2. SNOWFLAKE

2.1 Snowflake Data Warehouse

- It is a Cloud-based Analytical data warehouse.
- It has **multi-cluster shared architecture.**
- It runs on cloud infrastructure like AWS S3, Microsoft Azure, GCP
- It cannot be run on a private cloud or hosted infrastructure
- Snowflake uses OLAP as a foundational part of its database schema.

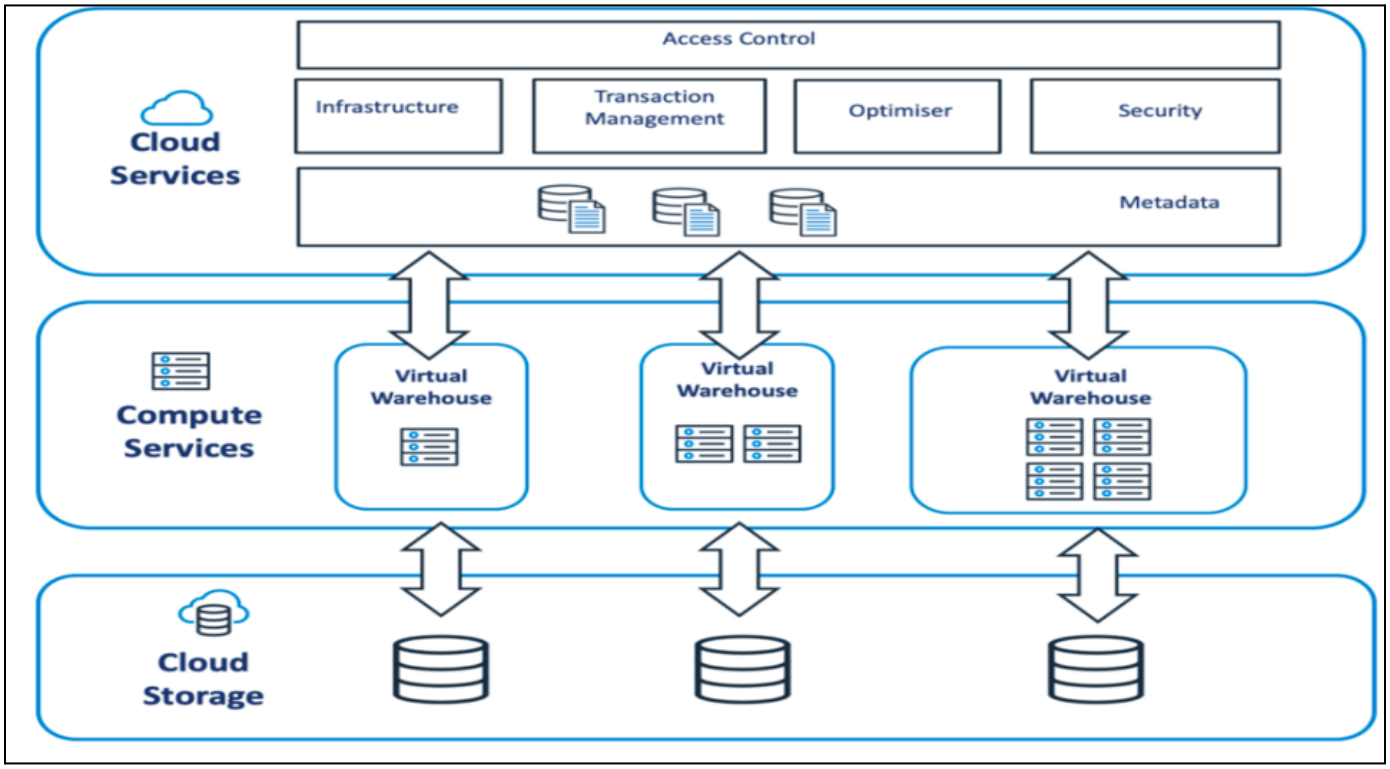
2.2 ETL tools for Snowflake

- Matillion
- Blendo
- Apache Airflow
- StreamSets
- Stitch
- Etleap
- Integrate.io
- Hevo Data
- Informatica

2.3 Access the Snowflake data warehouse

- **JDBC Drivers** (Java Database Connectivity)
- **ODBC Drivers** (Open Database Connectivity) by Microsoft
  - ➢ ODBC permits maximum interoperability, which means a single application can access different DBMS.
- Web User Interface
- Python Libraries
- SnowSQL Command-line Client

2.4 Snowflake Architecture



2.5 Snowpipe

- It is essentially a **COPY command** that sits on top of a cloud storage location.
- It is a serverless, scalable, and optimized data ingestion utility.
- Used for **continuously loading data** into Snowflake tables.
- Stages
    - ➢ Stages are **database objects** that store data files or reference data files in external storage locations.
- Serverless - It is a cloud-native development model that allows developers to build and run applications without having to manage servers. There are still servers in serverless, but they are abstracted away from app development. Serverless computing is a cloud computing execution model in which the cloud provider allocates machine resources on demand, taking care of the servers on behalf of their customers.

2.6 Zero Copy Cloning

- It provides a quick and easy way to create a copy of any table, schema, or the entire database without incurring any additional costs.

2.7 Streams

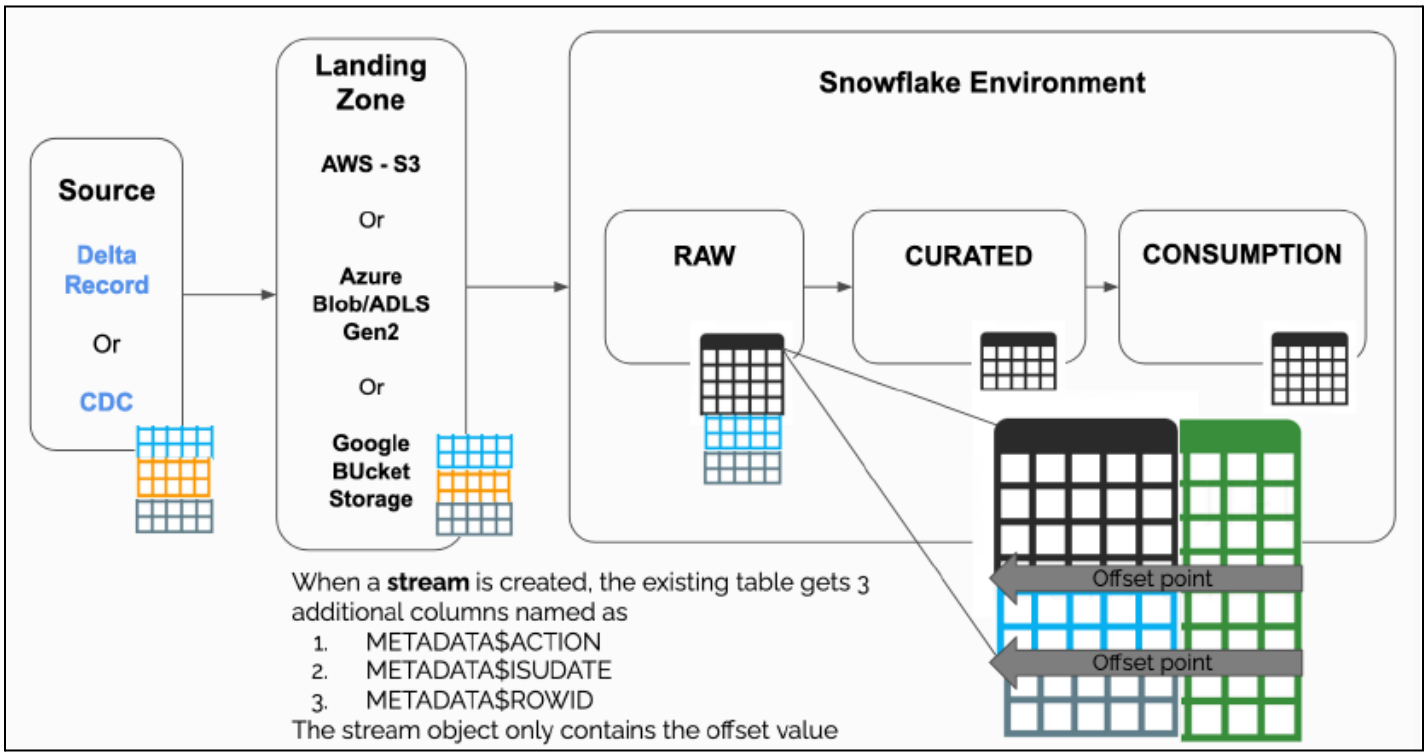
- It is a feature where you can track the changes happening in the data for a table (INSERT, UPDATES, DELETE)
- There are 3 additional columns that are added to the primary table where stream object is created (Metadata$**ACTION**, Metadata$**ISUPDATE**, Metadata$**ROW_ID**)
- Streams work by keeping track of an **offset**, a pointer that indicates the point in time since you last read the stream and it returns the **INSERTED** or **DELETED** rows from the table.
- This is referred to as the ***data retention*** of the table.
- **SHOW_INITIAL_ROWS** is true, which means the first time, the stream will return the rows that were present in the table.
- It supports 3 types of streams:
    - ➢ **Standard**: It supports tracking of all INSERT, UPDATES, DELETE
    - ➢ **Append only**: It supports tracking of all INSERT on the table.
        - FALSE: monitor for INSERTS, UPDATES, and DELETES.
        - TRUE: the stream will only return new rows means INSERT.
    - ➢ **Insert only**: It supports only external tables.
- **Stale**: A stream can become stale when the offset (the point in time since you last accessed the stream) becomes bigger than the data retention period. If this happens, you would need to recreate the stream. The offset will only change when a DML statement (insert, update, delete or merge) is used with a result.

2.7.1 CDC

- It means **change data capture.**
- A stream object records data manipulation language (DML) changes made to tables, including inserts, updates, and deletes, as well as metadata about each change, so that actions can be taken using the changed data.

2.7.2 Task

- An object that can schedule an SQL statement to be automatically executed as a recurring event.
- A task can have multiple tasks as its offsprings (task tree)



2.8 Questions

- How to get the historical data at a specific time using time travel in Snowflake?
  - ➢ By using AT clause along with SELECT query as follows:

```
Select * from table AT (TIMESTAMP => '2022-08-04 00:57:35.967'::timestamp);

Select * from table AT (STATEMENT => '<query_id>');

Create or replace new_table as
      (Select * from table BEFORE (STATEMENT => '<query_id>'));
```

- Show all dropped or delete schemas

```
Show SCHEMAS history;
```

- Retrieve the deleted databases in the Snowflake

```
Undrop database DB_name;
```

- Restore Objects in Snowflake

```
Undrop table table_name;
```

- Bulk loading data in Snowflake

```
COPY INTO table_name from @table_name
File_format=CSV;
```

- How to identify the Table Stage in Snowflake
  - ➢ `Table stages are referenced using @%table_name`

- Command used to upload file to table stage in Snowflake
  - ➢ **SnowSQL command is used.**

- Command to grant database to new user
  - ➢ `use role accountadmin;`
  - ➢ `create role junior_dba;`
  - ➢ `grant role junior_dba to user abhiyou;`
  - ➢ `grant usage on database citibike to role junior_dba;`

2.9 Clustering Key

- It is a subset of columns in a table or an expression that is explicitly used to co-locate the data in the table in the same micro-partition.

2.10 SnowPark

- It allows you to write Snowflake code in your preferred way, and execute it directly within Snowflake.
- Example use cases:
  - ➢ Data Transformation
  - ➢ ML Scoring / Inference to operationalize ML models in data pipelines
  - ➢ ETL system
  - ➢ Data apps

2.11 Stage

- Creates a new named internal or external stage to use for loading data from files into Snowflake tables and unloading data from tables into files.
  - **Internal Stage** - Stores data files internally within Snowflake. Internal stages can be either permanent or temporary.
  - **External Stage** - References data files stored in a location outside of Snowflake.
    - Ex: Amazon S3 bucket, Google Cloud Storage bucket, Microsoft Azure containers

2.12 Snowflake Cache Layers

- **Result Cache**: This holds the results of every query executed in the past 24 hours. These are available across virtual warehouses, so query results returned to one user are available to any other user on the system who executes the same query, provided the underlying data has not changed.
- **Local Disk Cache**: This is used to cache data used by SQL queries. Whenever data is needed for a given query it's retrieved from the Remote Disk storage, and cached in SSD and memory.
- **Remote Disk**: This holds long-term storage. This level is responsible for data resilience, which in the case of Amazon Web Services, means 99.999999999% durability. Even in the event of an entire data center failure.

2.13 Fail-safe

- It is used for disaster recovery of historical data.
- To reduce the risk factor, it performs complete and incremental data backups on a regular basis.
- This process consumes more storage space.

- The process of recovering data is expensive, takes time, and requires company downtime.
- Transient tables are similar to permanent tables with the key difference that they do not have a Fail-safe period. Transient tables persist until explicitly dropped and are available to all users with the appropriate privileges.

2.14 Loading data - view

- Create a database and table
- Create an external stage
- Create a file format for the data
- Copy into

2.15 Queries

- use role sysadmin;

```
use warehouse compute_ph;
use database wether;
use schema public;
```

- // create a new table

```
create table json_weather_data (v variant);
```

- // create a new stage

```
create stage nyc_weather
Url = 's3://snowflake-workshop-lab/weather-nyc';
list @nyc_weather;
```

-
```
copy into json_weather_data
from @nyc_weather
file_format = (type=json);
```

- // convert json file to view table (to more readable form)

```
create view json_weather_data_view as
select
v:time::timestamp as observation_time,
v:city.id::int as city_id,
v:city.name::string as city_name,
v:city.country::string as country,
v:city.coord.lat::float as city_lat,
v:city.coord.lon::float as city_lon,
v:clouds.all::int as clouds,
(v:main.temp::float)-273.15 as temp_avg,
(v:main.temp_min::float)-273.15 as temp_min,
(v:main.temp_max::float)-273.15 as temp_max,
v:weather[0].main::string as weather,
v:weather[0].description::string as weather_desc,
v:weather[0].icon::string as weather_icon,
v:wind.deg::float as wind_dir
v:wind.speed::float as wind_speed
From  json_weather_data
Where city_id = 5128638;
```

- // verify

```
Select * from json_weather_data_view
where date_trunc('month', observation_time) = '2018-01-01'
limit 20;
```

- SPECIAL Query
  - // undrop table

```
undrop table json_weather_data;
```

  - // get the historical data by session

```
set query_id =
(select query_id from table
(information_schema.query.history_by_session (result_limit => 5))
where query_text like 'update%' order by start_time limit 1);

create or replace table trips as
(select * from trips before (statement => $query_id));
```

  - // create new role

```
USE ROLE ACCOUNTADMIN;
CREATE ROLE junior_dba;
GRANT ROLE junior_dba TO USER abhiyou;
GRANT USAGE ON DATABASE citibike TO ROLE junior_dba;

CREATE USER jake PASSWORD = 'Welcome123' DEFAULT_ROLE =
ACCOUNTADMIN MUST_CHANGE_PASSWORD = TRUE;
GRANT ROLE SYSADMIN TO USER jake;

REVOKE ROLE SYSADMIN FROM USER john;
GRANT ROLE junior_dba TO ROLE analyst;
```

- // drop share database

```
drop share if exists trips_share;
```

2.16 Matillion

- ➔ Matillion ETL is an ETL/ELT tool built specifically for cloud database platforms including Amazon Redshift, Google BigQuery, Snowflake, and Azure Synapse.
- ➔ Agents
  - ◆ Agents are required only for setting up CDC pipelines.
  - ◆ An agent runs as a containerized image within your private cloud and/or your on-premises technology stack. The agent requires access to your source dataset, your cloud data lake or data platform destination, and your secrets management service.
- ➔ Environments
  - ◆ Create Stage
  - ◆
- ➔ Orchestration jobs
  - ◆ Orchestration jobs are primarily concerned with DDL statements (especially creating, dropping, and altering resources), and loading data from external sources.
  - ◆ DDL
    - Create External Table
    - Refresh External Table
    - Create Table
    - SQL Script
    - Truncate Table
  - ◆ Load/Unload
    - On-Premise & Cloud DBs
      - Cassandra Query
      - Couchbase Query
      - Google BigQuery
      - MongoDB Query
      - Salesforce Query
- ➔ Transformation jobs
  - ◆ Transformation jobs are used for transforming data that already exists within tables. This includes filtering data, changing data types, and removing rows.
  - ◆