

Snowflake



Develop a passion for learning.

COURSE OBJECTIVES

Attendees will leave

- understanding the Snowflake architecture
- knowing how to load and transform data
- knowing how to evaluate query constructs, DDL and DML Operations
- managing identity and access
- learning best practices for working with data
- employing Snowflake's method for continuous data protection
- understanding Snowflake security
- preparing for the Snowflake Core certification

Agility



Data Warehousing Overview

- Data warehousing evolution
- Cloud data warehousing
- Adapting to increasing demands for data access

Architecture and Overview

- Technical Overview
- Cloud Services Layer
- Compute Layer
- Storage Layer

Data Movement

- Data Loading
- Data Unloading
- Best Practices
- Data Sharing



Objects and Commands

- Query Constructs
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Local only resources

SQL Support for Data Analysis

- SQL Support and Query Best Practices
- SQL Analytic Functions
- High Performing Estimation Functions
- UDF and Stored Procedure
- Demo Query Profile

Managing Security

- Data Encryption
- Authentication
- Role-Based Access Control

Observability



Semi-structured data

- Working with semi-structured data
- Queries
- Data Optimization

Caching

- Caching Features
- Performance Improvements
- Cost Optimization

Clients and Ecosystem

- Clients
- Connectors
- SnowSQL

Launch



Security

- Continuous Data Protection
- Time Travel
- Cloning

Performance and Concurrency

- Query Profile
- Micro-Partitions
- Data Clustering
- Scaling a Virtual Warehouse

Account and Resources

Management and Monitoring

- System Resource Usage and Billing
- Managing Virtual Warehouses
- Workload Independence and Segmentation
- Resource Monitors
- Information Schema and Account Usage

MEET THE INSTRUCTOR

George Niece

Digital Transformation Consultant with a background in AppDev, Digital Data Supply Chain, DevOps, InfoSec, Chaos Engineering and Enterprise Architecture. Focused on cloud-native technologies: automation, containers & orchestration

Twitter
[@georgeniece](https://twitter.com/georgeniece)

LinkedIn
[Linkedin.com/in/GeorgeNiece](https://www.linkedin.com/in/GeorgeNiece)

Email
George.Niece@DigitalTransformationStrategies.net



Expertise

- Cloud
- Automation
- Elastic Stack
- DevSecOps
- Microservices
- Snowflake

INTRODUCTIONS

- Which statement best describes your Snowflake experience?
 - a. I am ***currently working*** with Snowflake on a project/initiative
 - b. I ***expect to work*** with Snowflake on a project/initiative in the future
 - c. I am ***here to learn*** about Snowflake outside of any specific work related project/initiative
- Rate yourself on (1 New Joiner – 5 SME) on
 - Snowflake
 - DBA
 - Data Warehouse
 - SQL
- Expectations for course (please be specific, if possible)
- Why are you here?
- What is your favorite dessert?

LOGISTICS



noon



- **Class Hours:**
- Start time is 2:00am CST
- End time is 10:00am CST
- Class times may vary slightly for specific classes
- Breaks mid-morning and afternoon (10-15 minutes)
- **Lunch:**
- Lunch is typically 5:00am CST to 6:15 am CST
- Yes, 1 hour and 15 minutes
- Extra time for email, phone calls, or simply a walk.



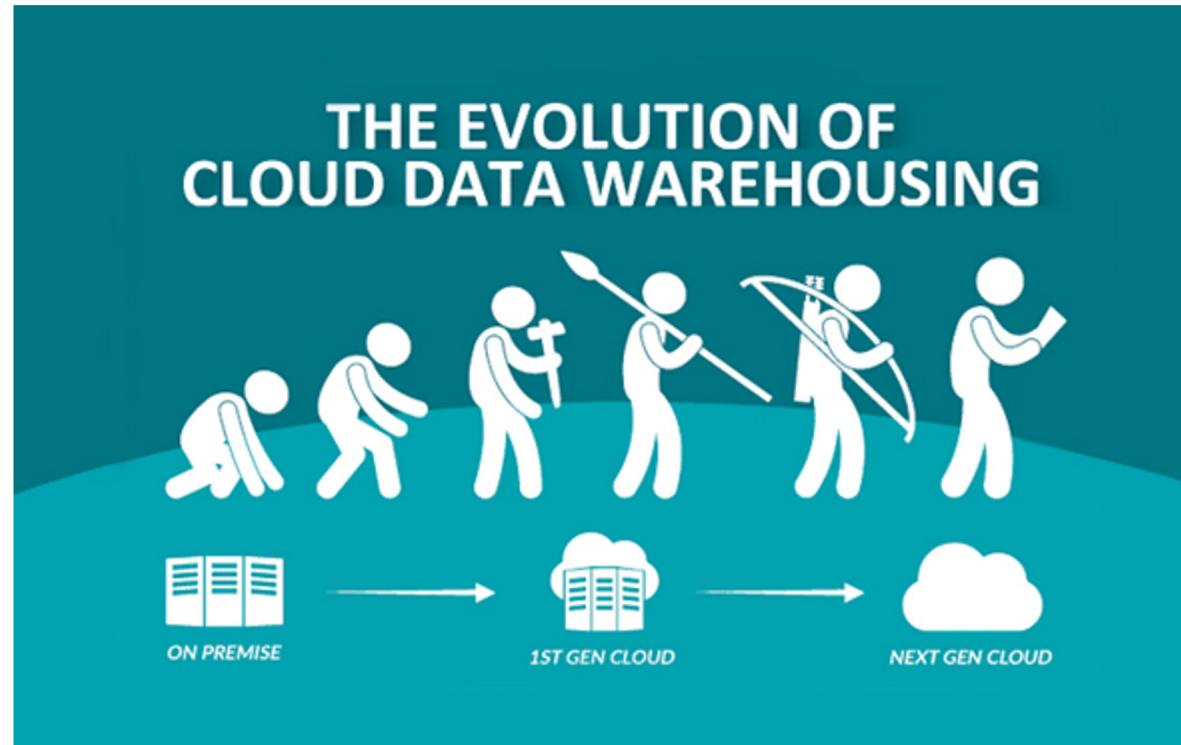
- **Telecommunication:**
- Turn off or set electronic devices to vibrate
- Reading or attending to devices can be distracting to other students

- **Miscellaneous**
- Courseware
- Bathroom

Data Warehouse Overview



DATA WAREHOUSING EVOLUTION



DATA WAREHOUSE CHARACTERISTICS

Warehouses are required for queries, as well as all DML operations, including loading data into tables. A warehouse is defined by its size, as well as the other properties that can be set to help control and automate warehouse activity.

Warehouses can be started and stopped at any time. They can also be resized at any time, even while running, to accommodate the need for more or less compute resources, based on the type of operations being performed by the warehouse.

SNOWFLAKE DATA WAREHOUSE

A **data warehouse** is a relational database that is designed for analytical rather than transactional work.

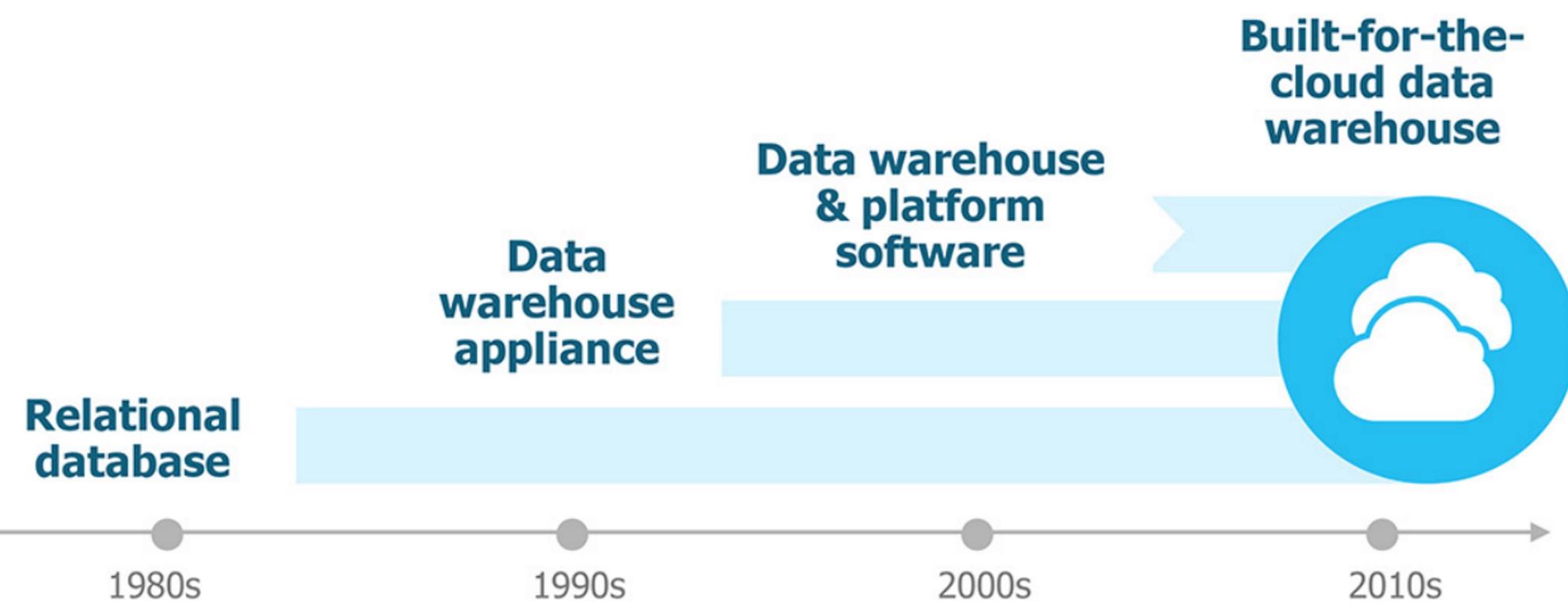
It collects and aggregates data from one or many sources so it can be analyzed to produce business insights.

It serves as a federated repository for all or certain data sets collected by a business's operational systems. **Snowflake is a cloud data warehouse**

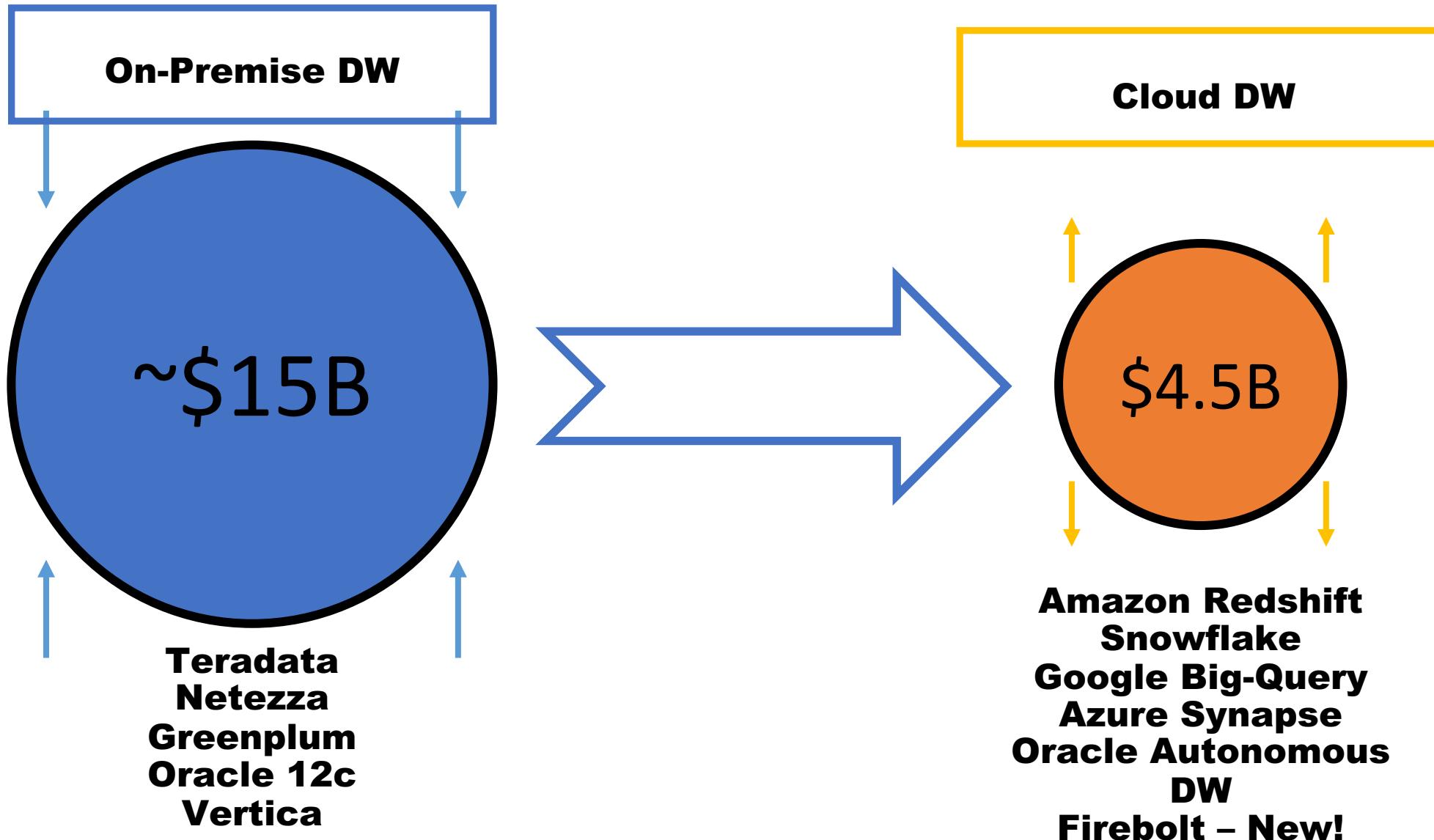
SNOWFLAKE DATA WAREHOUSE

- Founded in 2012 – product launched in 2015
- Runs on AWS, GCP & Azure
- Raised \$1.4B prior to IPO
- Snowflake had 4,532 customers as of May'2021
- Completed IPO in September
- Largest software IPO
- Currently valued at ~75B\$
- ~3,300 employees

CLOUD DATA WAREHOUSING

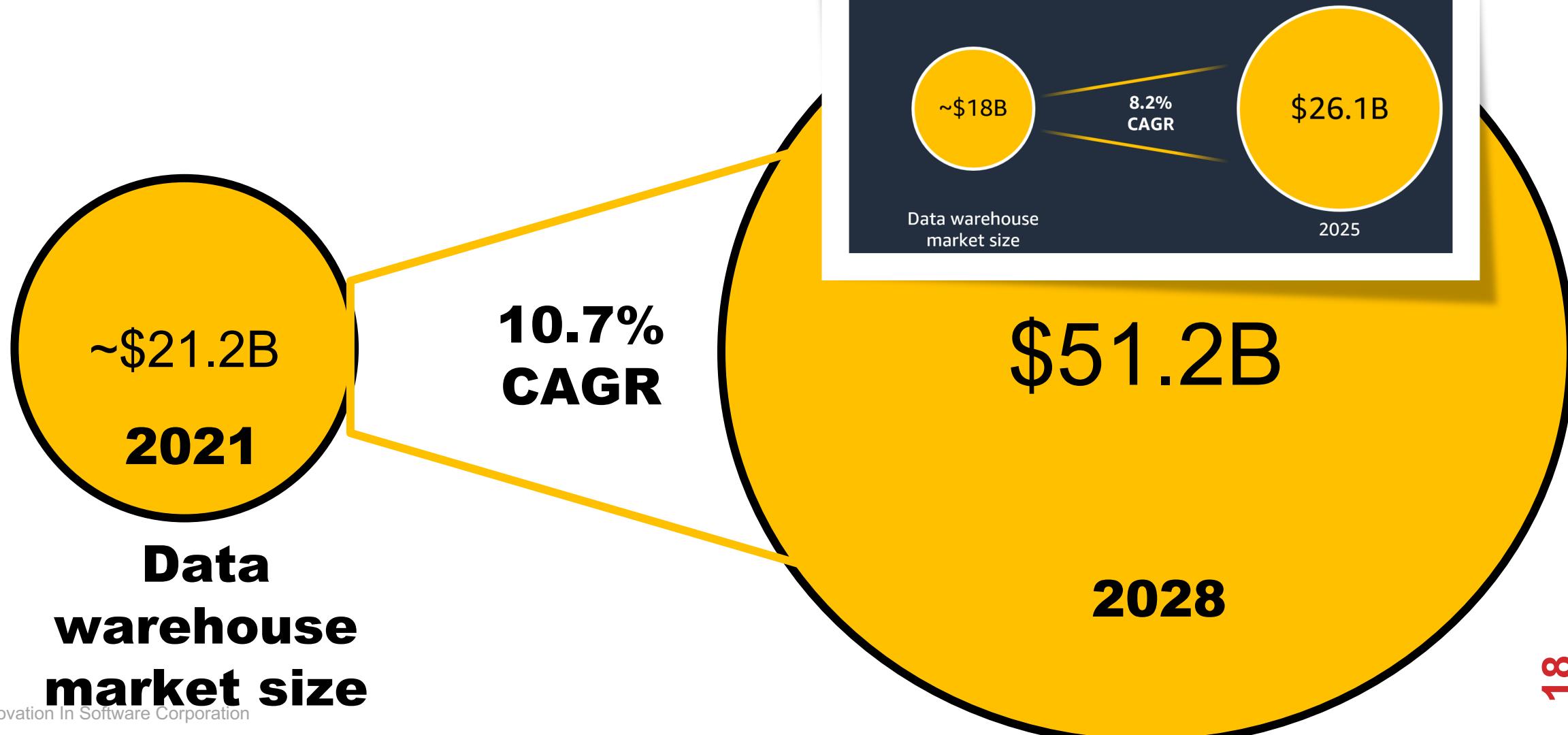


CLOUD DATA WAREHOUSING



DATA WAREHOUSE MARKET (2021)

Source: Allied Market Research



ADAPTING TO INCREASING DEMANDS FOR DATA ACCESS

5.8B

endpoints in use in
2020 at 21% growth

- Gartner

\$200B

data center market in
2021 at 6% growth

- Gartner

\$335B

global public cloud
market by 2022 at
12% growth

- Gartner

40%

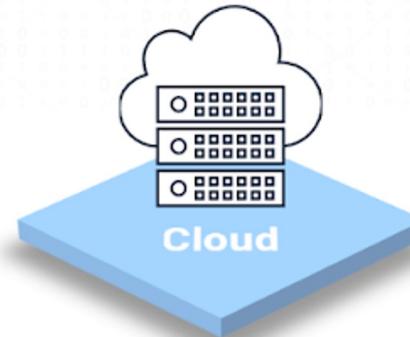
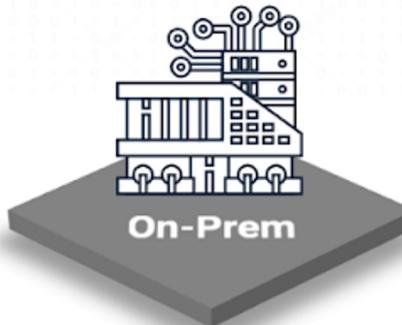
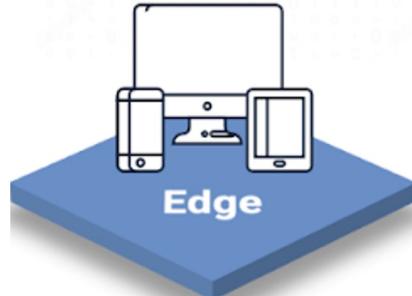
of companies in the
cloud will use 2+
providers

- Gartner

70%

think cloud is more
complex due to local
privacy regulations

- Privacera



WHAT CUSTOMERS DO WITH CLOUD DW

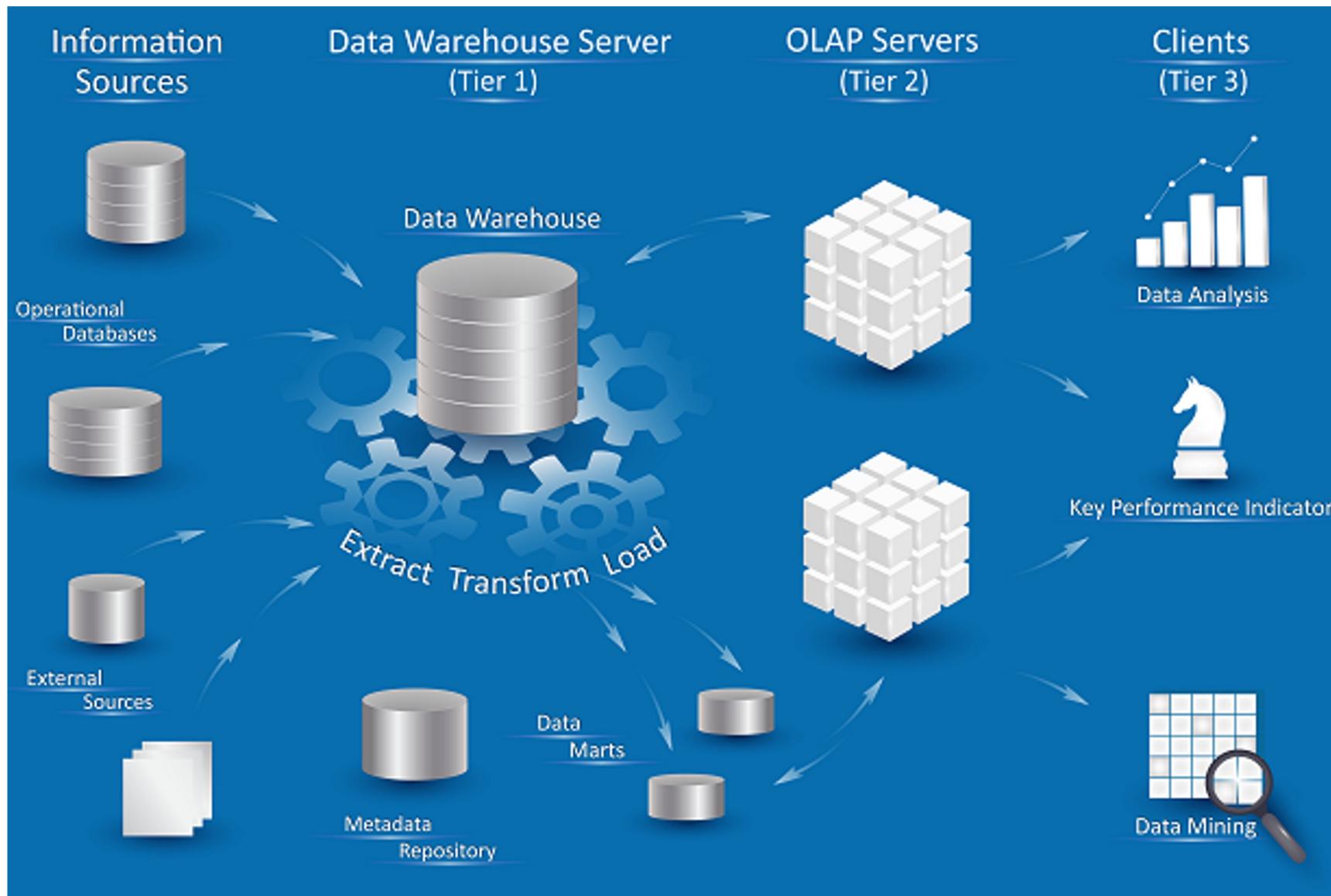
- Gaming company **replaced Hadoop + SQL database with Snowflake**
- Consumer retailer modernizing DW by **replacing the legacy appliance with Snowflake**
- Market research company **consolidated data marts** to reduce costs and data silos
- Mobile analytics **company shares live data** with clients

DATA WAREHOUSE SELECTION CRITERIA

When you're in the market for a **data warehouse**, a **checklist of criteria** will help determine which alternative best meets your needs.

- Supports Existing Skills, Tools, and Expertise
- Saves Your Organization Money
- Provides Data Resiliency and Recovery
- Secures Data at Rest and in Transit
- Streamlining the data pipeline
- Optimizes Your Time to Value

BUSINESS INTELLIGENCE



DATA WAREHOUSE VS. DATABASE

A data warehouse focuses on collecting data from multiple sources to facilitate broad access and analysis. They specialize in data aggregation and providing a longer view of an organization's data over time.

A data warehouse is optimized to store large volumes of historical data and enables fast and complex querying of that data. Standard operational databases focus on transactional functions such as real-time data updates for ongoing business processes.

DATA WAREHOUSE KEY FUNCTIONS

Data warehousing has two key functions.

First, it serves as a **historical repository** for integrating the information and data that is needed by the business, which may come from a variety of different sources.

Second, it serves as a **query execution and processing engine** for that data, enabling end users to interact with the data that is stored in the database.

COMPLEX QUERY EXECUTION ENGINE

Complex queries are very difficult to run without a temporary pause of database update operations. A frequently paused transactional database will inevitably lead to data errors and gaps.

Therefore, a data warehouse serves as a **separate platform for aggregation** across multiple sources and then for analytics tasks across those diverse sources. This separation of roles allows databases to remain focused on purely transactional jobs without interruption.

DIGITAL DATA SUPPLY CHAIN

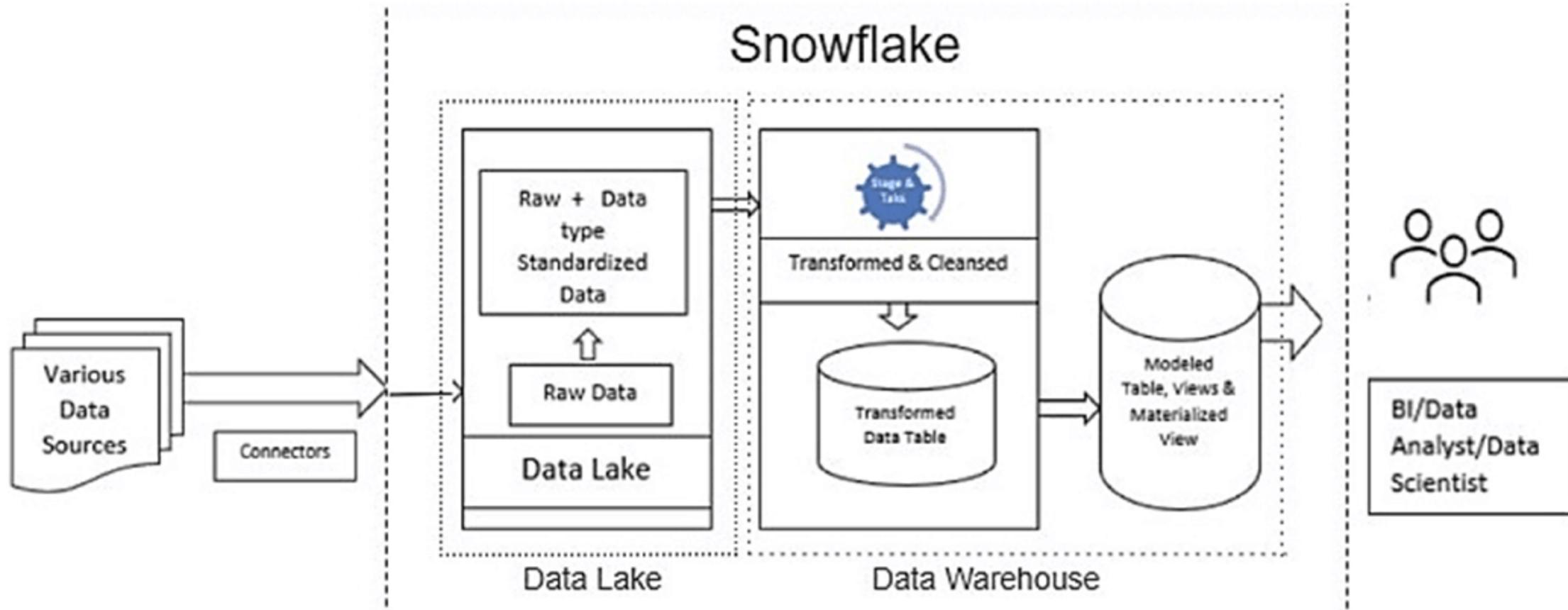


DATA WAREHOUSE COMPONENTS

A data warehouse usually consists of data sources from operational and transactional systems (ERP, CRM, finance apps, IoT devices, mobile and online systems) as well as:

- A data staging area for aggregation and cleaning
- A presentation/access area where data is warehoused for analytics (querying, reporting) and sharing
- A range of data tool integrations or APIs (BI software, ingestion and ETL tools, etc.).

DATA WAREHOUSE, LAKE, LAKEHOUSE



GETTING STARTED WITH CDW

Snowflake gives us a free Trial account with 400 usage credits to explore Cloud Data Warehouse.

Choose your Snowflake edition*

Standard

A strong balance between features, level of support, and cost.

Enterprise

Standard plus 90-day time travel, multi-cluster warehouses, and materialized views.

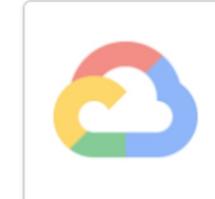
Business Critical

Enterprise plus enhanced security, data protection, and database failover/fallback.

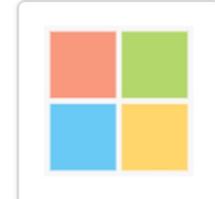
Choose your cloud provider*



Amazon Web Services



Google Cloud Platform



Microsoft Azure

US East (Northern Virginia) ▾

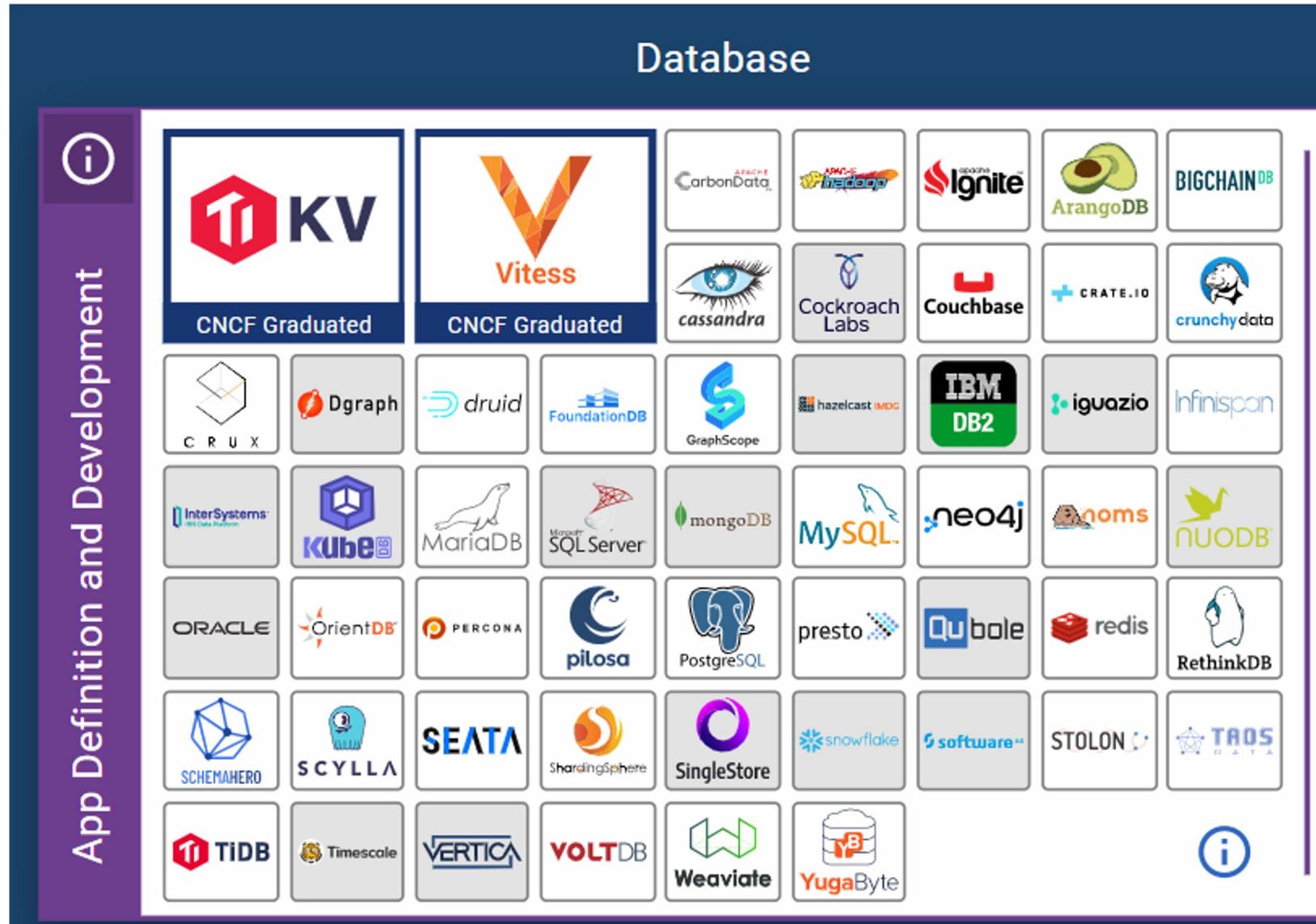
- Check here to indicate that you have read and agree to the terms of the Snowflake Self Service On Demand Terms.

GET STARTED

SNOWFLAKE CDW JUMPSTART

- Snowflake has free courses to get started.
- Snowflake has a simplified UI designed to help learners focus effectively during the learning process.
- Snowflake does not require an up-front purchase. Instead, it has free trial credits and then converts to a pay-as-you-go model.

SNOWFLAKE IN CLOUD-NATIVE CONTEXT



SNOWFLAKE ACCOUNT

The screenshot shows the Snowflake Worksheets interface. At the top, the URL is app.snowflake.com/pzqcza/if42890/worksheets. On the left, a sidebar menu includes 'George Niece ACCOUNTADMIN' (GN), 'Worksheets' (selected), 'Dashboards', 'Apps', 'Data', and 'Marketplace'. Below this is a dropdown for 'PZQCZHA' with 'IF42890' selected, showing 'SNOWFLAKECOURSEUPDATE - aws US East (Ohio)'. A 'Sign Into Another Account' link is also present. At the bottom, a footer shows 'IF42890' and a blue snowflake icon. The main area is titled 'Worksheets' with tabs for 'Recent', 'Shared with me', 'My Worksheets', and 'Folders'. The 'Recent' tab is selected, displaying three worksheets: '2023-07-17 10:07am Benchmarking Tutorials', 'Tutorial 1: Sample queries on TPC...', and 'Tutorial 2: Sample queries on TPC-D...'. To the right, account details for 'IF42890' are shown: Organization (PZQCZHA), Cloud (Amazon Web Services), Region (US East (Ohio)), Edition (Enterprise), and Locator (FS86256). A red number '32' is in the bottom right corner.

GN George Niece ACCOUNTADMIN

Worksheets

Recent Shared with me My Worksheets Folders

TITLE

2023-07-17 10:07am Benchmarking Tutorials

Tutorial 1: Sample queries on TPC-... Benchmarking Tuto...

Tutorial 2: Sample queries on TPC-D... Benchmarking Tuto...

PZQCZHA

IF42890 ✓

SNOWFLAKECOURSEUPDATE - aws US East (Ohio)

→] Sign Into Another Account

IF42890

32

Organization	PZQCZHA
Cloud	aws Amazon Web Services
Region	US East (Ohio)
Edition	Enterprise
Locator	FS86256

SNOWFLAKE ORGANIZATION NAME

For users who sign up for a Snowflake account using the self-service/trial option, an organization is automatically created with a **system-generated name** when the account is created.

For entities who work directly with Snowflake personnel to set up accounts, Snowflake can assign the organization a **custom name**.

This custom name must be unique across all other organizations in Snowflake. The name must **start with a letter and can only contain letters (lowercase and uppercase) and numbers. The name cannot contain underscores or other delimiters.**

SNOWFLAKE ACCOUNT IDENTIFIERS

The account identifier for an account in your organization takes one of the following forms, depending on where and how the identifier is used:

orgname-account_name (for most URLs and other general purpose usage)

orgname-account-name (for scenarios/features where underscores in an account name are not supported)

orgname.account_name (for SQL commands and operations)

Where:

orgname is the name of your Snowflake organization.

account_name is the unique name of your account within your organization.

TRIAL ACCOUNT DETAILS

The trial accounts have pregenerated strings for orgname, account_name and locator:

Trial accounts have random characters for orgname **PZQCZHA.IF42890**

Trial accounts have random characters for account_name **PZQCZHA.IF42890**

Trial account locator takes the form **FS86256**

Trial account URL takes the form

<https://fs86256.us-east-2.aws.snowflakecomputing.com>

Combining the account locator, cloud provider region and cloud provider type

ACCOUNT IDENTIFIERS USAGES

Account identifiers are required in Snowflake wherever you need to specify the account you are using, including:

- URLs for accessing any of the Snowflake web interfaces.
- SnowSQL and other clients (connectors, drivers, etc.) for connecting to Snowflake.
- 3rd-party applications and services that comprise the Snowflake ecosystem.
- Security features for protecting Snowflake internal operations and communication/interaction with external systems.
- Global features such as Secure Data Sharing and Replication and Failover/Failback.

SNOWFLAKE ACCOUNT URLs

If you have accounts with the same name in different regions, the cloud and region names are prepended to the account name in the new URL format.

For example, if the organization name is ACME, and there are two accounts named TEST, one in the AWS US East 2 region and the other in the Azure West US 2 region, the URLs will look as follows:

New account URL 1: <https://test.us-east-2.aws.snowflakecomputing.com>

New account URL 2: <https://test.west-us-2.azure.snowflakecomputing.com>

Legacy account URL 1: https://acme-test_aws_us_east_2.snowflakecomputing.com

Legacy account URL 2: https://acme-test_azure_west_us_2.snowflakecomputing.com

Experiment



Content
#00 - Getting Started

Scripts
N/A

SNOWFLAKE WORKSHEET

The screenshot shows a Snowflake Worksheet interface. The top navigation bar includes 'Worksheets' (selected), 'Tutorial 1: Sample queries ...', and a '+' button. The left sidebar shows 'Databases' and 'Worksheets' (selected). A search bar and a folder named 'Benchmarking Tutorials' containing several query files are visible. One file, 'Tutorial 1: Sample queries on TPC-H...', is highlighted with a blue background and white text. The main workspace displays a query script with numbered lines:

```
-- 29
-- 30
-- 31 use schema snowflake_sample_data.tpch_sf1;
-- 32 -- or tpch_sf100, tpch_sf1000
-- 33
-- 34 SELECT
-- 35 l_returnflag,
-- 36 l_linenumber,
-- 37 sum(l_quantity) as sum_qty,
-- 38 sum(l_extendedprice) as sum_base_price,
-- 39 sum(l_extendedprice * (1-l_discount))
```

SNOWFLAKE WORKSHEET FROM FILE

The screenshot shows the Snowflake Worksheet interface. At the top, the URL is app.snowflake.com/pzqczha/if42890/w2l1Z7Moz2EN/query. The main area displays a list of worksheets under the 'Worksheets' tab. A context menu is open over the worksheet titled 'Tutorial 1: Sample queries ...'. The menu options are: Rename, Go to Parent Folder, Import SQL from File, Move to (with a submenu arrow), Duplicate, Format Query, Delete Worksheet, and Show Shortcuts. The 'Import SQL from File' option is highlighted. On the right side of the interface, there is a user profile section showing 'ACCOUNTADMIN' with roles 'COMPUTE_WH' and 'SI'. Below the user info, there is a dropdown for 'SAMPLE_DATA.TPCDS_SF10TCL' and a 'Settings' button.

SNOWFLAKE QUERY GETTING STARTED

The screenshot shows the Snowflake interface with a query editor. The top navigation bar includes 'Tutorial 1: Sample queries ...' (selected), '+', 'PREVIEW', 'ksheets' (selected), and 'Share'. The main area shows a query titled 'SNOWFLAKE_SAMPLE_DATA' with the following code:

```
40     as sum_disc_price,
41 sum(l_extendedprice * (1-l_discount) *
42      (1+l_tax)) as sum_charge,
43 avg(l_quantity) as avg_qty,
44 avg(l_extendedprice) as avg_price,
45 avg(l_discount) as avg_disc,
46 count(*) as count_order
47
48 FROM
49 lineitem
50 WHERE
51 l_shipdate <= dateadd(day, -90, to_date('1998-12-01'))
52 GROUP BY
53 l_returnflag,
54 l_linenumber
55 ORDER BY
```

A tooltip for the 'Run All' button is visible, stating 'ctrl + shift + enter'.

QUERY RESULTS

Results Data Preview

✓ [Query ID](#) [SQL](#) 79ms  1 rows

Filter result... [!\[\]\(8a553b1d6dcefe437a312073e97c2384_img.jpg\)](#) [Copy](#)

Row	status
1	Statement executed successfully.

QUALIFIED AND UNQUALIFIED QUERIES

The screenshot shows a Snowflake worksheet interface. On the left, the sidebar displays the database structure:

- Databases: SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA
- Workshops: None (0)
- All Objects: SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA, INFORMATION_SCHEMA, TPCDS_SF100TCL
- TPCDS_SF100TCL: Tables (CALL_CENTER, CATALOG_PAGE)
- CALL_CENTER is selected.

The main area shows a query being run:

```
1 use database SNOWFLAKE_SAMPLE_DATA;
2
3 use schema tpcds_sf100tcl;
4
5 select cc_name, cc_manager from call_center;
```

The query results are displayed in a table:

CC_NAME
1 NY Metro
2 Mid Atlantic

SNOWFLAKE PROFILE DEFAULTS

Profile

Profile photo



Upload



Username

SNOWFLAKECOURSEUPDATE

First Name

George

Last Name

Niece

Password

.....

Email

snowflake.jedi@gmail.com

Default role & warehouse

ACCOUNTADMIN • COMPUTE_WH

Language

English



Notifications



Notify when queries finish in the background

Experiment



Content
#01-Snowflake-
Foundation.pdf

Scripts
foundation_experi
ment_scripts.sql

POP QUIZ:

Data Warehousing



Follow the link for Data Warehousing Pop Quiz:

<https://forms.gle/sdumUkHPqL683wmv9>

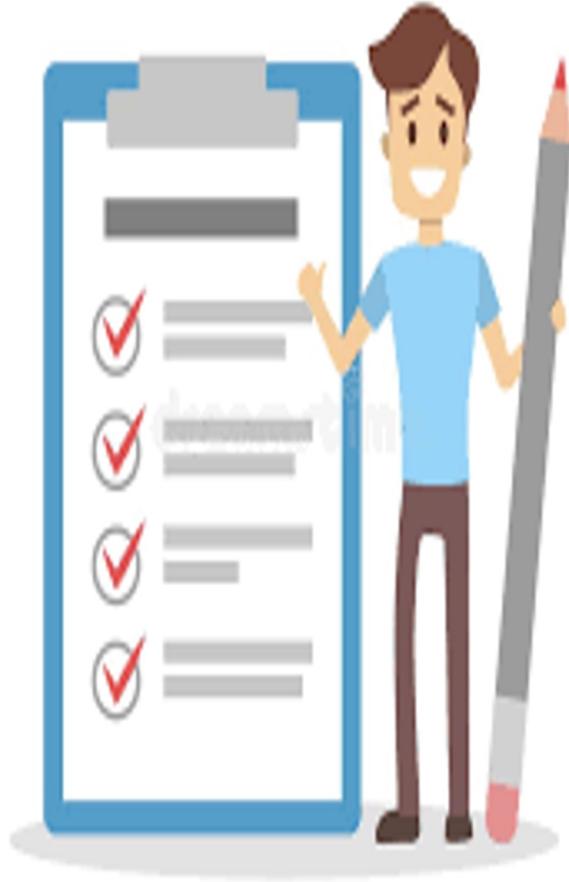


Snowflake Core Certification



What We have Covered so far

- Data Warehouse Introduction
- Data Warehouse Market
- Data Warehouse vs Database
- Data Warehouse Components
- Snowflake CDW
- Snowflake Account
- Snowflake Worksheet
- Snowflake Profile



CORE CERTIFICATION CRITERIA

Domain	Estimated Percentage Range
Account and Security	10 - 15%
Virtual Warehouses	15 - 20%
Data Movement	11 - 20%
Performance Management	5 - 10%
Snowflake Overview and Architecture	25 - 30%
Storage and Protection	10 - 15%

DOMAIN: ACCOUNT AND SECURITY

Snowflake Account Management

- Account usage
- Information schema

Security Principles

- Multi-factor Authentication (MFA)
- Data Encryption
- Network Security & Policies
- Access Control
- Federated Authentication
- Single Sign-On (SSO)

Snowflake Entities and Roles

- Outline how privileges can be granted and revoked
- Explain Role Hierarchy and Privilege Inheritance

Snowflake Data Governance

- Data masking
- Account usage views
- External Tokenization

DOMAIN: VIRTUAL WAREHOUSES

Outline compute principles

- Credit usage & billing
- Concurrency
- Caching

Virtual Warehouse best practices

- Scale up vs scale out
- Types of virtual warehouses
- Management/monitoring

DOMAIN: DATA MOVEMENT

Data Loading Commands

- COPY
- INSERT
- PUT
- GET
- VALIDATE

Bulk vs Continuous Loading

- COPY
- Snowpipe

Data Load Best Practices

- File size
- Folders

Data Unloading

- Data unloading formats
- Data unloading best practices

Semi-structured data handling

- Supported file formats
- VARIANT column
- Flattening the nested structure

DOMAIN: PERFORMANCE MANAGEMENT

Snowflake storage performance management

- Clustering
- Materialized views
- Search optimization
- Best practices

Snowflake virtual warehouse performance management

- Query performance and analysis
- Query profiles
- Query history
- SQL optimization
- Caching
- Best practices

DOMAIN: ARCHITECTURE

Snowflake's Cloud data platform architecture

- Data Types
- Optimizer
- Continuous data protection
- Cloning
- Types of Caching
- Web Interface (UI)
- Data Cloud/Data Sharing/Data Marketplace/Data Exchange

Snowflake data sharing capabilities

- Account types
- Data Marketplace & Exchange
- Access Control options
- Shares

Snowflake partner ecosystem

- Cloud Partners
- Connectors

DOMAIN: ARCHITECTURE

Snowflake architecture layers

- Storage Layer
- Compute Layer
- Cloud Services Layer

Snowflake by Edition

Security
Pricing
Features

Snowflake catalog and objects

- Database
- Schema
- Tables Types
- View Types
- Data Types
- External Functions

DOMAIN: STORAGE AND PROTECTION

Snowflake Storage concepts

- Micro partitions
- Metadata Types
- Clustering
- Data Storage
- Stage Types
- File Formats
- Storage Monitoring

Continuous Data Protection with Snowflake

- Time Travel
- Fail-safe
- Data Encryption
- Cloning

Architecture and Overview

Kickstart



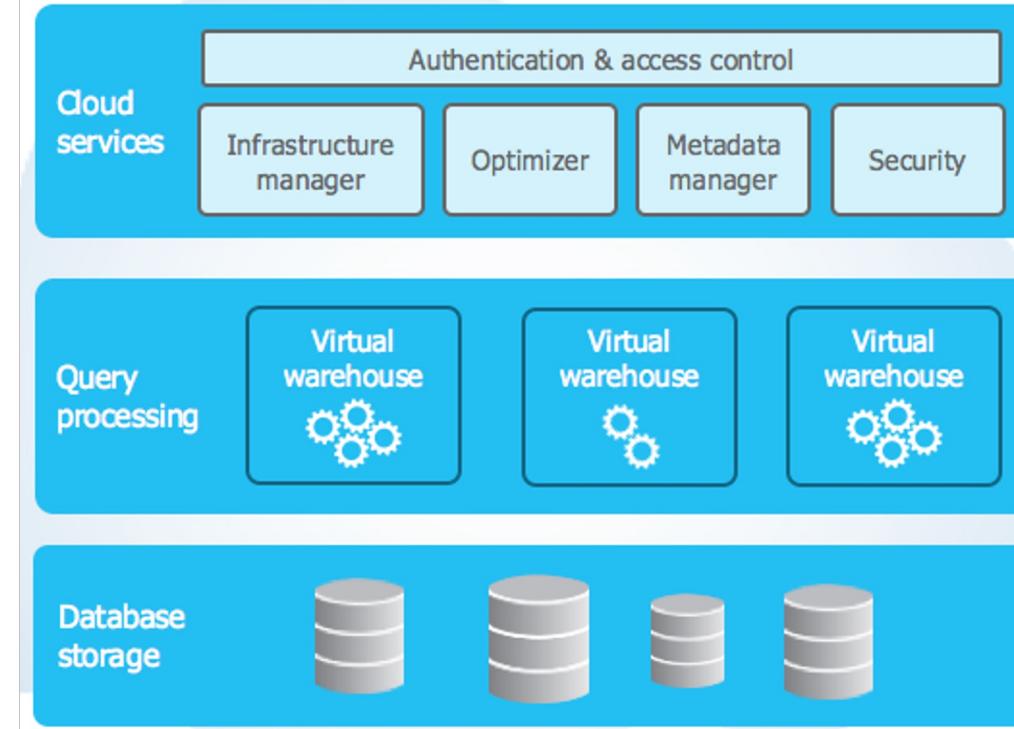
2

ARCHITECTURE OVERVIEW

- Data Platform as a Cloud Service
- Snowflake Architecture
- Database Storage
- Query Processing
- Cloud Services
- Connecting to Snowflake

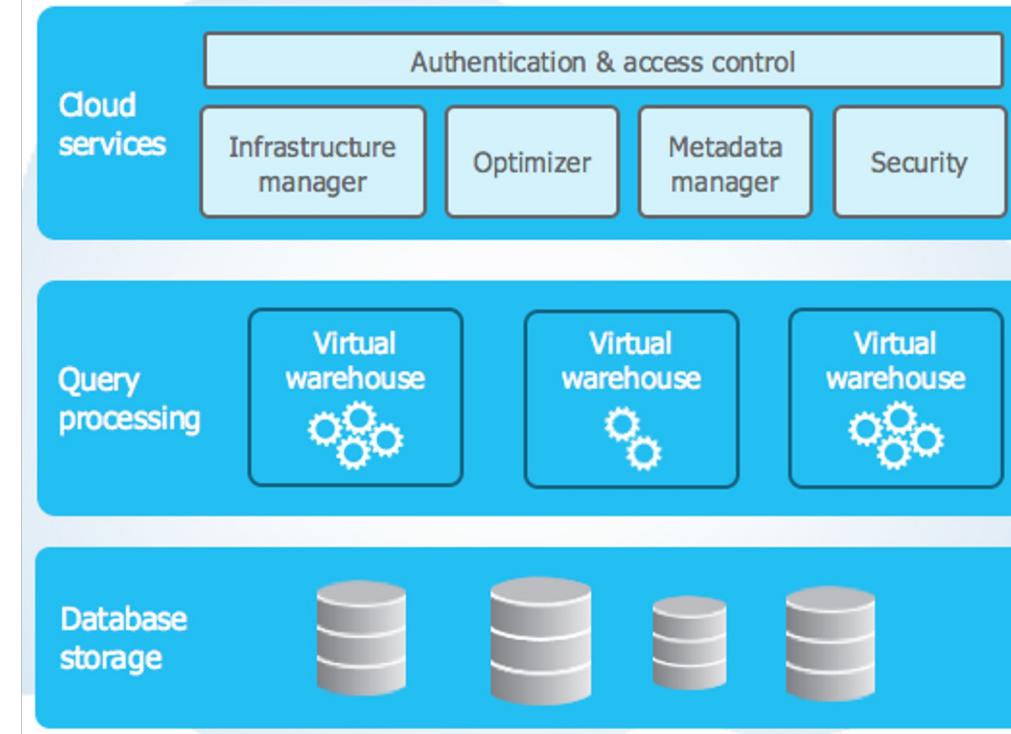
DATA PLATFORM LAYERS

- Storage Layer - Database Storage
- Compute Layer - Query Processing
- Cloud Services Layer



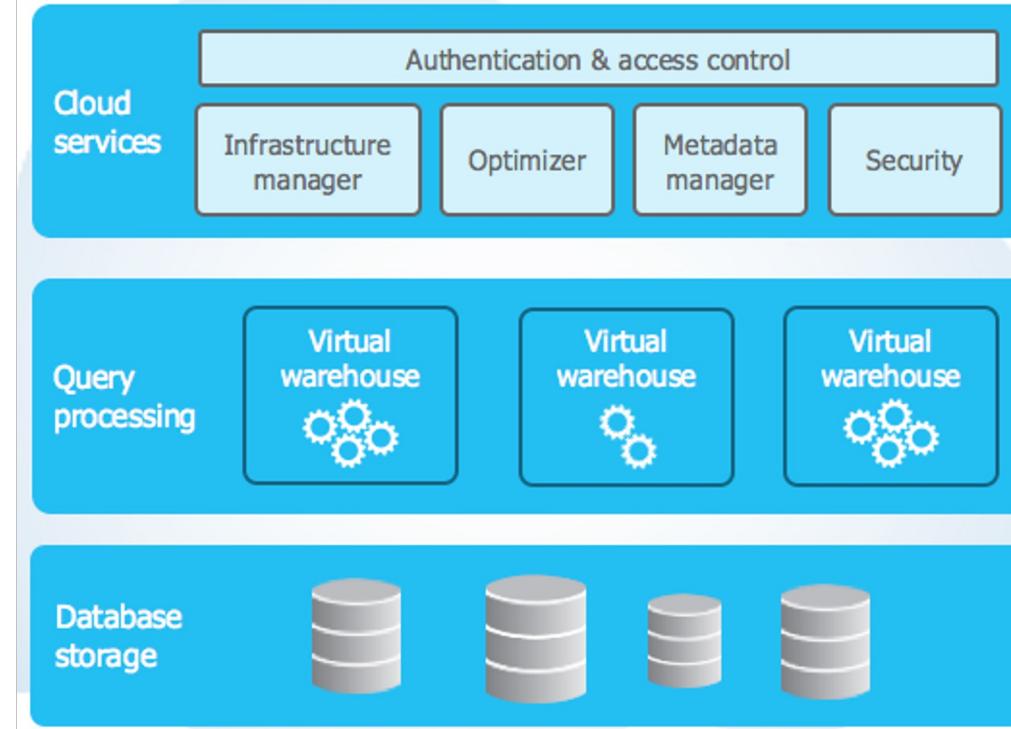
STORAGE LAYER - DATABASE STORAGE

- Storage Layer - Database Storage
- Computer Layer - Query Processing
- Cloud Services Layer



DATA PLATFORM LAYERS

- Storage Layer - Database Storage
- Computer Layer - Query Processing
- Cloud Services Layer



TECHNICAL OVERVIEW

- Understanding Snowflake Table Structures
- Working with Temporary and Transient Tables
- Working with External Tables
- Using the Search Optimization Service
- Overview of Views
- Working with Secure Views
- Working with Materialized Views
- Table Design Considerations
- Cloning Considerations
- Data Storage Considerations

FRAMING YOUR DATA IN SNOWFLAKE

All data in Snowflake is maintained in databases.

Each database consists of one or more schemas.

Schemas are logical groupings of database objects, such as tables and views.

Snowflake has **no hard limits** on the number of **databases**, **schemas** (within a database), or **objects** (within a schema) you can create.

DEFAULT DATABASES IN SNOWFLAKE

The following databases are default, although in the trial accounts the SNOWFLAKE DB is not available

- DEMO_DB
- SNOWFLAKE
- SNOWFLAKE_SAMPLE_DATA
- UTIL_DB

The SNOWFLAKE databases is not available to both SYSADMIN and ACCOUNTADMIN, but rather by default only to the ACCOUNTADMIN user

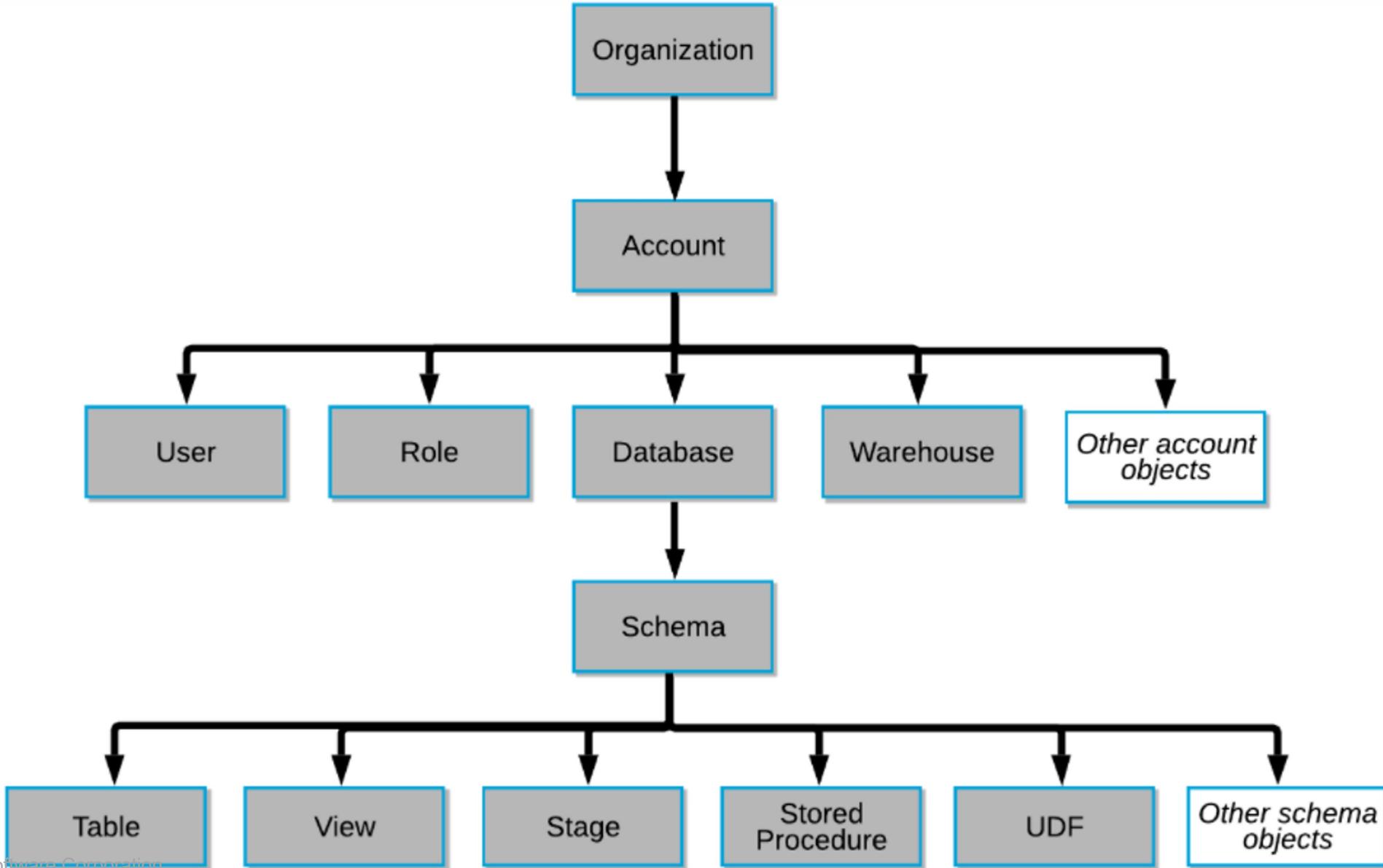
SNOWFLAKE TABLE STRUCTURES

All data in Snowflake is stored in database tables, logically structured as collections of columns and rows.

Snowflake physical table structure features ***micro-partitions*** and ***data clustering***.

These features provide guidance for explicitly defining ***clustering keys*** for very large tables (in the multi-terabyte range) to help optimize table maintenance and query performance.

SNOWFLAKE OBJECTS



MICRO PARTITIONS

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

Physical Structure

type	Micro-partition 1 (rows 1-6)			Micro-partition 2 (rows 7-12)			Micro-partition 3 (rows 13-18)			Micro-partition 4 (rows 19-24)		
name	A	C	C	Z	B	C	X	Z	Y	C	Z	Y
country	UK	SP	DE	DE	FR	SP	FR	NL	SP	FR	NL	SP
date	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/3	11/4	11/5
2	2	4	3	3	2	4	2	4	2	1	4	5
2	2	3	2	5	1	5	1	5	3	5	3	2
3	Z	B	C	X	A	A	B	X	A	C	Z	Y
3	B	A	C	X	A	A	B	X	A	B	X	Z
2	UK	SP	DE	DE	NL	NL	FR	NL	SP	SP	DE	UK
2	DE	FR	SP	FR	FR	FR	SP	DE	UK	SP	DE	UK
3	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/2	11/3	11/4	11/5
3	11/2	11/2	11/2	11/3	11/3	11/3	11/3	11/3	11/4	11/3	11/5	11/5

MICRO PARTITION BENEFITS

- Snowflake micro-partitions are **derived automatically**
- Micro-partitions are **small in size** (50 to 500 MB, before compression).
- Micro-partitions can **overlap** in their range of **values**.
- Snowflake uses ***columnar storage***.
- Columns are also compressed individually within micro-partitions. Snowflake automatically determines the most efficient compression algorithm for the columns in each micro-partition.

MICRO PARTITION METADATA

Snowflake stores metadata about all rows stored in a micro-partition, including:

- The range of values for each of the columns in the micro-partition.
- The number of distinct values.
- Additional properties used for both optimization and efficient query processing.

TEMPORARY AND TRANSIENT TABLES

Temporary Tables

Snowflake supports creating temporary tables for storing non-permanent, transitory data (e.g. ETL data, session-specific data).

Transient Tables

Transient tables are specifically designed for transitory data that needs to be maintained beyond each but does not need the same level of data protection and recovery provided by permanent tables.

TRANSIENT DATABASES AND SCHEMAS

Snowflake also supports creating transient databases and schemas.

All tables created in a transient schema, as well as all schemas created in a transient database, are transient by definition.

EXTERNAL TABLES

External tables reference data files located in a cloud storage (**Amazon S3, Google Cloud Storage, or Microsoft Azure**) data lake.

External tables store file-level metadata about the data files such as the file path, a version identifier, and partitioning information.

This enables querying data stored in files in a data lake as if it were inside a database.

EXTERNAL TABLES USAGES

External tables are **read-only**, therefore no DML operations can be performed on them; however, external tables can be **used for query and join operations**. Views can be created against external tables.

Querying data stored external to the database is likely to be **slower than querying native database tables**

SEARCH OPTIMIZATION SERVICE

The search optimization service aims to significantly improve the performance of selective point lookup queries on tables. A point lookup query returns only one or a small number of distinct rows.

Use case examples include:

- Business users who need fast response times for critical dashboards with highly selective filters.
- Data scientists who are exploring large data volumes and looking for specific subsets of data.

A user can register one or more tables to the search optimization service. Search optimization is a table-level property and applies to all columns with supported data types (see the list of supported data types further below).

HOW DOES SNOWFLAKE SEARCH OPTIMIZER SERVICE WORK

To **improve performance for point lookups**, the search optimization service relies on a persistent data structure that serves as an optimized search access path.

A **maintenance service** that runs in the background is responsible for creating and **maintaining the search access path**:

- When you add search optimization to a table, the maintenance service populates the data needed to perform the lookups.
- The process of populating data is done in the background
- When data in the table changes, the service automatically updates the search access path to reflect the changes to the data.
- Queries might run slower if the search access path hasn't updated but will always return up-to-date results.

QUERY PERFORMANCE OPTIMIZATION

Clustering: Can speed any of the following, as long as they are on the clustering key for **Range** and **Equality** searches. A table can be clustered on only a single key.

Search optimization service: speeds only **Equality** searches. However, this applies to all the columns of supported types in a table that has search optimization enabled.

Materialized view: speeds both **Equality** searches and **Range** searches, as well as some sort operations, but only for the subset of rows and columns included in the materialized view

QUERY PERFORMANCE COST CONSIDERATIONS

	Storage Cost	Compute Cost
Search Optimization Service	✓	✓
Materialized View	✓	✓
Clustering the Table		✓

VIEW EXAMPLE

```
create table hospital_table (patient_id integer,  
                            patient_name varchar,  
                            billing_address varchar,  
                            diagnosis varchar,  
                            treatment varchar,  
                            cost number(10,2));  
  
insert into hospital_table  
    (patient_id, patient_name, billing_address, diagnosis, treatment, cost)  
values  
    (1, 'Crabby Bob', 'Fort Meyers Florida', 'Tasty Crustration',  
     'Texas Roadhouse', 2000.00),  
    (2, Harry Potter', 'Gryffindoor', 'Parsel Tongue', 'Horcrux Removal',  
     100000.00)
```

VIEW EXAMPLE

```
create view doctor_view as  
select patient_id, patient_name, diagnosis, treatment from hospital_table;
```

```
create view accountant_view as  
select patient_id, patient_name, billing_address, cost from hospital_table;
```

SNOWFLAKE VIEWS

A view allows the result of a query to be accessed as if it were a table. The query is specified in the **CREATE VIEW** statement. This is a concept that's replicated from most mature RDBMS products like Oracle and IBM DB2.

Views serve a variety of purposes, including combining, segregating, and protecting data

Snowflake has two types of views:

- Materialized
- Non-materialized

NON-MATERIALIZED VIEWS

The term “view” generically refers to all types of views; however, the term is used here to refer specifically to non-materialized views.

A view is basically a named definition of a query. A non-materialized view’s results are created by executing a query.

Any query expression that returns a valid result can be used to create a non-materialized view, such as:

- Selecting some (or all) columns in a table.
- Selecting a specific range of data in table columns.
- Joining data from two or more tables.

SECURE VIEWS

Snowflake optimizations for views require access to the underlying data in the base tables for the view.

Secure views do not utilize these optimizations, ensuring that users have no access to the underlying data.

For security or privacy reasons, you might not wish to expose the underlying tables or internal structural details for a view.

With secure views, the view definition and details are only visible to authorized users

WHEN TO USE SECURE VIEWS

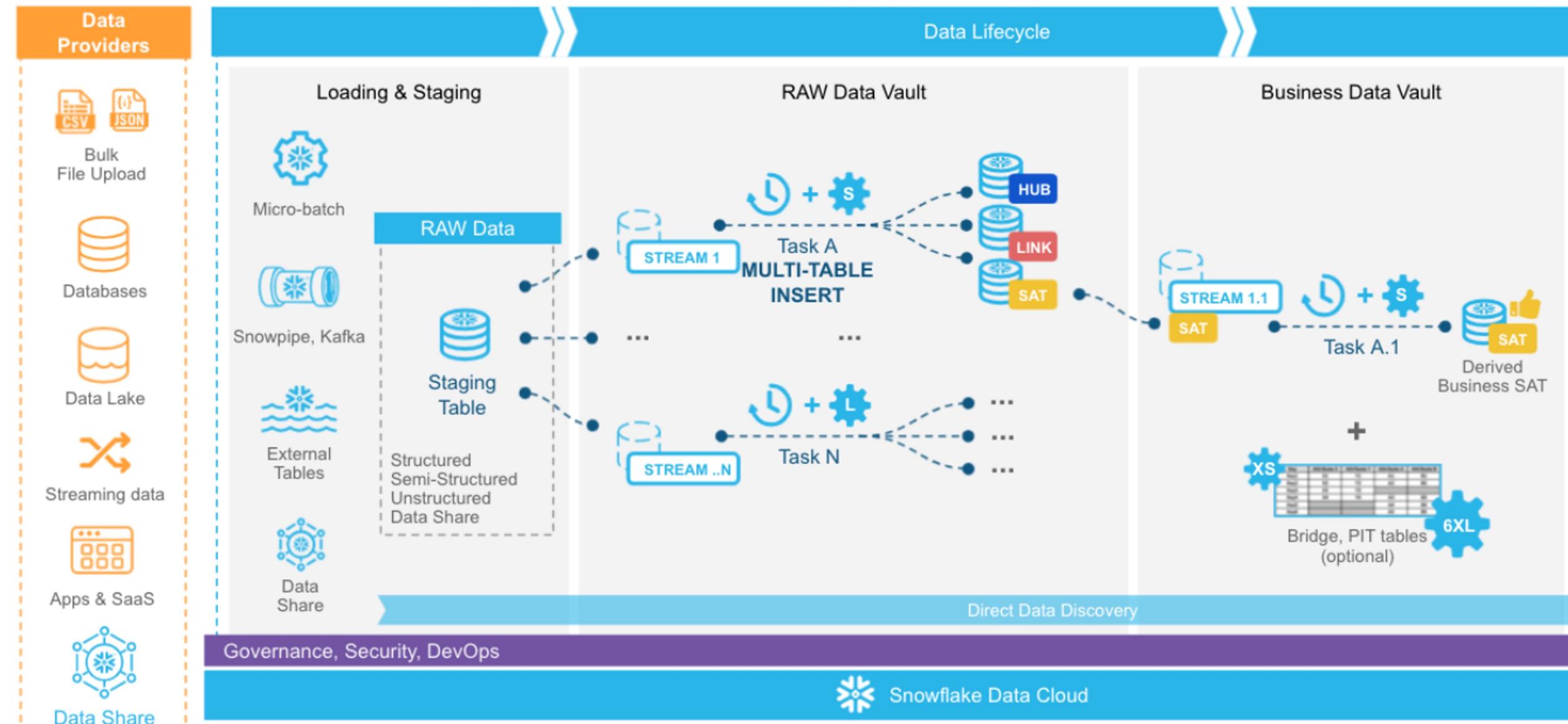
Views should be defined as secure when they are specifically designated for data privacy (i.e. to limit access to sensitive data that should not be exposed to all users of the underlying table(s)).

Secure views should ***not*** be used for views that are defined for query convenience, such as views created for simplifying querying data for which users do not need to understand the underlying data representation. This is because the Snowflake query optimizer, when evaluating secure views, bypasses certain optimizations used for regular views. This might result in some impact on query performance for secure views.

TABLE DESIGN CONSIDERATIONS

- Date/Time Data Types for Columns
- Referential Integrity Constraints
- When to Set a Clustering Key
- When to Specify Column Lengths
- Storing Semi-structured Data in a VARIANT Column vs. Flattening the Nested Structure
- Converting a Permanent Table to a Transient Table or Vice-Versa

DATA ENGINEERING PIPELINES



CLONING CONSIDERATIONS

A cloned object does not retain any granted privileges on the source object itself (i.e. clones do not automatically have the same privileges as their sources).

A system administrator or the owner of the cloned object must explicitly grant any required privileges to the newly-created clone.

However, if the source object is a database or schema, for child objects contained in the source, the clone replicates all granted privileges on the corresponding child objects:

- For databases, contained objects include schemas, tables, views, etc.
- For schemas, contained objects include tables, views, etc.

DATA STORAGE – CONTINUOUS DATA PROTECTION (CDP)

CDP, which includes Time Travel and Fail-safe, is a standard set of features available to all Snowflake accounts at no additional cost.

However, because your account is charged for all data stored in tables, schemas, and databases created in the account, CDP does have an impact on storage costs, based on the total amount of data stored and the length of time the data is stored.

Storage is calculated and charged for data regardless of whether it is in the Active, Time Travel, or Fail-safe state. Because these life-cycle states are sequential, updated/deleted data protected by CDP will continue to incur storage costs until the data leaves the Fail-safe state.

MANAGING COSTS – SHORT LIVED DATA

CDP is designed to provide long-term protection for your data. This data is typically stored in permanent tables. Unless otherwise specified at the time of their creation, tables in Snowflake are created as permanent.

During an ETL or data modeling process, tables may be created that are short-lived. For these tables, it does not make sense to incur the storage costs of CDP. **Snowflake provides two separate mechanisms to support short-lived tables:**

- Temporary tables
- Transient tables

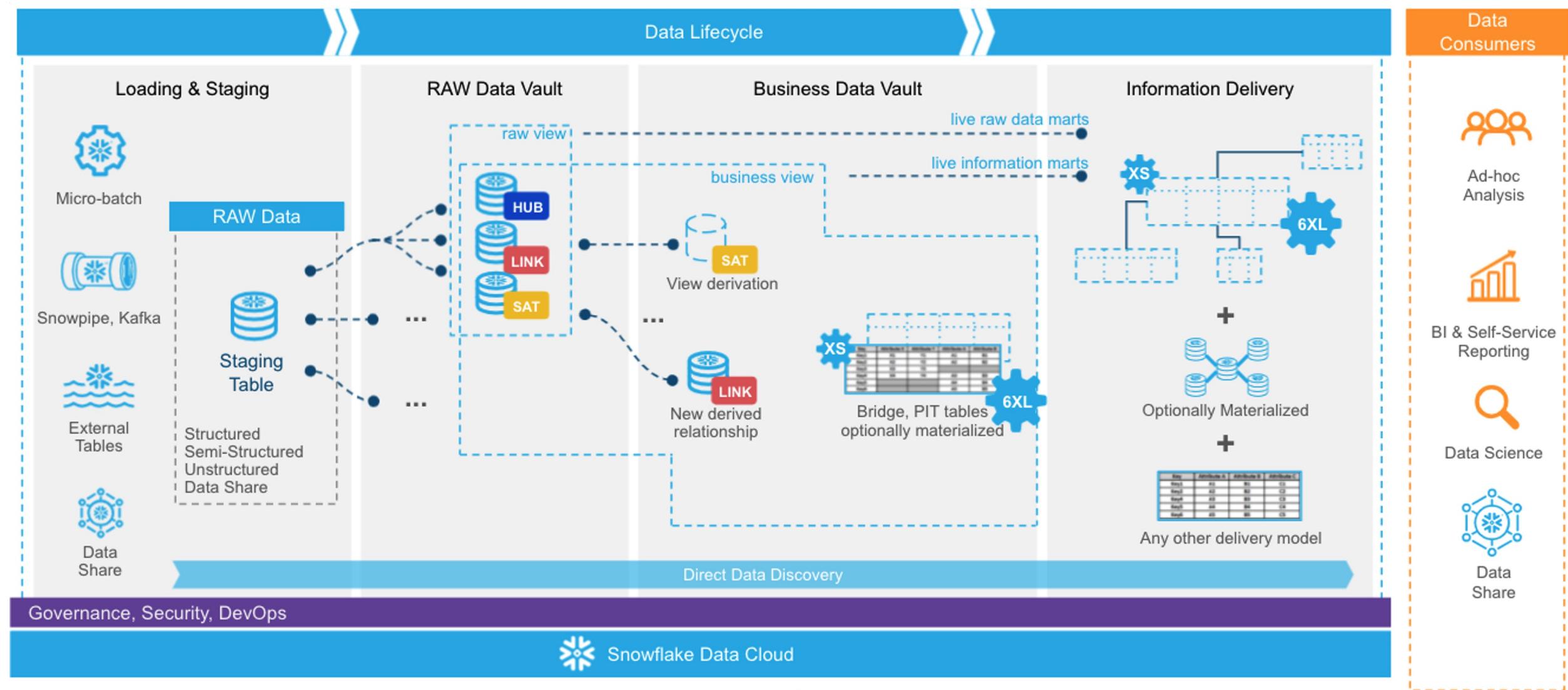
MANAGING COSTS – RESILIENT USEFUL DATA

Fact or Dimension tables, which have **different usage patterns** and, therefore, different storage considerations:

Fact tables are typically very large in size and experience a low degree of churn (row updates or deletes). Most changes to fact tables are inserts of new data or, in some cases, deletions of older data. CDP is ideal for fact tables as it provides full data protection at a very low storage cost.

Dimension tables have a different update pattern. Row updates and deletions are much more common in dimension tables.

COMPLEX DATA ARCHITECTURE - VAULT



POP QUIZ:

Snowflake Architecture



Follow the link for Snowflake Architecture Pop Quiz:

[https://forms.gle/BWNrrqRpch8
WiVRR8](https://forms.gle/BWNrrqRpch8WiVRR8)



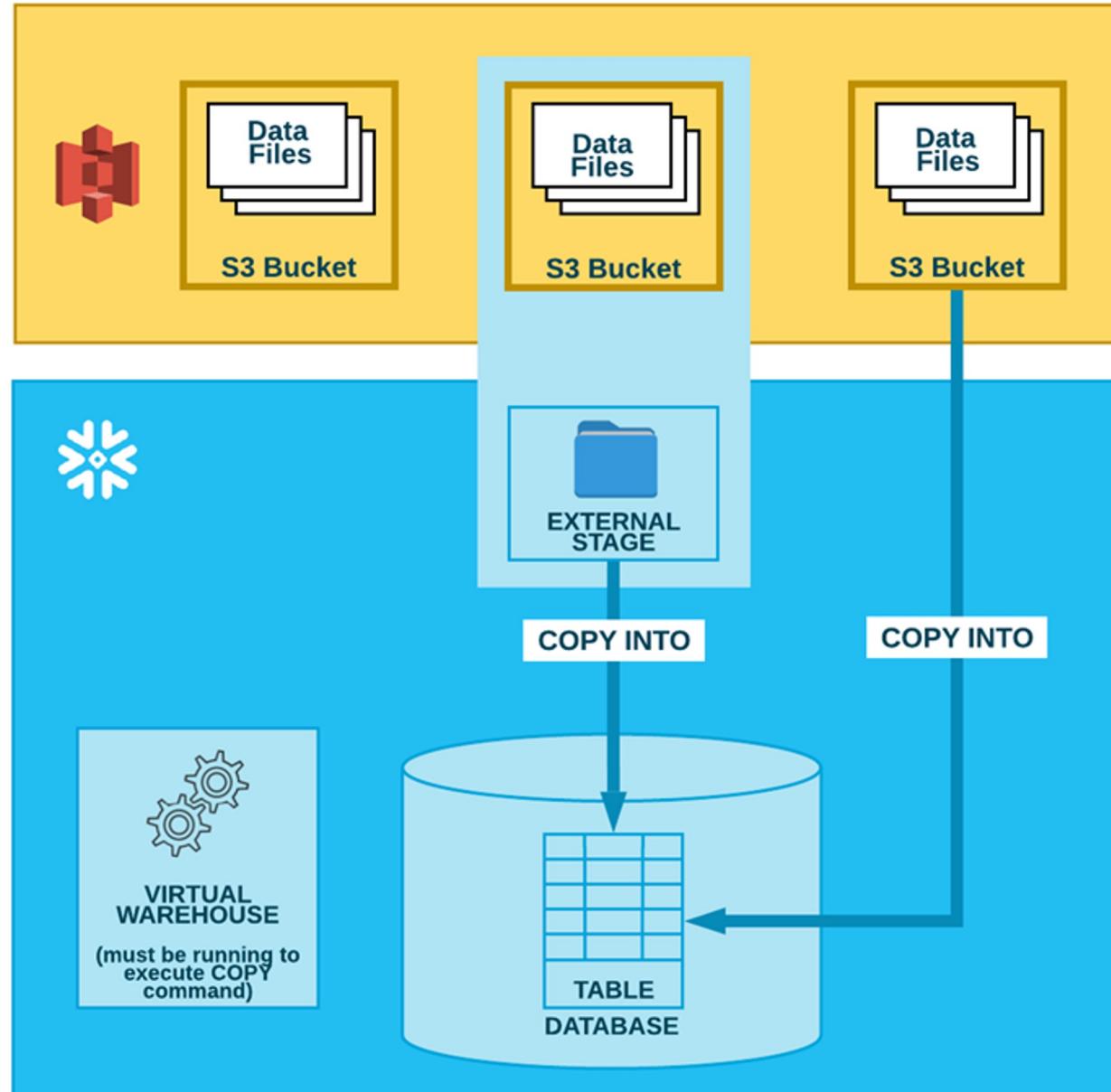
Data Movement



What We have Covered so far

- Domain:
 - Virtual Warehouses
 - Data Movement
 - Performance Management
 - Architecture
 - Storage and Protection
- Default Databases in Snowflake
- Micro Partition
- Snowflake Search Optimization
- Views
- Continuous Data Protection (CDP)

DATA LOADING



DATA LOADING TOPICS

- Supported File Locations
 - External Stages
 - Internal Stages
- Bulk vs Continuous Loading
 - Bulk Loading Using the COPY Command
 - Continuous Loading Using Snowpipe
- Loading Data from Apache Kafka Topics
- Alternatives to Loading Data
 - External Tables (Data Lake)

DATA LOADING STAGES

Snowflake refers to the location of data files in cloud storage as a **stage**.

The **COPY INTO** commands are used for both bulk and continuous data loads (i.e. Snowpipe) into stages

- **External Stage** - cloud storage accounts managed by your business entity – Azure/GCP/AWS
- **Internal Stage** - cloud storage contained in your Snowflake account

DATA LOADING: COPY

The **COPY INTO <table>** loads data from staged files to an existing table. The files must already be staged in one of the following locations:

- Named internal stage (or table/user stage). Files can be staged using the PUT command.
- Named external stage that references an external location (Amazon S3, Google Cloud Storage, or Microsoft Azure).
- External location (Amazon S3, Google Cloud Storage, or Microsoft Azure).

DATA UNLOADING: COPY

The **COPY INTO <location>** unloads data from a table (or query) into one or more files in one of the following locations:

- Named internal stage (or table/user stage). The files can then be downloaded from the stage/location using the GET command.
- Named external stage that references an external location (Amazon S3, Google Cloud Storage, or Microsoft Azure).
- External location (Amazon S3, Google Cloud Storage, or Microsoft Azure).

DATA LOADING: INSERT

Using a single `INSERT` command, you can insert multiple rows into a table by specifying additional sets of values separated by commas in the `VALUES` clause.

For example, the following clause would insert 3 rows in a 3-column table, with values 1, 2, and 3 in the first two rows and values 2, 3, and 4 in the third row:

```
values ( 1, 2, 3 ) , ( 1, 2, 3 ) , ( 2, 3, 4 )
```

To use the `OVERWRITE` option on `INSERT`, you must use a role that has `DELETE` privilege on the table because `OVERWRITE` will delete the existing records in the table.

Some expressions cannot be specified in the `VALUES` clause.

DATA LOADING: GET

Downloads data files from one of the following Snowflake stages to a local directory/folder on a client machine:

- Named internal stage.
- Internal stage for a specified table.
- Internal stage for the current user.

Typically, this command is executed after using the COPY INTO <location> command to unload data from a table into a Snowflake stage.

DATA LOADING: PUT

Uploads (i.e. stages) data files from a local directory/folder on a client machine to one of the following Snowflake stages:

- Named internal stage
 - Internal stage for a specified table
 - Internal stage for the current user
-
- Once files are staged, the data in the files can be loaded into a table using the COPY INTO <table> command.

DATA LOADING: VALIDATE

Validates the files loaded in a past execution of the [COPY INTO <table>](#) command and returns all the errors encountered during the load, rather than just the first error.

The validation returns no results for COPY statements that specify ON_ERROR = ABORT_STATEMENT (default value).

Validation fails if:

- The current user does not have access to table_name.
- The current user is not the user who executed query_id and does not have access control privileges on this user.
- If new files have been added to the stage used by query_id since the load was executed, the new files added are ignored during the validation.

FILE FORMATS

A named file format object provides a convenient means to store all of the format information required for loading data from files into tables.

Execute CREATE FILE FORMAT to create a file format that you can reference throughout your project or data warehouse usage.

This step is optional, but is recommended when you plan to load large numbers of files of a specific format.

CSV FILE FORMATS

To create a CSV file format:

TYPE - defaults to CSV

FIELD_DELIMITER – defaults to a comma

SKIP_HEADER an integer value defining the number of lines in the header. The COPY command skips these lines when loading data. The default value is 0.

```
create or replace file format mycsvformat
  type = 'CSV'
  field_delimiter = '|'
  skip_header = 1;
```

JSON FILE FORMATS

To create a JSON file format:

TYPE - defaults to JSON

STRIP_OUTER_ARRAY = TRUE. Instructs the JSON parser
to remove the root brackets [].

```
create or replace file format myjsonformat
  type = 'JSON'
  strip_outer_array = true;
```

CREATE STAGES

Create stages using the **CREATE STAGE** command.

A named external stage is a database object created in a schema.

This object stores the **URL** to files in cloud storage, the **settings** used to access the cloud storage account, and convenience settings such as the **options** that describe the format of staged files.

CREATE STAGES

Execute CREATE STAGE to create a named internal stage. This step is recommended when you plan to load data files regularly from the same source.

Our example CREATE STAGE commands create internal stages that specify the file formats that we showed for CSV and JSON formats.

COPY commands default to the file formats that are specified when we use CREATE STAGE and do not need to be specified on the COPY command.

CREATE CSV-FORMATTED STAGE

The following example creates an internal stage named my_csv_stage.

Parameter values that aren't specified use the default values (DATE_FORMAT = AUTO, COMPRESSION = AUTO, etc.).

```
create or replace stage my_csv_stage  
    file_format = mycsvformat;
```

CREATE JSON-FORMATTED STAGE

The following example creates an internal stage named my_json_stage.

Parameter values that aren't specified use the default values (DATE_FORMAT = AUTO, COMPRESSION = AUTO, etc.).

```
create or replace stage my_json_stage  
    file_format = myjsonformat;
```

STAGE FOLDER STANDARDIZATION

Normally a load will include a set of data files. Those files may be packaged in zip or gzip archives. Normally we'd unpack archives to our load file system, optimally setting a standard. That type of folder standardization allows teams to understand data project handling by convention.

We could unpack data file archives to any location. To follow our recommended best practice, we'll set standard directories to be referenced. This requires specifics based on platform, for the most commonly used client/server platforms that would equate to:

Linux or macOS: /tmp/load.

Windows: C:\temp\load.

EXTERNAL STAGES

Loading data from any of the following cloud storage services is supported regardless of the cloud platform that hosts your Snowflake account:

- Amazon S3
- Google Cloud Storage
- Microsoft Azure

Upload (i.e. stage) files to your cloud storage account using the tools provided by the cloud storage service.

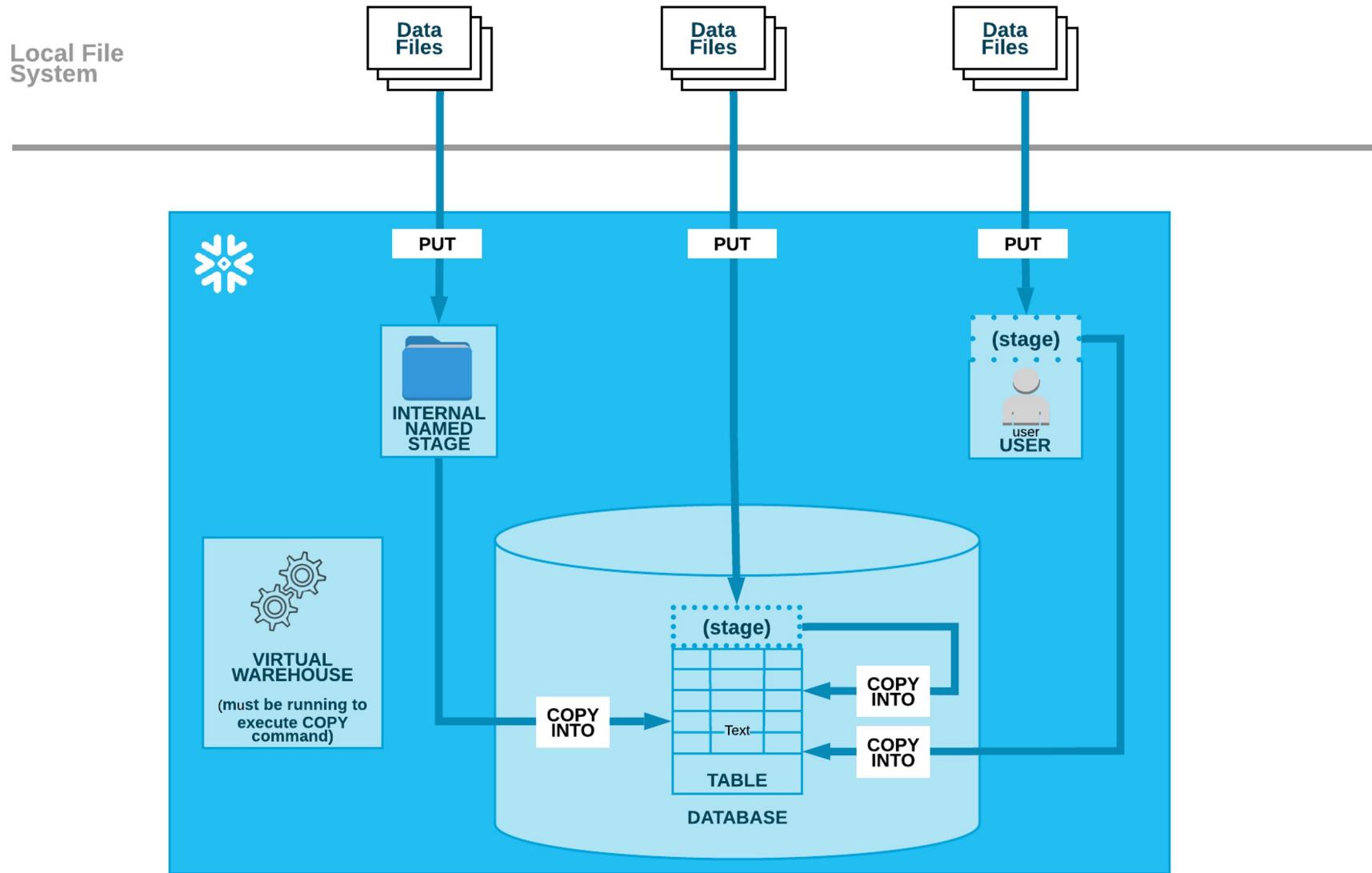
BULK LOADING

Bulk Loading enables loading batches of data from files already available in cloud storage.

Alternatively we can copy (i.e. staging) data files from a local machine to an internal (i.e. Snowflake) cloud storage location before loading the data into tables using the COPY command.

Bulk loading relies on user-provided virtual warehouses, which are specified in the COPY statement. Users are required to size the warehouse appropriately to accommodate expected loads.

BULK LOADING FROM LOCAL FILE



LOAD TRANSFORMATIONS

Snowflake supports transforming data while loading it into a table using the COPY command.

Options include:

- Column reordering
- Column omission
- Casts
- Truncating text strings that exceed the target column length

There is no requirement for your data files to have the same number and ordering of columns as your target table.

DATA WRANGLING

Data **wrangling** —also called data **cleansing**, data **transformation**, data **remediation**, or data **munging** —refers to a variety of processes designed to transform raw data into more readily used formats.

The exact methods differ from project to project depending on the data you're leveraging and the goal you're trying to achieve.

There are many tools that assist in the wrangling journey including ad-hoc query tools like Facebook Presto and AWS Athena.

DATA LOAD PROCESS

Step 1. Create File Format Objects

Step 2. Create Stage Objects

Step 3. Stage the Data Files

Step 4. Verify the Staged Files

Step 5. Copy Data into the Target Tables

Step 6. Resolve Data Load Errors Related to Data Issues

Step 7. Validate the Loaded Data

Step 8. Remove the Successfully Loaded Data Files

Experiment



Content
#03-Snowflake-
Data-Prep.pdf

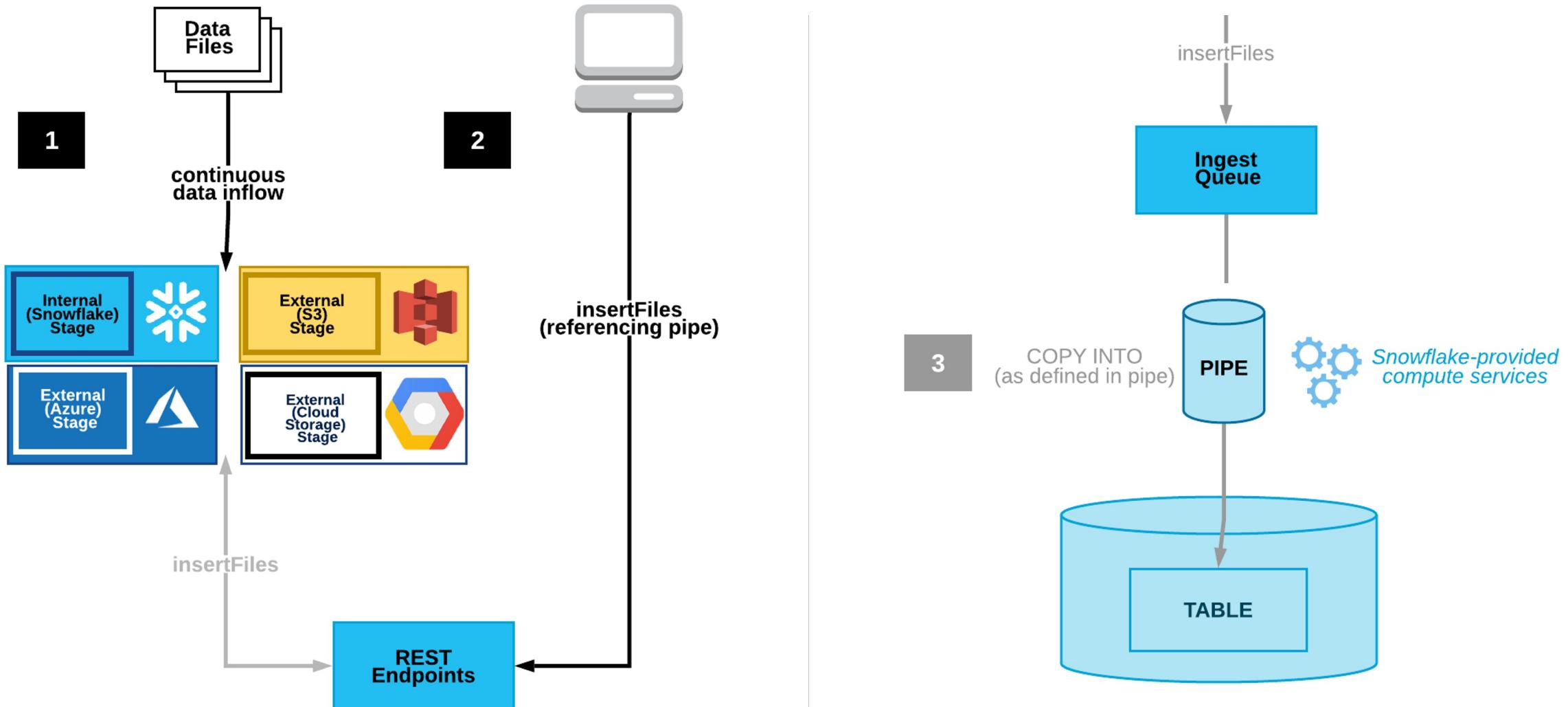
Scripts
foundation_experi
ment_scripts.sql

CONTINUOUS LOADING

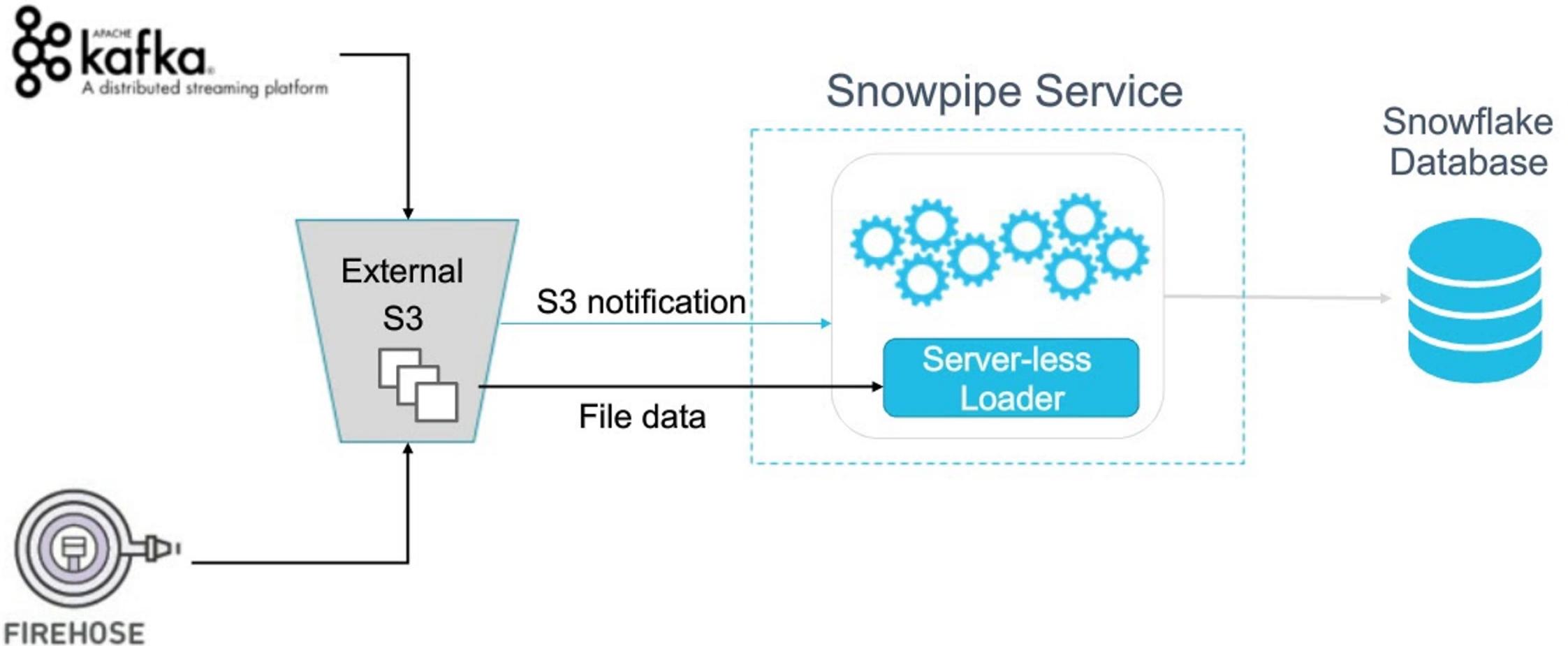
Snowpipe is Snowflake's **continuous data ingestion** service. Snowpipe loads data within minutes after files are added to a stage and submitted for ingestion.

With Snowpipe's **serverless** compute model, Snowflake manages load capacity, ensuring optimal compute resources to meet demand. In short, Snowpipe provides a "pipeline" for **loading data in micro-batches** as soon as it's available.

CONTINUOUS LOADING REST



CONTINUOUS LOADING S3



CONFIGURING SNOWPIPE 4 REST

- Create a named stage object where your data files will be staged. Snowpipe supports both internal (Snowflake) stages and external stages, i.e. S3 buckets.
- Create a pipe object using **CREATE PIPE**.
- Configure security for the user who will execute the continuous data load. If you plan to restrict Snowpipe data loads to a single user, you only need to configure key pair authentication for the user once. After that, you only need to grant access control privileges on the database objects used for each data load.
- Install a client SDK (Java or Python) for calling the Snowpipe public REST endpoints.

DATA UNLOADING

Similar to data loading, Snowflake supports bulk export (i.e. unload) of data from a database table into flat, delimited text files. The unloading process involves the following topics:

Bulk Unloading Process

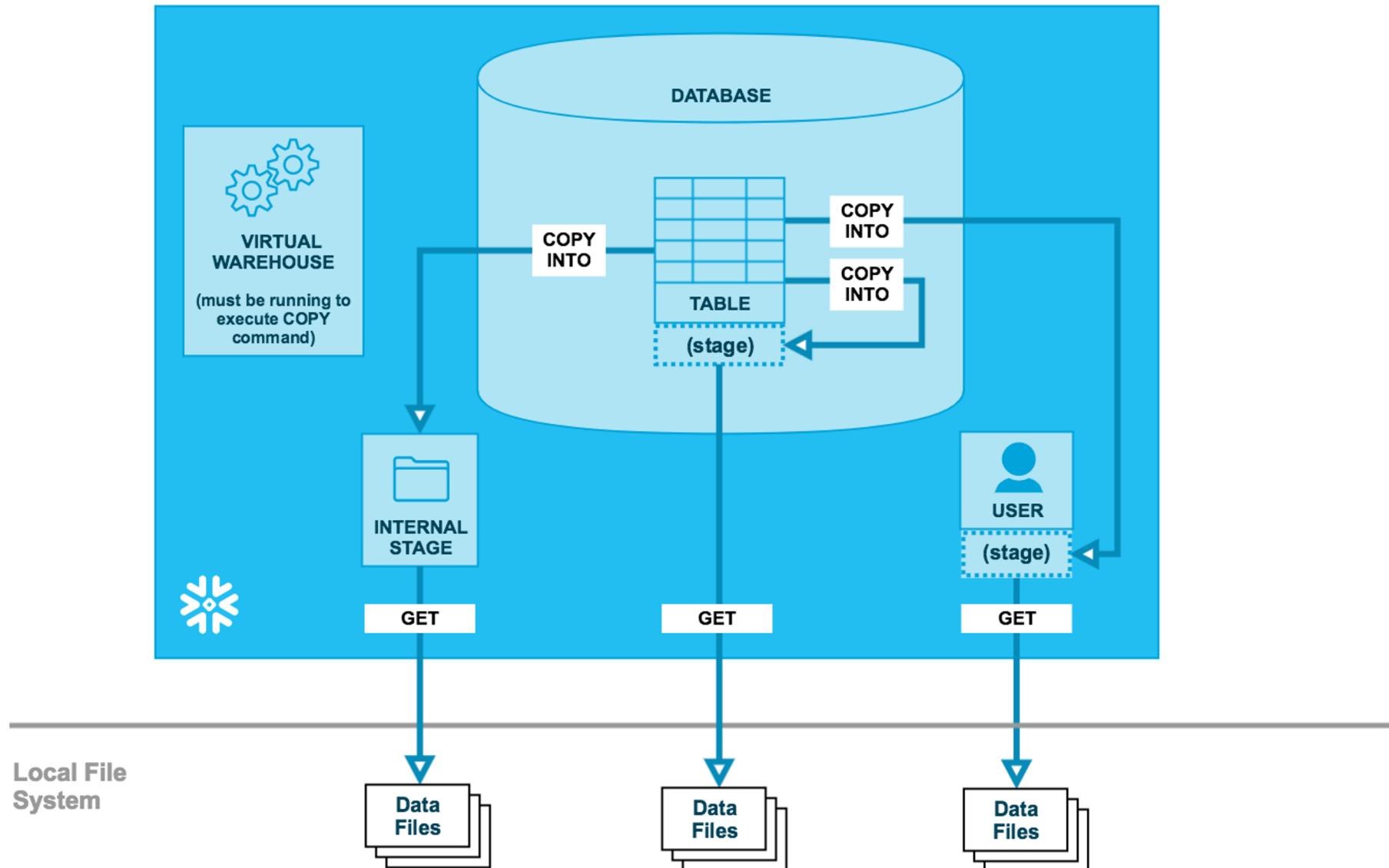
Bulk Unloading Using Queries

Bulk Unloading into Single or Multiple Files

Partitioned Data Unloading

Tasks for Unloading Data Using the COPY Command

DATA UNLOADING INTO STAGE



DATA UNLOADING PROCESS

The process for unloading data into files is the same as the loading process, except in reverse:

1. Use the COPY INTO <location> command to copy the data from the Snowflake database table into one or more files in a Snowflake or external stage.
2. Download the file from the stage:
 - Snowflake stage: use GET to download the data file(s).
 - S3: use the interfaces/tools provided by Amazon S3 to get the data file(s).
 - Azure: use the interfaces/tools provided by Azure to get the data file(s).

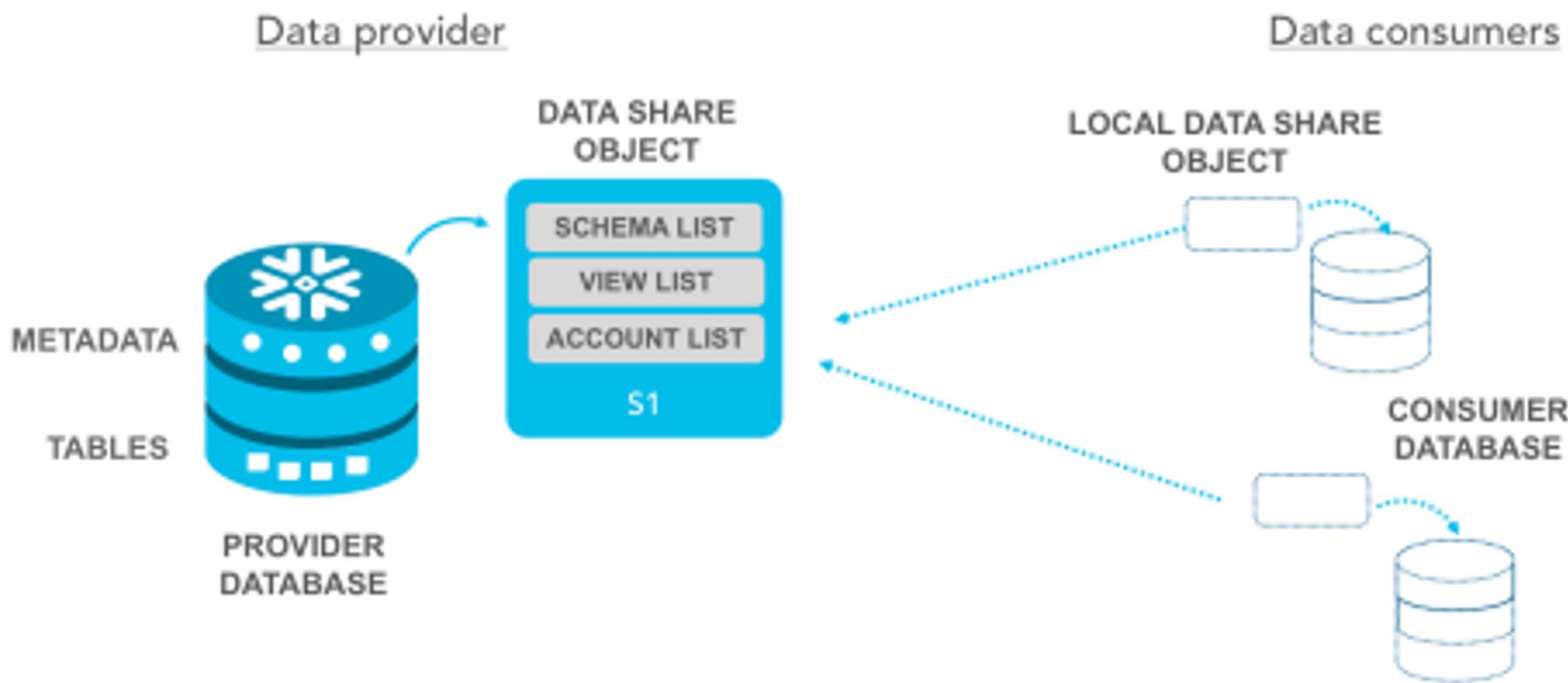
Experiment



Content
#04-Snowflake-
Data-Loading.pdf

Scripts
foundation_experiment_scripts.sql

DATA SHARING



DATA SHARING

Data sharing is the act of providing access to data — both within an enterprise and between enterprises that have determined they have valuable assets to share.

The organization that makes its data available, or shares its data, is a ***data provider***. The organization that wants to use the shared data is a ***data consumer***. Any organization can be a data provider, a data consumer, or both.

DATA SHARING IN SNOWFLAKE

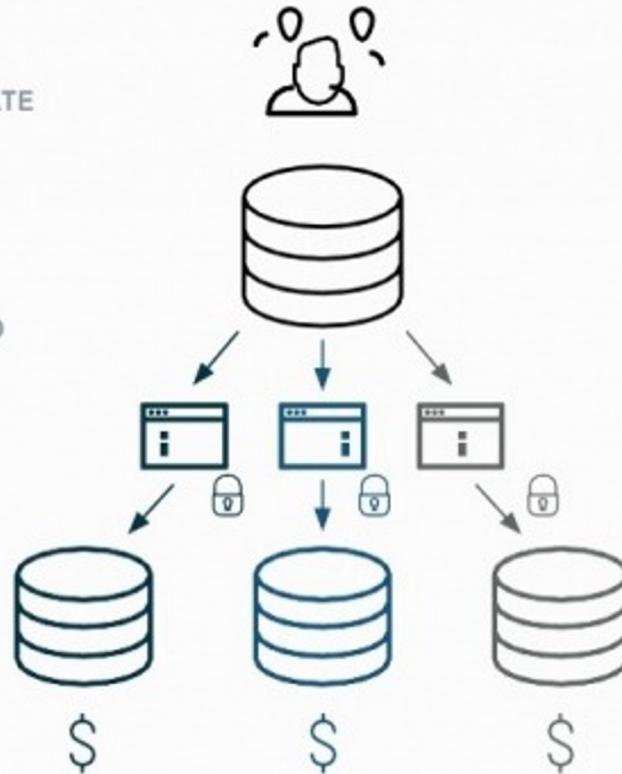
Distributing Data with Traditional

SIGNIFICANT CORPORATE

MANUAL UPDATES AND

REPETITIVE
OVERHEAD FOR

DUPLICATIVE

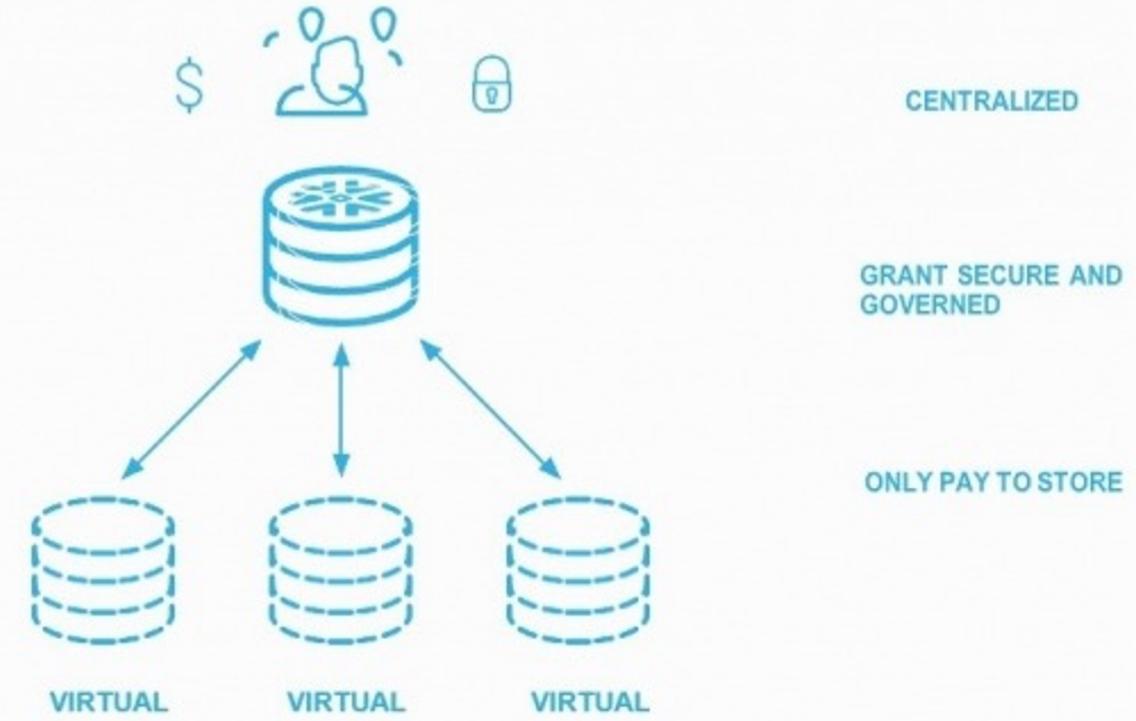


Sharing Data Across Business Units with

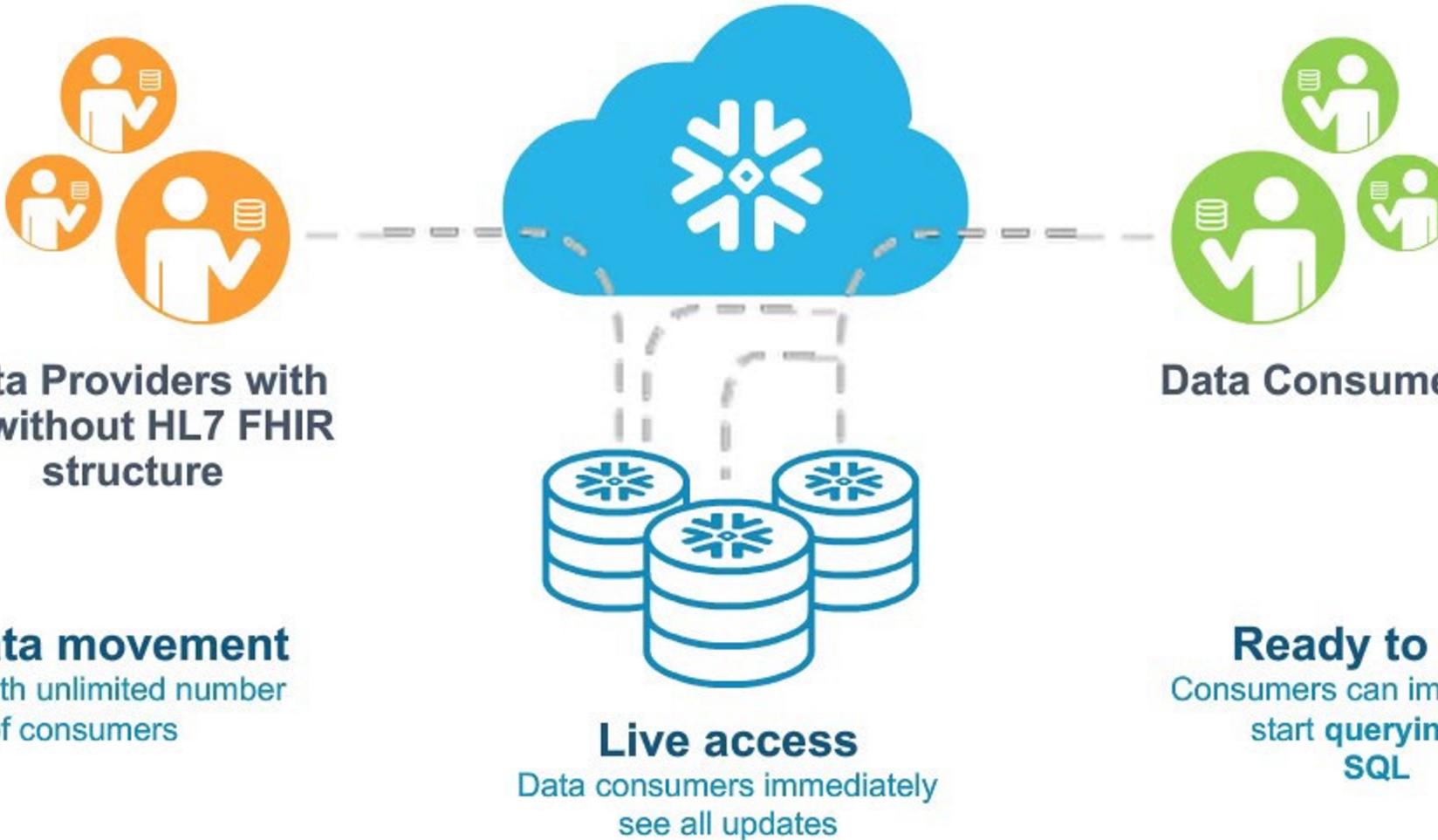
CENTRALIZED

GRANT SECURE AND
GOVERNED

ONLY PAY TO STORE



DATA SHARING VS MOVING

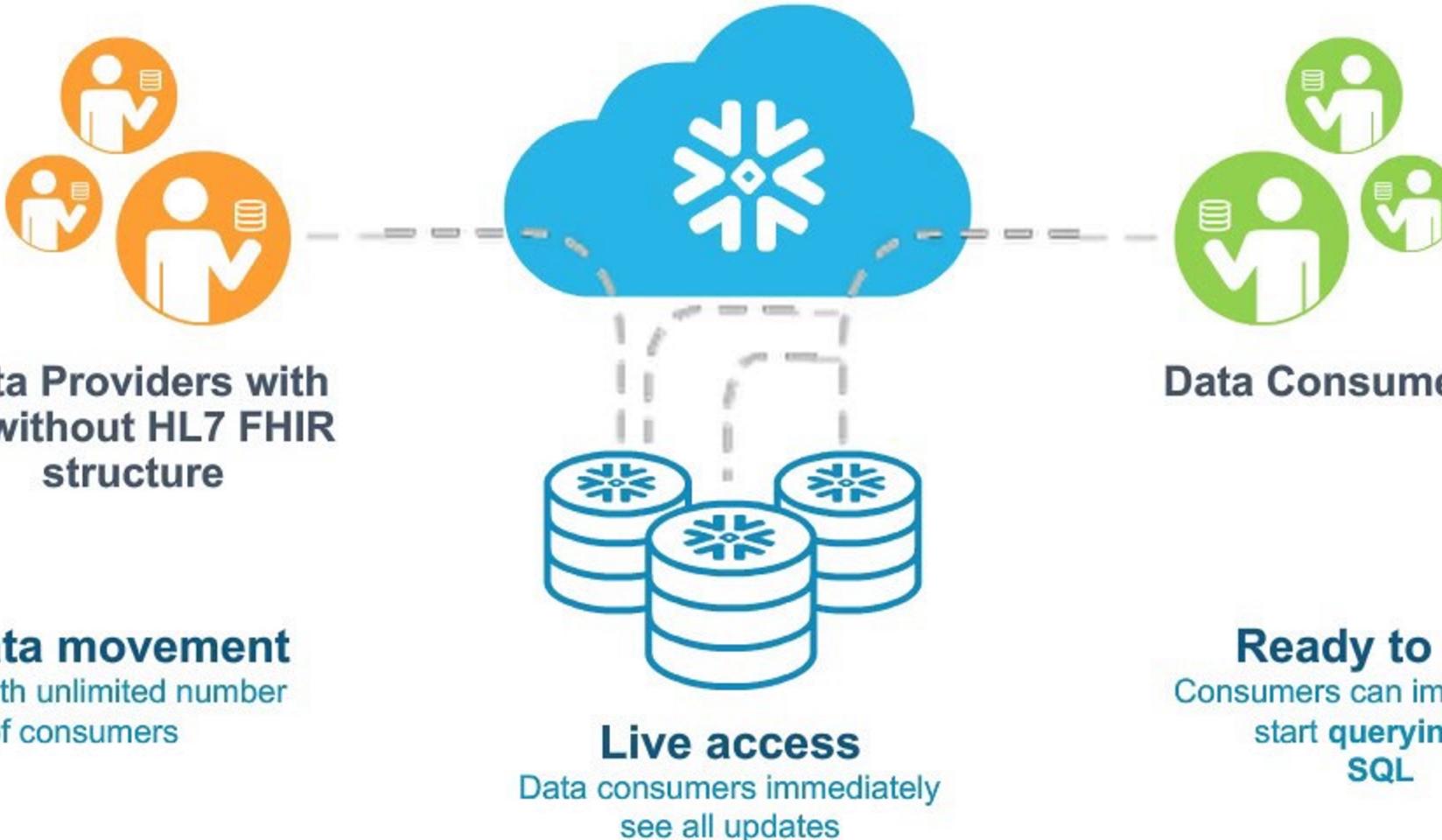


DATA SHARING EVOLUTION

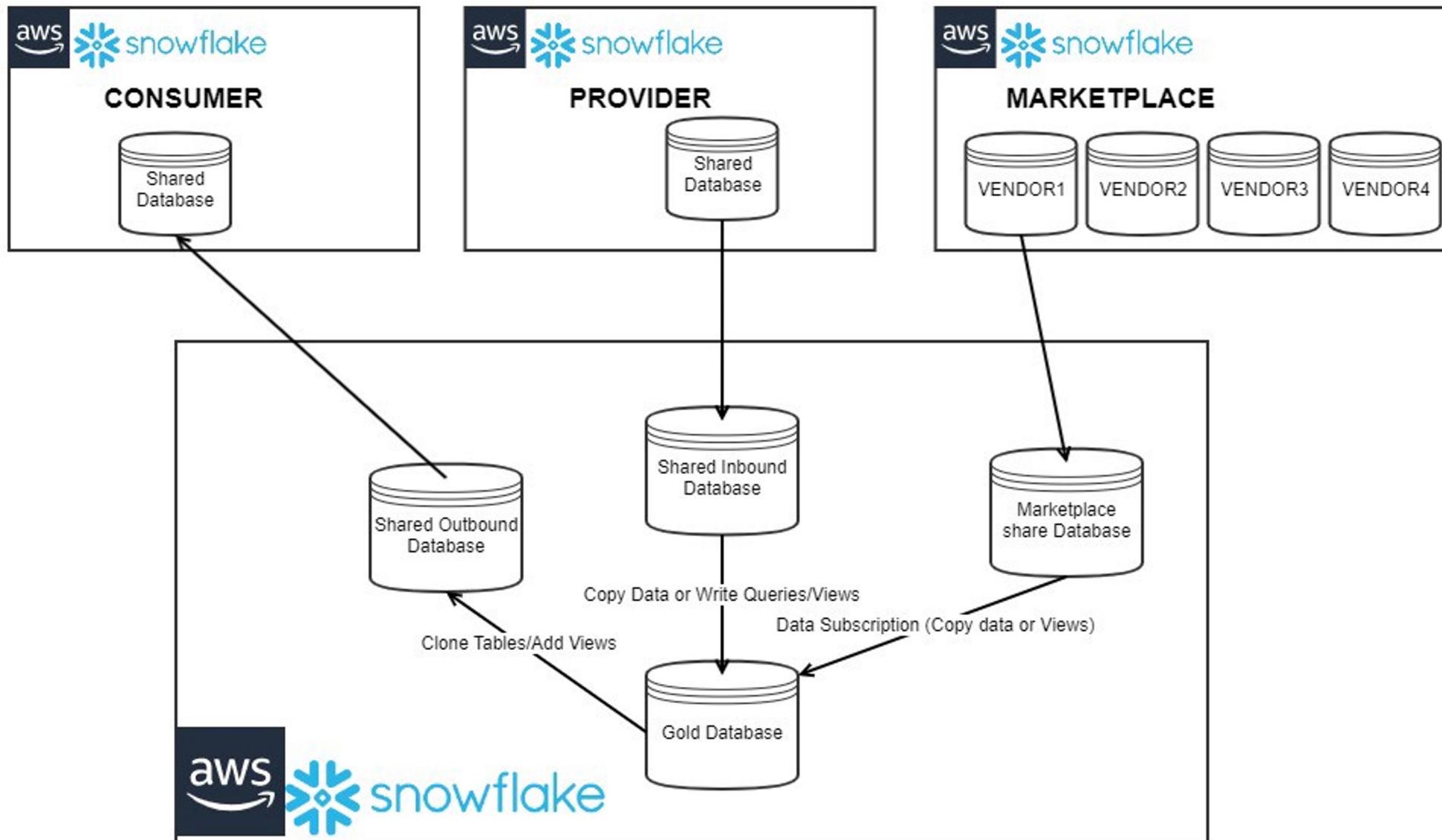
Most organizations that embark on a data sharing journey follow a familiar progression:

- 1. Internal collaboration**
- 2. Business insights**
- 3. Customer analytics**
- 4. Advanced analytics**
- 5. Data services**
- 6. Data exchange**

DATA SHARING VS MOVING



BEST PRACTICES



POP QUIZ:

Data Movement



Follow the link for Data Movement Quiz:

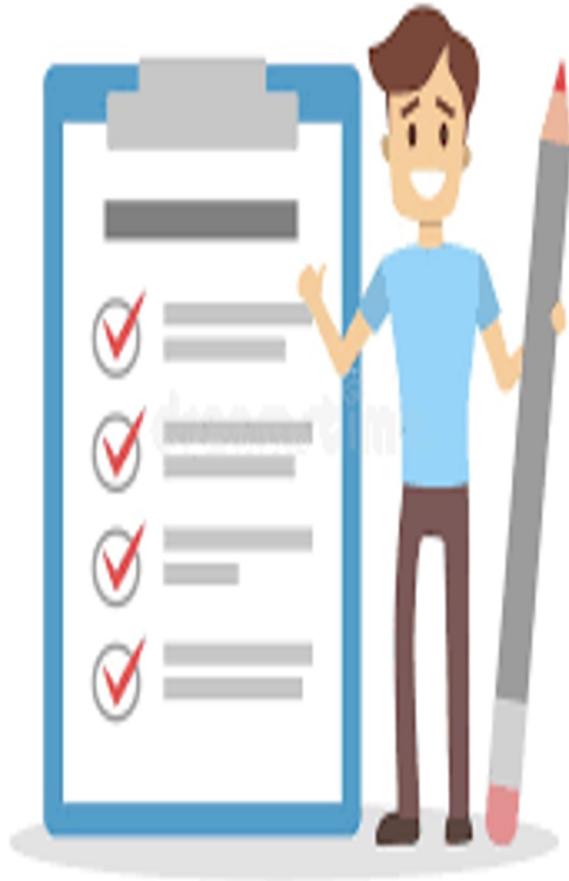
<https://forms.gle/PmbiySpQzJ8fGbwN7>

Objects and Commands



What We have Covered so far

- Data Loading
- File Formats
- Create stages
- Bulk Loading
- Continuous Loading
- Data Unloading
- Data Sharing



QUERY CONSTRUCTS

Snowflake supports standard SQL, including a subset of ANSI SQL:1999 and the SQL:2003 analytic extensions.

Snowflake also supports common variations for a number of commands where those variations do not conflict with each other.

QUERY EDITORS

Following are the list of commonly used Snowflake SQL editor tools.

- Snowflake SQL (SnowSQL)
- Snowflake Web UI
- SQuirrel SQL Client
- SQL Workbench
- DBeaver

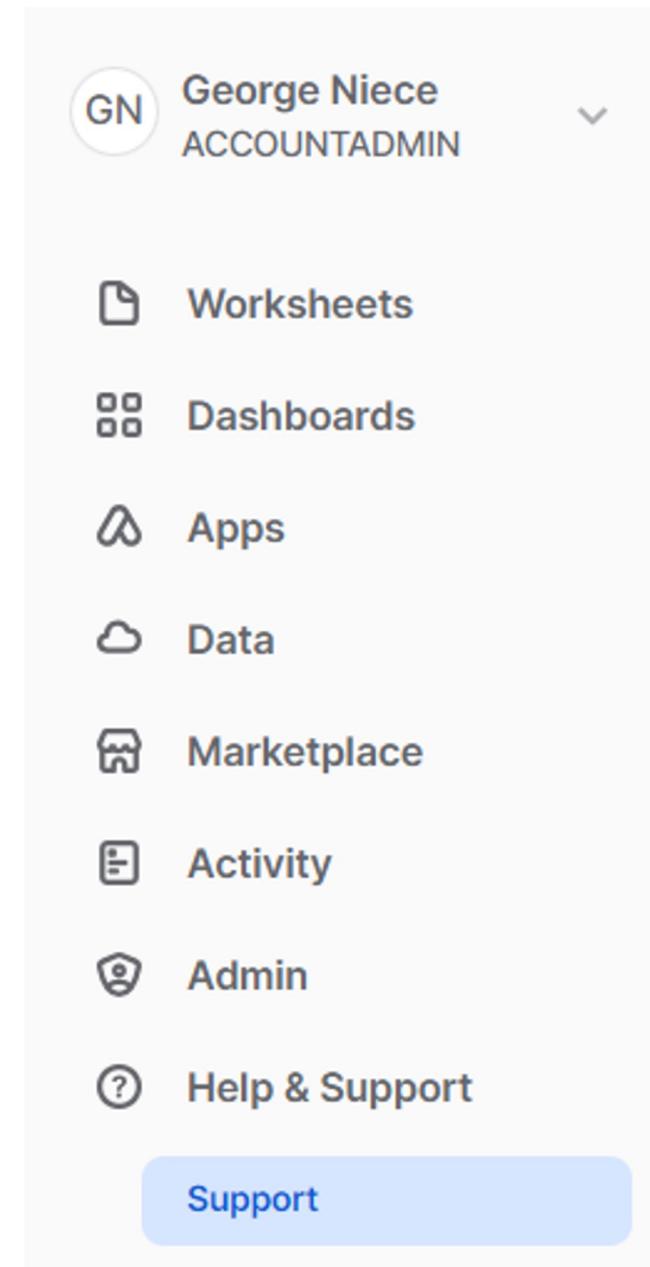
SNOWFLAKE QUERY EXECUTION

There are two interactive Snowflake tools that can be used to connect to a Snowflake instance:

- SnowSQL command-line utility (CLI)
- Snowflake WebUI.

The Snowflake WebUI is the most common way to connect to a Snowflake instance, execute queries, and perform administrative functions such as creating new database objects, managing virtual warehouses, managing security, and reviewing the costs associated with your instance.

SNOWFLAKE WEB UI



The image shows a screenshot of the Snowflake Web UI navigation menu. At the top left is a circular profile icon with the letters "GN". To its right is the name "George Niece" and the title "ACCOUNTADMIN". A downward arrow is located at the top right of the menu. Below the header, there are eight menu items, each with an icon and text: "Worksheets" (document icon), "Dashboards" (grid icon), "Apps" (cloud icon), "Data" (cloud icon), "Marketplace" (gift icon), "Activity" (list icon), "Admin" (key icon), and "Help & Support" (question mark icon). At the bottom of the menu is a blue button labeled "Support".

- Worksheets
- Dashboards
- Apps
- Data
- Marketplace
- Activity
- Admin
- Help & Support

Support

SNOWFLAKE WEB UI - WORKSHEETS

Worksheets

Recent Shared with me My Worksheets Folders

Search ... +

SQL Worksheet
Python Worksheet
Folder

TITLE	TYPE	VIEWED ↓	UPDATED	ROLE	
2023-07-17 10:07am Benchmarking Tutorials	SQL	8 hours ago	8 hours ago	ACCOUNTADMIN	
Tutorial 1: Sample queries on TPC-...	Benchmarking Tuto...	SQL	9 hours ago	9 hours ago	ACCOUNTADMIN
Tutorial 2: Sample queries on TPC-D...	Benchmarking Tut...	SQL		1 day ago	ACCOUNTADMIN
Tutorial 3: TPC-DS 10TB Complete Q...	Benchmarking Tu...	SQL		1 day ago	ACCOUNTADMIN
Tutorial 4: TPC-DS 100TB Complete ...	Benchmarking Tu...	SQL		1 day ago	ACCOUNTADMIN
Benchmarking Tutorials	Folder			1 day ago	

SNOWSQL DOWNLOAD



SNOWSQL FOR LINUX



SNOWSQL FOR MACOS



SNOWSQL FOR WINDOWS

<https://developers.snowflake.com/snowsql/>

SNOWSQL CLIENT

Downloads

CLI Client (snowsql)

JDBC Driver

ODBC Driver

Python Components

Node.js Driver

Spark Connector

Go Snowflake Driver

SnowCD

CLI Client (snowsql)

Download the latest version of SnowSQL for your platform from the [Snowflake Repository](#).

For installation and configuration details, see the [Snowflake Documentation](#)



Snowflake Repository
Linux, Windows, macOS



Snowflake GPG Public Key
For Linux

SNOWSQL CLIENT

Operating System	Supported Versions
Linux	CentOS 7, 8
	Red Hat Enterprise Linux (RHEL) 7, 8
	Ubuntu 16.04, 18.04
macOS	10.14, 10.15 (Support for 10.13 was deprecated in May, 2021.)
Microsoft Windows	Microsoft Windows 8, 8.1, 10
	Microsoft Windows Server 2012, 2016, 2019

SNOWSQL CLIENT

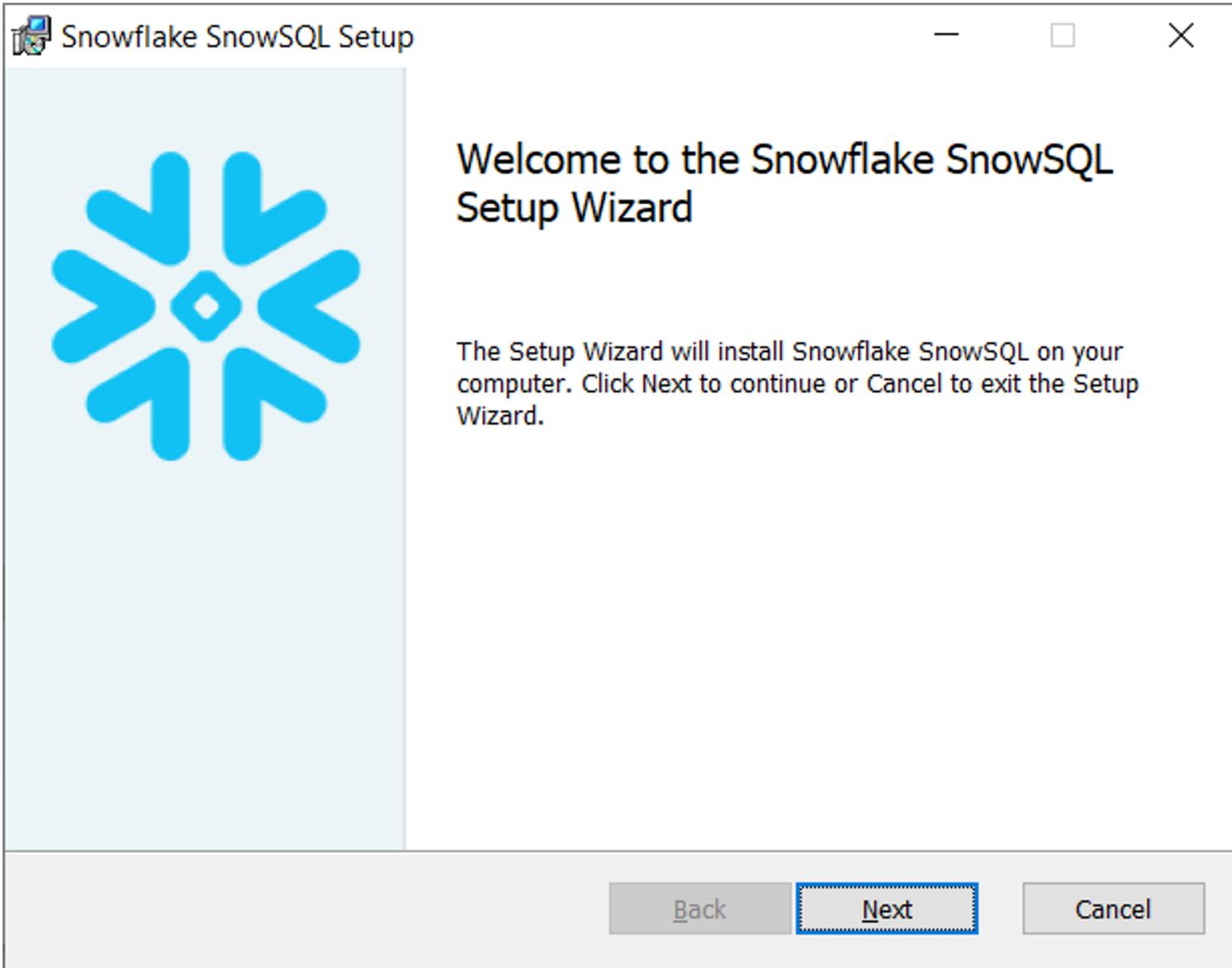
Install on MacOS

```
$ brew install --cask snowflake-snowsql
```

Download the latest Windows installation

https://sfc-repo.snowflakecomputing.com/snowsql/bootstrap/1.2/windows_x86_64/index.html

SNOWSQL CLIENT



SNOWSQL CONNECTION

```
| !result | !result <query id> | See the result of a query
| !set    | !set <option>=<value> | Set an option to the given value
| !source | !source <filename>, <url> | !load | Execute given sql file
| !spool  | !spool <filename>, off | Turn on or off writing results to file
| !system | !system <system command> | Run a system command in the shell
| !variables | !variables | !vars | Show all variables and their values
```

```
+-----+
-----+
susanmartin#COMPUTE_WH@DEMO_DB.PUBLIC>!exit
Goodbye!
```

```
c:\Users\kubernetes\.snowsql>snowsql -a LBA64704.us-east-1 -u susanmartin -d snowflake_sample_data -s tpcds_sf100tcl
Password:
* SnowSQL * v1.2.20
Type SQL statements or !help
susanmartin#COMPUTE_WH@SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL>select cc_name, cc_manager from call_center
```

CALL_CENTER

SNOWFLAKE COMMUNITY



COMMUNITY

JOIN THE SNOWFLAKE COMMUNITY!

Connect, share, and learn in the Data Cloud



DATA DEFINITION LANGUAGE (DDL)

- ALTER <object>
- COMMENT
- CREATE <object>
- DESCRIBE <object>
- DROP <object>
- SHOW <objects>
- USE <object>

DATA MANIPULATION LANGUAGE (DML)

Commands for inserting, deleting, updating, and merging data in Snowflake tables:

- INSERT
- INSERT (multi-table)
- MERGE
- UPDATE
- DELETE
- TRUNCATE TABLE

DATA MANIPULATION LANGUAGE (DML)

Data Loading / Unloading

Commands for bulk copying data into and out of Snowflake tables:

- COPY INTO <table> (loading/importing data)
- COPY INTO <location> (unloading/exporting data)

FILE STAGING COMMANDS

Data Loading / Unloading

These commands do not perform any actual DML, but are used to stage and manage files stored in Snowflake locations (named internal stages, table stages, and user stages), for the purpose of loading and unloading data:

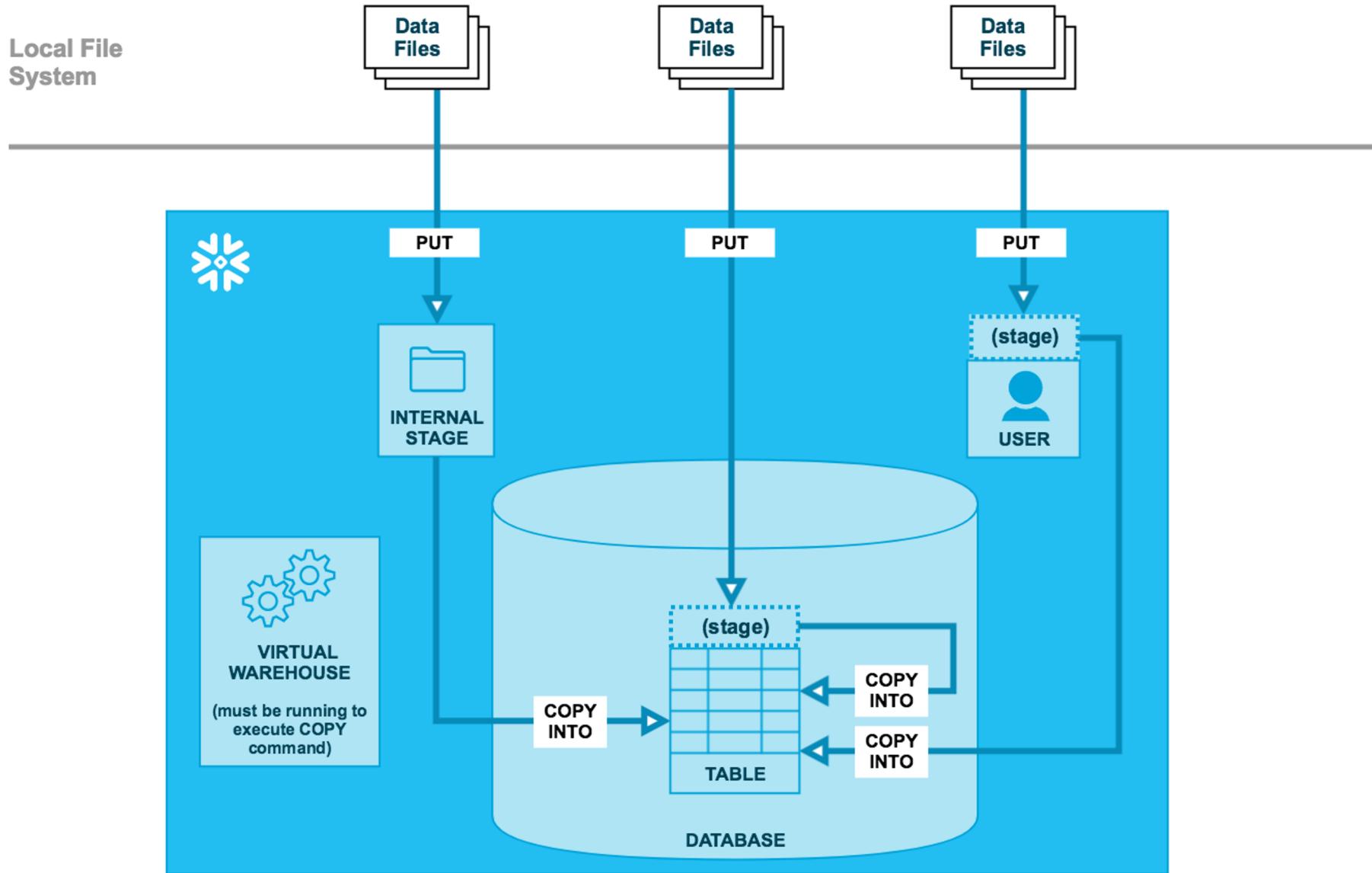
PUT
GET
LIST
REMOVE

ACCOUNT LOCATOR

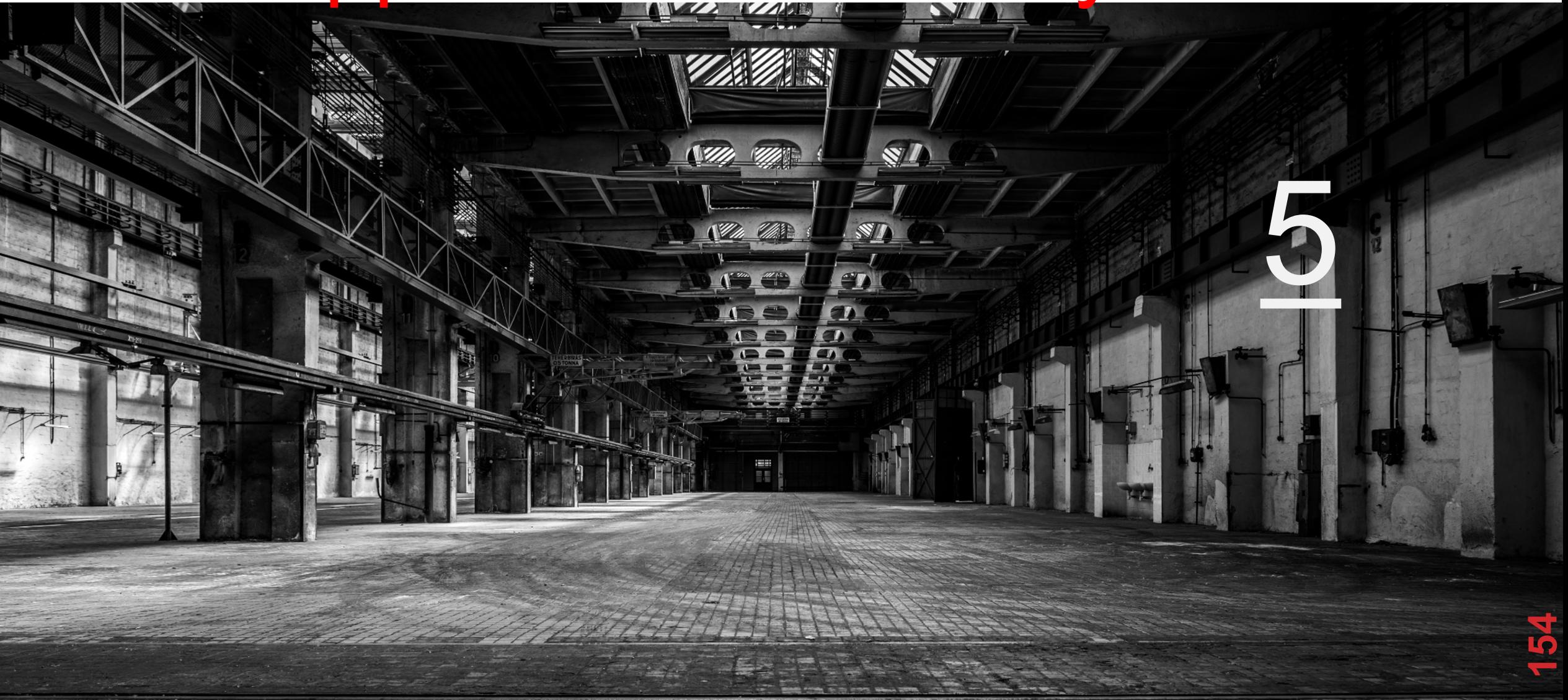
The screenshot shows the Snowflake Cloud UI interface. At the top, there is a navigation bar with several tabs: "Worksheet" (selected), "Snowflake Re", "New Tab", "Settings - Pas", and "New Tab". Below the navigation bar is a toolbar with back, forward, and refresh buttons, along with a URL bar displaying the address: "lba64704.us-east-1.snowflakecomputing.com/console#/internal/worksheet". A promotional message "Enjoy your free trial! Visit our documentation to learn more about using S" is visible. The main menu bar includes icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets (selected), History, and Account. On the left, a sidebar titled "New Worksheet" allows users to "Find database objects" by entering a search term "Starting with...". The results show two databases: "HOUSEHOLD_DEMOGRAPHICS" and "INCOME_BAND". In the main workspace, a query is displayed:

```
1 select cc_name,cc_manager from
2 "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL".
```

LOCAL ONLY RESOURCES

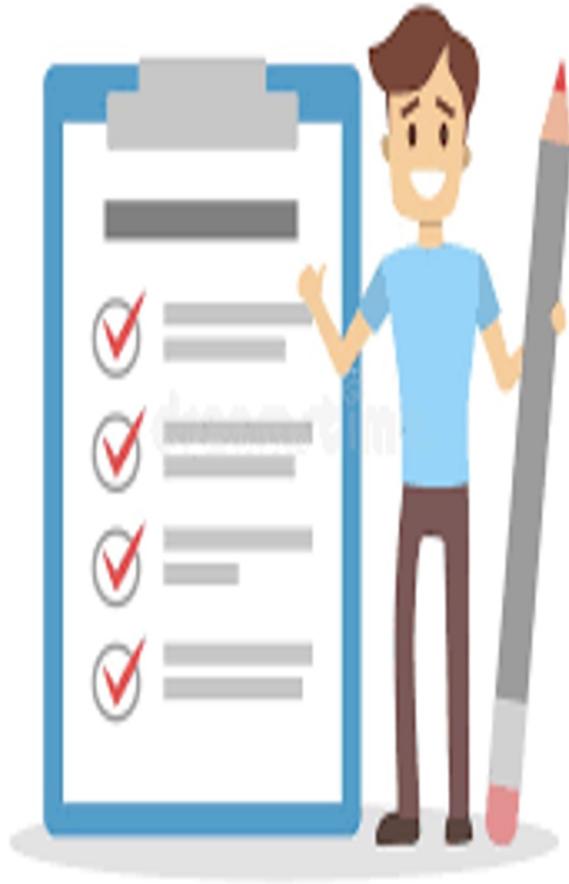


SQL Support for Data Analysis

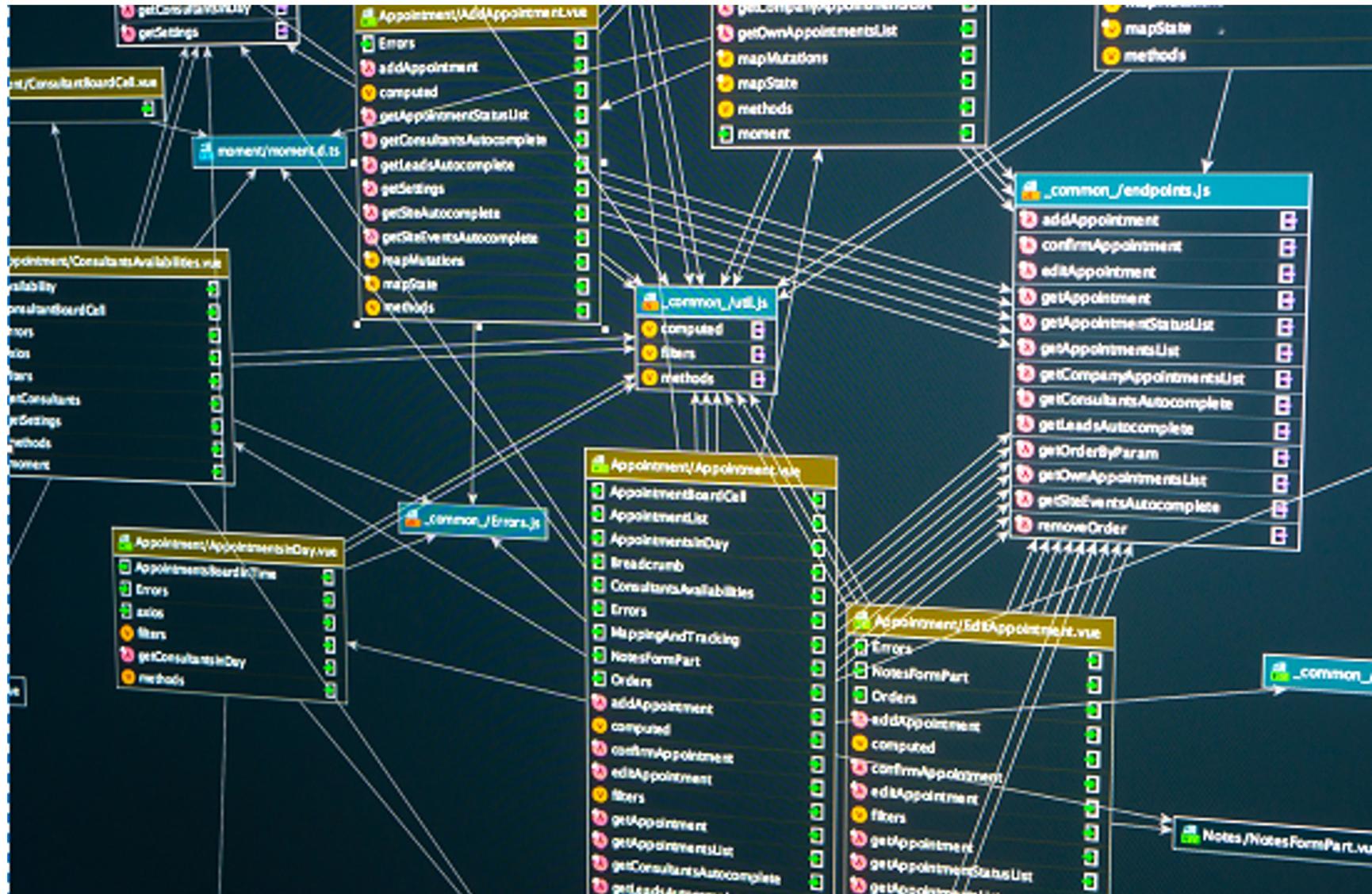


What We have Covered so far

- Snowflake Query
- Snowflake Web UI
- Snow SQL
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Account Locator



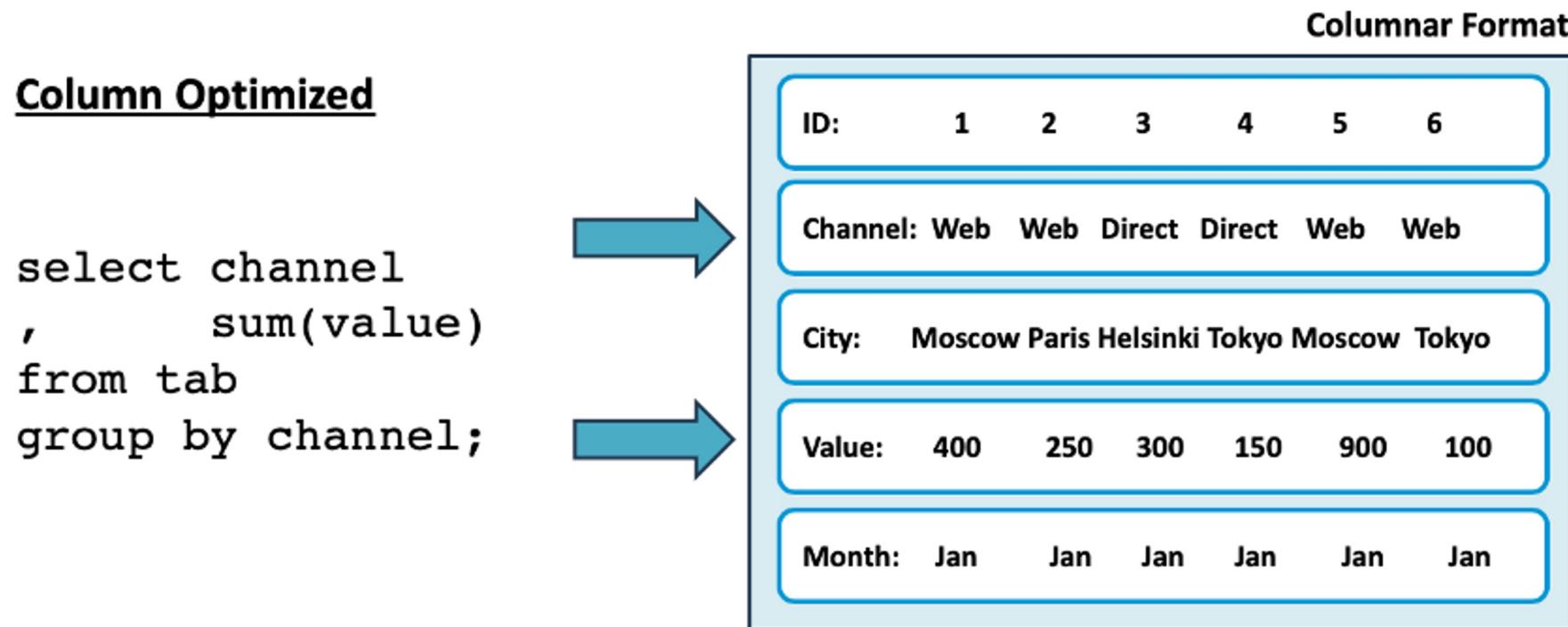
SQL SUPPORT



QUERY BEST PRACTICES

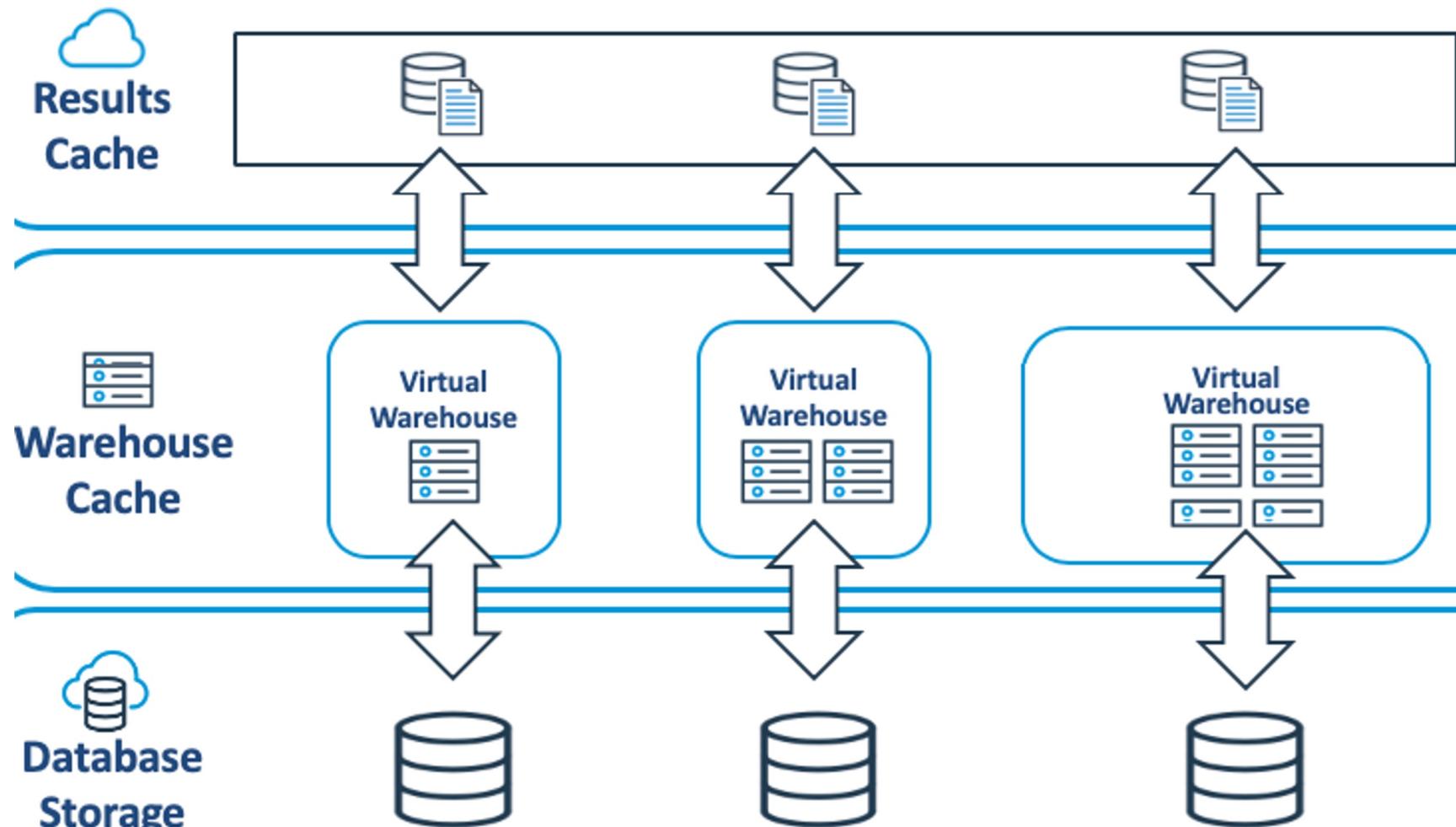
Select the Required Columns

Like many other data analytics platforms, Snowflake uses a columnar data store which is optimized to fetch only the attributes needed for the specific query, and this is illustrated in the diagram below:



QUERY BEST PRACTICES

Maximize cache usage



QUERY BEST PRACTICES

Large query optimization

- **Table Scans:** A high value of PCT_TABLE_SCAN and a large number of MB_SCANNED indicates potential poor query selectivity on large tables. Check the query WHERE clause and consider using a cluster key if appropriate.
- **Spilling:** Any value in SPILL_TO_LOCAL or SPILL_TO_REMOTE indicates a potentially large sort of operation on a small virtual warehouse. Consider moving the query to a bigger warehouse or scaling up the existing warehouse if appropriate.

QUERY BEST PRACTICES

Virtual Warehouse Cache Optimization

- **Fetch required attributes:** Avoid using SELECT * in queries as this fetches all data attributes from Database Storage to the Warehouse Cache.
- **Scale Up:** While you should never scale up to tune a specific query, scaling up adds additional servers, increasing the overall warehouse cache size.
- **Consider Data Clustering:** For tables over a terabyte in size, consider creating a cluster key to maximize partition elimination. This solution both maximizes query performance for individual queries and returns fewer micro-partitions making the best use of the Warehouse Cache.

SQL ANALYTIC FUNCTIONS

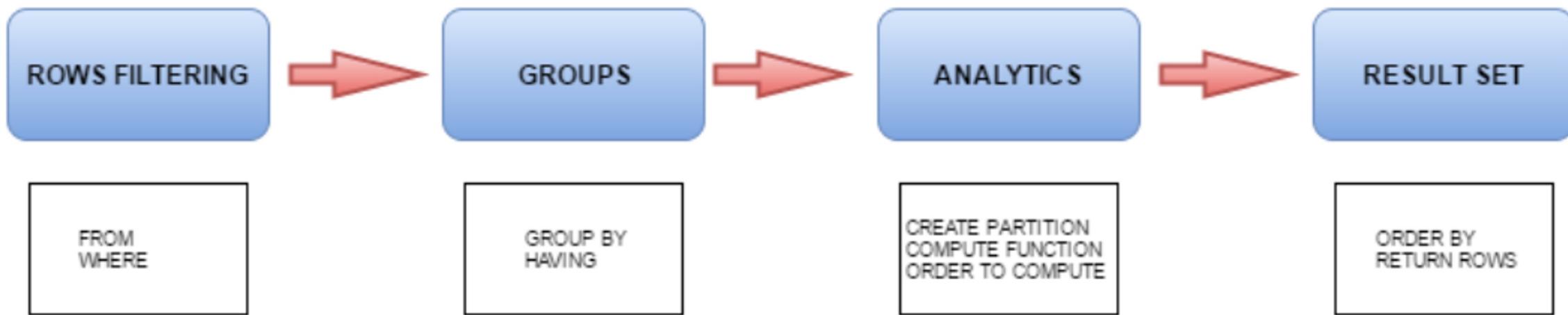
Snowflake includes sophisticated ***analytic*** and ***windowing*** functions as part of our data warehouse service. These functions use ANSI compliant **SQL**. This includes but is not limited to general aggregation functions (e.g. sum), nested virtual tables, sub queries, order by, and group by.

Functions like:

- CUME_DIST
- DENSE_RANK
- FIRST_VALUE
- LAG
- LAST_VALUE
- LEAD
- NTILE
- PERCENT_RANK
- RANK
- ROW_NUMBER

ANALYTIC FUNCTIONS FLOW

Executing analytical functions organizes data into partitions, computes functions over these partitions in a specified order, and returns the result.



HIGH PERFORMING ESTIMATION FUNCTIONS

The following Aggregate Functions are provided for using Space-Saving to estimate frequent values:

- APPROX_TOP_K: Returns an approximation of frequent values in the input.
- APPROX_TOP_K_ACCUMULATE: Skips the final estimation step and returns the Space-Saving state at the end of an aggregation.
- APPROX_TOP_K_COMBINE: Combines (i.e. merges) input states into a single output state.
- APPROX_TOP_K_ESTIMATE: Computes a cardinality estimate of a Space-Saving state produced by APPROX_TOP_K_ACCUMULATE and APPROX_TOP_K_COMBINE.

USER DEFINED FUNCTION (UDF)

User-defined functions (UDFs) let you extend the system to perform operations that are not available through the built-in, system-defined functions provided by Snowflake.

Snowflake currently supports the following languages for writing UDFs:

- **SQL**: A SQL UDF evaluates an arbitrary SQL expression and returns either scalar or tabular results.
- **JavaScript**: A JavaScript UDF lets you use the JavaScript programming language to manipulate data and return either scalar or tabular results.
- **Java**: A Java UDF lets you use the Java programming language to manipulate data and return either scalar or tabular results.

OVERLOADING UDF

Snowflake supports overloading of SQL UDF names. Multiple SQL UDFs in the same schema can have the same name, as long as their argument signatures differ, either by the number of arguments or the argument types. When an overloaded UDF is called, Snowflake checks the arguments and calls the correct function.

Consider the following examples, which create two SQL UDFs named add5:

```
create or replace function add5 (n number)
    returns number
    as 'n + 5';
```

```
create or replace function add5 (s string)
    returns string
    as 's || ''5''';
```

STORED PROCEDURE

Stored procedures are loosely **similar to functions**. As with functions, a stored procedure is created once and can be executed many times. A stored procedure is created with a **CREATE PROCEDURE** command and is executed with a **CALL** command.

A stored procedure returns a single value. Although you can run **SELECT** statements inside a stored procedure, the results must be used within the stored procedure, or be narrowed to a single value to be returned.

STORED PROCEDURE PROCESSING

Snowflake stored procedures use JavaScript and, in most cases, SQL:

- JavaScript provides the control structures (branching and looping).
- SQL is executed by calling functions in a JavaScript API.



SNOWPARK

The Snowpark library provides an intuitive API for querying and processing data in a data pipeline.

Using this library, you can build applications that process data in Snowflake without moving data to the system where your application code runs.

Experiment



Content
#05-Snowflake-AnalyticsInAction.pdf

Scripts
foundation_experiment_scripts.sql

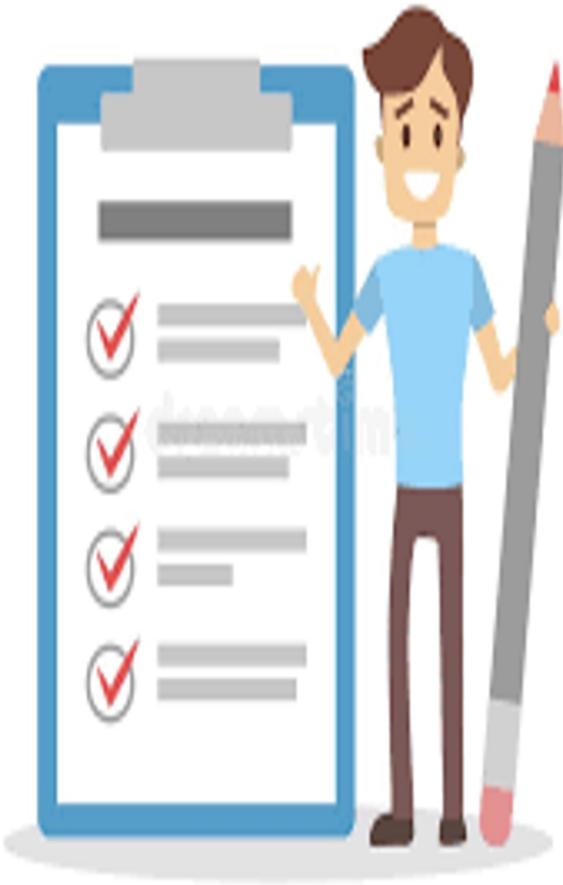
Managing Security

In Snowflake

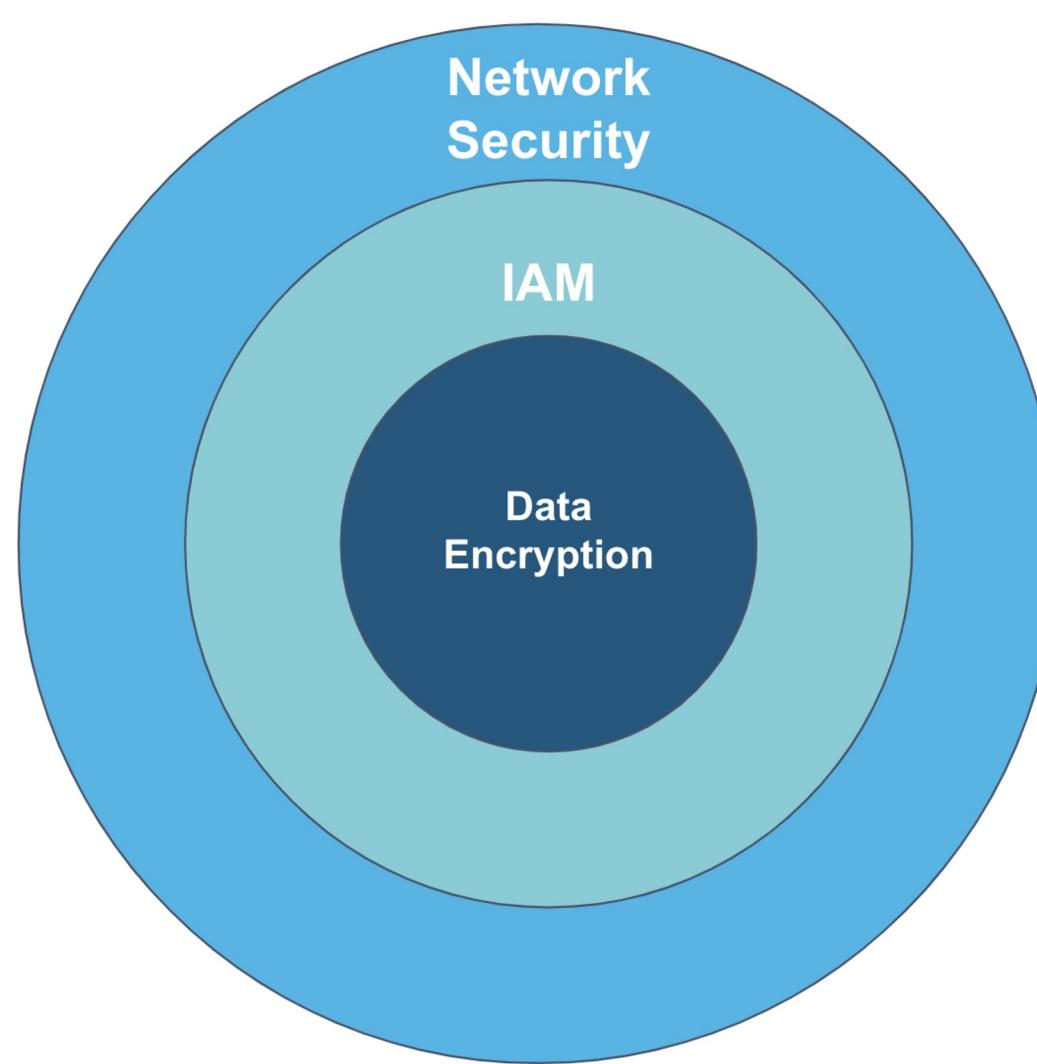


What We have Covered so far

- SQL Support
- Query Best Practices
- SQL Analytics Functions/ Flow
- User Defined Function (UDF)
- Stored Procedure Processing
- SnowPark



DEFENSE IN DEPTH



DATA ENCRYPTION

Encrypting data means applying an encryption algorithm to translate the clear text into cipher text.

Encrypt data:

- from the time it leaves your premises
- through the Internet
- into the cloud data warehouse
- stored on disk
- moved into a staging location
- placed within a database object
- cached within a virtual data warehouse.

AUTHENTICATION

Any data service or data warehouse should always authorize users, authenticate credentials, and grant users access only to the data they're authorized to access.

Data breaches often result from users selecting weak passwords coupled with rudimentary authentication procedures.

IDENTITY VS ACCESS

Identity – who you are

Access – what you're allowed to do

Authentication – proving your identity, e.g user SUSANMARTIN with password Sn0wFl@ke

Authorization – the right to do something,
SUSANMARTIN – ACCOUNTADMIN aka Snowflake superuser

AUTHORIZATION AND RBAC

Snowflake authorization allows you to do things in the data warehouse **based on your role**, that's **called Role-Based Access Control (RBAC)**.

ROLE SWITCHING

Snowflake allows you to **switch roles** without doing further authentication.

The standard roles are
ACCOUNTADMIN
SYSADMIN
SECURITYADMIN
USERADMIN
PUBLIC

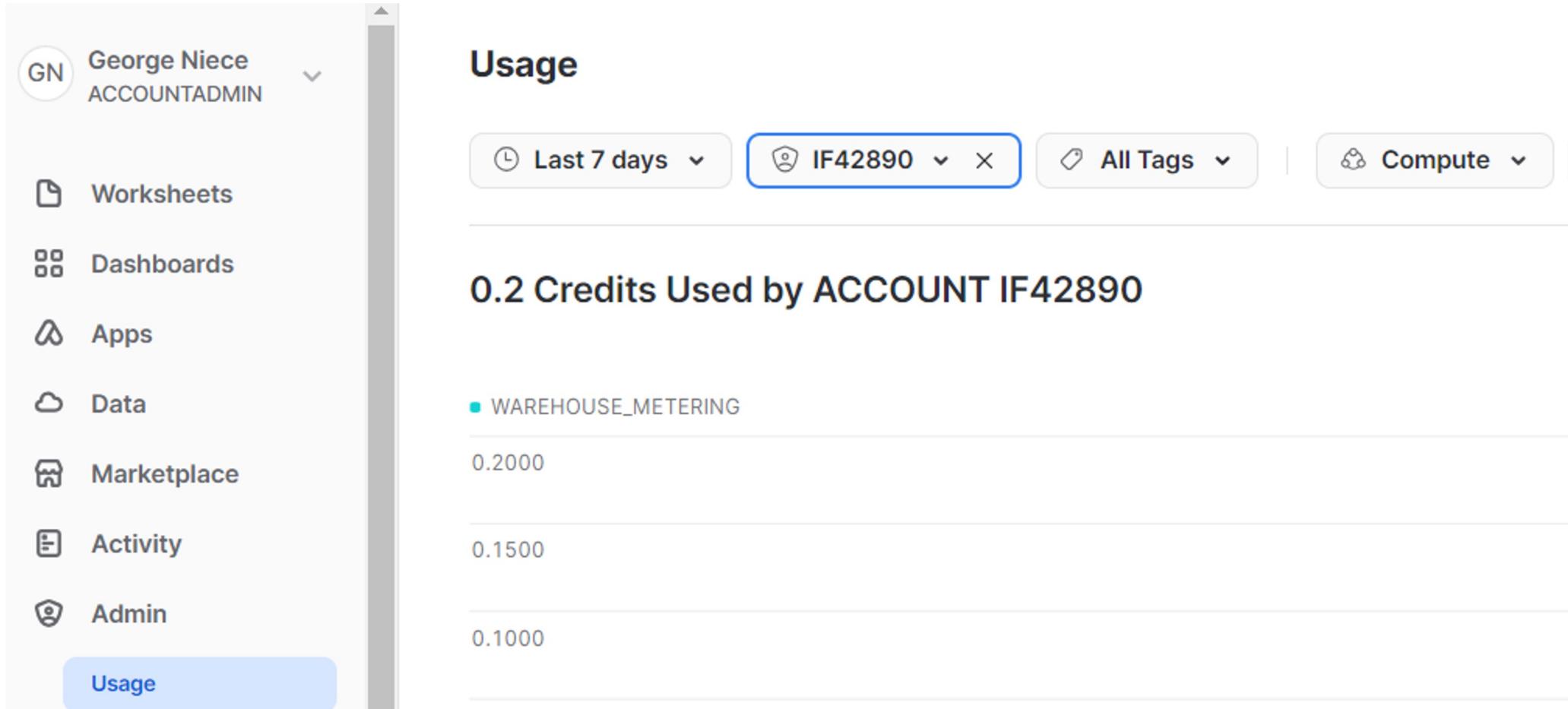
PUBLIC DATABASE ACCESS

Snowflake provides functions to view role availability. For example, to see what roles you have available you could execute

```
select current_available_roles();
```

USERS & ROLES

ACCOUNTADMIN is the only role that has default access to the Account Usage in the Snowflake Console



MANAGING USERS AND ROLES

Snowflake recommends using SCIM where supported by your Identity Provider to provision and externally manage users and roles in Snowflake.

Identity Providers can be further configured to synchronize users and roles with your Active Directory users and groups. Examples include Okta SCIM and Azure AD SCIM



SAML OR OAUTH

SAML and OAuth are great protocols within identity management. SAML is based around user authentication and OAuth is based around authorization.

If you're using an external identity provider such as Okta, it's likely you have the ability to use both SAML and OAuth for a variety of applications and integrations. Specifically with regards to Okta, it would be recommended to use OAuth because the specification has more levels of trust and security, and you can configure different authorization flows for each one of these.

In either case, this is an area that requires experience and Subject Matter Expertise (SME) to most correctly secure.

ACCESS CONTROL FRAMEWORK

Snowflake's approach to access control combines aspects from both of the following models:

- **Discretionary Access Control (DAC):** Each object has an owner, who can in turn grant access to that object.
- **Role-based Access Control (RBAC):** Access privileges are assigned to roles, which are in turn assigned to users.

ACCESS CONTROL KEY CONCEPTS

Securable object: An entity to which access can be granted. Unless allowed by a grant, access will be denied.

Role: An entity to which privileges can be granted. Roles are in turn assigned to users. Note that roles can also be assigned to other roles, creating a role hierarchy.

Privilege: A defined level of access to an object. Multiple distinct privileges may be used to control the granularity of access granted.

User: A user identity recognized by Snowflake, whether associated with a person or program.

SNOWFLAKE AUTHENTICATION METHODS

Snowflake Drivers, CLI

Password

OAuth

Key Pair

External Browser

Okta native auth

Snowflake UI

Password

SAML

Snowpipe

Key Pair

AUTHENTICATION BEST PRACTICE

Preference #1: OAuth (either Snowflake OAuth or External OAuth)

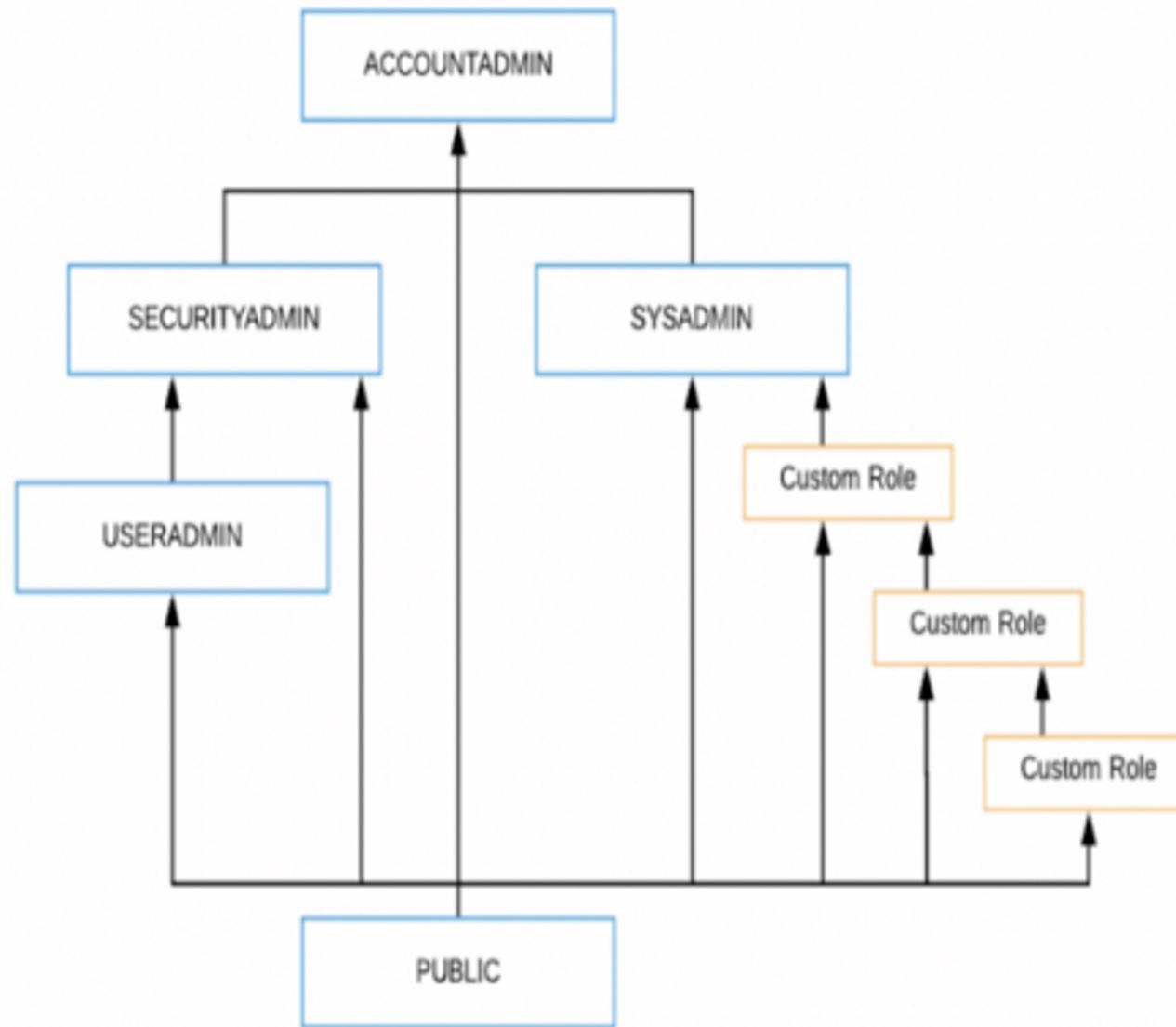
Preference #2: External Browser, if it's a desktop application that doesn't support OAuth

Preference #3: Okta native authentication, if you're using Okta, and the app supports this method while not supporting OAuth or external browser authentication yet.

Preference #4: Key Pair Authentication, mostly used for service account users. Since this requires the client application to manage private keys, complement it with your internal key management software.

Preference #5: Password, this should be the last option for applications that don't support any of the above options. This option is commonly used for service account users connecting from 3rd party ETL apps.

ROLE-BASED ACCESS CONTROL (RBAC)



LOGICAL RBAC LEVELS

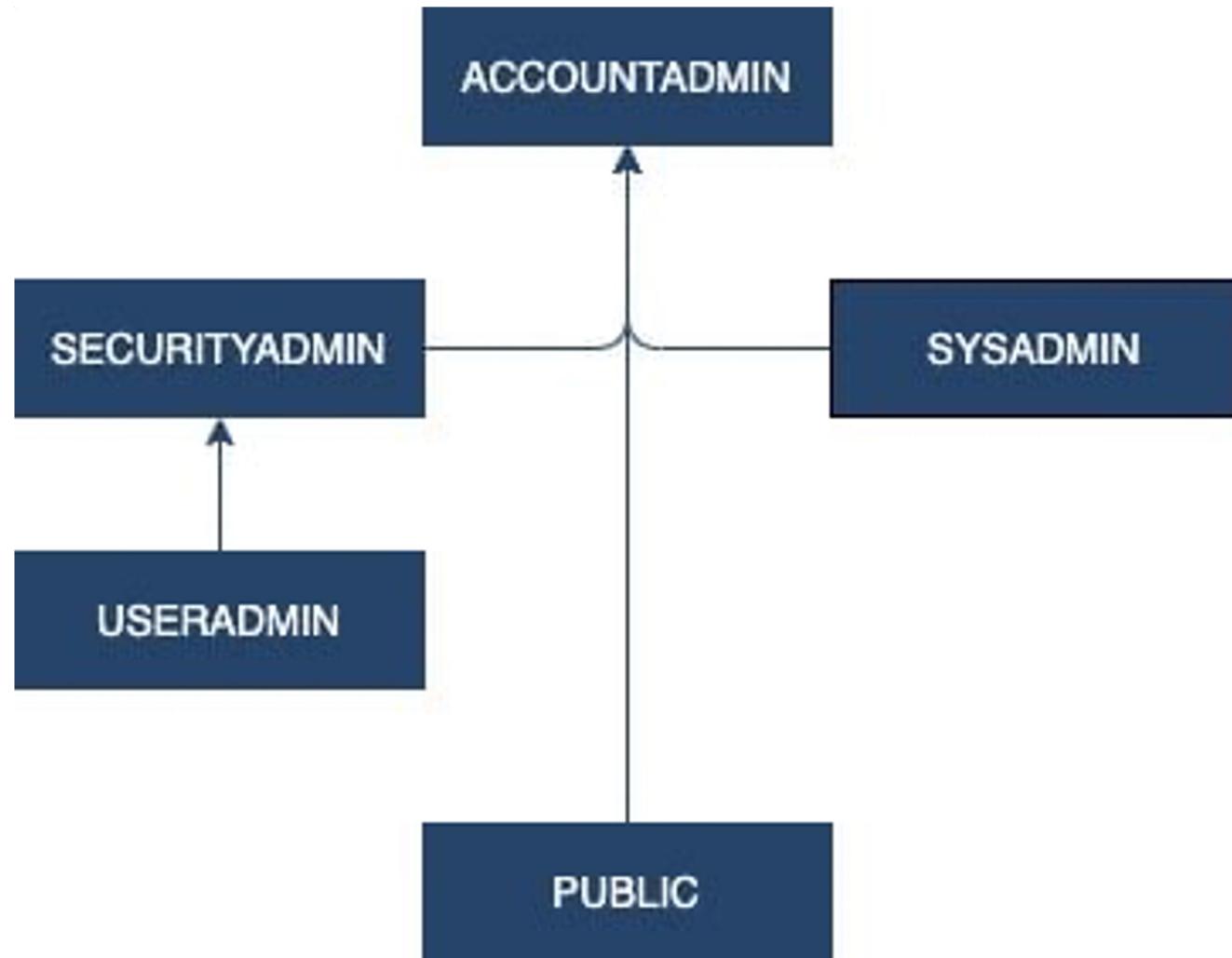
Access Roles: These roles will be the lowest level which will have actual access privileges on DB objects. Maintain distinct access roles based on user requirements at a schema level.

Functional Roles: These roles map to the actual real-world roles of the users in the organization and are assigned to the Snowflake users.

Domain Roles: If the organization has a requirement to have multiple independent domains under the same account, this will help to realize that. E.g. when there needs to be a separation of production or UAT/development domains.

System Roles: Native Snowflake system role to which all the Domain roles should be rolling up to.

SYSTEM-DEFINED ROLES



SYSTEM-DEFINED ROLES

ACCOUNTADMIN

This is the highest level [role in Snowflake](#). It should be heavily limited in access and encapsulates SYSADMIN and SECURITYADMIN

SECURITYADMIN

This role is used for managing any [object grant](#) globally, and therefore is granted the MANAGE GRANTS privilege by default. This role also inherits the USERADMIN role.

USERADMIN

This role is dedicated to user/role management. By default, this role can create/modify users and their respective roles (assuming those roles/users haven't been transferred to another role).

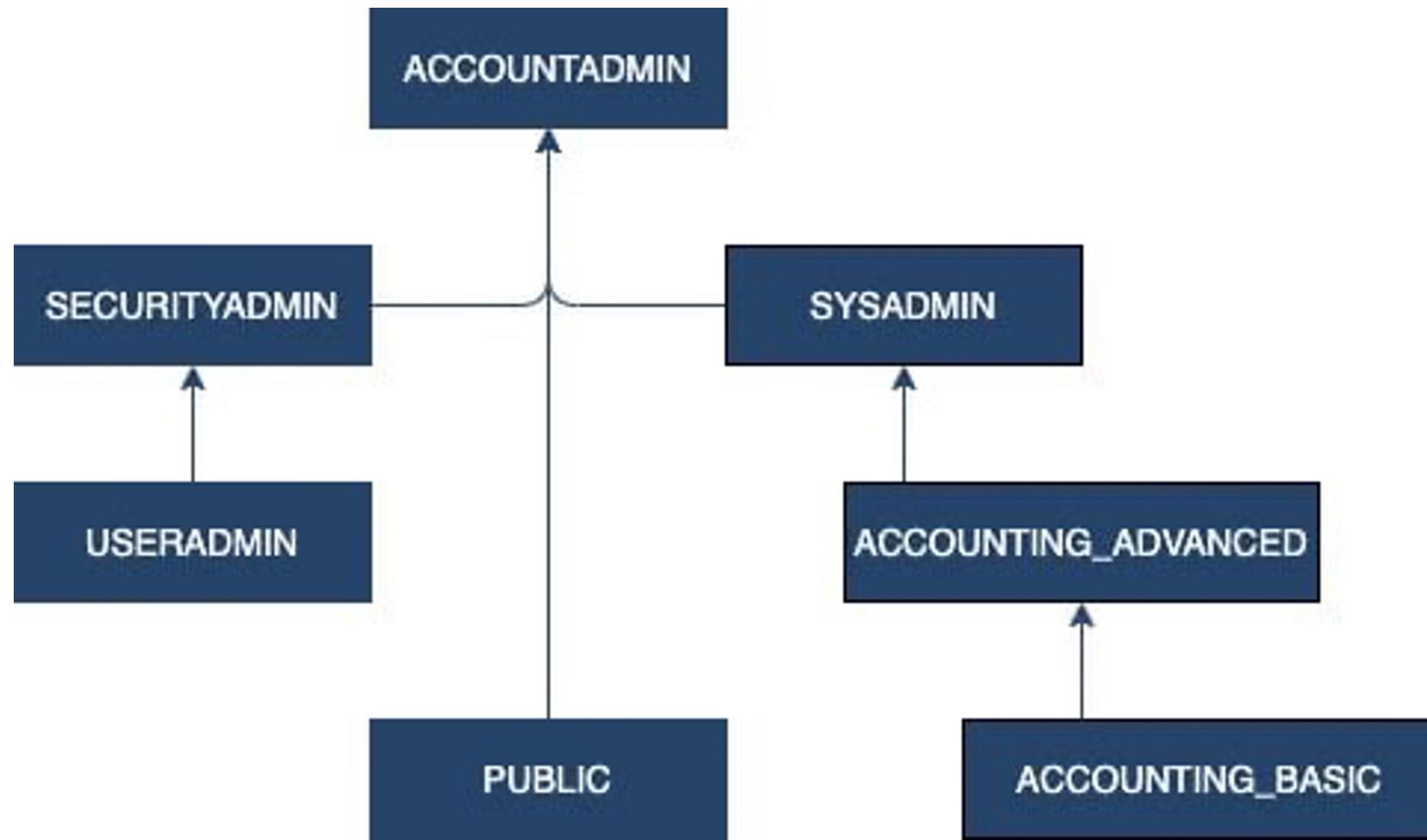
SYSADMIN

This role is dedicated to system object management. It is recommended by [Snowflake to assign all custom roles](#) to the SYSADMIN role, so this role can grant these privileges to other roles.

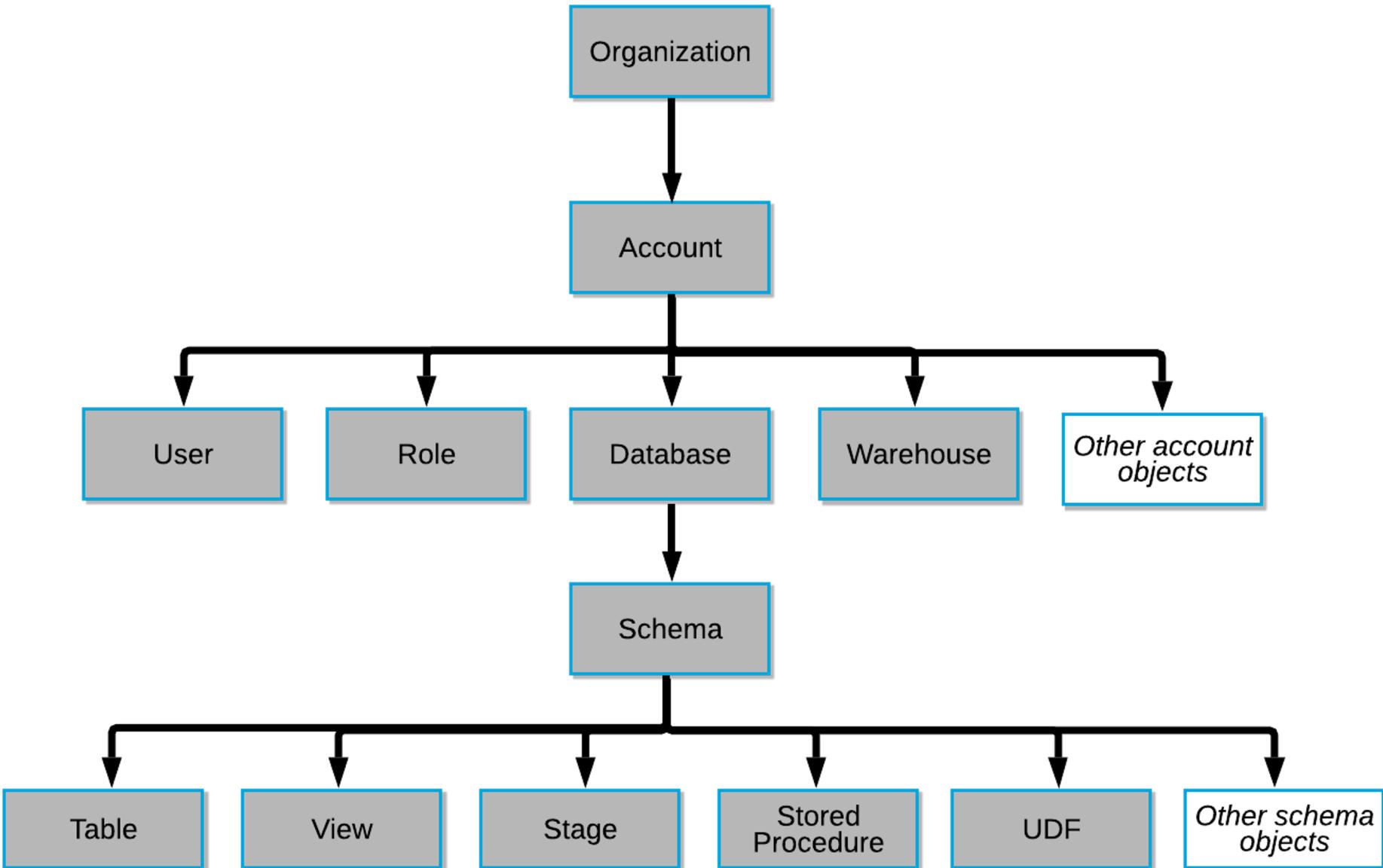
PUBLIC

This is a pseudo role that every user/role in the account gets. Things owned by the public role are, well, public!

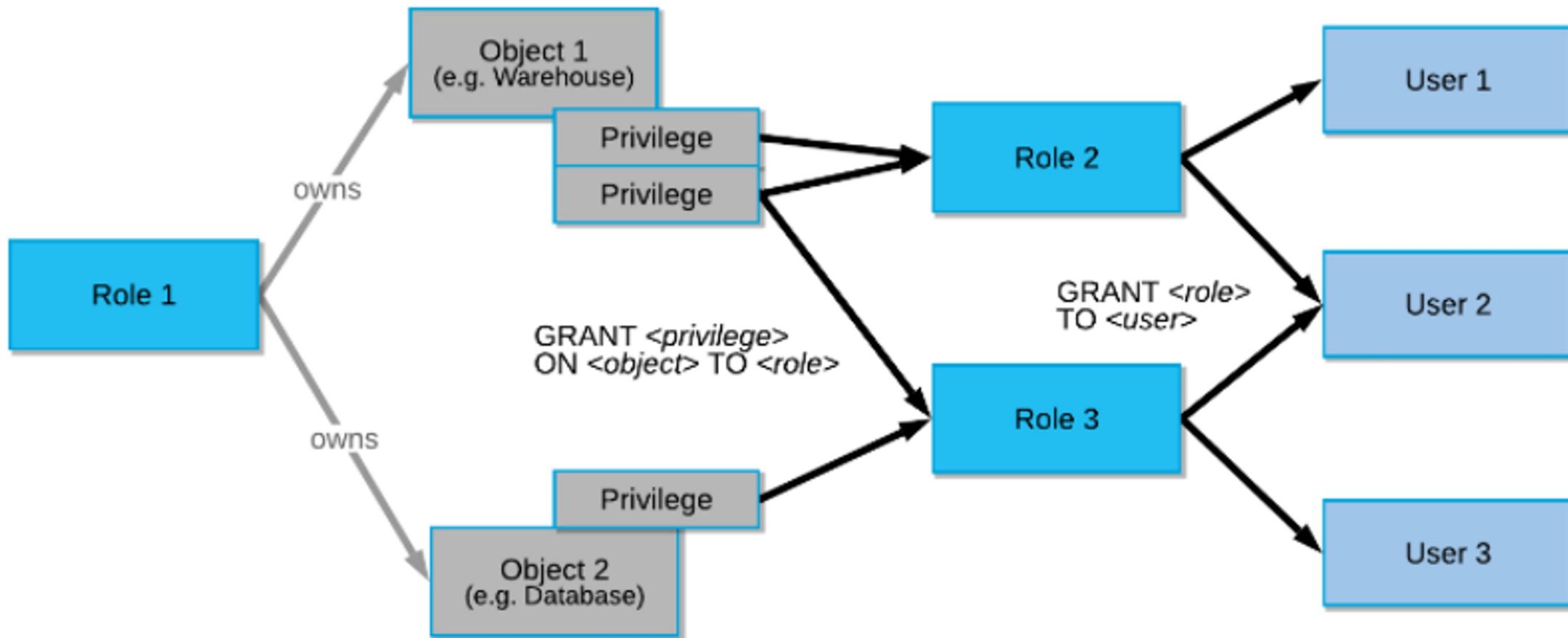
CUSTOM ROLES



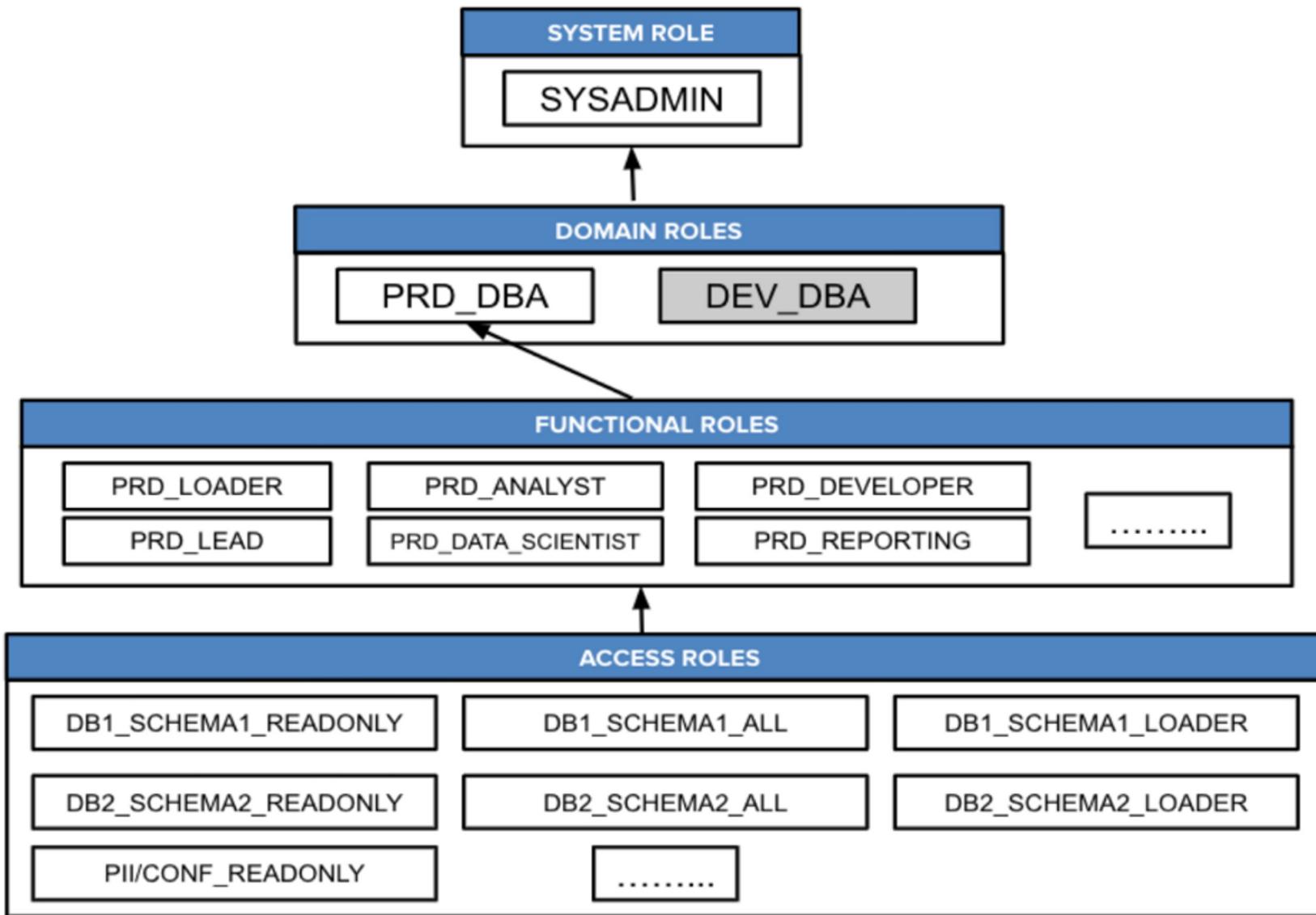
SECURABLE OBJECTS



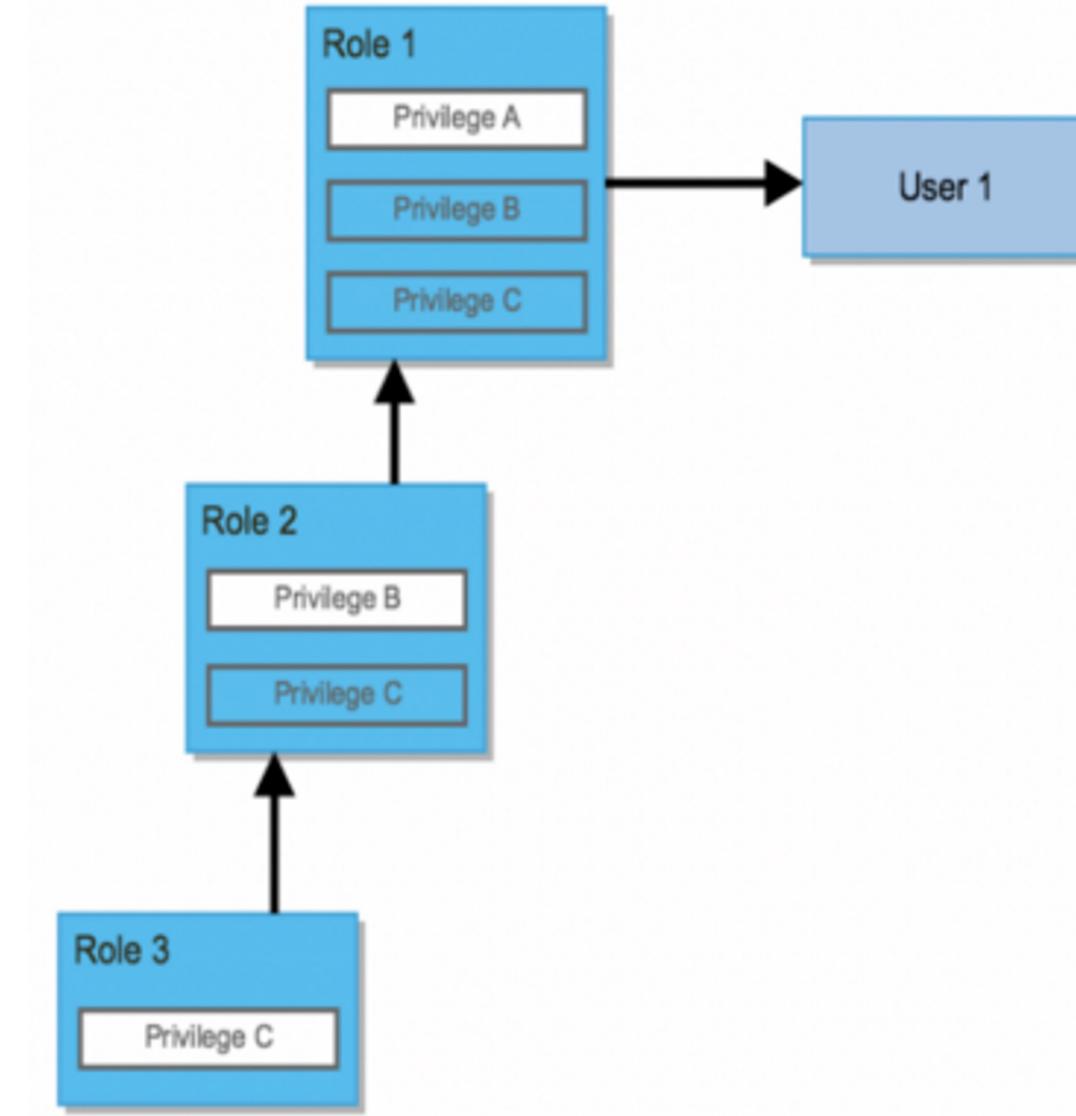
PRIVILEGES



RBAC ROLE HIERARCHY



PRIVILEGE INHERITANCE



DEFAULT TRIAL OWNER ROLE

The screenshot shows the Snowflake interface with a sidebar on the left and a main content area on the right.

Left Sidebar:

- Admin
- Usage
- Warehouses
- Resource Monitors
- Users & Roles
- Security
- Billing & Terms
- Contacts
- Accounts** (highlighted with a blue background)
- Partner Connect
- Help & Support

Main Content Area:

Accounts

1 Account in PZQCZHA ⓘ

ACCOUNT ↑	EDITION
IF42890	Enterprise

Account Details for GEOFFSNOW55113:

Disable User	Reset Password	Drop
Login Name	GEOFFSNOW55113	
Display Name	GEOFFSNOW55113	
Default Role	ACCOUNTADMIN	
Default Warehouse	COMPUTE_WH	
Default Namespace		
MFA		
Last Login	Today at 3:07:12 AM	
Status	Enabled	

POP QUIZ:

Managing Security



Follow the link for Security Management Quiz:

<https://forms.gle/57kSuoPxTuQDGeqQ6>



Data

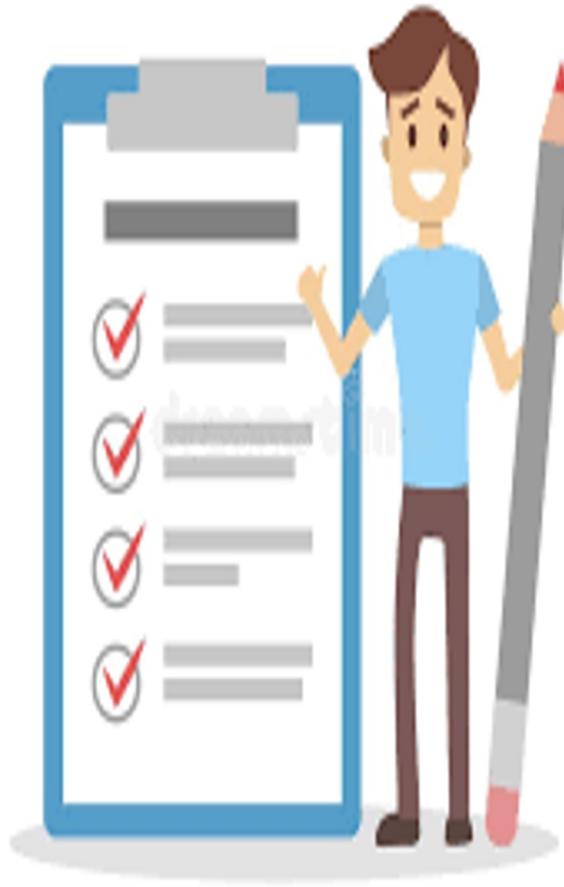
Semi-structured/Binary/Structured/Time

7

197

What We have Covered so far

- Data Encryption/ Authentication
- Identity vs Access
- Authorisation and RBAC
- Users and Roles
- SAML and OAUTH
- Access Control Framework
- Snowflake Authentication
- System Defined Roles
- Custom Roles
- Privileges



BINARY

Snowflake supports three binary formats or encoding schemes:

- Hex - The “hex” format refers to the hexadecimal, or base 16, system. In this format, each byte is represented by two characters (digits from 0 to 9 and letters from A to F).
- Base64 - The “base64” format encodes binary data (or string data) as printable ASCII characters (letters, digits, and punctuation marks or mathematical operators). (The base64 encoding scheme is defined in RFC 4648.)
- UTF-8 - The UTF-8 format refers to the UTF-8 character encoding for Unicode. Unlike hex and base64, which are binary-to-text encodings, UTF-8 is a text-to-binary encoding.

SEMI-STRUCTURED

Data can come in multiple forms from numerous sources, including an ever-expanding amount of machine-generated data from applications, sensors, mobile devices, etc.

To support these new types of data, semi-structured data formats, such as JSON, Avro, ORC, Parquet, and XML, with their support for flexible schemas, have become popular standards for transporting and storing data.



SEMI-STRUCTURED DATA CONSIDERATIONS

Snowflake provides native support for semi-structured data, including:

- Flexible-schema data types for loading semi-structured data without transformation.
- Automatic conversion of data to optimized internal storage format.
- Database optimization for fast and efficient SQL querying.

SEMI-STRUCTURED DATA CONSIDERATIONS

File Size – Best load performance is typically around 80MB file chunk. It may vary in your case. As per Snowflake recommendations, it should be between 10 to 100MB for best performance.

Complex and binary data types – If you have complex and binary (varbyte, BLOB, etc.) datatypes that you want to migrate, then you will have to do proper encoding for CSV. Hex is usually best performance.

Validation errors – If you want to capture load error messages and records using Snowflake's validation function, then CSV is the only option.

JSON BASICS

Machine data can be generated by a variety of devices, such as servers, cell phones, browsers, and so on.

Messages sent from a device are called *events*. An event describes any single user action or occurrence that you want to track.

Machines that collect large numbers of events may organize them into *batches*.

A batch is a container that includes header information common to all of the events; e.g. source device and user information.

JSON BASICS

In a common data collection scenario, a scalable web endpoint collects POSTed data from different sources and writes them to a queuing system such as Amazon Kinesis, Apache Kafka, or RabbitMQ.

An ingest service/utility then writes the data to an object storage bucket e.g. S3 or Blob Storage, from which you can load the data into Snowflake.



JSON DATA PARTITIONING

It is important to partition the event data in your S3 bucket using logical, granular paths.

Create a partitioning structure that includes identifying details such as application or location, along with the date when the event data was written to the S3 bucket.

You can then copy any fraction of the partitioned data into Snowflake with a single command.

You can copy data into Snowflake by the hour, day, month, or even year when you initially populate tables.

JSON DATA PARTITIONING ON S3

s3://bucket_name/application_one/2016/07/01/11/

s3://bucket_name/application_two/location_one/2016/07/01/14/

Where:

bucket_name

The unique S3 URI used to access your data.

application_one , application_two , location_one , etc.

Identifying details for the source of all data in the path. The data can be organized by the date when it was written. An optional 24-hour directory reduces the amount of data in each directory.

TIME SERIES

Snowflake's date, time, timestamp, and time zone support enables you to calculate and store consistent information about the time of events and transactions.

A set of session parameters determines how date, time, and timestamp data is passed into and out of Snowflake, as well as the time zone used in the time and timestamp formats that support time zones.

The parameters can be set at the account, user, and session levels. Execute the [SHOW PARAMETERS](#) command to view the current parameter settings that apply to all operations in the current session.

Look for any parameter matching DATE_XXXX, TIME_XXXX, TIMESTAMP_XXXX

DATE FILE FORMATS

Separate from the input and output format parameters, Snowflake provides three file format options to use when loading data into or unloading data from Snowflake tables:

- DATE_FORMAT
- TIME_FORMAT
- TIMESTAMP_FORMAT

The options can be specified directly in the COPY command or in a named stage or file format object referenced in the COPY command. When specified, these options override the corresponding input formats (when loading data) or output formats (when unloading data).

DATA HIERARCHIES

Many types of data are best represented as a **hierarchy**, such as a **tree**.

For example, employees are usually organized in a hierarchy, with a company President at the top of the hierarchy.

Another **example** of a hierarchy is a “**parts explosion**”. For example, a car contains an engine; an engine contains a fuel pump; and a fuel pump contains a hose.

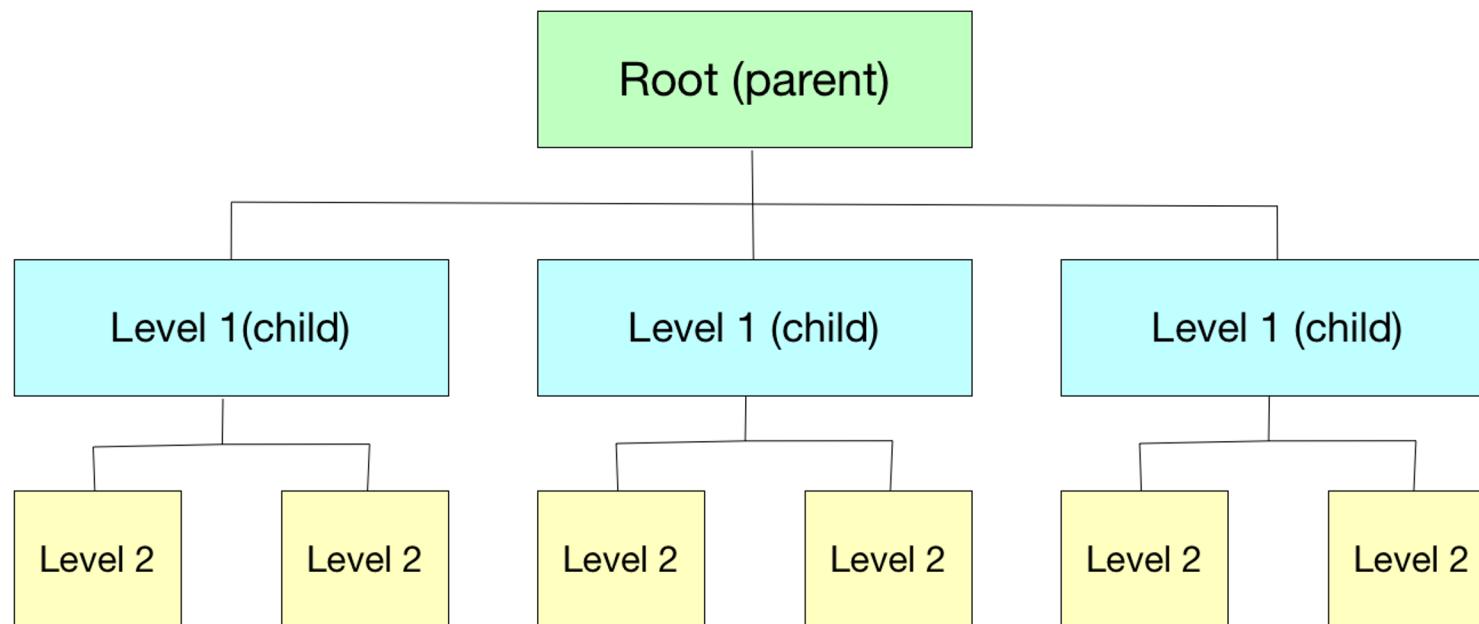
You can store hierarchical data in:

- A hierarchy of tables.
- A single table with one (or more) columns representing the hierarchy (e.g. indicating each employee's direct manager).

MULTI-TABLE RDBMS HIERARCHIES

Relational databases often store hierarchical data by using different tables.

For example, one table might contain “parent” data and another table might contain “child” data. When the entire hierarchy is known in advance, one table can be created for each layer in the hierarchy.



MULTI-TABLE HIERARCHY FLATTENED

In some situations, the number of levels in the hierarchy might change.

For example, a company that started with a two-level hierarchy (President and other employees) might increase the number of levels as the company grows. The company might expand to include a President, Vice Presidents, and regular employees.

If the number of levels is unknown, so that it's not possible to create a hierarchy with a known number of tables, then in some cases the hierarchical data can be stored in one table. For example, a single table can contain all employees, and can include a column that stores each employee's manager_ID, which points to another employee in that same table.

HIERARCHY FLATTENING BY JOIN

In a three-level RDBMS hierarchy, we can use a 3-way join:

```
select
```

```
    emps.title,  
    emps.employee_id,  
    mgrs.employee_id as manager_id,  
    mgrs.title as "MANAGER TITLE"  
  
    from employees as emps left outer join employees as  
    mgrs  
        on emps.manager_id = mgrs.employee_id  
    order by mgrs.employee_id nulls first,  
    emps.employee_id;
```

HIERARCHY FLATTENING BY JOIN

TITLE	EMPLOYEE_ID	MANAGER_ID	MANAGER_TITLE
President	1	NULL	NULL
Vice President Engineering	10	1	President
Vice President HR	20	1	President
Programmer	100	10	Vice President Engineering
QA Engineer	101	10	Vice President Engineering
Health Insurance Analyst	200	20	Vice President HR

QUERYING VARIABLE HIERARCHICAL LEVELS

Snowflake provides two ways to query hierarchical data in which the number of levels is not known in advance:

- Recursive CTEs (common table expressions).
- CONNECT BY clauses.

A **recursive CTE** allows you to create a WITH clause that can refer to itself. This lets you iterate through each level of your hierarchy and accumulate results.

A **CONNECT BY** clause allows you to create a type of JOIN operation that processes the hierarchy one level at a time and allows each level to refer to data in the prior level.

COMMON TABLE EXPRESSIONS (CTE)

A CTE (common table expression) is a named subquery defined in a WITH clause.

You can think of the CTE as a temporary view for use in the statement that defines the CTE.

The CTE defines the temporary view's name, an optional list of column names, and a query expression (i.e. a SELECT statement).

The result of the query expression is effectively a table.

Each column of that table corresponds to a column in the (optional) list of column names.

RECURSIVE CTE

```
with recursive current_layer
  (employee_id, manager_id, sort_key) as (
    -- This allows us to add columns, such as
    sort_key, that are not part
    -- of the employees table.
    select employee_id, manager_id, employee_id
    as sort_key
    ...
  )
```

CONNECT BY IN A NUTSHELL

Joins a table to itself to process hierarchical data in the table. The CONNECT BY subclause of the FROM clause iterates to process the data.

For example, you can create a query that shows a “parts explosion” to recursively lists a component and the sub-components of that component.

The **Snowflake syntax** for **CONNECT BY** is mostly compatible with the **Oracle syntax**.

CONNECT BY

```
select indent(level) || employee_id, manager_id, title  
from employees
```

- This sub-clause specifies the record at the top of the hierarchy,

- but does not allow additional derived fields, such as the sort key.

- start with title = 'President'

- connect by ...

DYNAMIC QUERY OPTIMIZATION

Snowflake you don't need to pre-plan partitioning or distribution keys, or build indexes to get great performance.

That is all handled as part of our Dynamic Query Optimization feature that uses our secure cloud-based metadata store and sophisticated feedback loop to monitor and tune your queries based on data access patterns and resource availability among other things.

Experiment



Content
#06-Snowflake-
Data-
Wrangling.pdf

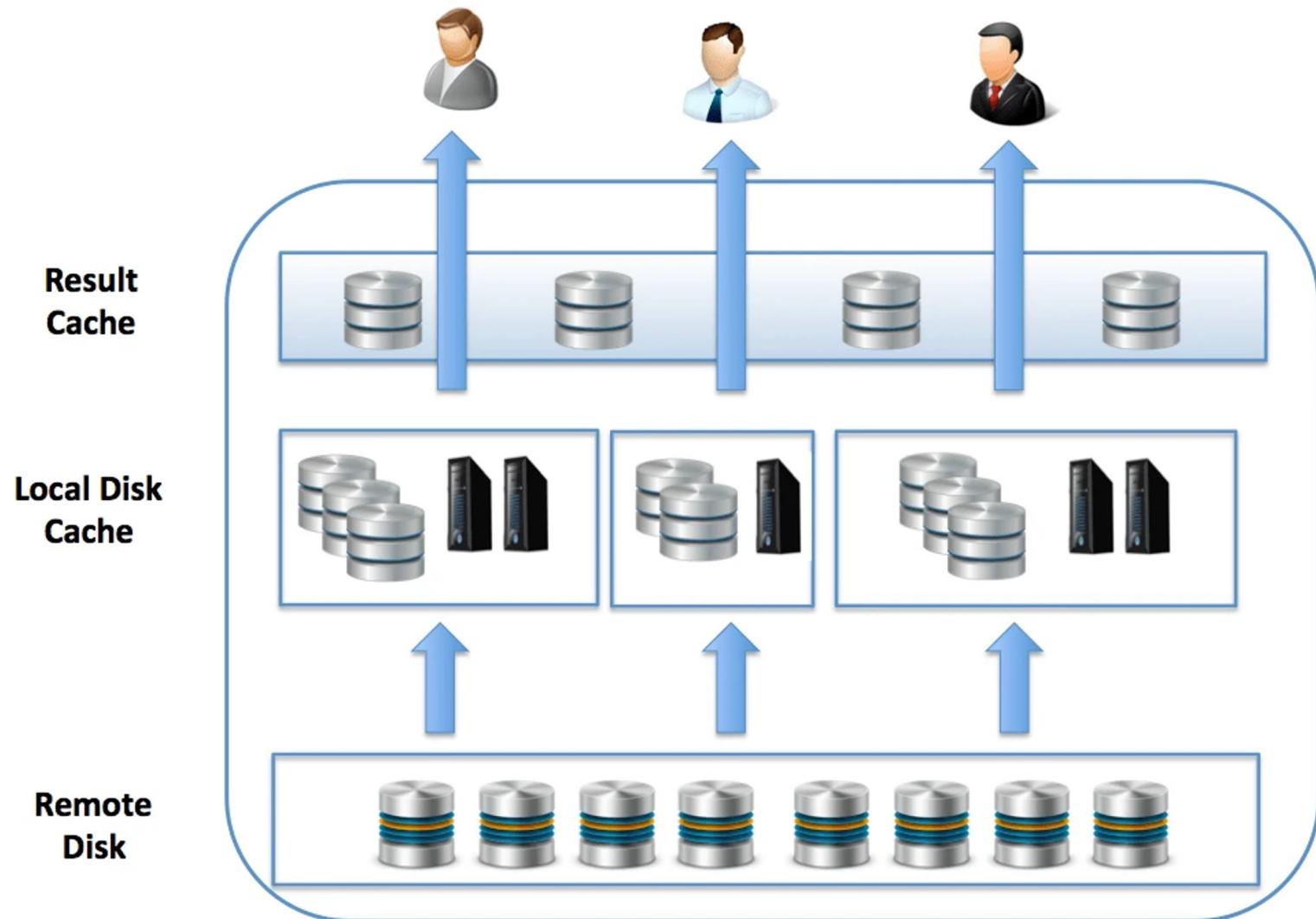
Scripts
foundation_experi
ment_scripts.sql

Caching

Kickstart



CACHING FEATURES



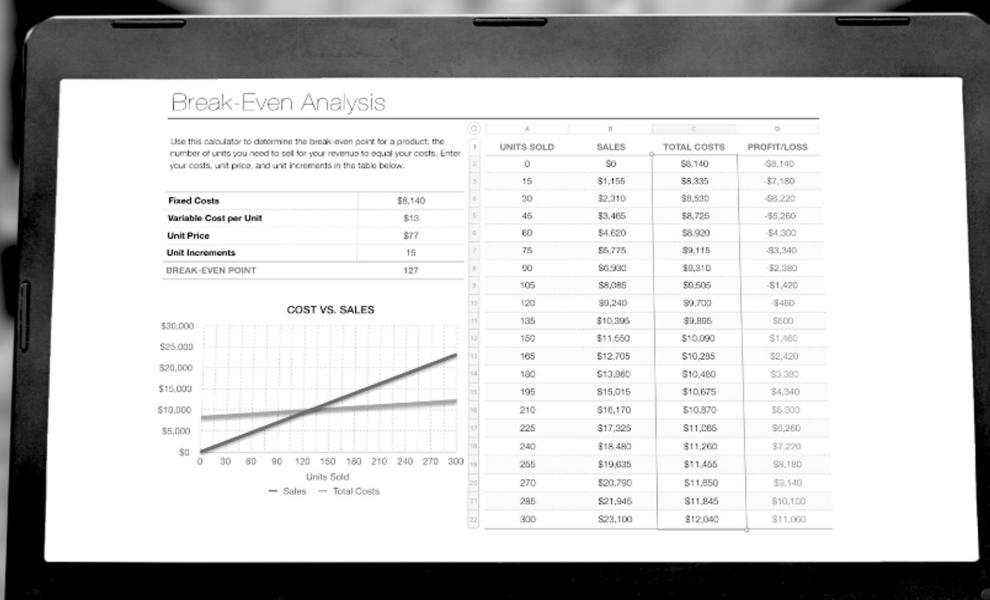
PERFORMANCE IMPROVEMENTS

- **Auto-Suspend:** By default, Snowflake will auto-suspend a virtual warehouse (the compute resources with the SSD cache after 10 minutes of idle time.
- **Scale up for large data volumes:** If you have a sequence of large queries to perform against massive (multi-terabyte) size data volumes, you can improve query performance by scaling up.
- **Scale down - but not too soon:** Once your large task has completed, you could reduce costs by scaling down or even suspending the virtual warehouse.
- **Don't thrash** - don't keep adjusting the warehouse sizing, or you'll lose the benefits of the automatic scaling and caching features.

Cost Optimization

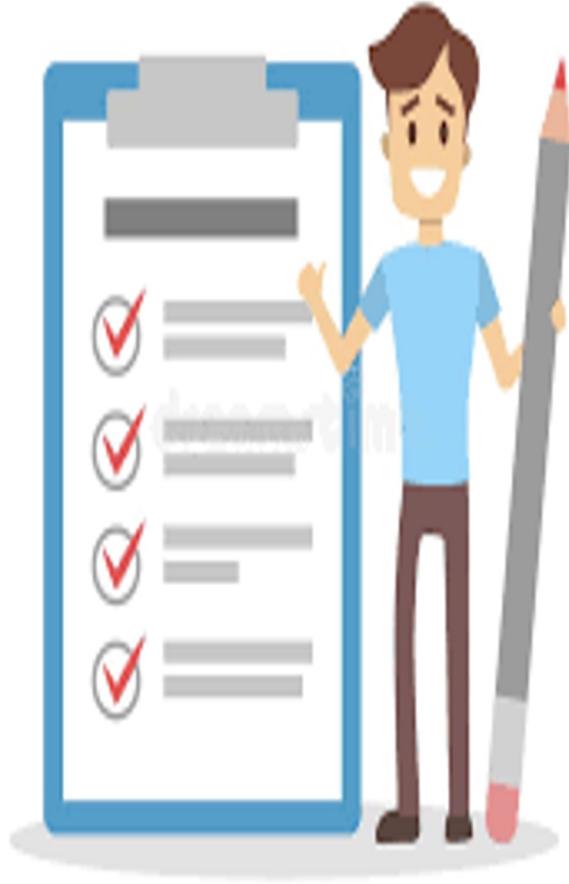
Kickstart

19



What We have Covered so far

- Binary
- Semi Structured
- JSON Basics
- MultiTable RDBMS
- Common Table Expressions (CTE)
- Connect By
- Dynamic Query Optimisation
- Caching



SNOWFLAKE COSTS

Expenses to account for with Snowflake include:

- . Warehouses
- . Snowpipe
- . Materialized Views
- . Cloud Services
- . Data Transfers
- . Storage

SNOWFLAKE RESOURCE TAGGING

While it is possible to minimize or prevent expenses in some areas, you should still plan for all potential expense sources. And whatever model you choose, you will need to track these expenses within Snowflake to determine how much consumption has occurred and how to charge it to the right budget.



Simplest method is to create a naming convention for database objects that allows you to identify the owner and associated budget.

CHARGE BACK MODEL

Unlike traditionally licensed on-premises data solutions, Snowflake operates with a flexible pay-as-you-go model, allowing you to create an account and start using it without delay.

However, without the proper planning to ensure governance and visibility around utilization, this model can also make it easy to run up a significant bill as multiple business units ask for access.



CHARGE DETERMINATION

Unlike traditionally licensed on-premises data solutions, Snowflake operates with a flexible pay-as-you-go model, allowing you to create an account and start using it without delay.

However, without the proper planning to ensure governance and visibility around utilization, this model can also make it easy to run up a significant bill as multiple business units ask for access.



WAREHOUSE SIZING

The “t-shirt size” of your warehouse (XS, S, M, L, XL) will significantly impact your credit usage.

Sizing your warehouses to perform well and remain cost-efficient is always a challenge, and it’s a bit more art than science. But while you may need some trial and error, there are some rules of thumb for picking your starting point.

Begin with a size that roughly correlates to how much data the warehouse in question will be processing for a given query:

XS: Multiple megabytes

S: A single gigabyte

M: Tens of gigabytes

L: Hundreds of gigabytes

XL: Terabytes

Clients and Ecosystem

All brains in the game



10

CLIENTS

When developing an application that connects to Snowflake, you can use the [native programmatic interfaces](#) (connectors, drivers, and client APIs) provided by Snowflake. These interfaces support many popular programming languages and development platforms.

- [Go Snowflake Driver](#)
- [JDBC Driver](#)
- [.NET Driver](#)
- [Node.js Driver](#)
- [ODBC Driver](#)
- [PHP PDO Driver for Snowflake](#)
- [SQL API](#)

CONNECTORS

[Snowflake Connector for Kafka](#) - The Snowflake Connector for Kafka (“Kafka connector”) reads data from one or more [Apache Kafka](#) topics and loads the data into a Snowflake table.

[Snowflake Connector for Python](#) - The Snowflake Connector for Python provides an interface for developing Python applications that can connect to Snowflake and perform all standard operations. It provides a programming alternative to developing applications in Java or C/C++ using the Snowflake JDBC or ODBC drivers.

[Snowflake Connector for Spark](#) - The Snowflake Connector for Spark (“Spark connector”) brings Snowflake into the Apache Spark ecosystem, enabling Spark to read data from, and write data to, Snowflake. From Spark’s perspective, Snowflake looks similar to other Spark data sources (PostgreSQL, HDFS, S3, etc.).

KAFKA CONNECTOR

Kafka Connect is a framework for connecting Kafka with external systems, including databases. A Kafka Connect cluster is a separate cluster from the Kafka cluster. The Kafka Connect cluster supports running and scaling out connectors (components that support reading and/or writing between external systems).

The Kafka connector is designed to run in a Kafka Connect cluster to read data from Kafka topics and write the data into Snowflake tables.

Snowflake provides two versions of the connector:

- Confluent package version of Kafka
- Open-source software (OSS) Apache Kafka package

PYTHON CONNECTOR

The Snowflake Connector for Python provides an interface for developing Python applications that can connect to Snowflake and perform all standard operations. It provides a programming alternative to developing applications in Java or C/C++ using the Snowflake JDBC or ODBC drivers.

The connector is a native, pure Python package. It can be installed using pip on Linux, macOS, and Windows platforms where **Python 3.6, 3.7, 3.8, or 3.9** is installed.

The connector supports developing applications using the **Python Database API v2 specification (PEP-249)**

SPARK CONNECTOR

The Snowflake Connector for Spark (“Spark connector”) brings Snowflake into the Apache Spark ecosystem, enabling Spark to **read data from, and write data to**, Snowflake. From Spark’s perspective, Snowflake looks similar to other Spark data sources (PostgreSQL, HDFS, S3, etc.).

Snowflake supports three versions of Spark: **Spark 2.4, Spark 3.0, and Spark 3.1**. There is a separate version of the Snowflake Connector for Spark for each version of Spark. Use the correct version of the connector for your version of Spark.

The connector runs as a **Spark plugin** and is provided as a Spark package (**spark-snowflake**).

SNOWSQL

SnowSQL is the next-generation command line client for connecting to Snowflake to execute SQL queries and perform all DDL and DML operations, including loading data into and unloading data out of database tables.

SnowSQL (snowsql executable) can be run as an interactive shell or in batch mode through stdin or using the -f option.

SnowSQL is an example of an application developed using the Snowflake Connector for Python; however, the connector is not a prerequisite for installing SnowSQL. All required software for installing SnowSQL is bundled in the installers./

SNOWFLAKE PARTNER ECOSYSTEM

The Snowflake Partner Network unlocks the potential of the Data Cloud with a broad array of tools and partners.

Certified partnerships and integrations enable customers to leverage Snowflake's flexibility, performance, and ease of use to deliver more meaningful data insights.

As a customer, whether you are looking for certified services partners to help you migrate or make the most of your Snowflake deployment, or whether you are looking for integrated technologies, the Snowflake Partner Network is a great place to start.

SNOWFLAKE CLOUD PARTNERS

Snowflake is available on **AWS**,
Azure, and **GCP** in countries
across North America, Europe,
Asia Pacific, and Japan.



Thanks to a global approach to
cloud computing, customers can
get a **single and seamless**
experience with deep integrations
with our cloud partners and their
respective regions.



SNOWFLAKE DATA PROVIDERS

As a Snowflake customer, easily and securely access data from hundreds of data providers that comprise the ecosystem of the Data Cloud.

Customers can engage data service providers to complete your data strategy and obtain the deepest, data-driven insights possible.



SNOWFLAKE SERVICES PARTNERS

Snowflake Services Partners provide customers with trusted and validated experts and services around implementation, migration, data architecture and data pipeline design, BI integration, ETL/ELT integration, performance, running POCs, performance optimization, and training.



SNOWFLAKE TECHNOLOGY PARTNERS

Snowflake Technology Partners integrate their solutions with Snowflake, so our customers can easily get data into Snowflake and insights out Snowflake by creating a single copy of data for their cloud data analytics strategy.



POWERED BY SNOWFLAKE

Powered by Snowflake program is designed to help software companies and application developers build, operate, and grow their applications on Snowflake.

The program offers technical advice, access to support engineers who specialize in app development, and joint go-to-market (GTM) opportunities.



Experiment



Content

#11-
WorkingWithSno
wSQL.pdf

Scripts

N/A

Data

data-load-
internal.zip

Experiment



Content
#12-
WorkingWithSnowflakePythonConnector.pdf

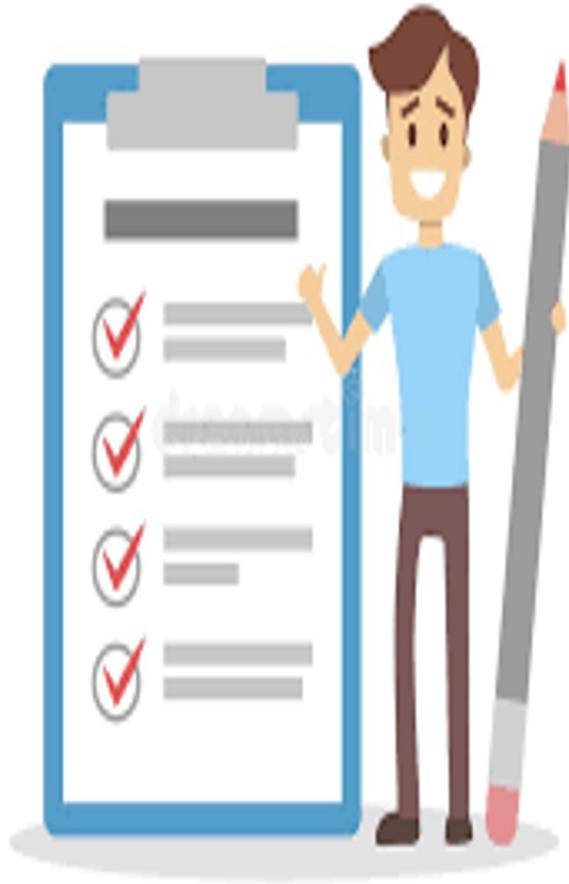
Scripts
PythonConnector
Sample.py

Security



What We have Covered so far

- Snowflake Costs
- Chargeback Model
- Clients
- Kafka Connector
- Spark Connector
- Snowflake Partners



CONTINUOUS DATA PROTECTION

Continuous Data Protection (CDP) encompasses a comprehensive set of features that help protect data stored in Snowflake against human error, malicious acts, and software or hardware failure.

At every stage within the data lifecycle, Snowflake enables your data to be accessible and recoverable in the event of accidental or intentional modification, removal, or corruption.

NETWORK POLICIES

Network policies provide options for managing network configurations to the Snowflake service.

Currently, network policies allow restricting access to your account based on user IP address. Effectively, a network policy enables you to create an IP allowed list, as well as an IP blocked list, if desired.

Account-level network policy management can be performed through the new web interface, the classic web interface, or SQL.

MULTI-FACTOR AUTHENTICATION (MFA)

Snowflake supports multi-factor authentication (i.e. MFA) to provide increased login security for users connecting to Snowflake. MFA support is provided as an integrated Snowflake feature, powered by the Duo Securityservice, which is managed completely by Snowflake.

Users do not need to separately sign up with Duo or perform any tasks, other than installing the Duo Mobile application, which is supported on multiple smart phone platforms (iOS, Android, Windows, etc.).

MFA is enabled on a per-user basis; however, at this time, users are not automatically enrolled in MFA. To use MFA, users must enroll themselves.

MFA BEST PRACTICE

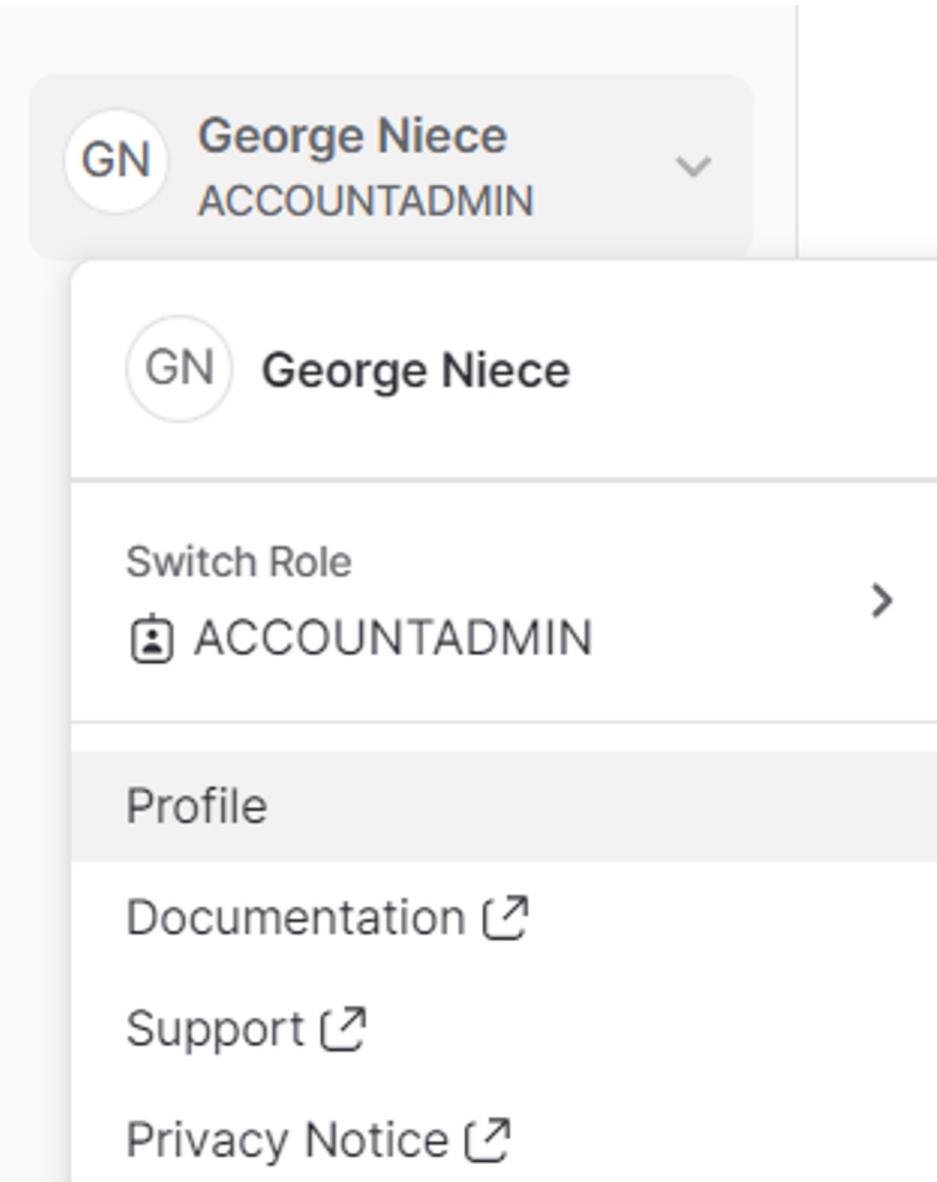
To ensure the highest level of security for your Snowflake account, we strongly recommend that any user who can modify or view sensitive data be required to use multi-factor authentication (MFA) for login.

This recommendation applies particularly to users with the ACCOUNTADMIN role, but can also be expanded to include users with the SECURITYADMIN and SYSADMIN roles.

SETUP MFA

MFA setup is per-user and from the Snowflake console.

Select the menu from your username and choose **Preferences**



SETUP MFA

Choose
Enroll option
under Multi-
factor
authentication

Profile

Username	SNOWFLAKECOURSEUPDATE
First Name	George
Last Name	Niece
Password 
Email	snowflake.course.update@gmail.com
Default role & warehouse	 ACCOUNTADMIN • COMPUTE_WH
Language	 English 
Notifications	<input type="checkbox"/> Notify when queries finish in the background
Multi-factor authentication	
Each time you sign in to Snowflake, you'll use your password and a verification code. Learn more	
Enroll	

SETUP MFA

Multi-factor authentication

 Protect Your Snowflake (AWS US East2) Account

[What is this? !\[\]\(f83ba58d2ce80f1436476049b08891eb_img.jpg\)](#)
[Need help?](#)

Secured by Duo

Two-factor authentication enhances the security of your account by using a secondary device to verify your identity. This prevents anyone but you from accessing your account, even if they know your password.

This process will help you set up your account with this added layer of security.

[Start setup](#)

MFA DEVICES

Setup Duo

X



[What is this? ↗](#)

[Need help?](#)

Secured by Duo

What type of device are you adding?

Mobile phone RECOMMENDED

Tablet (iPad, Nexus 7, etc.)

Landline

Continue

FEDERATED AUTHENTICATION & SSO

Federated authentication enables your users to connect to Snowflake using secure SSO (single sign-on). With SSO enabled, your users authenticate through an external, SAML 2.0-compliant identity provider (IdP).

For example, in the Snowflake web interface, a user connects by clicking the IdP option on the login page:

If they have already been authenticated by the IdP, they are immediately granted access to Snowflake.

If they have not yet been authenticated by the IdP, they are taken to the IdP interface where they authenticate, after which they are granted access to Snowflake.

FEDERATED AUTHENTICATION & SSO

Federated authentication enables your users to connect to Snowflake using secure SSO (single sign-on). With SSO enabled, your users authenticate through an external, SAML 2.0-compliant identity provider (IdP).

For example, in the Snowflake web interface, a user connects by clicking the IdP option on the login page:

If they have already been authenticated by the IdP, they are immediately granted access to Snowflake.

If they have not yet been authenticated by the IdP, they are taken to the IdP interface where they authenticate, after which they are granted access to Snowflake.

ACCESS CONTROL IN SNOWFLAKE

Snowflake provides granular control over access to objects — who can access what objects, what operations can be performed on those objects, and who can create or alter access control policies.

SNOWFLAKE SUPER USER

The account administrator (i.e users with the ACCOUNTADMIN system role) role is the most powerful role in the system.

This role alone is responsible for configuring parameters at the account level.

Users with the ACCOUNTADMIN role can view and operate on all objects in the account, can view and manage Snowflake billing and credit data, and can stop any running SQL statements.

In the default access control hierarchy, the other administrator roles are children of ACCOUNTADMIN.

ACCOUNTADMIN CHILD ROLES

The user administrator (USERADMIN) role includes the privileges to create and manage users and roles (assuming ownership of those roles or users has not been transferred to another role).

The security administrator (i.e users with the SECURITYADMIN system role) role includes the global MANAGE GRANTS privilege to grant or revoke privileges on objects in the account. The USERADMIN role is a child of this role in the default access control hierarchy.

The system administrator (SYSADMIN) role includes the privileges to create warehouses, databases, and all database objects (schemas, tables, etc.).

ACCESS CONTROL PRIVILEGES

- [Global Privileges](#)
- [User Privileges](#)
- [Role Privileges](#)
- [Resource Monitor Privileges](#)
- [Virtual Warehouse Privileges](#)
- [Connection Privileges](#)
- [Integration Privileges](#)
- [Network Policy Privileges](#)
- [Data Exchange Privileges](#)
- [Data Exchange Listing Privileges](#)
- [Database Privileges](#)
- [Schema Privileges](#)
- [Table Privileges](#)
- [External Table Privileges](#)
- [View Privileges](#)
- [Stage Privileges](#)
- [File Format Privileges](#)
- [Pipe Privileges](#)
- [Stream Privileges](#)
- [Task Privileges](#)
- [Masking Policy Privileges](#)
- [Row Access Policy Privileges](#)
- [Tag Privileges](#)
- [Sequence Privileges](#)
- [Stored Procedure Privileges](#)
- [User-Defined Function \(UDF\) and External Function Privileges](#)

DATA ENCRYPTION

Protecting customer data is one of Snowflake's highest priorities. Snowflake encrypts all customer data by default, using the latest security standards, at no additional cost.

Snowflake provides best-in-class key management, which is entirely transparent to customers.

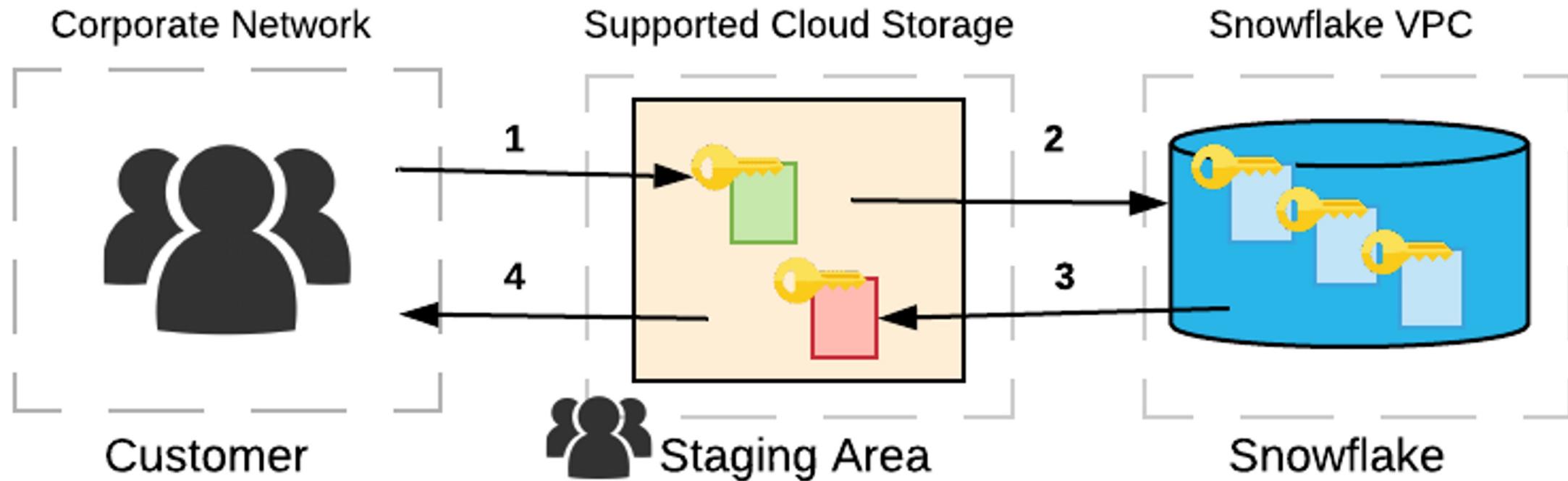
This makes Snowflake one of the easiest-to-use and most secure data platforms available.

END 2 END ENCRYPTION (E2EE)

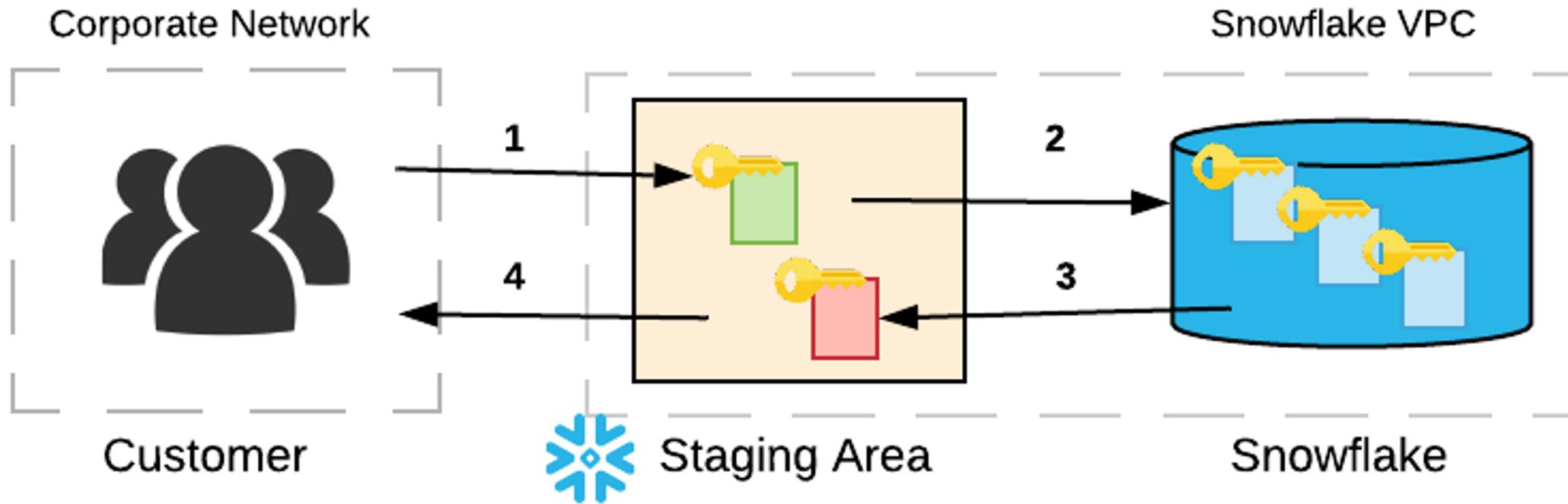
End-to-end encryption (E2EE) is a form of communication in which no one but end users can read the data. In Snowflake, this means that only a customer and the runtime components can read the data. No third parties, including Snowflake's cloud computing platform or any ISP, can see data in the clear.

E2EE minimizes the attack surface. In the event of a security breach of the cloud platform, the data is protected because it is always encrypted, regardless of whether the breach exposes access credentials indirectly or data files directly, whether by an internal or external attacker.

CUSTOMER PROVIDED STAGING



SNOWFLAKE PROVIDED STAGING



STAGING PROCESSING

Snowflake supports both internal and external stages for data files. Snowflake provides internal stages where you can upload and group your data files before loading the data into tables (**Snowflake-provided**).

Customer-provided stages are containers or directories in a supported cloud storage platform (i.e. Amazon S3) that you own and manage (**Customer-provided**).

Customer-provided stages are an attractive option for customers that already have data stored on these platforms, which they want to copy into Snowflake. Snowflake supports E2EE for both types of stage.

E2EE FLOW

1. A user uploads one or more data files to a stage. If the stage is a customer-managed container in a cloud storage service (Customer-provided), the user may optionally encrypt the data files using client-side encryption.
2. If the stage is an internal (i.e. Snowflake) stage (Snowflake-provided), data files are automatically encrypted by the client on the local machine prior to being transmitted to the internal stage.
3. The user loads the data from the stage into a table. The data is transformed into Snowflake's proprietary file format and stored in a cloud storage container ("data at rest").
4. Query results can be unloaded into a stage. Results are optionally encrypted using client-side encryption when unloaded into a customer-provided stage, and are automatically encrypted when unloaded to a Snowflake-provided stage.
5. The user downloads data files from the stage and decrypts the data on the client side.

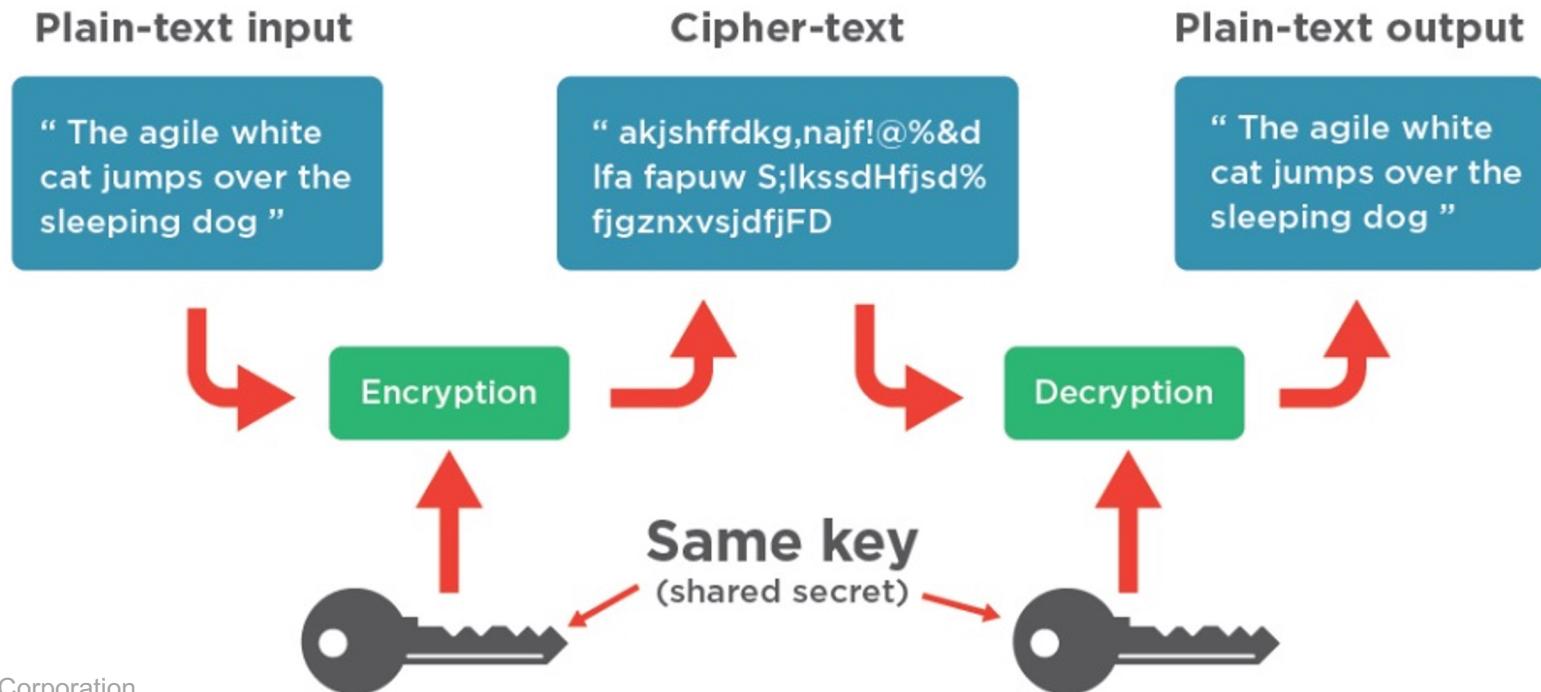
CLIENT-SIDE ENCRYPTION FLOW

- 1.Client-side encryption follows a specific protocol defined by the cloud storage service. The service SDK and third-party tools implement this protocol.
- 2.Customer creates a secret master key, which remains with the customer.
- 3.The client (provided by the cloud storage service) generates a random encryption key and encrypts the file before uploading it into cloud storage. The random encryption key, in turn, is encrypted with the customer's master key.
- 4.Both the encrypted file and the encrypted random key are uploaded to the cloud storage service. The encrypted random key is stored with the file's metadata.
- 5.When downloading data, the client downloads both the encrypted file and the encrypted random key. The client decrypts the encrypted random key using the customer's master key. Next, the client decrypts the encrypted file using the now decrypted random key. All encryption and decryption happens on the client side.

SNOWFLAKE ENCRYPTION

All ingested data stored in Snowflake tables is encrypted using AES-256 strong encryption.

All files stored in internal stages for data loading and unloading operations is automatically encrypted using AES-256 strong encryption.

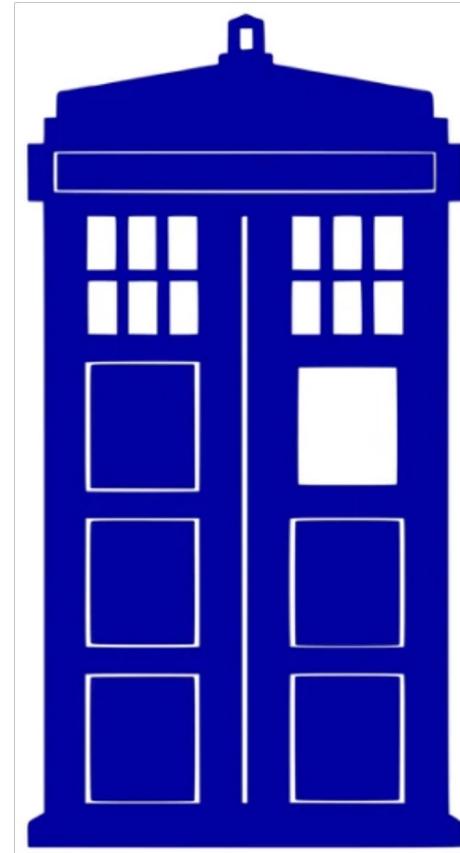


TIME TRAVEL

Snowflake provides powerful CDP features for ensuring the maintenance and availability of your historical data (i.e. data that has been changed or deleted):

Querying, cloning, and restoring historical data in tables, schemas, and databases for up to 90 days through Snowflake Time Travel.

These features are included standard for all accounts, i.e. no additional licensing is required; however, standard Time Travel is 1 day. Extended Time Travel (up to 90 days) requires Snowflake Enterprise Edition.

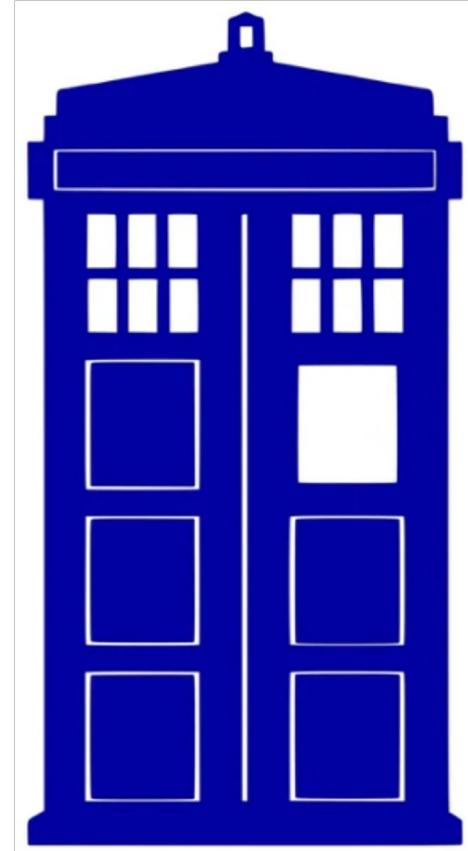


TIME TRAVEL IN ACTION

You accidentally DROP a table in production

In other RDBMS and data warehouse systems you start to work on a restore.

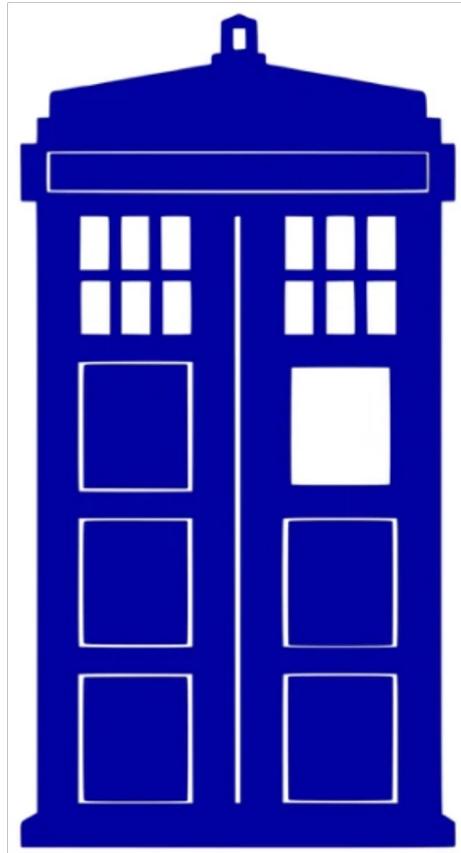
In Snowflake we turn to the UNDROP command



SNOWFLAKE UNDROP

UNDROP TABLE *TABLE_NAME*

- Restoring tables is only supported in the current schema or current database, even if the table name is fully-qualified.
- If a table with the same name already exists, an error is returned.



TIME TRAVEL AT | BEFORE

The AT or BEFORE clause is used for Snowflake Time Travel. In a query, it is specified in the FROM clause immediately after the table name and it determines the point in the past from which historical data is requested for the object:

The AT keyword specifies that the request is inclusive of any changes made by a statement or transaction with timestamp equal to the specified parameter.

The BEFORE keyword specifies that the request refers to a point immediately preceding the specified parameter.

TIME TRAVEL AT | BEFORE

```
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));
```

TIME TRAVEL LIMITATIONS

Databases, schemas, and tables can be recovered by Time Travel.

With tables, Time Travel support depends on the type of table:

- Only **permanent** can be recovered longer than 1 day ago.
- **Temporary** and **transient** tables can leverage Time Travel for up to one day.
- **External** tables can't use Time Travel, as they are not in Snowflake.

FAIL-SAFE

Separate and distinct from Time Travel, Fail-safe ensures historical data is protected in the event of a system failure or other catastrophic event, e.g. a hardware failure or security breach.

Disaster recovery of historical data (by Snowflake) through Snowflake Fail-safe.

Both Time Travel and Fail-safe require additional data storage, which has associated fees.



FAIL-SAFE DETAILS

Fail-safe is **not** provided as a means for accessing historical data after the Time Travel retention period has ended.

It is for use **only** by Snowflake to recover data that may have been lost or damaged due to extreme operational failures.

Data recovery through Fail-safe may take from several hours to several days to complete.



FAIL-SAFE LIMITATIONS

Databases, schemas, and tables can be recovered by Fail-Safe.

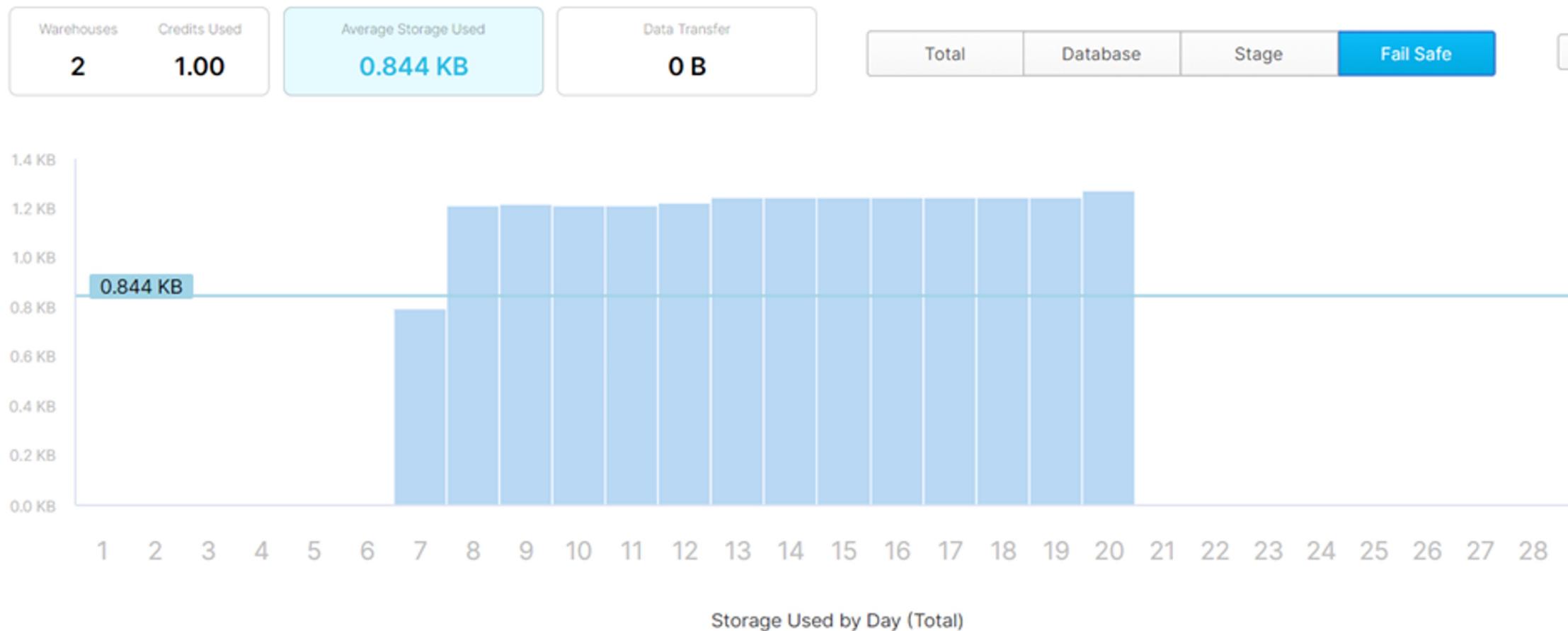
When it comes to tables in Snowflake, we have the following rules:

- Permanent tables can leverage Fail-Safe.
- Temporary and transient tables are not able to setup Fail-Safe.
- External tables can't use Fail-Safe as the data is not stored in Snowflake.

ACCOUNT – USAGE - FAIL-SAFE

LBA64704 - Usage

November 20



FAIL-SAFE IN ACTION

At high level, Fail-safe provides a (non-configurable) 7-day period during which historical data is recoverable by Snowflake support team.

You will be billed for storage for & days for your databases and tables in Fail-Safe.



CONTINUOUS DATA PROTECTION LIFECYCLE

Standard operations allowed:
Queries, DDL, DML, etc.

Time Travel allowed:
SELECT ... AT | BEFORE ...
CLONE ... AT | BEFORE ...
UNDROP ...

No user operations allowed
(data recoverable only by
Snowflake)

Current Data Storage

Time Travel Retention
(1-90 Days)

Fail-Safe
(transient: 0 days,
Permanent: 7 days)

Experiment



Content
#07-Snowflake-
Time-Travel.pdf

Scripts
foundation_experiment_scripts.sql

CLONING

Cloning also referred to as “**zero-copy cloning**” creates a copy of a database, schema or table. A snapshot of data present in the source object is taken when the clone is created and is made available to the cloned object.

The cloned object is **writable** and is **independent** of the clone source. That is, changes made to either the source object or the clone object are not part of the other.

Cloning a database will clone all the schemas and tables within that database. Cloning a schema will clone all the tables in that schema.

CLONED PRIVILEGES

A cloned object does not retain any granted privileges on the source object itself (i.e. clones do not automatically have the same privileges as their sources). A system administrator or the owner of the cloned object must explicitly grant any required privileges to the newly-created clone.

However, if the source object is a database or schema, for child objects contained in the source, the clone replicates all granted privileges on the corresponding child objects:

- For databases, contained objects include schemas, tables, views, etc.
- For schemas, contained objects include tables, views, etc.

CLONING AND STAGES

Individual external named stages can be cloned. An external stage references a bucket or container in external cloud storage; cloning an external stage has no impact on the referenced cloud storage.

Internal (i.e. Snowflake) named stages **cannot** be cloned.

When cloning a database or schema:

- External named stages that were present in the source when the cloning operation started are cloned.
- Tables are cloned, which means the internal stage associated with each table is also cloned. Any data files that were present in a table stage in the source database or schema are not copied to the clone (i.e. the cloned table stages are empty).
- Internal named stages are **not** cloned.

CLONING AND DDL CHANGES

Cloning is fast, but not instantaneous, particularly for large objects (e.g. tables). As such, if DDL statements are executed on source objects (e.g. renaming tables in a schema) while the cloning operation is in progress, the changes may not be represented in the clone. This is because DDL statements are atomic and not part of multi-statement transactions.

Furthermore, Snowflake does not record which object names were present when the cloning operation started, and which names changed. As such, DDL statements that rename (or drop and recreate) source child objects compete with any in-progress cloning operations and can cause name conflicts.

Experiment



Content
#08-Snowflake-RBAC.pdf

Scripts
foundation_experiment_scripts.sql

POP QUIZ:

Security



Follow the link for Security Quiz:

[https://forms.gle/zQd3ftwM5pjf
B7FMA](https://forms.gle/zQd3ftwM5pjfB7FMA)



Performance and Concurrency

impactful enterprise workloads

| 6

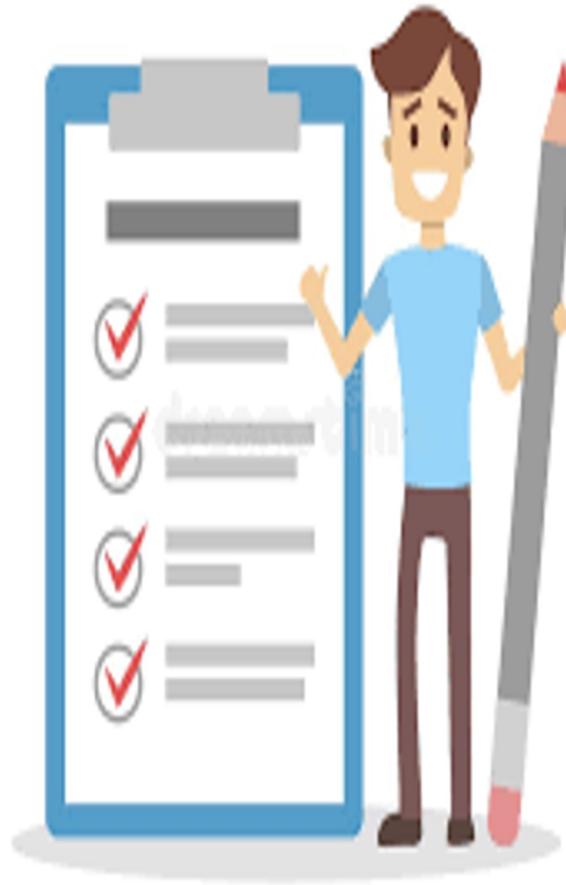


Develop a passion for learning.

290

What We have Covered so far

- Continuous Data Protection (CDP)
- Multi Factor Authentication (MFA)
- Federation Authentication and SSO
- Snowflake SuperUser
- Data Encryption
- SnowFlake Staging
- Client side Encryption
- Time Travel
- Fail-Safe
- Cloning

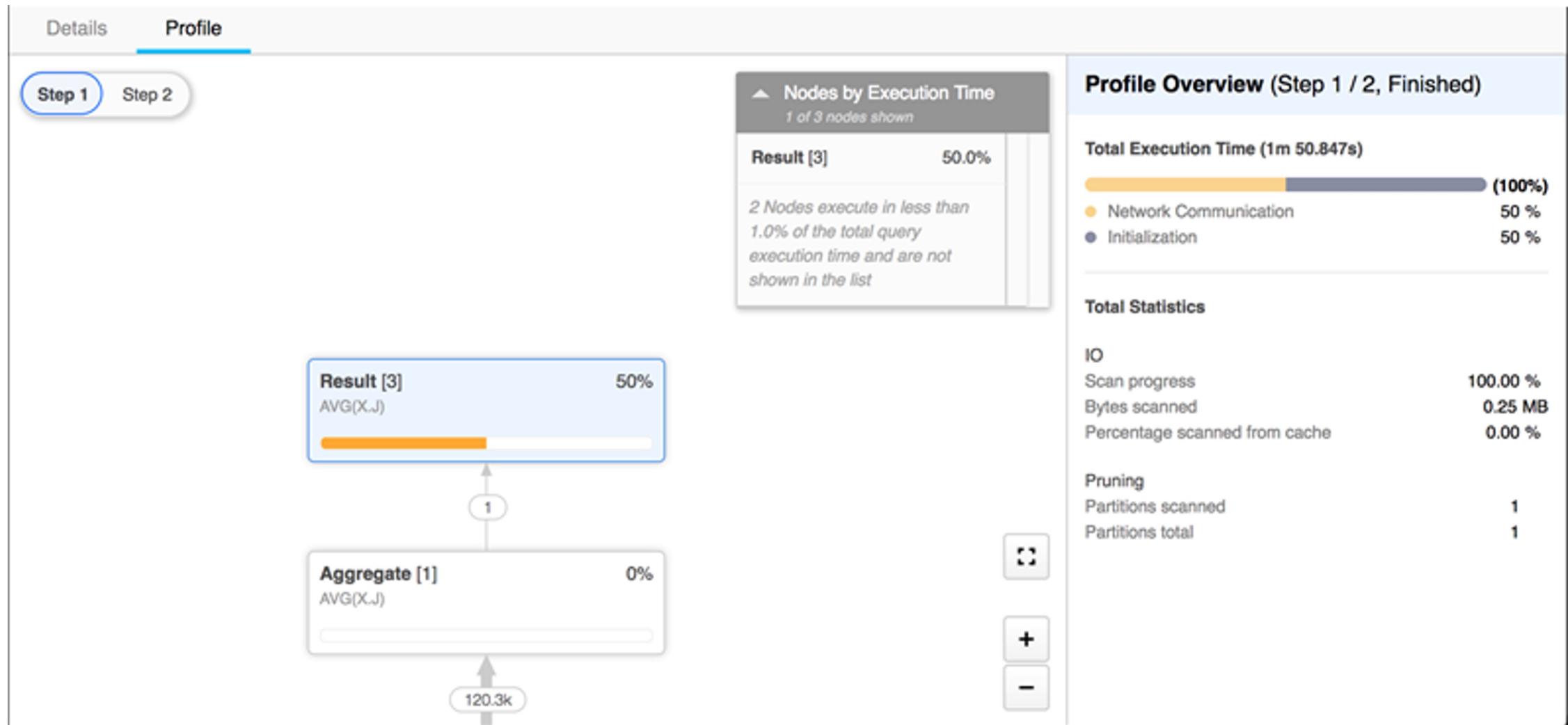


QUERY PROFILE

Query Profile, available through the Snowflake web interface, provides execution details for a query.

For the selected query, it provides a graphical representation of the main components of the processing plan for the query, with statistics for each component, along with details and statistics for the overall query.

QUERY PROFILE



MICRO-PARTITIONS

Traditional data warehouses rely on static partitioning of large tables to achieve acceptable performance and enable better scaling. In these systems, a *partition* is a unit of management that is manipulated independently using specialized DDL and syntax; however, static partitioning has a number of well-known limitations, such as maintenance overhead and data skew, which can result in disproportionately-sized partitions.

In contrast to a data warehouse, the Snowflake Data Platform implements a powerful and unique form of partitioning, called *micro-partitioning*, that delivers all the advantages of static partitioning without the known limitations, as well as providing additional significant benefits.

MICRO-PARTITION IN A NUTSHELL

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 (note that the actual size in Snowflake is smaller because data is always stored compressed). Groups of rows in tables are mapped into individual micro-partitions, organized in a columnar fashion.

Snowflake stores metadata about all rows stored in a micro-partition, including:

- The range of values for each of the columns in the micro-partition.
- The number of distinct values.
- Additional properties used for both optimization and efficient query processing.

MICRO-PARTITION 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 can overlap in their range of values, which, combined with their uniformly small size, helps prevent 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.

QUERY PRUNING

The micro-partition metadata maintained by Snowflake enables precise pruning of columns in micro-partitions at query run-time, including columns containing semi-structured data. In other words, a query that specifies a filter predicate on a range of values that accesses 10% of the values in the range should ideally only scan 10% of the micro-partitions.

For example, assume a large table contains one year of historical data with date and hour columns. Assuming uniform distribution of the data, a query targeting a particular hour would ideally scan the micro-partitions that contain the data for the hour column; Snowflake uses columnar scanning of partitions so that an entire partition is not scanned if a query only filters by one column.

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.

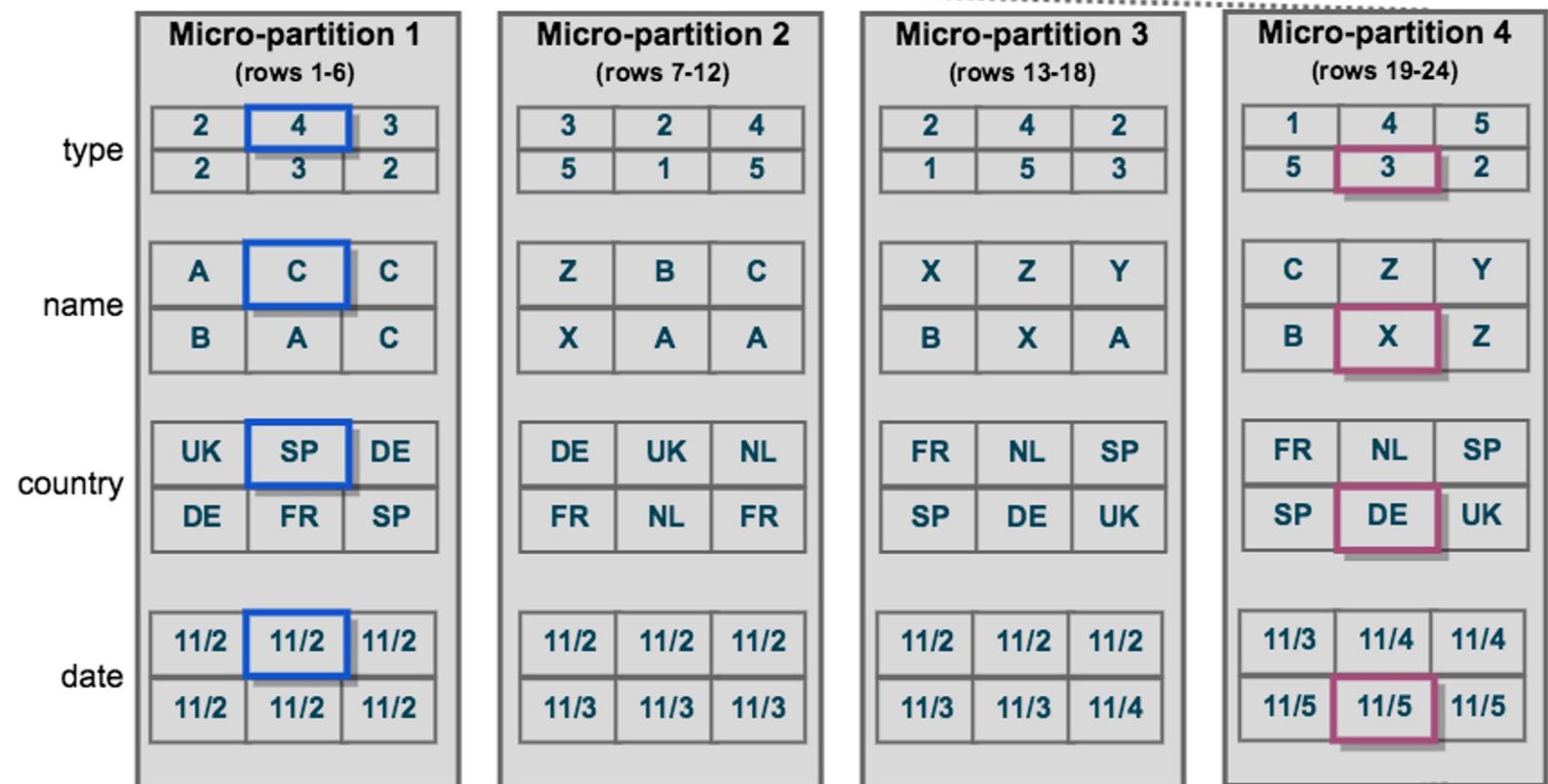
In Snowflake, as data is inserted-loaded into a table, clustering metadata is collected and recorded for each micro-partition created during the process. Snowflake then leverages this clustering information to avoid unnecessary scanning of micro-partitions during querying, significantly accelerating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

DATA CLUSTERING

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

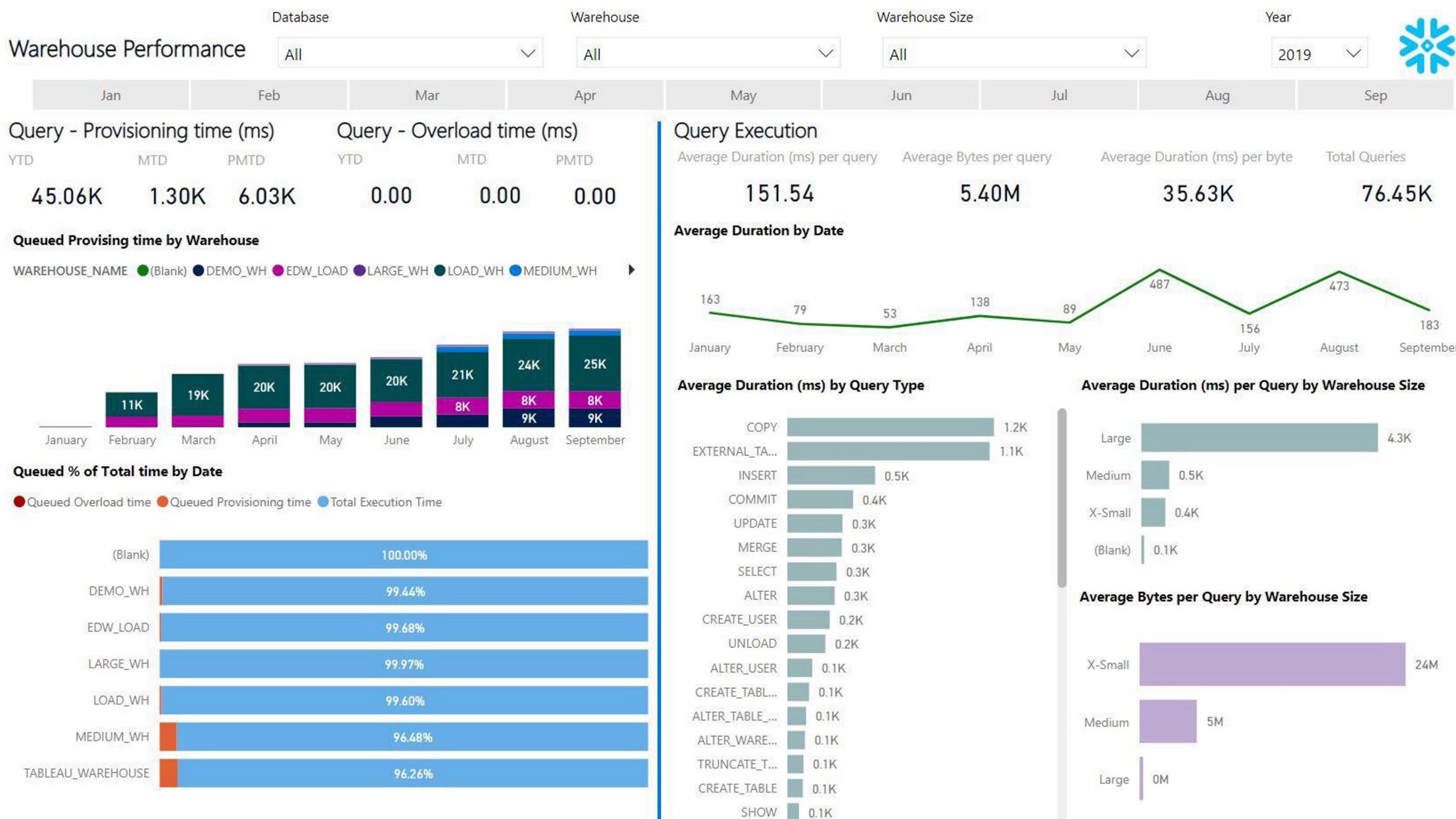


SCALING A VIRTUAL WAREHOUSE

Scaling up is all about increasing the compute power of the existing warehouse node. This should assist long-running queries, queries that require a lot of bytes scanned and queries with storage spillage. This would be done by increasing the size property of the warehouse.

Scaling Out is the process of adding more clusters to an existing warehouse. This will assist when there are a large number of concurrent queries being executed in the same warehouse. Scaling Out will allow for those queued queries to be executed on the new provisioned cluster.

WAREHOUSE PERFORMANCE REPORT



WAREHOUSE REPORT KEY METRICS

Queued Overload Time - Time (in milliseconds) spent in the warehouse queue, due to the warehouse being overloaded by the current query workload.

Check the Autoscaling Policy - Check the number of clusters is large enough for the number of concurrent queries

Queued Provisioning time - Time (in milliseconds) spent in the warehouse queue, waiting for the warehouse servers to provision, due to warehouse creation, resume, or resize.

Query Type - DML, query, etc. If the query is currently running, or the query failed, then the query type may be UNKNOWN. This will show the type of query that is running (e.g. Select, Copy Into, Alter).

Execution Time - Execution time (in milliseconds)

Bytes Scanned - Number of bytes scanned by this statement. The time taken and bytes scanned may indicate a larger warehouse is required.

Experiment



Content
#09-Snowflake-
Data-Sharing.pdf

Scripts
foundation_experiment_scripts.sql

Experiment



Content
#10-Snowflake-
Foundation-
Wrapup.pdf

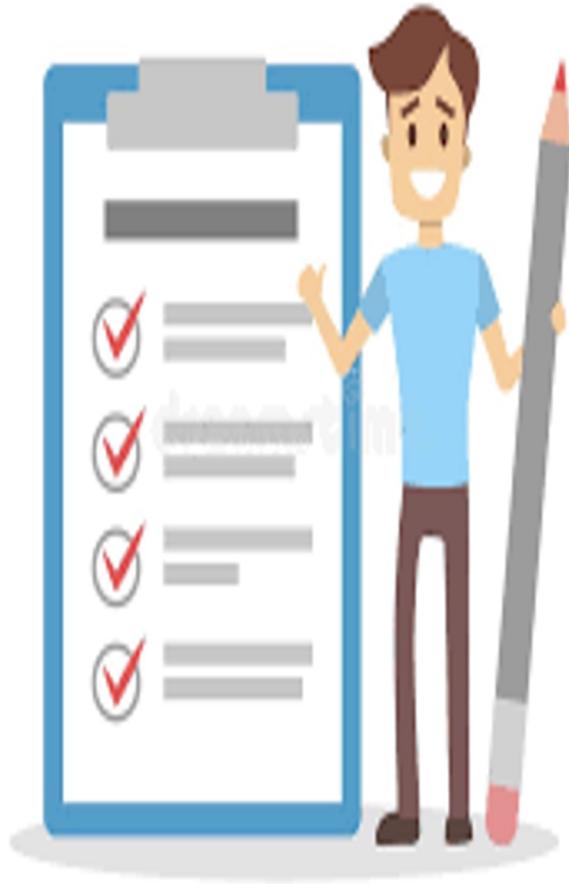
Scripts
foundation_experi
ment_scripts.sql

Account and Resources Management and Monitoring



What We have Covered so far

- Query Profile
- Micro-Partitions
- Data Clustering
- Virtual Performance



SYSTEM RESOURCE USAGE

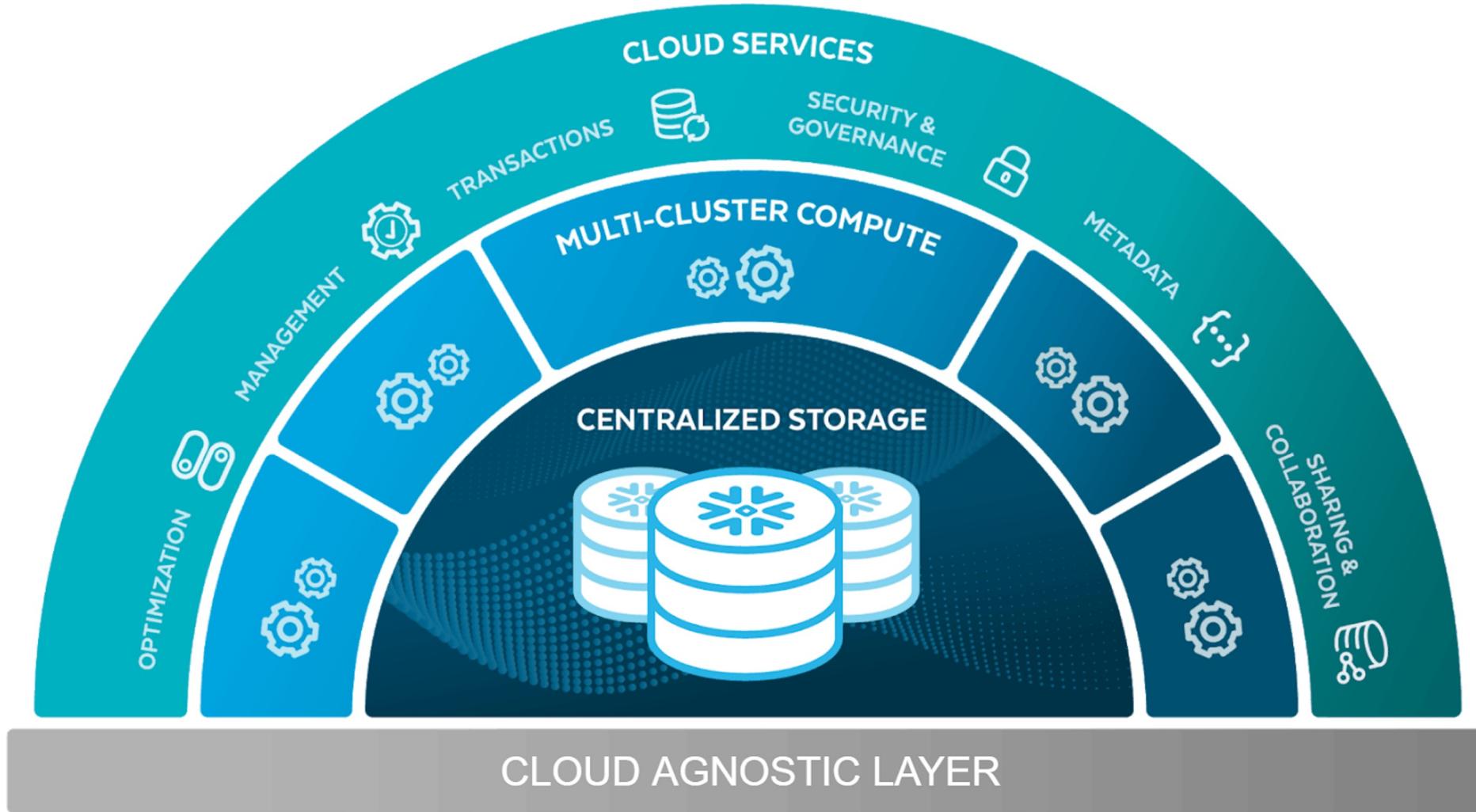
To help control costs and avoid unexpected credit usage caused by running warehouses, Snowflake provides *resource monitors*. A virtual warehouse consumes Snowflake credits while it runs.

Resource monitors can be used to impose limits on the number of credits that are consumed by:

- User-managed virtual warehouses
- Virtual warehouses used by cloud services

The number of credits consumed depends on the size of the warehouse and how long it runs.

MANAGING VIRTUAL WAREHOUSES



Google Cloud

AWS

Azure

MANAGING WAREHOUSE PERMISSIONS

Snowflake provides a set of permissions that can be assigned to roles to simplify warehouse management in a multi-warehouse environment:

USAGE: Grants the role the ability to use the warehouse and to run commands using the warehouse.

MONITOR: Grants the role the ability to see details within an object (for example, queries and usage within a warehouse). It is best practice to provide this ability to the role so the workload using the warehouse can monitor the activities in the warehouse and the role will be able to see the warehouse load over time using the Snowflake UI.

OPERATE: Grants the role the ability to start, stop, suspend, or resume a virtual warehouse. This is a sensitive permission. It is highly recommended to grant this permission to ONLY the warehouse administration role.

MANAGING VIRTUAL WAREHOUSES

Snowflake's unique architecture makes it possible to run multiple workloads concurrently, without performance impact, by using more than one concurrently running warehouse. Each workload, for example, an ETL workload and an analytics workload, can have its own isolated resources even while each is operating on the same databases and tables. That means the ETL workload does not impact the performance of the analytics workload and vice versa.

To get most of the value out of Snowflake virtual warehouses, it is important to use the correct warehouse for the work you intend to do, and it's also important to not mix multiple workloads in a single warehouse.

For example, a heavy batch operation workload should not use the warehouses designated for ad hoc queries and analytics.

WORKLOAD INDEPENDENCE

Ad-hoc Analytics: Self-service Analytics or ad-hoc heavy queries

Data Loading: A COPY command that constantly loads data from an external data source

Data transformations: A series of commands to transform your raw data into a more consumable format

Reporting: Dashboards and other reports refreshed on a schedule, or on-demand by executives

Applications: End-user applications that are displaying data based on query results

Batch: A massive batch transformation that runs frequently

WORKLOAD ANALYSIS

Frequency: How often the workload runs

Concurrency: How many processes will run simultaneously

Scan Size: What's the average size of data scan the query will perform

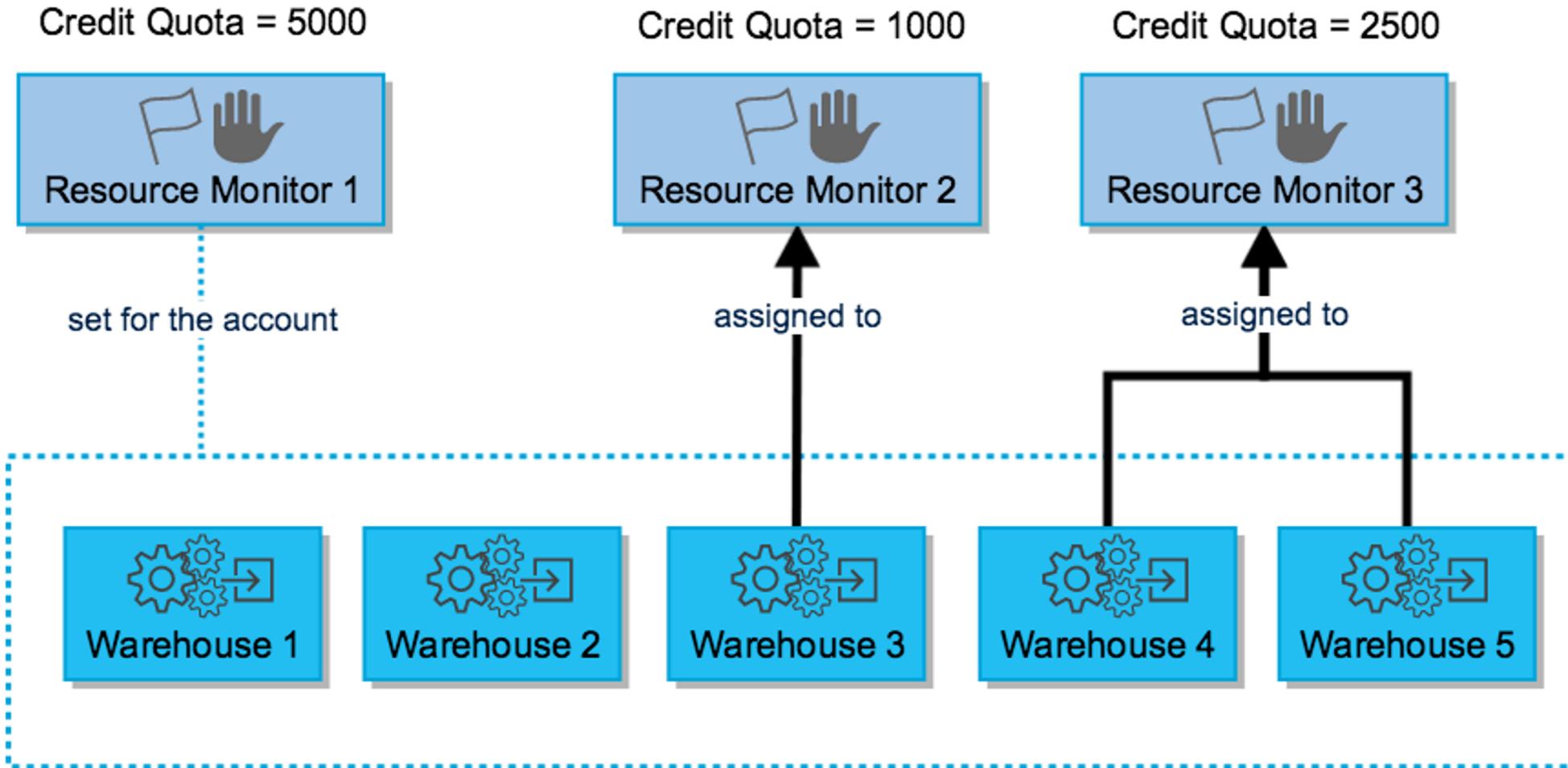
Copy Files: How many files will be copied simultaneously

SLA Minutes: The number of minutes stipulated by the SLA

RESOURCE SIZING

Workload Description	Workload Type	Frequency	Concurrency	Scan Size	Copy Files	SLA Minutes	Warehouse Name	Warehouse Size
Data Transformation	Batch	Hourly	1	1 TB		10	transform_wh	XL
Third-Party Data Ingestion	Loader	Hourly	2		2	5	load_wh	XS (see this tip)
Tableau	Ad-Hoc Analytics	Ad hoc	8	2 TB		1	analytics_wh	XL (see this tip)
Alerts, Monitoring...	Batch	Hourly	2	500 GB		1	app_wh	M
Console	Applications	On demand	16	1 TB		0.1	application_wh	XL
Snowflake UI	Ad-Hoc Analytics	Ad hoc	8	3 TB		1	analytics_wh	XL
Amazon S3 Loader	Loader	Hourly	5		5	5	load_wh	XS
Database, Machine Learning, or Python Connector	Batch	Hourly	2	5 TB		5	transform_wh	XL

RESOURCE MONITORS



RESOURCE MONITOR TRIGGERS

Triggers AKA Actions specify a threshold, as a percentage of the credit quota for the resource monitor, and the action to perform when the threshold is reached within the specified interval. Note that actions support thresholds greater than 100.

Resource monitors support the following actions:

Notify & Suspend - Send a notification (to all account administrators with notifications enabled) and suspend all assigned warehouses after all statements being executed by the warehouse(s) have completed.

Notify & Suspend Immediately - Send a notification (to all account administrators with notifications enabled) and suspend all assigned warehouses immediately.

Notify - Perform no action, but send an alert notification (to all account administrators with notifications enabled).

INFORMATION SCHEMA

The Snowflake Information Schema (aka “Data Dictionary”) consists of a set of system-defined views and table functions that provide extensive metadata information about the objects created in your account.

The Snowflake Information Schema is based on the SQL-92 ANSI Information Schema, but with the addition of views and functions that are specific to Snowflake.

INFORMATION SCHEMA USAGE

Each INFORMATION_SCHEMA schema is read-only (i.e. the schema, and all the views and table functions in the schema, cannot be modified or dropped).

Queries on INFORMATION_SCHEMA views do not guarantee consistency with respect to concurrent DDL. For example, if a set of tables are created while a long-running INFORMATION_SCHEMA query is being executed, the result of the query may include some, none, or all of the tables created.

The output of a view or table function depend on the privileges granted to the user's current role. When querying an INFORMATION_SCHEMA view or table function, only objects for which the current role has been granted access privileges are returned.

SNOWFLAKE DB - ACCOUNT USAGE

SNOWFLAKE is a system-defined, read-only shared database, provided by Snowflake.

The database is automatically imported into each account from a share named ACCOUNT_USAGE.

The SNOWFLAKE database is an example of Snowflake utilizing Secure Data Sharing to provide object metadata and other usage metrics for your account.

The SNOWFLAKE database contains two schemas (also read-only).

- ACCOUNT_USAGE
- READER_ACCOUNT_USAGE

ACCOUNT_USAGE

Views that display object metadata and usage metrics for your account.

In general, these views mirror the corresponding views and table functions in the Snowflake Information Schema, but with the following differences:

- Records for dropped objects included in each view.
- Longer retention time for historical usage data.
- Data latency.

READER_ACCOUNT_USAGE

Views that display object metadata and usage metrics for all the reader accounts that have been created for your account (as a Secure Data Sharing provider).

These views are a small subset of the ACCOUNT_USAGE views that apply to reader accounts, with the exception of the RESOURCE_MONITORS view, which is available only in READER_ACCOUNT_USAGE.

each view in this schema contains an additional READER_ACCOUNT_NAME column for filtering results by reader account.

Experiment



Content
#21-#25-
Snowflake-Data-
Pipeline (5 PDFs)

Scripts
iam-trust-
policy.json
s3-iam-role.json
ModernDataPipeli
nes_Experiment.s
ql (4 scripts)

Experiment

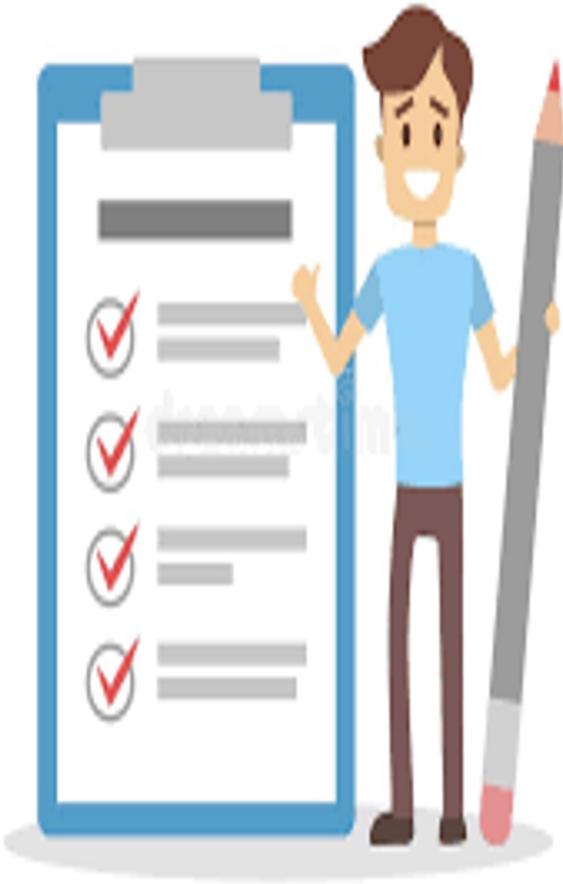


Content
#13-
WorkingWithSnowpipeAndSnowflakeNextGenUI.pdf

Scripts
Snowpipe_lab_scripts.sql

What We have Covered so far

- Managing Virtual Warehouses
- Workload Independence
- Resource Sizing
- Resource Monitor
- Information Schema
- Snowflake DB - Account Usage



Snowflake Practice Test



POP QUIZ:

SNOWFLAKE



Follow the link for Practice
Test 2 Quiz:

<https://forms.gle/9QBd5A5ebMBTkzBXA>



Snowflake Cert Resources



SNOWFLAKE CORE CERT GETTING READY

- ▶ Practice Test 1: <https://youtu.be/p0G-YSAUcZQ>
- ▶ Practice Test 2: <https://youtu.be/rSTcrXmSX5g>
- ▶ Complete SnowPro Guide: <https://bit.ly/35S7Rcb>
- ▶ SnowPro Practice Test (60 Questions): <https://bit.ly/2Ubernv>
- ▶ Revised Syllabus (with Stream & Task): <https://youtu.be/jhNoKeB0CVY>
- ▶ Stream/Task Jump Start: <https://bit.ly/3nFWD0u>
- ▶ Stream/Task Beginner's Guide: <https://bit.ly/3nQYeAq>