The Snowflake Elastic Data Warehouse

By Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, Philipp Unterbrunner

Presented at SIGMOD '16: Proceedings of the 2016 International Conference on Management of Data on June 14, 2016

Nathan Eng, Joo Lim, Kyungmin Park

Georgia Tech

Background & Motivation

Fixed resource design

Traditional data warehouses have fixed compute and storage resources that can't scale elastically.

Lack of elasticity

Resources can't expand or shrink based on workload demands resulting in poor efficiency.

Complex ETL

Moving data from sources into the warehouse requires complex and brittle ETL jobs.

Physical tuning

Performance tuning requires physical changes like indexing, partitioning, materialized views etc.

These limitations make traditional data warehouses a poor fit for highly dynamic cloud environments.

Why It Matters

Unlimited Resources

The cloud provides unlimited compute, storage, and network resources that can scale on demand to meet requirements.

Flexibility

Cloud-based systems can leverage different types of resources as needed for changing workloads and data processing needs.

Cost Efficiency

Only pay for the cloud resources used, which provides great cost efficiency compared to on-premises infrastructure.

Agility

Quickly spin up and configure cloud resources to support new types of data and workloads.

Innovation

Utilize latest cloud services for AI, machine learning, analytics to gain new insights from data.

Global Reach

Deploy cloud-based systems globally to bring insights to users anywhere in the world.

Related Work: Limitation of Previous Technologies

- Low query performance
- Difficult to scale
- High maintenance costs
- Limited cloud capabilities

Big Idea: Snowflake

- Snowflake is a cloud-based data warehouse that separates storage and compute. This allows for extreme elasticity, scalability, and pay-as-you-go pricing.

- Snowflake supports standard SQL, ACID transactions,and semi-structured data.

*ACID: Atomocity, Consistency, Isolation, Durability

Overview: Snowflake Architecture

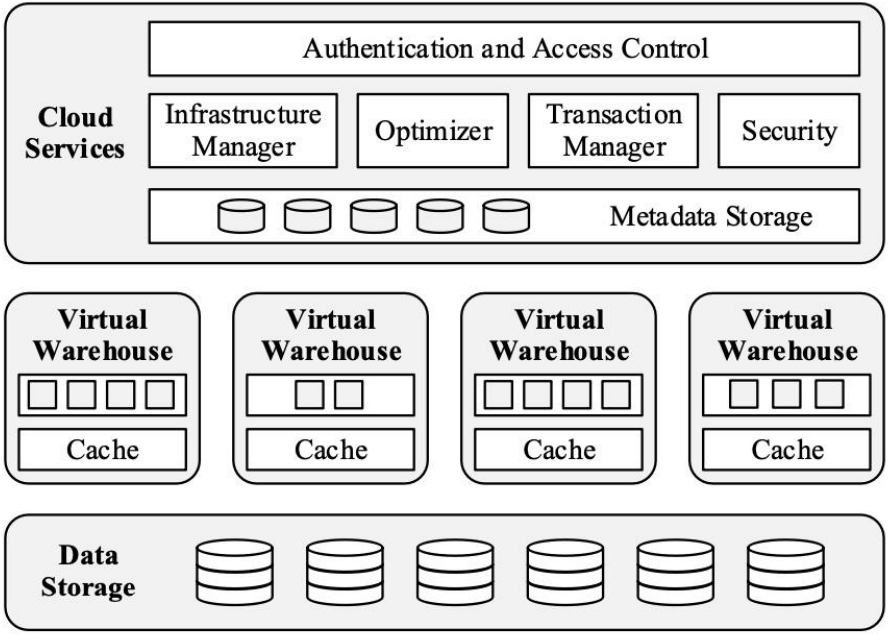


Figure 1: Multi-Cluster, Shared Data Architecture

amazon EC2

amazon S3

Proposed Solutions & Contribution

Multi-tenant and Highly Scalable Architecture

Security and Ease of Access

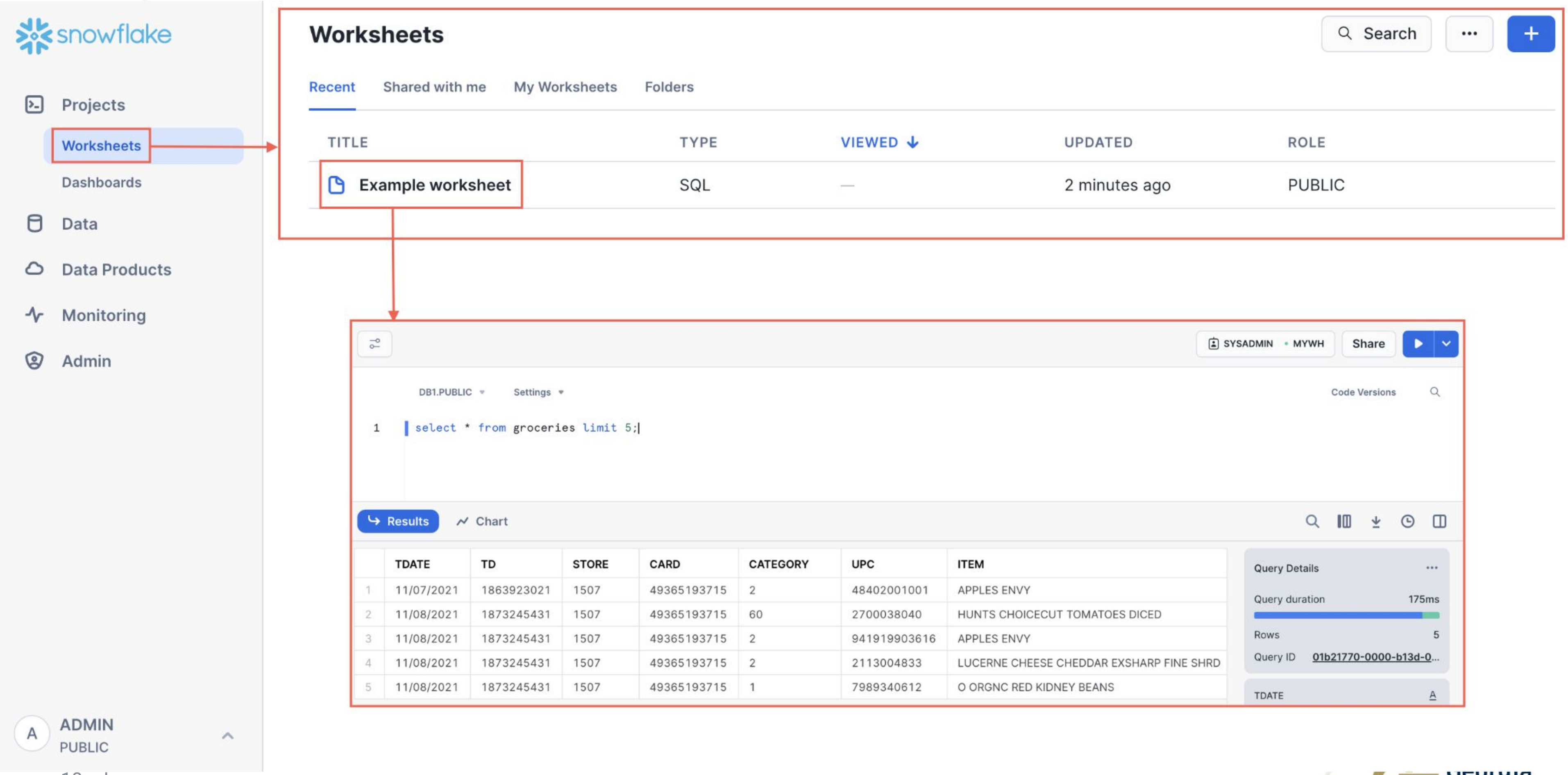
Easier to scale storage & compute separately

Technical Details

1. Pure Software-as-a-Service Experience
2. Continuous Availability
3. Semi-Structured and Schema-Less Data
4. Time Travel and Cloning
5. Security

Pure Software-as-a-Service Experience

- Standard database interfaces supported
  - JDBC, ODBC, Python Database
- Web UI and data accessibility
  - Only need a web browser, no download
- Continuous and comprehensive UI Functionality
  - Online collaboration features, user feedback
- Focus on ease-of-use
  - Priority is the simplicity
  - No failure modes, tuning knob



https://docs.snowflake.com/en/user-guide/ui-snowsight-quick-tour

Continuous Availability

Fault Resilience

- Tolerate node failures at all levels
- Replicated across multiple AZs:
  - Data Storage layer (S3)
  - Metadata store
- Guarantees:
  - 99.99% data availability
  - 99.999999999% durability
- Stateless nodes in Cloud Services with load balancer
- VWs not distributed across AZs

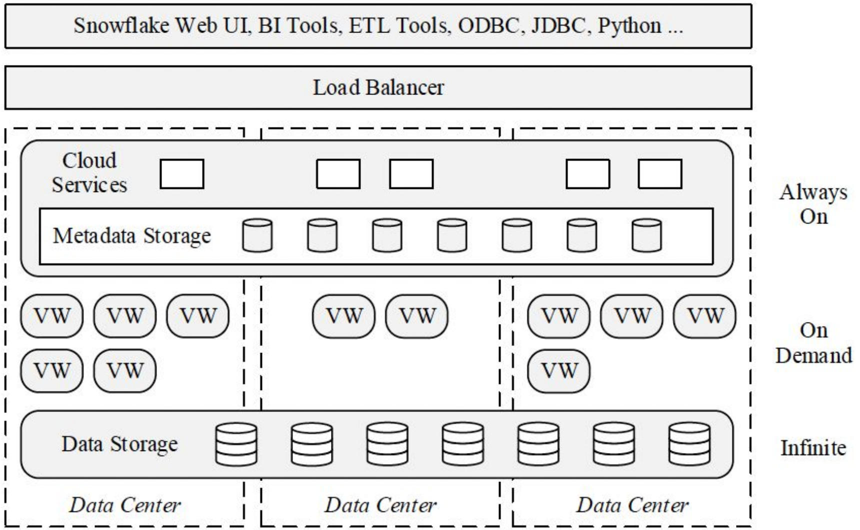


Figure 2: Multi-Data Center Instance of Snowflake

Georgia Tech

Continuous Availability

Online Upgrade

- Multiple versions to be deployed side-by-side
- All services are stateless
- Metadata versioning and schema evolution
- Smooth software upgrade
    - No downtime or performance degradation
- Weekly upgrade cycle
- Continuous testing and exercise of upgrade/downgrade mechanism

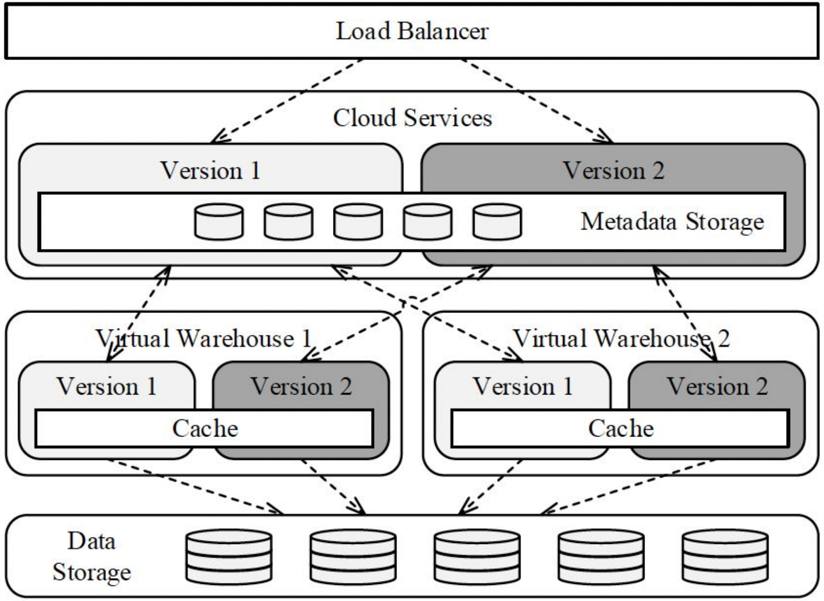
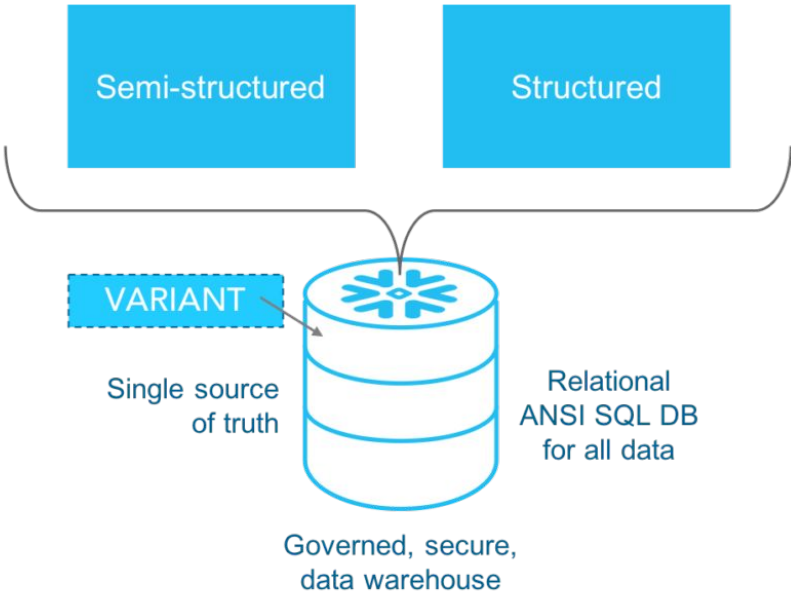


Figure 3: Online Upgrade

Semi-Structured and Schema-Less Data

- Semi-structured data type:
  - VARIANT, ARRAY, OBJECT
- VARIANT can store JSON, Avro, XML
- ELT rather than traditional ETL
- Efficient querying from post-relational operations
  - Data extraction
  - Flattening
  - Aggregation
- Columnar storage
- Nearly on par with conventional relational data



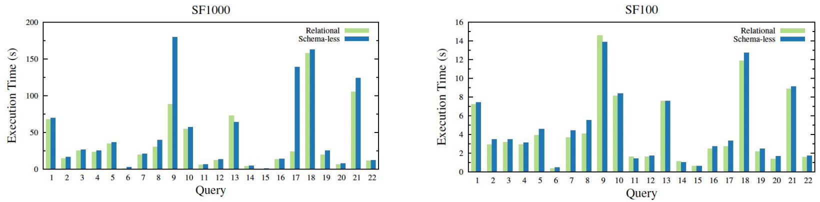


Figure 4: TPC-H SF100 and SF1000 Performance: Relational vs. Schema-less Row Format

https://www.snowflake.com/blog/5-reasons-to-love-snowflakes-architecture-for-your-data-warehouse/

Time Travel and Cloning

- Implements Snapshot Isolation on top of MVCC
- File retention up to 90 days
- Time travel queries using AT or BEFORE syntax

```
SELECT * FROM my_table AT(TIMESTAMP =>
  'Mon, 01 May 2015 16:20:00 -0700'::timestamp);
SELECT * FROM my_table AT(OFFSET => -60*5); -- 5 min ago
SELECT * FROM my_table BEFORE(STATEMENT =>
  '8e5d0ca9-005e-44e6-b858-a8f5b37c5726');
```

```
SELECT new.key, new.value, old.value FROM my_table new
JOIN my_table AT(OFFSET => -86400) old -- 1 day ago
ON new.key = old.key WHERE new.value <> old.value;
```

Time Travel and Cloning

- UNDROP for quick restoration

```
DROP DATABASE important_db; -- whoops!
UNDROP DATABASE important_db;
```

- CLONE for efficient snapshot for backups

```
CREATE DATABASE recovered_db CLONE important_db BEFORE(
  STATEMENT => '8e5d0ca9-005e-44e6-b858-a8f5b37c5726');
```

Security

- Two-Factor Authentication
- Encrypted data handling
- Role-Based Access Control (RBAC)
- Key Hierarchy
- Key Life Cycle Management
- End-to-End Security

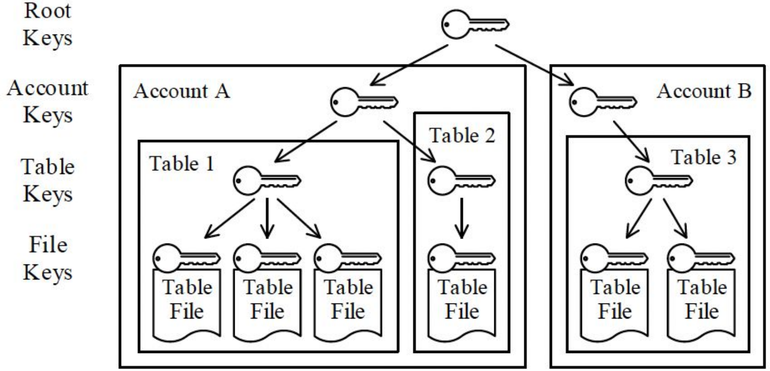


Figure 5: Encryption Key Hierarchy

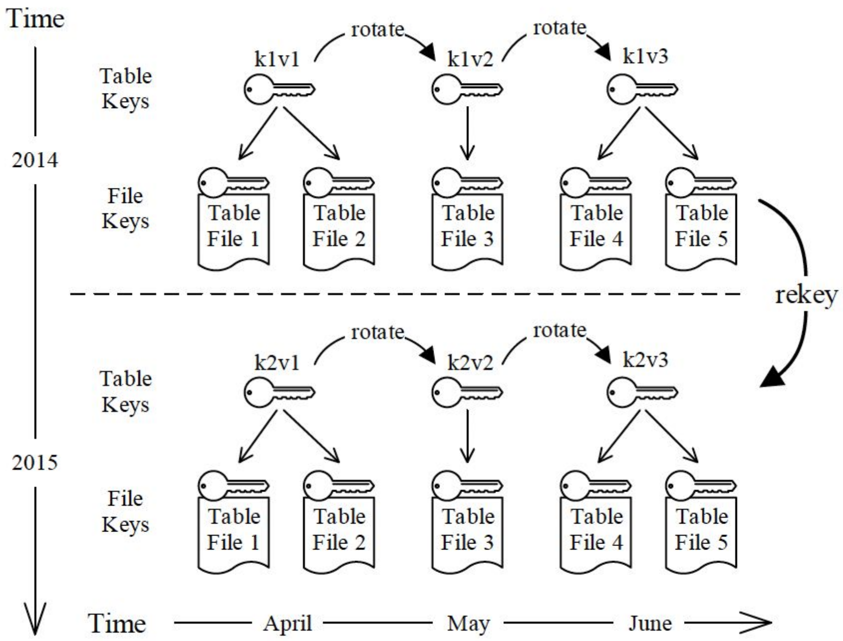


Figure 6: Table Key Life Cycle

Georgia Tech

Author Hypotheses

- Traditional data warehousing tools meant for fixed resources and predictable data
- Build a flexible, enterprise-ready data warehousing solution for the cloud
  - dynamic resources
- Snowflake - separate storage and compute

Results & Performance

- TPC-H-like data and queries
- 2 DB schemas - relational TPC-H schema and schema-less
- 10% overhead for schema-less storage and query processing

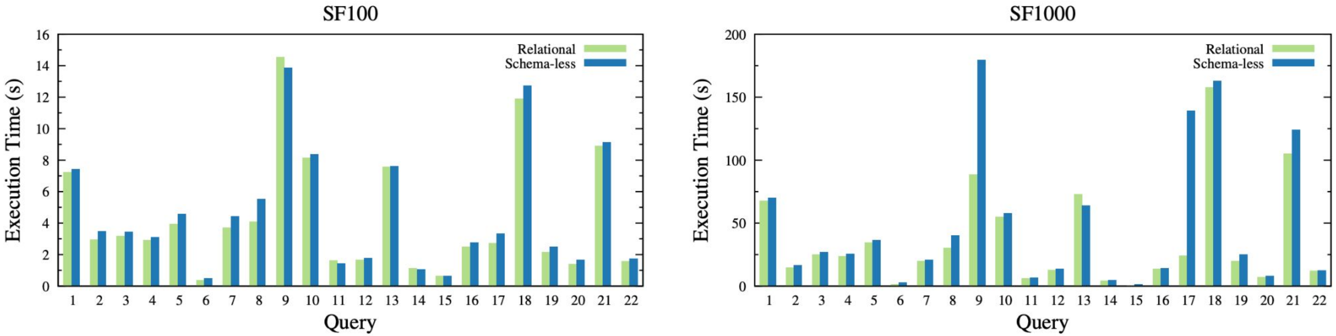


Figure 4: TPC-H SF100 and SF1000 Performance: Relational vs. Schema-less Row Format

Related Work

- Amazon Redshift for data warehousing
  - evolved from parallel database system ParAccel
  - first real data warehouse system offered as a service
  - utilizes classic shared-nothing architecture
    - scalable, but adding or removing compute resources = data redistribution
- GCP BigQuery
  - run parallelized queries on terabytes of data
  - support for JSON and nested data
  - harder to use with SQL-based products
- Document stores and big data
  - MongoDB, Apache Cassandra, etc.

Georgia Tech

Conclusion & Future Work

- SaaS model made it easy for users to try out and adopt system
- Found Snowflake replacing traditional database systems and Hadoop clusters
- Comparisons against other solutions with deep ETL pipelines (performance-wise)
- Performance improvements, although scalability has already offset that (primary focus)
- Migration to other cloud platforms?
  - heavily relies on AWS S3 architecture and runs virtual warehouses on EC2 instances
- Security
- Improving data access performance by providing additional metadata structures and data-reorganization tasks
- Biggest challenge - transition to a full self-service model

2 Study Questions

1. How does Snowflake performance compare to other solutions out there (in terms of quantitative metrics)?
2. Does Snowflake benefit smaller companies that deal with less data or is it more advantageous to explore alternative solutions?

FeedBack

1. Please explain how Snowflake differs from other database systems in terms of indexing?

a) Automatic Indexing and Optimization: Snowflake automatically handles indexing and optimization behind the scenes. Traditional databases often require manual index creation and optimization to accelerate query performance. Snowflake uses metadata about the data stored to automatically optimize queries without the need for explicit indexes or tuning. This reduces the administrative overhead and makes it easier for users to work with large datasets without deep optimization knowledge.

b) No Need for Traditional Index Management: Because of its automatic and dynamic data partitioning and optimization, Snowflake eliminates the need for traditional index management. This is a departure from databases like Oracle, MySQL, or SQL Server, where careful planning and continuous management of indexes are crucial for maintaining performance. This can significantly reduce the time and expertise required to manage data at scale.

FeedBack

2. How do AWS S3 influence the design of Snowflake's architecture?

a) Unlimited Scalability: S3 provides virtually unlimited storage, allowing Snowflake to scale data storage needs without the constraints typically associated with traditional data warehouse solutions. This scalability is crucial for accommodating the exponential growth of data, enabling Snowflake to serve a wide range of customers, from those with small datasets to large enterprises with petabytes of data.

b) Durability and Availability: S3 guarantees high durability (99.999999999% or 11 nines) for objects stored, which means that data is safely stored across multiple facilities and can withstand the loss of two facilities concurrently. This level of durability reassures Snowflake users that their data is safe and reliably stored, a critical consideration for businesses that depend on data for their operations.

c) Decoupling of Storage and Compute: The use of S3 allows Snowflake to separate storage from compute resources. This decoupling means that customers can scale their compute resources up or down, depending on their processing needs, without having to scale their storage simultaneously. This flexibility leads to cost savings and efficiency because customers don't need to over-provision resources to handle peak loads.

Reference by: https://aws.amazon.com/ko/s3/faqs/

FeedBack

3. Why doesn't Snowflake use services like BigQuery or Azure in terms of Data Storage?

a) Early Market Position and Maturity of AWS: When Snowflake was initially developed, AWS was the leading cloud service provider with a mature ecosystem. S3, being one of AWS's earliest services, offered proven scalability, durability, and a wealth of features that were attractive for building a reliable and scalable cloud data warehouse. The maturity of AWS's infrastructure and services provided a solid foundation for Snowflake to build upon.

b) Broad Adoption and Ecosystem: AWS's broad adoption by enterprises, startups, and developers meant that a significant portion of Snowflake's potential customer base was already using AWS for their cloud infrastructure needs. Integrating closely with AWS and leveraging S3 allowed Snowflake to easily fit into the existing cloud architecture of many organizations, facilitating adoption and integration.

c) Cloud-Agnostic Strategy: Despite the initial selection of S3, Snowflake's architecture is designed to be cloud-agnostic. This means that while it may leverage S3 for its AWS deployments, it also supports and utilizes the object storage services of other cloud providers like Azure Blob Storage and Google Cloud Storage for deployments in those environments. This approach allows Snowflake to serve customers across different cloud platforms and not limit its market reach.

Questions?