by either a user-specified distribution key or by a default algorithm. Changing the data distribution typically requires completely redistributing data across the cluster or even unloading and reloading data. This is a slow and disruptive operation, often requiring queries to pause and blocking queries that modify data.

Further, shared-nothing architectures make it very difficult to select the right balance of storage and compute. Because the cluster must be sized to house all data, compute resources may be more than needed for actual queries or may be insufficient for the queries run on the system. Because of the time required to resize the cluster (if even possible to do so), organizations frequently overprovision these clusters, resulting in wasted resources and spend.

## Limitations of noSQL

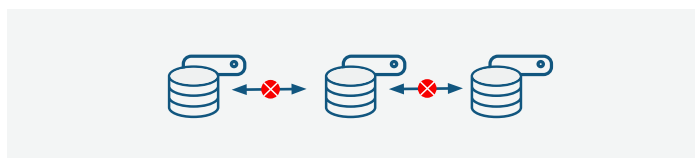


Fig. 2: Shared nothing architecture is limited by the need to distribute and query data across nodes

The limited flexibility of traditional data warehouse architectures and their inability to scale cost-effectively to handle the massive data volumes of the modern business helped lead to interest in emerging noSQL alternatives like Hadoop. The ability of noSQL solutions to store non-relational data without first requiring transformation, leverage inexpensive commodity servers for scaling to large data volumes, and support diverse custom programming led organizations to experiment with noSQL solutions

in a variety of use cases. Many wondered whether noSQL solutions could even replace the data warehouse.

However, as organizations have looked more closely at these solutions, it has become clear that they have limitations of their own that make them unable to replace the data warehouse. Most noSQL solutions, including Hadoop, rely on the same shared-nothing architecture that underlies traditional data warehouses. As a result, key limitations of shared-nothing architectures also hinder these solutions—data frequently needs to be shuffled among nodes, compute cannot be sized independently of storage, and clusters often need to be overprovisioned.

Not only that, but noSQL systems generally don't fully support ANSI SQL and are extremely complex to manage. As a result of their inefficiency, they also suffer from poor performance and struggle to support higher levels of concurrency. In short, Hadoop and noSQL tools are fundamentally poor at analytics.

## SNOWFLAKE: DATA WAREHOUSE BUILT FOR THE CLOUD

At Snowflake, as we considered the limitations of existing systems, we realized that the cloud is the perfect foundation to build this ideal data warehouse. The cloud offers near-infinite resources in a wide array of configurations, available at any time, and you only pay for what you use. Public cloud offerings have matured such that they cannot support a large and growing set of enterprise computing needs, often delivering higher data durability and overall availability than private datacenters, all without the upfront capital costs.

Although a small number of data warehouses are marketing themselves as “cloud” solutions, they weren't designed for the cloud. These offerings are either managed service offerings of existing on-premises products, or simply an installation of existing software in a public cloud infrastructure. Conversely, there are cloud vendors offering “cloud data warehouses” that were never intended to be data warehouses in the first place, and lack full support for basic features like ANSI-SQL compatibility.

Snowflake was founded by a team with deep experience in data warehousing. Guided by their experiences and frustrations with existing systems, our team built a completely new data warehouse designed to deliver dynamic infrastructure, performance, and flexibility at a fraction of the cost. Most importantly, they built Snowflake from scratch for the cloud rather than starting with existing software like Postgres or Hadoop.

The Snowflake solution? First of all, Snowflake was built in the cloud and for the cloud to offer completely unlimited storage and compute. Snowflake is a massively parallel processing (MPP) database that is fully relational, ACID compliant, and processes standard SQL natively without translation or simulation. It was designed as a software service that can take full advantage of cloud infrastructure, while retaining the positive attributes of existing solutions.

“With Snowflake's speed, we can explore this information map at the speed of thought, and move from data, to information, to a decision, 10 times faster.”

— Chris Frederick, Business Intelligence Manager  
University of Notre Dame

## A new architecture: Multi-cluster, shared data

Snowflake's novel design physically separates but logically integrates storage, compute and services like security and metadata; we call it multi-cluster, shared data and it consists of 3 components:

- **Storage:** the persistent storage layer for data stored in Snowflake
- **Compute:** a collection of independent compute resources that execute data processing tasks required for queries
- **Services:** a collection of system services that handle infrastructure, security, metadata, and optimization across the entire Snowflake system

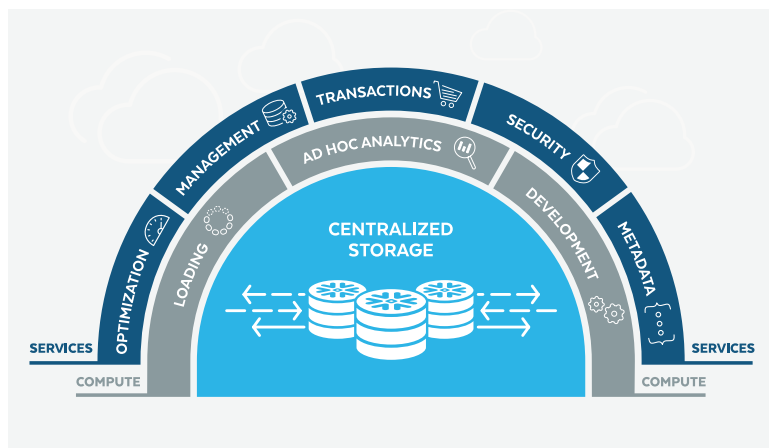


Fig. 3: Built from the ground up for the cloud, Snowflake's unique architecture physically separates and logically integrates compute, storage and services

In a traditional data warehouse, storage, compute, and database services are tightly coupled. This can stem from either the configuration of the physical nodes (even in the cloud), or the architecture of the data warehouse appliance. Even “big data” platforms tie storage, compute and services tightly together within the same nodes. Big data platforms can scale compute and storage to some degree, but they still suffer from the same predictable performance limitations as the number of workloads and users increase.

Snowflake dynamically brings together the storage, compute and services layers, delivering exactly the resources needed exactly when they are needed. The database storage layer resides in a scalable cloud storage service, such as Amazon S3, which ensures data replication, scaling and availability without any management by customers. Snowflake optimizes and stores data in a columnar format within the storage layer, organized into databases as specified by the user.

To allocate compute resources for tasks like loading, transformation and querying, users create “virtual warehouses” which are essentially MPP compute clusters. These virtual warehouses have the ability to access any of the databases in the database storage layer to which they have been granted access, and they can be created, resized and deleted dynamically as resource needs change. When virtual warehouses execute queries, they transparently and automatically cache data from the database storage layer. This hybrid architecture combines the unified storage of a shared-disk architecture with the performance benefits of a shared-nothing architecture.

The cloud services layer consists of a set of services that manage the Snowflake system—metadata, security, access control, and infrastructure. The services in this layer seamlessly communicate with client applications (including the Snowflake web user interface, JDBC, and ODBC clients) to coordinate query processing and return results. The services layer retains metadata about the data stored in Snowflake and how that data has been used, making it possible for new virtual warehouses to immediately use that data.

Unlike a traditional data warehouse, Snowflake can dynamically bring together the optimal set of resources to handle a multitude of different usage scenarios, with the right balance of IO, memory, CPU, etc. This flexibility is what makes it possible to

support data warehouse workloads with different query and data access patterns in a single service. Snowflake's architecture enables the following key capabilities:

- Support for all of your data in one system
- Support for all of your use cases with dynamic elasticity
- True ease of use with a self managing service and automatic adaptation

## HOW SNOWFLAKE DELIVERS ON THE PROMISE OF THE CLOUD DATA WAREHOUSE

### Support all of your data in one system

Snowflake designed a data warehouse that allows you to store all of your business data in a single system. That is a sharp contrast from current products, which are typically optimized for a single type of data, forcing you to create silos for different data or use cases.

### Native Support for Semi-Structured Data

Traditional database architectures were designed to store and process data in strictly relational rows and columns. These architectures built their processing models and optimizations around the assumption that this data consistently contained the set of columns defined by the database schema. This assumption made performance and storage optimizations like indices and pruning possible, but at the cost of a static, costly-to-change data model.

Structured, relational data will always be critical for reporting and analysis. But a significant share of data today is machine-generated and delivered in semi-structured data formats such as JSON, Avro, and XML.

Semi-structured data like this is commonly hierarchical and rarely adheres to a fixed schema. Data elements may exist in some records but not others, while new elements may appear at any time in any record. Correlating the information in this semi-structured data with structured data is important to extract and analyze the information within it.

Using semi-structured data in a traditional relational database requires compromising flexibility or performance. One approach is to transform that data into a relational format by extracting fields and flattening hierarchies so that it can be loaded into a relational database schema. This approach effectively puts the constraints of a fixed schema on that semi-structured data, sacrificing information and flexibility. Fields not specified for extraction are lost, including new fields that appear in the data. Adding fields requires a redesign of the data pipeline and updating all of the data that was previously loaded to include the new fields.

The alternative to this approach, which some databases have implemented, is a special datatype for storing semi-structured data as a complex object or simply as a string type. Although this approach preserves the information and flexibility in the semi-structured data, it sacrifices performance because the relational database engine can't optimize processing for semi-structured data types. For example, accessing a single element in an object commonly requires a full scan of the entire object in order to locate the element.

Because traditional data warehouses do not support the capabilities needed to effectively store and process semi-structured data, many customers have turned to alternative approaches, such as Hadoop, for processing this type of information. While Hadoop systems can load semi-structured data without requiring a defined schema, they require

specialized skills and are inefficient at processing structured data.

With either approach, there are massive sacrifices. In desperation, many organizations are adopting both a traditional data warehouse and Hadoop alongside one another. This creates additional complexity and subjects them to the negative aspects of both systems.

## Storing all types of data in Snowflake

Snowflake took a novel, different approach, designing a data warehouse that can store and process diverse types of data in a single system without compromising flexibility or performance. Snowflake's patented approach provides native storage of semi-structured data together with native support for the relational model and the optimizations it can provide.

"I can't say enough about how fantastic the native JSON support is. Snowflake lets us load our JSON data as is, flatten it all out, load it into the event tables, and then parse that into views. My analysts are really happy about this."

— Josh McDonald, Director of Analytics Engineering, KIXEYE

Snowflake started by making it possible to flexibly store semi-structured records inside a relational table in native form. This is accomplished through a custom datatype (Snowflake's VARIANT datatype) that allows schema-less storage of hierarchical data, including JSON, Avro, XML and Parquet. This makes it possible to load semi-structured data directly into Snowflake without pre-processing, losing information, or defining a schema. You simply create a table containing a column with Snowflake's VARIANT datatype and then load files containing semi-structured data into that table.

When Snowflake loads semi-structured data, it automatically discovers the attributes and structure that exist in the data to optimize how it's stored. It looks for repeated attributes across records, and then organizes and stores those repeated attributes separately, enabling better compression and fast access similar to the way that a columnar database optimizes column storage. Statistics about these pseudo-columns are also calculated and stored in Snowflake's metadata repository for use in optimizing queries. This storage optimization is completely transparent to the user.

Snowflake also enables you to query that data through extensions to SQL, making it simple to use relational queries that can combine access to structured and semi-structured data in a single query. Because of Snowflake's approach to storing semi-structured data, the Snowflake query optimizer has metadata information about the semi-structured data that allows it to optimize access to that data. For example, statistics in the metadata allow the optimizer to apply pruning to reduce the amount of data that needs to be read from the storage layer.

## Single System for All Business Data

Traditional architectures create isolated silos of data. Structured data is processed in a data warehouse. Semi-structured data is processed with Hadoop. Complex, multi-step operations are required to bring this data together. Scalability limits force organizations to separate workloads and data into separate data warehouses and data marts, essentially creating islands of data that have limited visibility and access to data in other database clusters.

All of these silos make it possible to configure a data warehouse, datamart, or Hadoop cluster that is tuned for a particular workload, but at greater cost and overhead. Even with a significant amount of infrastructure, it is often difficult to actually find

insights in the data because each silo of data only contains a part of the relevant data.

## Support all of your use cases elastically

The ideal data warehouse would be able to size up and down on-demand to provide exactly the capacity and performance needed, exactly when it is needed. However, traditional products are difficult and costly to scale up, and almost impossible to scale down. That forces an upfront capacity planning exercise that typically results in an oversized data warehouse, optimized for the peak workload but running underutilized at all other times.

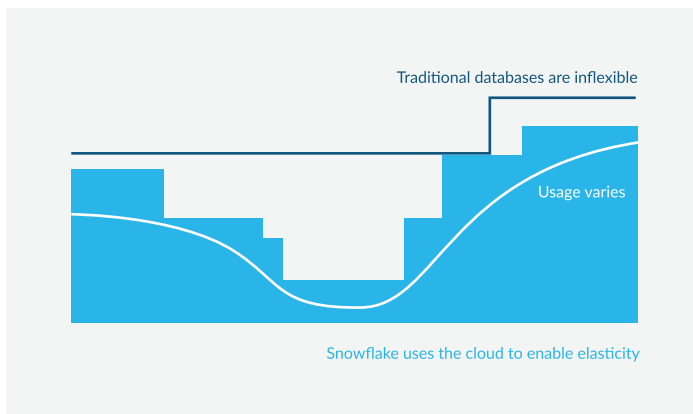


Fig. 4: Traditional data warehouses must be manually sized to the highest workload, if they are configurable at all. Cloud warehouses could be more elastic.

Cloud infrastructure uniquely enables full elasticity because resources can be added and discarded at any time. That makes it possible to have exactly the resources you need for all users and workloads, but only with an architecture designed to take full advantage of the cloud.

Snowflake's separation of storage, compute, and system services makes it possible to dynamically modify the configuration of the system. Resources can be sized and scaled independently and transparently, on-the-fly. This makes it possible for Snowflake to deliver full elasticity across multiple dimensions:

- **Data:** The amount of data stored can be increased or decreased at any time. Unlike shared-nothing architectures where the ratio of storage to compute is fixed, the compute configuration is determined independently of the volume of data in the system. This architecture also makes it possible to store data at a very low cost because no compute resources are required to store data in the database.



Fig. 5: Snowflake's unique architecture enables it to elastically support any scale of data, processing, and workloads

- **Compute:** The compute resources being used for query processing can also be scaled up or down at any time as the intensity of the workload on the system changes. Because storage and compute are decoupled, and the data is dynamically distributed, changing compute resources does not require reshuffling the data. Compute resources can be changed on-the-fly, without disruption.
- **Users:** With most data warehouses, there's a fundamental limit to scaling concurrency because all of the queries are competing for the same resources. As more users and workloads are added, the system gets slower and slower. Regardless of how large the cluster becomes, eventually the system cannot support additional concurrency and the only option is to create a new datamart. This brings with it the extra management burden of replicating or migrating data across systems. Snowflake can scale to support more users and workloads without performance impact because multiple virtual warehouses can be deployed on-demand, all with access to the same data.

## ENABLE EASE OF USE

### The need for a self-managing system

Conventional data warehouses and “big data” platforms require significant care and feeding. They rely on skilled administrators constantly exerting themselves to maintain the data platform: choosing data distribution schemes, creating and maintaining indices, updating metadata, cleaning up files, and more.

Manual optimization was feasible in an environment where queries were predictable and workloads were few, but it doesn’t scale when there are a large number of ever-changing workloads. The time and effort required to optimize the system for all those different workloads quickly gets in the way of actually analyzing data.

In contrast, Snowflake set out to build a data warehouse as a service where users focus on analyzing data rather than spending time managing and tuning. That required Snowflake to design a data warehouse that would:

- **Eliminate the management of hardware and software infrastructure.** The data warehouse should not require users to think about how to deploy and configure physical hardware. Similarly, users should not need to worry about installing, configuring, patching, and updating software.
- **Enable the system to learn and adapt.** Rather than requiring users to invest time configuring and tuning (and retuning) a wide array of parameters, Snowflake designed a data warehouse that sees how it is being used and dynamically adapts based on that information.

## Eliminating Software and Infrastructure Management

The Snowflake data warehouse was designed to completely eliminate the management of infrastructure. It is built on cloud infrastructure, which it transparently manages for the user. Users simply log in to the Snowflake service and it is immediately available, without complex setup required.

Ongoing management of the software infrastructure is also managed by Snowflake. Users do not need to manage patches, upgrades, and system security. The Snowflake service automatically manages the system.

Capacity planning, a painful requirement during the deployment of a conventional on-premises data warehouse, is all but eliminated because Snowflake makes it possible to add and subtract resources on the fly. Because it is easy to scale up and down based on need, you are not forced into a huge upfront cost in order to ensure sufficient capacity for future needs.

Other manual actions within traditional data warehouses that Snowflake automates include:

- **Continuous data protection:** Time Travel enables you to immediately revert any table, database or schema to a previous state. It’s enabled automatically and stores data as it’s transformed for up to 24 hours, or 90 days in enterprise versions.
- **Copying to clone:** Most data warehouses require you to copy data to clone, forcing a large amount of manual effort and a significant time investment. Snowflake’s multi-cluster, shared data architecture ensures that you never need to copy any data, because any warehouse or database automatically references the same centralized data store.

- **Data distribution** is managed automatically by Snowflake based on usage. Rather than relying on a static partitioning scheme based on a distribution algorithm or key chosen by the user at load time, Snowflake automatically manages how data is distributed in the virtual warehouse. Data is automatically redistributed based on usage to minimize data shuffling and maximize performance.
- **Loading data** is dramatically simpler because complex ETL data pipelines are no longer needed to prepare data for loading. Snowflake natively supports and optimizes diverse data, both structured and semi-structured, while making that data accessible via SQL.
- **Dynamic query optimization** ensures that Snowflake operates as efficiently as possible by looking at the state of the system when a query is dispatched for execution, not just when it is first compiled. That adaptability is a crucial component within Snowflake's ability to scale up and down.
- **Scaling compute:** Autoscaling is a feature that can be enabled within any Snowflake multi-cluster data warehouse that will match the number of compute clusters to the query or load, without needing manual intervention or input.

“Snowflake is faster, more flexible, and more scalable than the alternatives on the market. The fact that we don't need to do any configuration or tuning is great because we can focus on analyzing data instead of on managing and tuning a data warehouse.”

—Craig Lancaster, CTO, Jana

## PAY AS YOU GO

### Up front capital expenditures no longer make sense

As technology changes at an ever increasing pace, the old model of paying for licensed software and hardware in a massive up-front expenditure no longer makes sense. Data warehouses can be particularly painful to pay for in this model, as many traditional systems can cost tens of millions of dollars.

Newer cloud models that charge by the query aren't any better. Query based pricing can lead to runaway, unpredictable charges and frequent query failures as cost limits are hit. What's more, there isn't any way to define the compute power dedicated to each query, so you have to trust that the system is choosing the resource sizing that makes the most sense for your query.

### Enabling the data warehouse as a usage-based service

Snowflake is paid for as a usage based service. Each month, you pay for the data you store (at a cost similar to the raw storage costs of Amazon S3), and the number of Snowflake Compute Credits you use for compute. Each Credit costs around \$2, and one Credit provides enough usage for an XS data warehouse for one hour. A Small data warehouse -the next size up- costs 2 credits per hour and delivers approximately twice the compute horsepower. Each successive size of data warehouse continues to double both the compute horsepower and price in credits. This linear model makes it easy to plan for your expenditures, and keep them low in the first place.

Snowflake also addresses the limitations of the query based pricing model. Since you pay for each warehouse by the hour, costs are always known and understood. What's more, your query will never fail due to cost limits, it'll just take longer. The basic premise is that you have ultimate control over every piece of the warehouse, so if you want your query to move faster you can choose to move to a larger warehouse. Again, these are choices that you aren't given with inflexible query based models.

### Seamless sharing of data

Snowflake's architecture vastly simplifies the process of sharing data, particularly between different organizations. Instead of needing to manually create copies of data and sending them over FTP, EDI, or cloud file services, Snowflake Data Sharing allows any Snowflake customer to share access to their data with any other Snowflake customer.

Instead of sending a file, you send access to the underlying data. The schema and database structure can be imported automatically by the consumer so there's very little manual effort involved in using the shared data. What's more, when the data updates

in the provider account, it's automatically and immediately visible in the consumer account. Detailed permissions and role based access can be applied to that data, ensuring that information is only shared with the people who it is meant to be shared with.

### Use the SQL that you already know

The last benefit of Snowflake's architecture is the simplest, but in many ways the most important: you can use the SQL that your team already knows. noSQL systems and query based data stores have become more common recently, but they both fail to fully support standard ANSI SQL. This not only limits the way you interact with your data and transform it, but it requires you to hire people familiar with those systems, or train your existing people to use these new systems.

Snowflake allows your team to use the SQL they already know and love to transform and query all of your data. This simplicity pays endless dividends over time as you save time and resources you would otherwise devote to supporting bespoke and "oddball" systems.

## THE IMPACT OF REINVENTION

By reimagining and reinventing the data warehouse, Snowflake has addressed all of the key limitations of today's technology. Doing so required a new architecture that was completely different from data warehouses of the past. As a result, you can easily store all your data, enable all your users with zero management, paying the way you want to and using the SQL you already rely on. Rather than being bottlenecked waiting for the availability of overstretched IT and data science resources, analysts get rapid access to data in a service that can operate at any scale of data, users, and workloads.

To learn more about Snowflake, join us for a live demo at <https://www.snowflake.net/webinar/snowflake-livedemo/>



Snowflake Computing, the cloud data warehousing company, has reinvented the data warehouse for the cloud and today's data. Snowflake is built from the cloud up with a patent-pending new architecture that delivers 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 is headquartered in Silicon Valley and can be found online at [snowflake.net](https://www.snowflake.net).