Snowflake

Contents

- Database architecture?
- Introduction to Snowflake
- Multi-cluster, shared data architecture
- Data architecture
- Virtual Warehouses
- Query pruning
- Micro-partitions (benefits)
- Data Clustering
- Semi structured data support
- Comparison between Snowflake and MongoDB and SQL Server

Database architecture?

- Complexity: Managing both **infrastructure** as well as **data**
- Limited scalability Can’t support **all** data, users, workload
- Rigid costs Keep on the lights **24/7**
- Inadecuate Elasticity Stuck with rigid **inflexible** architectures

Introduction to Snowflake

A **cloud-based data warehousing platform** designed to handle **large volumes** of data and enable organizations to store, manage, and **analyze** their data efficiently.

- Multi-Cloud Support: Snowflake supports major cloud platforms, including **AWS**, **Azure**, and **Google Cloud**, offering flexibility and choice to users.
- Unique Architecture: Snowflake's architecture is built on a **multi-cluster, shared data architecture**, providing advantages in terms of

Multi-cluster, shared data architecture

- **Multi-cluster:**
  - In Snowflake's architecture, multiple compute clusters, also known as **virtual warehouses**, can operate **concurrently**.
  - Each virtual warehouse is a set of **compute resources** that can process queries and perform data operations **independently** of other warehouses.
  - The multi-cluster aspect allows Snowflake to handle **high concurrency**, with multiple users or workloads running **simultaneously** without interference.
- **Shared data:**
  - The data in **Snowflake is stored separately from the compute resources**. It resides in cloud-based object storage, such as Amazon S3, Azure Blob Storage, or Google Cloud Storage.
  - **Multiple** virtual warehouses share access to the **same underlying data**, promoting collaboration and avoiding data duplication.
  - Data is stored in a **columnar** format, and Snowflake employs a **metadata layer** to manage and organize this data efficiently.

Data architecture

The Snowflake data warehouse architecture has three layers

- **Database Storage Layer**
  - Snowflake saves and manages data on the cloud using a shared-disk approach, making data management simple.

- **Query Processing Layer**
  - For query execution, Snowflake uses the Virtual Warehouse. The query processing layer is separated from the disk storage layer in the Snowflake data architecture. You can use the data from the central storage layer to run queries in this layer.

- **Cloud Services Layer**
  - The cloud services layer contains all of the operations that coordinate throughout Snowflake, such as **authentication**, **security**, **data management**, **and query optimization**. A cloud service is a stateless computing resource that operates across different availability zones and uses highly accessible and usable information. The cloud services layer provides a SQL client interface for data operations such as DDL and DML.

Virtual Warehouses

- A virtual warehouse, often referred to simply as a “warehouse”, is a cluster of compute resources in Snowflake.
- A warehouse provides the required resources, such as CPU, memory, and temporary storage, to perform the following operations in a Snowflake session:
  - Executing SQL SELECT statements that require compute resources (e.g. retrieving rows from tables and views).
  - Performing DML operations, such as:
    - Updating rows in tables (DELETE , INSERT , UPDATE).

Query pruning

- Query pruning is a Snowflake query optimization technique that involves **reducing the amount of data that must be scanned during query execution**. It accomplishes this by utilizing the power of the Snowflake **micro-partitions** that make up a table to determine which ones contain relevant data and which ones can be skipped.

Micro-partitions

- All data in Snowflake tables is **automatically** divided into micro-partitions, which are contiguous units of storage.
- Each micro-partition contains between 50 MB and 500 MB of uncompressed data
- Groups of rows in tables are mapped into individual micro-partitions, organized in a columnar fashion.
- This size and structure allows for extremely granular pruning of very large tables, which can be comprised of millions, or even hundreds of millions, of micro-partitions.

Micro-partitions: benefits

- In contrast to traditional static partitioning, Snowflake micro-partitions are derived **automatically**; they don't need to be explicitly defined up-front or maintained by users.
- Micro-partitions are **small** in size (50 to 500 MB, before compression), which enables extremely efficient DML and fine-grained pruning for faster queries.
- Micro-partitions can **overlap** in their range of values, which, combined with their uniformly small size, helps **prevent data skew**.
- Columns are stored **independently** within micro-partitions, often referred to as *columnar storage*. This enables **efficient scanning of individual columns**; only the columns referenced by a query are scanned.
- Columns are also **compressed** individually within micro-partitions. Snowflake automatically determines the most efficient compression algorithm for the columns in each micro-partition.

What is Data Clustering?

- Typically, data stored in tables is sorted/ordered along natural dimensions (e.g. date and/or geographic regions). This “clustering” is a key factor in queries because table data that is not sorted or is only partially sorted may impact query performance, particularly on very large tables.

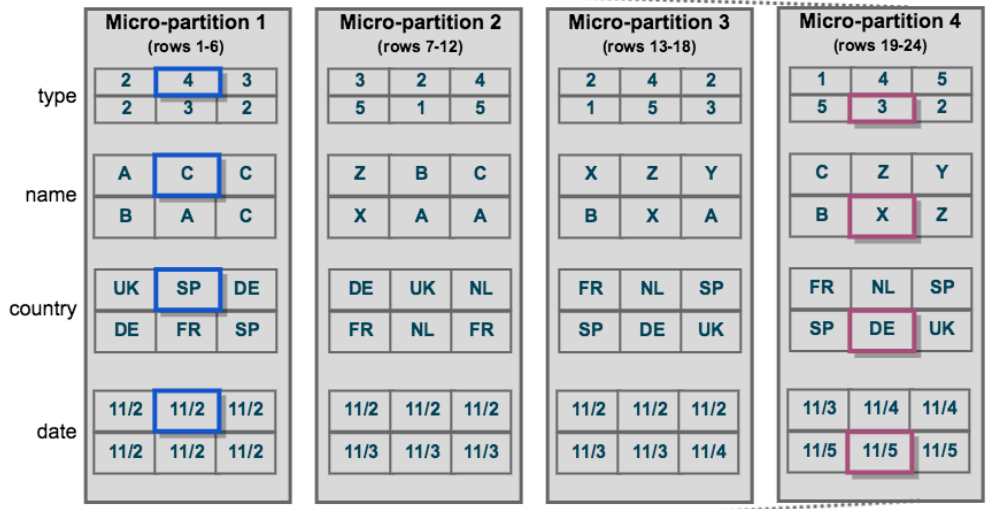
- In Snowflake, as data is inserted/loaded into a table, **clustering metadata is collected** and recorded for each micro-partition created during the process. Snowflake then leverages this clustering information to **avoid unnecessary scanning** of micro-partitions during querying, significantly accelerating the performance of queries that reference these columns.

Table: `t1`

Logical Structure

type	name	country	date
2	A	UK	11/2
4	C	SP	11/2
3	C	DE	11/2
2	B	DE	11/2
3	A	FR	11/2
2	C	SP	11/2
3	Z	DE	11/2
2	B	UK	11/2
4	C	NL	11/2
5	X	FR	11/3
1	A	NL	11/3
5	A	FR	11/3
2	X	FR	11/2
4	Z	NL	11/2
2	Y	SP	11/2
1	B	SP	11/3
5	X	DE	11/3
3	A	UK	11/4
1	C	FR	11/3
4	Z	NL	11/4
5	Y	SP	11/4
5	B	SP	11/5
3	X	DE	11/5
2	Z	UK	11/5

Physical Structure



Semi structured data support

```
/*---------------------------------------------------------------------------
We can query the JSON data just the same as if we had structured it on load
---------------------------------------------------------------------------*/

-- note the simple syntax for accessing a key-value pair in the JSON
-- when does it snow in NYC?
select
    v,
    t,
    v:city.name :: string city,
    v:weather [0].main :: string conditions
from weather
where v:city.name = 'New York'
    and v:weather [0].main = 'Snow'
limit 100;
```

Objects | Query | Results | Chart

	V	T	CITY	CONDITIONS
1	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORI	2019-01-22 11:05:10	New York	Snow
2	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORI	2019-01-11 1[illegible]7:23	New York	Snow
3	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORI	2019-01-22 11:05:10	New York	Snow
4	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORI	2019-01-01 09:07:55	New York	Snow
5	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORI	2019-01-25 07:05:11	New York	Snow
6	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORI	2020-05-11 22:02:44	New York	Snow
7	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORI	2020-05-11 23:02:36	New York	Snow

Query Details

Query duration 2.9s

Rows 100

V {[

100% filled

	Datamodel	Architecture	Key Features	Use case
Snowflake	**Type**: **Data Model**: Tables and views with defined schemas. **Storage Format**:	**Architecture**: **Scalability**: **Storage**:	**Key Features:** Multi-cluster, shared data architecture, data sharing, zero-copy cloning, real-time analytics. **Notable**:	**Use Cases:** Data warehousing, analytics, data sharing, collaborative projects. **Strengths**:
SQL Server	**Type**: Relational database. **Data Model:** Tables with defined schemas. **Storage Format**: Row-based storage.	**Architecture**: Traditional client-server architecture. **Scalability**: Vertical scaling by adding more resources (CPU, RAM) to a single server. **Storage**: Typically uses traditional disk-based storage.	**Key Features**: ACID compliance, support for stored procedures, triggers, and complex transactions. **Notable**: Robust relational database features, integration with Microsoft ecosystem.	**Use Cases**: Traditional relational database applications, transactional systems. **Strengths**: ACID compliance, support for complex transactions.

	Datamodel	Architecture	Key Features	Use case
Snowflake	**Type**: Relational database with support for structured and semi-structured data. **Data Model**: Tables and views with defined schemas. **Storage Format**: Stored in a columnar format.	**Architecture**: Multi-cluster, shared data architecture with separate storage and compute layers. **Scalability**: Elastic scalability with the ability to scale storage and compute independently. **Storage**: cloud-based object storage	**Key Features:** Multi-cluster, shared data architecture, data sharing, zero-copy cloning, real-time analytics. **Notable**: Separation of storage and compute, automatic scaling.	**Use Cases:** Data warehousing, analytics, data sharing, collaborative projects. **Strengths**: Handling large-scale analytics, separation of storage and compute.
SQL Server	**Type**: Relational database. **Data Model:** Tables with defined schemas. **Storage Format**: Row-based storage.	**Architecture**: Traditional client-server architecture. **Scalability**: Vertical scaling by adding more resources (CPU, RAM) to a single server. **Storage**: Typically uses traditional disk-based storage.	**Key Features**: ACID compliance, support for stored procedures, triggers, and complex transactions. **Notable**: Robust relational database features, integration with Microsoft ecosystem.	**Use Cases**: Traditional relational database applications, transactional systems. **Strengths**: ACID compliance, support for complex transactions.

	Datamodel	Architecture	Key Features	Use case
Snowflake	**Type**: Relational database with support for structured and semi-structured data. **Data Model**: Tables and views with defined schemas. **Storage Format**: Stored in a columnar format.	**Architecture**: Multi-cluster, shared data architecture with separate storage and compute layers. **Scalability**: Elastic scalability with the ability to scale storage and compute independently. **Storage**: cloud-based object storage	**Key Features:** Multi-cluster, shared data architecture, data sharing, zero-copy cloning, real-time analytics. **Notable**: Separation of storage and compute, automatic scaling.	**Use Cases:** Data warehousing, analytics, data sharing, collaborative projects. **Strengths**: Handling large-scale analytics, separation of storage and compute.
MongoDB	**Type**: NoSQL document database. **Data Model:** Collections of JSON-like documents (BSON format). **Storage Format:** BSON (Binary JSON) format.	**Architecture**: Distributed, horizontal scaling with sharding for data distribution. **Scalability**: Scales horizontally by adding more servers to a MongoDB cluster. **Storage**: Data is distributed across shards, and each shard is a replica set.	**Key Features:** JSON-like document model, horizontal scaling through sharding, flexible schema. **Notable**: NoSQL flexibility, automatic sharding.	**Use Cases:** Document-oriented applications, real-time applications, content management systems. **Strengths**: Flexible schema, scalability for write-intensive workloads.

Use cases for Snowflake

- **Data warehousing**
  - Centralized Data Storage
  - Snowflake serves as a centralized repository for storing and managing large volumes of structured and semi-structured data. It facilitates efficient data warehousing for analytics.

- **Data sharing**
  - Secure Collaboration
  - Organizations can securely share data with external partners, customers, or other departments. This feature is valuable for collaborative projects and data monetization strategies.

- **Advanced analytics**
  - Complex Analytics and Reporting
  - Snowflake's high-performance analytics capabilities make it suitable for running complex queries, data exploration, and generating insightful reports.