## SNOWFLAKE

1. [Introduction](#_Chapter_1_:)
2. [Self registration](#_Chapter_2_:)
3. [Snowflake Architecture](#_Chapter_3_:)
4. [Clustering](#_Chapter_4_:)
5. SnowSight (Modern Web UI)
6. Snowflake Unique features
7. Container, Database, Schema & Tables
8. Snowflake Objects
9. Bulk/Batch Load
10. Continuous Data Load
11. External Table & Data Load
12. Virtual Warehouse
13. Snowflake Clustering Concept
14. Time Travel Feature
15. Data Cloning Feature
16. Snowflake Data Sharing + Reader’s Account
17. Snowflake Stream Objects
18. Snowflake Task Objects
19. ETL in Snowflake (Stream & Task Together)
20. Snowflake Role Hierarchy
21. Stored Procedure, UDFs & Views
22. Snowflake Information Schema
23. Snowflake Monitors & Billing
24. Snowflake Connectors (JDBC/ODBC/Python)
25. PowerBI & Tableau with Snowflake

# Chapter 1 : Introduction

* 1. Evolution of Cloud Data Platform

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Database with solid fill | Cloud with solid fill | A cartoon of a yellow elephant  Description automatically generated |  |
|  | On premise data | 1st Gen Cloud EDW | Data Lake, Hadoop | Cloud data platform |
| All Data | X | X | X | Checkmark with solid fill |
| All Users | X | X | - | Checkmark with solid fill |
| Fast answers | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| SQL Databases | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |

* + 1. 1st Evolution
       - Traditional On – premise Enterprise Data warehouse.
       - They were able to answer the business questions (queries) very fast.
       - They were all SQL Databases.
       - Challenges:
         * Not designed for all kind of data – structured & unstructured.
         * Not designed for all the users who may have demand for that data.
         * Not scalable.
    2. 2nd Evolution
       - 1st Generation Cloud EDW
       - They moved the same code into the public cloud.
       - They fast answer to queries & SQL Database but inherent problem still existed like:
* Unable to onboard all type of data (structured and unstructured).
* Access to all your different users was still unavailable.
  + 1. 3rd Evolution
       - Hadoop / Data Lake
       - You can load any kind of data.
       - A lot of users who have a demand for the data can access it to a certain extent.
       - Slow and no native support to SQL.
       - Performance & scalability is challenging (operational/manageability).
    2. 4th Evolution – Cloud Data Platform
       - All users, all data, fast answer, SQL – all features are in a single platform.
       - Natively ingest structure/semi – structured data.
       - Concurrent access to as many users within organisation.
       - Standard SQL DB, hence easy to manage.
  1. Modern Data Architecture with Snowflake

A screen shot of a data architecture

Description automatically generated

* + 1. Build for the cloud
       - Build from scratch (cloud – native), optimise for cloud (AWS/Azure/GCP), storage & compute is decoupled.
       - All kind of use case are supported (Data warehouse, Data Lake, Data Engineering, Streaming, Data Application, Data Science, Data Exchange).
       - Share data to internal as well as external customers securely (regulator/partner, etc.) without data leaving your premise.
    2. Software as a Service (SaaS)
       - No software, infrastructure, or upgrades to manage.
       - Any update, maintenance, releases are done by the snowflake, and it is available to all customers automatically.
       - In all the 3 major cloud provides.
    3. Pay as you go
       - Storage and compute charged independently and only for usage (an issue in Hadoop).
       - If we store TBs of data and no processing is performed, we are charged only for the storage and not computing.
    4. Scalable
       - Virtual warehouses enable computing scaling independently form storage.
       - The scalability is virtually unlimited and storage and compute both can scale as needed and independent from each other.
  1. The Value of a Cloud Data Platform
     + - Unlimited performance and scale
       - Near – zero maintenance, as a service.
       - One platform, one copy of data, many workloads.
       - Secure governed access to all data.
         * **Snowflake supports all these types of use cases. Hence is it a true cloud data platform.**

# Chapter 2 : Self registration

* + - * Simply go to [Snowflake Trial](https://signup.snowflake.com/?utm_source=google&utm_medium=paidsearch&utm_campaign=ap-in-en-brand-core-exact&utm_content=go-eta-evg-ss-free-trial&utm_term=c-g-snowflake-e&_bt=579374818706&_bk=snowflake&_bm=e&_bn=g&_bg=133380614608&gclsrc=aw.ds&gad_source=1&gclid=EAIaIQobChMI7J6lsLHnhAMV8KlmAh2zlQZXEAAYASAAEgJ2mPD_BwE) and register yourself on the snowflake website (Choose Enterprise edition + AWS).
      * Snowflake supports 3 cloud service providers to host account:

AWS

Azure

GCP

* + - * Please make sure to choose the same region where the data is to be fetched from. E.g. If the data is in US East (Northern Virginia) in AWS S3 bucket then choose the same region in Snowflake to reduce extra cost.
      * When I host snowflake account on AWS it will only use AWS services to store data and utilize AWS compute resources.
      * When you create a database, two default schemas are created:

1. Public
2. Information schema

* Every query executed in Snowflake has its own query id.
* Database consists of:
  + Tables
  + Views
  + Schema
  + Stages
  + Pipes
  + Sequences
  + File format

# Chapter 3 : Snowflake architecture

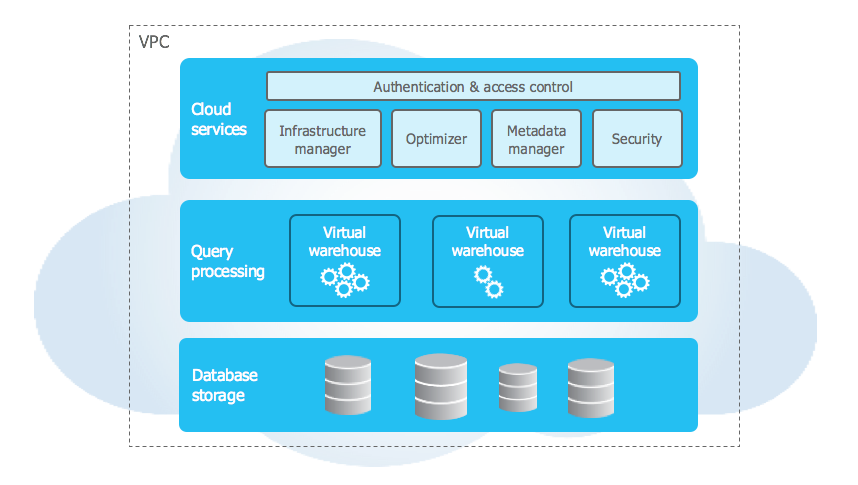
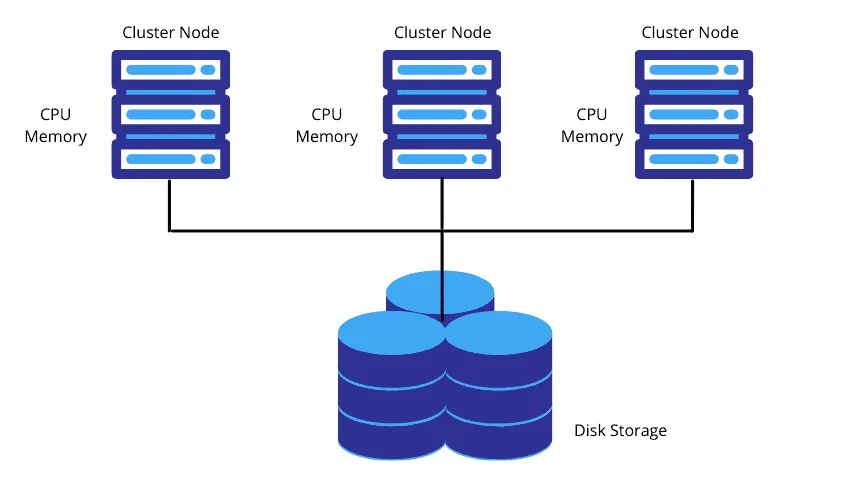


Figure 1. Multi-cluster, shared data architecture

1. Data Storage Layer : This layer stores all table data and query results (AWS, Azure or GCP).
2. Query processing : Handles query execution within elastic clusters of virtual machines. This is also called as “**muscle of the system**”.
3. Cloud Services Layer : It is a collection of services used to manage virtual warehouses, queries we execute on Snowflake, transactions and metadata information. It provides access control, usage statistics, and query optimization and other services. Hence, it is called as “**brain of the system**”.

* For every query we execute, we’ll be charged, hence we need to understand the architecture which helps us to save cost.
* Let’s understand other architectures :
* Shared disk architecture : In this architecture, we’ll have one shared disk which will be accessed by multiple database nodes. Every database node will try to read and write data on this shared disk.



* + Limitations:
    - Scalability : With increase in number of nodes, it will become difficult to read and write data, as shared disk will become a bottleneck.
    - Hard to maintain data consistency across the cluster : This becomes an issue if two or more nodes are trying to read/write data at the same time.
    - Bottleneck of communication with shared disk.
* Shared Nothing architecture : To solve issues found in ‘shared disk architecture’ in this architecture, we assign storage for every compute node. A diagram of a server

  Description automatically generated
  + Advantages:
    - It scales processing and compute together.
    - Eliminates bottleneck issue.
    - It moves data storage close to compute.
  + Disadvantages:
    - With increase in data volumes, no. of nodes will increase. Each node is only responsible for the data it has stored on the disk. If a node wants data from a different node, data should get shuffled b/w the nodes.
    - What if a node fails? Data should get shuffled to other nodes. Such failures will impact the performance of the whole system. Performance is heavily dependent on how data is distributed across the nodes in the system.
    - Choosing right balance b/w storage and compute is difficult. If you wish to increase storage without adding compute, you can’t do that as **data and compute are tightly coupled together** in this architecture. Compute can’t be sized independently of storage.
    - **Heterogeneous workload and homogenous hardware** : During loading data activity, we will be doing “*bulk loading*”, such process is only intended to copy data quickly inside the system. During such process **we will not be utilizing heavy compute, but we will be using high bandwidth**. In some other cases, we will be cleansing data, we do aggregation operations and other transformation activity on the data, such process requires **heavy compute and low I/O bandwidth.** In share nothing architecture, hardware is homogenous, hence we can’t change/configure our hardware based on the activity we are performing. Hence, configuration needs to be a trade off with low average utilization.
    - Membership changes : If a node fails or user chooses to resize the system, larger amounts of data need to be reshuffled. If we want to increase storage size or process efficiency, change has to be made to the larger number of nodes. This limits elasticity and availability.
    - Upgrades : What if we want to upgrade software in every node ? If system hardware is heterogenous, an upgrade in one node may not work in another. We expect all nodes to be homogenous and tightly coupled (i.e. identical hardware).
* Multi-cluster shared data architecture (Snowflake) : To solve the above problems, Snowflake separated data storage layer and compute layers. Now as they both are independent of each other (Query processing/ Compute layer, Storage Layer), both can grow and shrink according to the activity needs.
  + But the problem here is that how do we coordinate b/w these two layers? How data compute layer should know which data to process ? how data storage should know which data to return ?
  + To solve these problems, Snowflake created another layer, i.e. “Cloud Services Layer”. This layer handles many things like authentication and access control, metadata storage, query optimization, security, etc.
  + So, how does the *Storage Layer* work?
    - Snowflake stores all data into databases and the database is a logical grouping of objects within a snowflake instance.
    - These objects are primarily Tables (Permanent, Temporary, Transient) and Views (Standard and Materialised).
    - These objects are part of one or more schemas.
    - You can store any structured relational data (standard SQL data types) or semi structured non-relational data (JSON Parquet, Avro, ORC XML) (variant data types).
    - Snowflake uses highly secure cloud storage to maintain your structured and semi structured data.
    - As data is loaded into the table following activities happen:
      * Snowflake convert them into optimised columnar compressed format(proprietary to snowflake ).
      * This optimised columnar compressed format brings a lot of data access efficiency (faster workload, low compute and storage cost).
      * Add Encrypt AES 256 Strong Encryption.
      * Based on the cloud platform data is loaded to the cloud storage (S3/Azure Blob/GCP Bucket), however, this is not visible to the user how it is stored and retrieved and overhead is taken care of by the snowflake.
      * These compressed and secure data is accessible only via SQL and there are no other means that it is accessible.
      * Data storage cost is calculated on the daily average amount of data (in bytes) (short lived or long lived tables).
      * If the time travel feature is enabled it is also part of the data storage cost.
    - Snowflake data storage costs are calculated based on **compressed size and amount stored – daily average.**
  + And how does *Compute Layer* work?
    - Compute layer is where queries (select queries, join queries, data loading, stored procedure, etc.) close are executed.
    - Before any query is executed, compute machines need to be provisioned and in Snowflake, they are called virtual warehouse (VWH).
    - This virtual warehouse (compute resource) has access to same data storage or data layer.
    - You can choose a virtual warehouse as per the workload required without any contention or performance degradation.
    - To create a virtual warehouse, you simply give a name and size (bigger the size more the computer power) and Snowflake handles all the provisioning and configuration of the underlying computer resources (in the case of AWS, that is EC2 instance and for Azure it is Azure VM).
    - The virtual warehouse can be scaled up and down at any time during the query execution without any hiccups. When rescaling is done, all the subsequent queries take the advantage of new size of the warehouse.
    - One multiple virtual warehouses (of different sizes) are running in parallel, Snowflake takes care of concurrency.
  + Let’s explore how *Cloud Service Layer* works:
    - It manages and coordinates the entire system.
    - This layer ensures and takes care of:
      * Authentication and authorization (via WebUI or Web connector or SnowSQL or native connector or MFA etc).
      * User and session management.
      * Query compilation, optimization and data caching.
      * Virtual warehouse management, coordinate data storage / updates and transaction management.
      * Metadata management (one of the core activity) and this feature supports:
        + Zero copy cloning.
        + Time travel
        + Data sharing
        + Caching.
      * Management maintain the life cycle of a query.
    - The service layer is highly scalable and distributed across multiple Availability Zones.
* Snowflake caching :
  + When we first run a query, it first reaches “*Cloud services layer “*  where any optimizations needed are checked. Then it goes to the “Compute *layer “ where* the query is processed and then finally it reaches the last layer i.e. “  *Storage Layer “* form where data is processed.
  + But if we run the same query second time, it won’t reach the third layer, instead it will directly return the result from the “*caching area*” of the “Cloud *services layer”* (not virtual warehouse)*.*
  + The above two statements can be confirmed by checking the activity of the warehouse, in the first case it will get active while in the second it won’t.
  + We can also use the “*caching area”*  of the virtual warehouse by altering the session cache by this query “ ALTER SESSION SET use\_cache\_result = false; “ . This will stop using the caching area of the “ *cloud services layer “* and start using the caching area of the virtual warehouse layer. NOTE : If the VWH is set to auto suspend it will purge the cache memory upon hitting the limit of suspension.
  + Lesson learnt:
    - Always use limit clause while using ‘ *Select \* ‘* queries in Snowflake to avoid unnecessary wastage of credits.
    - During development activity, always keep auto suspend to a higher time limit, at least 15 minutes.
    - Share your virtual warehouse when a group of users are working on the common tables to use shared caching area.
    - Never disable your cloud services result cache.
    - Reusing query result is free in Snowflake.
  + If there are any changes in the table, Snowflake will not use the cache memory to show the result.
  + This query will be cached (result cache) 24 hours from the time of last execution. If we execute the query on the 23rd hour, it will again get cached for the next 24 hours.
  + If two users use the same query, Snowflake will intelligently identify it and show the same result without processing the query once again. Snowflake leverages cached result across multiple users.
  + Snowflake will not charge to store the cache result.
  + Following are Snowflake cache layers:
    - Result cache – belongs to service layer. Can be disabled.
    - Local disk cache – belongs to compute layer (VWH). Can’t be disabled.
    - Remote cache – blob storage area like AWS S3. Can’t be disabled.
  + If there are manipulations in the query where we need to fetch a subset of the query result (stored in Cloud service layer cache), then it is fetched from the cache area of local disk i.e. virtual warehouse cache area.

# Chapter 4 : Clustering

* The process of grouping micro partitions is called ‘**CLUSTERING‘.**
* We can also provide clustering key in the query to customize clustering according to our needs:
  + E.g.: CREATE TABLE EMPLOYEE (type, name, country, date) CLUSTER BY (date);
    - In the above query the clustering will be performed according to the date.
* What are micro partitions?
  + When you load the files in Snowflake table tables are horizontally partitioned into large immutable files, which are equivalent to blocks or pages in a traditional database system.
  + Immutable table files which got generated are called as micro partitions or table file.
  + Micro-partitions are automatically performed on all Snowflake tables.
  + Table are transparently partitioned according to the ordering of the data it is inserted/loaded.
  + Within each file, the values of columns or attributes are grouped together and stored independently. This is referred to as **columnar storage**, and columns are heavily compressed using a scheme called PACs or hybrid columnar.
  + Each table file will be having a header file which contains the offset of each the table file. And it will hold other metadata information like **min max, distinct column offset,** etc.
  + Size of micro – partition when uncompressed ranges b/w **50-500 MB.**
* What is micro partition depth?
  + There will be overlap of data in micro partitions when thousands of tables are created.
  + The degree of overlap of micro partition is represented by micro partition depth in Snowflake.
  + There are certain micro partitions which are not overlapping with each other, such micro partitions are called constant micro partitions. Their depth is always one.
* Pre – cautions before applying clustering keys:
  + Clustering keys are not intended for all tables.
  + The size of the table, as well as the query performance of the table should dictate weather to define a clustering key for the table.
  + Table must be large enough to consist of a sufficiently large number of micro partitions, and the column(s) defined in the clustering key must provide sufficient filtering to select a subset of these micro – partitions.
  + In general, tables in the multi – terabyte (TB) range will experience the most benefit from clustering, particularly if DML commands are performed regularly on the tables.
* **Parsing, object resolution, access control**, and **plan optimization** are part ‘query life cycle ‘ in cloud services layer.
* Query execution plan will be submitted to ‘worker nodes in virtual warehouse layer’.
* All query information and statistics are stored for audits and performance analysis in **cloud services layer.**
* Based on the header file information, only required columns will be downloaded from remote disk.
* Snowflake scans micro partitions well if you use numeric values as filter than string value. Over numeric values snowflake will maintain stats internally about min, max values. Which helps to scan micro partitions easily.
* We can check the clustering information by running below command:
  + Select SYSTEM$CLUSTERING\_INFORMATION(‘*tableName’, ‘(cluster\_key\_name)’);*
    - Note : We can also give multiple cluster\_keys, also we can run this query to check how our clustering will look like if we want to perform clustering using same keys, although the result will not be always accurate(we do this to check the cardinality of columns to be put as cluster\_key).