



Snowflake Fundamentals

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

INTRODUCTIONS

- Name
- Company & Role
- Experience with Snowflake
 - What do you already know?
 - What do you want to learn?
- Tell us something about yourself



COURSE OBJECTIVES

By the end of this course you will be able to set up and administer Snowflake to your users. You will also be able to effectively use snowflake to get business value out of your data. You will be able to:

- Configure and manage account parameters
- Configure users and roles with appropriate privileges
- Load and transform data in Snowflake
- Query data, and size virtual warehouses for different use cases
- Work with semi-structured data in Snowflake
- Understand Snowflake's micro-partitioning and how it contributes to query optimization
- Use Time Travel and failsafe storage to recover data and provide continuous data protection
- Use zero-copy cloning to provide groups with different use cases access to the same data
- Use Data Sharing to send your data in real time to customers, partners or other company accounts



COURSE AGENDA

Overview & Architecture

Clients, Connectors, & Ecosystems

SQL Support in Snowflake

Data Movement

Snowflake Functions

Managing Security

Access Control and User Management

Semi-Structured Data

Snowflake Caching

Continuous Data Protection

Data Sharing

Performance & Concurrency

Account and Resource Management

Preview Features



Introduction

Prepare for Class

10 minutes

- Log in to Snowflake University (MindTickle)
 - <https://snowflakeuniversity.mindtickle.com/login>
- Access student materials
 - Available for 90 days
 - Need access to workbook for labs
- Download lab files
 - Don't copy/paste SQL from the workbook



Module 1

Overview and Architecture

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



MODULE AGENDA

- The Snowflake Difference
- Snowflake Structure
- Cloud Services Layer
- Data Storage Layer
- Compute Layer



avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

The Snowflake Difference

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



SNOWFLAKE

A FULL DATA WAREHOUSE BUILT FOR THE CLOUD

Our vision

Allow our customers to access all their data in one place to make actionable decisions anytime, anywhere, with any number of users.



Our solution

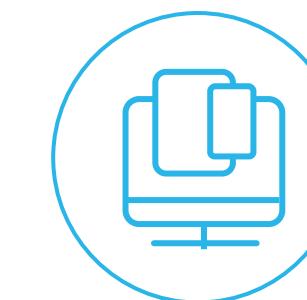
Next-generation data warehouse built from the ground up for the cloud to address today's data and analytics challenges.



**SQL Data
Warehouse**



**Built for
the cloud**



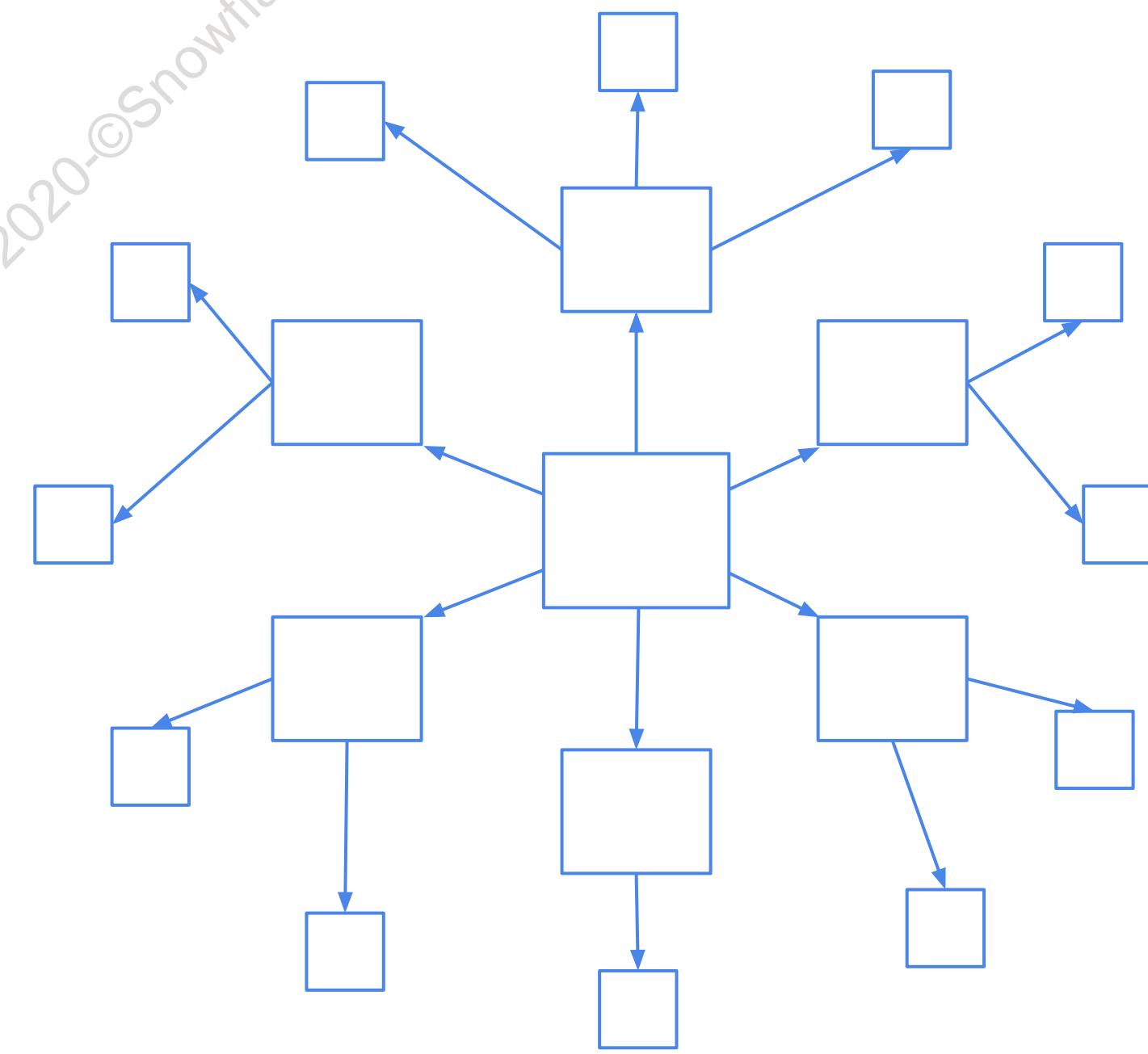
**Delivered as
a service**



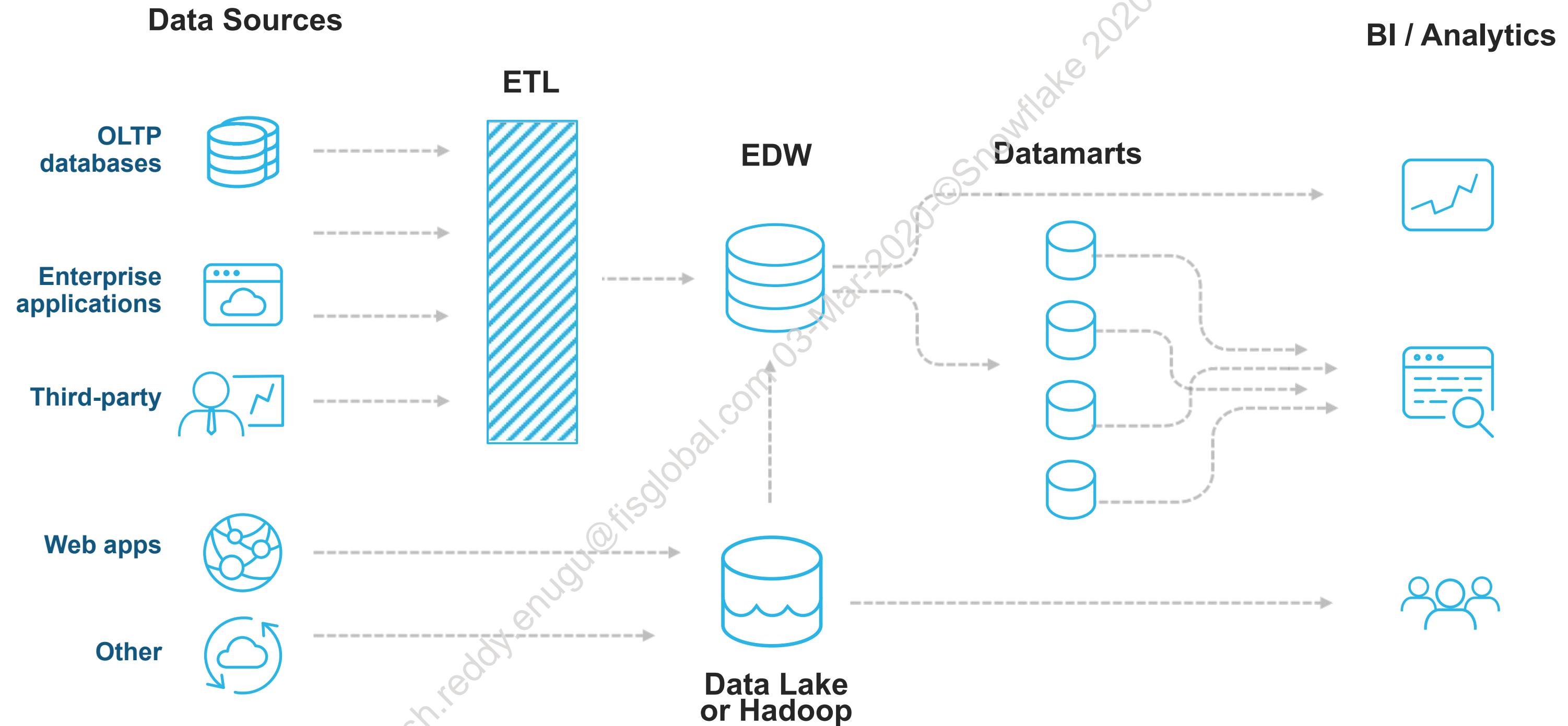
POP QUIZ

How did the Snowflake founders decide on the name Snowflake?

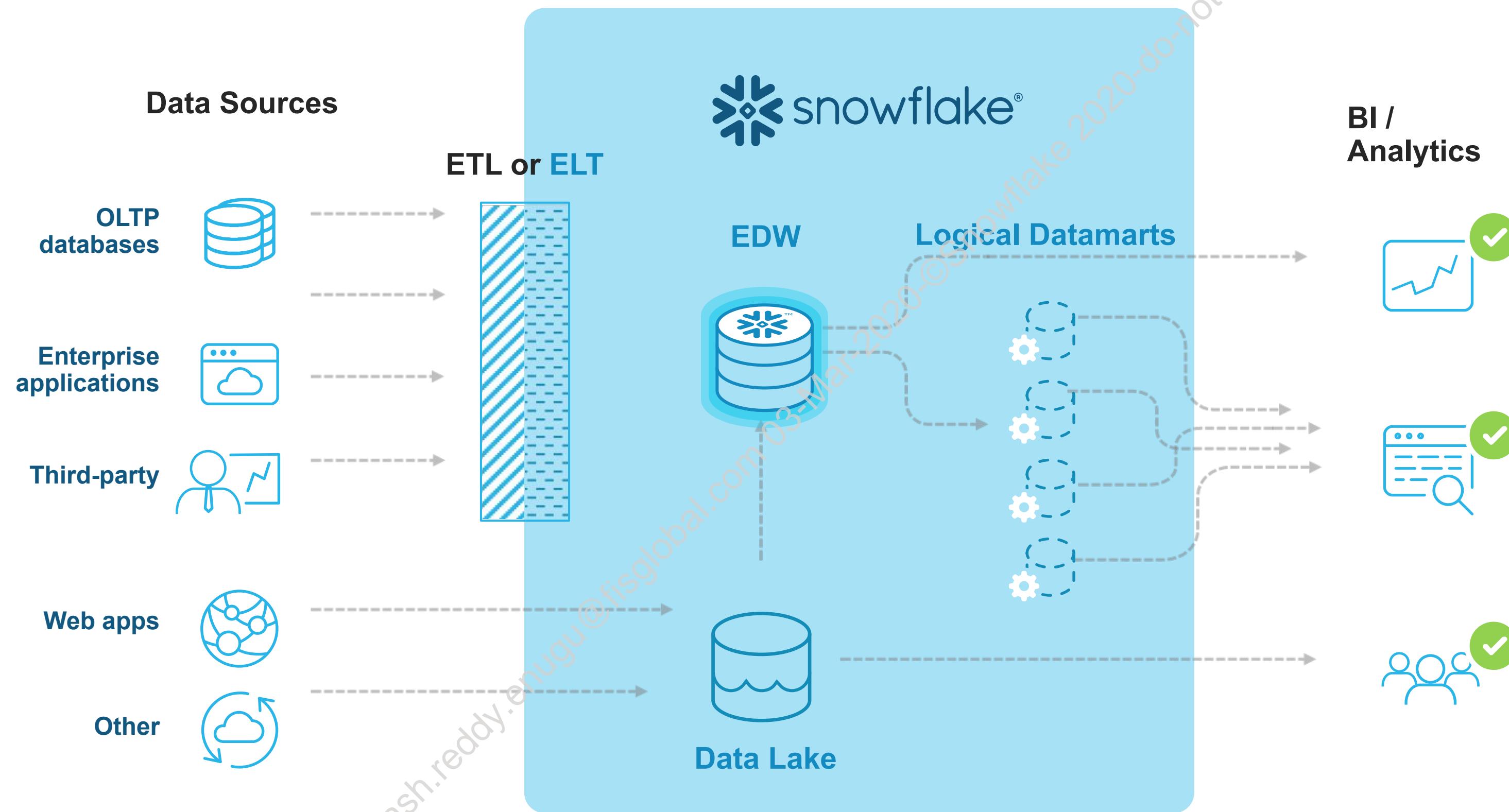
- They love the snow
- Snowflakes are born in the cloud
- Each snowflake is unique, like our architecture
- Data warehousing term (Snowflake schema)



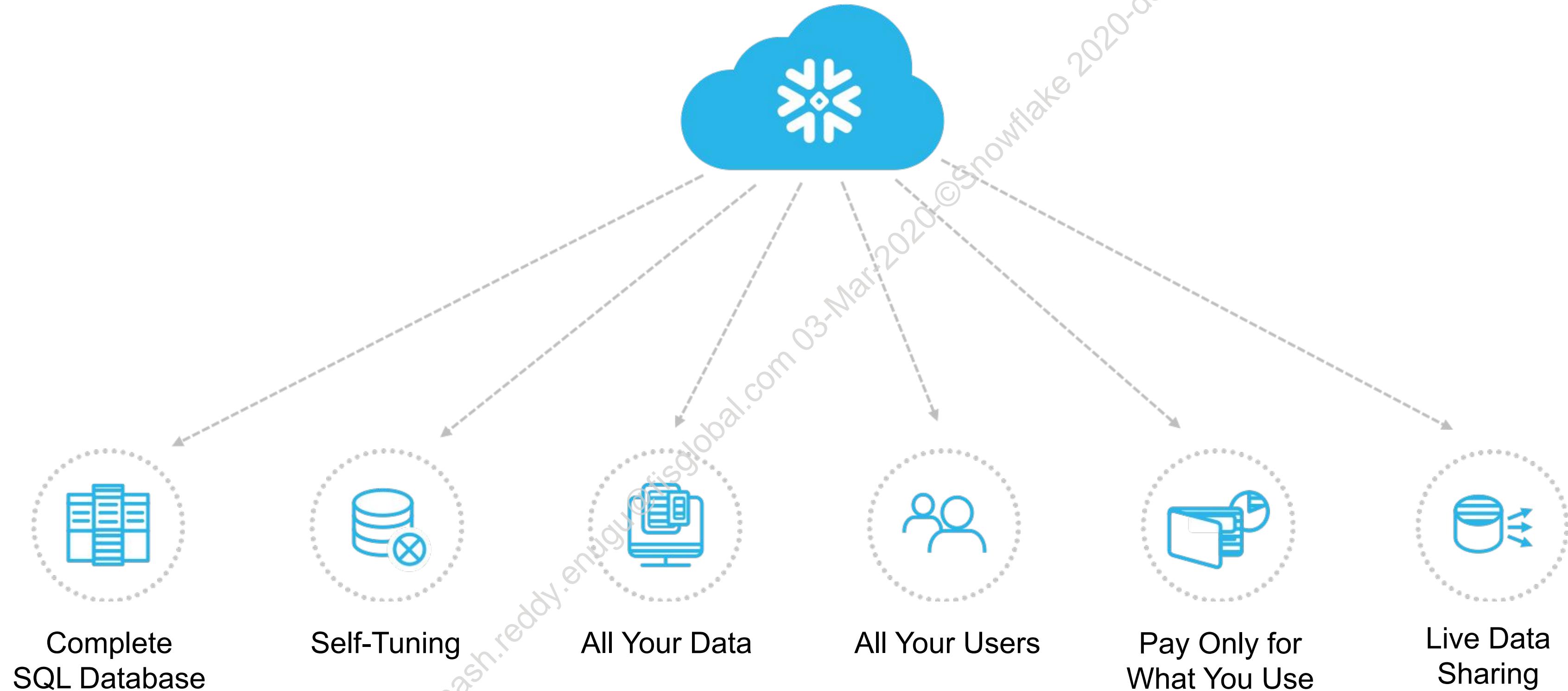
LEGACY DATA LANDSCAPE



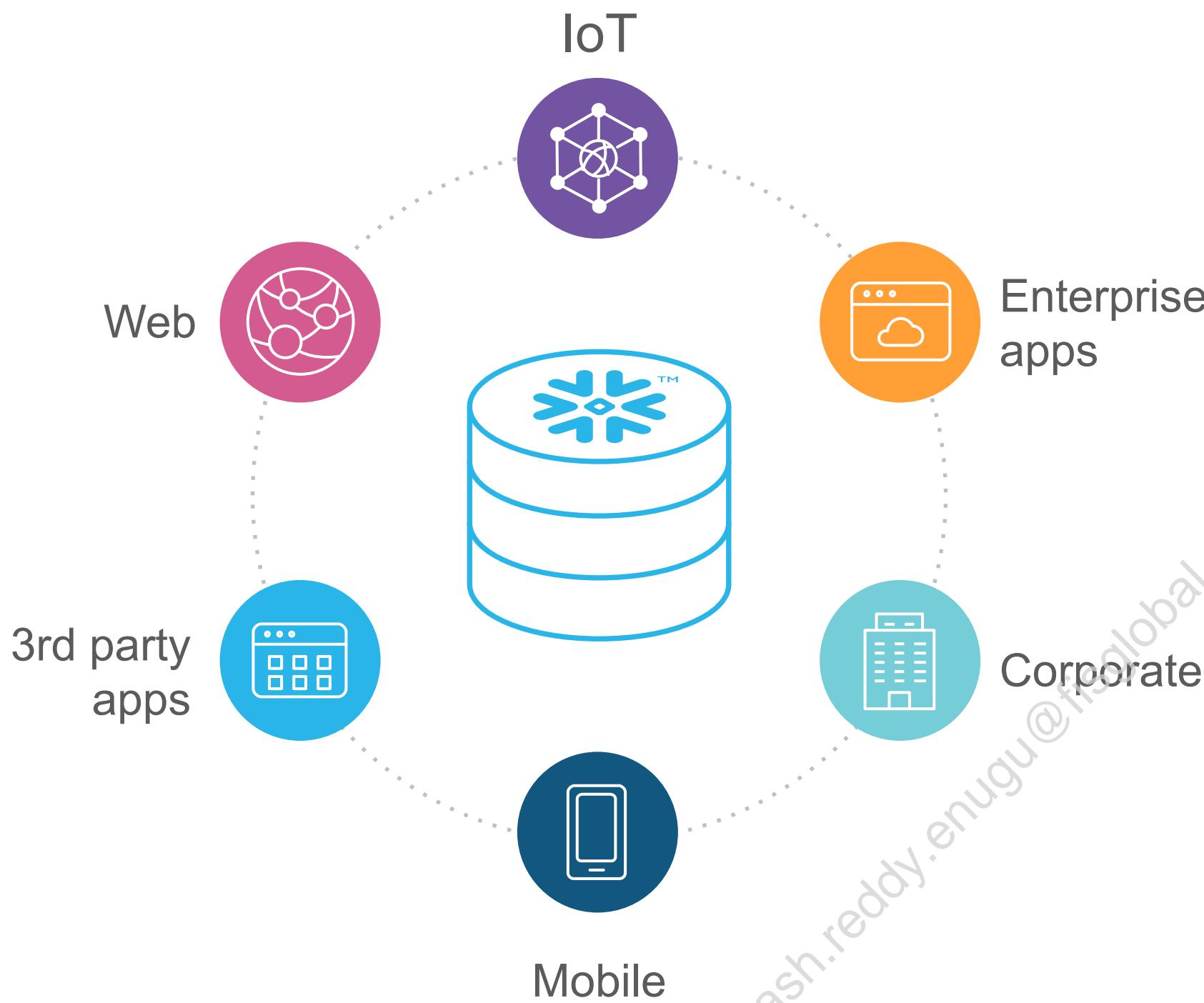
MODERN DATA LANDSCAPE WITH SNOWFLAKE



DATA WAREHOUSE BUILT FOR THE CLOUD

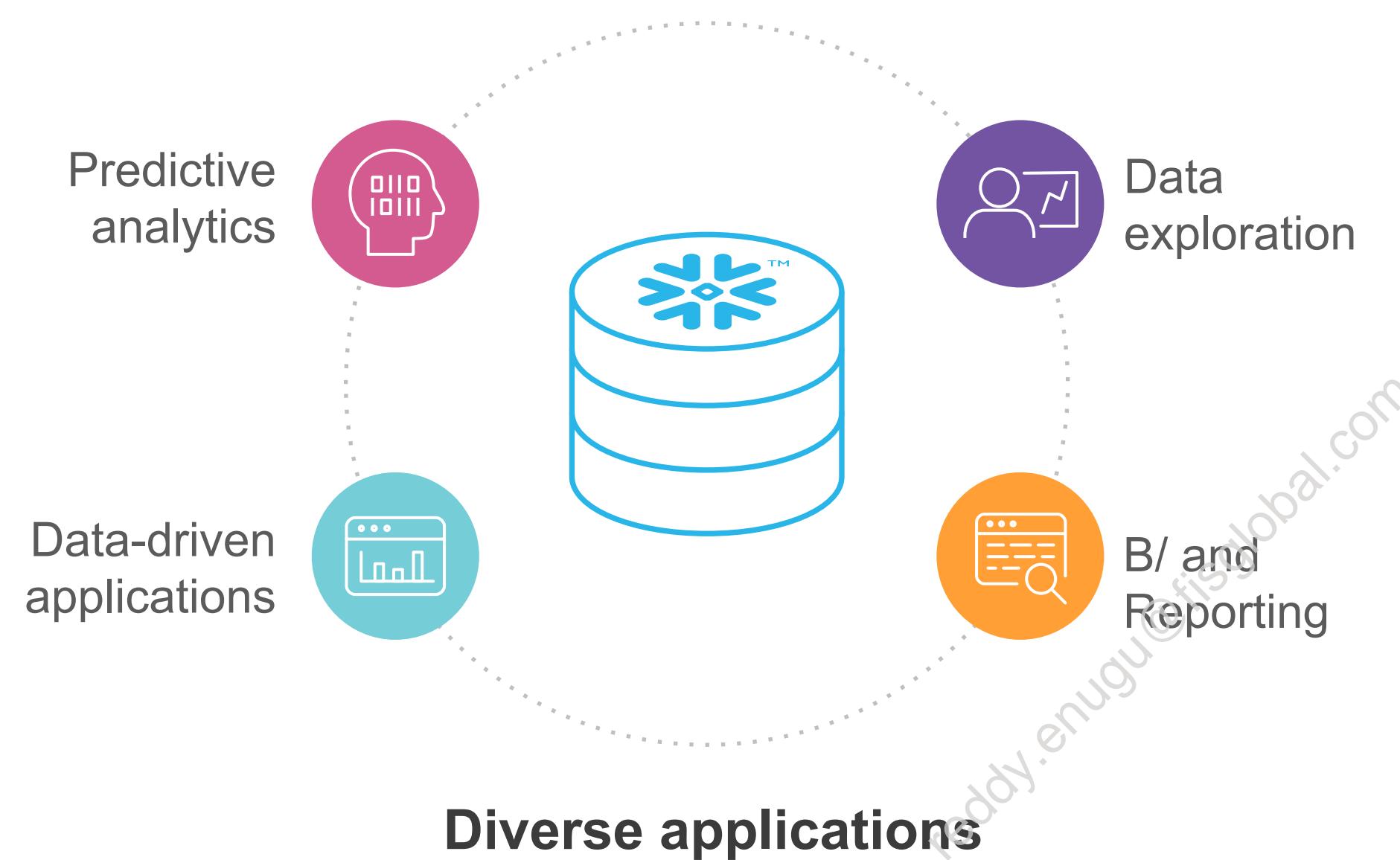


ALL YOUR DATA



- Structured and semi-structured data (JSON, XML, Avro)
- Centralized storage of data, accessible by any user and application
- Multi-petabyte scale

ALL YOUR USERS



- Multiple groups access data at the same time with no performance degradation
- Supports an unlimited number of simultaneous users
- Resources grow and shrink automatically
- Loading data does not impact query performance



NEAR-ZERO MANAGEMENT

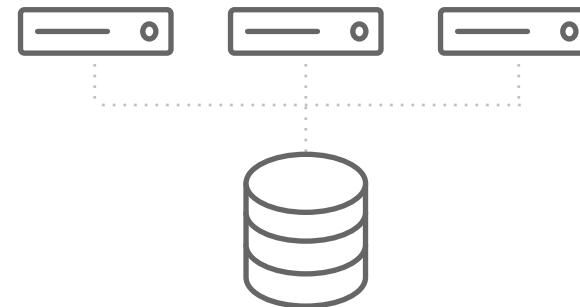


- Load data and run queries, we do the rest
- Zero infrastructure and admin costs
- Secure and highly available
- Fully managed with no knobs or tuning required
- No indexes, distribution keys, partitioning, or vacuuming

A NEW ARCHITECTURE FOR DATA WAREHOUSING

MULTI-CLUSTER, SHARED DATA, IN THE CLOUD

Traditional Architectures



Shared-disk

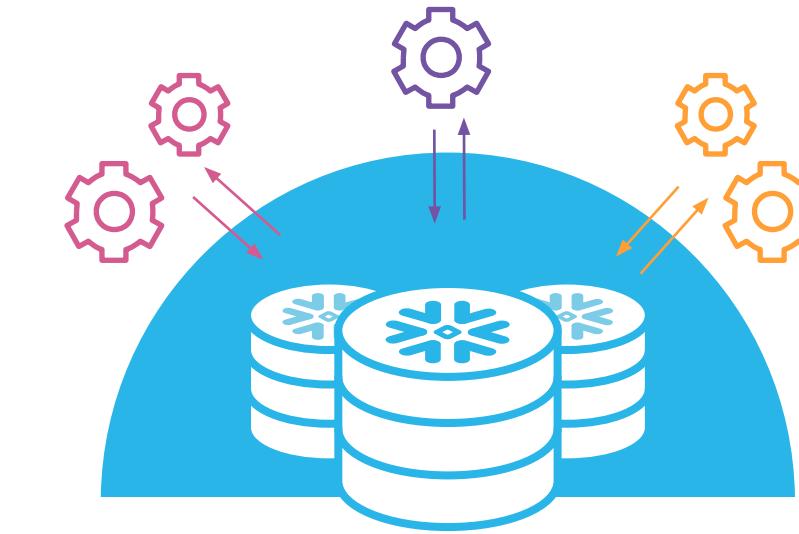
Shared storage
Single cluster



Shared-nothing

Decentralized, local storage
Single cluster

Snowflake

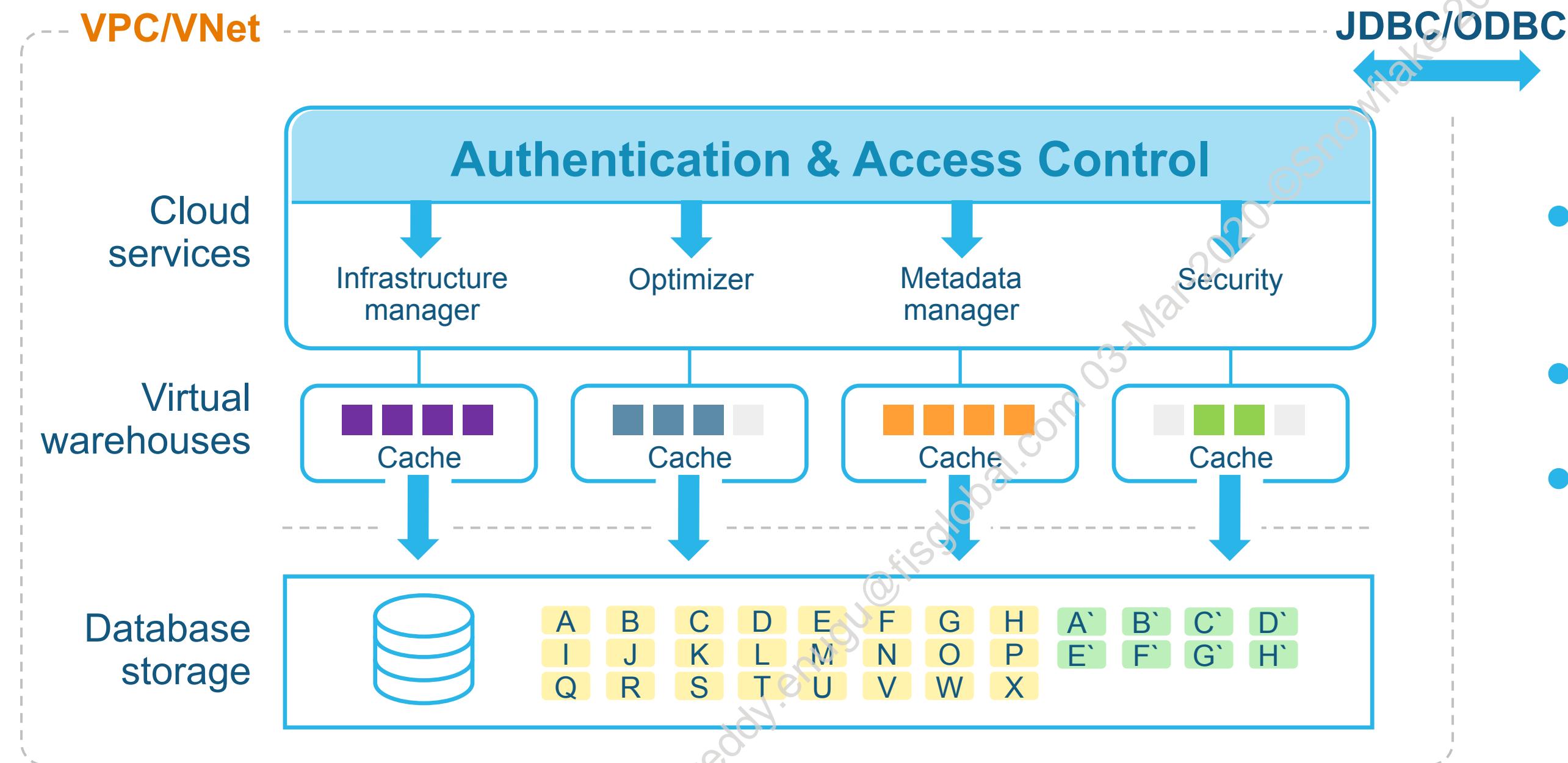


Multi-cluster, shared data

Centralized, scale-out storage
Multiple, independent compute clusters



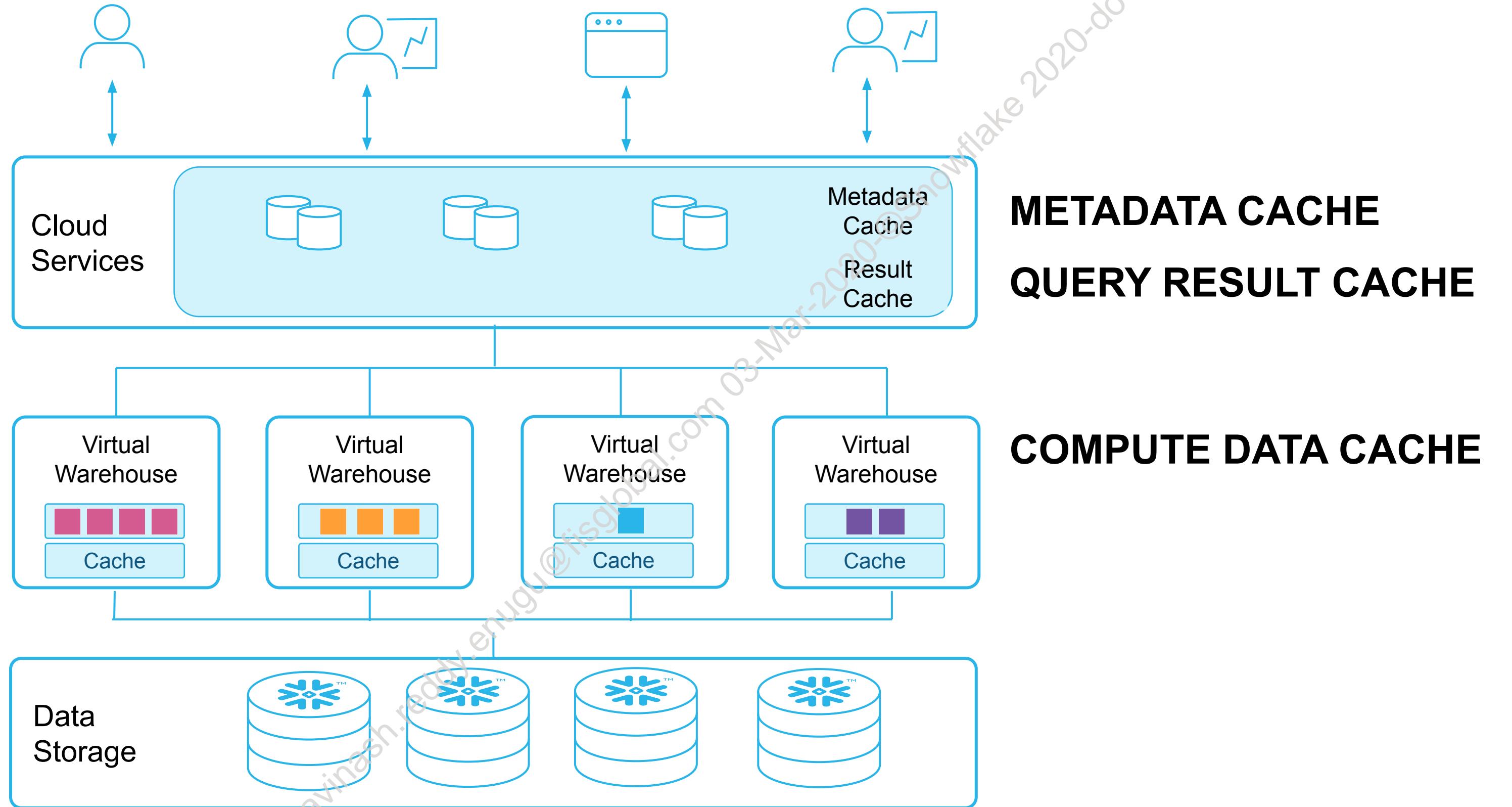
MULTI-CLUSTER SHARED DATA ARCHITECTURE



- Storage decoupled from compute
- All data in one place
- Dynamically combine storage and compute

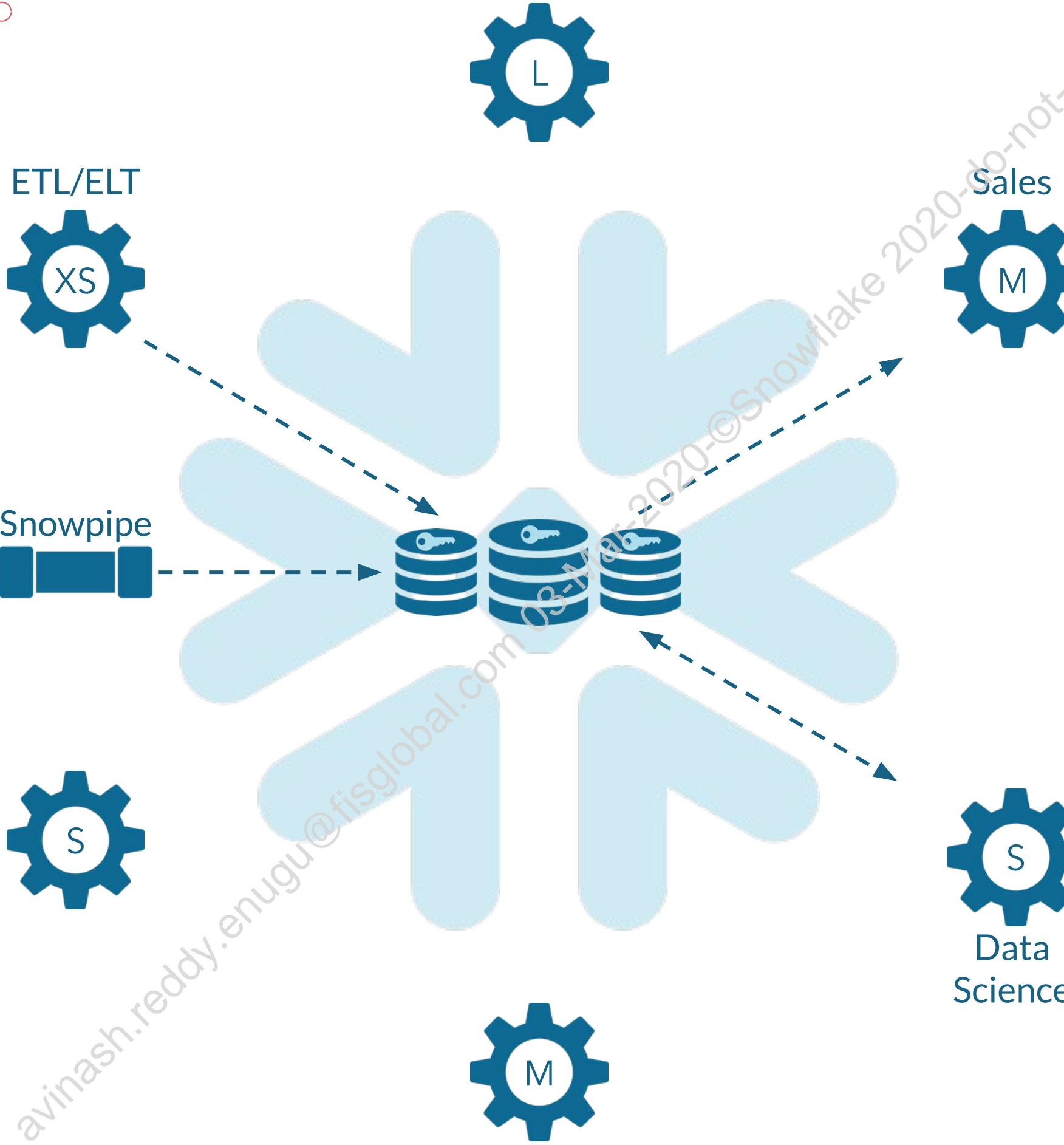
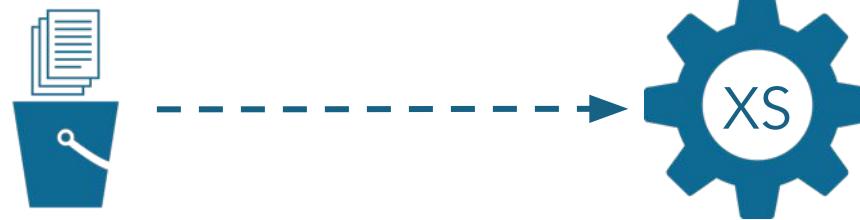


CACHING IN SNOWFLAKE





ETL/ELT

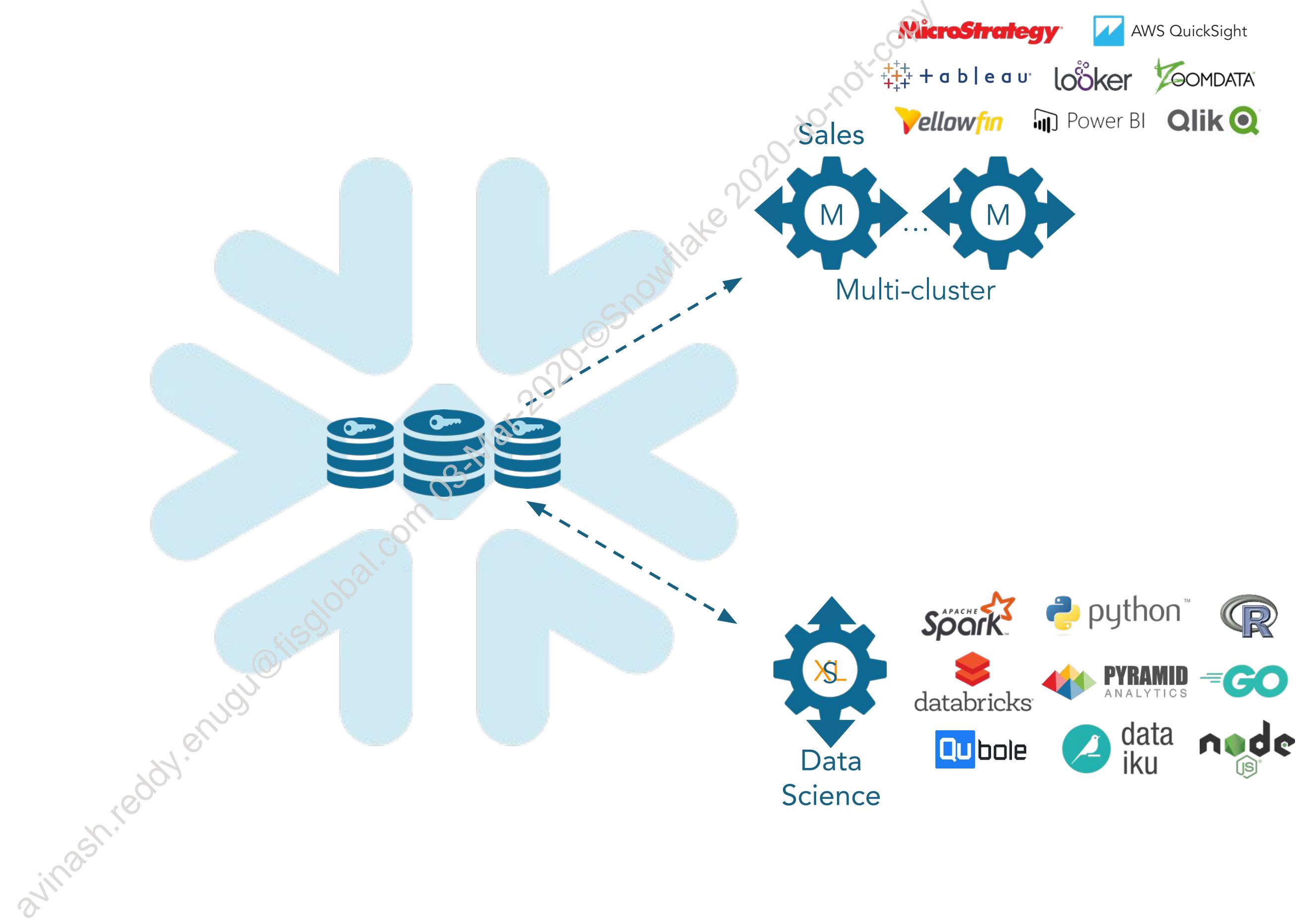
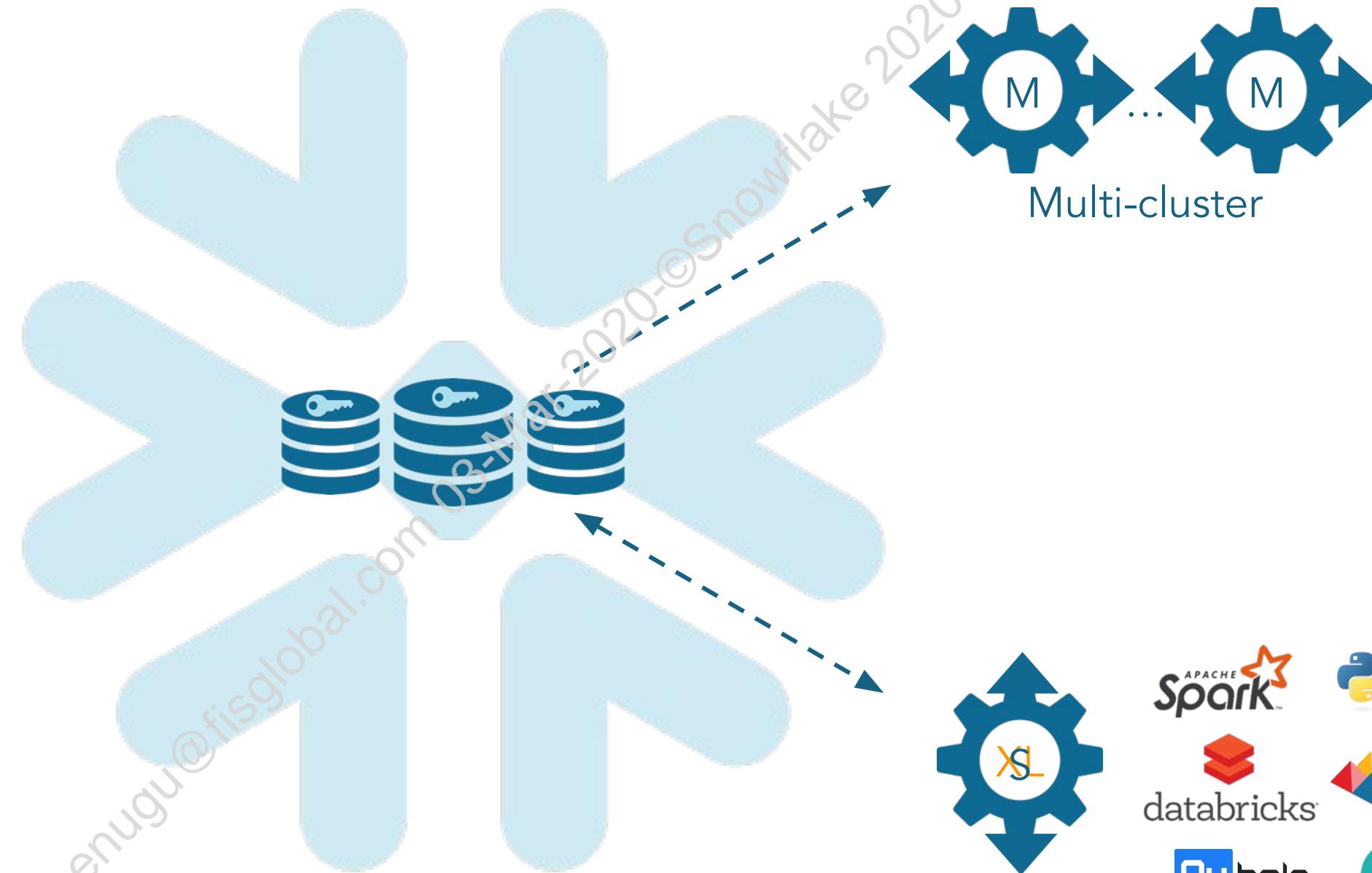


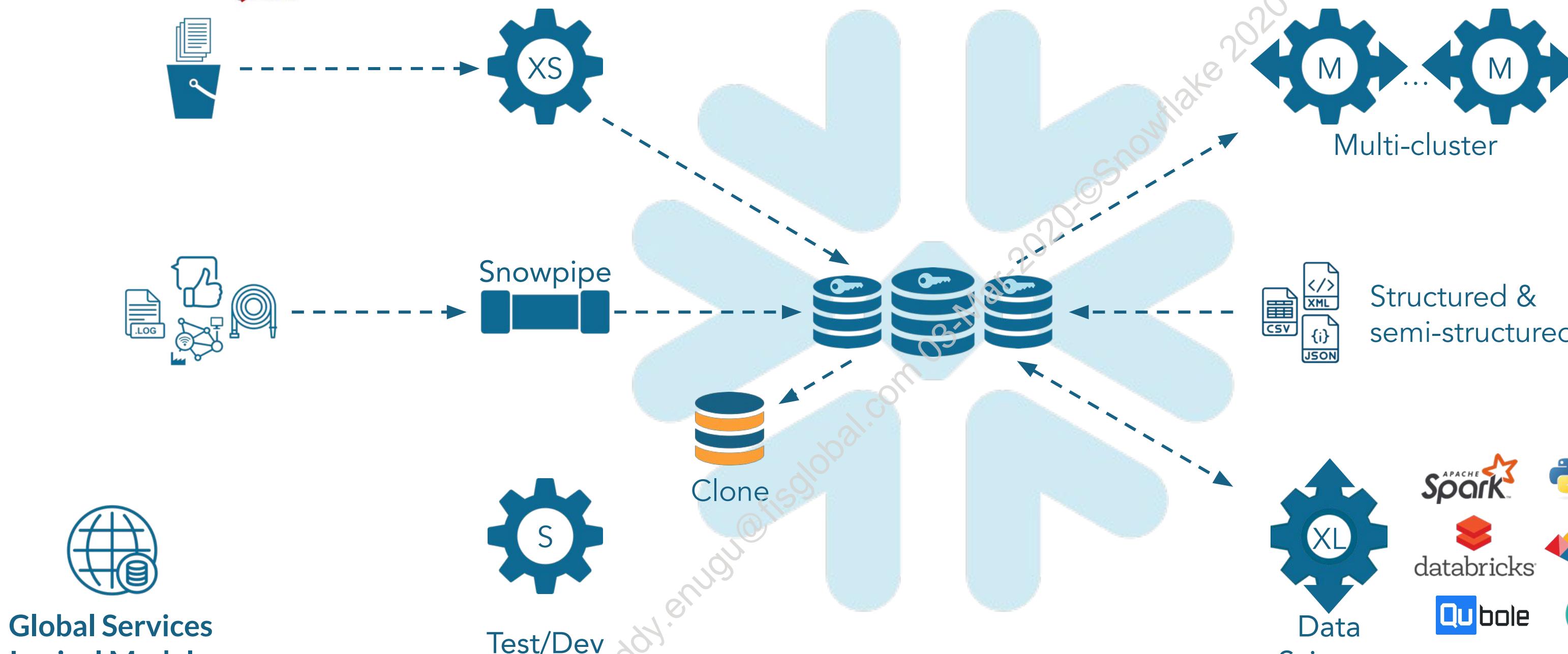
Data Science

**Global Services
Logical Model**

**Security
Query Planning & Optimization
Transactional Control**

 **Global Services**
Logical Model
Security
Query Planning & Optimization
Transactional Control





Global Services
Logical Model

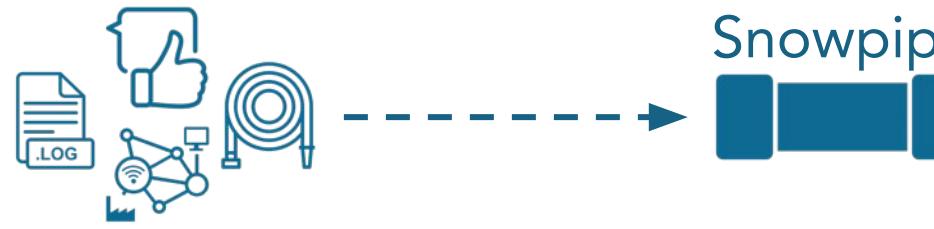
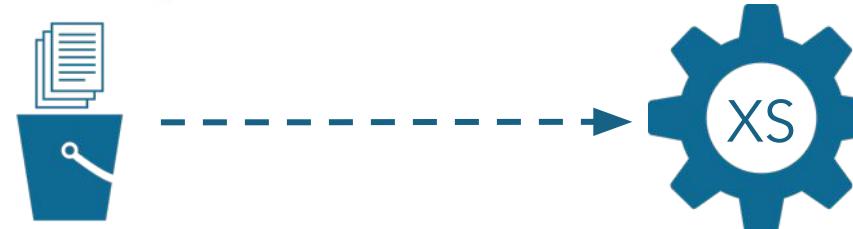
Security

Query Planning & Optimization

Transactional Control



ETL/ELT



**Global Services
Logical Model**

Security

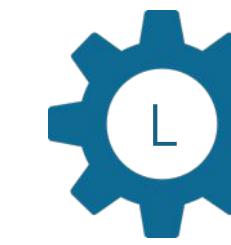
Query Planning & Optimization

Transactional Control



© 2019 Snowflake Computing Inc. All Rights Reserved

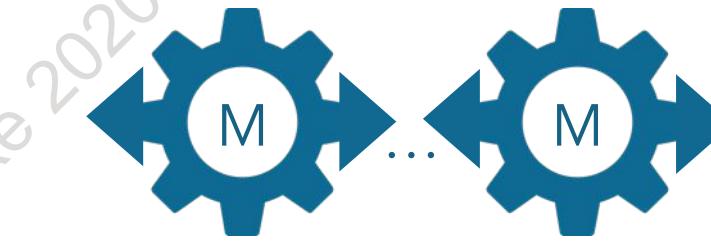
Finance



Data protection & time travel



Sales



Multi-cluster

Structured &
semi-structured



Data
Science

Clone



External

Test/Dev



Test/Dev



External

Share



python™ R



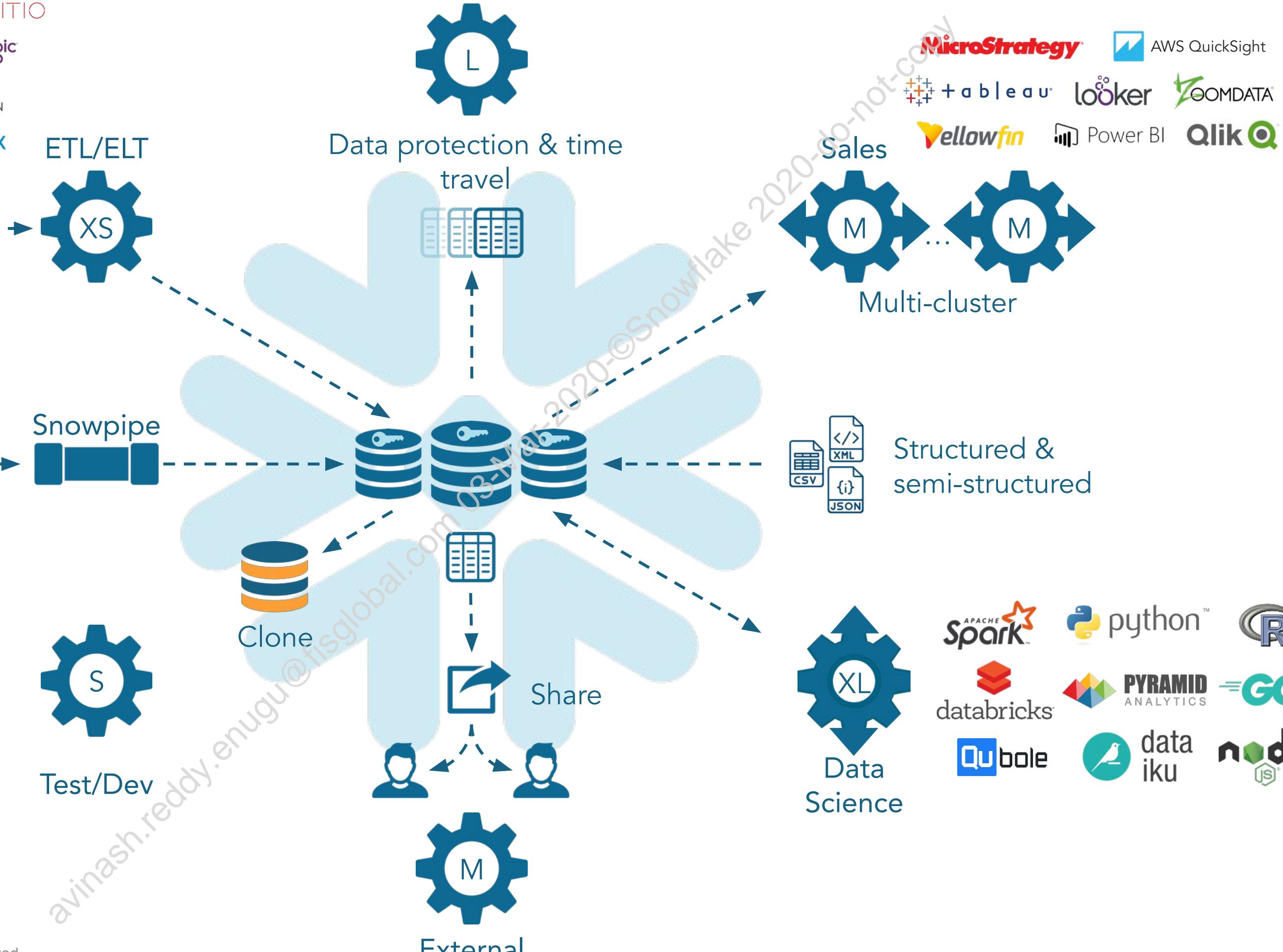
=GO



data iku



node



SNOWFLAKE EDITIONS

Standard	Premier	Enterprise	Business Critical	Virtual Private Snowflake (VPS)
<p>Standard</p> <ul style="list-style-type: none">• Complete SQL Data Warehouse• Secure Data Sharing across regions / clouds• Business hour support M-F• 1 day of time travel• Always-on enterprise grade encryption in transit and at rest• Customer dedicated virtual warehouses• Federated authentication• Database Replication	<p>Premier</p> <ul style="list-style-type: none">Standard+<ul style="list-style-type: none">• Premier Support 24 x 365• Faster response time• SLA with refund for outage	<p>Enterprise</p> <ul style="list-style-type: none">Premier +<ul style="list-style-type: none">• Multi-Cluster warehouse• Up to 90 days of time travel• Annual review of all encrypted data• Materialized Views• AWS PrivateLink available for extra fee	<p>Business Critical</p> <ul style="list-style-type: none">Enterprise +<ul style="list-style-type: none">• HIPAA support• PCI compliance• Data encryption everywhere• Tri-Secret Secure using customer-managed keys (AWS)• AWS PrivateLink support• Enhanced security policy• Database Failover and Fallback for business continuity	<p>Business Critical +</p> <ul style="list-style-type: none">• Customer dedicated virtual servers wherever the encryption key is in memory• Customer dedicated metadata store• Additional operational visibility



GLOBAL SNOWFLAKE

CURRENTLY SUPPORTED REGIONS



aws
US West 2
US East 1
CA Central 1
EU West 1 (Ireland)
EU Central 1 (Frankfurt)
AP Southeast 1 (Singapore)
AP Southeast 2 (Sydney)



Azure
East US 2
US Gov Virginia
Canada Central
West Europe
Australia East
Southeast Asia



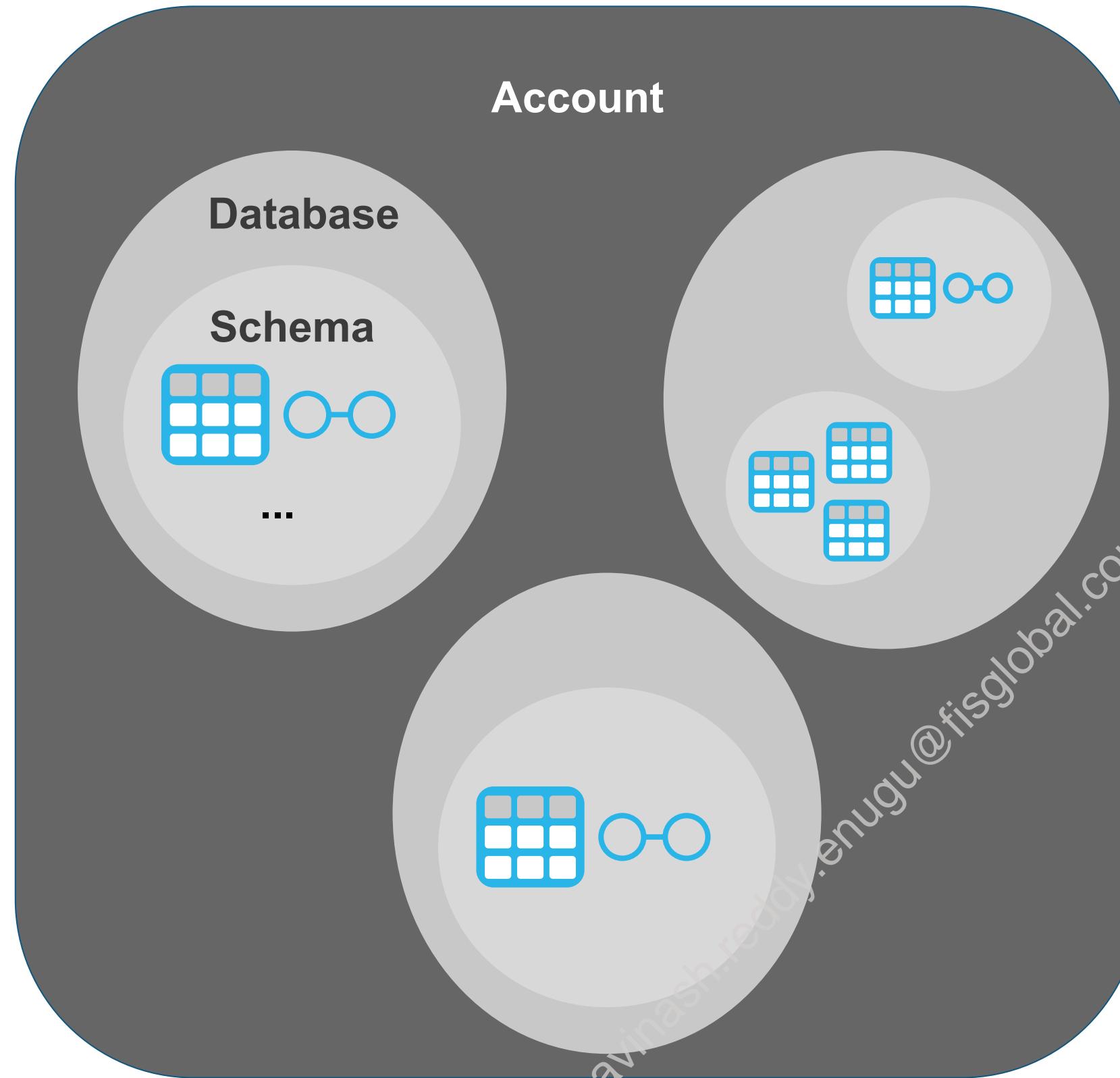
<https://docs.snowflake.net/manuals/user-guide/intro-regions.html>

Snowflake Structure

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



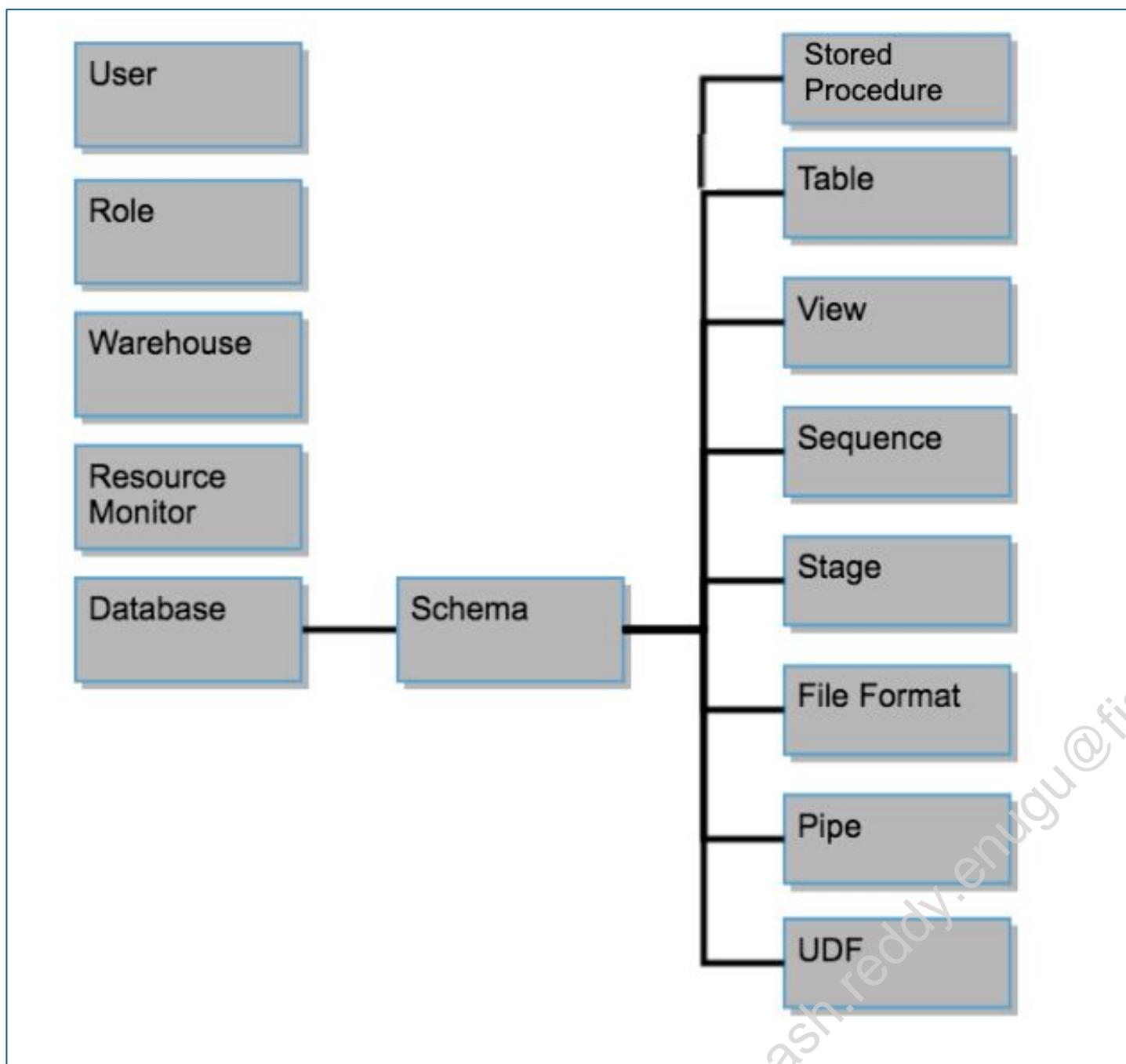
LOGICAL DATA ORGANIZATION



- Databases and schemas logically organize data within a Snowflake account
- A database is a logical grouping of schemas
 - Each database belongs to a single account
- A Schema is a logical grouping of database objects, such as tables and views

SNOWFLAKE OBJECTS

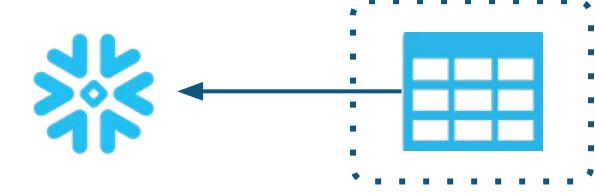
Account



- All Snowflake objects reside within logical containers, with the top level container being the Snowflake Account
- All objects are individually securable
- Users can perform operations on objects in the system via “privileges” that are “granted” to roles, and roles are then granted to users
- Example Privileges:
 - Create a virtual warehouse
 - List tables in a schema
 - Insert data into a table
 - Select data from a table



TABLE TYPES



PERMANENT

- Persist until dropped
- Designed for data that requires the highest level of data protection and recovery
- Default table type

TEMPORARY

- Persist and tied to a session (think single user)
- Used for transitory data (for example, ETL/ELT)

TRANSIENT*

- Persist until dropped
- Multiple user
- Used for data that needs to persist, but does not need the same level of data retention as a permanent table

EXTERNAL

- Persist until removed
- Snowflake “over” an external data lake
- Data accessed via an external stage
- Read-only

Time Travel

Up to 90 days with Enterprise

0 or 1 days

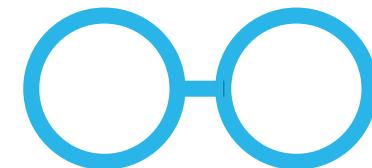
0 or 1 days

x

Fail-Safe



VIEW TYPES



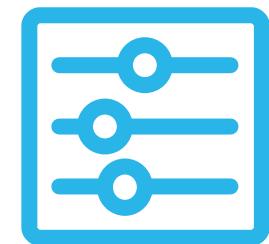
Standard View

- Default view type
- Named definition of a query--SELECT statement
- Executes as executing role
- Underlying DDL available to any role with access to the view.



Secure View

- Definition and details only visible to authorized users
- Executes as owning role
- Snowflake query optimizer bypasses optimizations used for regular views



Materialized View

- Behaves more like a table
- Results of underlying query are stored
- Auto-refreshed
- Secure Materialized View is also supported

Module 1: Architecture & Overview

DEMO: Using the Snowflake Web UI

15 minutes

- Overview
- Worksheet Context
- Tips and Tricks



Module 1: Architecture & Overview

Exercise 1.1: Take a Quick Test Drive

25 minutes

Note:

In this lab, you will define your user's default role, namespace, and warehouse so they will automatically be set in every worksheet you open.

Tasks:

- Log in to your Snowflake training account
- Create a database and a table for later use
- Run queries on sample data

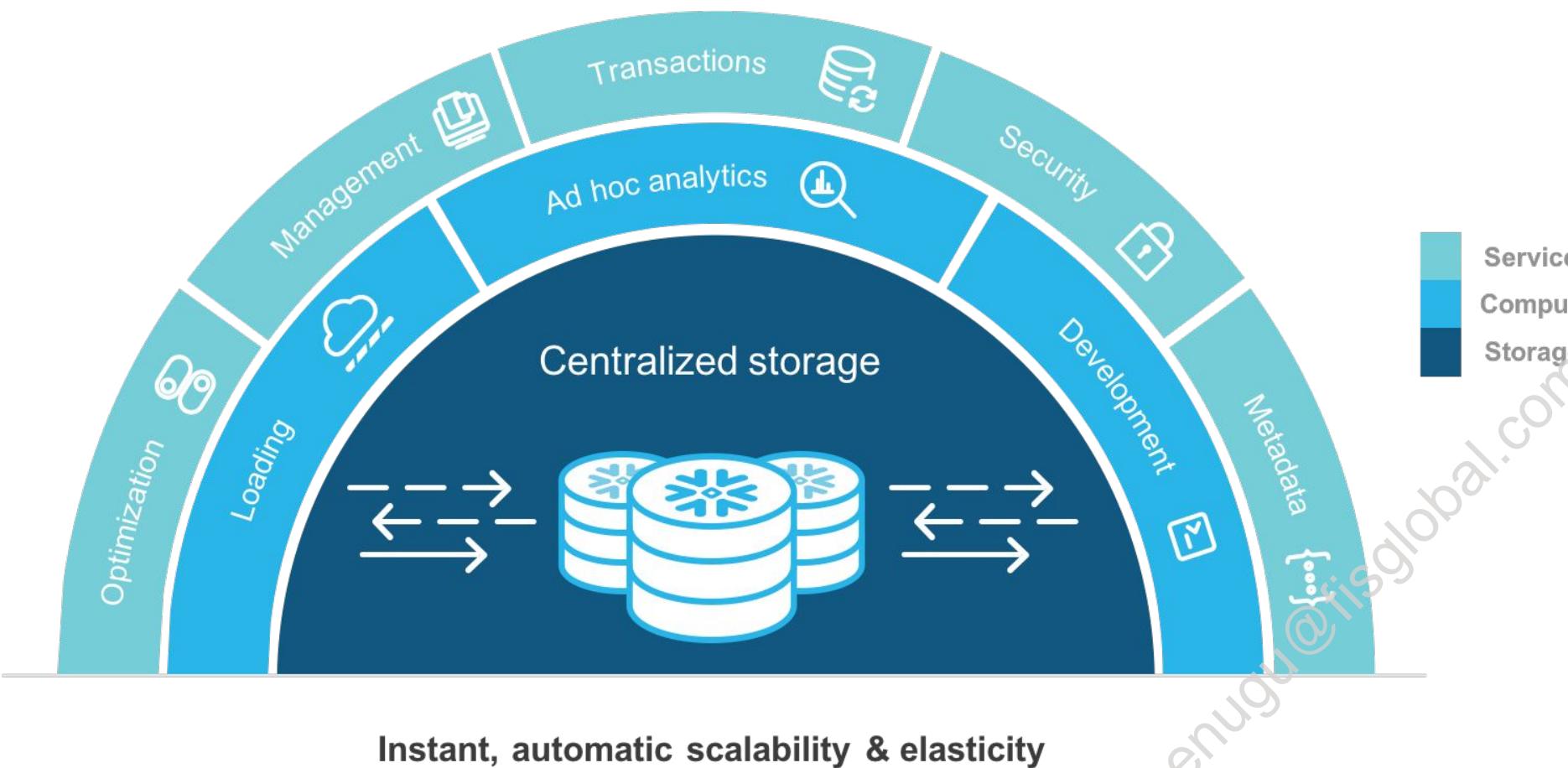


Cloud Services Layer

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



SNOWFLAKE ARCHITECTURE

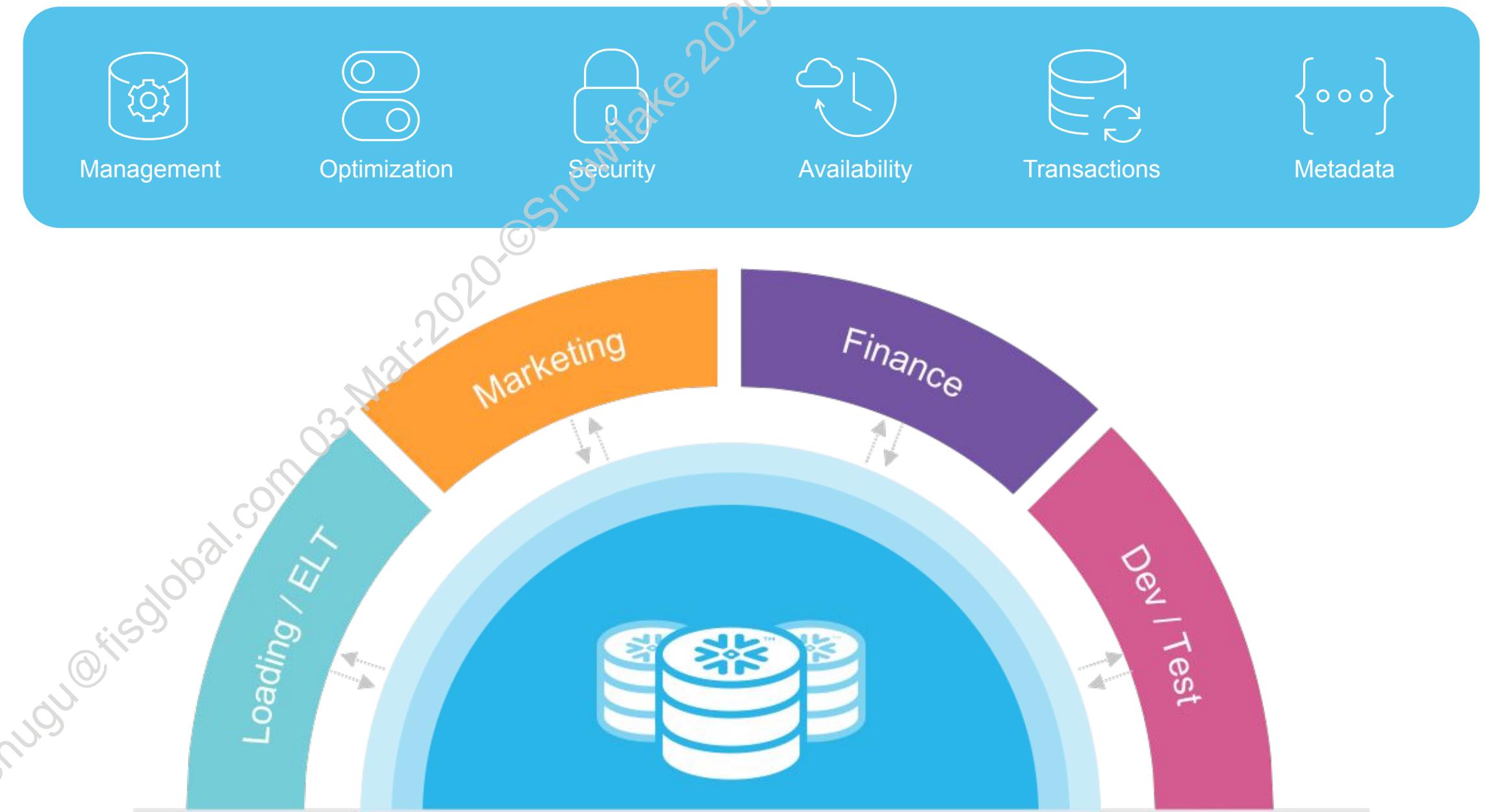


3 architectural layers:

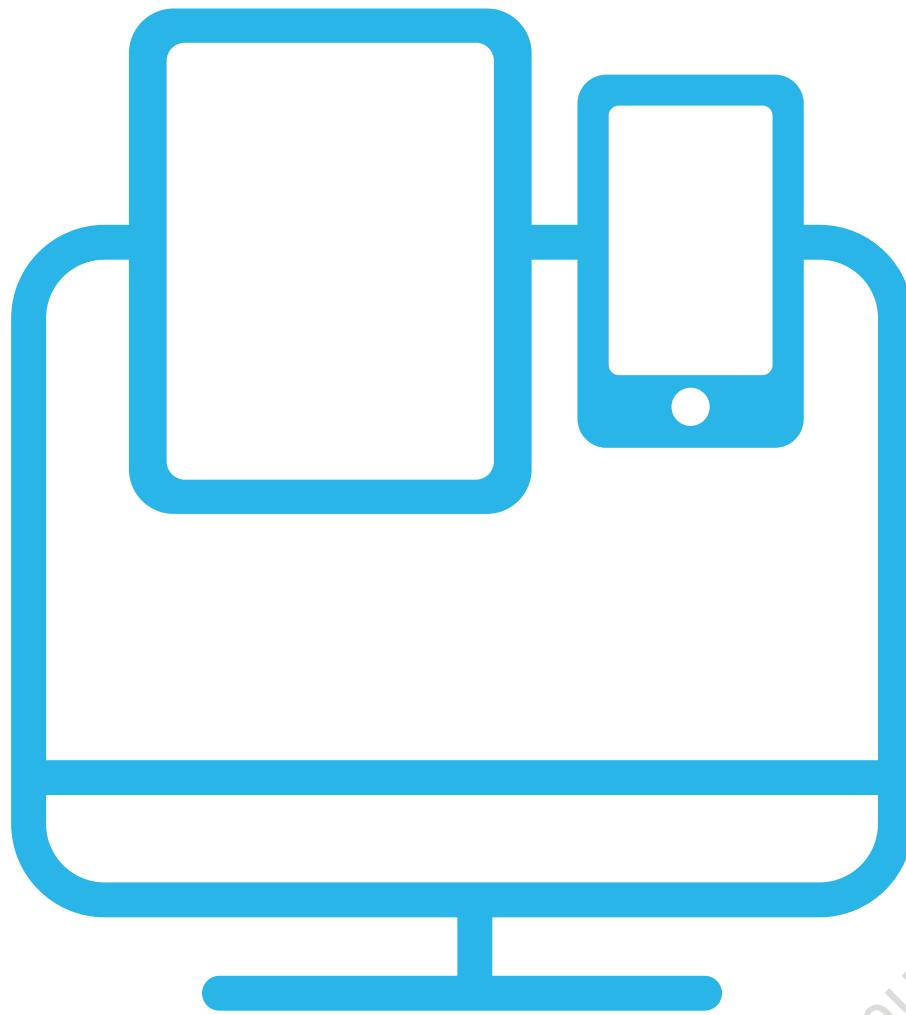
- Storage Layer
- Compute (Virtual Warehouse) Layer
- Global Services Layer

SNOWFLAKE ARCHITECTURE: CLOUD SERVICES

- "Brains" of the service
- Coordinates activities across Snowflake
- Runs on compute instances provisioned by Snowflake

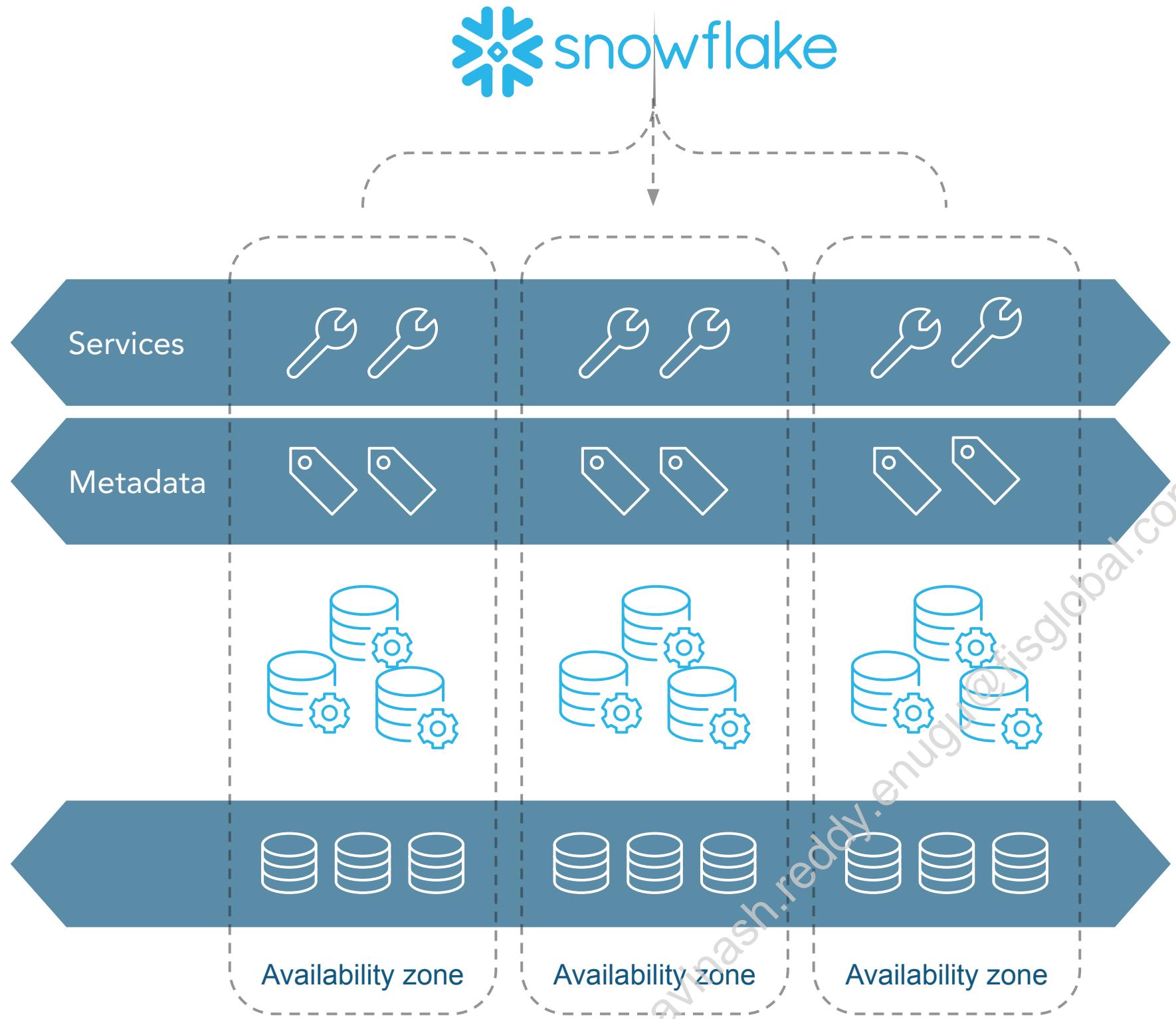


MANAGEMENT



- Centralized management for all storage
- Manages the compute that works with the storage

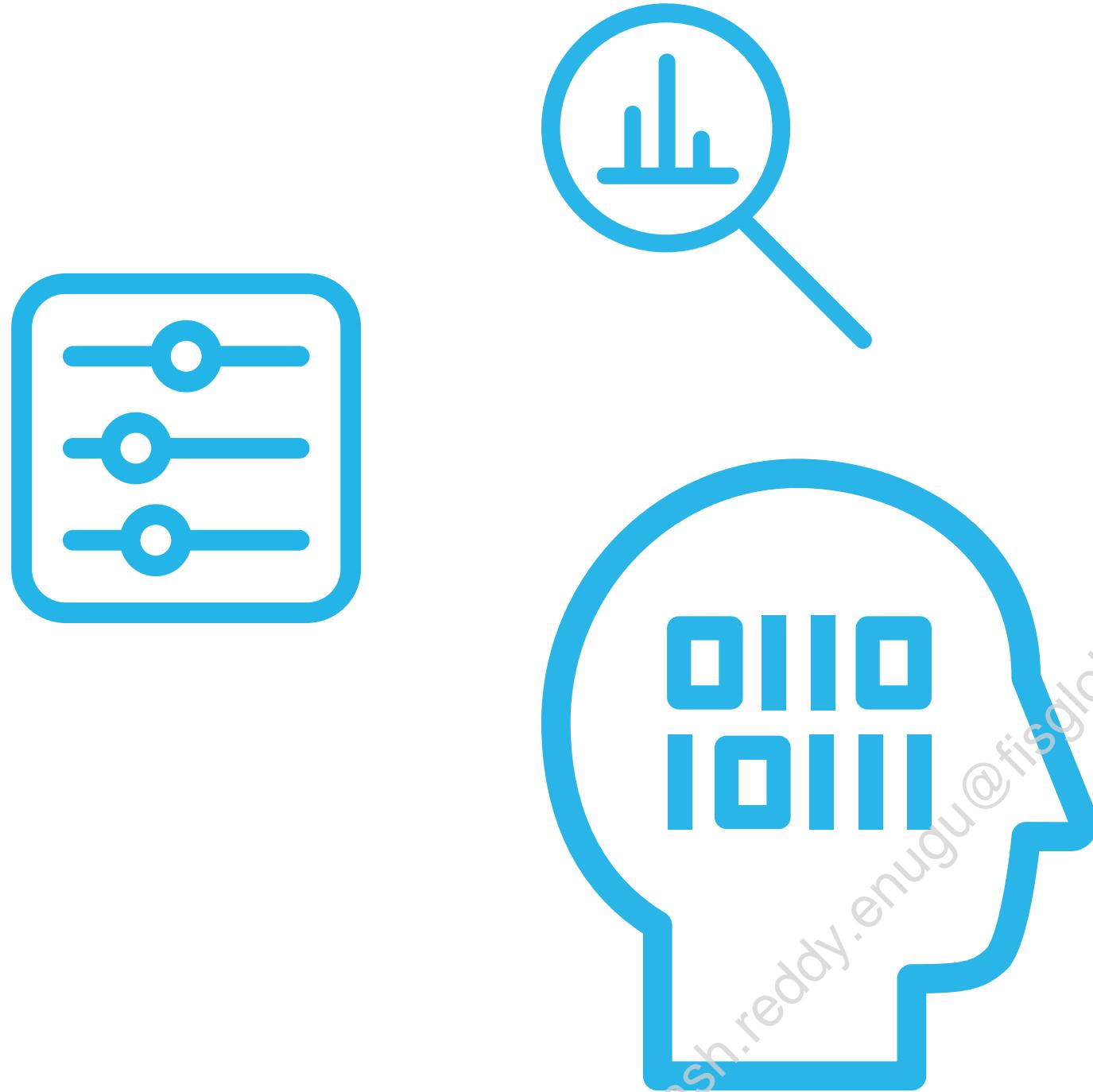
BUILT-IN CONTINUOUS AVAILABILITY



- Transparently synchronizes data across availability zones
 - Geographic separation
 - Separate power grids
- Fully online updates and patches
- Fully managed by Snowflake



OPTIMIZER SERVICE



- SQL Optimizer
 - Cost-based optimization (CBO)
- Automatic JOIN order optimization
 - No user input or tuning required
- Automatic statistics gathering
- Pruning using metadata about micro-partitions

TRANSACTION SERVICE



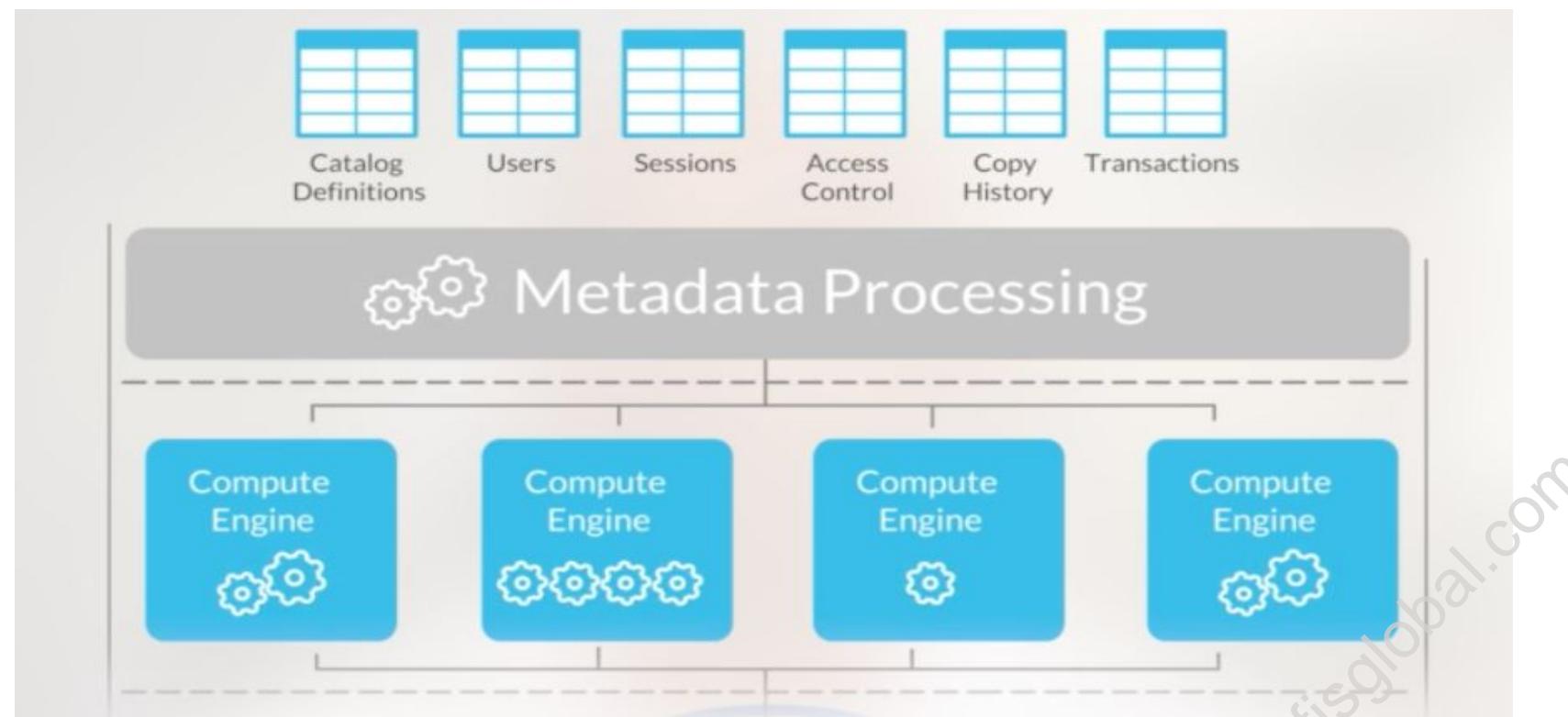
- Full CRUD operations
- Fully ACID (Atomic, Consistent, Isolated, Durable)
- Concurrency management

SECURITY



- Authentication
- Access control for users and roles
- Access control for shares
- Encryption and key management

METADATA MANAGEMENT



- Stores metadata as data is loaded into the system
- Handles queries that can be processed completely from metadata
- Used for Time Travel and Cloning
- Every aspect of Snowflake architecture leverages metadata

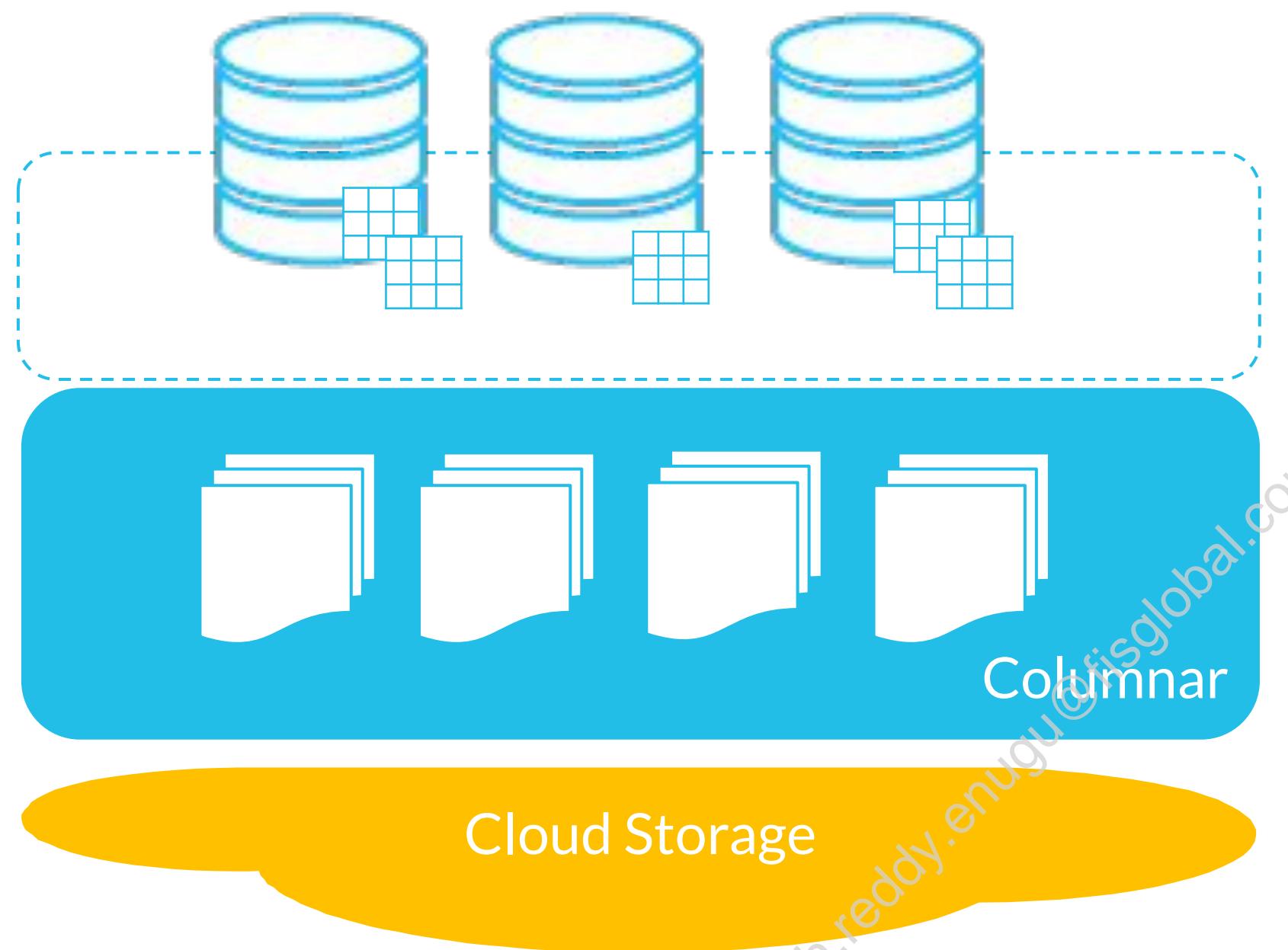


Data Storage Layer

avinash.reddy.enugu@fisglobal.com 03 Mar 2020 ©Snowflake 2020-do-not-copy

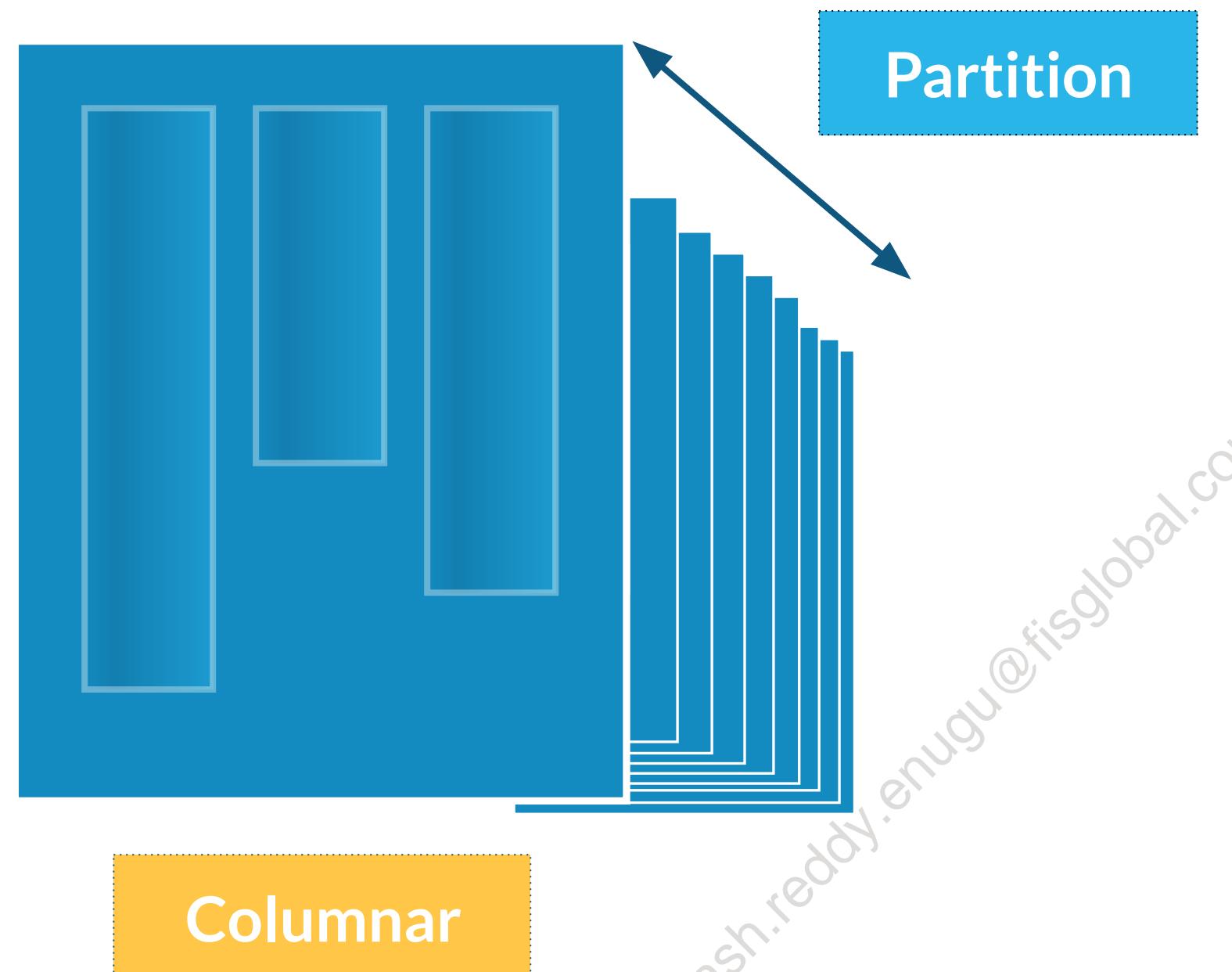


STORAGE LAYER



- Hybrid Columnar
- Automatic micro-partitioning
- Natural data clustering and optimization
- Native Semi-Structured data support and optimization
- All storage within Snowflake is billable (compressed)

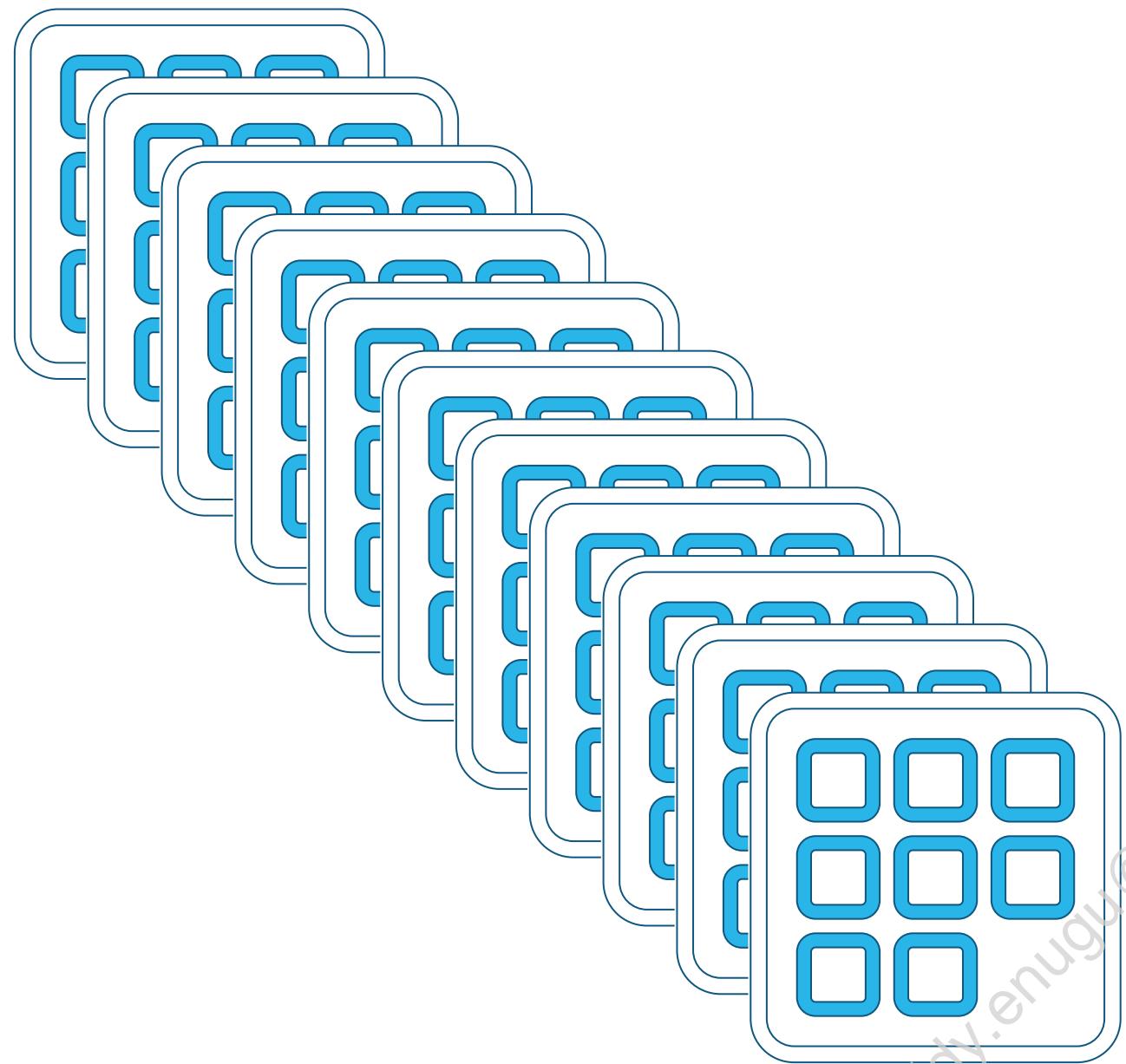
COLUMNAR COMPRESSION



- Ingestion automatically analyzes and compresses data into table on load
- Find the optimal compression scheme for each data type
 - Storing same data type enables efficient compression
- Columns grow and shrink independently
- Significant performance benefit by reduction I/O and storage



MICRO-PARTITIONS



- Contiguous units of storage that hold table data
 - 50 - 500 MB of uncompressed data
 - Generally...Max 16MB (Compressed)
- MANY micro-partitions per table
- **IMMUTABLE !!!!**
- Services layer stores metadata about every micro-partition
 - MIN/MAX (Range of values in each column)
 - Number of distinct values

MICRO-PARTITIONING

ID	Name
1	John
2	Scott
3	Mary
4	Jane
5	Jack
6	Claire

Table written to
micro-partitions

ID	1	2	3
Name	John	Scott	Mary

P 1

ID	4	5	6
Name	Jane	Jack	Claire

P 2

- Physical data files that comprise Snowflake's logical tables
- Automatically-created contiguous units of storage, partitioned based on ingestion order
 - Attempts to preserve natural data co-location
- Immutable - updates create new micro-partition versions



METADATA

MICRO-PARTITION FILES

ID	1	2	3
Name	John	Scott	Mary

PARTITION METADATA

ID: 1 - 3
Name: J - S

ID	4	5	6
Name	Jane	Jack	Claire

ID: 4 - 6
Name: C - J

- Snowflake automatically collects and maintains metadata about tables and their underlying micro-partitions, including:

Table level

- Row count
- Table size (in bytes)
- File references and table versions

Micro-partition column level:

- Range of values
- Number of distinct values
- MIN and MAX values
- NULL count

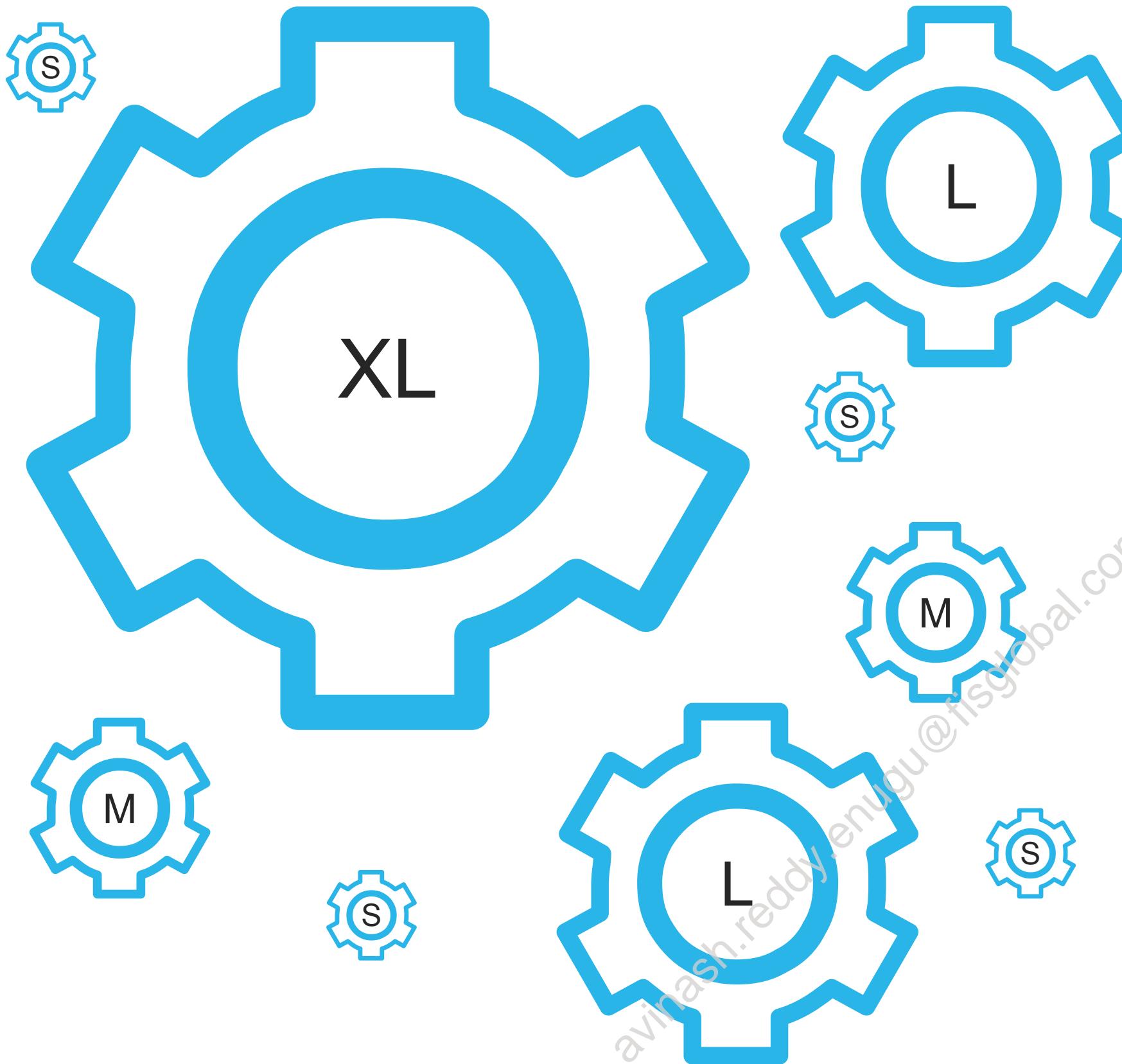


Compute Layer

avinash.reddy.enugu@fisglobal.com 02-Mar-2020 ©Snowflake 2020-do-not-copy



VIRTUAL WAREHOUSES



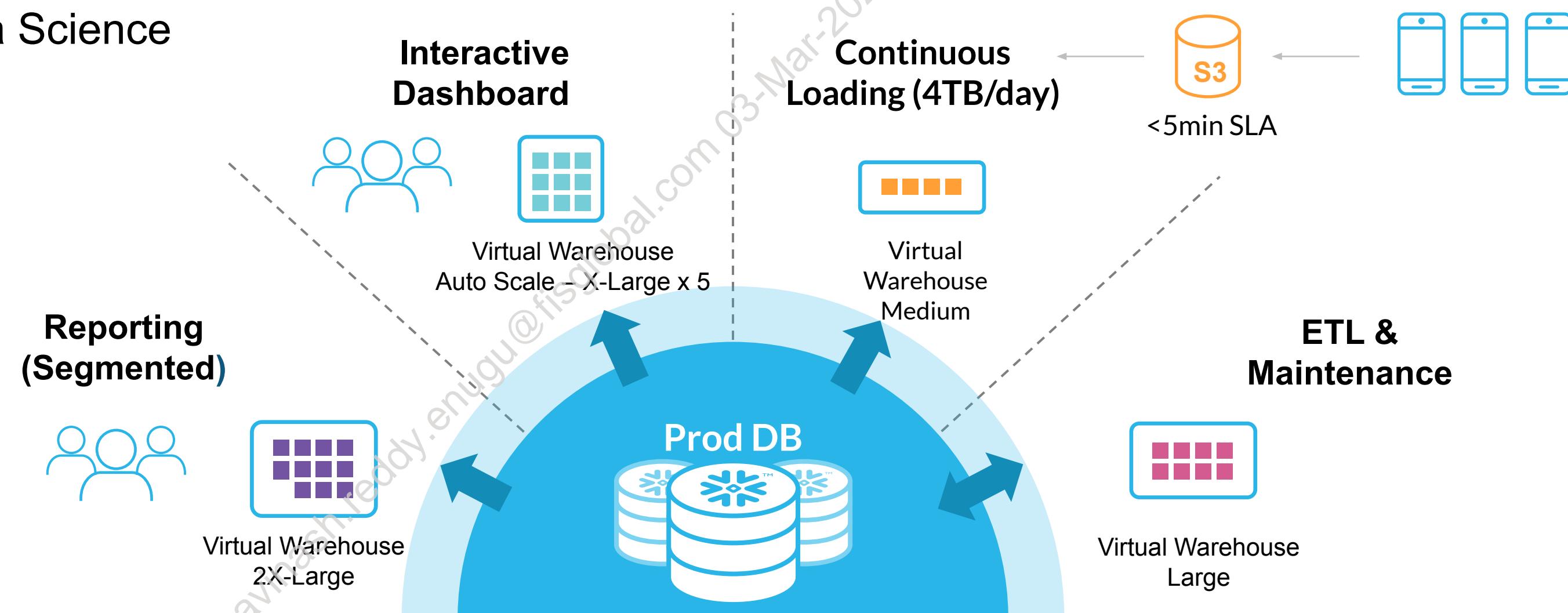
- A named wrapper around a cluster of servers with CPU, memory, and SSD in the cloud
- The larger the warehouse, the more servers in the cluster, the more resources it has
- Extra Small (XS) has one server per cluster
 - Each size up doubles in size
- Jobs requiring compute run on virtual warehouses
- While running, a virtual warehouse consumes Snowflake credits
 - You are charged for compute



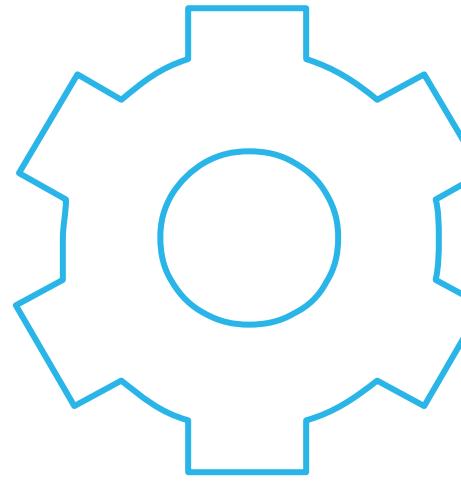
WORKLOAD SEGMENTATION

- Should reflect units of workload management

- ETL
- BI / Dashboards
- Data Science

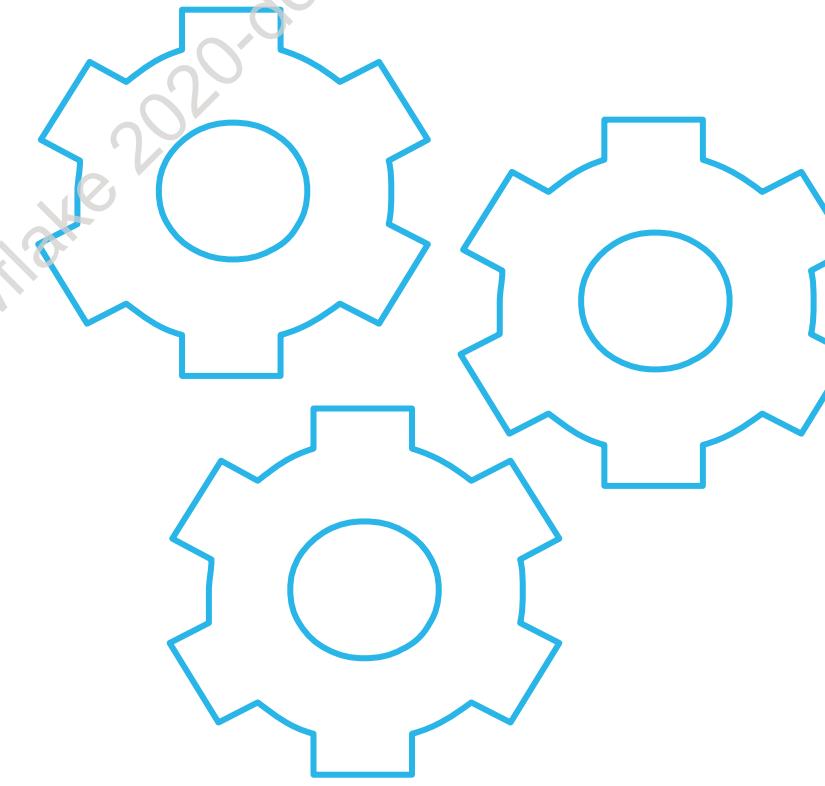


VIRTUAL WAREHOUSE TYPES



Standard

- Will only ever have a single Compute Cluster
- Cannot “scale out”

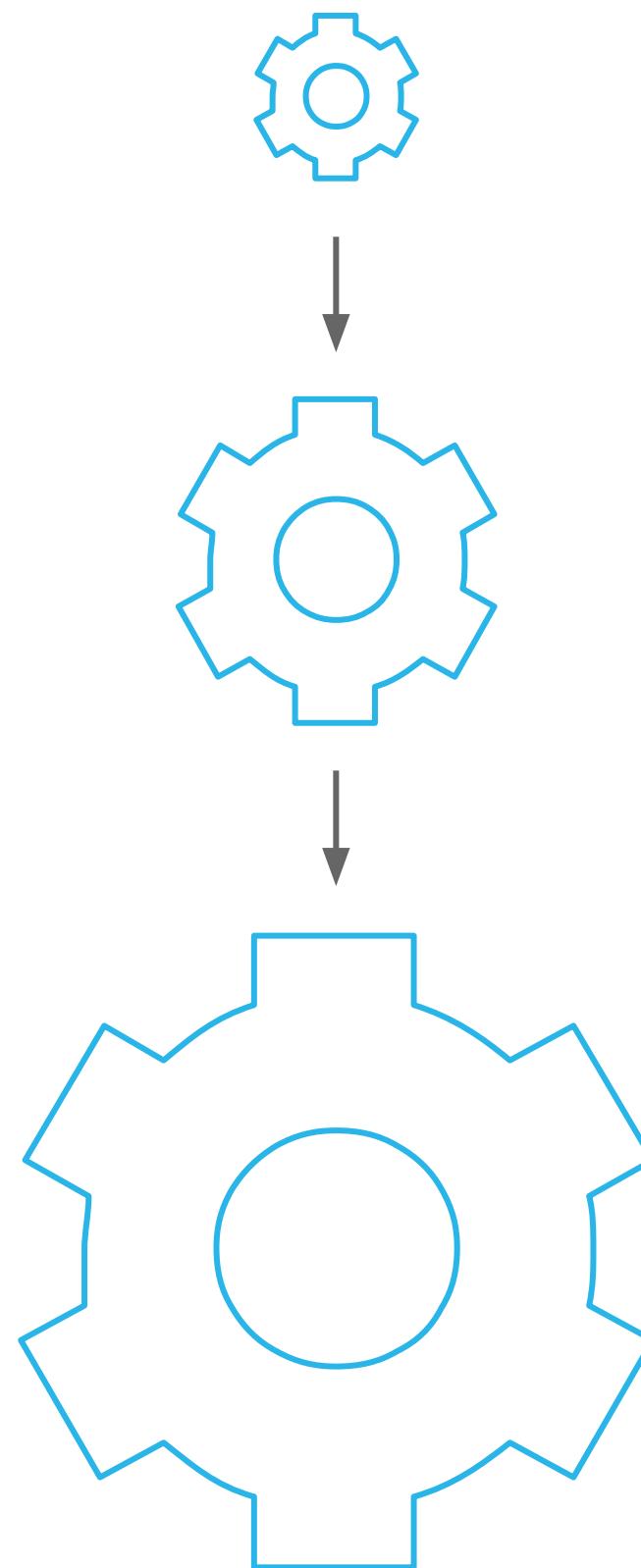


Multi-Cluster Warehouse (MCW)

- Can spawn additional Compute Clusters (scale out) to manage changes in user and concurrency needs
- Enterprise Edition feature

VIRTUAL WAREHOUSE SIZING

- Warehouses are sized in “t-shirt” sizing
- Size determines the number of servers that comprise each **cluster** in a **warehouse**
- Each larger size is double the preceding, in both VMs in the cluster and in Snowflake credits consumed



CREDITS



- Credits are how you are billed for *compute* usage
 - You may have a set number of credits
 - You may be billed monthly for your credits
- Credits are charged based on the number of virtual warehouses you use, their size, and how long you use them
- Warehouse usage (or *compute*) is charged per-second, with a one-minute minimum

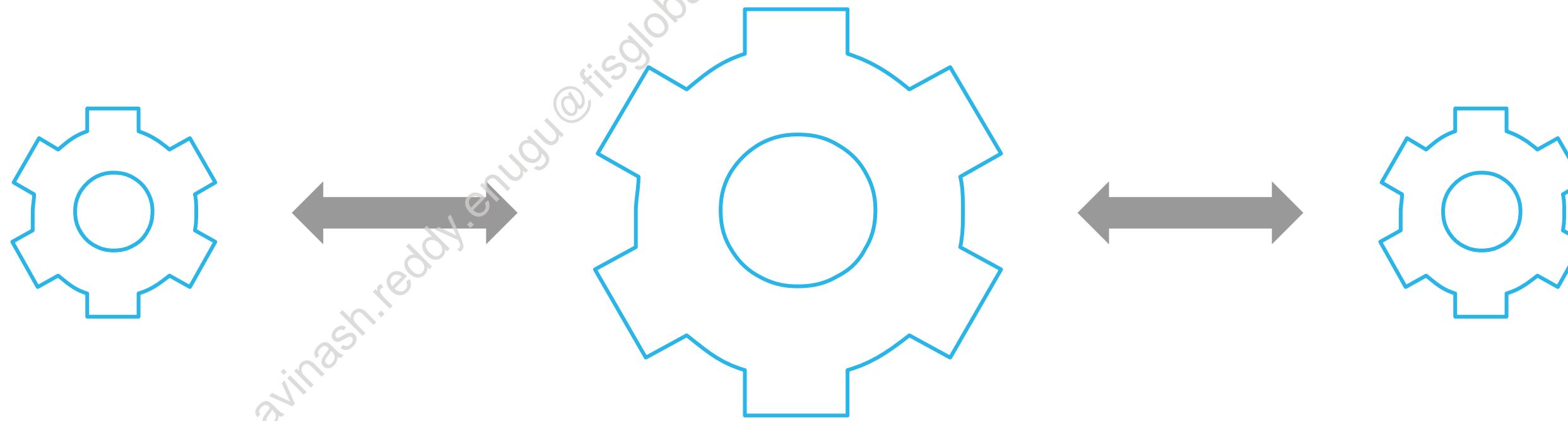
VIRTUAL WAREHOUSE CREDITS

Warehouse Size	Servers	Credits / Hour	Credits / Second
X-Small	1	1	0.0003
Small	2	2	0.0006
Medium	4	4	0.0011
Large	8	8	0.0022
X-Large	16	16	0.0044
2X-Large	32	32	0.0089
3X-Large	64	64	0.0178
4X-Large	128	128	0.0356



RESIZING A WAREHOUSE

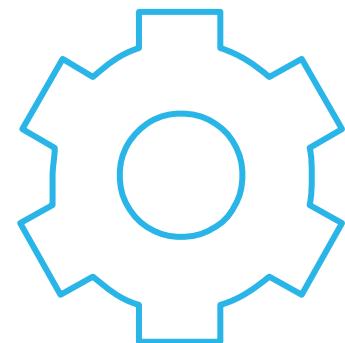
- Can be completed at any time, even when running
- Completed via ALTER WAREHOUSE statement or the UI
- Effects of resizing:
 - Suspended Warehouse: will start at new size upon next resume
 - Running Warehouse: immediate impact; running queries complete at current size, while queued queries run at new size



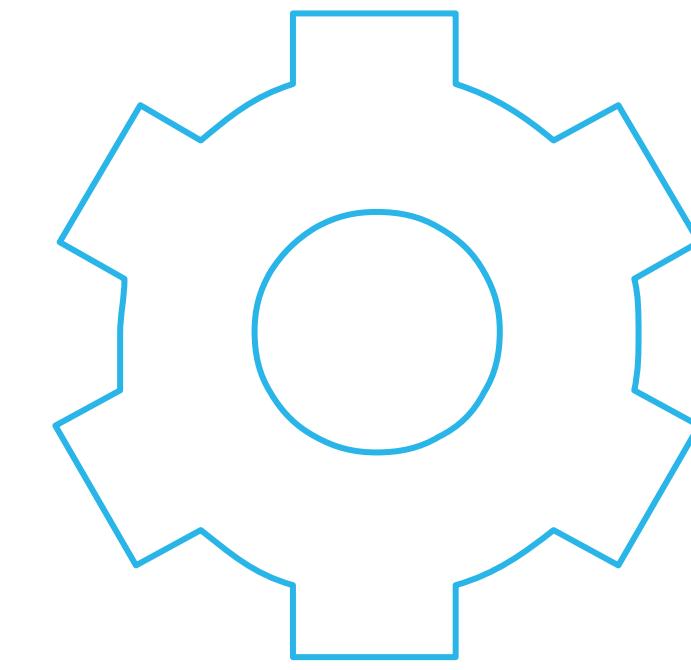
SCALE UP FOR PERFORMANCE

Elastic Processing Power (CPU, RAM, SSD)

- Raw performance boost for complex queries or ingesting large data sets
- Typically more complex queries on larger datasets require larger Warehouses
- Not intended for handling concurrency issues (more users/queries)



Medium



Large

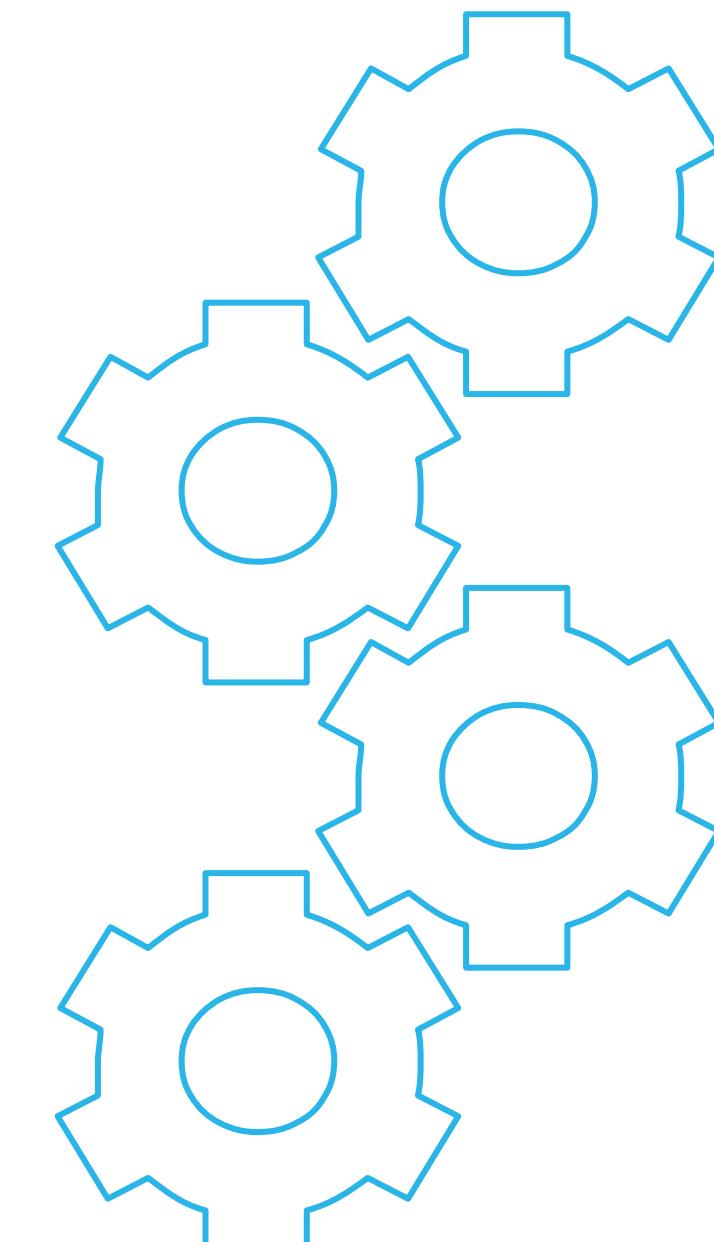
SCALE OUT FOR CONCURRENCY

General Functionality and Considerations

- Single Virtual Warehouse with multiple compute clusters
- Delivers consistent SLA, automatically adding and removing compute clusters based on concurrent usage
- Scale out during peak times and scale back during slow times
- Queries are load balanced across the clusters in a Virtual Warehouse
- Deployed across availability zones for high availability

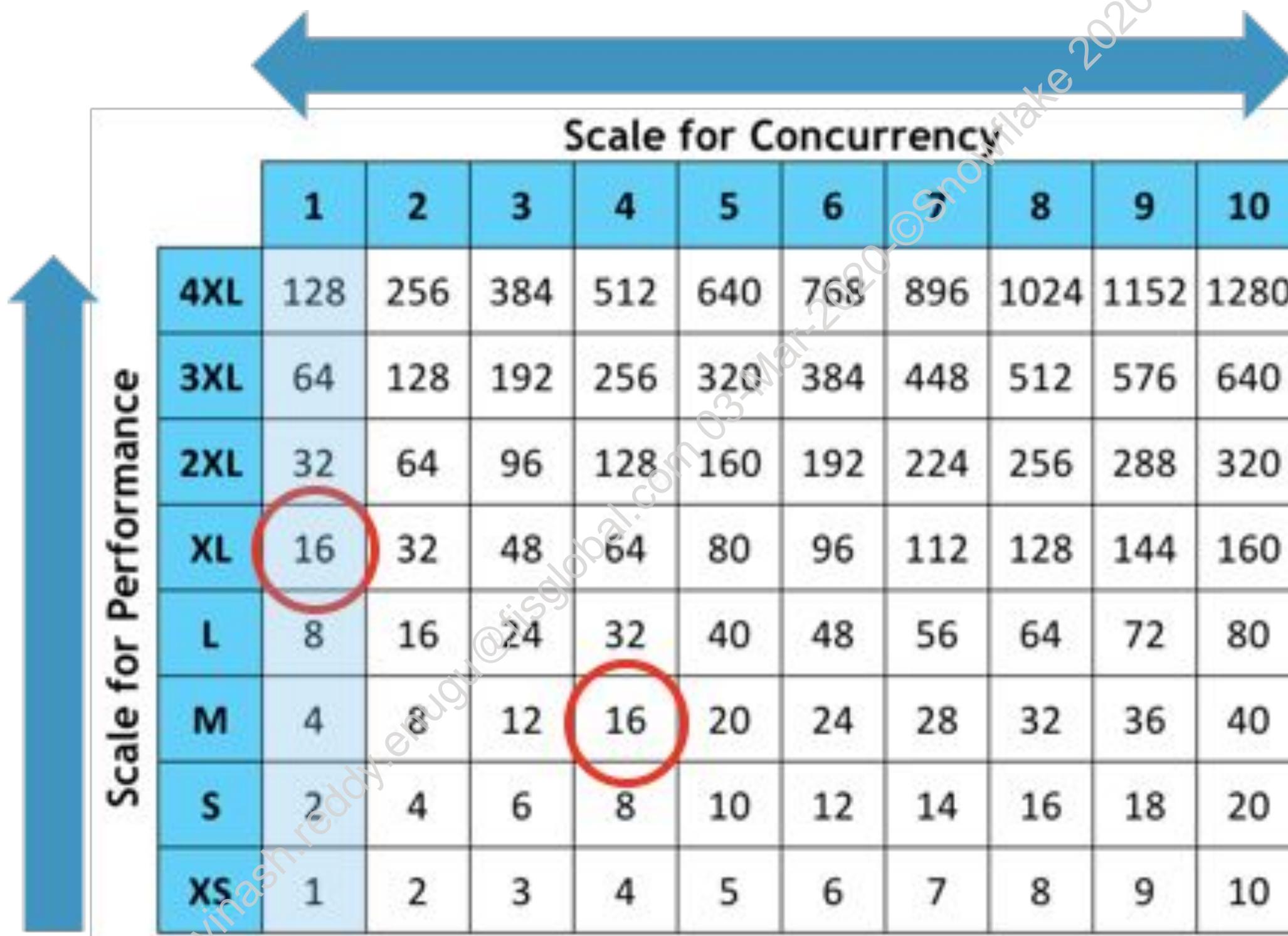
Guidelines

- MIN_CLUSTER_COUNT : 1-10 (default 1)
- MAX_CLUSTER_COUNT: 1-10 (default 1) \geq MIN_CLUSTER_COUNT



SCALING UP VS. OUT EXAMPLE

CREDITS PER HOUR



Module 1: Architecture and Overview

Exercise 1.2: Storage and Compute

25 minutes

Tasks:

- Review the TRAINING_DB database
- Create and organize objects
- Review storage usage
- Run commands without compute cost
- Work with virtual warehouses



Module 2

Clients, Connectors & Ecosystems

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



MODULE AGENDA

- Clients & Interfaces
- SnowSQL
- Ecosystem Overview
- Connectors

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



Clients & Interfaces

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

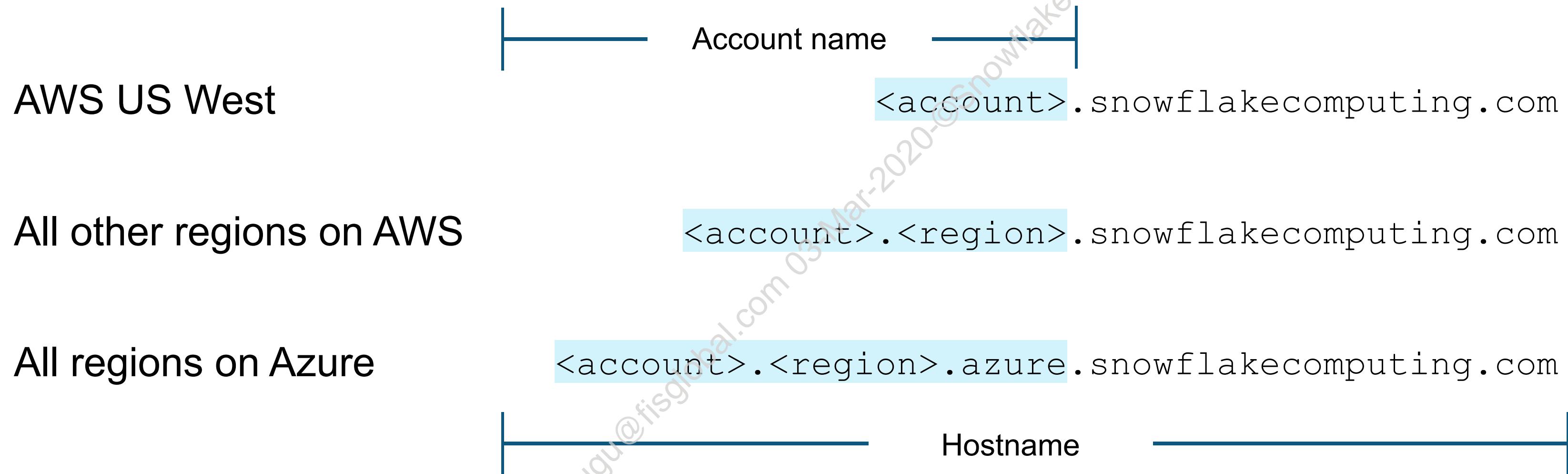


WAYS TO CONNECT & USE SNOWFLAKE

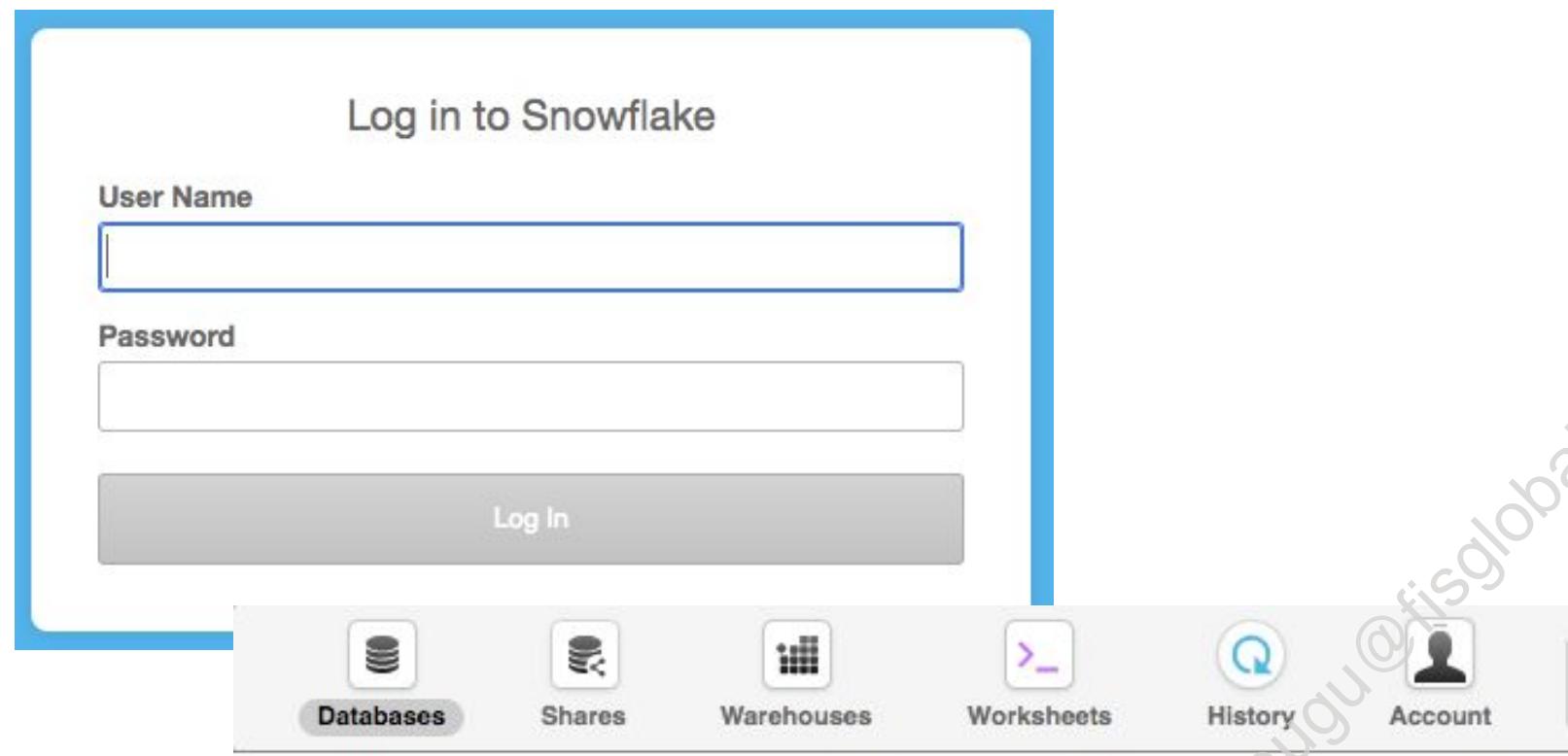
- **Web-based user interface** - access all aspects of using and managing Snowflake
- **Command-line clients** - access all aspects of using and managing Snowflake
 - For example, SnowSQL
- **ODBC and JDBC drivers** - used by other applications to connect to Snowflake
 - For example, Tableau
- **Native connectors** - used to develop applications for connecting to Snowflake
 - For example, Python
- **Third-party solutions** - leverage native connectors to connect to Snowflake
 - For example, ETL and BI tools



SNOWFLAKE WEB INTERFACE URLs



SNOWFLAKE WEB UI



- Account Management
- Virtual Warehouse Management
- Database Management

Simple Data Loading

- Query Execution, Monitoring, & Profiling
- Password Management
- Preference Management



SNOWFLAKE WEB UI: WORKSHEETS

Visual Interface for creating and submitting SQL queries

The screenshot illustrates the Snowflake Web UI Worksheets interface, divided into several panes:

- Object Browser:** Located on the left, it shows a tree view of database objects. A specific node under the SNOWFLAKE_SAMPLE_DATA schema, labeled "Tables", is highlighted with an orange border.
- Worksheet Context:** At the top right, it displays the current context information: Role (TRAINING_ROLE), Database (DRL_MED_WH), Schema (SNOWFLAKE_SAMPLE_DATA), and Warehouse (TPCH_SF1).
- Query Pane:** The central pane contains the SQL query: "SELECT * FROM customer LIMIT 10;".
- Result Pane:** Below the query pane, the results are displayed in a table format. The table has columns: Row, C_CUSTKEY, C_NAME, C_ADDRESS, C_NATIONKEY, C_PHONE, C_ACCTBAL, C_MKTSEG..., and C_COMMENT. The first four rows of the result set are shown:

Row	C_CUSTKEY	C_NAME	C_ADDRESS	C_NATIONKEY	C_PHONE	C_ACCTBAL	C_MKTSEG...	C_COMMENT
1	60001	Customer#00...	9li4zQn9cX			9957.56	HOUSEHOLD	I theodolites b...
2	60002	Customer#00...	ThGBMjDwfk			742.46	BUILDING	beans. fluffy ...
3	60003	Customer#00...	Ed hbPtTXMT...	16	26-859-847-7...	2526.92	BUILDING	fully pending ...
4	60004	Customer#00...	NivCT2RVAav...	10	20-573-674-7...	7975.22	AUTOMOBILE	furiously abov...



WORKSHEET CONTEXT

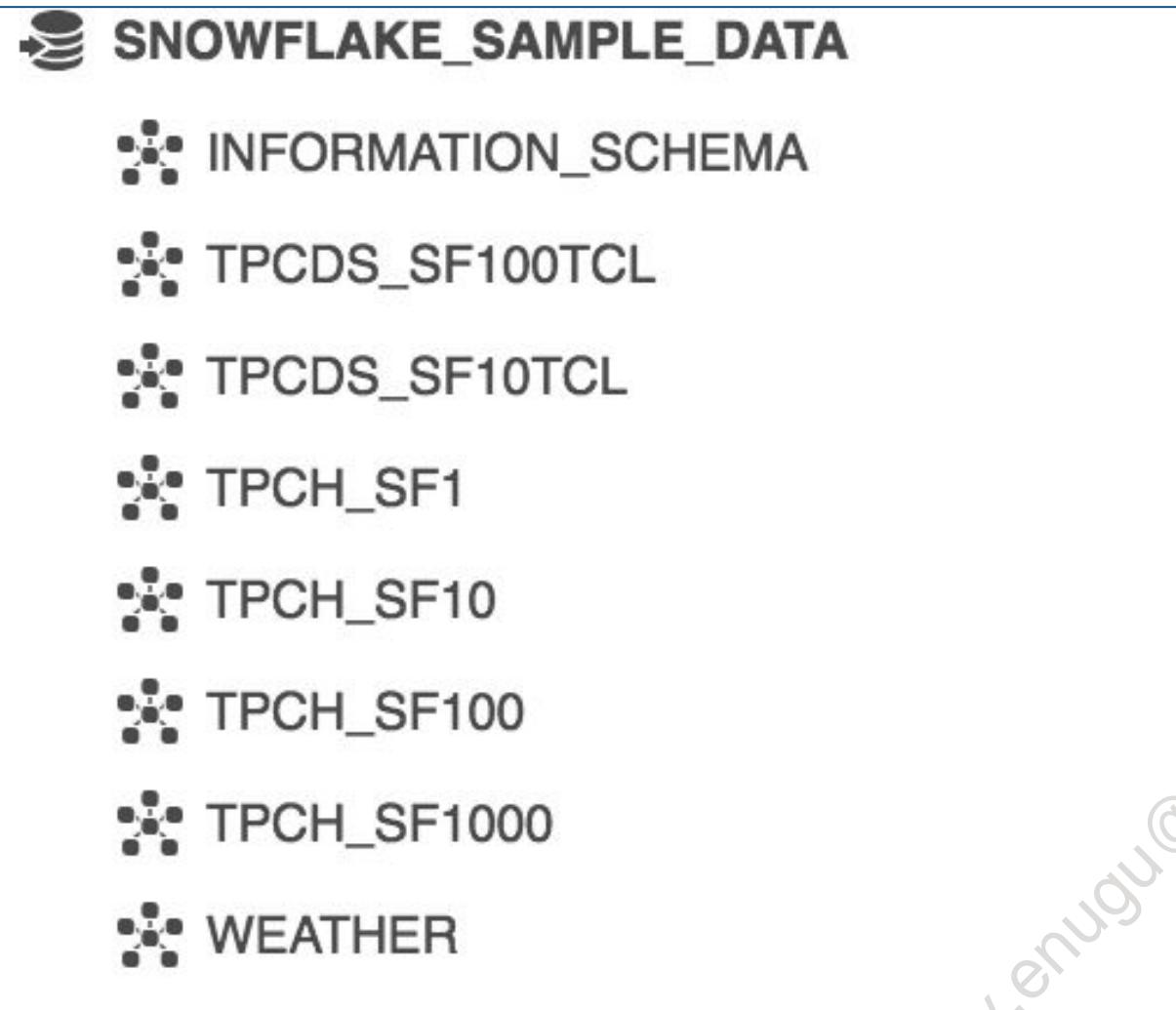
- Defines the default objects for the session
 - Role
 - Warehouse
 - Database
 - Schema



- Can be changed with SQL commands



SNOWFLAKE SAMPLE DATA



- Industry- standard TPC-DS and TPC-H benchmarks
 - Shared via SNOWFLAKE_SAMPLE_DATA database (read only)
- Do not incur storage charges, but do require a Virtual Warehouse to run queries



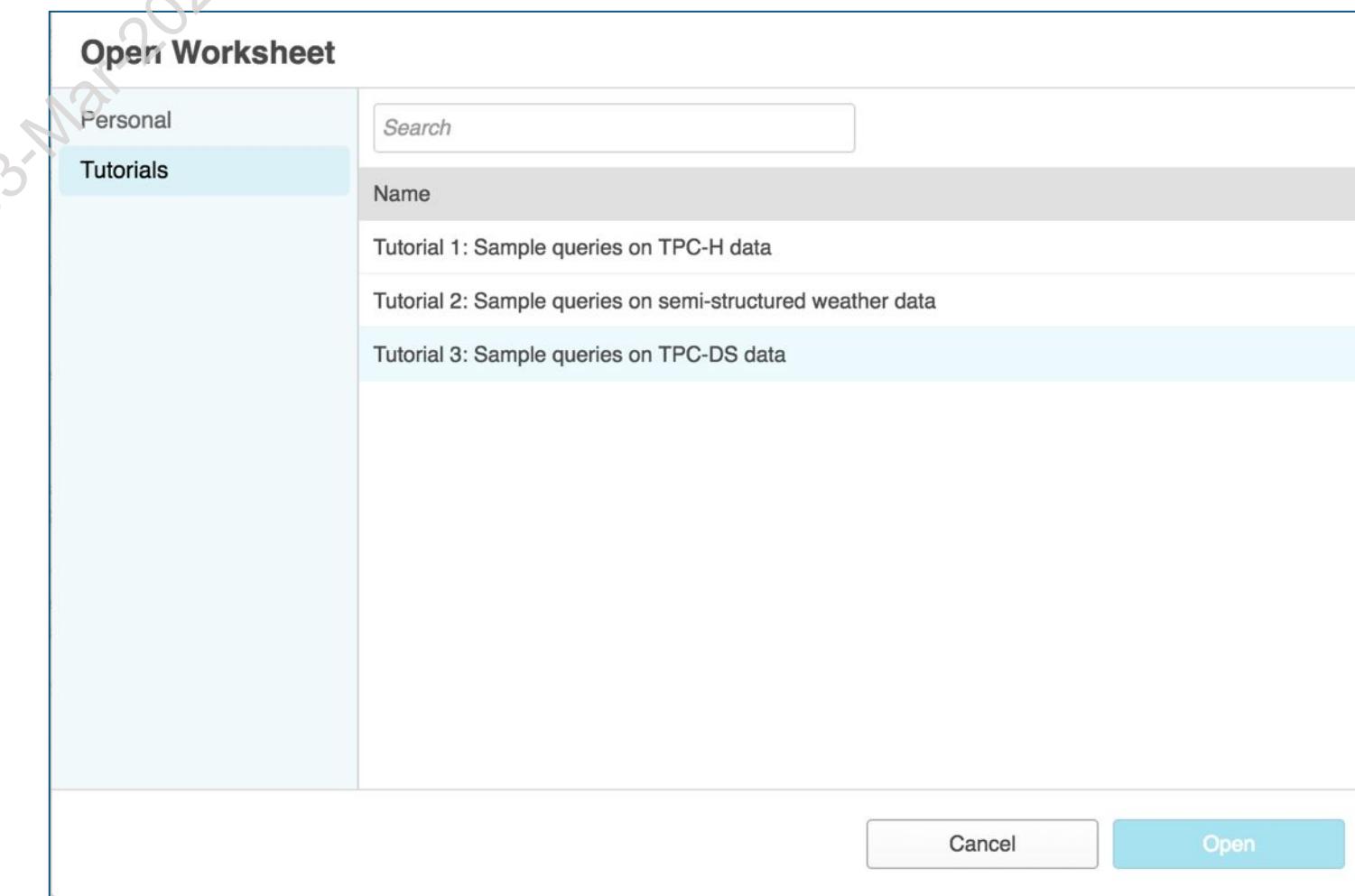
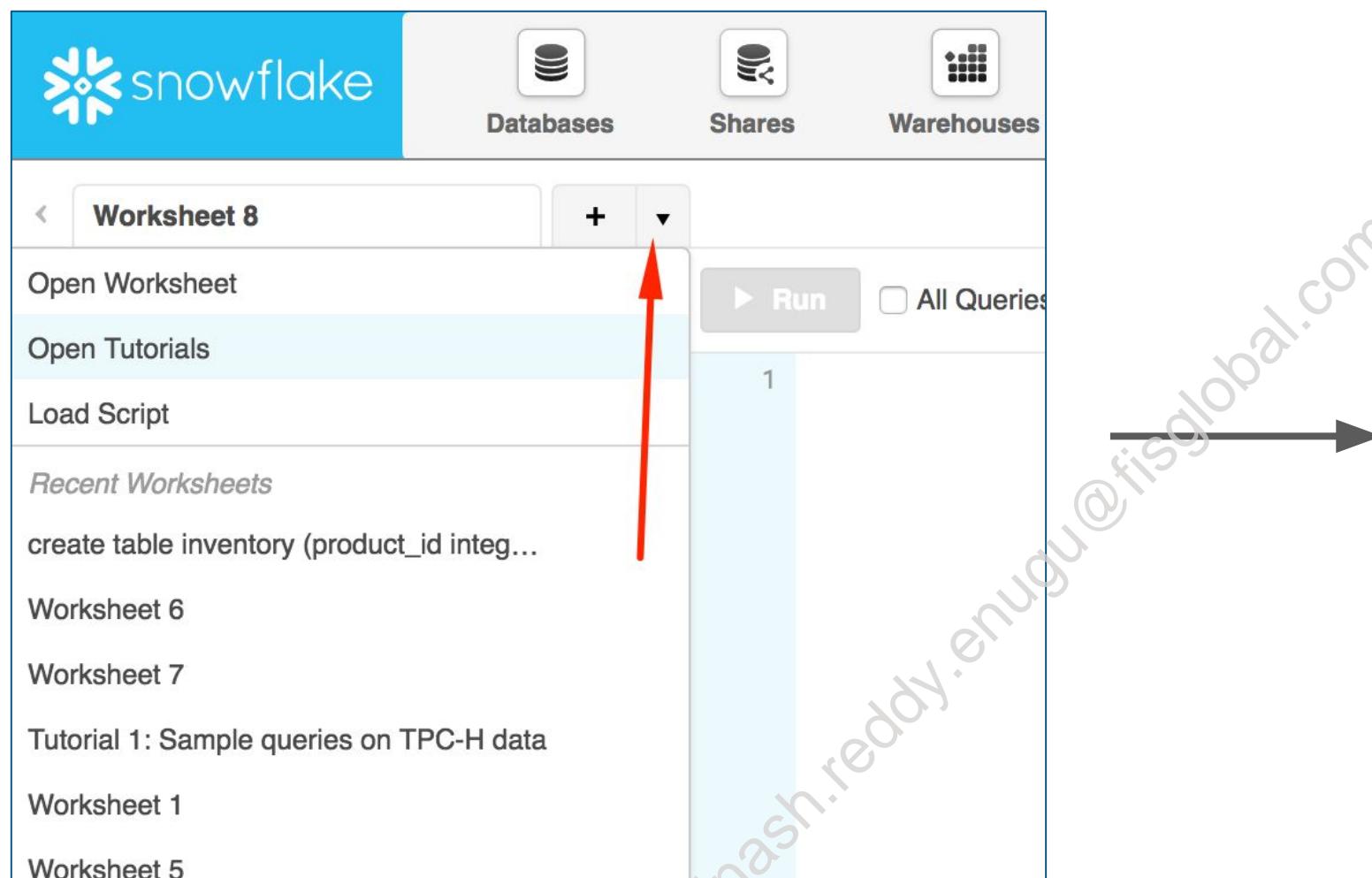
SNOWFLAKE_SAMPLE_DATA

- TPC-H data
 - Order and supplier information for retailer
- TPC-DS data
 - Sales and support information for a catalog company
- Weather Data from OpenWeatherMap
 - JSON format
 - Daily, hourly, and recent weather for 200,000+ cities
 - July 2016 through present



TUTORIALS

- Available through the UI
- Guided walk-throughs and exercises atop the SNOWFLAKE_SAMPLE_DATA
- Additional tutorials are detailed in the documentation



SnowSQL

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



SNOWSQL

- Command-line client for connecting to Snowflake
- Developed using the Snowflake connector for Python
- Versions for Windows, MacOS, Linux



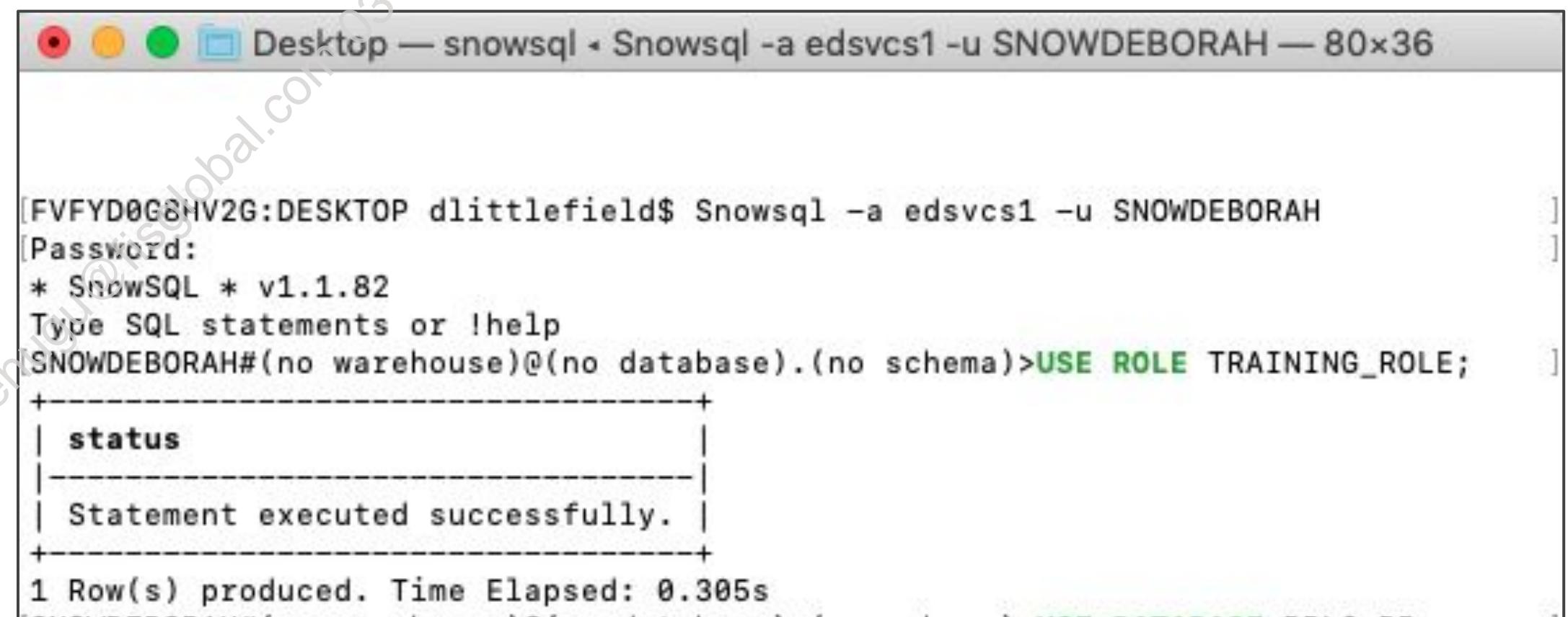
The screenshot shows a terminal window titled "Desktop — snowsql". The command entered is "Snowsql -a edsvcs1 -u SNOWDEBORAH". The output shows the password prompt, the version ("* SnowSQL * v1.1.82"), the SQL prompt ("Type SQL statements or !help"), and the result of executing the "USE ROLE TRAINING_ROLE;" command. The output indicates that the statement was executed successfully and produced one row.

```
[FVFYD0G8HV2G:DESKTOP dlittlefield$ Snowsql -a edsvcs1 -u SNOWDEBORAH
>Password:
* SnowSQL * v1.1.82
Type SQL statements or !help
[SNOWDEBORAH#(no warehouse)@(no database).(no schema)]>USE ROLE TRAINING_ROLE;
+-----+
| status
+-----+
| Statement executed successfully.
+-----+
1 Row(s) produced. Time Elapsed: 0.305s
```



SNOWSQL

- Available for download from the UI: Help -> Downloads
- Run as an interactive shell, or in batch mode
- Can leverage a configuration file to pass parameters like account, user, role, database, schema, etc.



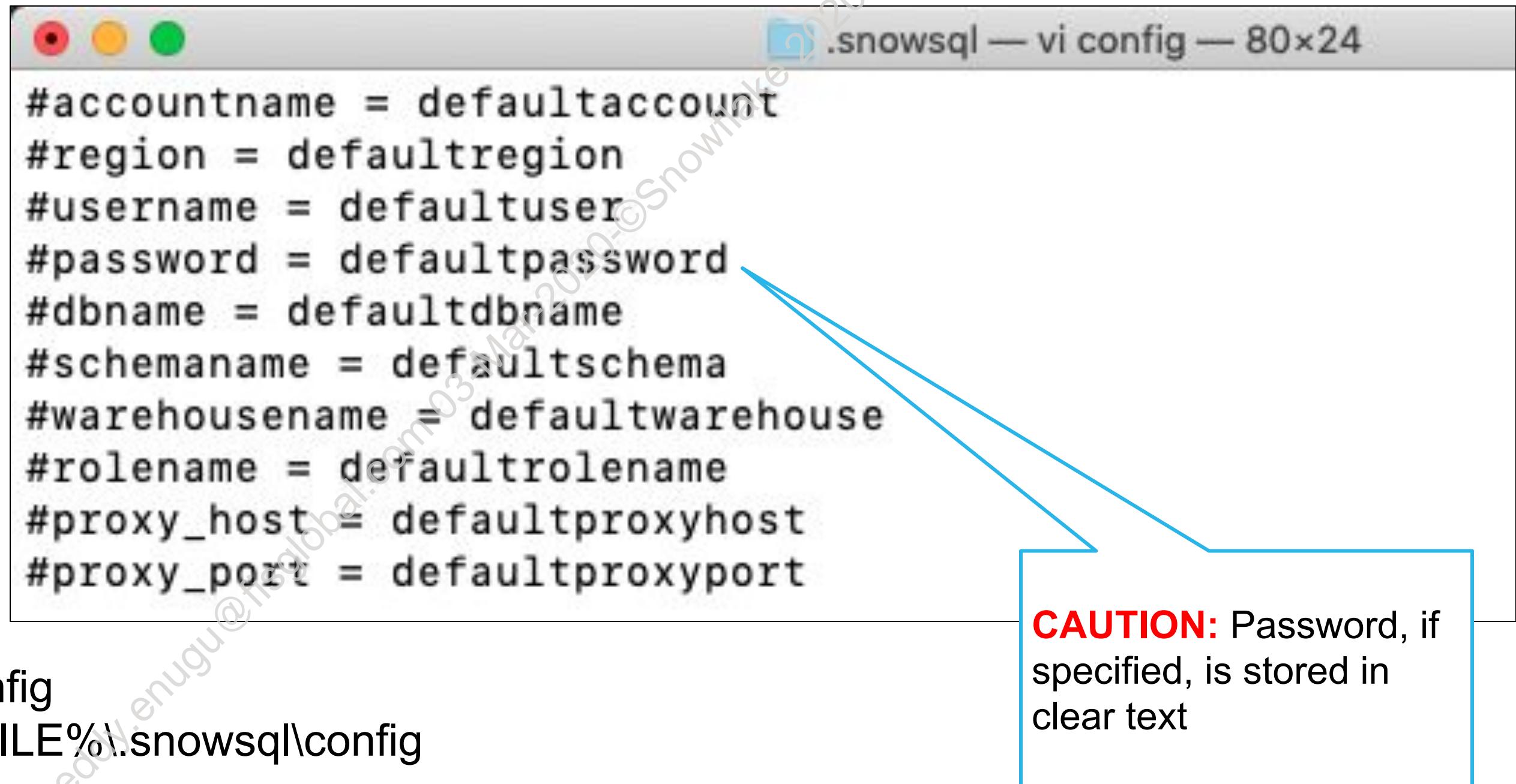
```
[FVFYD0G8HV2G:DESKTOP dlittlefield$ Snowsql -a edsvcs1 -u SNOWDEBORAH
[Password:
* SnowSQL * v1.1.82
Type SQL statements or !help
SNOWDEBORAH#(no warehouse)@(no database).(no schema)>USE ROLE TRAINING_ROLE;
+-----+
| status
+-----+
| Statement executed successfully.
+-----+
1 Row(s) produced. Time Elapsed: 0.305s
```

SNOWSQL CONFIGURATION FILE

- Set defaults to be used in SnowSQL
- Can set up multiple connectors

Locations:

- Linux/Mac: `~/.snowsql/config`
- Windows: `%USERPROFILE%\.snowsql\config`



```
#accountname = defaultaccount
#region = defaultregion
#username = defaultuser
#password = defaultpassword
#dbname = defaultdbname
#schemaname = defaultschema
#warehousename = defaultwarehouse
#rolename = defaultrolename
#proxy_host = defaultproxyhost
#proxy_port = defaultproxyport
```

CAUTION: Password, if specified, is stored in clear text



SNOWSQL CONFIGURATION FILE

- Configure multiple connections

The screenshot shows a terminal window titled "dlittlefield — vi .snowsql/config". The configuration file contains the following content:

```
[connections.edsvcs3]
accountname = edsvcs3
username = SNOWDRL
dbname = SNOWFLAKE_SAMPLE_DATA
schemaname = WEATHER
warehousename = DRL_WH
rolename = TRAINING_ROLE
```

Two sections of the configuration file are highlighted with red boxes: the connection section [connections.edsvcs3] and the individual connection parameters (username, dbname, schemaname, warehousename, rolename).

- Connect with -c option

The screenshot shows a terminal window titled "dlittlefield — snowsql -c edsvcs3". The session output is as follows:

```
[FVFYD0G8HV2G:~ dlittlefield$ snowsql -c edsvcs3
[Password:
* SnowSQL * v1.1.84
Type SQL statements or !help
SNOWDRL#DRL_WH@SNOWFLAKE_SAMPLE_DATA.WEATHER>
```

The connection information (host, port, database, schema, warehouse) is highlighted with a red box, and the resulting session prompt is also highlighted with a red box.



CHECK CURRENT ROLE IN SNOWSQL

```
dlittlefield — snowsql + snowsql -c edsvcs3 — 80x24
[FVFYD0G8HV2G:~ dlittlefield$ snowsql -c edsvcs3
[Password:
* SnowSQL * v1.1.84
Type SQL statements or !help
[SNOWDRL#(no warehouse)@SNOWFLAKE_SAMPLE_DATA.WEATHER>SELECT CURRENT_ROLE();
+-----+
| CURRENT_ROLE() |
|-----|
| TRAINING_ROLE   |
+-----+
1 Row(s) produced. Time Elapsed: 0.125s
SNOWDRL#(no warehouse)@SNOWFLAKE_SAMPLE_DATA.WEATHER>
```



Module 2: Clients and Connectors

Exercise 2.1: Work with SnowSQL

35 minutes

Note:

- This lab requires the ability to install software on your laptop

Tasks:

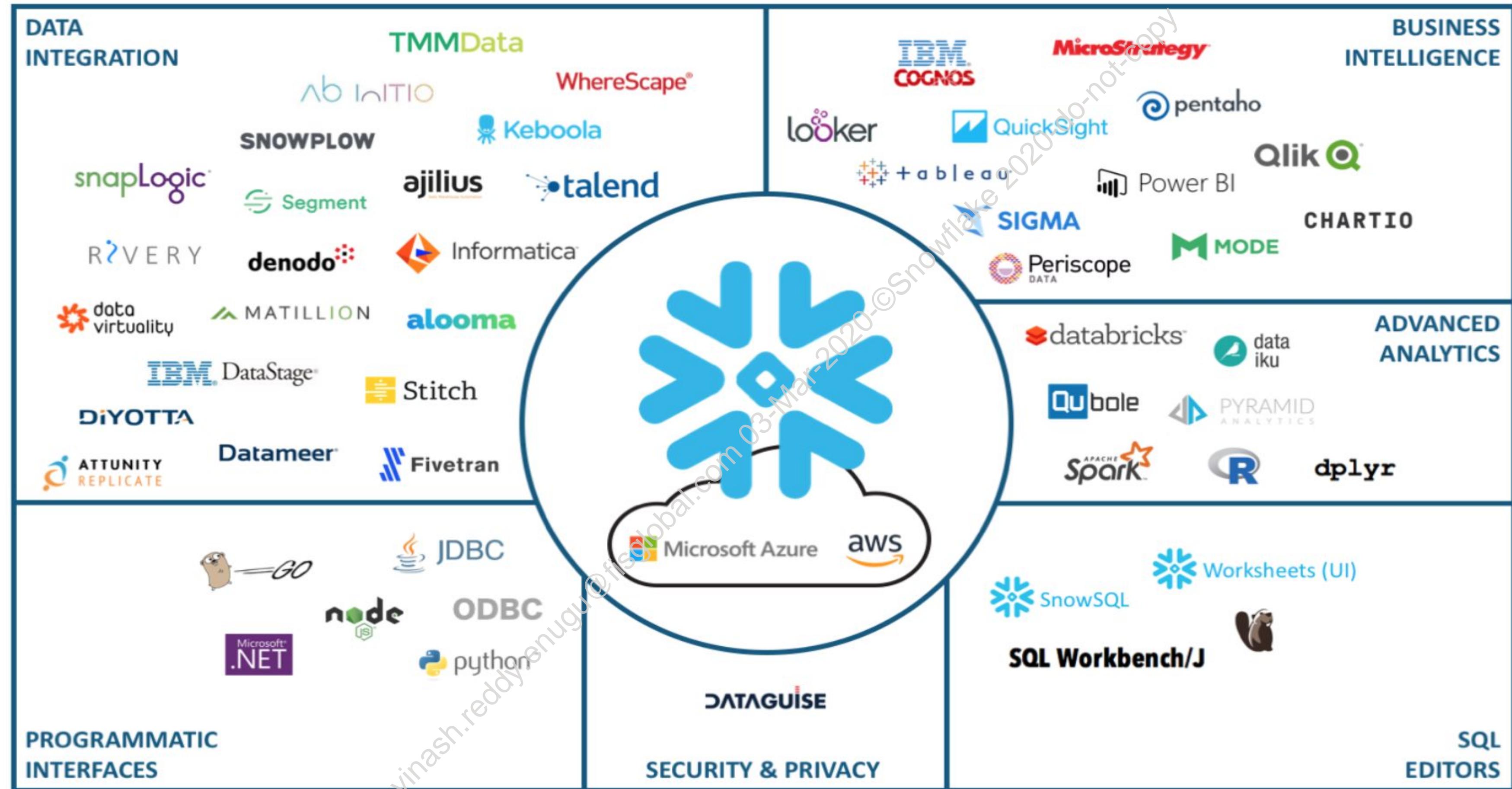
- Install SnowSQL
- Run SQL Commands using SnowSQL
- Create & use a configuration file
- Run a script using SnowSQL



Ecosystem Overview

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy





PARTNER CONNECT

- Create trial account with selected Snowflake partners
- Use limited to ACCOUNTADMIN
- During connection, partner creates:
 - PC_%partnername_USER
 - PC_%partnername_ROLE
 - PC_%partnername_DB
 - PC_%partnername_WH

The screenshot shows the Snowflake Partner Connect interface. At the top, there's a navigation bar with icons for Databases, Shares, Warehouses, Worksheets, History, Account, Partner Connect (which is highlighted), Help, and user DBRYANT SYSADMIN. Below the navigation bar, the title "Snowflake Partner Connect" is displayed, followed by a sub-header "Get started with loading and analyzing your data in minutes. Automatically connect your Snowflake account with our partner applications available for a free trial." A note says "Check back often as we will be adding new partners regularly." Below this, five partner applications are listed in boxes: Fivetran, Alooma, Stitch, Sigma, and Periscope Data. Each box contains the partner's logo, name, and a brief description.

Partner	Description
Fivetran	Built for analysts, 5-minute setup, great schemas, Snowflake platinum partner.
Alooma	Connect all of your data with Alooma, the enterprise data pipeline built for the cloud
Stitch	Stitch moves data into Snowflake in minutes. Unlimited sources and a free-forever tier.
Sigma	A spreadsheet UI for Snowflake. Easily explore and analyze all your data.
Periscope Data	Periscope Data brings data science and advanced analytics to the world of BI.

Module 2: Clients, Connectors, & Ecosystems

DEMO: Partner Connect

5 minutes



Connectors

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

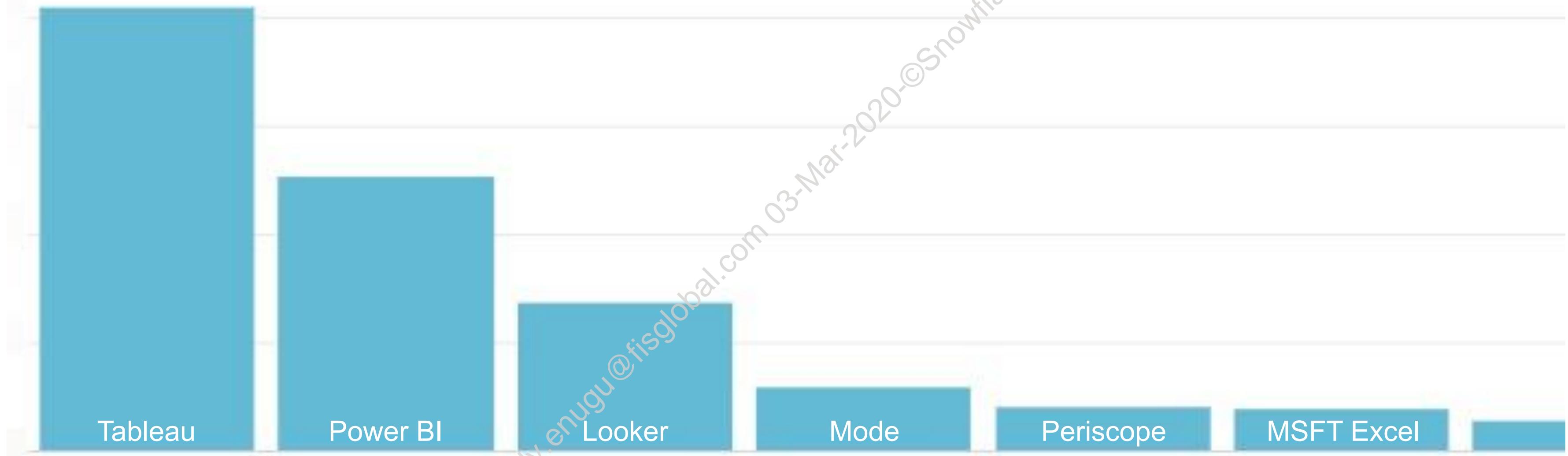


SUPPORTED CONNECTORS

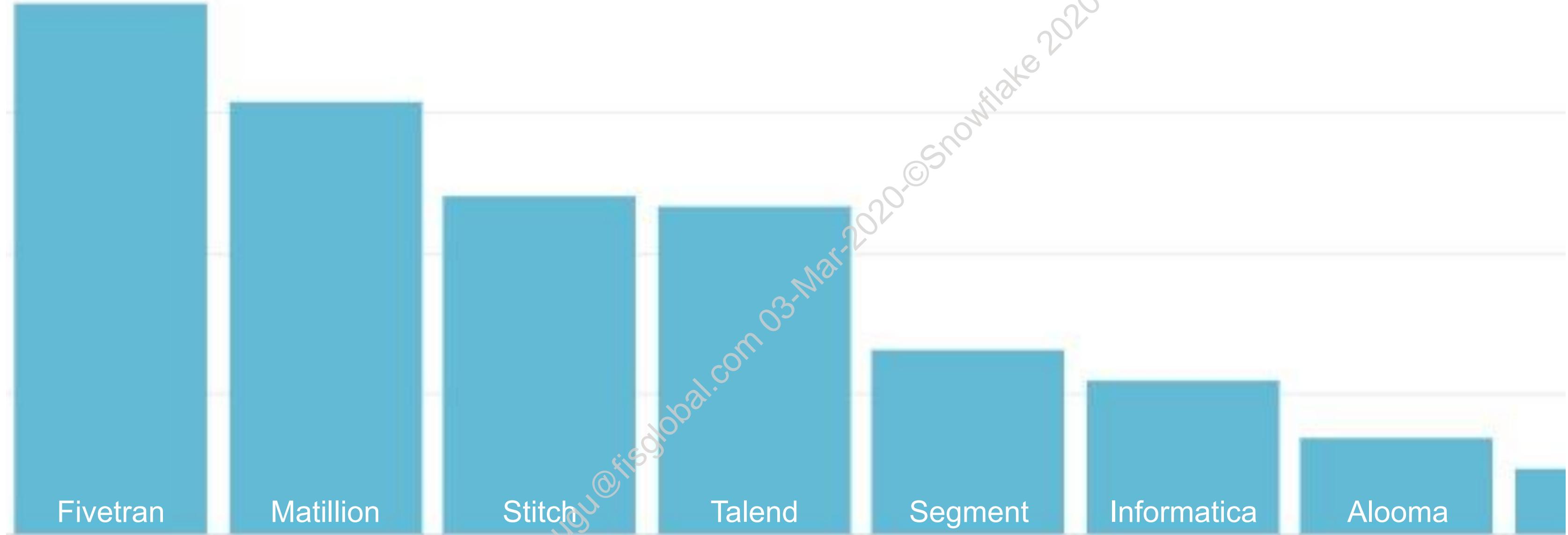
JDBC	Java
ODBC	C/C++
Python	Python
Go	Go
Node.js	Node.js
Spark	Spark Dataframe (Scala, Python)
R	Java, C, C++ (RJDBC, RODBC)
.NET	Visual Basic, C#, F#, C++
SnowSQL	CLI client for SQL (Python)
Web Interface	UI for Admin, SQL, and more



BI TOOL USAGE



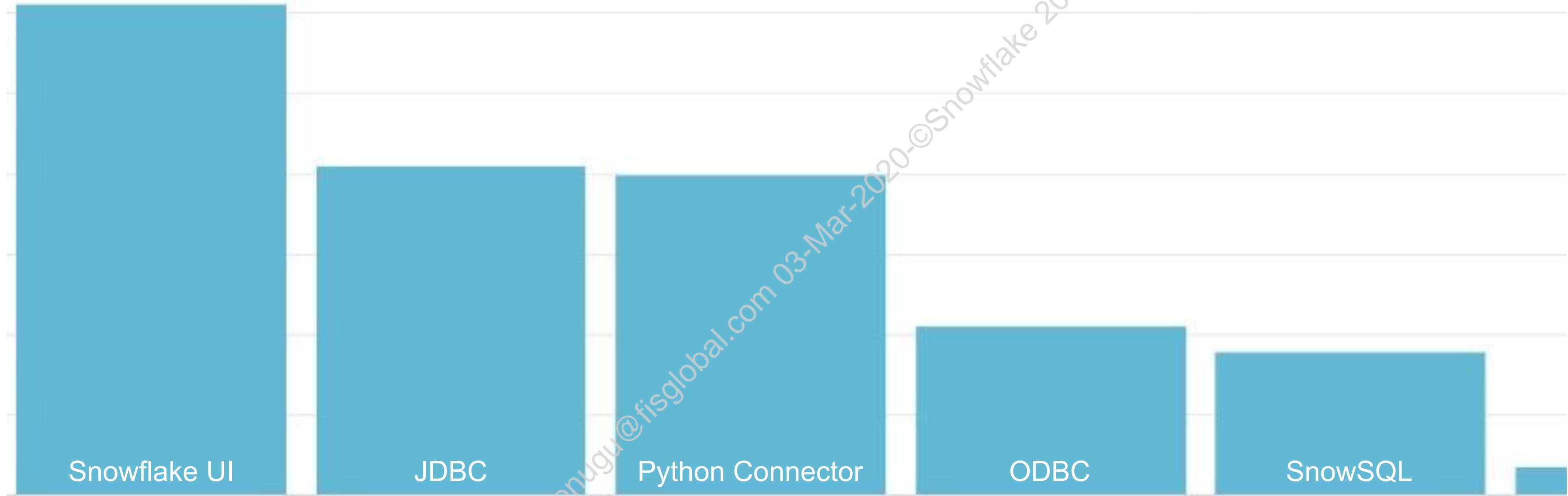
ETL TOOL USAGE



ANALYTICS TOOLS USAGE



WORKLOAD BY CONNECTING CLIENT



Module 3

SQL Support in Snowflake

avinash.reddy.enugu@fisglobal.com 03-May-2020 ©Snowflake 2020-do-not-copy



MODULE AGENDA

- Data Description Language (DDL)
- Data Manipulation Language (DML)
- Querying and Filtering
- Collations
- Subqueries
- Query Profile



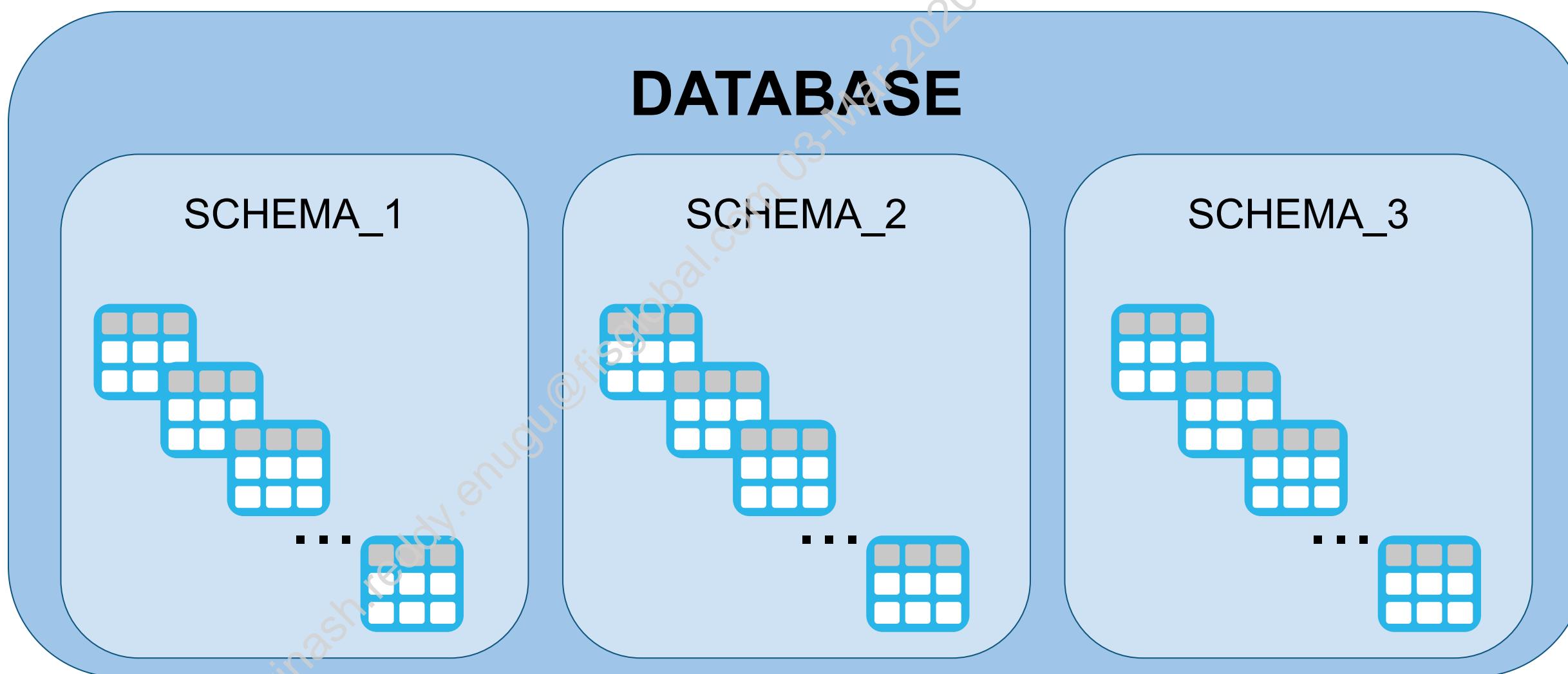
Data Description Language (DDL)

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



DATABASE VS SCHEMA

- Database - logical grouping of schemas
- Schema - logical grouping of objects (tables, views, functions, procedures)
- Database name must be unique in Snowflake account



MANAGE DATABASE OBJECTS

- Database Management

CREATE/ALTER/DROP/**UNDROP**/USE DATABASE
SHOW DATABASES

- Schema Management

CREATE/ALTER/DROP/**UNDROP**/USE SCHEMA
SHOW SCHEMAS

- **Share Management**

CREATE/ALTER/DROP/DESCRIBE **SHARE**
SHOW SHARES



CREATE AND MANAGE TABLES

- CREATE OR REPLACE TABLE
- CREATE TABLE IF NOT EXISTS
- CREATE TABLE ... AS SELECT
- CREATE TABLE ... LIKE (empty copy of existing table)
- CREATE TABLE ... **CLONE**
- ALTER/DROP TABLE
- SHOW TABLES
- CREATE TEMPORARY/ **TRANSIENT** TABLE



OBJECT IDENTIFIER CASE SENSITIVITY

Table Creation Statement

```
CREATE TABLE My_Table
```

Resulting Table Name

MY_TABLE

```
CREATE TABLE my_table
```

MY_TABLE

VS.

Table Create Statement

```
CREATE TABLE "My_Table"
```

Resulting Table Name

My_Table

```
CREATE TABLE "my_table"
```

my_table



DDL SUPPORT - EXAMPLES

```
CREATE TABLE list(  
    id INTEGER,  
    region STRING,  
    item STRING);
```

Table Name = list

ID	REGION	ITEM

Column accepts : INTEGER STRING STRING



DDL SUPPORT - EXAMPLES

Table Name = list

ID	REGION	ITEM
1	East	Binder
2	West	Pen
3	East	Pencil
4	Central	Chair
5	West	Laptop

```
CREATE TABLE eastlist AS  
SELECT region, item  
FROM list  
WHERE region = 'East';
```

Table Name = **eastlist**

REGION	ITEM
East	Binder
East	Pencil

DDL SUPPORT - EXAMPLES

Table Name = list

ID	REGION	ITEM

INTEGER STRING STRING

```
ALTER TABLE list  
    RENAME COLUMN item TO cur_date;
```

```
ALTER TABLE list  
    ALTER COLUMN cur_date TYPE DATE;
```

Table Name = list

ID	REGION	CUR_DATE

INTEGER STRING **DATE**

MANAGE VIEWS AND SEQUENCES

- View Management
 - CREATE OR REPLACE VIEW
 - CREATE VIEW IF NOT EXISTS
 - CREATE OR REPLACE **SECURE VIEW**
- Materialized View Management
 - CREATE OR REPLACE **MATERIALIZED VIEW**
 - CREATE MATERIALIZED VIEW IF NOT EXISTS
 - CLUSTER BY (<expr>)
- Sequence Management
 - CREATE OR REPLACE SEQUENCE
 - CREATE SEQUENCE IF NOT EXISTS
 - START WITH/INCREMENT BY



EXTENDED DDL SUPPORT

- Stage Management

CREATE/ALTER/DROP/DESCRIBE **STAGE**
SHOW STAGES

- File Format Management

CREATE/ALTER/DROP/DESCRIBE **FILE FORMAT**
SHOW FILE FORMATS

- Pipe Management

CREATE/ALTER/DROP/DESCRIBE **PIPE**
SHOW PIPES



EXTENDED DDL SUPPORT - EXAMPLES

```
CREATE FILE FORMAT mycsvformat
  type = 'CSV'
  field_delimiter = ' | '
  skip_header = 1;
```

```
CREATE STAGE my_stage
  FILE_FORMAT = (FORMAT_NAME = mycsvformat);
```



Module 3: Query Data in Snowflake

Exercise 3.1: Work With Database Objects

45 minutes

Tasks:

- Work with permanent, temporary, and transient tables
- Work with views
- Work with sequences



Data Manipulation Language (DML)

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



STANDARD DML

Commands for inserting, deleting, updating and merging data

- INSERT
- MERGE
- UPDATE
- DELETE
- TRUNCATE



EXTENDED DML

File Staging Commands

- PUT (to a stage)
- GET (from a stage)
- LIST
- REMOVE

Data Loading/Unloading DML

- COPY INTO <table>
- COPY INTO <location>
- VALIDATE



EXTENDED DML - EXAMPLE

```
PUT file:///tmp/load/contacts*.csv @my_stage AUTO_COMPRESS=false;  
  
COPY INTO mytable  
FROM @my_stage/contacts1.csv.gz  
FILE_FORMAT = (format_name = mycsvformat)  
ON_ERROR = 'skip_file';
```



EXTENDED DML - EXAMPLE

```
COPY INTO my_table
FROM @my_stage
FILE_FORMAT = (FORMAT_NAME = mycsvformat)
pattern='.*contacts[1-5].csv.gz'
ON_ERROR = 'skip_file';
```

```
CREATE TABLE save_copy_errors AS
SELECT * FROM TABLE(VALIDATE(my_table, job_id=>'<query_id>>'));
```



Querying and Filtering

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



QUERY FORMULATION

Component	What it does
SELECT	Specifies which columns/aggregates/scalar transforms to return
FROM	Defines the data set (table or query) to work with
DISTINCT	Returns only unique values
WHERE	Filters values returned by the FROM clause
LIMIT	Limits the number of records in the returned result set
GROUP BY	Defines how data should be grouped in the results
HAVING	Specifies conditions related to the grouped data
ORDER BY	Specifies how the rows should be ordered
JOIN	Joins multiple tables based on common columns
PIVOT	Rotates a table (turns unique values in a column, into columns)



SELECT

Specifies which columns, aggregates, or scalar transforms to return

Samples:

```
SELECT * FROM mytable;
```

```
SELECT col3, col2 FROM mytable;
```

```
SELECT col1, SUM(col3) FROM mytable GROUP BY col1;
```

```
SELECT col1 AS price, CAST(col3 AS DECIMAL(10,2)) FROM mytable;
```

```
SELECT CURRENT_ROLE();
```

```
SELECT (1+1);
```



FROM

Table, view, or table function to use in a SELECT statement

Samples:

```
SELECT * FROM mytable;
```

```
SELECT * FROM myview;
```

```
SELECT a.col1, b.col2  
FROM lefhtable a JOIN righttable b ON a.id = b.empid;
```

```
SELECT *  
FROM TABLE(INFORMATION_SCHEMA.login_history_by_user());
```



DISTINCT

Returns only unique values

Samples:

```
SELECT DISTINCT (col15) FROM mytable;
```

```
SELECT COUNT (DISTINCT col15) FROM mytable;
```



WHERE

Filters the result of the FROM clause, using a *predicate*

Samples:

```
SELECT * FROM invoices WHERE invoice_date < '2018-01-01';
```

```
SELECT * FROM invoices  
WHERE amount < (SELECT AVG(amount) FROM invoices);
```

```
SELECT * FROM invoices  
WHERE invoice_date < DATEADD('DAYS', -30, CURRENT_DATE())  
AND paid=FALSE;
```



LIMIT

ID	NAME	SALARY
1	Joe	53000
2	Rajesh	67000
3	Saira	125000
4	Jenn	98750
...
12390	Anis	45890

```
SELECT * FROM employees  
LIMIT 1;
```

ID	NAME	SALARY
287	June	69340

- Limits the number of rows returned to the value specified
- Does not just return the first n rows (unless the query uses ORDER BY)



GROUP BY

ITEM_ID	ITEM_TYPE	QTY
1	Pen	57
2	Chair	3
3	Chair	7
4	Table	5
5	Pen	245

```
SELECT item_type, SUM(qty)
FROM inventory
GROUP BY item_type;
```



ITEM_TYPE	QTY
Chair	10
Pen	302
Table	5

- Defines how data should be grouped in the results
- Groups rows that have the same values in a specified column
- Computes aggregate functions for the resulting group if needed



HAVING

Specifies conditions related to grouped data

Samples:

```
SELECT dept FROM employees  
GROUP BY dept HAVING COUNT(*) < 10;
```

```
SELECT item, SUM(qty) FROM orders  
GROUP BY item HAVING SUM(qty) < 100;
```



ORDER BY

MFG	ITEM_TYPE	QTY
OfficeMax	Pen	57
OfficeGuys	Chair	3
Staples	Chair	7
Bob's	Table	5
Staples	Pen	245

```
SELECT item_type, mfg, qty  
FROM inventory ORDER BY item_type;
```

ITEM_TYPE	MFG	QTY
Chair	OfficeGuys	3
Chair	Staples	7
Pen	OfficeMax	57
Pen	Staples	245
Table	Bob's	5

- Orders values in ascending or descending order on a specified column
- Can have multi-level ORDER BY
 - By default, sorts in ascending (ASC) order
 - Use DESC to sort in descending order



JOIN

Combines rows from two tables/views/functions based on common columns

Samples:

```
SELECT lefhtable.col1, righttable.col2  
FROM lefhtable JOIN righttable  
ON lefhtable.last = righttable.last_name;
```

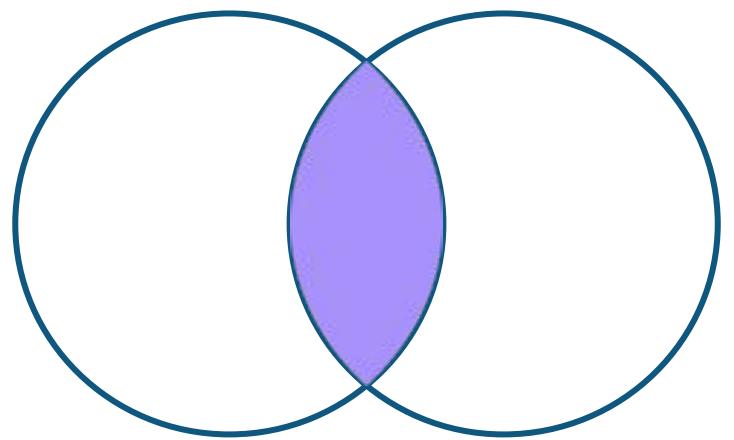
```
SELECT * FROM lefhtable a JOIN righttable b ON a.key = b.key;
```

```
SELECT t1.* , t2.* , t3.* FROM t1  
LEFT OUTER JOIN t2 ON (t1.key = t2.key)  
RIGHT OUTER JOIN t3 ON (t3.id = t2.empid);
```

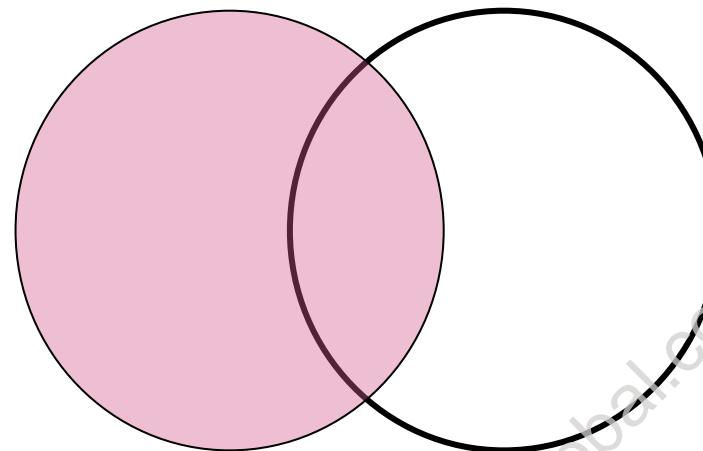


SOME JOIN TYPES

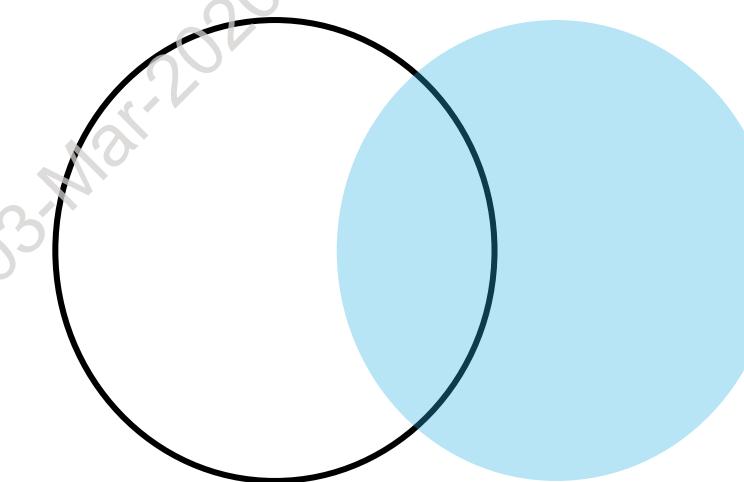
Inner Join (Join)



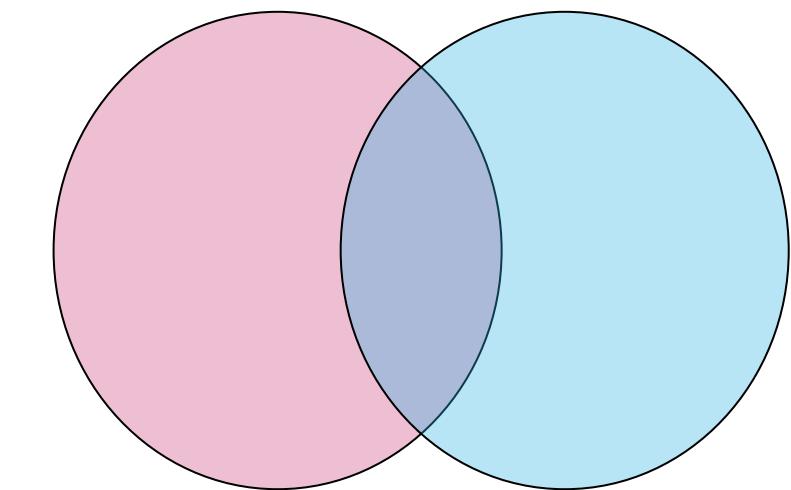
Left Join



Right Join



Full Outer Join



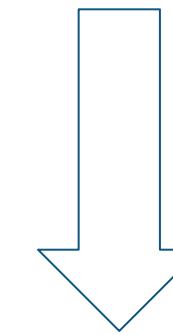
PIVOT

Turns unique values in a column into multiple columns

EmpID	Sales	Month
1	10000	JAN
1	8000	JAN
2	12000	JAN
2	4000	JAN
1	6000	FEB
1	13000	FEB
2	15000	FEB
1	10000	MAR
2	3000	MAR
1	25000	APR
2	31000	APR



```
SELECT * FROM monthly_sales  
PIVOT (SUM(sales) for month  
IN ('JAN', 'FEB', 'MAR', 'APR'))  
ORDER BY empid;
```



EmpID	JAN	FEB	MAR	APR
1	18000	19000	10000	25000
2	16000	15000	3000	31000



Collations

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



COLLATION

- **Collate:** to collect and combine in proper order
 - Proper order is open to interpretation
- With Snowflake, strings are stored internally in UTF-8
 - UTF-8 uses the numeric representation of characters to sort
 - Upper-case letter have a lower numeric representation than lower-case

Example:

Anne, Bob, Jerry, charles, deborah, klaus



SUPPORTED COLLATIONS

SPECIFY HOW TEXT SHOULD BE SORTED/COMPARED

Collation	Use	Examples
Language locale	Language- and country-specific rules to apply	en, en_US, fr, fr_CA
Case sensitivity	Whether to consider case when comparing values	cs or ci
Accent sensitivity	Whether to consider accented characters equal to, or different from, their base characters	as or ai
Punctuation sensitivity	Whether non-letter characters matter	ps or pi
First letter preference	Whether to sort upper-case or lower-case letters first	f1 or fu
Case conversion	Convert to upper or lower case before comparisons	upper or lower
Space trimming	Remove leading spaces, trailing spaces, or both	trim, ltrim, rtrim



SPECIFYING COLLATIONS ON COLUMNS

```
CREATE TABLE test (utf8 VARCHAR, french VARCHAR collate 'fr', spanish VARCHAR collate 'es');
```

```
INSERT INTO test VALUES
```

```
('Apple', 'Apple', 'Apple'),  
('apple', 'apple', 'apple'),  
('AB CD', 'AB CD', 'AB CD'),  
('ABC', 'ABC', 'ABC'),  
('pino', 'pino', 'pino'),  
('piñata', 'piñata', 'piñata'),  
('ab\'cd', 'ab\'cd', 'ab\'cd'),  
('abc', 'abc', 'abc');
```

```
SELECT utf8 FROM test;
```

unsorted
UTF8
Apple
apple
AB CD
ABC
pino
piñata
ab'cd
abc



SPECIFYING COLLATIONS ON COLUMNS

```
CREATE TABLE test (utf8 VARCHAR, french VARCHAR collate 'fr', spanish VARCHAR collate 'es');
```

```
INSERT INTO test VALUES
```

```
('Apple', 'Apple', 'Apple'),  
('apple', 'apple', 'apple'),  
('AB CD', 'AB CD', 'AB CD'),  
('ABC', 'ABC', 'ABC'),  
('pino', 'pino', 'pino'),  
('piñata', 'piñata', 'piñata'),  
('ab\'cd', 'ab\'cd', 'ab\'cd'),  
('abc', 'abc', 'abc');
```

```
SELECT utf8 FROM test ORDER BY utf8;
```

unsorted	sorted by utf8
UTF8	UTF8
Apple	AB CD
apple	ABC
AB CD	Apple
ABC	ab'cd
pino	abc
piñata	apple
ab'cd	pino
abc	piñata



SPECIFYING COLLATIONS ON COLUMNS

```
CREATE TABLE test (utf8 VARCHAR, french VARCHAR collate 'fr', spanish VARCHAR collate 'es');
```

```
INSERT INTO test VALUES
```

```
('Apple', 'Apple', 'Apple'),  
('apple', 'apple', 'apple'),  
('AB CD', 'AB CD', 'AB CD'),  
('ABC', 'ABC', 'ABC'),  
('pino', 'pino', 'pino'),  
('piñata', 'piñata', 'piñata'),  
('ab\'cd', 'ab\'cd', 'ab\'cd'),  
('abc', 'abc', 'abc');
```

```
SELECT utf8 FROM test ORDER BY utf8;
```

```
SELECT utf8 FROM test ORDER BY french;
```

unsorted	sorted by utf8	sorted by french
UTF8	UTF8	UTF8
Apple	AB CD	AB CD
apple	ABC	ab'cd
AB CD	Apple	abc
ABC	ab'cd	ABC
pino	abc	apple
piñata	apple	Apple
ab'cd	pino	piñata
abc	piñata	pino

SPECIFYING COLLATIONS ON COLUMNS

```
CREATE TABLE test (utf8 VARCHAR, french VARCHAR collate 'fr', spanish VARCHAR collate 'es');
```

```
INSERT INTO test VALUES
```

```
('Apple', 'Apple', 'Apple'),  
('apple', 'apple', 'apple'),  
('AB CD', 'AB CD', 'AB CD'),  
('ABC', 'ABC', 'ABC'),  
('pino', 'pino', 'pino'),  
('piñata', 'piñata', 'piñata'),  
('ab\'cd', 'ab\'cd', 'ab\'cd'),  
('abc', 'abc', 'abc');
```

```
SELECT utf8 FROM test ORDER BY utf8;
```

```
SELECT utf8 FROM test ORDER BY french;
```

```
SELECT utf8 FROM test ORDER BY spanish;
```

	unsorted	sorted by utf8	sorted by french	sorted by spanish
	UTF8	UTF8	UTF8	UTF8
	Apple	AB CD	AB CD	AB CD
	apple	ABC	ab'cd	ab'cd
	AB CD	Apple	abc	abc
	ABC	ab'cd	ABC	ABC
	pino	abc	apple	apple
	piñata	Apple	apple	Apple
	ab'cd	pino	piñata	pino
	abc	piñata	pino	piñata



SPECIFYING COLLATIONS IN COMPARISONS

```
CREATE TABLE test (c1 VARCHAR, c2 VARCHAR)
```

```
INSERT INTO test VALUES
  ('its', 'it\s'),
  ('log in', 'login'),
  ('MyTable', 'mytable');
```



C1	C2
its	it's
log in	login
MyTable	mytable



SPECIFYING COLLATIONS IN COMPARISONS

```
CREATE TABLE test (c1 VARCHAR, c2 VARCHAR)
```

```
INSERT INTO test VALUES
  ('its', 'it\s'),
  ('log in', 'login'),
  ('MyTable', 'mytable');
```

```
SELECT * FROM test
WHERE c1=c2;
```



C1	C2
its	it's
log in	login
MyTable	mytable

C1	C2

SPECIFYING COLLATIONS IN COMPARISONS

```
CREATE TABLE test (c1 VARCHAR, c2 VARCHAR)
```

```
INSERT INTO test VALUES
  ('its', 'it\s'),
  ('log in', 'login'),
  ('MyTable', 'mytable');
```



C1	C2
its	it's
log in	login
MyTable	mytable

```
SELECT * FROM test
WHERE c1=c2;
```



C1	C2
its	it's

```
SELECT * FROM test
WHERE c1=c2 collate 'en-ci';
```



C1	C2
MyTable	mytable

SPECIFYING COLLATIONS IN COMPARISONS

```
CREATE TABLE test (c1 VARCHAR, c2 VARCHAR)
```

```
INSERT INTO test VALUES
  ('its', 'it\'s'),
  ('log in', 'login'),
  ('MyTable', 'mytable');
```

```
SELECT * FROM test
WHERE c1=c2;
```

```
SELECT * FROM test
WHERE c1=c2 collate 'en-ci';
```

```
SELECT * FROM test
WHERE c1=c2 collate 'en-ci-pi';
```



C1	C2
its	it's
log in	login
MyTable	mytable

C1	C2

C1	C2
MyTable	mytable

C1	C2
its	it's
log in	login
MyTable	mytable

Subqueries

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



SUBQUERY OVERVIEW

A Subquery is a query within another query

Outer/main query

Generally executed 2nd

```
SELECT <columns>  
FROM <table>  
WHERE column_name <expression> <operator>
```

Subquery

Generally executed 1st

```
( SELECT <columns>  
  FROM <table>  
  WHERE ...  
);
```



ADDITIONAL SUBQUERY INFO

- Can be in WHERE, HAVING, and FROM clauses
- Can be used with SELECT, UPDATE, INSERT, DELETE along with an expression operation
- Subqueries are on the right side of the comparison operator:

```
SELECT rep, total_sales  
FROM sales_results  
WHERE total_sales >  
  (SELECT AVG(total_sales)  
   FROM sales_results);
```



ADDITIONAL SUBQUERY INFO

- Single-row subqueries:

- Can return either 0 or 1 rows (single value)
- Use single-row operators (MIN, MAX, AVG...)

```
SELECT name, salary, dept FROM employees  
WHERE salary = (SELECT MIN(salary) FROM payroll);
```

- Multiple-row subqueries:

- Can return more than one row
- Use multiple-row operators (IN, ANY, ALL...)

```
SELECT name, salary, dept FROM employees  
WHERE name IN (SELECT emp_name FROM stats  
                WHERE years_service >= '10');
```



USE TEMPORARY TABLE FOR REPETITIVE SUBQUERIES

If you're using the same subquery multiple times, use a temporary table to materialize subquery results:

- Improves performance by reducing repeated I/O
- Saves cost by reducing repeated computation
- Temporary table exists only within session

```
create temporary table  
mydb.public.customer_total_return as (  
    select sr_customer_sk as ctr_customer_sk,  
          sr_store_sk as ctr_store_sk,  
          sum(SR_RETURN_AMT_INC_TAX) as ctr_total_return  
     from store_returns  
       ,date_dim  
    where sr_returned_date_sk = d_date_sk  
      and d_year =1999  
   group by sr_customer_sk,sr_store_sk  
);  
  
select c_customer_id  
  from mydb.public.customer_total_return ctr1  
 ,store  
 ,customer  
 where ctr1.ctr_total_return >  
       (select avg(ctr_total_return)*1.2  
        from mydb.public.customer_total_return ctr2  
       where ctr1.ctr_store_sk = ctr2.ctr_store_sk  
     )  
   and s_store_sk = ctr1.ctr_store_sk  
   and s_state = 'NM'  
   and ctr1.ctr_customer_sk = c_customer_sk  
  order by c_customer_id  
 limit 100;
```

Module 3: Querying Data in Snowflake

Exercise 3.2: Practice Query Fundamentals

60 minutes

Tasks:

- SELECT, DISTINCT, and FROM
- WHERE and LIMIT
- GROUP BY, HAVING, and ORDER BY
- JOIN
- PIVOT
- Subqueries



LAB RECAP

Exercise 3.2, Task 2

In task 2, you ran a query with LIMIT on two tables: the NATION table and the LINEITEM table. The results returned from the NATION table were always the same, but the results returned from the LINEITEM table were sometimes different. Why do yo think this happened?

- The NATION table has 1 micropartition.
- The LINEITEM table has 10 micropartitions.

This has to do with query optimization. Snowflake will return rows from the same micropartition in response to a LIMIT. Since NATION has only a single micropartition, you will always get the same results back. LINEITEM has 10 micropartitions, so your results will cycle between those 10 micropartitions.



LAB RECAP

Exercise 3.2, Task 5

Step 5. Run a query with PIVOT that will list the daily average sales instead of daily total sales. Have the days of the week as rows, and the names across the top.

```
SELECT * FROM weekly_sales  
PIVOT (AVG(amount)  
FOR name IN ('Fred', 'Rita', 'Mark')) ;  
--Columns to use  
--Value to "pivot" around  
--Values to appear as columns
```



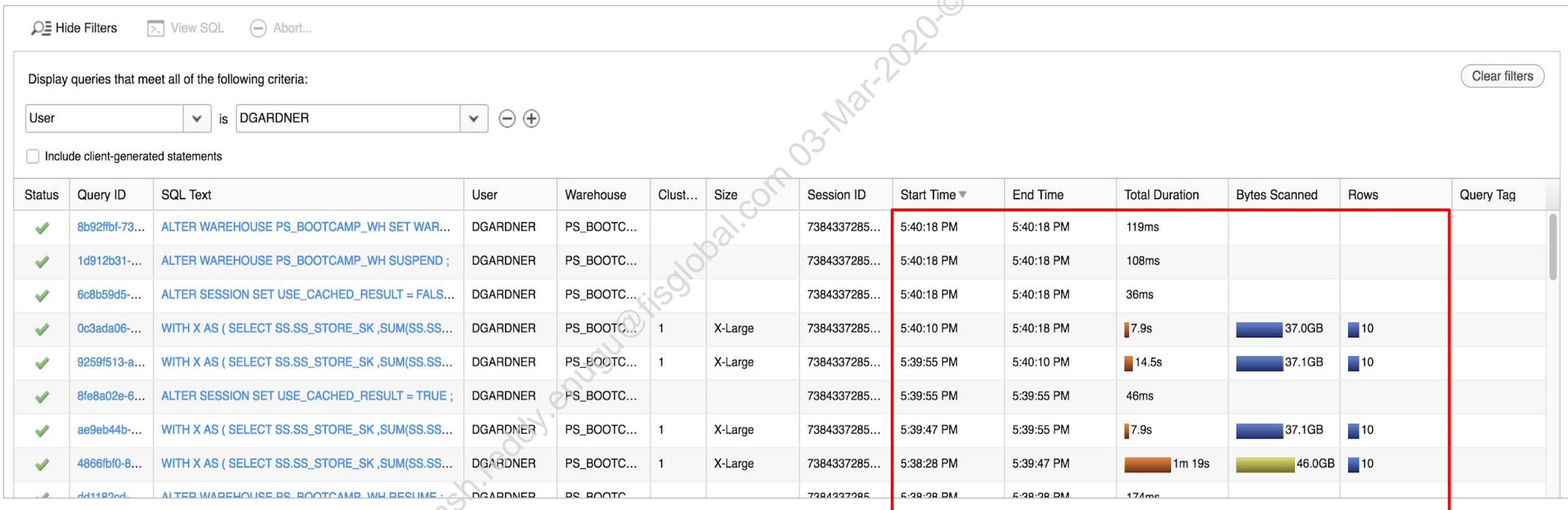
Query Profile

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



QUERY STATISTICS - HISTORY TAB

- Provides a tabular, high-level view of each query
- Includes basic performance metrics (duration, bytes scanned, rows)
- Color-coded to provide quick insights



Hide Filters View SQL Abort...

Display queries that meet all of the following criteria:

User is DGARDNER

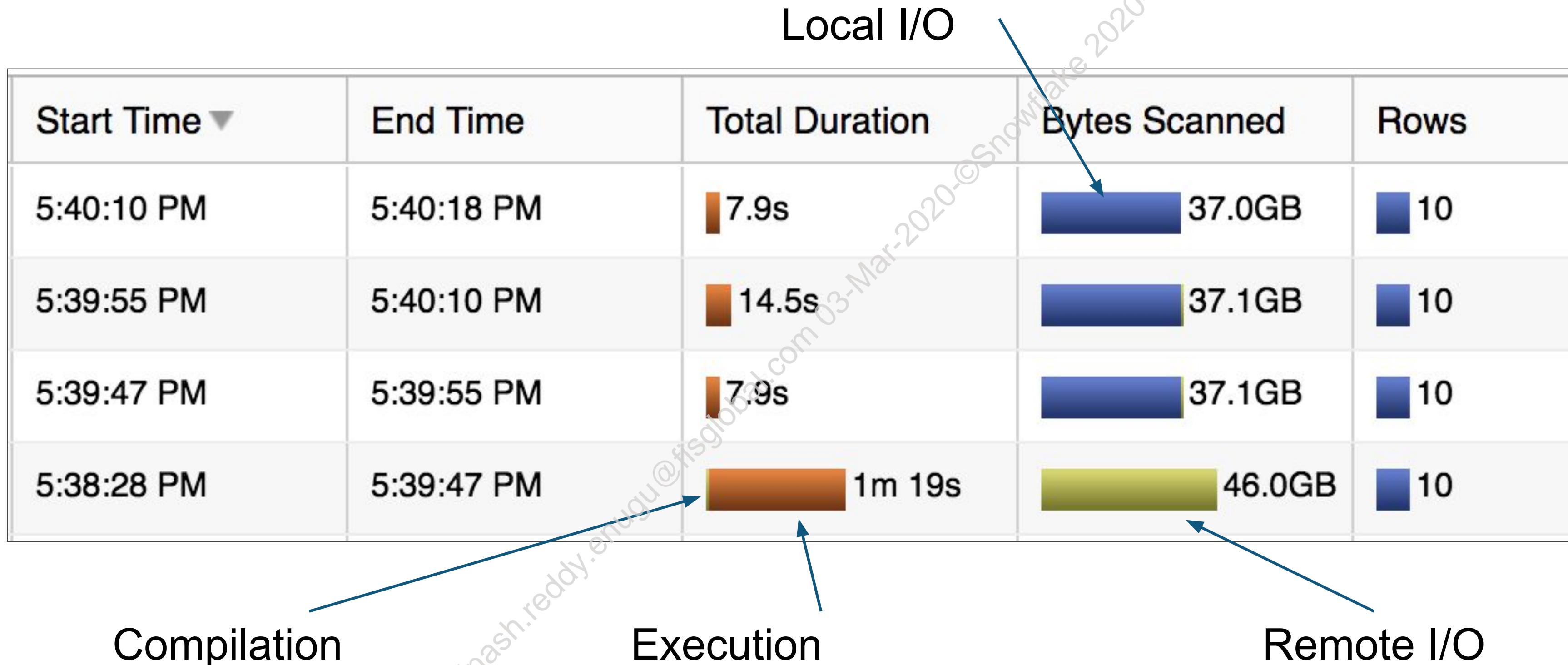
Include client-generated statements

Clear filters

Status	Query ID	SQL Text	User	Warehouse	Clust...	Size	Session ID	Start Time	End Time	Total Duration	Bytes Scanned	Rows	Query Tag
✓	8b92ffbf-73...	ALTER WAREHOUSE PS_BOOTCAMP_WH SET WAR...	DGARDNER	PS_BOOTC...			7384337285...	5:40:18 PM	5:40:18 PM	119ms			
✓	1d912b31-...	ALTER WAREHOUSE PS_BOOTCAMP_WH SUSPEND ;	DGARDNER	PS_BOOTC...			7384337285...	5:40:18 PM	5:40:18 PM	108ms			
✓	6c8b59d5-...	ALTER SESSION SET USE_CACHED_RESULT = FALS...	DGARDNER	PS_BOOTC...			7384337285...	5:40:18 PM	5:40:18 PM	36ms			
✓	0c3ada06-...	WITH X AS (SELECT SS.SS_STORE_SK ,SUM(SS.SS...	DGARDNER	PS_BOOTC...	1	X-Large	7384337285...	5:40:10 PM	5:40:18 PM	7.9s	37.0GB	10	
✓	9259f513-a...	WITH X AS (SELECT SS.SS_STORE_SK ,SUM(SS.SS...	DGARDNER	PS_BOOTC...	1	X-Large	7384337285...	5:39:55 PM	5:40:10 PM	14.5s	37.1GB	10	
✓	8fe8a02e-6...	ALTER SESSION SET USE_CACHED_RESULT = TRUE ;	DGARDNER	PS_BOOTC...			7384337285...	5:39:55 PM	5:39:55 PM	46ms			
✓	ae9eb44b-...	WITH X AS (SELECT SS.SS_STORE_SK ,SUM(SS.SS...	DGARDNER	PS_BOOTC...	1	X-Large	7384337285...	5:39:47 PM	5:39:55 PM	7.9s	37.1GB	10	
✓	4866fbf0-8...	WITH X AS (SELECT SS.SS_STORE_SK ,SUM(SS.SS...	DGARDNER	PS_BOOTC...	1	X-Large	7384337285...	5:38:28 PM	5:39:47 PM	1m 19s	46.0GB	10	
✓	dd1182ed...	ALTER WAREHOUSE PS_BOOTCAMP_WH RESUME ;	DGARDNER	PS_BOOTC...			7384337285...	5:39:29 PM	5:39:29 PM	174ms			

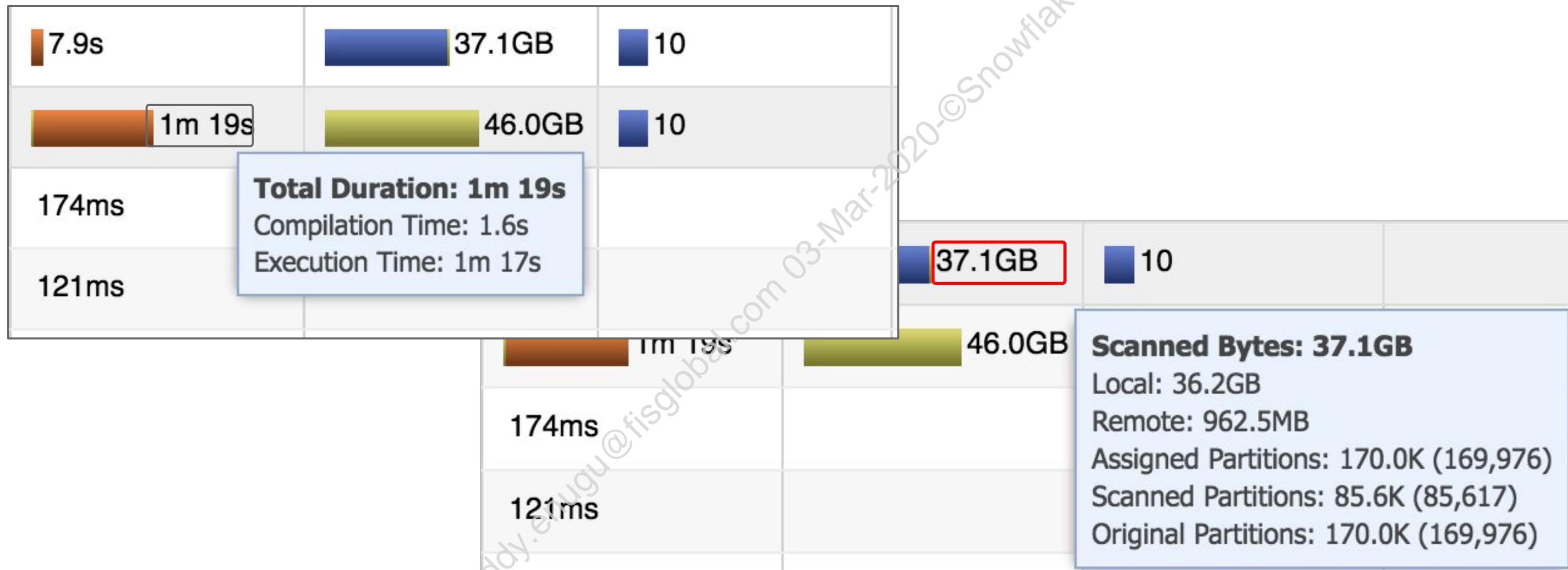


QUERY STATISTICS - HISTORY TAB



QUERY STATISTICS - HISTORY TAB

Hover over numeric value to see breakdown



ACCESS THE QUERY PROFILE

Multiple ways to access the SQL Profiler

Query results pane in worksheet

Row	C_CUSTKEY	C_NAME	C_ADDRESS
1	60001	Customer#0000...	9li4zQn9cX
2	60002	Customer#0000...	ThGBMjDwKzko...
3	60003	Customer#0000...	Ed hbPtTXMTAs...

History tab

Status	Query ID	SQL Text	User	Warehouse	Cluster
✓	34e4e084...	ALTER WAREHOUSE PS_BOOTCAMP_WH SET WAR...	DGARDNER	PS_BOOTC...	
✓	db40cb27...	ALTER WAREHOUSE PS_BOOTCAMP_WH SUSPEND ;	DGARDNER	PS_BOOTC...	
✓	2685f614-e...	WITH X AS (SELECT SS.SS_STORE_SK ,SUM(SS.SS...	DGARDNER	PS_BOOTC...	1
✓	b932d4ea-	Query ID: 2685f614-e220-4cbf-a5e9-3856eddeddd4c RESUME ;	DGARDNER	PS_BOOTC...	



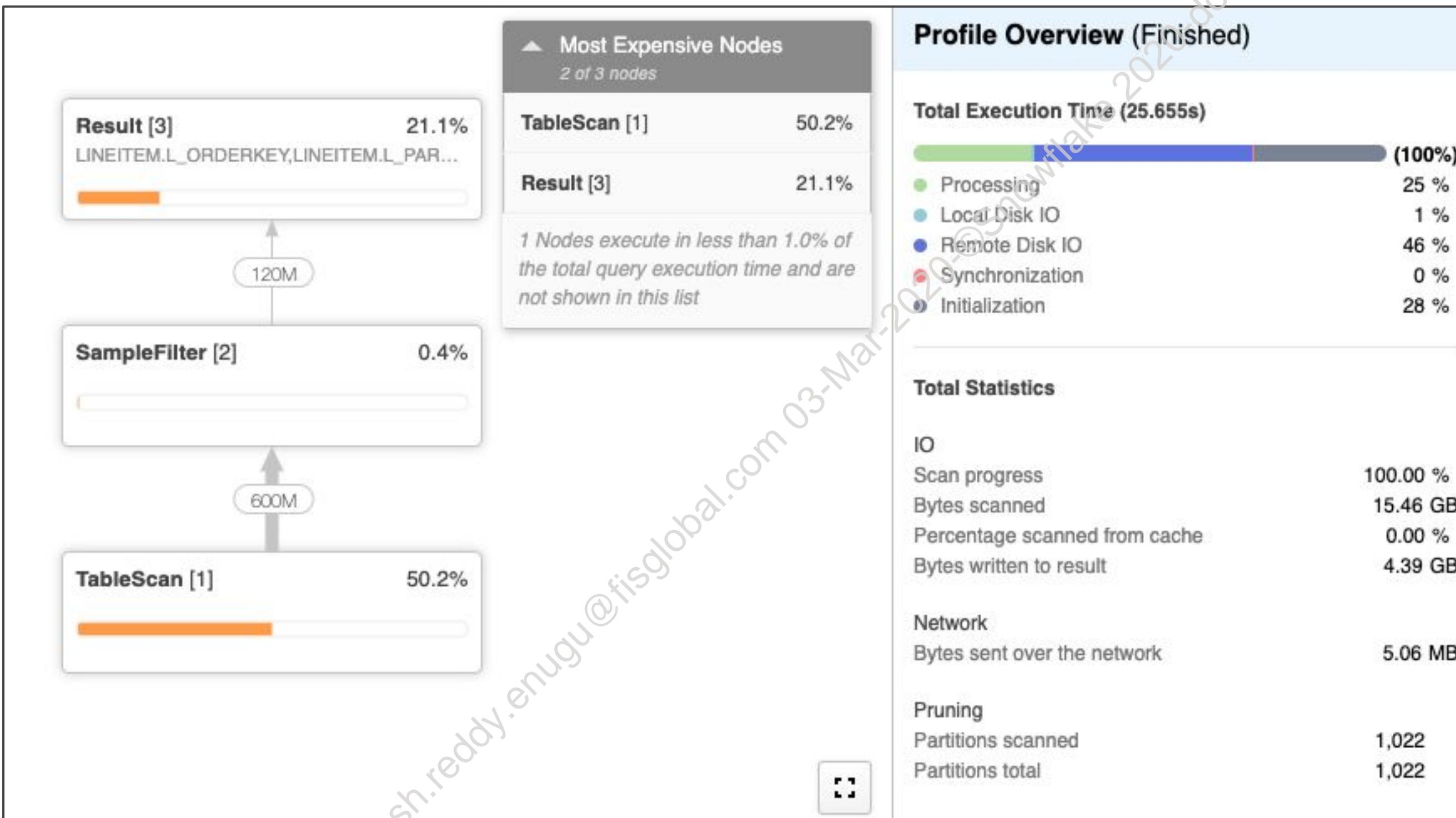
QUERY PROFILE - DETAILS TAB

Clicking on a Query ID brings up this view:

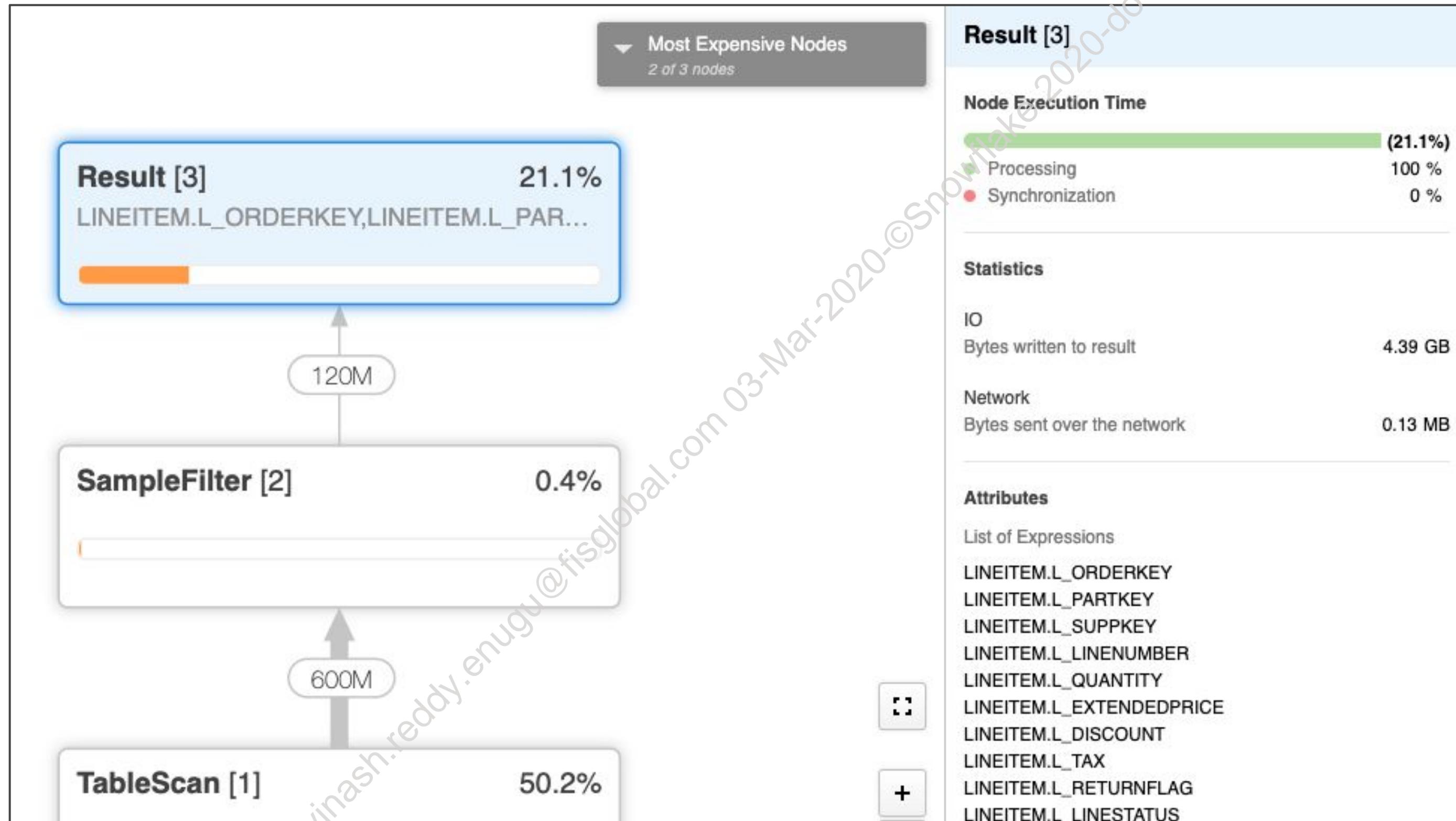
Details		Profile
Status		Success
User		DGARDNER
Warehouse		PS_BOOTCAMP_WH
Start Time		11/15/18 4:32:23 PM
End Time		11/15/18 4:32:34 PM
Total Duration		11.4s
Scanned Bytes		879.3MB
Rows		10
Query ID		2685f614-e220-4cbf-a5e9-3856eddedd4c
Session ID		4401484
SQL Text		
<pre>1 WITH X AS (2 SELECT SS.SS_STORE_SK 3 ,SUM(SS.SS_EXT_SALES_PRICE) AS SUM_SS_EXT_SALES_PRICE 4 FROM STORE_SALES_CLUST_DATE_ITEM SS 5 JOIN SAMPLE_DATA.TPCDS_SF10TCL.DATE_DIM DD 6 ON DD.D_DATE_SK = SS.SS SOLD_DATE_SK 7 WHERE DD.D_DATE >= DATEADD(DAY, -6, '2003-01-02' ::DATE) 8 AND DD.D_DATE <= '2003-01-02' ::DATE 9 AND SS.SS_STORE_SK IS NOT NULL 10 GROUP BY SS.SS_STORE_SK 11 ORDER BY SUM_SS_EXT_SALES_PRICE DESC 12 LIMIT 10 13) 14 SELECT S.S_STORE_NAME 15 ,X.SUM_SS_EXT_SALES_PRICE 16 FROM X 17 JOIN SAMPLE_DATA.TPCDS_SF10TCL.STORE S 18 ON S.S_STORE_SK = X.SS_STORE_SK 19 ORDER BY X.SUM_SS_EXT_SALES_PRICE DESC 20 ;</pre>		



QUERY PROFILE - PROFILE TAB



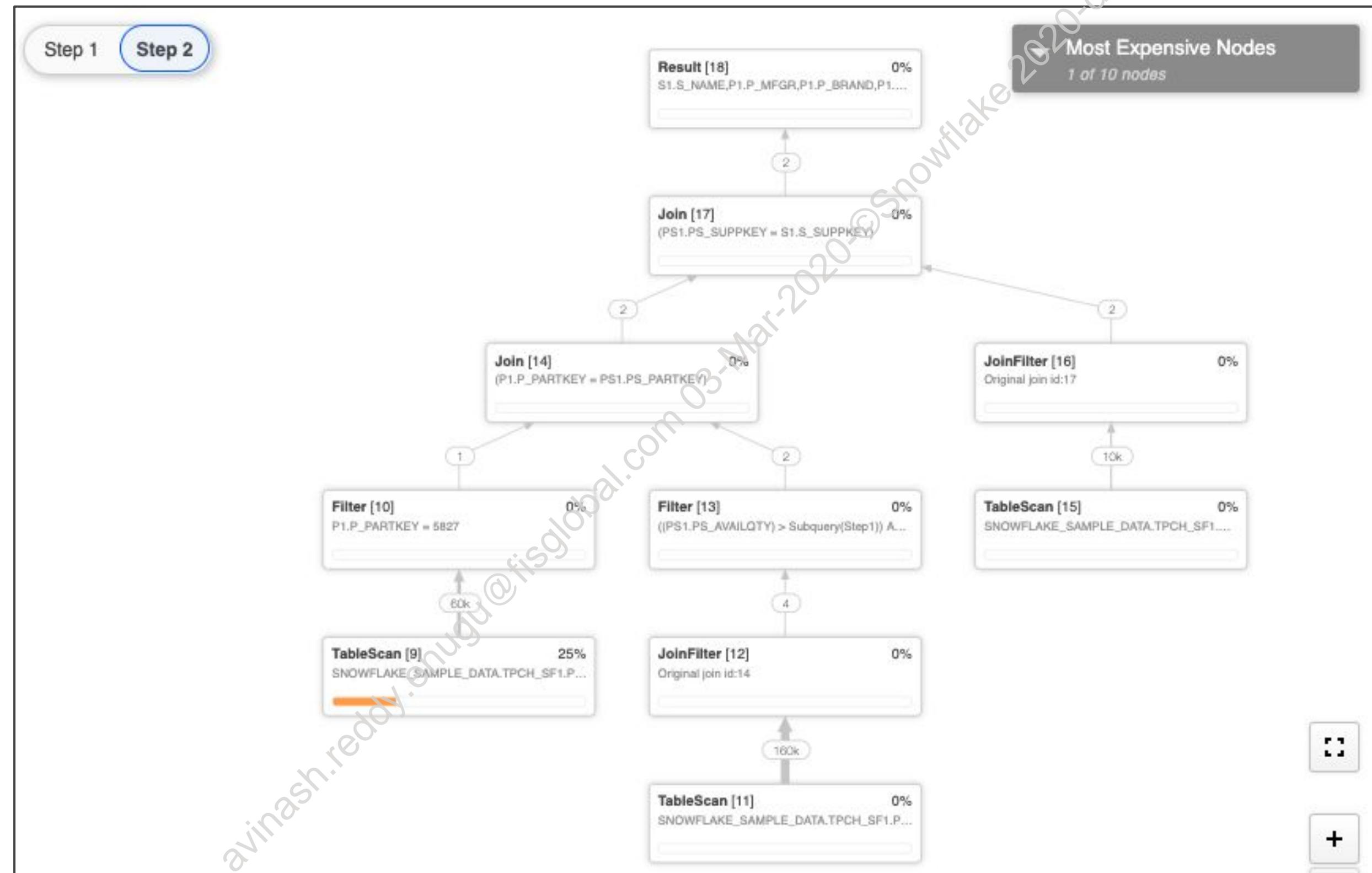
QUERY PROFILE - MORE DETAILS



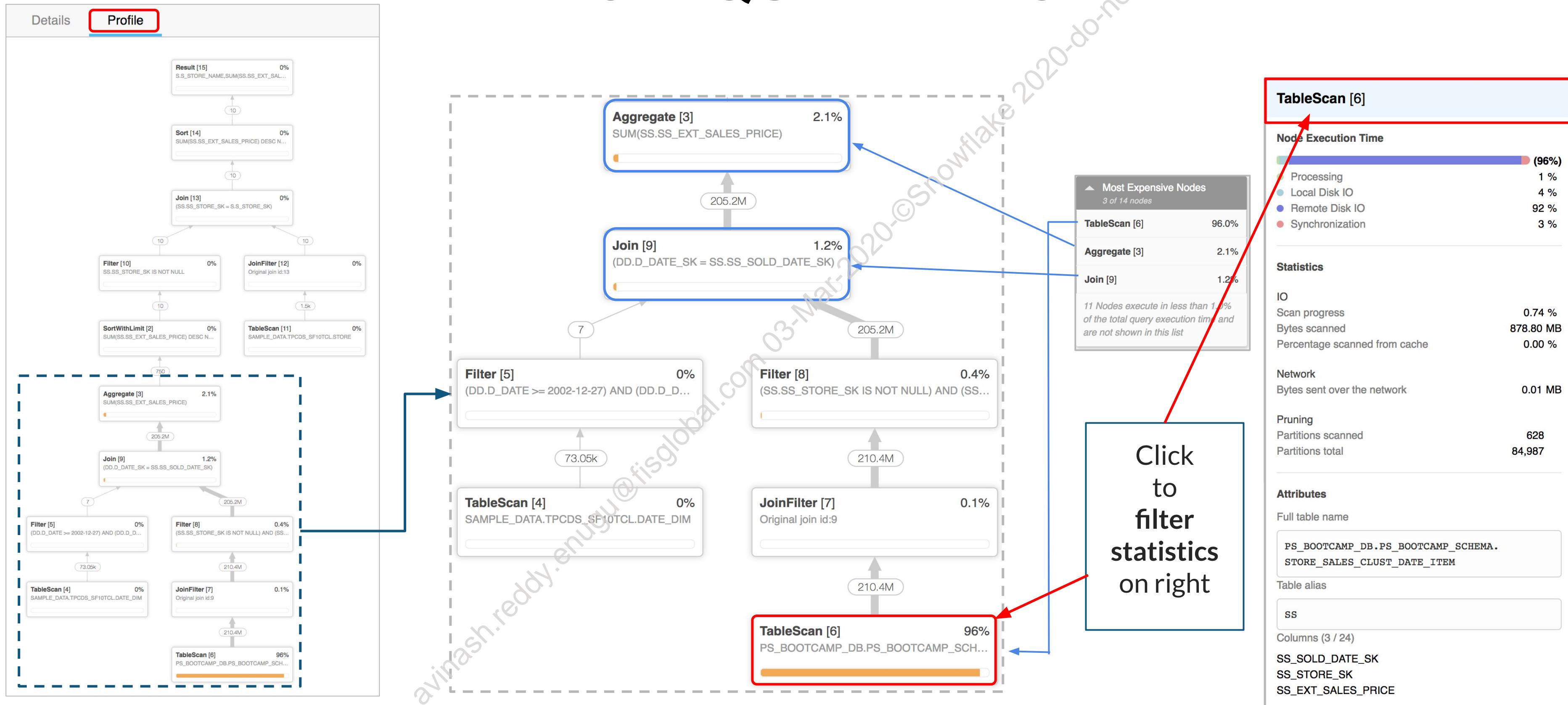
MULTI-STEP QUERIES - STEP 1



MULTI-STEP QUERIES - STEP 2

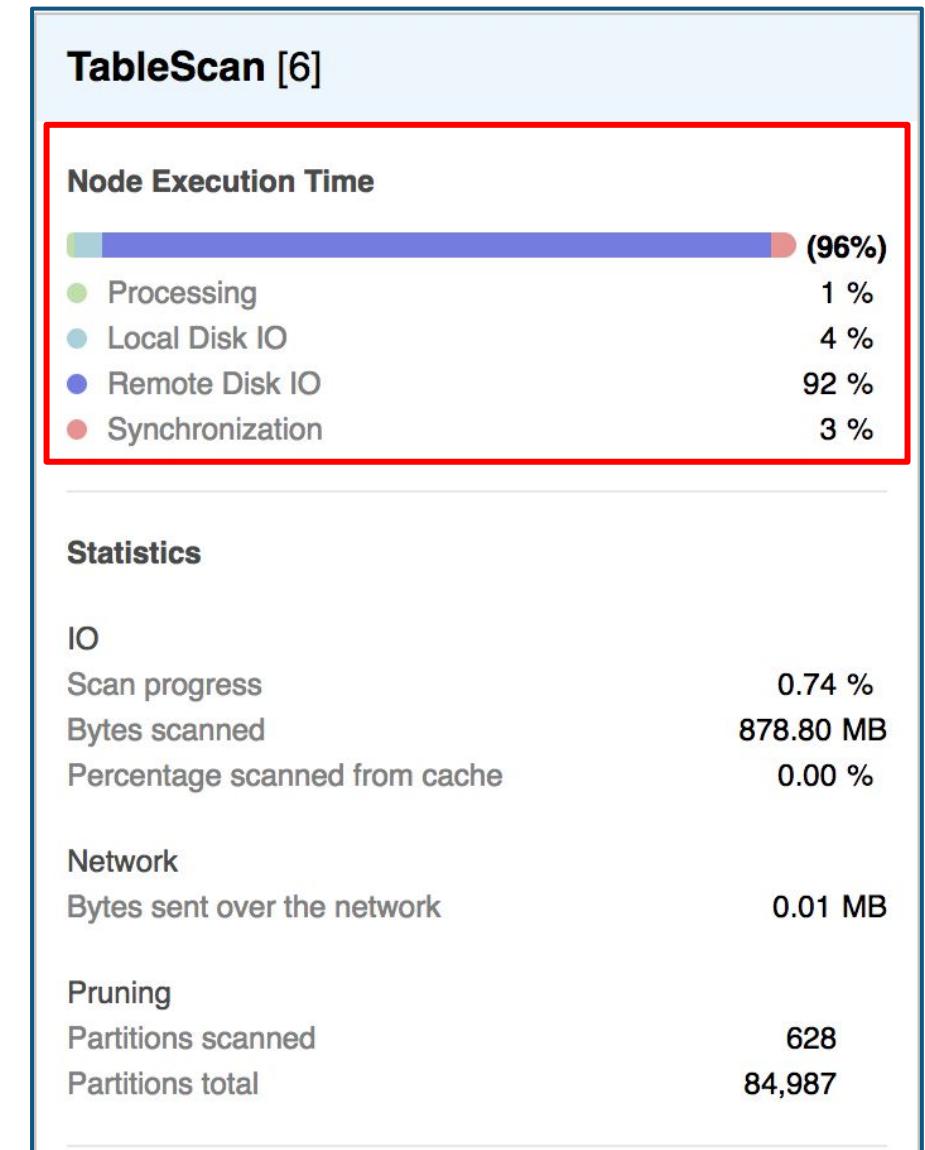


READING A QUERY PROFILE



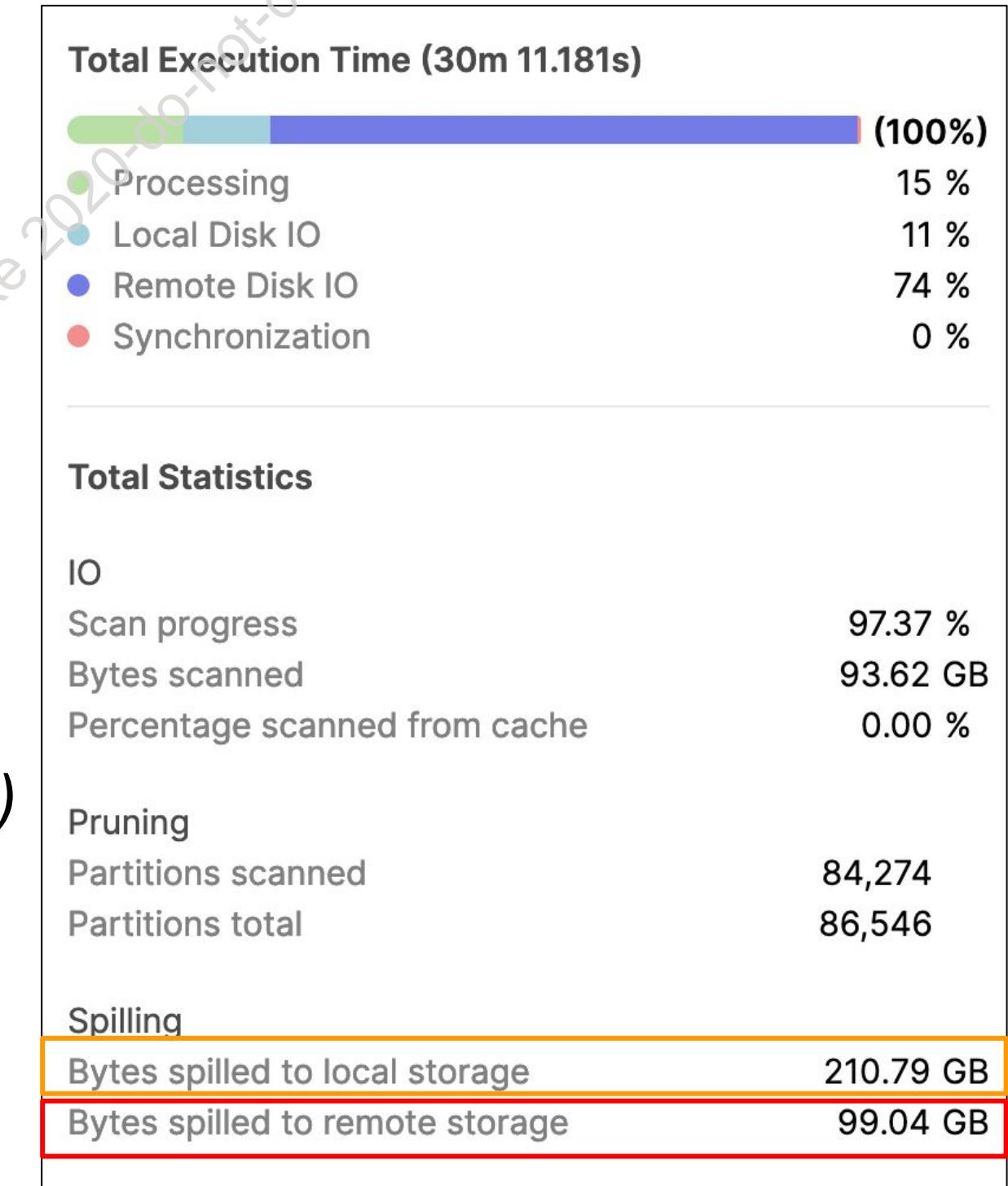
NODE EXECUTION TIME

- **Initialization** = *setup activities prior to processing*
- **Processing** = *CPU data processing*
- **Local Disk I/O** = blocked on (*read from/write to*) local SSD on node
- **Remote Disk I/O** = blocked on (*read from/write to*) remote cloud storage
- **Network Communication** = *blocked on network data transfer (read from/write to)*
- **Synchronization** = *various sync activities between processes*



STATISTICS

- **Scan progress** = % of data scanned for table thus far
- **Bytes scanned** = local + remote I/O
- **Percentage scanned from cache** = local / (local + remote)
- **Bytes written** = bytes into Snowflake (table)
- **Bytes written to result** = bytes to result object
- **Bytes read from result** = bytes from result object
- **External bytes scanned** = bytes from external object (stage)
- **Bytes sent over the network** = peer-peer data exchange
- **Partitions scanned** = number of micro-partitions read (local + remote)
- **Partitions total** = number of micro-partitions in table
- **Bytes spilled to local storage** = written to local SSD on node (*insufficient memory*)
- **Bytes spilled to remote storage** = written to remote cloud storage (*insufficient local SSD*)



Data Movement

Module 4



avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

MODULE AGENDA

- Concepts
- Bulk Loading Overview
- Data Loading Recommendations
- Continuous Data Loading
- Data Unloading

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

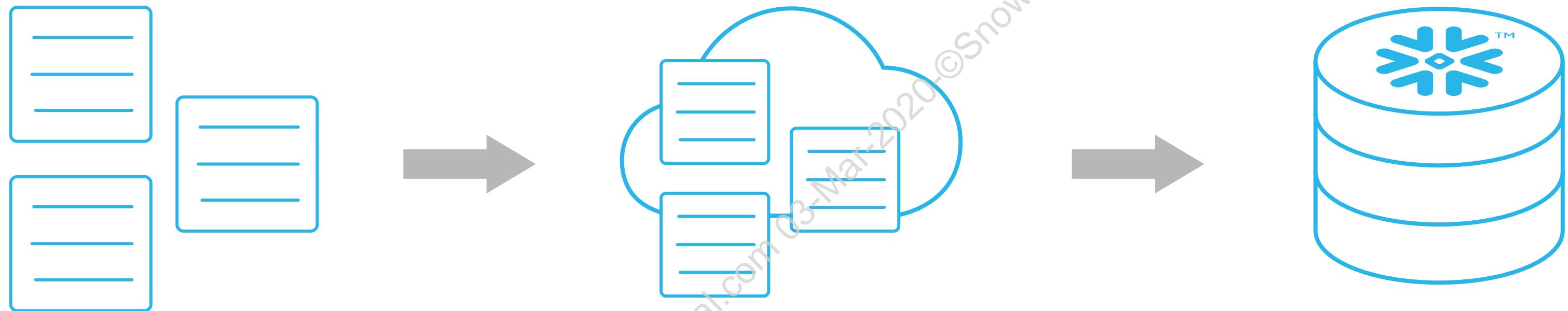


Concepts

avinash.reddy.enugu@fisglobal.com 03 Mar 2020 ©Snowflake 2020-do-not-copy



DATA LOADING STEPS



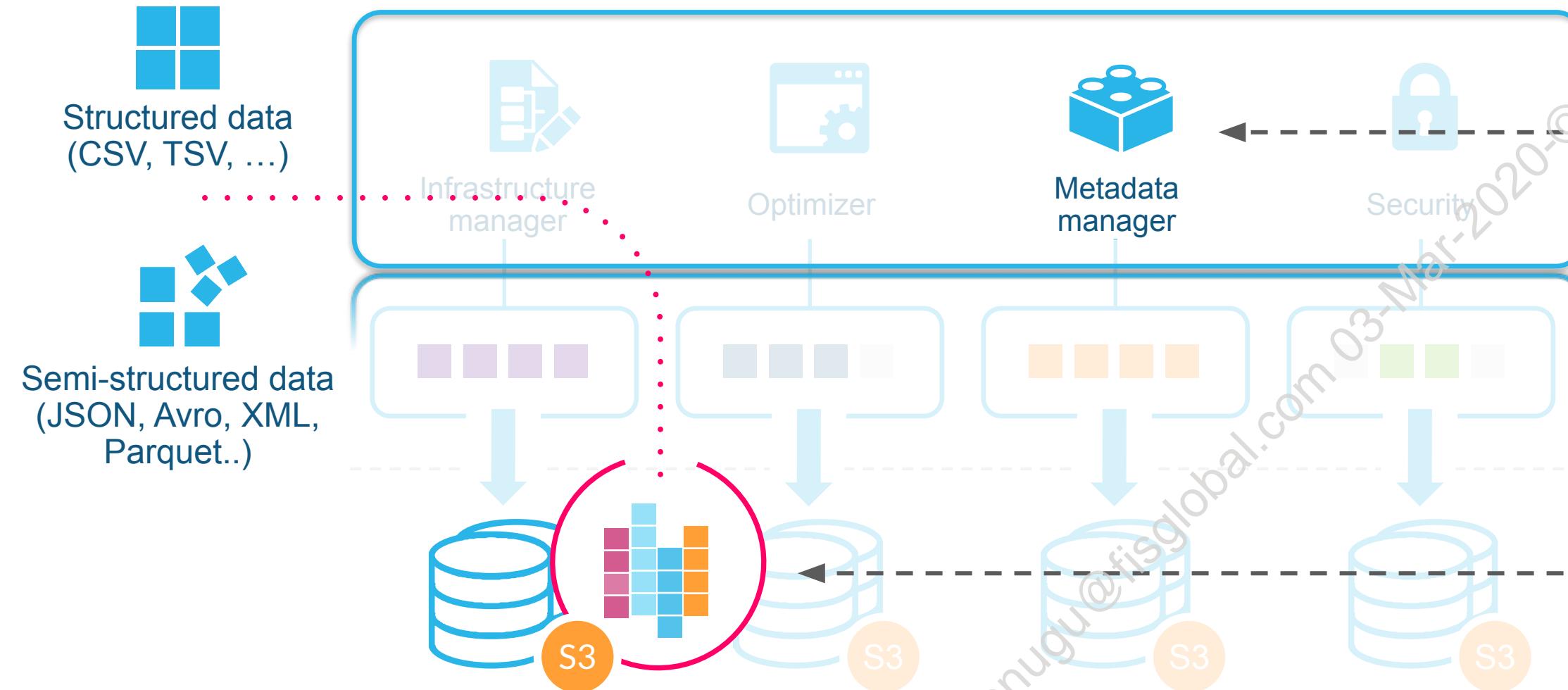
Output Data to Files

Stage Files to Cloud Storage

Load Data from Cloud Storage Into Snowflake



DATA LOADING IN SNOWFLAKE



Metadata created

Statistics for databases, tables, columns, & files calculated & stored in cloud services layer

Database data stored

Actual data in Snowflake databases & tables
Stored in Snowflake-managed cloud storage
Optimized, proprietary file format
Automatically compressed & encrypted



DATA LOADING

TERMINOLOGY



- Stage
- File Format
- Database / Schema / Table
- Pipe

STAGE



- Cloud file repository
- Simplifies and streamlines bulk loading and unloading
- Can be internal or external
 - Internal: stored internally in Snowflake
 - External: stored in an external location
- **Best Practice:** Create stage object to manage ingestion workload

TYPES OF STAGES

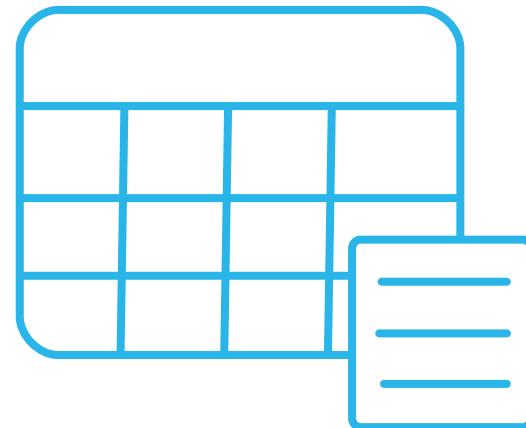
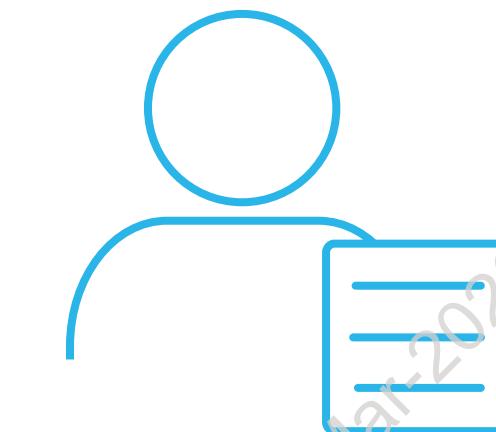


Table Stage
@%[TABLE_NAME]



User Stage
@~[LOGIN]



Named Stage
@[STAGE_NAME]

- Created automatically
- Internal

- Created manually
- Internal or External

FILE FORMAT

```
Col1,Col2,Col3,Col4  
123,ABC,987,FED  
234,BCD,876,EDC  
345,CDE,765,DCB
```

```
CREATE FILE FORMAT DEMO_FF  
TYPE = 'CSV'  
FIELD_DELIMITER = ','  
RECORD_DELIMITER = '\n'  
SKIP_HEADER = 1;
```

- Named object that stores information to parse files during load/unload
 - File type (CSV, JSON, etc.)
 - Type-specific formatting options
- Create FILE FORMAT object as part of the STAGE, or specify within the COPY command



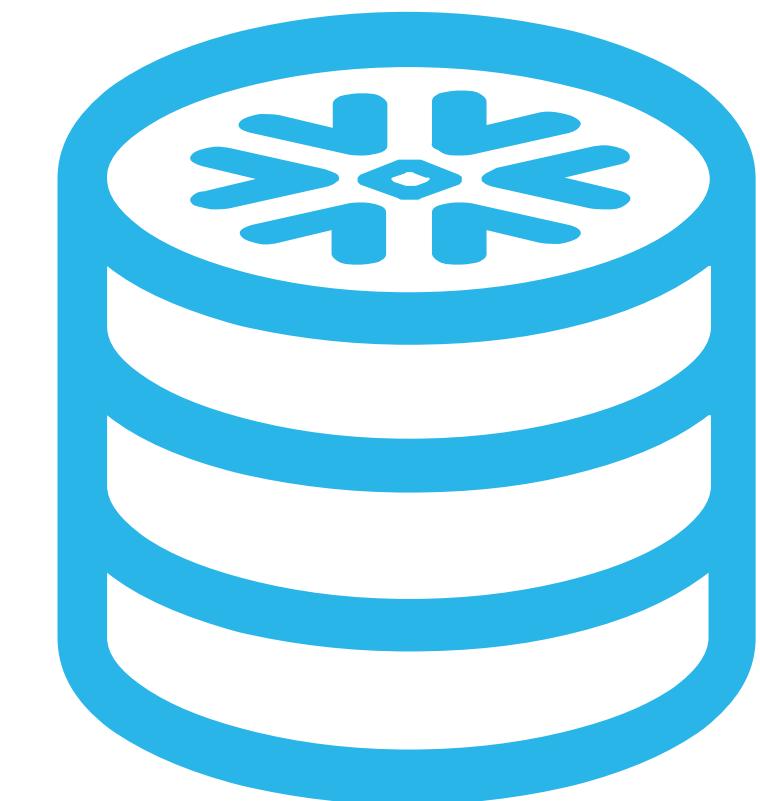
WORKFLOW



CREATE FILE FORMAT

CREATE TABLE

COPY INTO TABLE

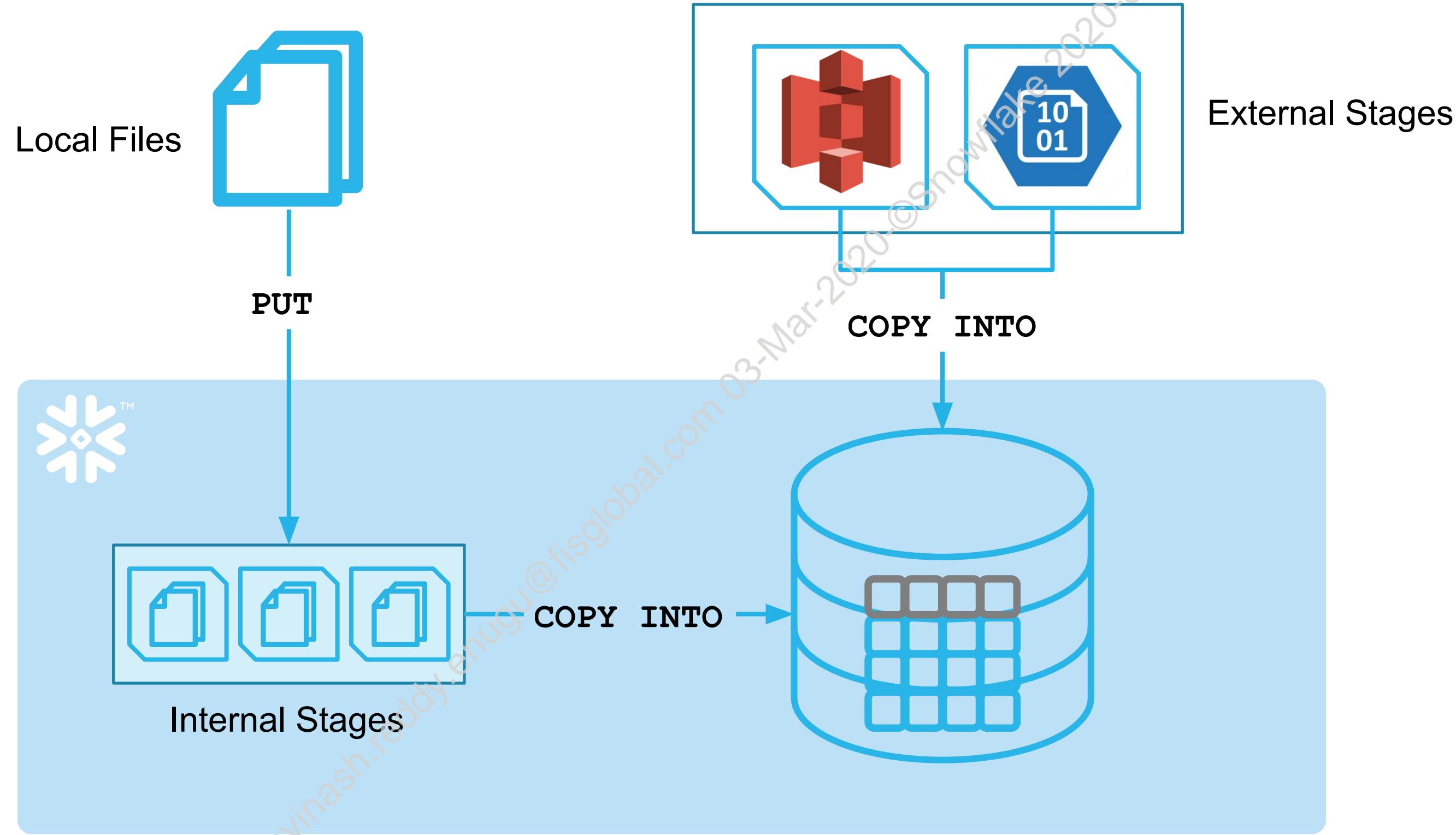


Bulk Loading Overview

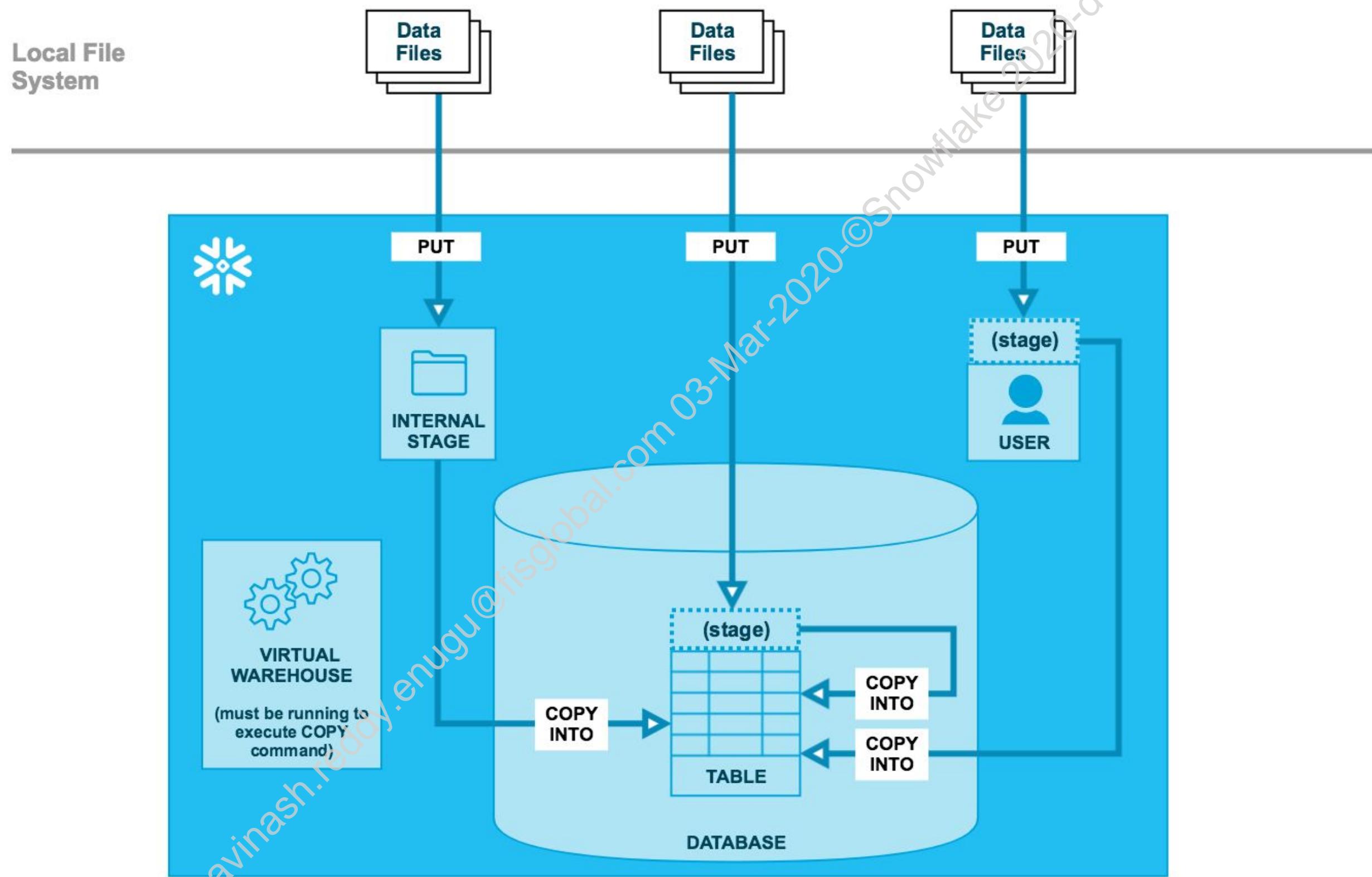
avinash.reddy.enugu@fisglobal.com 02-Mar-2020 ©Snowflake 2020-do-not-copy



BULK LOADING



LOADING FROM LOCAL FILE SYSTEM



EXAMPLE FROM LOCAL STORAGE

Named internal Stage

```
CREATE STAGE my_stage  
FILE_FORMAT = my_csv_format;
```

PUT data to Stage

```
PUT FILE:///data/data.csv @my_stage;
```

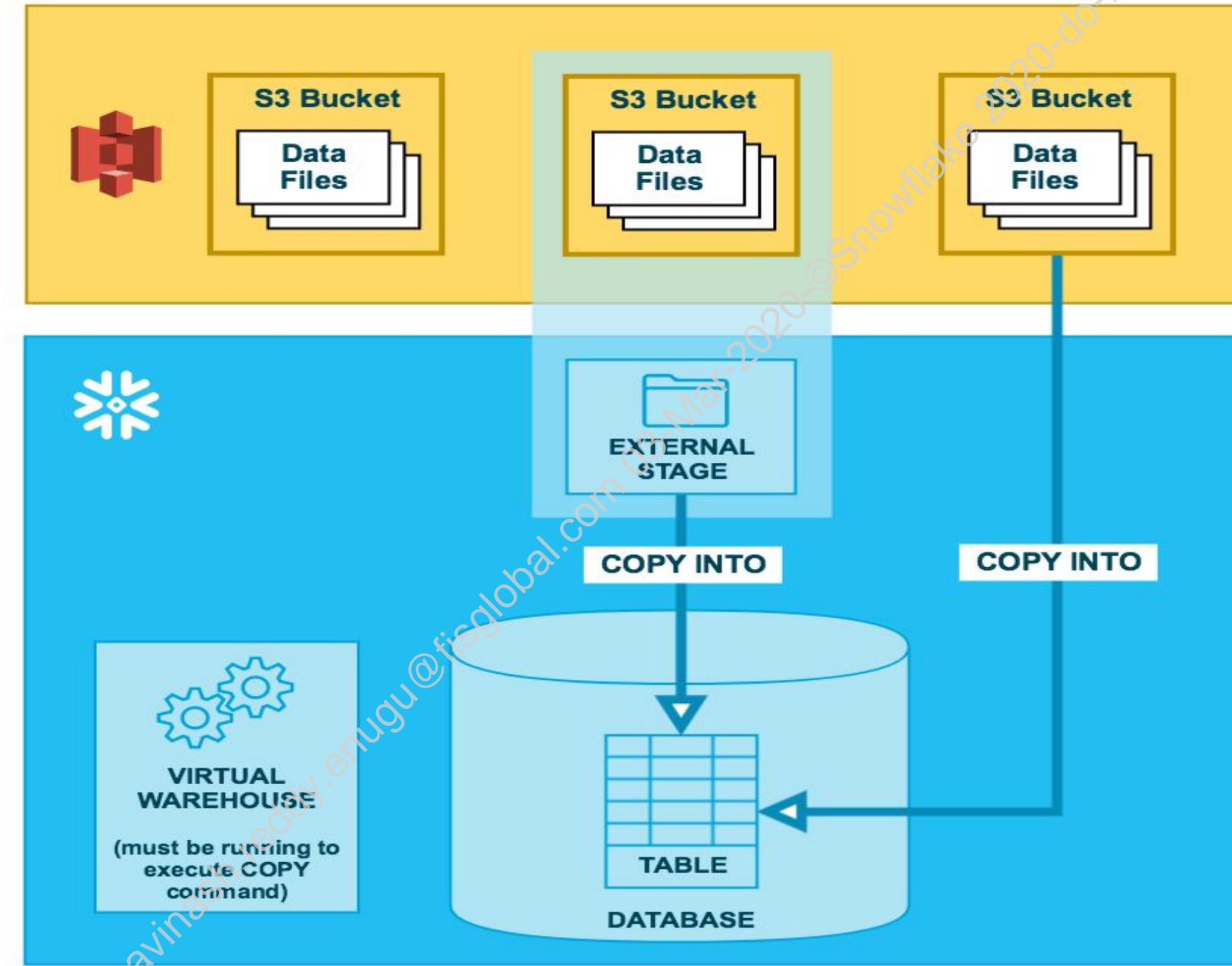
COPY data to Table

```
COPY INTO my_table FROM @my_stage;
```

Note: Compression during the PUT uses local resources. The local host needs sufficient memory, and space in /tmp, or the PUT will fail.



LOAD FROM CLOUD STORAGE



EXAMPLE FROM CLOUD STORAGE

Create external stage (Pointer to S3 bucket)

```
CREATE STAGE my_s3_stage
url='s3://mybucket/encrypted_files/'
CREDENTIALS=(aws_key_id='*****'
aws_secret_key='*****')
ENCRYPTION=(master_key = '*****')
FILE_FORMAT = my_csv_format;
```

Load data using COPY

```
COPY INTO my_table
FROM @my_s3_stage
PATTERN='.*sales.*.csv';
```



COPY COMMAND PARAMETERS

Data can be copied directly from a Stage or using a SELECT statement.

```
COPY INTO [ <namespace>.]<table_name>
    FROM { internalStage | externalStage | externalLocation }
[ FILES = ( '<file_name>' [ , '<file_name>' ] [ , ... ] ) ]
[ PATTERN = '<regex_pattern>' ]
[ FILE_FORMAT = ( { FORMAT_NAME = '[<namespace>.]<file_format_name>' |
                    TYPE = { CSV | JSON | AVRO | ORC | PARQUET | XML } [ formatTypeOptions ] } ) ]
[ copyOptions ]
[ VALIDATION_MODE = RETURN_<n>_ROWS | RETURN_ERRORS | RETURN_ALL_ERRORS ]
```



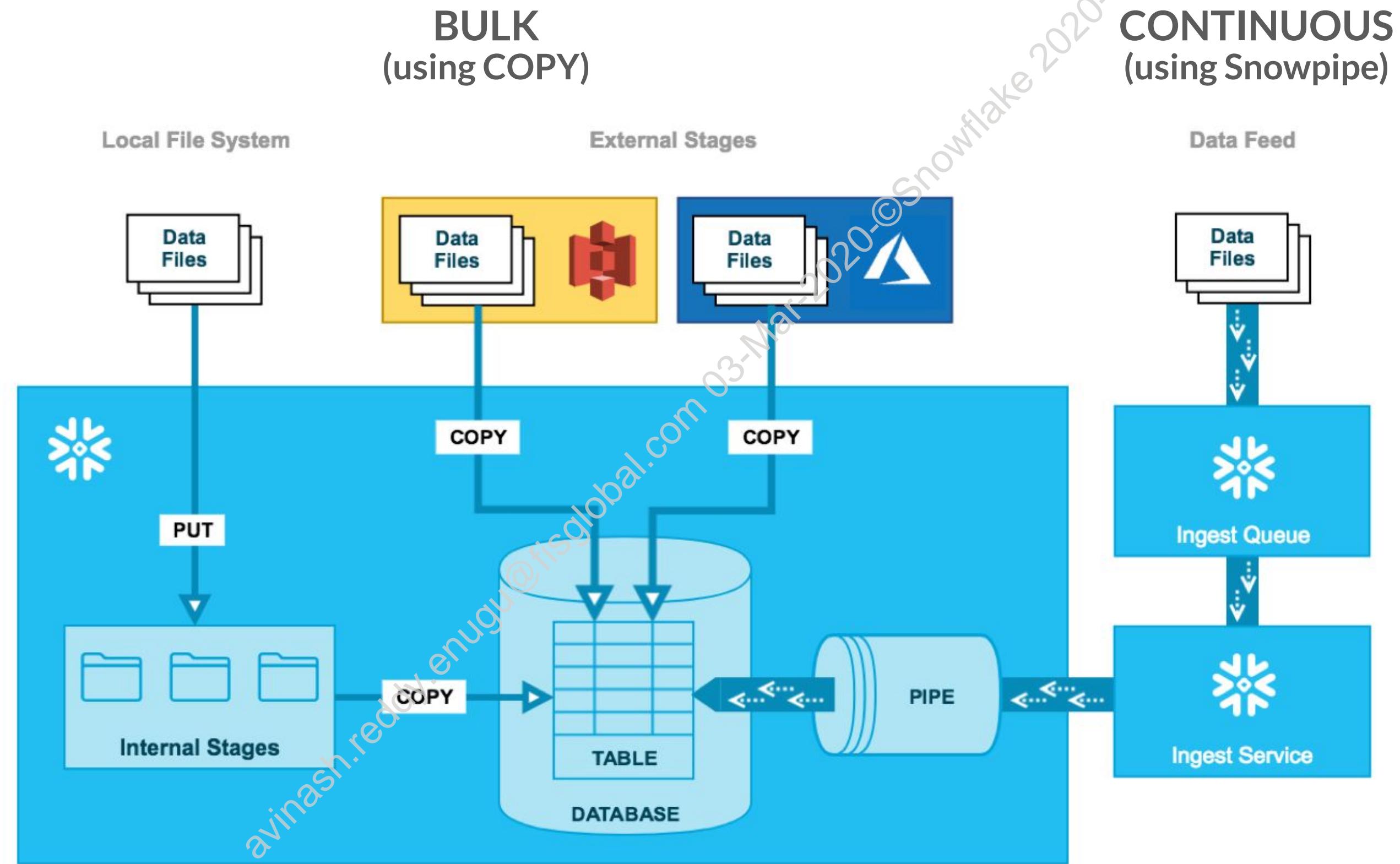
COPY COMMAND OPTIONS

The COPY command has several options for handling data and errors.

```
copyOptions ::=  
    ON_ERROR = { CONTINUE | SKIP_FILE | SKIP_FILE_<num> | SKIP_FILE_<num>% | ABORT_STATEMENT }  
    SIZE_LIMIT = <num>  
    PURGE = TRUE | FALSE  
    RETURN_FAILED_ONLY = TRUE | FALSE  
    ENFORCE_LENGTH = TRUE | FALSE  
    TRUNCATECOLUMNS = TRUE | FALSE  
    FORCE = TRUE | FALSE
```



DATA LOADING APPROACHES



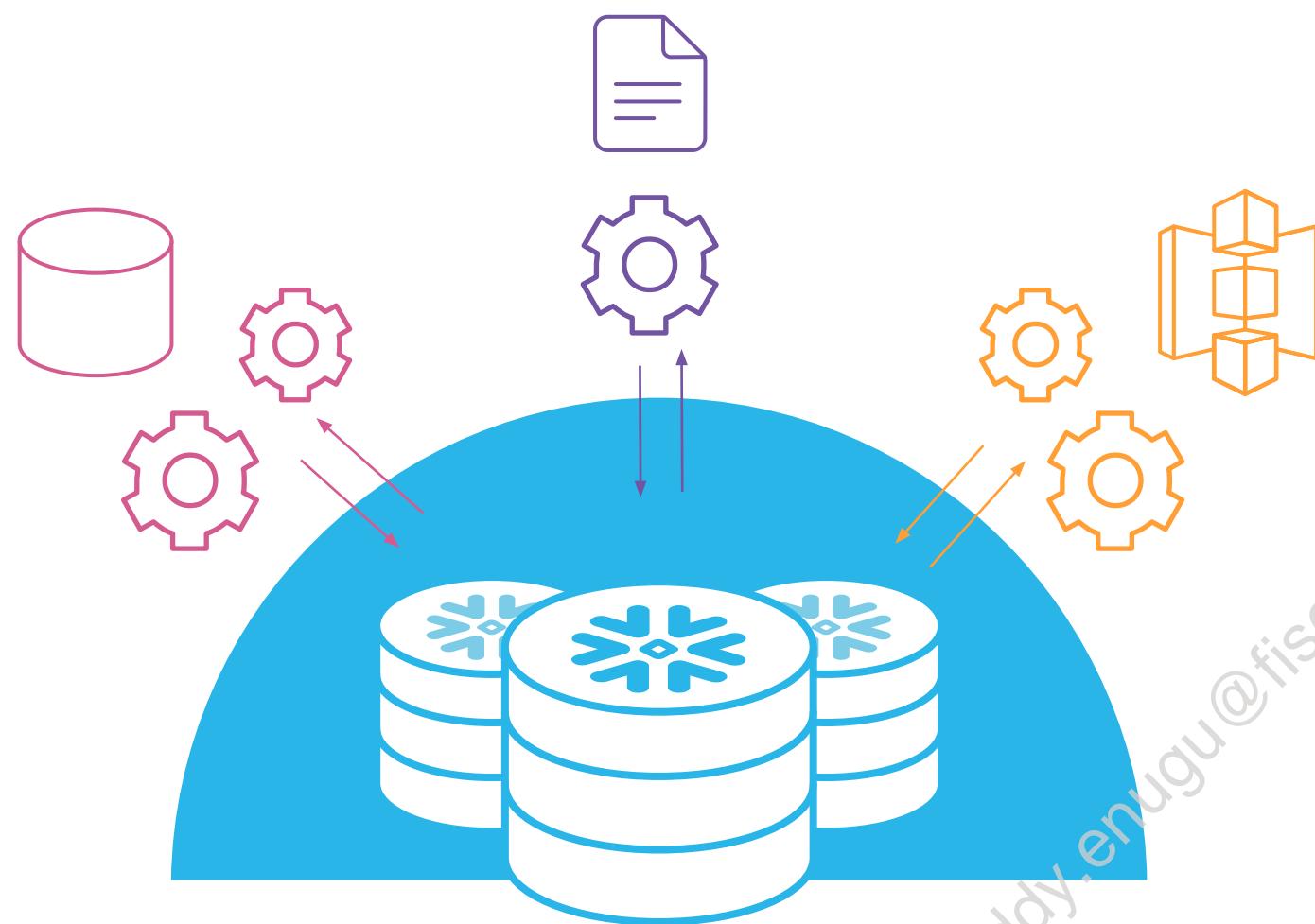
Data Loading Recommendations

avinash.reddy.enugu@fisglobal.com, 03-Mar-2020, ©Snowflake 2020-do-not-copy



RECOMMENDATIONS

DATA LOADING



- File size
- Location path
- File type and format
- Other considerations (concurrency, compression, etc.)

FILE SIZE AND NUMBER

File size and number of files are crucial to optimizing load performance

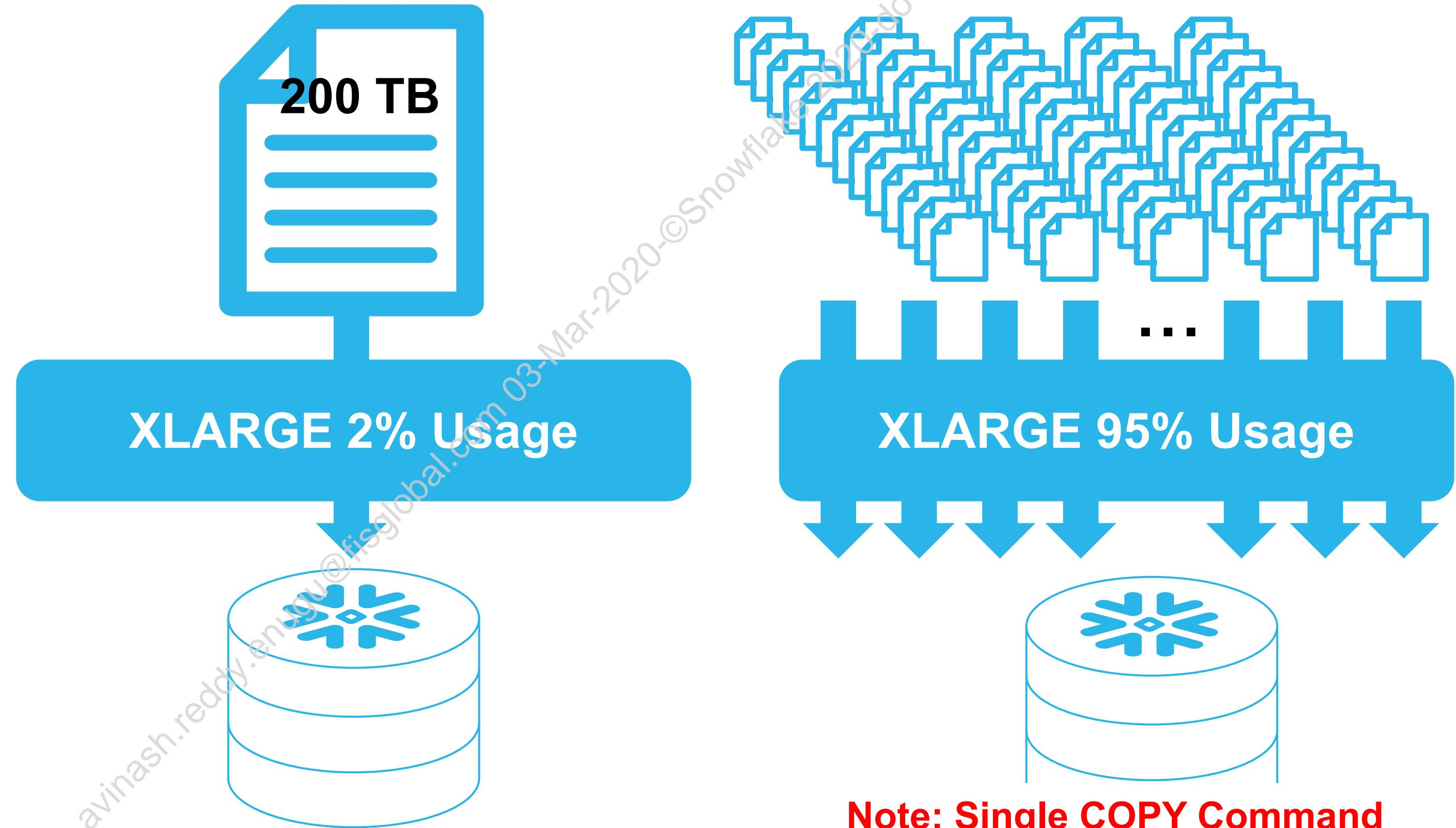
- Split large files before loading into Snowflake
- Recommended: **10MB to 100MB** (compressed)

Warehouse Size	# files in parallel
XS	8
S	16
M	32
L	64
XL	128



SERIAL COPY VS PARALLEL COPY

Data Files
Virtual Warehouse
Snowflake Tables



FILE ORGANIZATION

- Organize data in logical paths (e.g., subject area and create date)

/system/market/daily/2018/09/05/

- Use wildcards late in the file path definition, to reduce scanning:

```
COPY INTO table FROM /system/market/daily/2018/*
```



FILE LOCATIONS



Single Location with Many Files
Slower: must scan more files to find needed files



Many Locations with Fewer Files
Faster: targeted directories allow for scanning fewer files

FILE TYPE AND FORMAT

Source Format	Target Layout	Load Time (sec)	TB/Hour (uncompressed)
CSV (Gzipped)	Structured	1104	15.4
Parquet (Snappy comp)	Semi-Structured	3518	4.8
Parquet (Snappy comp)	Structured	3095	5.4
ORC (Snappy comp)	Semi-Structured	3845	4.4
ORC (Snappy comp)	Structured	2820	6.0



CONCURRENCY

- COPY and INSERT typically do not acquire a partition lock.
 - They acquire table lock for a brief time during transaction commit.
- UPDATE, DELETE and MERGE acquire partition lock, hence:
 - Use staging tables to manage MERGE
 - Consolidate UPDATEs and DELETEs when possible
- Fully ACID compliant



Module 4: Data Movement

Exercise 4.1: Load Structured Data Using the UI

25 minutes

Note: This lab uses an external stage that has already been created for you:

`@DBHOL.SCHOL.AWS_LOAD1 (@<database>.<schema>.<stage>)`

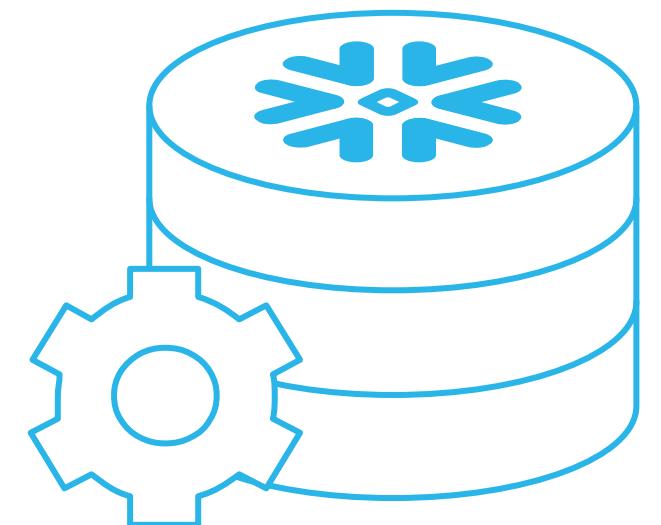
Tasks:

- Download and unzip lab file (if not already done)
- Create tables and file formats
- Load the REGION table from your external stage



TRANSFORMING DATA DURING LOAD

- The COPY command supports column reordering, column omission, and CAST using a SELECT statement
 - NOT SUPPORTED: Joins, filters, aggregations
 - Can include SEQUENCE columns, current_timestamp(), or other column functions during data load
- The VALIDATION_MODE parameter does not support transformations in COPY statements



TRANSFORMING DATA DURING LOAD EXAMPLES

```
COPY INTO home_sales (city, zip, sale_date, price)
FROM (SELECT SUBSTR(t.$2,4), t.$1, t.$5, t.$4 FROM @my_stage t)
FILE_FORMAT = (FORMAT_NAME = MYCSVFORMAT);
```

```
COPY INTO casttb(col1, col2, col3)
FROM (SELECT TO_BINARY(t.$1,'utf-8'),
TO_DECIMAL (t.$2, '99.9', 9, 5), TO_TIMESTAMP_NTZ(t.$3)
FROM @~/datafile.csv.gz t) FILE_FORMAT = (TYPE = CSV)
```



COPY INTO VS INSERT

- Snowflake is optimized for bulk load and batched DML using the COPY INTO command
- Use COPY INTO to load data rather than INSERT with SELECT
- Use INSERT only if needed for transformations not supported by COPY INTO
- Batch INSERT statements
 - INSERT w/ SELECT
 - CREATE TABLE AS SELECT (CTAS)
 - Minimize frequent single row DMLs



QUERYING STAGE DATA

File format

```
177 select
178     substr($1, 0, 15) PTS,
179     substr($1, 16, 3) REC_TYPE,
180     substr($1, 19, 60) COMPANY_NAME,
181     substr($1, 79, 10) CIK,
182     IFF(substr($1, 16, 3) = 'CMP', substr($1, 89, 4), substr($1, 40, 4)) STATUS
183   from @TPCDI_FILES/tpcdi-5/Batch1/FINWIRE
184   (FILE_FORMAT => 'TXT_FIXED_WIDTH')
185  where substr($1, 16, 3) = 'CMP';
186
```

Several functions can be used

Results Data Preview

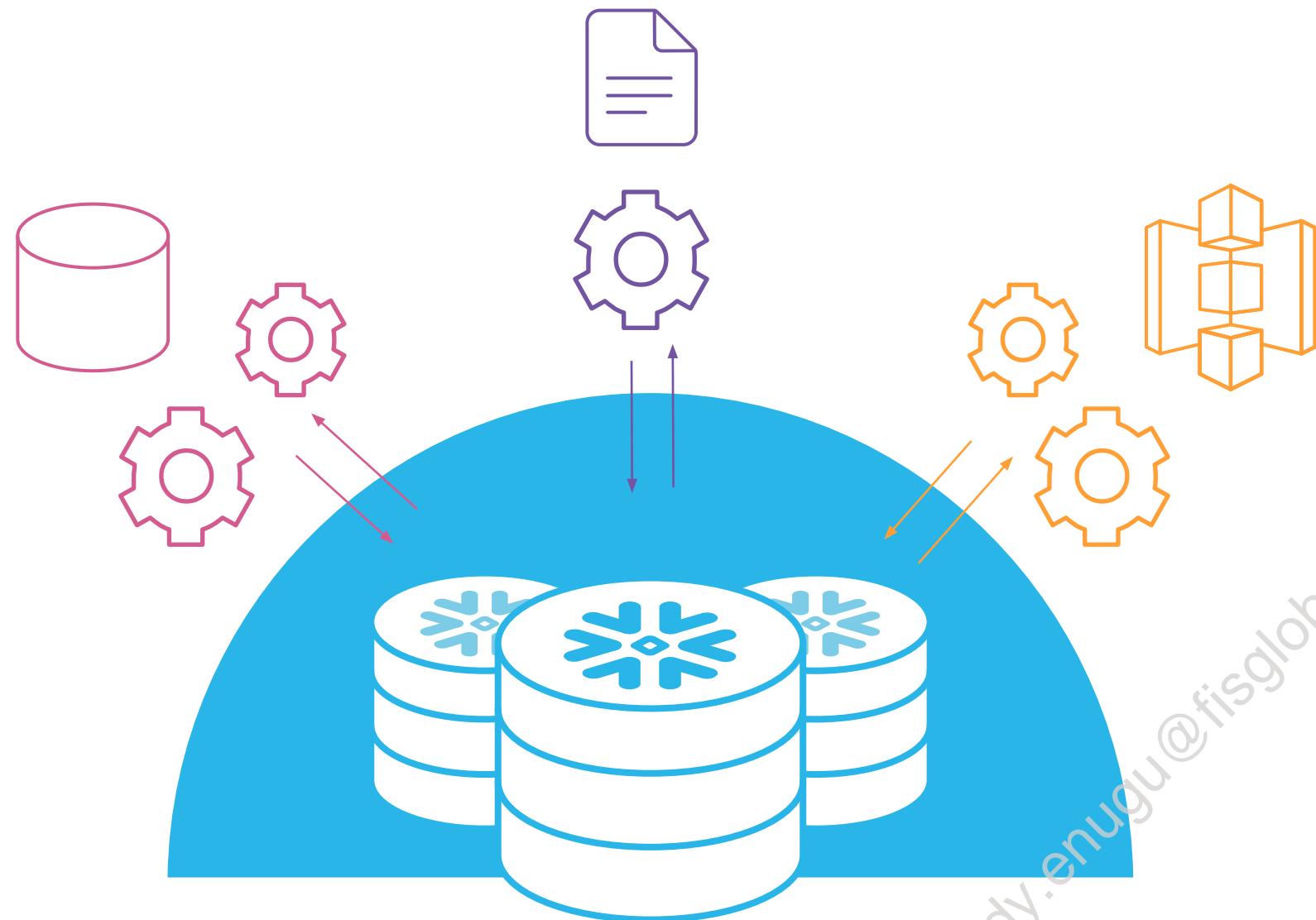
✓ Query ID SQL 2.46s 2,500 rows Add filters

Filter result...

Row	PTS	REC_TYPE	COMPANY_NAME	CIK	STATUS
1	19671209-013009	CMP	RSWvpXShAohbCnBYubXQUbMLtPavDHUQn	0000000102	ACTV
2	19671210-112810	CMP	VNUMGRITCSQKEfjLfhaUANwnlPAHDkoFxyHfyd...	0000000027	INAC
3	19671211-075534	CMP	FQCxtQRoMnVInjwEERVRrjOJDAlyWTEKGCbB...	0000000100	ACTV
4	19671215-123401	CMP	dBBCsCzeivSxrOWJZkMbNfLitfJSVqvvAy	0000000105	ACTV
5	19671216-191413	CMP	OKehKyGaHgbEeBcBDAnFaXwXmCfzDClig	0000000098	ACTV



DATA LOAD MANAGEMENT



- Error handling
- Validating before and after load
- Monitoring

ERROR HANDLING

Use the ON_ERROR option to control how errors in the data load are handled

Supported Values	Notes
CONTINUE	Continue loading the file.
SKIP_FILE	Skip file if any errors encountered in the file.
SKIP_FILE_<num> (e.g. SKIP_FILE_10)	Skip file when the number of errors in the file is equal to or exceeds the specified number.
SKIP_FILE_<num>% (e.g. SKIP_FILE_10%)	Skip file when the percentage of errors in the file exceeds the specified percentage.
ABORT_STATEMENT	Abort the COPY statement if any error is encountered.



VALIDATE BEFORE LOAD

Execute the COPY command in validation mode using VALIDATION_MODE

```
COPY INTO my_table  
FROM @my_stage/mylife.csv.gz  
VALIDATION_MODE=return_all_errors;  
  
SET qid=LAST_QUERY_ID();  
  
COPY INTO @my_stage/errors/load_errors.txt  
FROM (SELECT rejected_record FROM TABLE(result_scan($qid))));
```



VALIDATE AFTER LOAD

Validates the files loaded in a past execution of the COPY INTO and returns *all* errors encountered during the load, rather than just the first error

```
SELECT * FROM table(VALIDATE(mytable, job_id => '<query_id>'));
```

ERROR	FILE	LINE	CHARACTER
Error parsing JSON: unterminated string	tables/530049/bad.json.gz	120	[NULL]
Error parsing JSON: misplaced colon	tables/530049/bad.json.gz	214	18
Error parsing JSON: unknown keyword "tru"	tables/530049/bad.json.gz	1467	[NULL]
Error parsing JSON: unknown keyword "stat"	tables/530049/bad.json.gz	1469	13
Error parsing JSON: unknown keyword "ok"	tables/530049/bad.json.gz	1469	20
Error parsing JSON: invalid character outside of a string: '\\'	tables/530049/bad.json.gz	1469	21
Error parsing JSON: misplaced }	tables/530049/bad.json.gz	1470	3



MONITORING

- Monitor the status of each COPY command run on the History tab page of the Snowflake UI

TABLE_NAME	LAST_LOAD_TIME	STATUS	ROW_COUNT	ROW_PARSED	FIRST_ERROR_MESSAGE
TEST	2018-11-20 07:44:22.868 -0800	LOAD_FAILED	0	3	Date '1,2,3' is not recognized

- Use the LOAD_HISTORY Information Schema view to retrieve the history of data loaded into tables using the COPY command

```
SELECT * FROM information_schema.load_history
WHERE SCHEMA_NAME=current_schema() AND
TABLE_NAME='my_table' AND
LAST_LOAD_TIME > 'Fri, 01 APR 2016 16:00:00 -800';
```



Module 4: Data Movement

Exercise 4.2: VALIDATION_MODE and ON_ERROR

10 minutes

Tasks:

- Detect file format problems with VALIDATION_MODE
- Load data with various ON_ERROR settings



Continuous Data Loading

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



USE CASE

COPY command (BATCH)

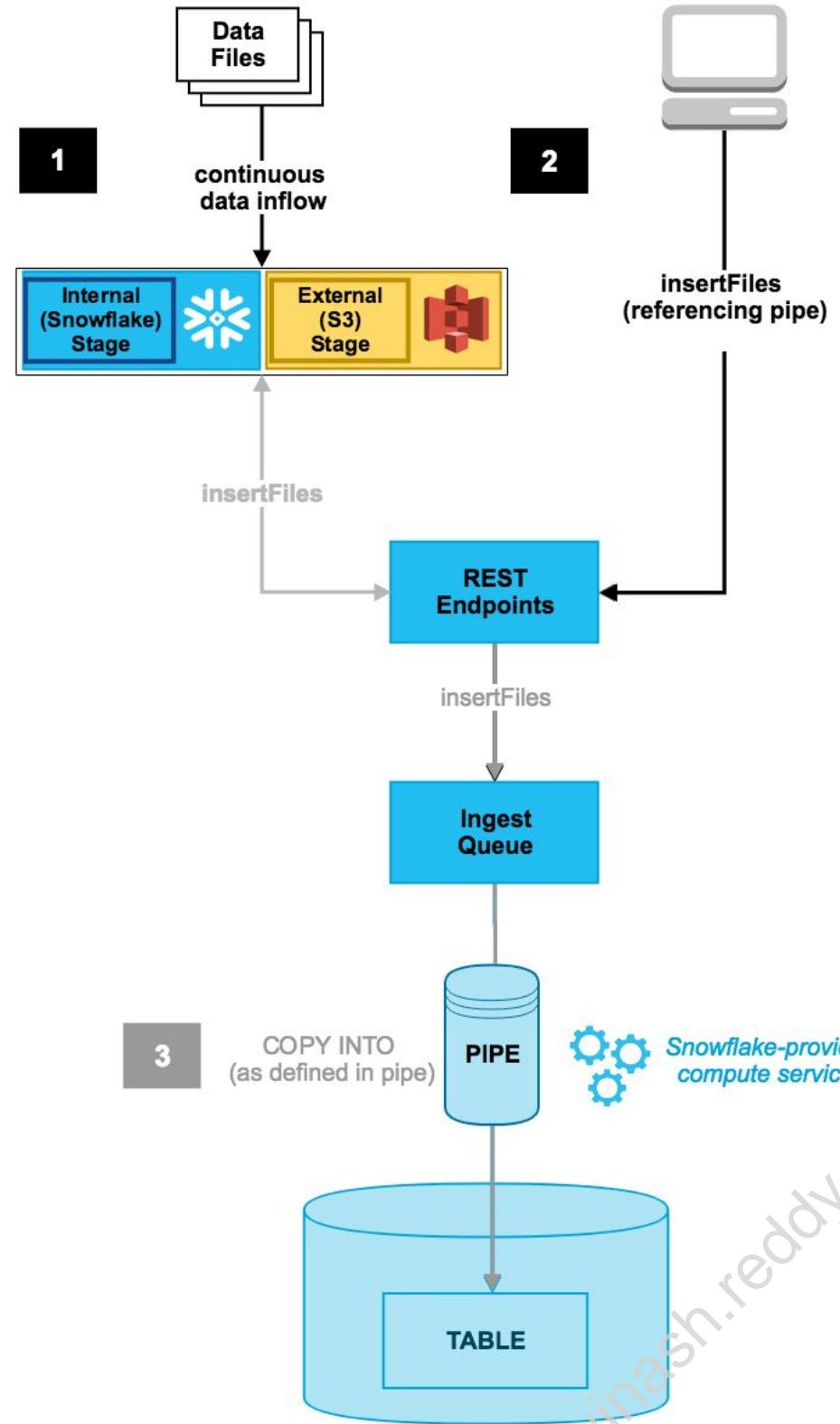
- Migration from traditional data sources
- Transaction boundary control
 - BEGIN / START TRANSACTION / COMMIT / ROLLBACK
- Independently scale compute resources for different ingestion workloads

Snowpipe (CONTINUOUS)

- Ingestion from modern data sources
- Continuously generated data is available for analysis in seconds
- No scheduling (with auto-ingest)
- Serverless model with no user-managed virtual warehouse needed



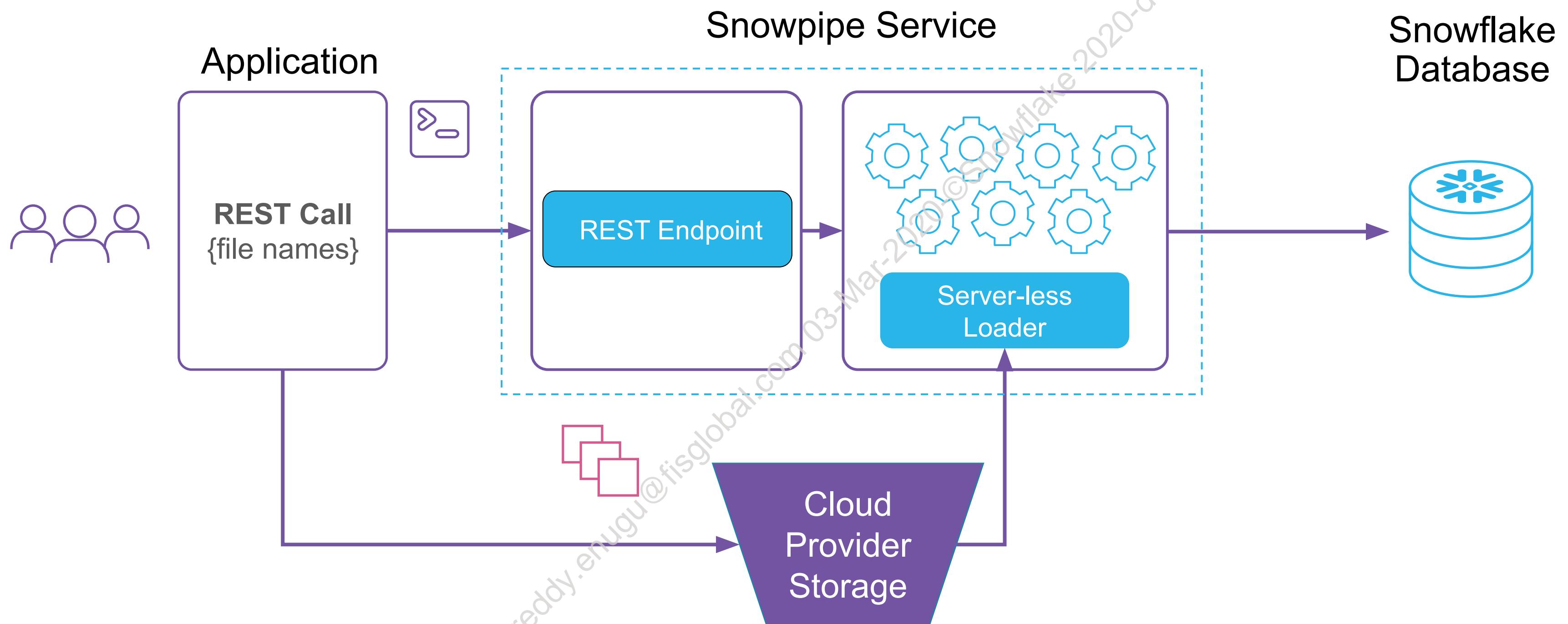
PIPE



- Named object contains a `COPY` statement used by Snowpipe
 - Source stage for data files
 - Target table
- Loads data into tables continuously from an ingestion queue
- Can be paused/resumed, return status
- **Best Practice:** Size files between 10MB and 100MB (compressed) when staging files for ingest with Snowpipe

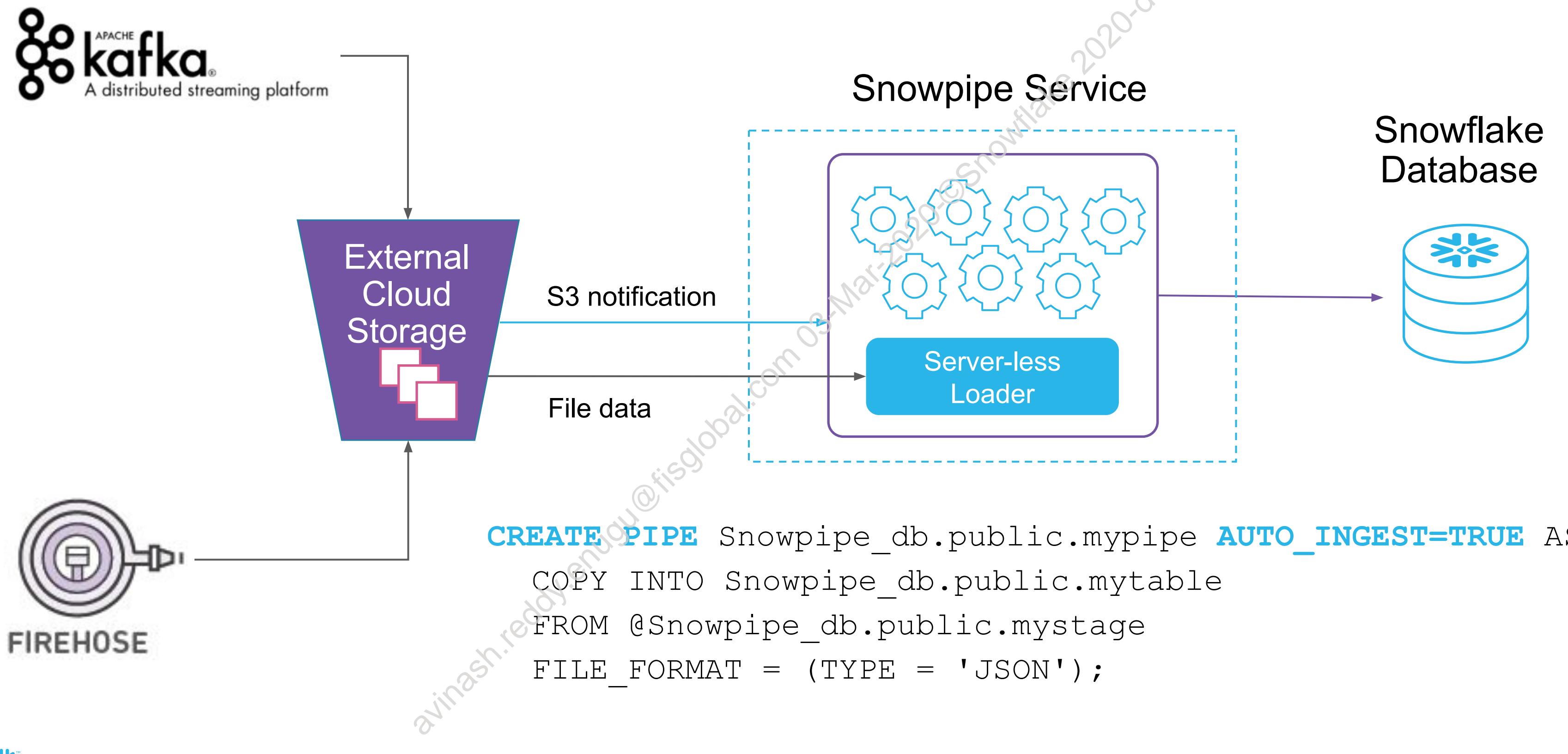


SNOWPIPE REST API



```
CREATE PIPE IF NOT EXISTS mypipe AS COPY INTO mytable FROM @mystage;
```

AUTO-INGEST

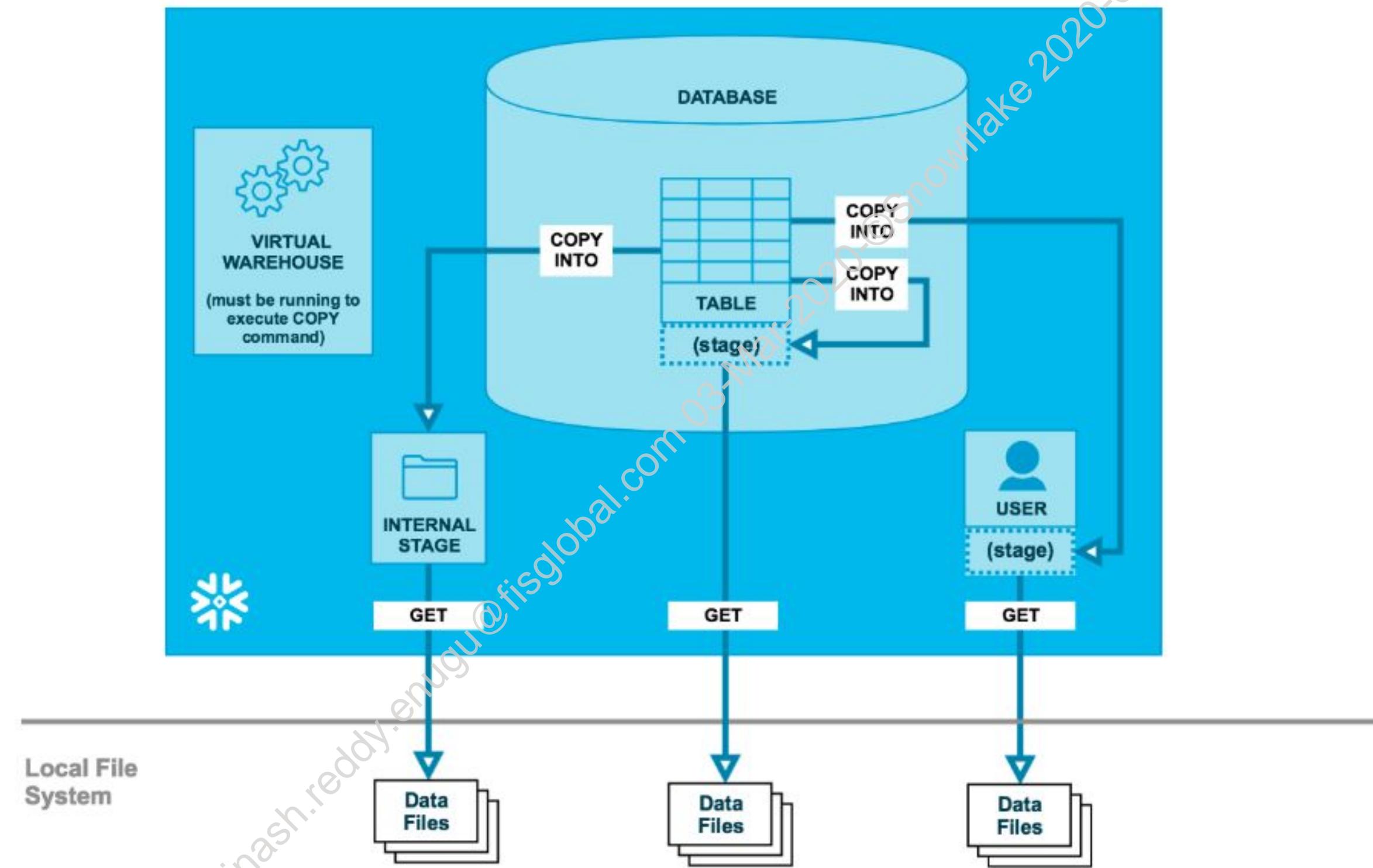


Data Unloading

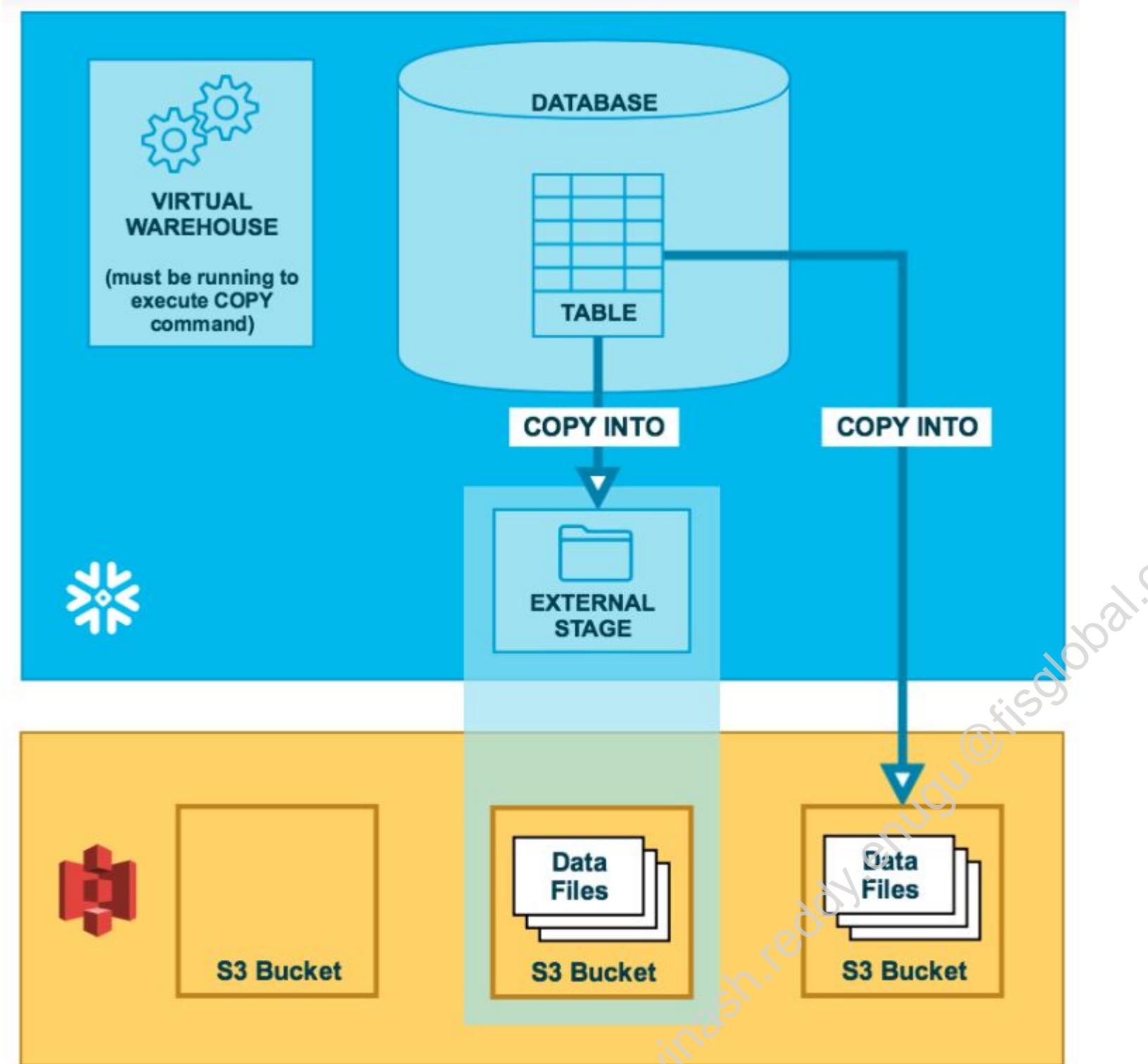
avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



UNLOAD TO LOCAL FILE SYSTEM



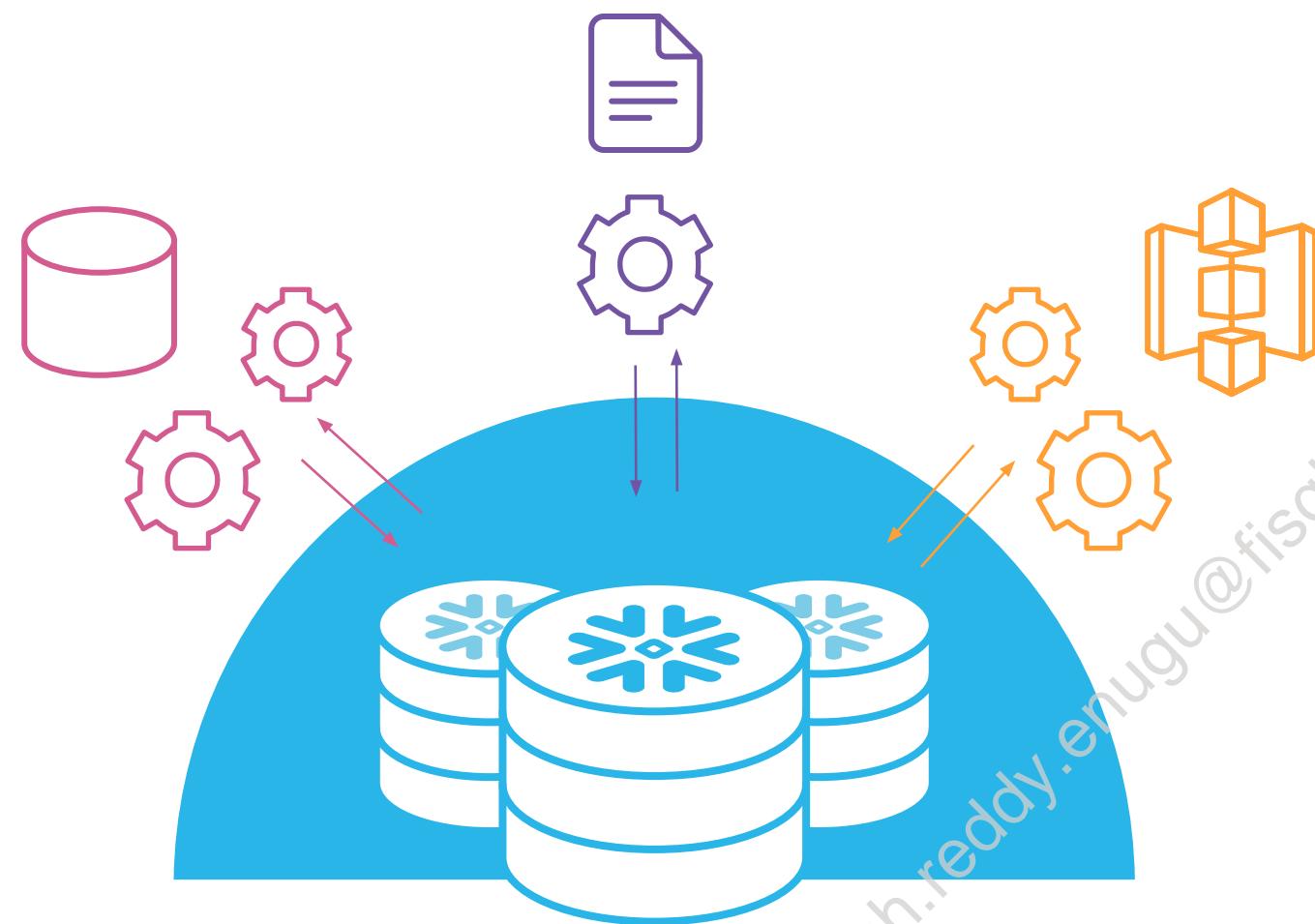
UNLOAD TO CLOUD



- Use COPY INTO
- Recommend using external named stage
- Can also unload directly by specifying the URI and any necessary credentials

DATA UNLOADING

CONSIDERATIONS



- File formats
- Empty strings vs NULL values
- Unloading relational table to semi-structured format
- File splitting
- Compression

FILE FORMATS

- Any flat, delimited plain text format (CSV, TSC, etc.)
- JSON:
 - Data must be unloaded from a column of VARIANT data type

```
-- Unload the data to a file in a stage
copy into @mystage
from (select object_construct('id', id, 'first_name', first_name, 'last_name', last_name, 'city', city, 'state', state) f
file_format = (type = json);
```

- Parquet:
 - Use a SELECT statement to unload a table to Parquet as multiple columns

```
copy into @mystage/myfile.parquet from (select id, name, start_date from mytable)
file_format=(type='parquet')
header = true;
```



EMPTY STRINGS VS NULL VALUES

Handle NULLs and empty strings appropriately on unload

- Enclose strings in double or single quotes:

```
FIELD_OPTIONALLY_ENCLOSED_BY = 'character' | NONE
```

- Determine how empty fields are handled:

```
EMPTY_FIELD_AS_NULL = TRUE | FALSE
```

- Convert SQL NULL values:

```
NULL_IF = ( 'string1' [ , 'string2' ... ] )
```



FILE SPLITTING

- Using the option **SINGLE**, control whether the unload process will create a single file, or multiple files
- Set **MAX_FILE_SIZE** to handle files larger than the default 16MB
 - The file size limit for single-file mode is 5GB

```
COPY INTO @mystage/myfile.csv.gz FROM mytable  
FILE_FORMAT = (type=csv compression='gzip')  
SINGLE=TRUE MAX_FILE_SIZE=4900000000
```



COMPRESSION

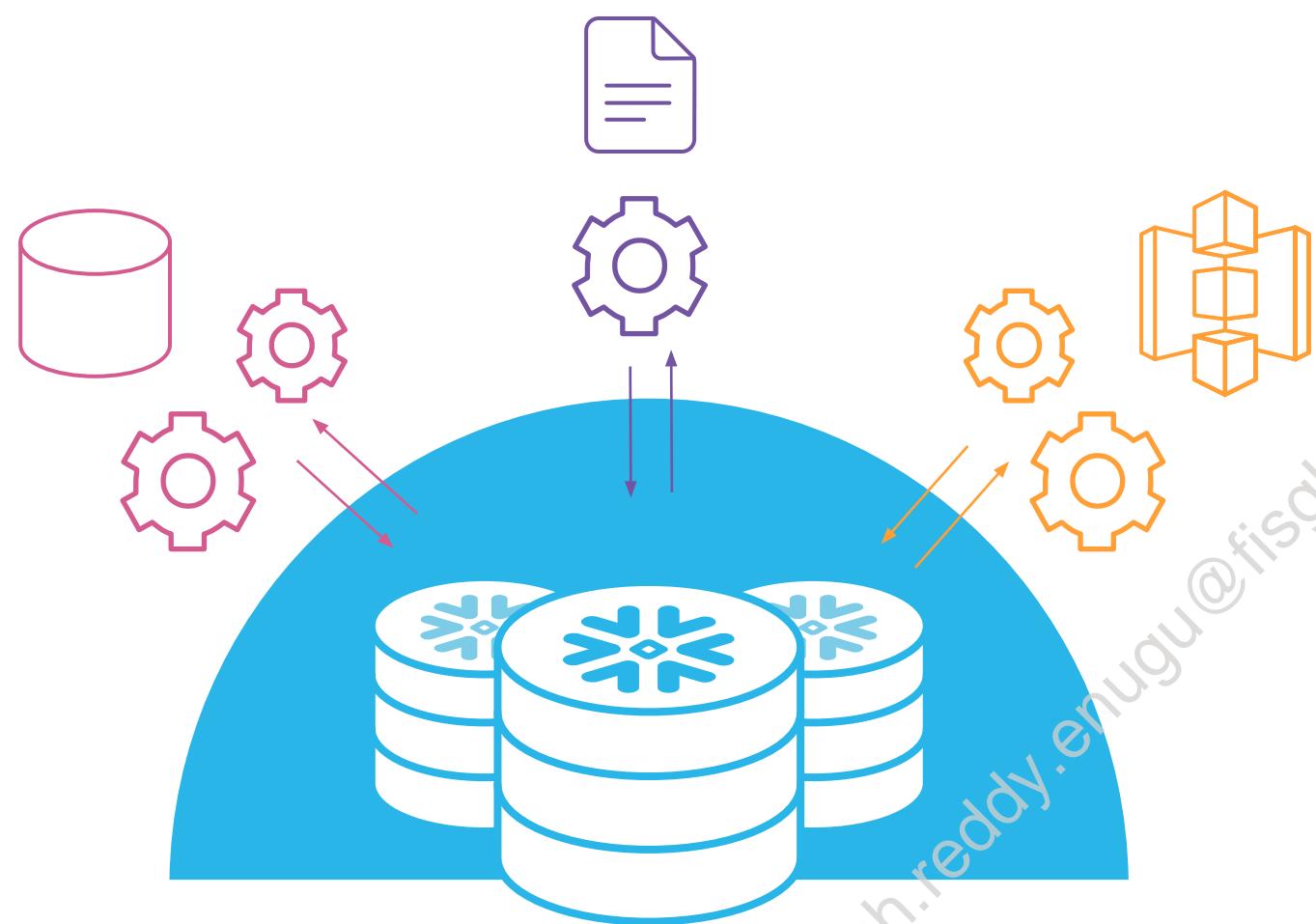
By default, unloaded files are compressed using gzip

- Type of compression can be controlled with the COMPRESSION option
 - gzip
 - bzip2
 - Brotli (preview)
 - Zstandard (preview)



DATA UNLOADING

ADDITIONAL FUNCTIONALITY



- Using queries in unload
- Listing files in stage
- Removing files from stage



UNLOAD WITH SELECT

- SELECT queries in COPY statements support the full syntax of Snowflake SQL queries

```
COPY INTO @my_stage  
FROM (SELECT column1, column2 FROM my_table)  
FILE_FORMAT = (FORMAT_NAME = 'my_format');
```

- JOIN clauses enable downloading data from multiple tables

```
COPY INTO @my_stage  
FROM (SELECT name, id1  
      FROM my_table1  
      JOIN my_table2 ON id1 = id2)  
FILE_FORMAT = (FORMAT_NAME = 'my_format');
```



LIST COMMAND

List the files stored in a stage object

```
LIST @%monthly_sales_agg;
```

name	size	md5	last_modified
data	144	055c5ee1d38271580a6a925ad1a69d45	Sun, 2 Dec 2018 17:48:22 GMT
data_0_0_0.csv	320	aefac11b7e3b543809988ea3f90924c5	Sun, 2 Dec 2018 18:15:06 GMT
data_0_0_0.csv.gz	144	0f01461eab63268987260ae6622c3ec6	Sun, 2 Dec 2018 18:15:19 GMT
data_0_0_0.json.gz	208	72c0f0d577b590233cda48464184204b	Sat, 1 Dec 2018 16:44:43 GMT
monthly_sales_dec_0_0_0.json.gz	208	37888eaf7698a6ac9eee178eaa509e57	Sat, 1 Dec 2018 16:47:26 GMT



REMOVE COMMAND

Removes files that have been stored in an **internal** stage

```
REMOVE @%monthly_sales_agg PATTERN='.*data*.';
```

name	result
data_0_0_0.json.gz	removed
data_0_0_0.csv.gz	removed
data	removed
data_0_0_0.csv	removed

```
LIST @%monthly_sales_agg;
```

name	size	md5	last_modified
monthly_sales_dec_0_0_0.json.gz	208	37888eaf7698a6ac9eee178eaa509e57	Sat, 1 Dec 2018 16:47:26 GMT



Module 4: Data Movement

Exercise 4.3: Unload Structured Data

30 minutes

- Unload a pipe-delimited file
- Unload part of a table
- JOIN and unload



Module 5

Snowflake Functions

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



MODULE AGENDA

- Snowflake Functions Overview
- High-Performing Functions
- User-Defined Functions
- Stored Procedures

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



Snowflake Functions Overview

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



SUPPORTED FUNCTION TYPES

Type	Description
Scalar	Takes a single value, returns a single value.
Aggregate	Performs a calculation on a set of values, and returns a single value.
Window	Aggregate functions that operate on a subset of rows within the input rows.
Analytic	Performs an aggregate calculation on a set of rows; can return multiple rows for each group.
Table	Produces a collection of rows (either a nested table or a vararray) that can be queried like a physical database table. Use in the FROM clause of a query.
System	Used to execute action in the system, or return information about the system.
User-Defined	Accept parameters, perform an action, and return the result. The result can either be a scalar, or a result set.



SCALAR FUNCTIONS

Performs a calculation on a single value, and returns a single value

Value
1
8
12
7

```
SELECT CAST(value TO DECIMAL(5,2))  
FROM mytable  
WHERE value < 10;
```

Value
1.00
8.00
7.00



AGGREGATE FUNCTIONS

Performs a calculation on a set of values, and returns a single value

Value
1
8
12
7

```
SELECT SUM(value)  
FROM mytable;
```

Value
28

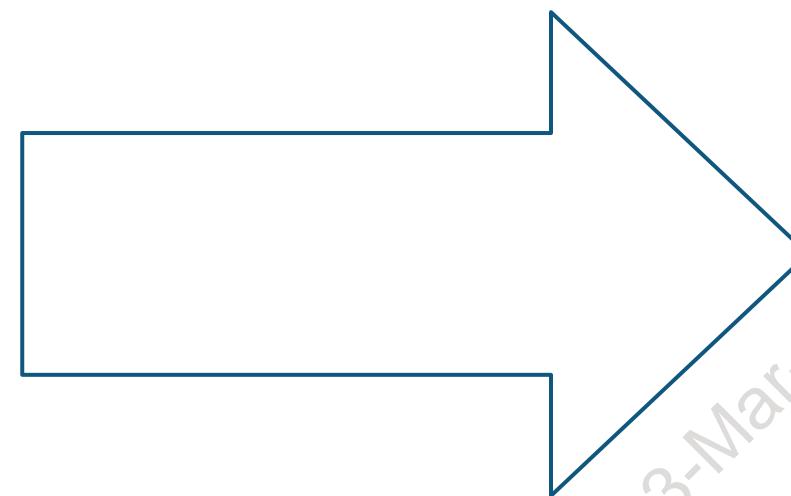
- Often used with the GROUP BY clause of the SELECT statement
- Except for COUNT, aggregate functions ignore null values.



WINDOW FUNCTIONS

Aggregate functions that work on a subset (window) of the input rows

DATE	SALES
2019-08-01	8.73
2019-08-02	129.95
2019-08-03	13.75
2019-08-04	21.75
2019-08-05	115.87



DATE	SALES	MTD
2019-08-01	8.73	8.73
2019-08-02	129.95	136.68
2019-08-03	13.75	152.43
2019-08-04	21.75	174.18
2019-08-05	115.87	290.05

```
SELECT date, sales, SUM(sales)
OVER(ORDER BY date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
AS mtd FROM aug_sales;
```



ANALYTIC FUNCTIONS

Function	Window Frame Support?
CUME_DIST	YES
DENSE_RANK	YES
FIRST_VALUE	YES
LAG	
LAST_VALUE	YES
LEAD	
NTH_VALUE	YES
NTILE	
PERCENT_RANK	YES
RANK	YES
ROW_NUMBER	
WIDTH_BUCKET	



TABLE FUNCTIONS

- Take scalar expressions as input
- Return a set of rows instead of a single scalar value
- Appear in the FROM clause

```
SELECT event_timestamp AS time, user_name  
FROM TABLE (login_history_by_user())  
ORDER BY event_timestamp;
```

	TIME	USER_NAME
2019-08-12	08:11:00.166 -0700	DEBORAH
2019-08-12	09:14:08.128 -0600	DAVE
2019-08-12	08:09:17.004 -0800	JOON



SYSTEM FUNCTIONS

CONTROL

- ABORT_SESSION
- ABORT_TRANSACTION
- CANCEL_QUERY
- CANCEL_ALL_QUERIES
- PIPE_FORCE_RESUME
- TASK_DEPENDENTS_ENABLE
- WAIT

INFORMATION

- CLUSTERING_DEPTH
- CLUSTERING_INFORMATION
- CLUSTERIN_RATIO
- CURRENT_USER_TASK_NAME
- PIPE_STATUS
- STREAM_HAS_DATA
- WHITELIST



USER-DEFINED FUNCTIONS

- Custom functions written to simplify repetitive operations
- Use when appropriate system function does not exist
- SQL or Javascript
- Do not support DDL or DML

Covered in more detail later...



Module 5: Snowflake Functions

Exercise 5.1: Snowflake Function Library

25 minutes

Tasks:

- Scalar Functions
- Regular and Window Aggregate Functions
- Table and System Functions



High-Performing Functions

avinash.reddy.enugu@fisglobal.com 83-Mar-2020 ©Snowflake 2020-do-not-copy



APPROXIMATION

- Snowflake uses HyperLogLog for a set of aggregate functions to estimate the approximate number of distinct values in a data set.
- HyperLogLog is a state-of-the-art cardinality estimation algorithm, capable of estimating distinct cardinalities of trillions of rows with an average relative error of a few percent.

What does that mean?

```
SELECT COUNT(cust_orders),  
       COUNT(distinct cust_orders),  
       APPROX_COUNT_DISTINCT(cust_orders)  
  FROM customers;
```

Count (cust_orders)	Count (distinct cust_orders)	HLL (cust_orders)
2205	2205	2220

COUNTING LARGE DISTINCT DATASETS

COUNT (DISTINCT x)

- Slower
- Scalable
- Exact

```
SELECT COUNT(DISTINCT o_orderkey)  
FROM orders;
```

COUNT(DISTINCT O_ORDERKEY)

15000000

APPROX_COUNT_DISTINCT (x)

- Much faster
- Approximate but within ~1.62% error
- Less memory intensive for column with a large number of distinct values

```
APPROX_COUNT_DISTINCT(o_orderkey)  
FROM orders;
```

APPROX_COUNT_DISTINCT(O_ORDERKEY)

145660677



ESTIMATION VS EXACT

MEDIAN (x)

- Slower
- Scalable
- Exact

```
35 select median(ss_sales_price), ss_store_sk  
36 from snowflake_sample_data.tpcds_sf10tcl.store_sales  
37 group by ss_store_sk;
```

Results Data Preview

Query ID	SQL	Time
✓	35 select median(ss_sales_price), ss_store_sk 36 from snowflake_sample_data.tpcds_sf10tcl.store_sales 37 group by ss_store_sk;	18m59s

APPROX_PERCENTILE (x)

- t-Digest Algorithm
- Much faster than MEDIAN
- The estimation uses a constant amount of space regardless of the size of the input.

```
42 select approx_percentile(ss_sales_price, 0.5), ss_store_sk  
43 from snowflake_sample_data.tpcds_sf10tcl.store_sales  
44 group by ss_store_sk;  
45  
46 select median(ss_sales_price)
```

Results Data Preview

Query ID	SQL	Time
✓	42 select approx_percentile(ss_sales_price, 0.5), ss_store_sk 43 from snowflake_sample_data.tpcds_sf10tcl.store_sales 44 group by ss_store_sk; 45 46 select median(ss_sales_price)	3m2s

FREQUENCY ESTIMATION

APPROX_TOP_K (<expr> [, <k> [, <counters>]])

Returns estimated frequency of most frequent values

APPROX_TOP_K_ACCUMULATE (<expr> , <counters>)

Skips the final estimation step and returns the Space-Saving state at the end of an aggregation

APPROX_TOP_K_COMBINE (<state> [, <counters>])

Combines (i.e. merges) input states into a single output state

APPROX_TOP_K_ESTIMATE (<state> [, <k>])

Computes a cardinality estimate of a Space-Saving state produced by APPROX_TOP_K_ACCUMULATE and APPROX_TOP_K_COMBINE .



SIMILARITY ESTIMATION

MINHASH(<k> , [distinct] expr+)

Estimates the approximate similarity between two or more data sets.

MINHASH_COMBINE([distinct] <state>])

Combines input MinHash states into a single MinHash output state. This Minhash state can then be input to the APPROXIMATE_SIMILARITY function to estimate the similarity with other MinHash states.

APPROXIMATE_SIMILARITY ([distinct] <expr> [, ...])

Returns an estimation of the similarity (Jaccard index) of inputs based on their MinHash states.

Returns a value between 0.0 (no similarity) to 1.0 (identical).

Equivalent to APPROXIMATE_JACCARD_INDEX.



Module 5: Snowflake Functions

Exercise 5.2: High-Performing Functions

40 minutes

Tasks:

- Approximate Count Functions
- Percentile Estimation Functions

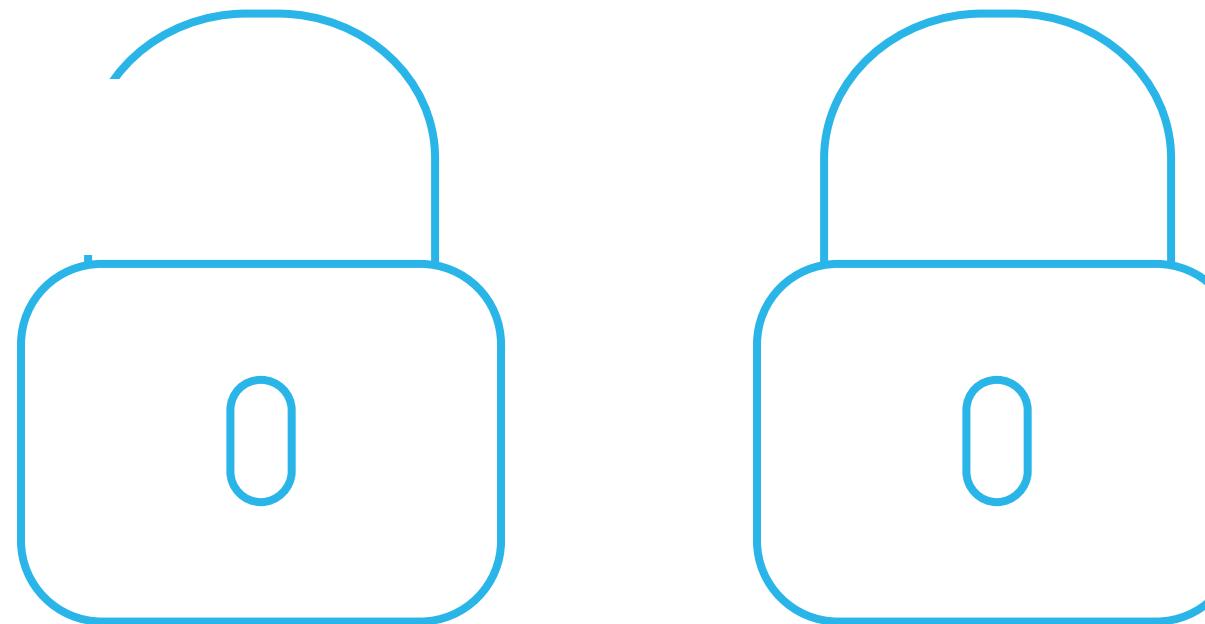


User-Defined Functions

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



USER-DEFINED FUNCTIONS



SQL | JAVASCRIPT

- Perform custom operations that are not available through the built-in functions
- SQL and JavaScript supported
- No DDL/DML support
- Can be unsecure or secure
- Return a singular scalar value or, if defined as a table function, a set of rows

SQL UDF EXAMPLE

1. Create the table and data to use

number_sold	wholesale_price	retail_price
3	10.00	20.00
5	100.00	200.00

2. Create the UDF

```
CREATE FUNCTION profit() RETURNS NUMERIC(11,2) AS  
$$  
SELECT SUM((retail_price - wholesale_price) * number_sold)  
FROM purchases  
$$;
```

1. Call the UDF

```
SELECT profit();
```

PROFIT()
\$530.00



JAVASCRIPT UDF

```
create function variant_nulls(v variant)
    returns variant
language javascript
as '
if (v === undefined) {
    return "input SQL null";
} else if (v === null) {
    return "input variant null";
} else if (v === 1) {
} else if (v === 2) {
} else if (v === 3) {
    return {
        key1 : undefined,
        key2 : null
    };
} else {
    return v;
}
';
```



Stored Procedures

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



STORED PROCEDURES



JAVASCRIPT

- Allow procedural logic and error handling that straight SQL does not support
- Implemented through JavaScript and, optionally (commonly), SQL.
- JavaScript provides the control structures
- SQL is executed within the JavaScript by calling functions in an API
- Argument names are case-insensitive in the SQL portion of stored procedure code case-sensitive in the JavaScript portion

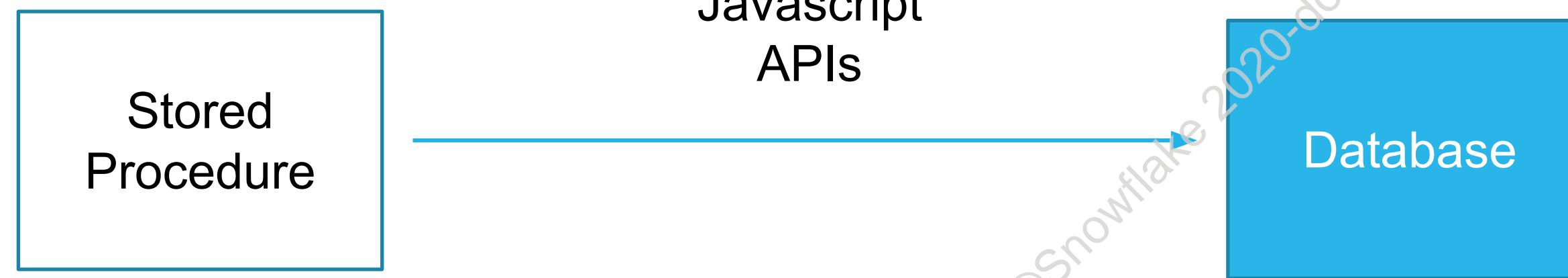


COMPONENTS OF STORED PROCEDURE

1. Create stored procedure
 2. Data type returned by stored procedure
 3. Language for the stored procedure. Only Javascript is supported
 4. SQL statements the stored procedure will execute
 5. Invoke the stored procedure
- ```
CREATE PROCEDURE
stproc1(float_param1 FLOAT)
RETURNS STRING
LANGUAGE JAVASCRIPT
STRICT
AS
$$ -- marks the beginning and end of the JavaScript code
try {
 snowflake.execute (
 {sqlText: "INSERT INTO stproc_test_table1 (num_col1) VALUES
 (" +FLOAT_PARAM1 + ")"});
 return "Succeeded." // Return a success/error indicator.
} catch (err) {
 return "Failed: " + err; // Return a success/error indicator.
}
$$;
call stproc1(5.14::FLOAT);
```



# HOW IT WORKS



## Snowflake-provided Javascript APIs

- Javascript objects and methods
- Error handling and procedural logic

## SQL and database access/operations

- Embed SQL codes in the Javascript
- The SQL codes execute database operations
- Migrate other databases' stored procedure (SQL) codes by embedding the SQL in Javascript



# STORED PROCEDURE SQL

## EXECUTED THROUGH API OBJECTS

| Object Class                | Method                                                                       |
|-----------------------------|------------------------------------------------------------------------------|
| <b>Snowflake Object</b>     | Create statement object<br>Create resultset object by executing SQL          |
| <b>SQL Statement Object</b> | Create resultset object by executing a prepared statement<br>Access metadata |
| <b>ResultSet Object</b>     | Hold query result set<br>Navigate result set                                 |



# STORED PROCEDURES VS UDFS

## STORED PROCEDURE

- Called as a single statement
  - Call statement does not handle returned values
  - JS in a calling Stored Procedure can handle return values from a called Stored Procedure
- **MAY** return a value
- Can access database objects and issue SQL statements and nested queries via an API

## USER-DEFINED FUNCTION

- Called inside another statement
- **MUST** return a value
- DDL and DML operations not permitted



# WHEN TO USE WHAT

## Stored Procedure

- When migrating stored procedures
- When you need to perform database operations
- For administrative tasks on database objects

## User-Defined Function

- When migrating UDFs
- When you need a function to work as part of a SQL statement and it must return a value
- When you want to return a value for each row ,and each row provides a value



# Module 5: Snowflake Functions

## Exercise 5.3: User-Defined Functions and Stored Procedures

30 minutes

### Tasks:

- JavaScript User-Defined Functions
- SQL User-Defined Functions
- Creating Stored Procedures



# Module 6

# Managing Security

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Security Overview
- Authentication Methods
- Data Encryption
- Network Security

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



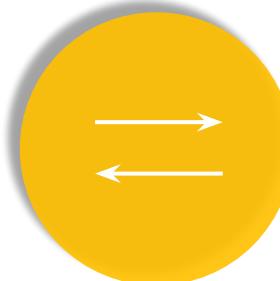
# Security Overview

avinash.reddy.enugu@fisglobal.com 03-May-2020 ©Snowflake 2020-do-not-copy



# SECURITY AT A GLANCE

ALL EDITIONS COMBINED



Access



Authentication



Authorization



Data Protection



Infrastructure



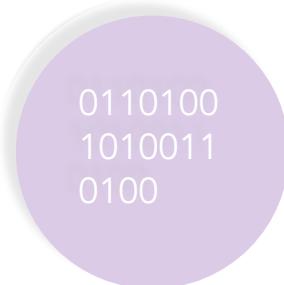
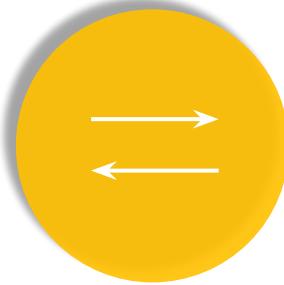
Snowflake operational controls

- NIST 800-53
- SOC2 Type 2
- SOC1 Type II
- ISO/IEC 27001:2013
- HIPAA
- PCI
- FedRAMP



# ACCESS

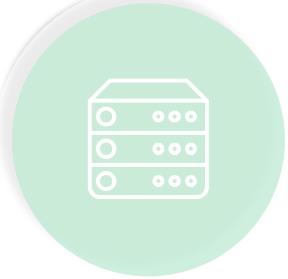
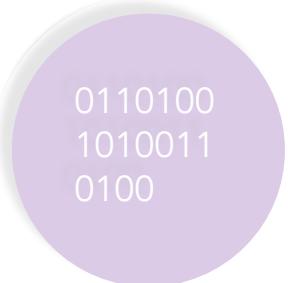
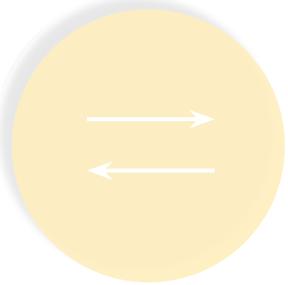
CAN YOU GET TO THE SNOWFLAKE ACCOUNT?



- All communication secured & encrypted
- TLS 1.2 encryption for all client communications
- IP whitelisting
- Support for AWS PrivateLink

# AUTHENTICATION

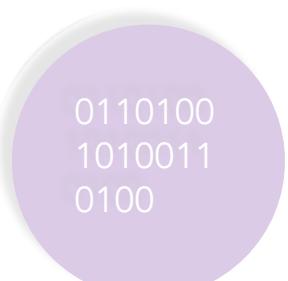
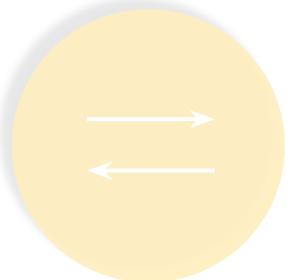
WHO ARE YOU?



- Password policy enforcement
  - 8-256 characters
  - At least one digit
  - At least one upper-case character
  - At least one lower-case character
- Multi-factor authentication can be enabled on a per-user basis
- Federated authentication (SAML 2.0)
- Support for Key-Pair authentication

# AUTHORIZATION

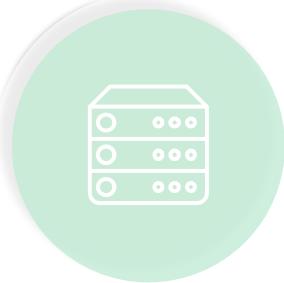
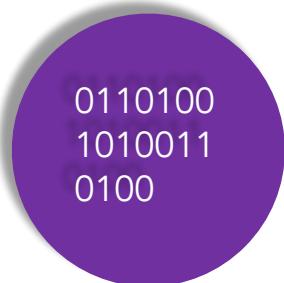
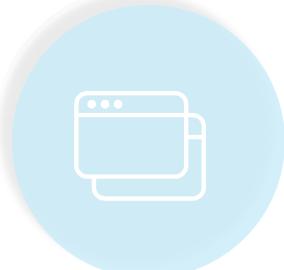
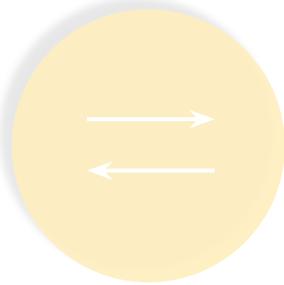
WHAT CAN YOU DO?



- Flexible and granular authorization controls
- Role-based access control for granular permissions
- Role-based access control for data and actions
- Secure views and UDFs to protect information access

# DATA PROTECTION

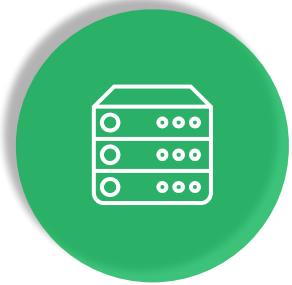
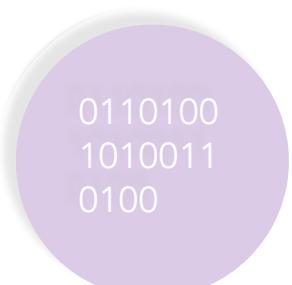
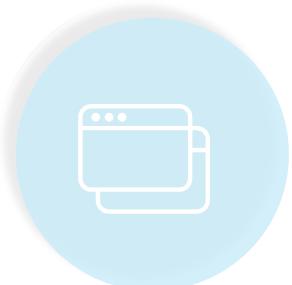
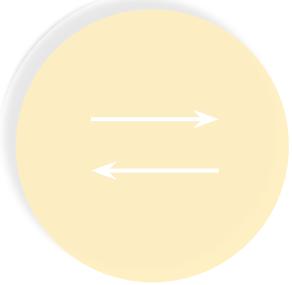
IS YOUR DATA SAFE?



- Encrypted at rest and in motion
  - Hierarchical key model rooted in Cloud HSM
  - Automatic key rotation
  - Periodic re-keying
- Tri-Secret Secure (BYOK)
- Time Travel 1-90 days protection
- Failsafe for additional protection

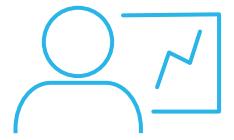
# INFRASTRUCTURE

IS YOUR INFRASTRUCTURE SECURE?

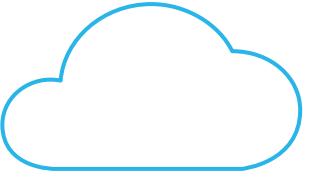


- Cloud provider's physical security
- Cloud provider's redundancy
- Limitless elasticity
- Regional data centers - US, EU, AP

# LAYERED SECURITY



1

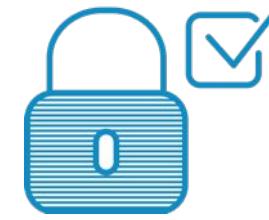


## Access

Restrict access to specified IP address or range

Optionally: Restrict via Secure Private Network

2



## Authentication

Authenticate users using a Password, MFA, or SSO

3



## Authorization

Restrict access to specific objects using Roles and privileges

4



## Data Encryption

Protect customer data using AES 256 bit encryption, and periodic re-keying

# ACCESS FLOW STEPS

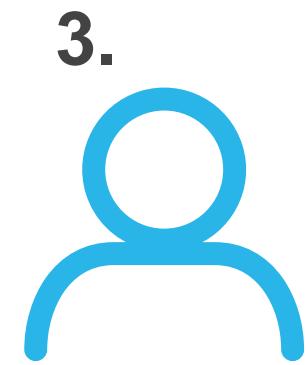
## CONNECTION REQUEST



**Enforce network policy set on the account**



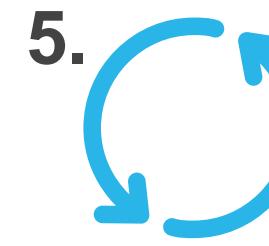
**Verify that the account is not locked / disabled**



**Verify that the user is not locked / disabled**



**Resolve the account + user from the given request**



**Perform basic authentication checks**

# Authentication Methods

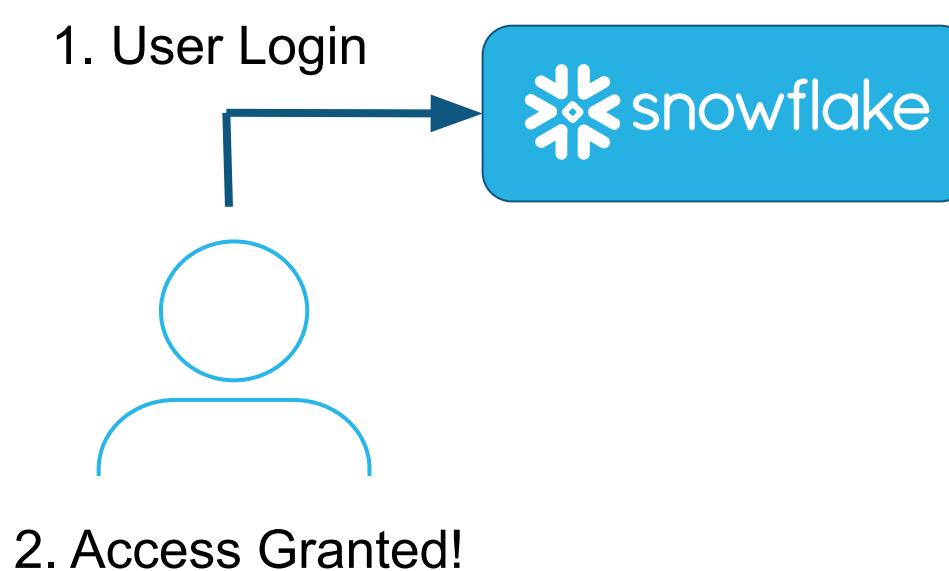
avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# AUTHENTICATION METHODS

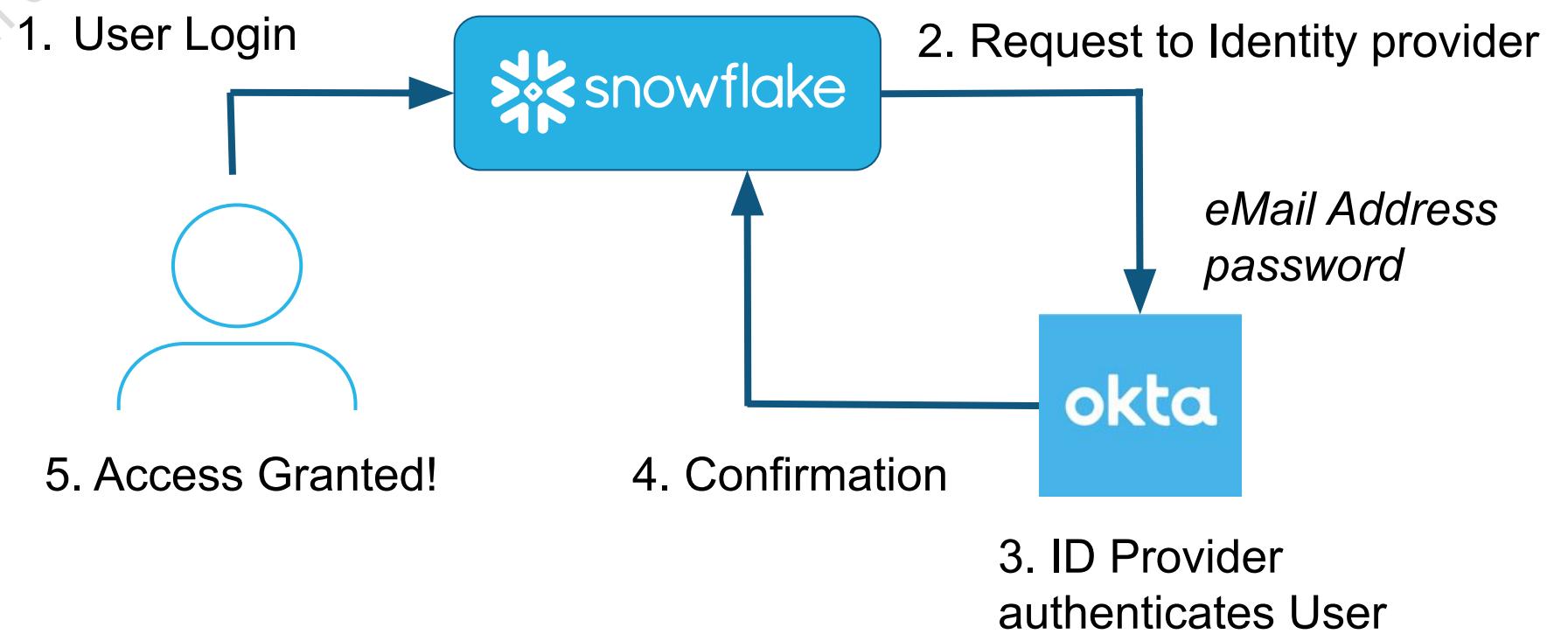
## Snowflake Username / Password

Verify that the password given matches the one associated with the user



## Federated Authentication

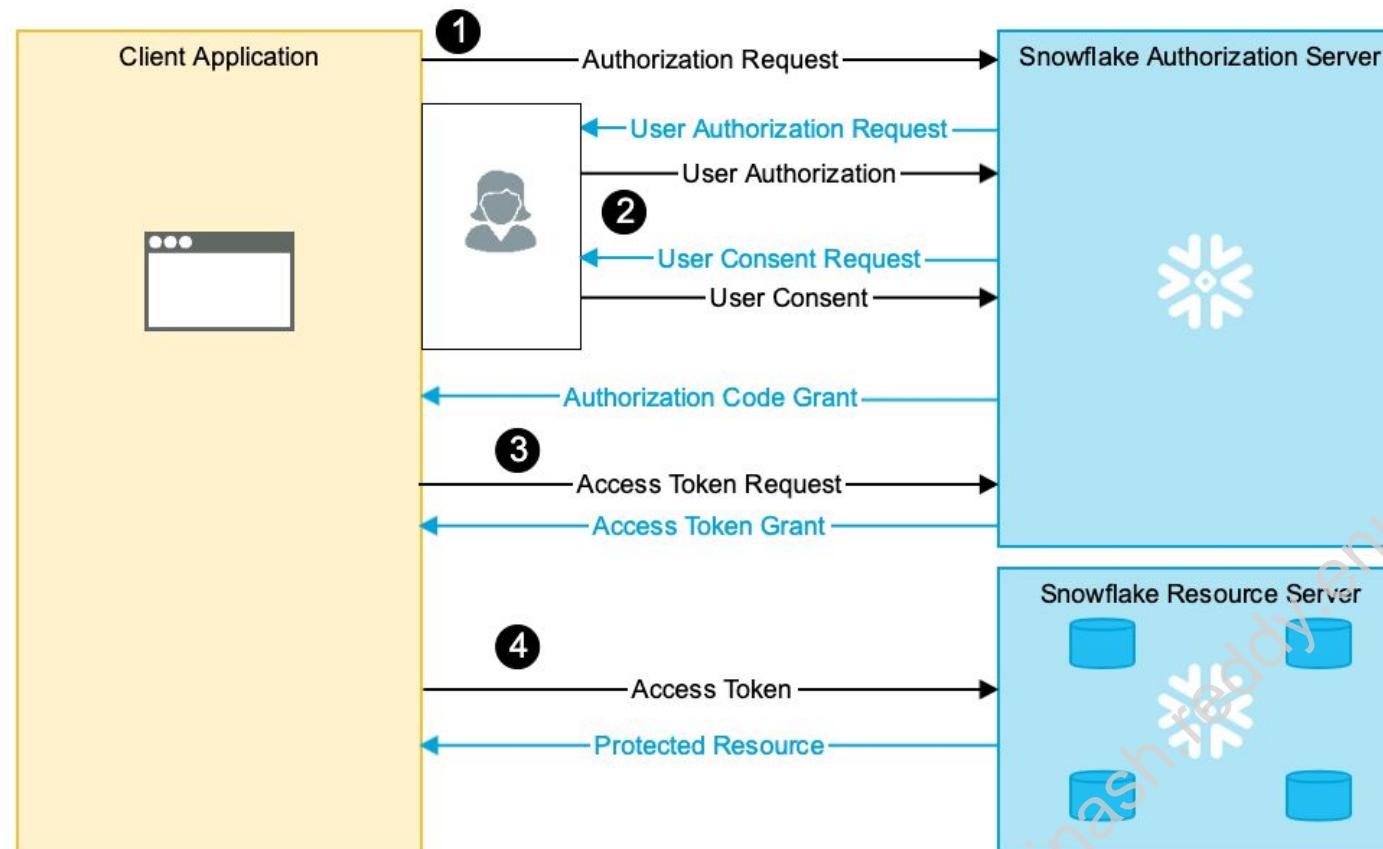
Users authenticate through an external, SAML 2.0-compliant identity provider (IdP)



# AUTHENTICATION METHODS

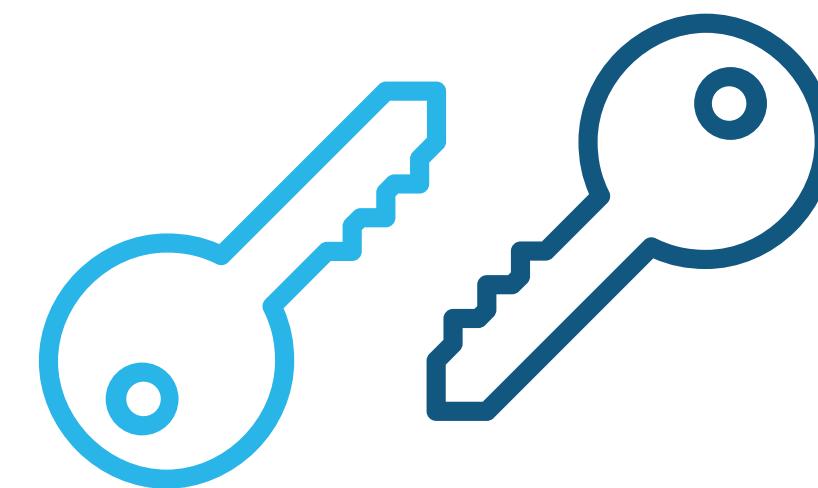
## OAuth

Allows for securing authorizations on behalf of a user from a third party service / application



## KeyPair

Keypair-based authentication with JSON Web Token (JWTs) signed using a public/private key pair with RSA encryption



Best Practice: Use key-pair authentication for use in non-interactive, headless connections to Snowflake

# BASIC AUTHENTICATION PASSWORD POLICY



## Absolute minimum:

- At least 8 characters
- At least one digit
- At least one upper/lower case

## Strongly recommended:

- Never re-use passwords
- Multiple random mixed-case letters and special characters
- Enroll in MFA; recommended minimum requirement for ACCOUNTADMIN



# MULTI-FACTOR AUTHENTICATION

|                          |                                                                                                                                                                  |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose:</b>          | Provides increased login security in <b>all</b> Snowflake editions                                                                                               |
| <b>Method:</b>           | <ul style="list-style-type: none"><li>● <b>Password plus Duo Security Code</b> on Smartphone</li><li>● Users enroll themselves using the web interface</li></ul> |
| <b>Result:</b>           | Any login attempt (Web UI, CLI or application) requires user credentials <u>plus</u> Duo-generated temporary passcode                                            |
| <b>Absolute Minimum:</b> | Snowflake <b>strongly</b> recommends users with ACCOUNTADMIN must use MFA, but ideally all logins                                                                |
| <b>Prerequisites:</b>    | Smartphone or similar (IOS, Android, Windows, etc.) plus valid phone number with Duo App installed                                                               |



# AUTHENTICATION METHODS BY EDITION

| Snowflake Edition | Basic Auth | Key Pair Auth | MFA | SSO | OAuth |
|-------------------|------------|---------------|-----|-----|-------|
| Standard          | ✓          | ✓             | ✓   | ✓   | ✓     |
| Premium           | ✓          | ✓             | ✓   | ✓   | ✓     |
| Enterprise        | ✓          | ✓             | ✓   | ✓   | ✓     |
| ESD               | ✓          | ✓             | ✓   | ✓   | ✓     |
| VPS               | ✓          | ✓             | ✓   | ✓   | ✓     |



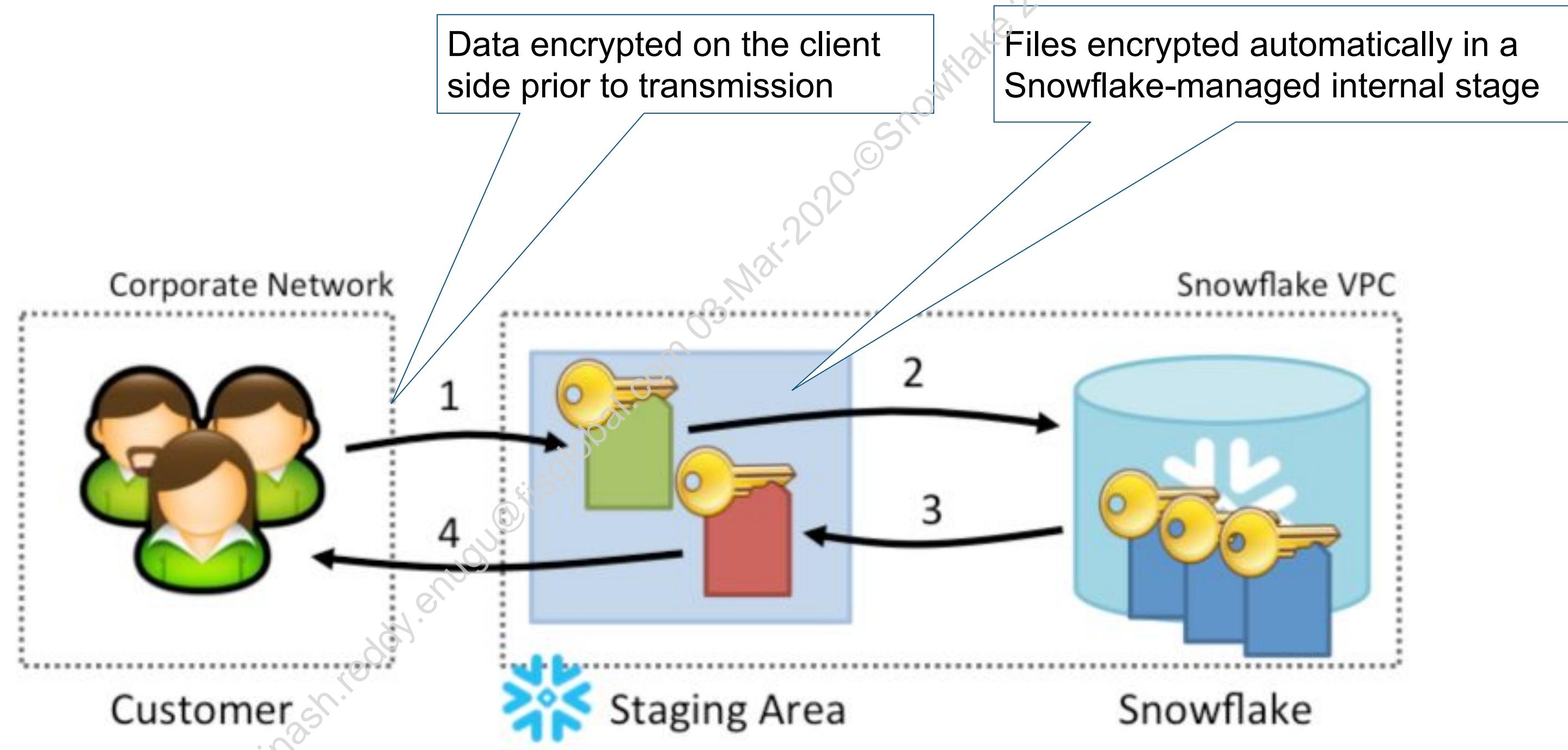
# Data Encryption

avinash.reddy.enugu@fisglobal.com 03 Mar 2020 ©Snowflake 2020-do-not-copy



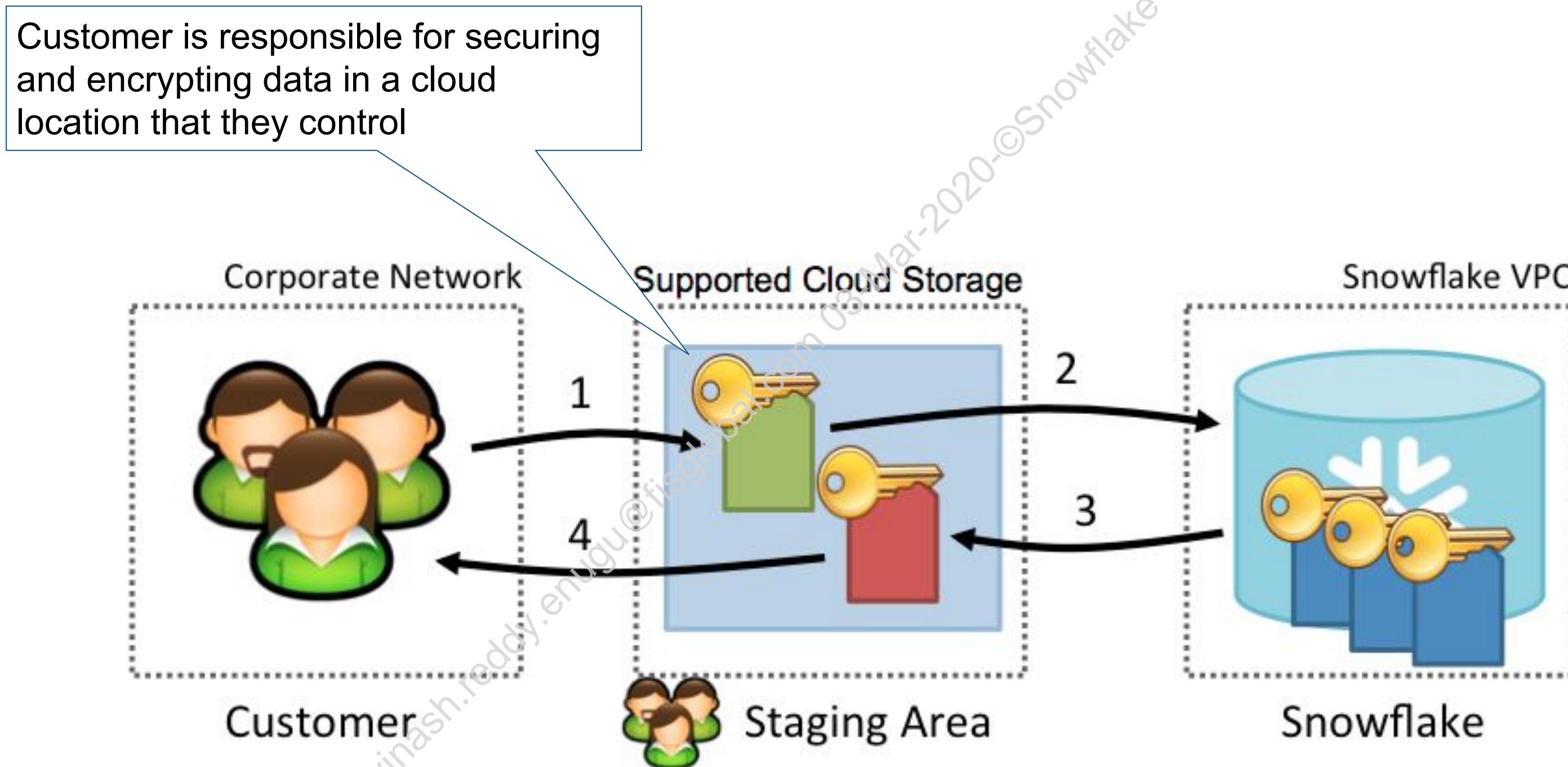
# SNOWFLAKE-PROVIDED STAGING AREA

Automatic E2EE



# CUSTOMER-PROVIDED STAGING

Encryption/Security Managed by Customer

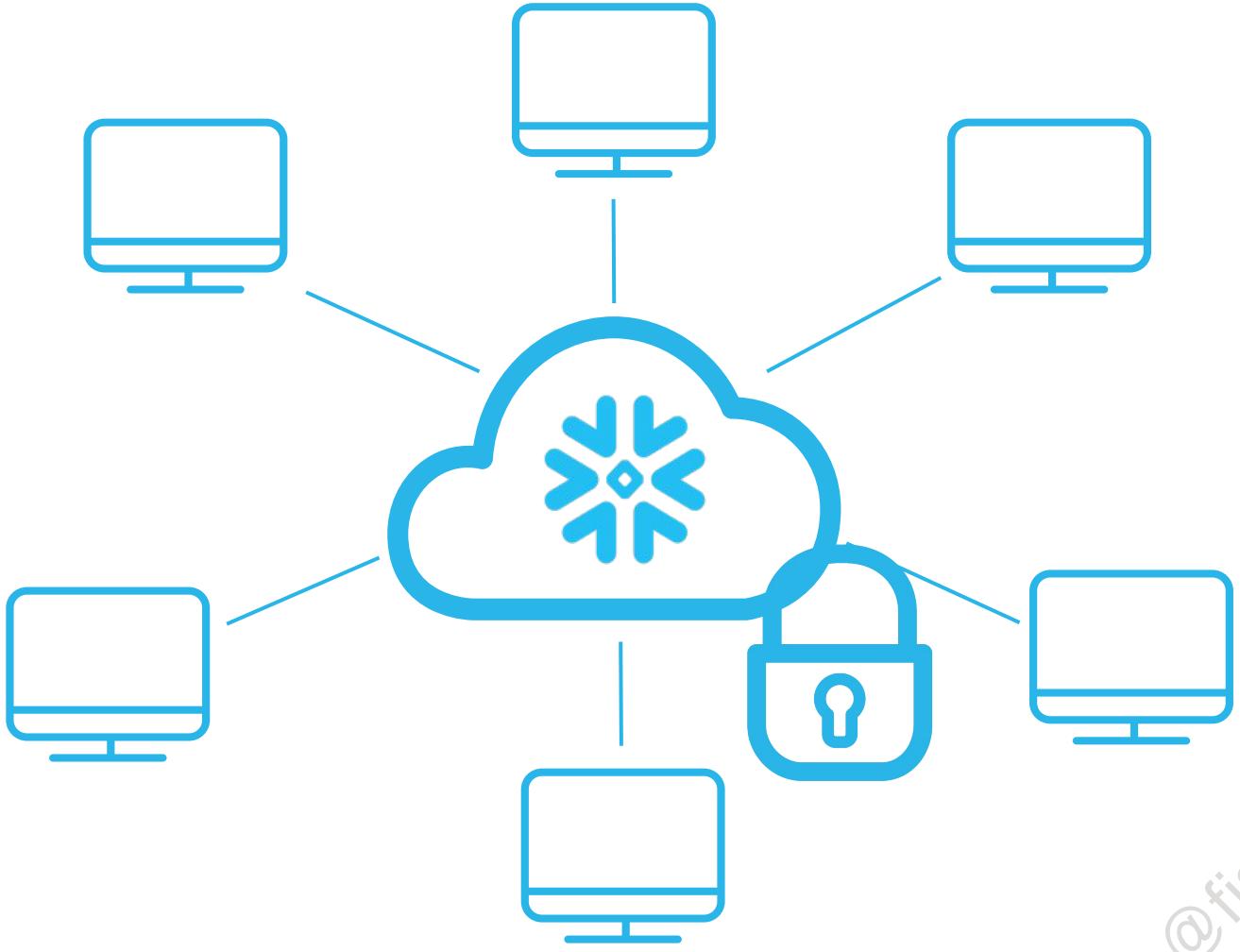


# Network Security

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# NETWORK SECURITY



- Network Policy support for IP whitelisting and blacklisting
- Online certificate status protocol (OCSP) x.509 validation native to all Snowflake connectors
- Support for cloud provider connectivity options: like:
  - PrivateLink (AWS) - no traffic over the public internet
  - DirectConnect (AWS) - dedicated network connection on-premise to AWS
  - Azure options on roadmap



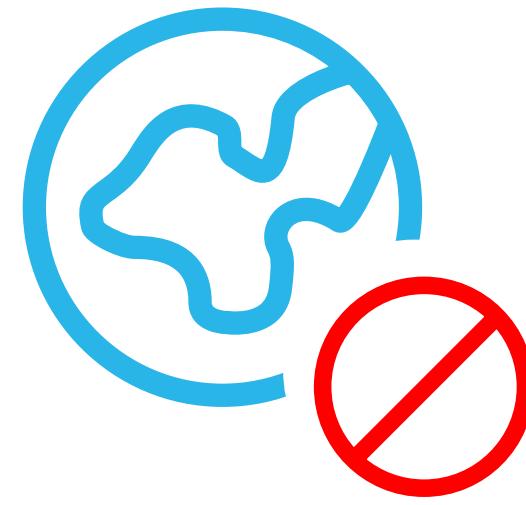
# NETWORK POLICY

- By default, Snowflake allows users to connect from any computer or device IP address
- Network policies allow/deny access to Snowflake to a list of one or more IP addresses
- A Network Policy consists of the following properties:



## Allowed IPs (Required)

List of IPv4 addresses (with optional subnets) that are allowed access to the Snowflake account



## Blocked IPs (Optional)

List of IPv4 addresses (with optional subnets) that are denied access to the Snowflake account

# NETWORK POLICIES

|                       |                                                                                                                                                                                                                                                                     |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose:</b>       | Restrict access to a Snowflake account to authorized network points                                                                                                                                                                                                 |
| <b>Method:</b>        | Restrict account access by IP address (IPv4 only)                                                                                                                                                                                                                   |
| <b>Details:</b>       | <ul style="list-style-type: none"><li>• Uses CIDR Notation to block a subnet range</li><li>• Blocked overrides Allowed</li><li>• IPs not on Allowed are AUTOMATICALLY Blocked</li><li>• Only one policy (at a time) may be active/ applied to the ACCOUNT</li></ul> |
| <b>Commands:</b>      | CREATE NETWORK POLICY - to create policy(s)<br>ALTER ACCOUNT - to apply a policy to the current account                                                                                                                                                             |
| <b>Prerequisites:</b> | Role: ACCOUNTADMIN or SECURITYADMIN                                                                                                                                                                                                                                 |



# ENSURING CONNECTIVITY

- Whitelist the following ports:
  - 443: Required for general Snowflake traffic
  - 80: Required for OCSP cache server, which listens for all Snowflake client communication
- Hostname Whitelisting:
  - All Snowflake clients require temporary access to cloud storage
  - Whitelist hostname patterns for the required hosts
    - See documentation for details
  - Use `$SYSTEM$WHITELIST` to obtain hostnames for your Snowflake account



## Module 7

# Access Control and User Management

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Concepts
- System Roles
- Custom Roles and Inheritance
- Ownership
- Configure and Manage Access
- Configuration Samples
- Recap



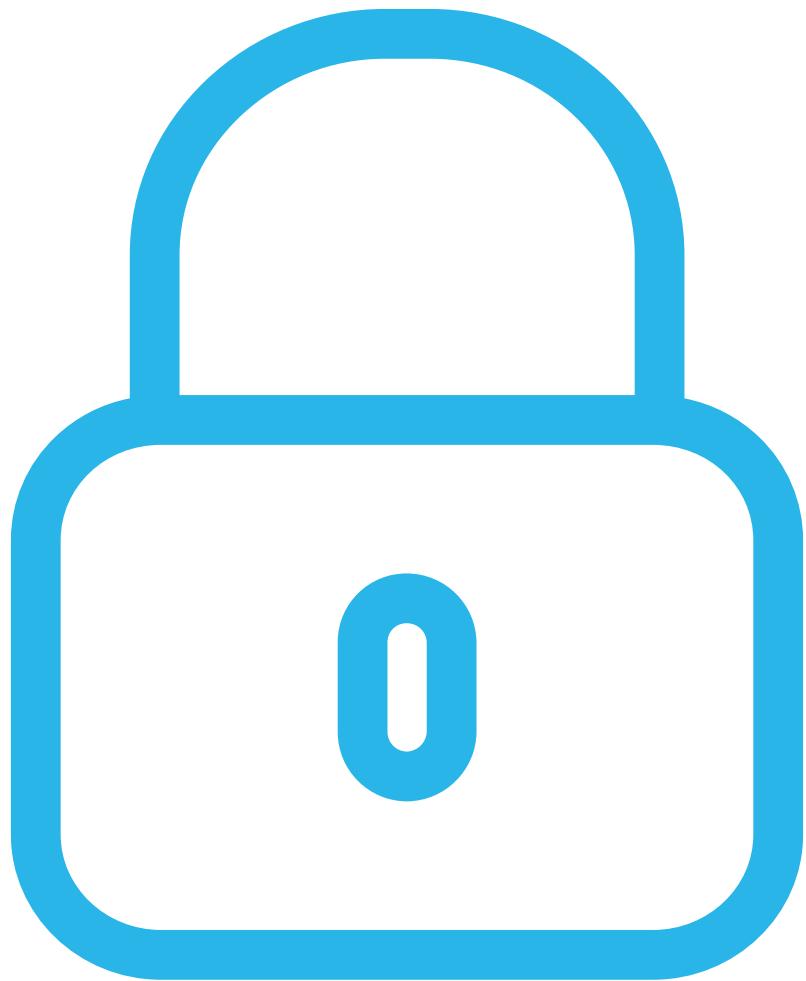
# Concepts

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# ACCESS CONTROL

What can you do?



- Access Control is part of Snowflake's authorization model
- Access Control defines:
  - Who can use which roles
  - Who can access which objects
  - What operations can be performed on those objects
  - Who can create or alter access control policies



# USERS AND ROLES

## USERS

SREEDHAR

ANNA

EVAN

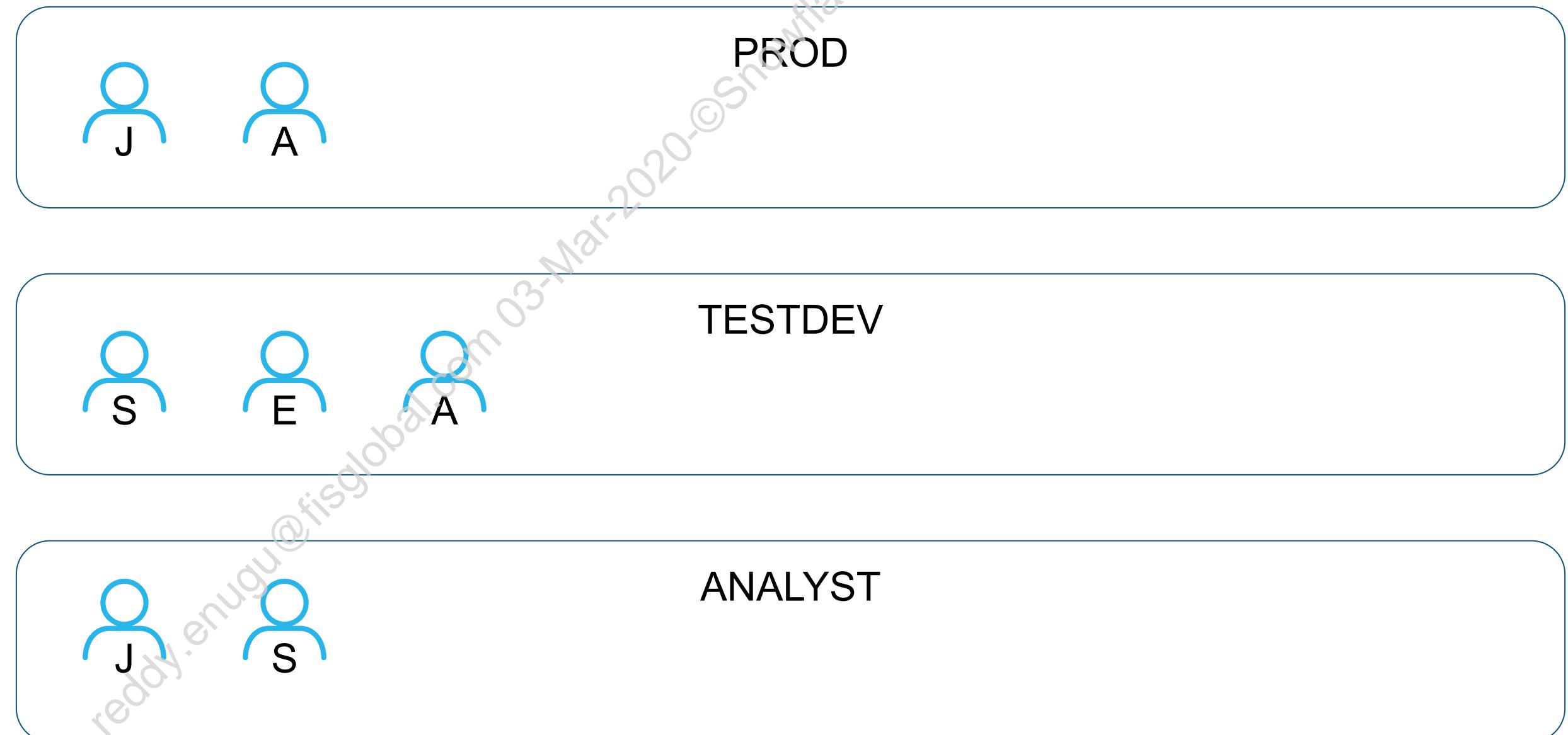
JENN

## ROLES

PROD

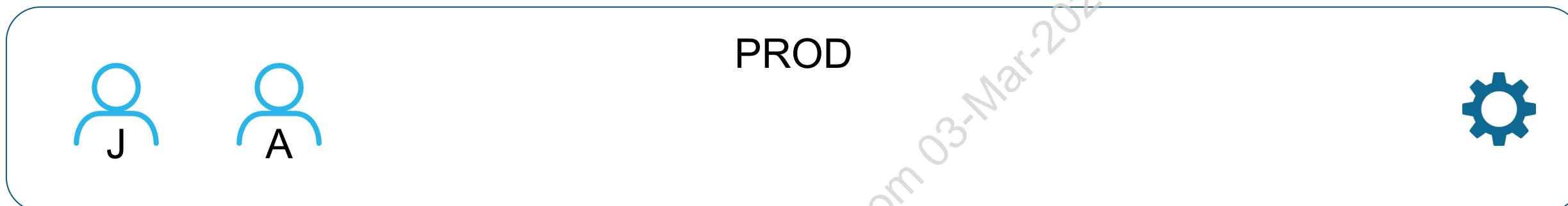
TESTDEV

ANALYST



# PRIVILEGES AND GRANTS

- A **privilege** is the right to do something on or with a Snowflake securable object
- A **grant** gives specific privileges to a role, or gives a user the right to use a role



```
GRANT ROLE PROD TO USER JENN;
GRANT ROLE PROD TO USER ANNA;
GRANT USAGE OF WAREHOUSE prod_wh TO ROLE PROD;
```

```
GRANT CREATE SCHEMA ON DATABASE prod_db TO ROLE PROD;
```

privilege

grant



# GRANT ROLES TO USERS

Grant roles to a user, to enable the user to access (use) those roles

```
GRANT ROLE <name> TO USER <name>;
```

Examples:

- **GRANT ROLE** dba **TO USER** barney;
- **GRANT ROLE** analyst, dba **TO USER** gail, henry;



# GRANT PRIVILEGES TO ROLES

Grant **privileges** on **securable objects** to **roles**

```
GRANT <privileges> ON <object> TO ROLE <name>;
```

Examples:

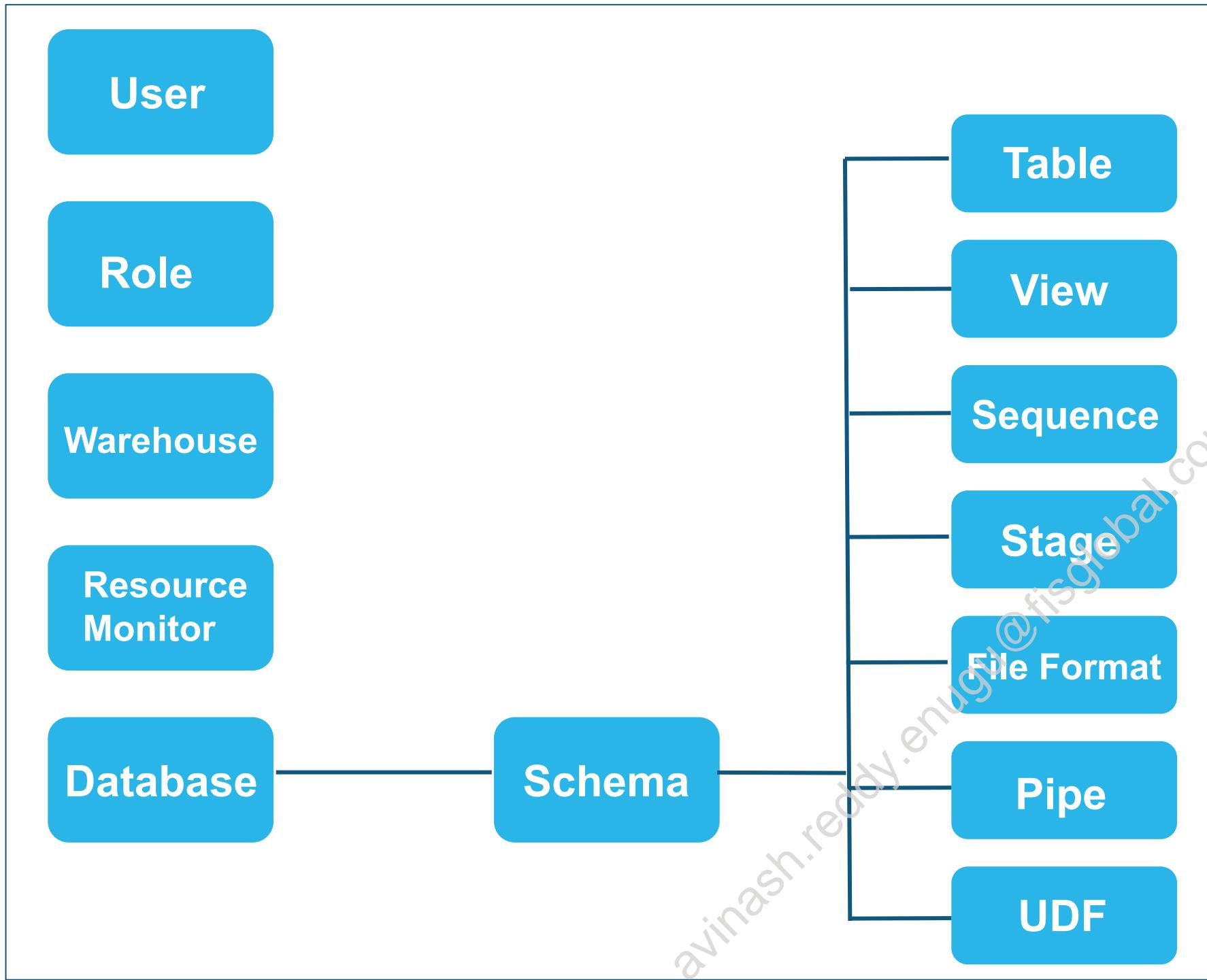
```
GRANT USAGE ON WAREHOUSE my_wh TO ROLE developers;
```

```
GRANT SELECT, INSERT, DELETE ON ALL TABLES IN SCHEMA my_schema
TO ROLE analyst, dba;
```



# SECURABLE OBJECTS

## Account



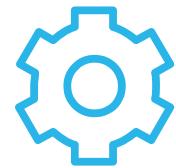
- Securable objects are constructs within Snowflake
- You grant privileges on securable objects
- Available privileges depend on the object type



# AVAILABLE PRIVILEGES DEPEND ON THE OBJECT



CREATE ROLE | USER | WAREHOUSE | DATABASE, MANAGE GRANTS, MONITOR USAGE... **ON ACCOUNT**



GRANT USAGE | MONITOR | MODIFY | OPERATE **ON WAREHOUSE**



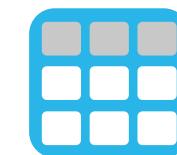
GRANT MODIFY | MONITOR **ON RESOURCE MONITOR**



GRANT MODIFY | MONITOR | USAGE | CREATE SCHEMA **ON DATABASE**



GRANT MODIFY | MONITOR | USAGE | CREATE TABLE | CREATE VIEW... **ON SCHEMA**



GRANT SELECT | INSERT | UPDATE | DELETE | TRUNCATE... **ON TABLE**



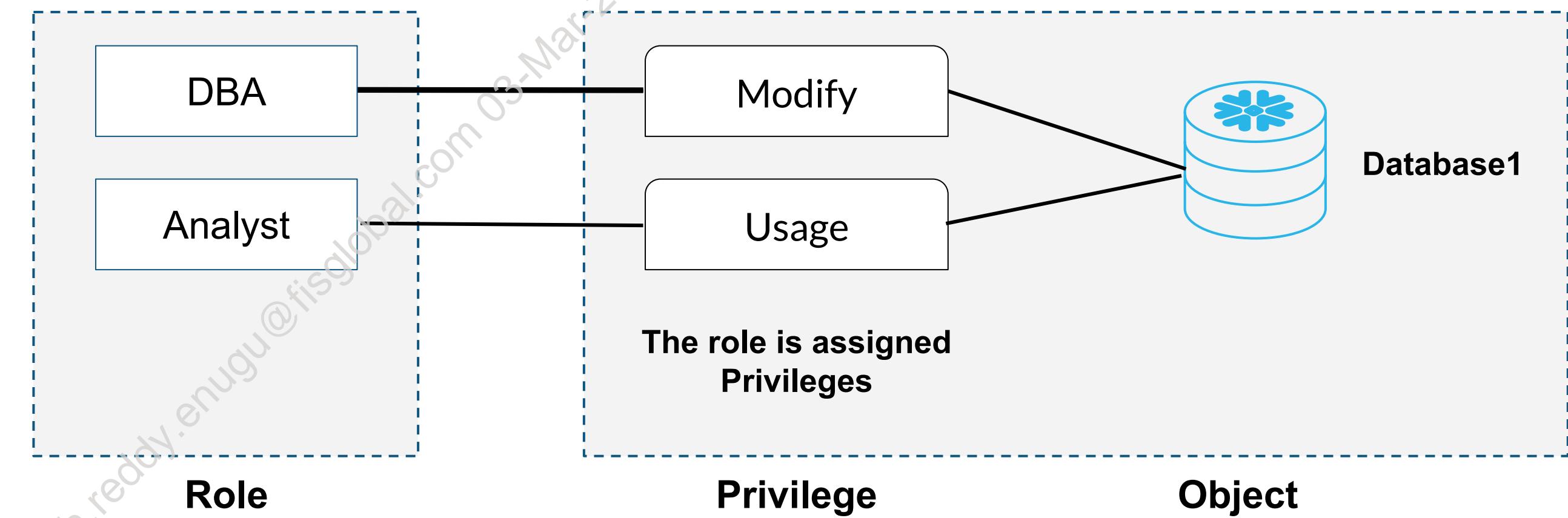
# ABOUT PRIVILEGES

- Warehouse privileges **USAGE** and **OPERATE** are separate and specific
  - **USAGE**: use a warehouse to run queries
  - **OPERATE**: start, stop, change the settings of a virtual warehouse
  -
- Acting on an object requires **USAGE** on its logical container(s)
  - Example: to use **SELECT** privileges on a table, you must have **USAGE** on the schema and database
- **CREATE**, **SELECT**, **INSERT**, **UPDATE**, and **DELETE** are all separate privileges



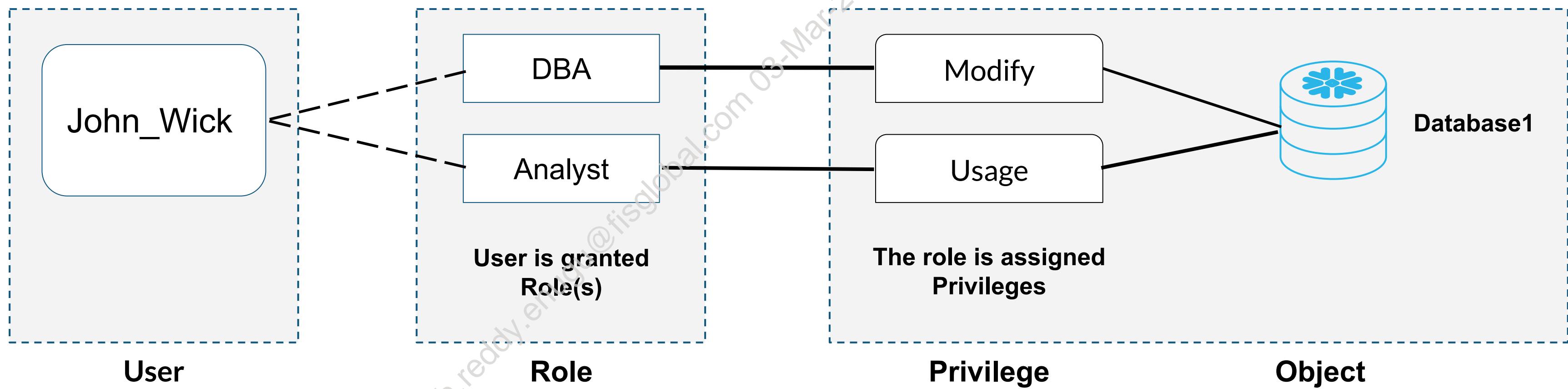
# USERS, ROLES, AND GRANTS

- Roles are granted permissions (privileges) to access objects



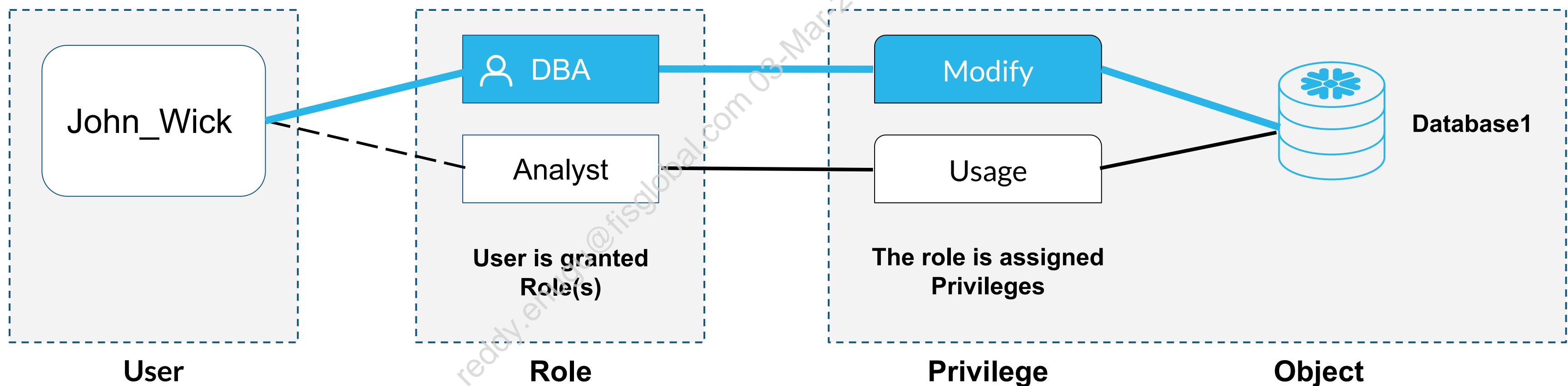
# USERS, ROLES, AND GRANTS

- Roles are granted permissions (privileges) to access objects
- Users are granted roles which they can use



# USERS, ROLES, AND GRANTS

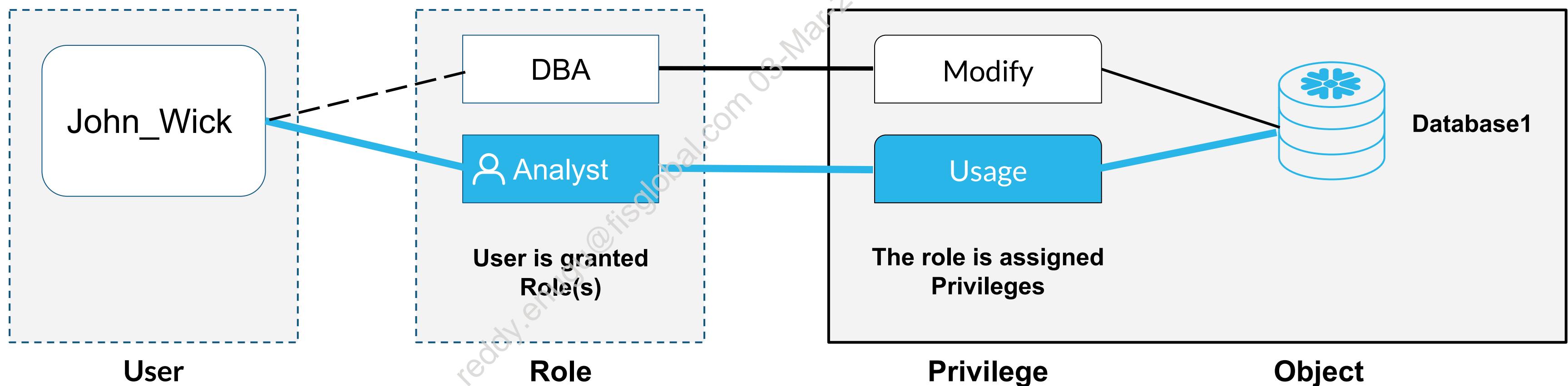
- Roles are granted permissions (privileges) to access objects
- Users are granted roles which they can use
- Users USE a role, and get the role's privileges while using the role



John has privileges to **Modify** the database while using the **DBA** role

# USERS, ROLES, AND GRANTS

- Roles are granted permissions (privileges) to access objects
- Users are granted roles which they can use
- Users USE a role, and get the role's privileges while using the role
- Users can be assigned to multiple roles, but only use one (current role) at a time



John has privileges to **USE** the database while using the **Analyst** role

# System Roles

avinash.reddy.enugu@fisglobal.com 05-Mar-2020 ©Snowflake 2020-do-not-copy



# SYSTEM ROLES

ACCOUNTADMIN

SECURITYADMIN

SYSADMIN

PUBLIC

- ACCOUNTADMIN
  - Has all privileges
- SECURITYADMIN
  - Create, monitor, manage users and roles
  - Grant or revoke privileges to users and roles
- SYSADMIN
  - Create warehouses and databases
  - Create other objects
- PUBLIC
  - No privileges
  - Default role for all users, unless otherwise specified



# DIVISION OF RESPONSIBILITY

## ACCOUNTADMIN

ACCOUNTADMIN has ALL privileges - either granted, or inherited. Grant this role to a limited number of users.

### DEFAULT GRANTS:

- CREATE SHARE
- IMPORT SHARE
- MONITOR USAGE

By default, only ACCOUNTADMIN can do these things

ACCOUNTADMIN

SECURITYADMIN

SYSADMIN



# BEST PRACTICES

## ACCOUNTADMIN

- Assign to a limited number of users
- Require MFA on all users with this role
- Do not make it the default role for any user
- Never execute scripts using ACCOUNTADMIN

ACCOUNTADMIN

SECURITYADMIN

SYSADMIN



# DIVISION OF RESPONSIBILITY

## SECURITYADMIN

SECURITYADMIN should create all users and roles, and GRANT them to SYSADMIN (or somewhere in the SYSADMIN hierarchy)

### DEFAULT GRANTS:

- CREATE USER on ACCOUNT
- CREATE ROLE on ACCOUNT
- MANAGE GRANTS on ACCOUNT

ACCOUNTADMIN

SECURITYADMIN

SYSADMIN



# DIVISION OF RESPONSIBILITY

SYSADMIN

SYSADMIN should grant PRIVILEGES to ROLES

## DEFAULT GRANTS:

- CREATE DATABASE on ACCOUNT
- CREATE WAREHOUSE on ACCOUNT

ACCOUNTADMIN

SECURITYADMIN

SYSADMIN



# DIVISION OF RESPONSIBILITY

## PUBLIC

PUBLIC can't do anything but log in. It's the default role for new users, unless changed.

### DEFAULT GRANTS:

- Zero
- Zip
- Zilch
- Nada

PUBLIC

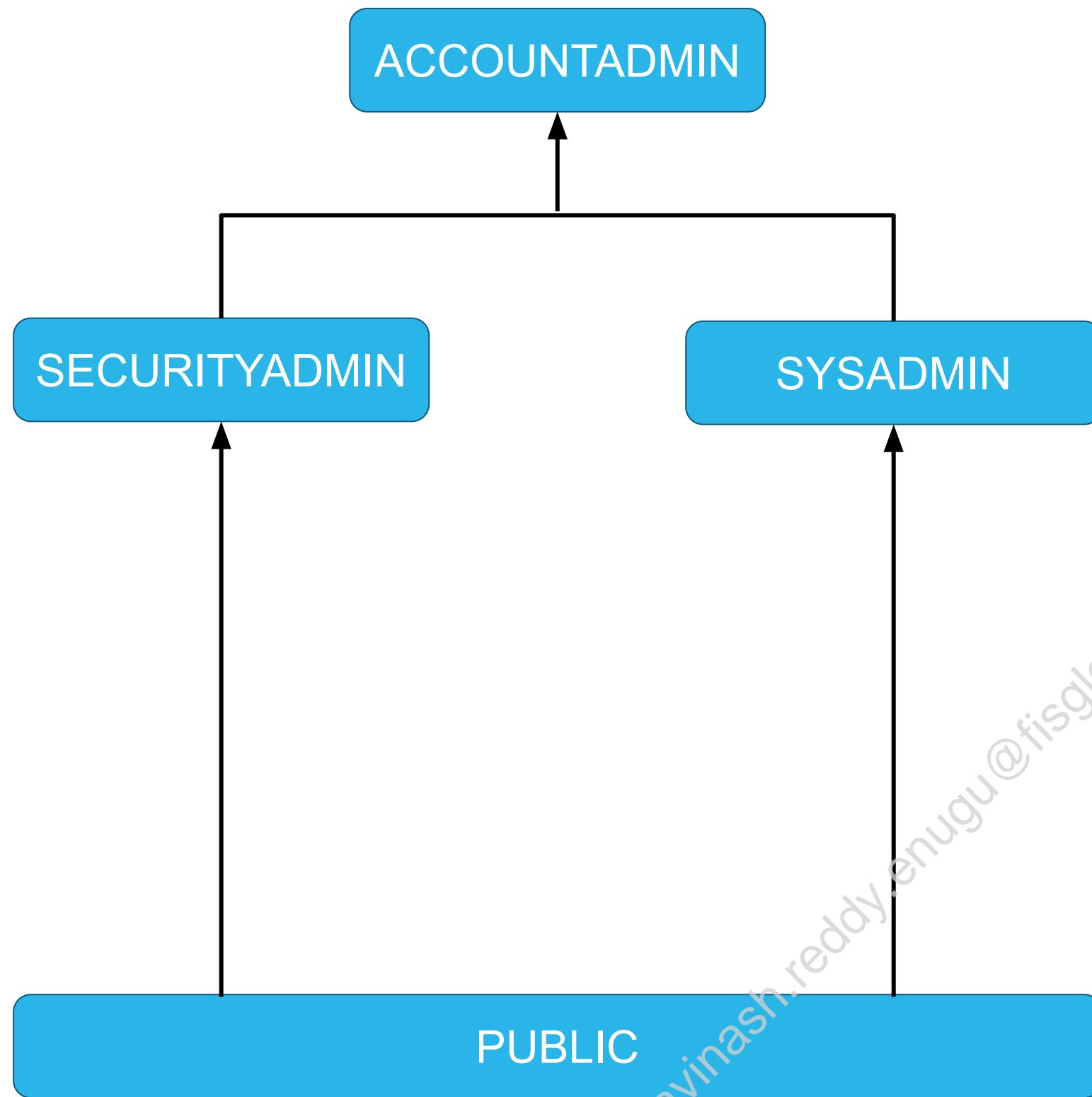


# Custom Roles and Inheritance

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# ROLE INHERITANCE



- A role inherits all the privileges of its underlying roles (those "lower" in the hierarchy)
- ACCOUNTADMIN inherits privileges from SECURITYADMIN, SYSADMIN, and PUBLIC
- SECURITY ADMIN and SYSADMIN inherit privileges from PUBLIC
- PUBLIC inherits nothing



# CREATING CUSTOM ROLES

```
USE ROLE securityadmin;
```

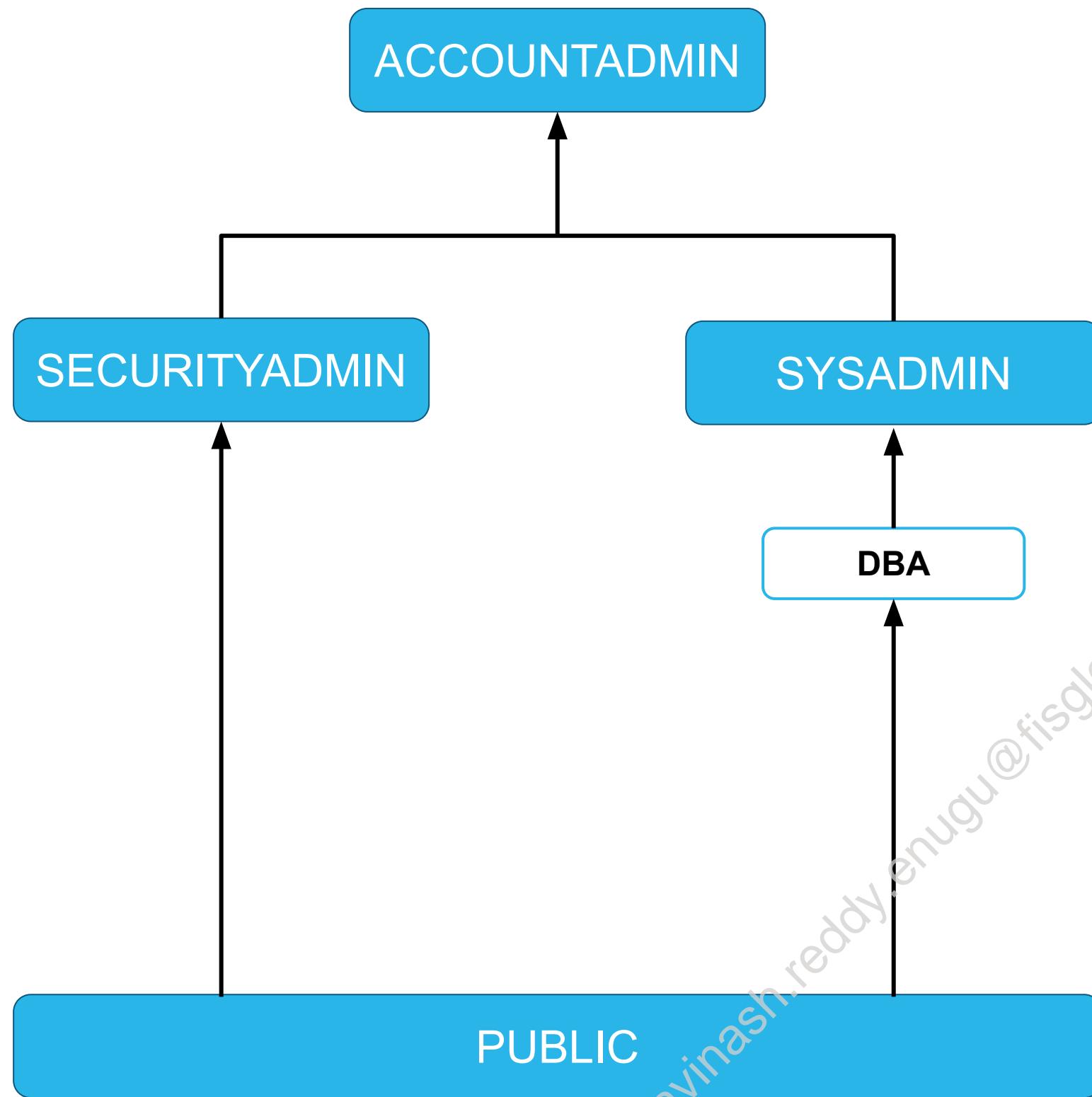
```
CREATE ROLE dba;
```

```
GRANT ROLE dba TO ROLE sysadmin;
```

```
GRANT ROLE dba TO USER jenna, tom;
```

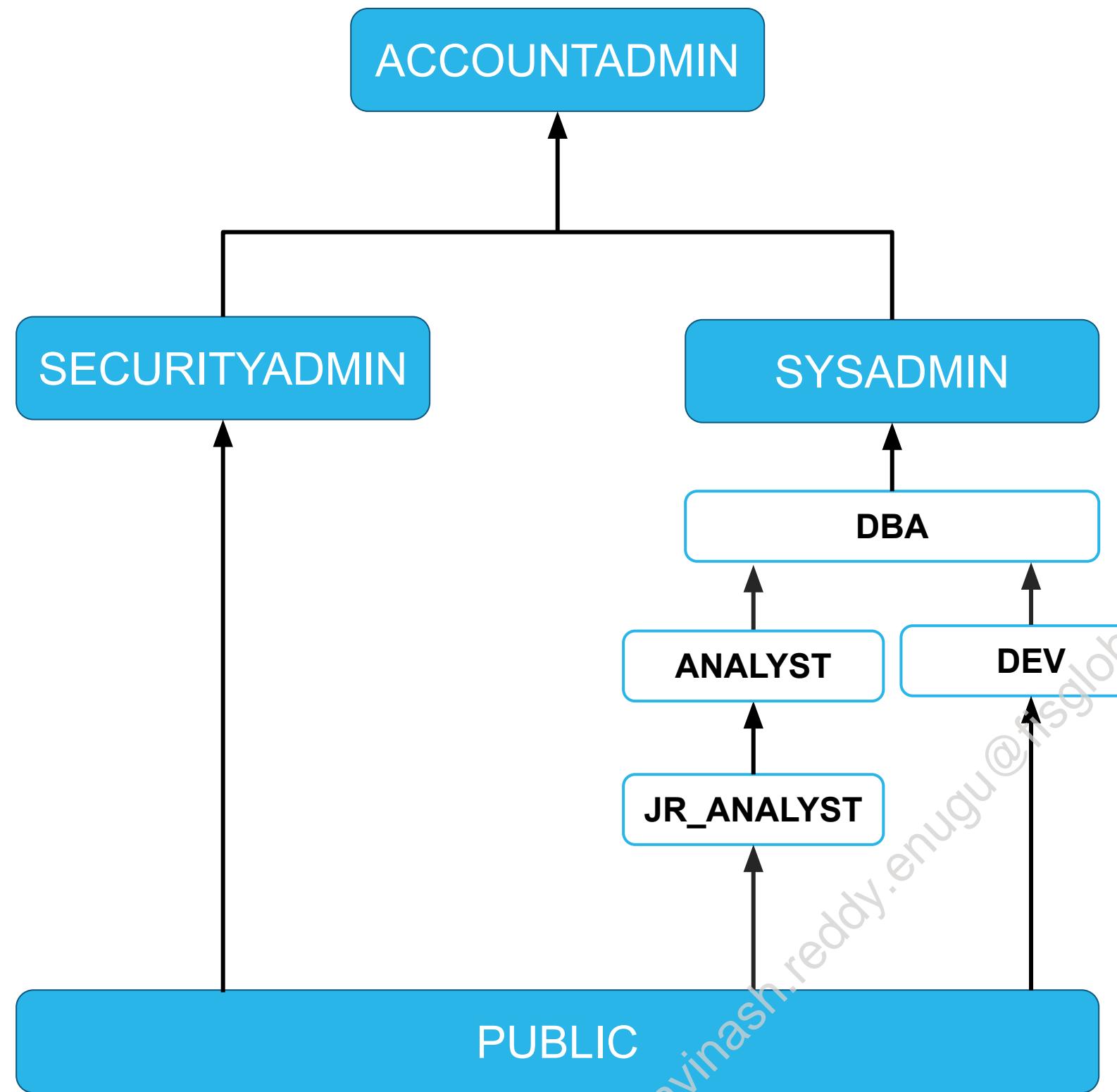


# ROLE HIERARCHY



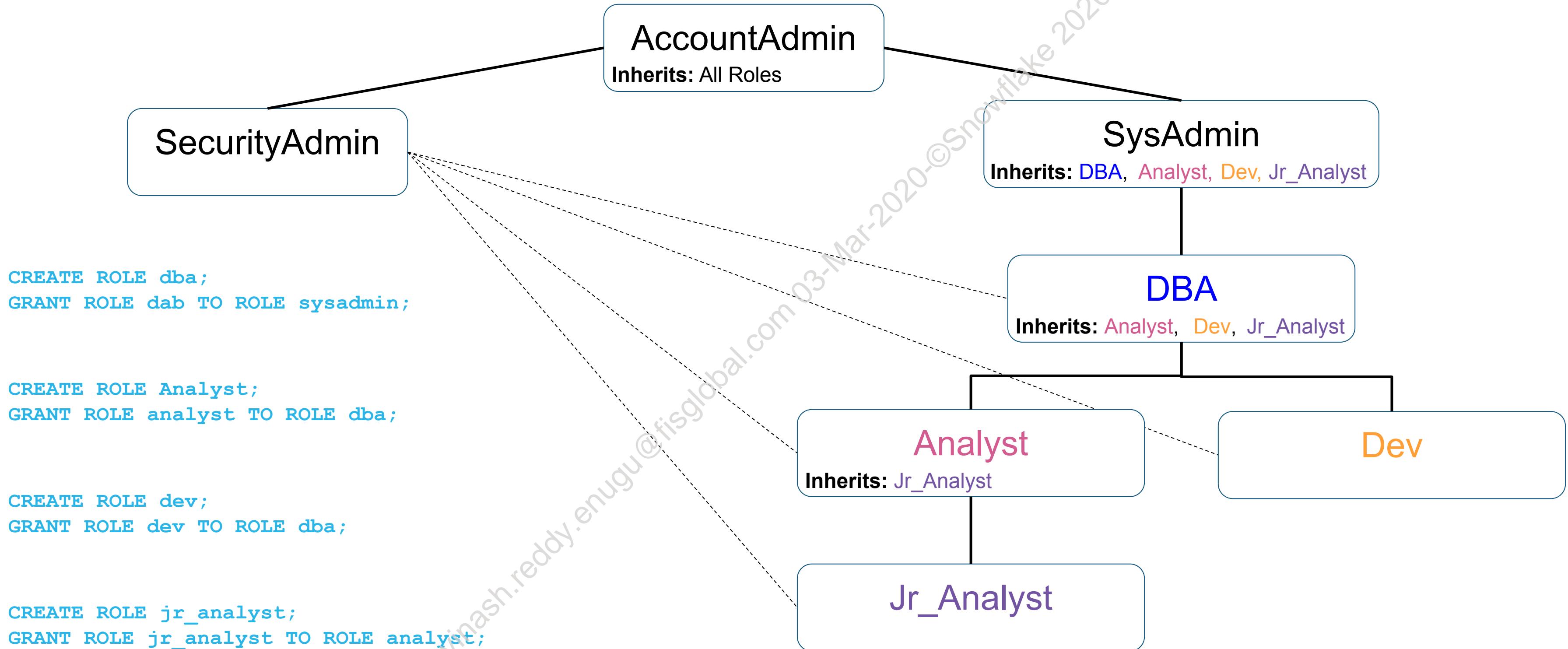
- Roles fit into the hierarchy based on which role(s) they are granted to
- Custom roles also inherit privileges from the roles "beneath" them
- Here:
  - Role DBA has been granted to role SYSADMIN
  - Role SYSADMIN inherits privileges from role DBA
  - Role DBA inherits privileges from role PUBLIC

# ROLE HIERARCHY

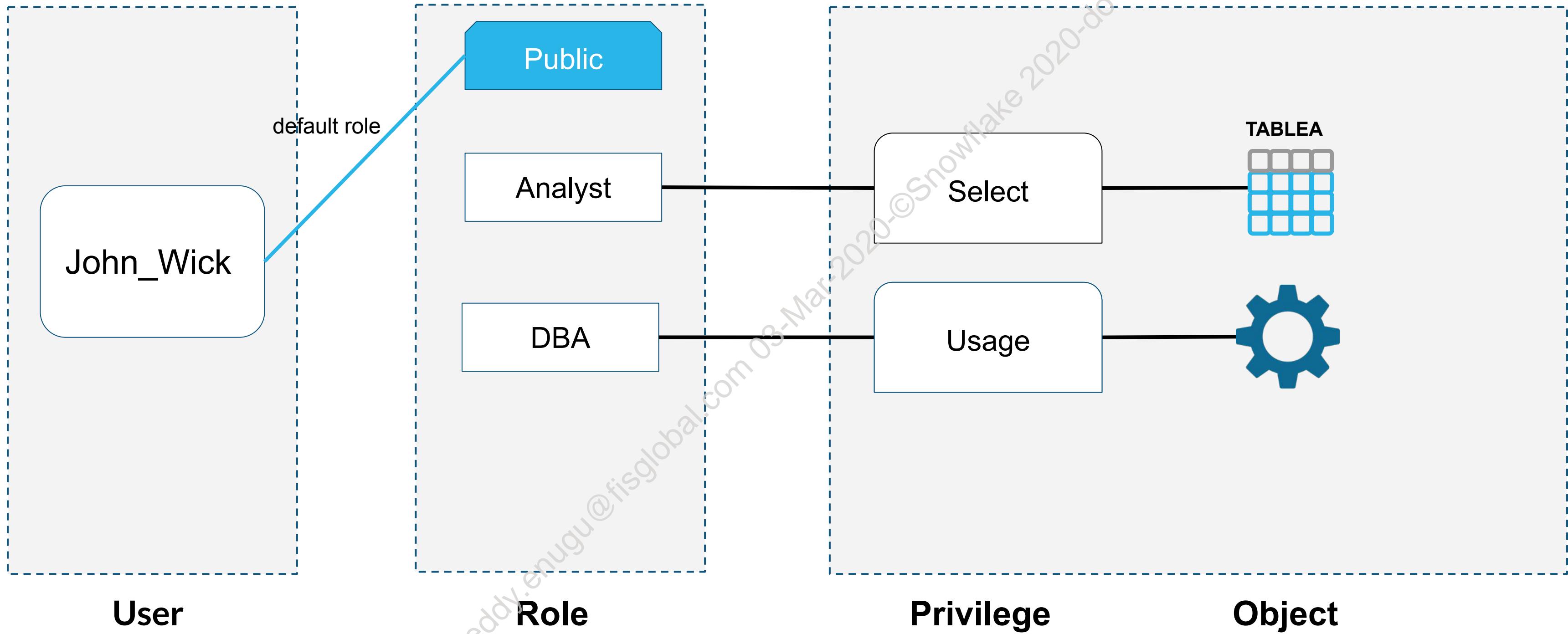


- Grants:
  - Role DBA has been granted to Role SYSADMIN
  - Role ANALYST has been granted to role DBA
  - Role DEV has been granted to role DBA
  - Role JR\_ANALYST has been granted to role ANALYST
- Inheritance:
  - SYSADMIN inherits from DBA and roles below it
  - DBA inherits from DEV, ANALYST, and roles below them
  - ANALYST inherits from JR\_ANALYST

# ROLE HIERARCHY VISUALIZATION

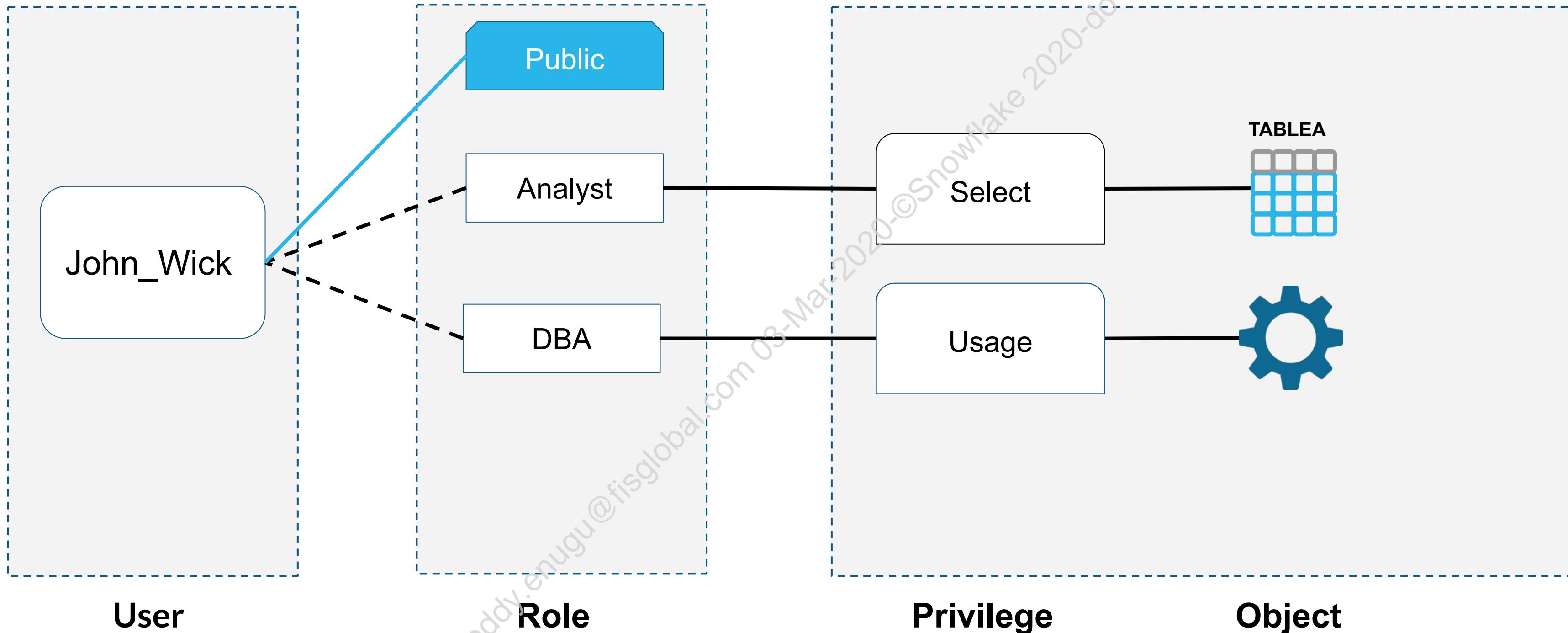


# ROLES AND INHERITANCE



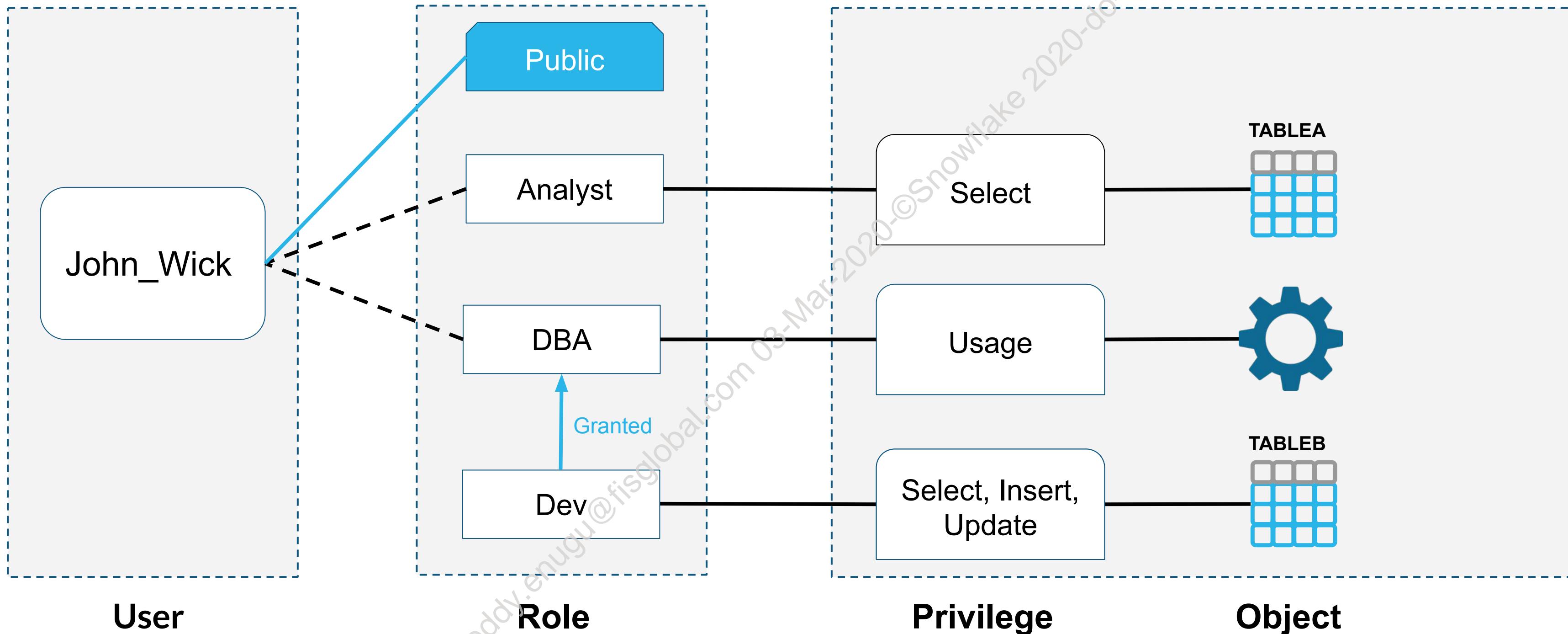
Upon connecting to Snowflake, a user takes on their default role.

# ROLES AND INHERITANCE



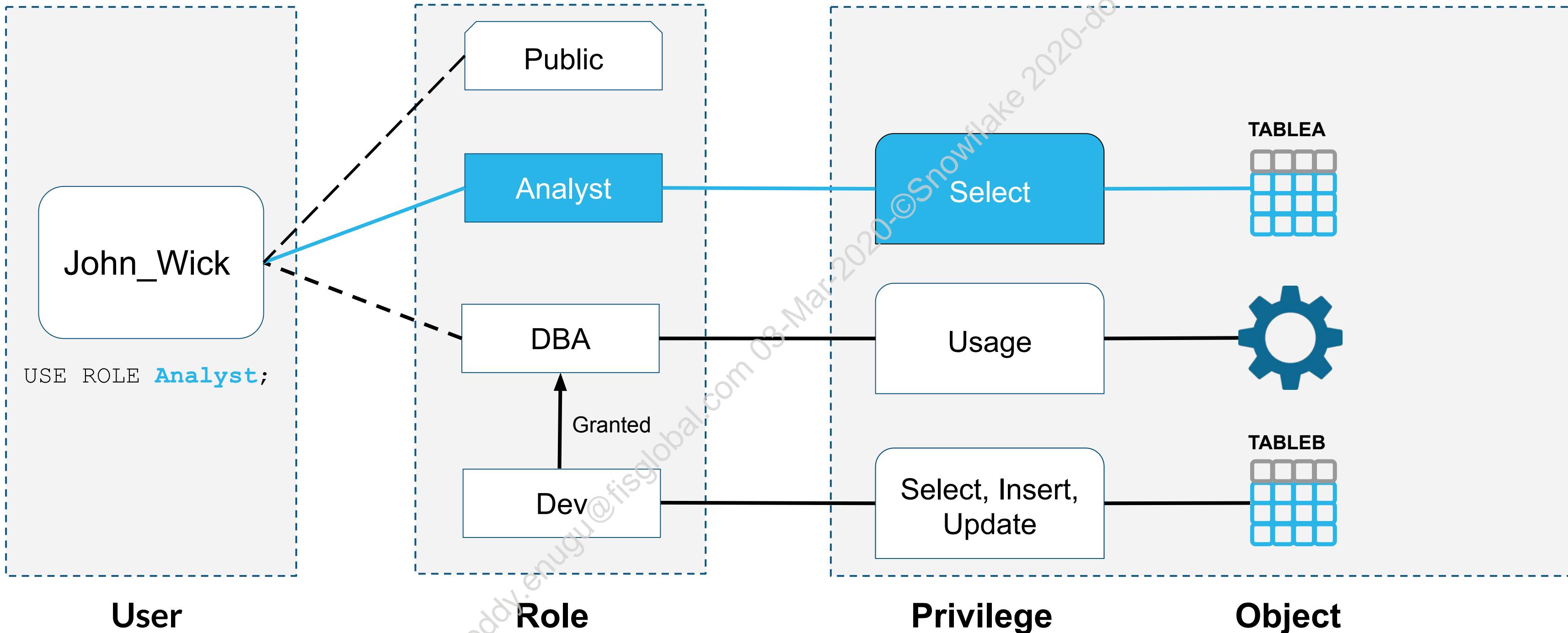
User **John\_Wick** has been granted the roles Analyst, and DBA.

# ROLES AND INHERITANCE



The role Dev has been granted to the role DBA.

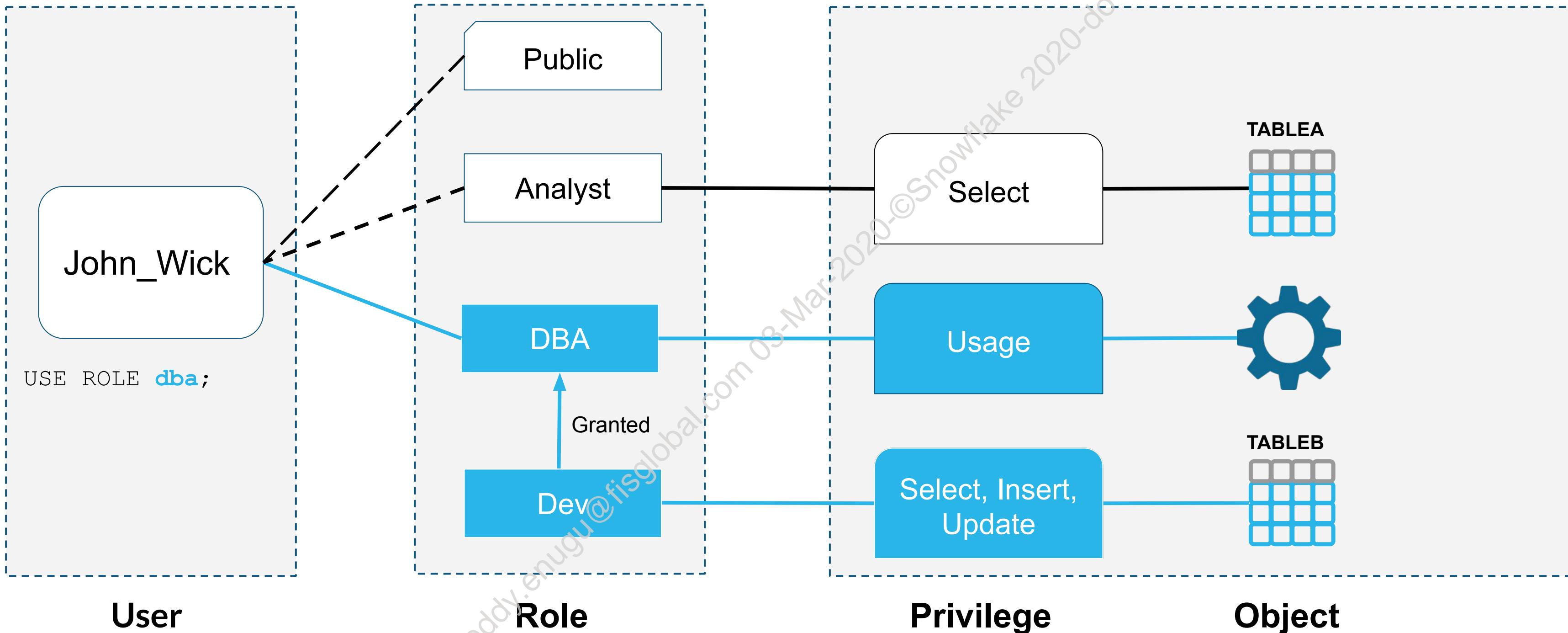
# ROLES AND INHERITANCE



When using the **Analyst** role, John can **SELECT** from **TABLEA**.

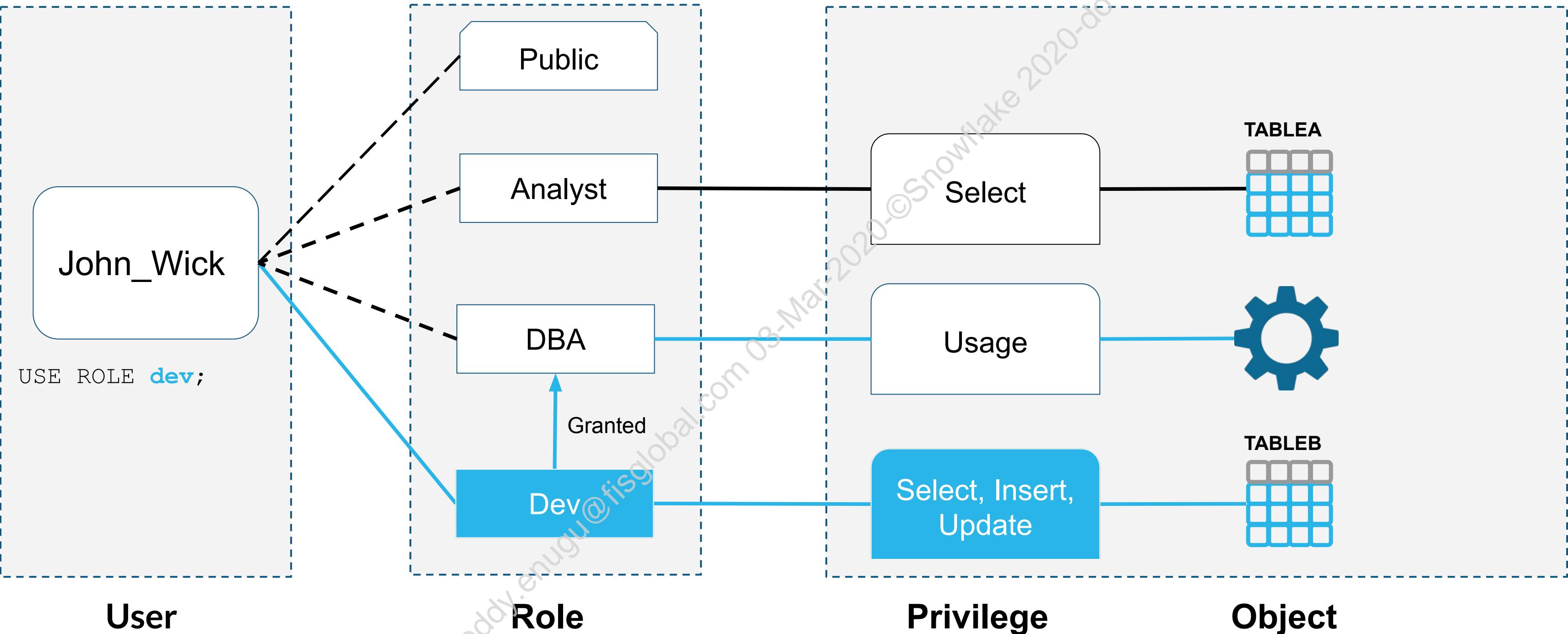


# ROLES AND INHERITANCE



With the **DBA** role, he can **USE** the warehouse, and **SELECT, INSERT, and UPDATE TABLEB**.

# ROLES AND INHERITANCE



He can also **USE** the Developer role, since it was granted to a role he has the right to use.

# Ownership

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# OWNERSHIP

- When you create an **object**, your **current role** becomes the **owner**
- Every **role** or **object** is owned by a **single role**
  - All users in that role share ownership, when they are using that role
- The owner can do anything with the object
- The owner can grant privileges to itself or other roles
- Ownership can be transferred by the owner

```
GRANT OWNERSHIP ON WAREHOUSE dev_ws TO ROLE dba
```



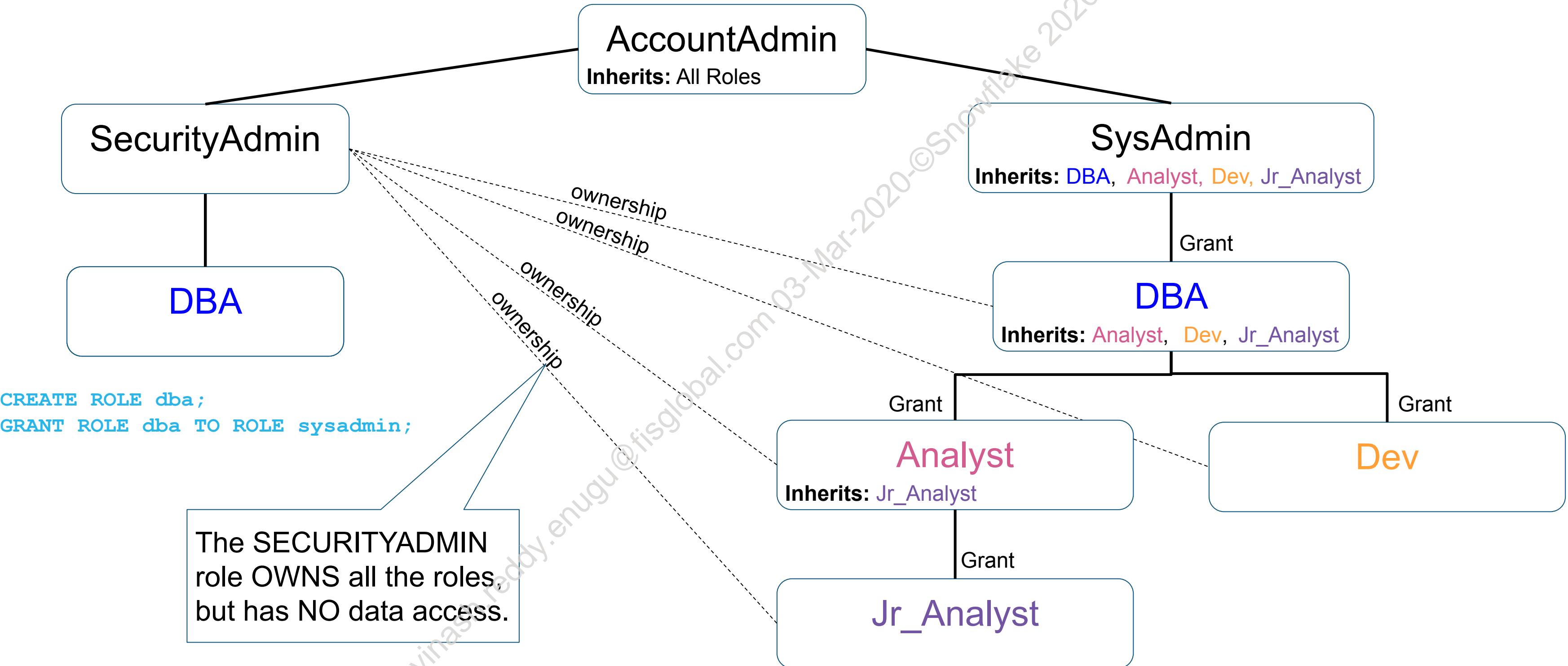
# OWNERSHIP

- **Warning:** Object ownership will cause confusion.
  - Be aware of which role you're using when you create an object.
  - When you change roles, your object may no longer be accessible (because **the object belongs to your role when you created it, not to you**).
  - If an object goes "missing," check with the role owner.
- **Best Practice:** Have SECURITYADMIN own all roles
- **Example:**

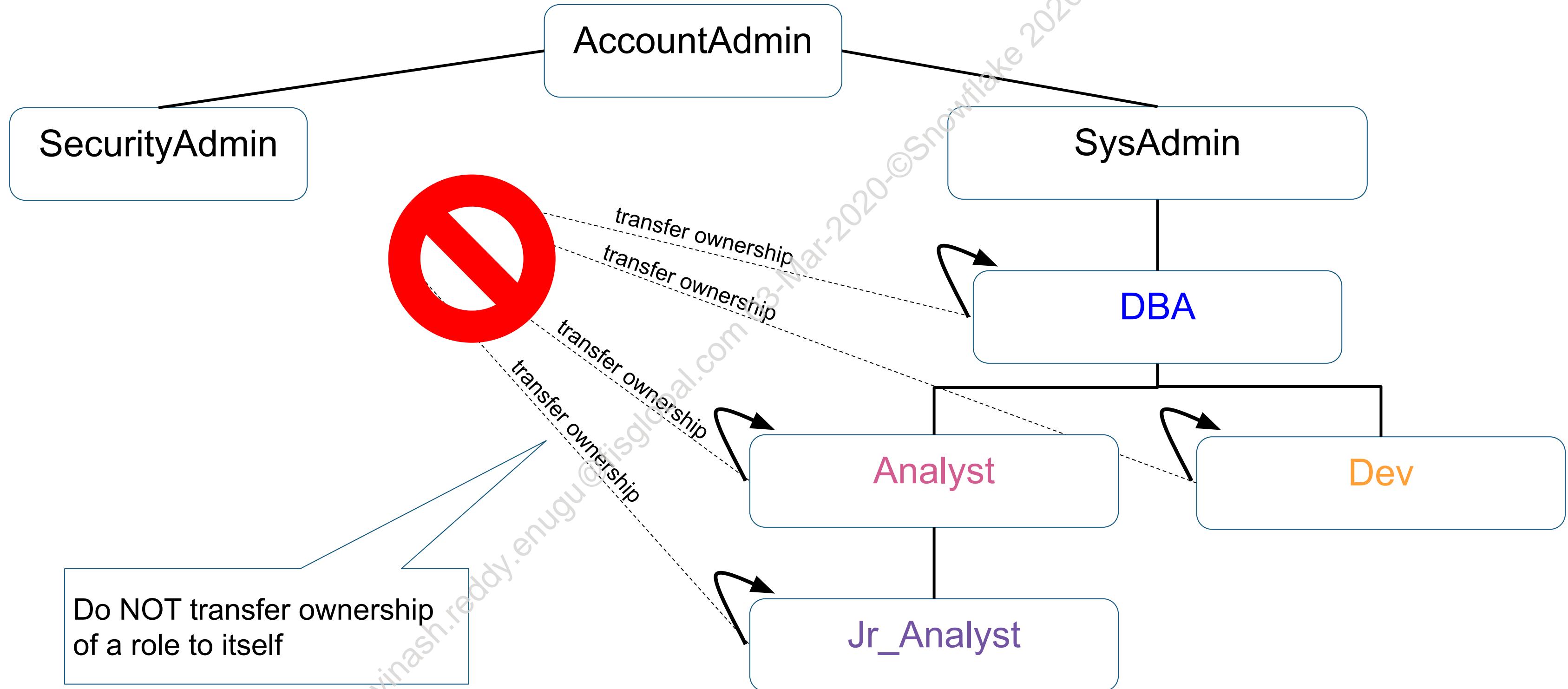
```
GRANT CREATE DATABASE ON ACCOUNT TO ROLE dba;
USE ROLE dba;
CREATE DATABASE my_db1;
CREATE SCHEMA my_db1.myschema;
```



# ROLE OWNERSHIP VERSUS GRANT



# ROLE OWNERSHIP VERSUS GRANT



# OWNERSHIP AND GRANTS

- Granted privileges are different from ownership
- GRANTs give a role the right to do something with an object, or to use a role
- The OWNER has the right to GRANT privileges on the objects they own
- The OWNER also has all rights to the objects they own



# Configure and Manage Access

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# CREATE USERS AND ROLES

```
CREATE USER johnsmith
 PASSWORD='rose-bud'
 default_role = developer
 must_change_password=true;
```

```
DESC USER johnsmith;
```

```
CREATE ROLE dba;
```

```
GRANT ROLE dba TO USER johnsmith;
```

- Users with SECURITYADMIN role or above can manage users and roles using SQL or the web UI
- When creating a user, strongly recommend forcing a password change on first login
- Can show users, describe a given user, or alter/drop a user
- Once created, assign users to roles



# MANAGE USERS

```
ALTER USER johnsmith
 SET PASSWORD = 'rose-bud-II'
 must_change_password = true;
```

```
ALTER USER johnsmith
 RESET password;
```

```
ALTER USER johnsmith
 SET disabled = true;
```

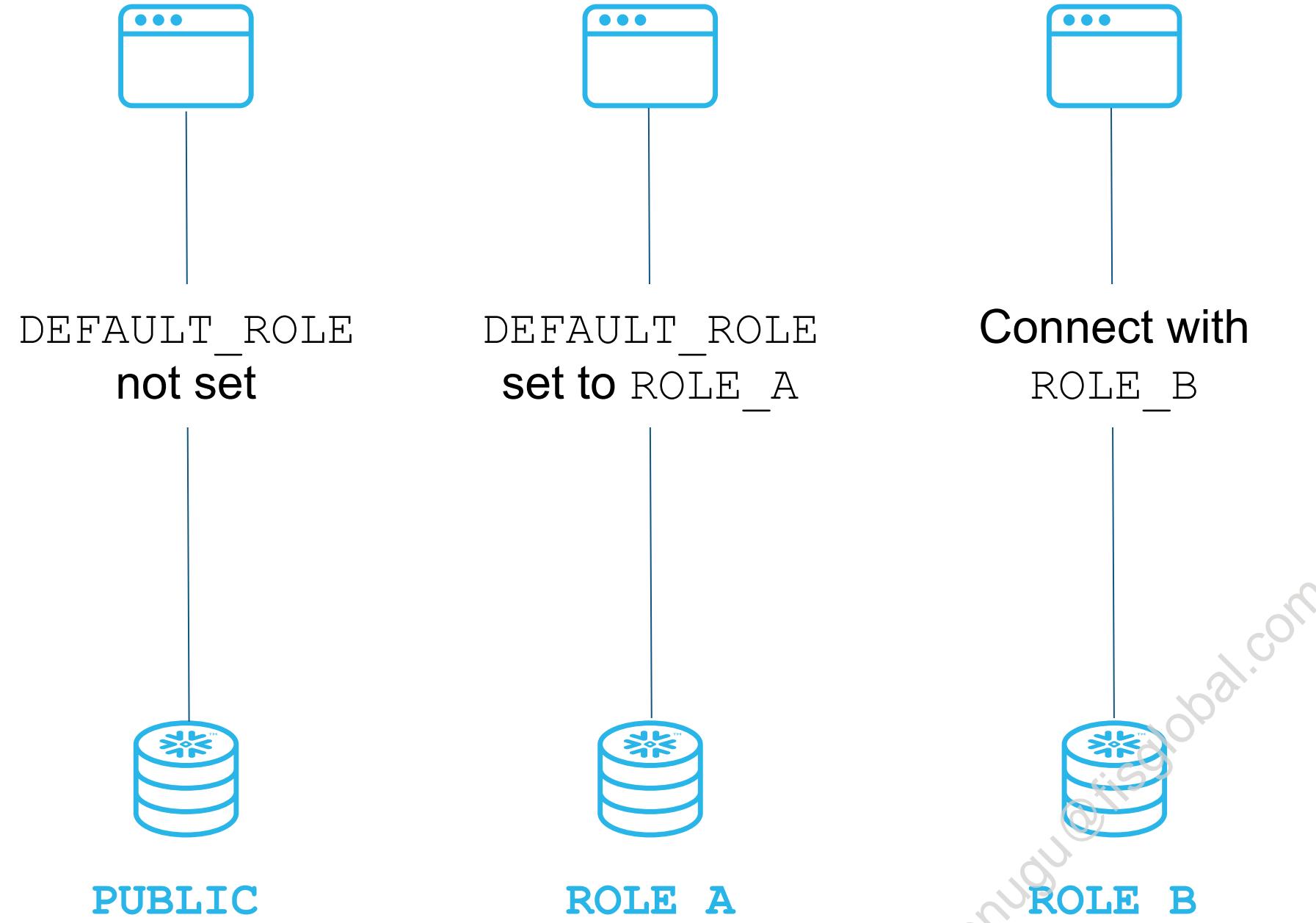
```
ALTER USER johnsmith
 SET minutes_to_unlock = 0;
```

```
ALTER USER johnsmith
 SET PASSWORD = 'XXX';
```

- Use ALTER USER to change the password for another user
- After 5 login failures, user is locked out for 15 minutes
- RESET PASSWORD generates a URL to share with the user
  - Old password valid until changed by user
  - URL expires in 4 hours
- Disabling a user terminates user sessions/locks them out immediately



# ASSIGN A DEFAULT ROLE



- Default role is **PUBLIC**
- Set **DEFAULT\_ROLE** for user to change the default
  - `ALTER USER name SET DEFAULT_ROLE role`
- Set the role in the connection string from the client (JDBC, ODBC, SnowSQL...)
  - Role set in connection string overrides configured default role



# VIEW CURRENT PRIVILEGES

- View the current roles granted to a user:

```
SHOW GRANTS TO USER <role_name>;
```

- View the current set of privileges granted to a role:

```
SHOW GRANTS TO ROLE <role_name>;
```



# GRANT PRIVILEGES

```
GRANT ROLE warehouse_manager TO USER kelly;
```

```
GRANT ROLE dba TO USER allison;
```

```
GRANT ROLE analyst TO USER marie;
```

```
GRANT OPERATE ON WAREHOUSE wh1 TO ROLE warehouse_manager;
```

```
GRANT USAGE ON WAREHOUSE wh1 TO ROLE dba, analyst
```

```
GRANT CREATE SCHEMA ON DATABASE db1 TO ROLE dba;
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA schema1 TO ROLE analyst;
```



# FUTURE GRANTS

- Grants, by default, apply only to existing objects

```
GRANT SELECT ON ALL TABLES IN SCHEMA mydb.myschema TO ROLE role1;
```

- Use **ON FUTURE** to grant privileges to objects that will be created in the future
  - See documentation for details on future grants

```
GRANT SELECT ON FUTURE TABLES IN SCHEMA mydb.myschema TO ROLE role1;
```



# Configuration Samples

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

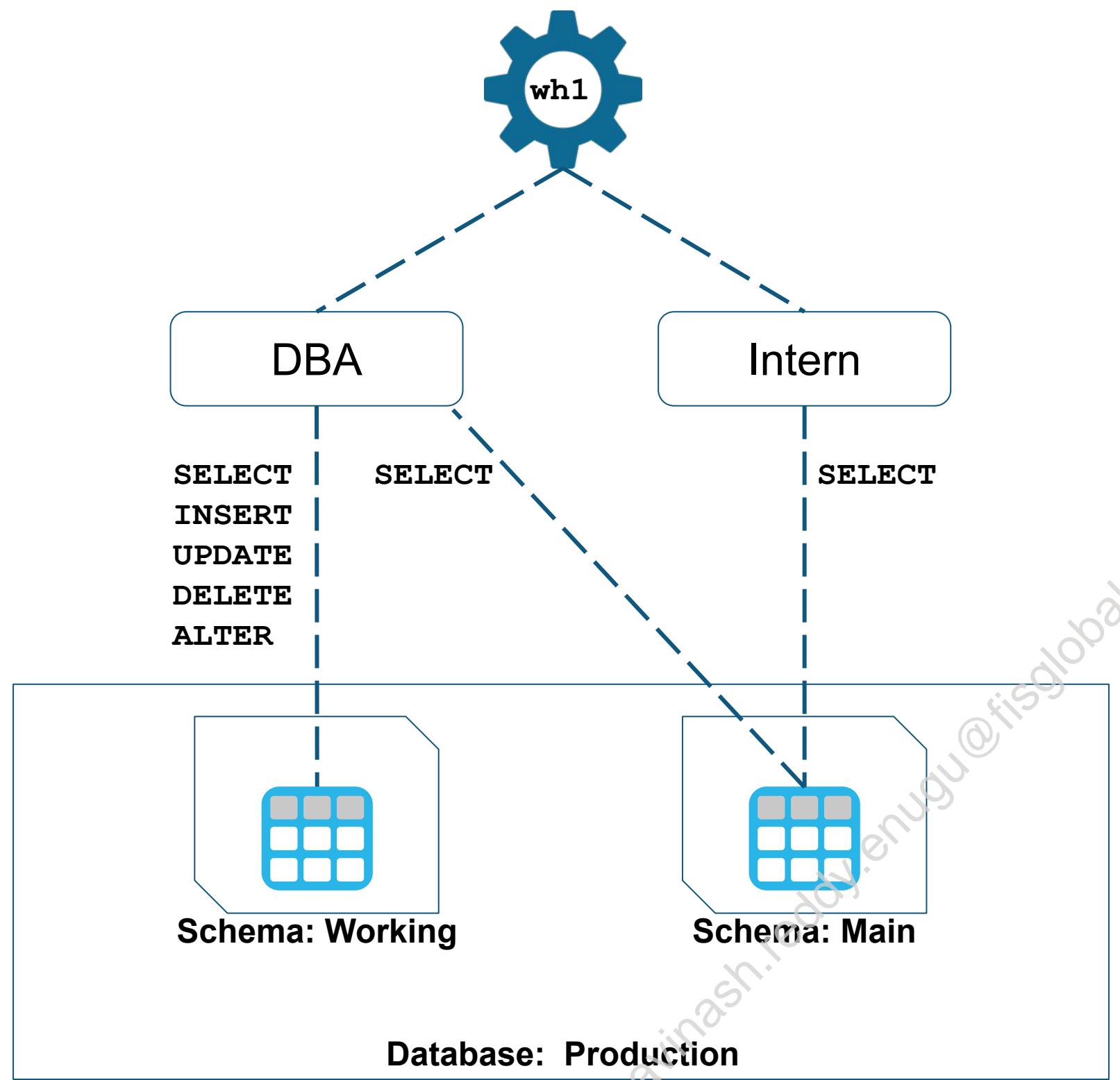


# BUILD OUT YOUR PRIVILEGES

1. Diagram your required roles and privileges from the bottom up
  - a. Try not to assign overlapping permissions
  - b. Have roles with the least permissions lower in the hierarchy
2. Verify your diagram several times
3. Build your roles from the top down
4. Assign privileges to roles
5. Assign users to roles
6. Test your roles, grants, and inheritance



# CREATE A READ-ONLY ROLE



```
CREATE ROLE intern;
GRANT ROLE intern TO ROLE SYSADMIN;

GRANT USAGE ON DATABASE production
TO ROLE dba, intern;

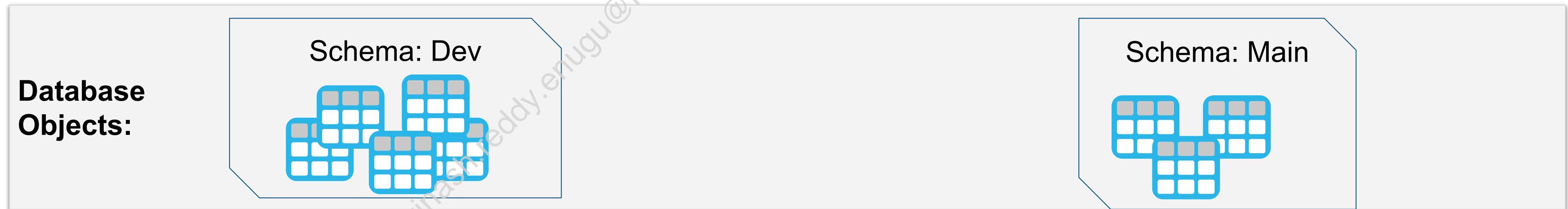
GRANT USAGE ON SCHEMA main
TO ROLE dba, intern;

GRANT SELECT ON ALL TABLES IN
SCHEMA main TO ROLE dba, intern;

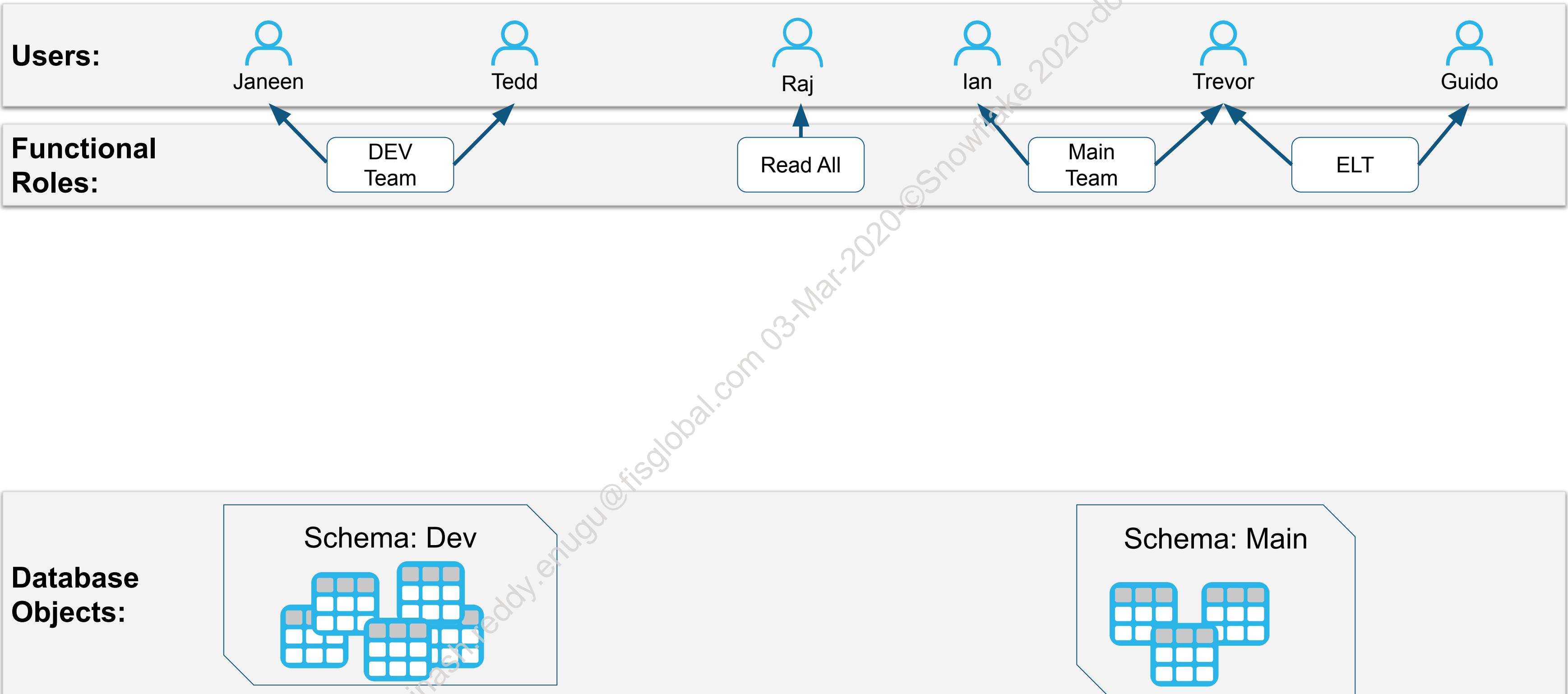
GRANT USAGE ON WAREHOUSE wh1
TO ROLE dba, intern;

GRANT SELECT, INSERT, UPDATE,
DELETE, ALTER ON ALL TABLES IN
SCHEMA working TO ROLE dba;
```

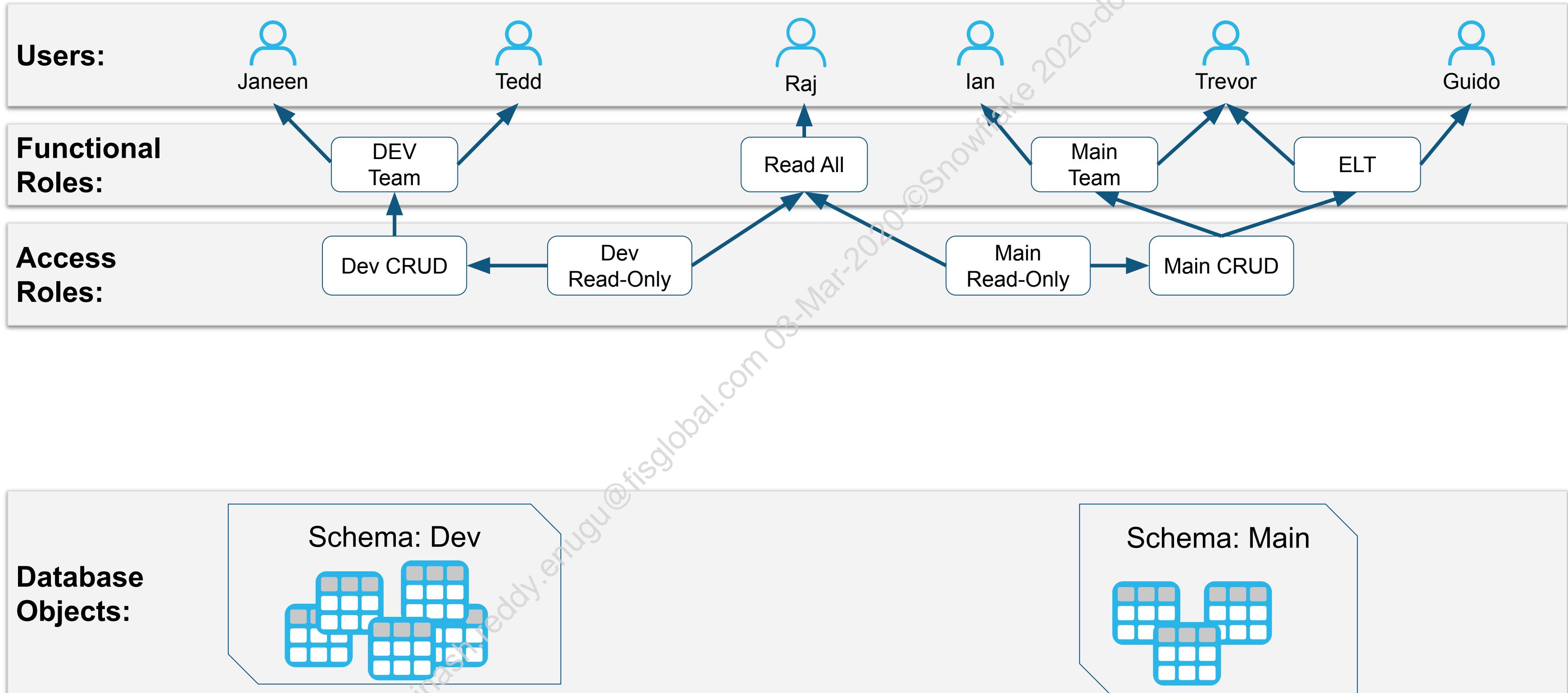
# LAYERED ACCESS CONTROL



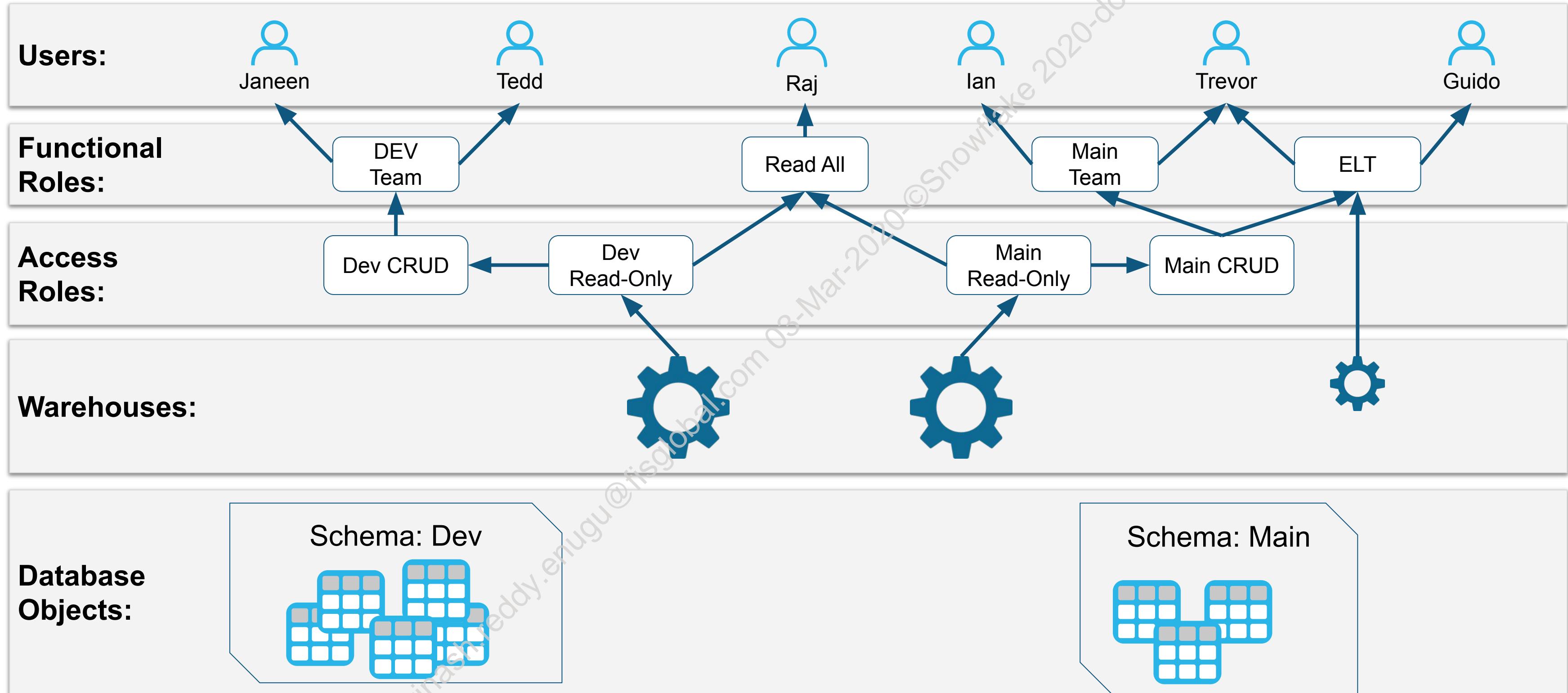
# LAYERED ACCESS CONTROL



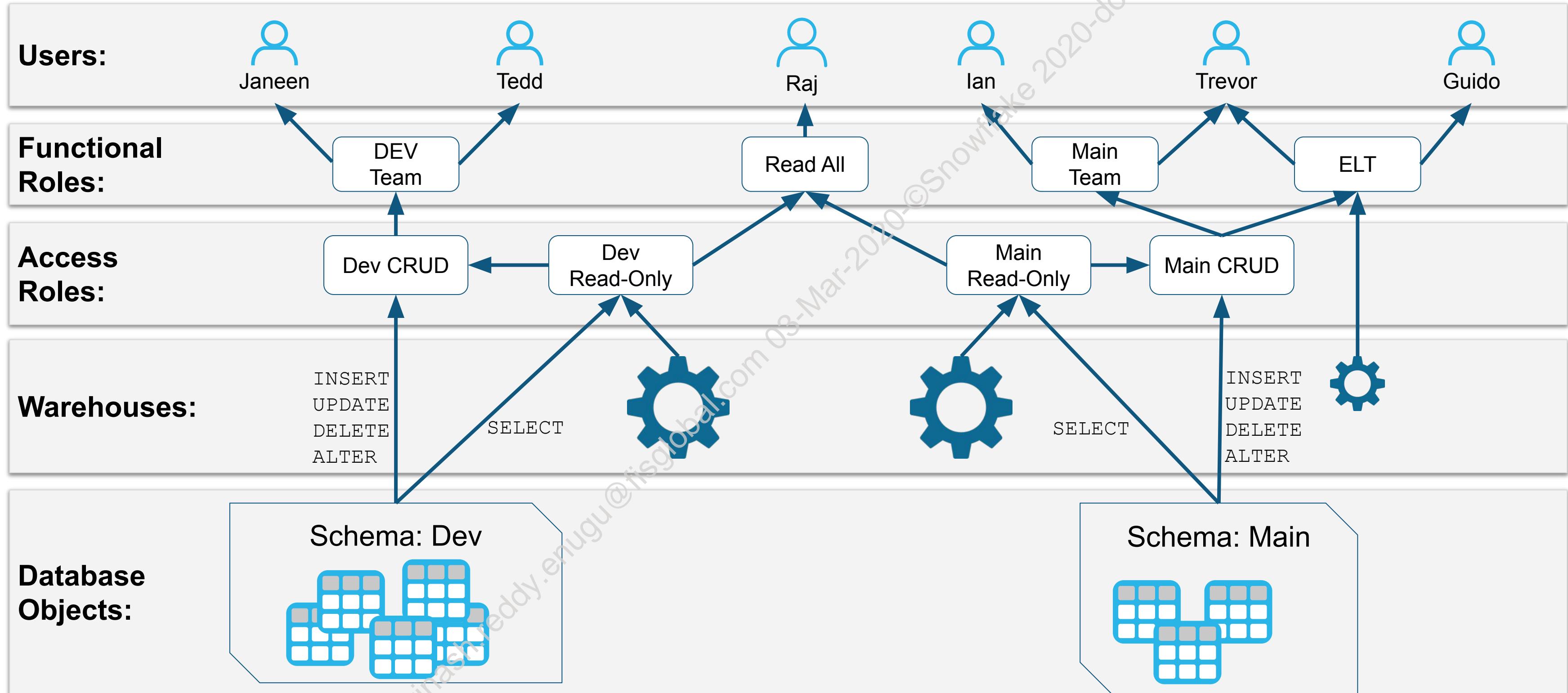
# LAYERED ACCESS CONTROL



# LAYERED ACCESS CONTROL



# LAYERED ACCESS CONTROL



# RESOURCE CONTROL

## CENTRALIZED VERSUS DISTRIBUTED

### CENTRALIZED

- SYSADMIN creates all warehouses and databases, grants privileges

```
CREATE DATABASE db1;
```

```
CREATE WAREHOUSE wh1;
```

```
GRANT USAGE ON WAREHOUSE wh1 TO ROLES
 dba, analyst, developer
```

```
GRANT USAGE ON DATABASE db1 TO ROLES
 dba, analyst, developer;
```

```
GRANT CREATE SCHEMA ON db1 TO ROLES
 dba, analyst, developer
```

### DISTRIBUTED

- SYSADMIN grants ability to create databases/warehouses to specific roles

```
GRANT CREATE DATABASE ON ACCOUNT
 TO ROLE dba;
```

```
GRANT CREATE WAREHOUSE ON ACCOUNT
 TO ROLE warehouse_manager;
```



# Recap

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



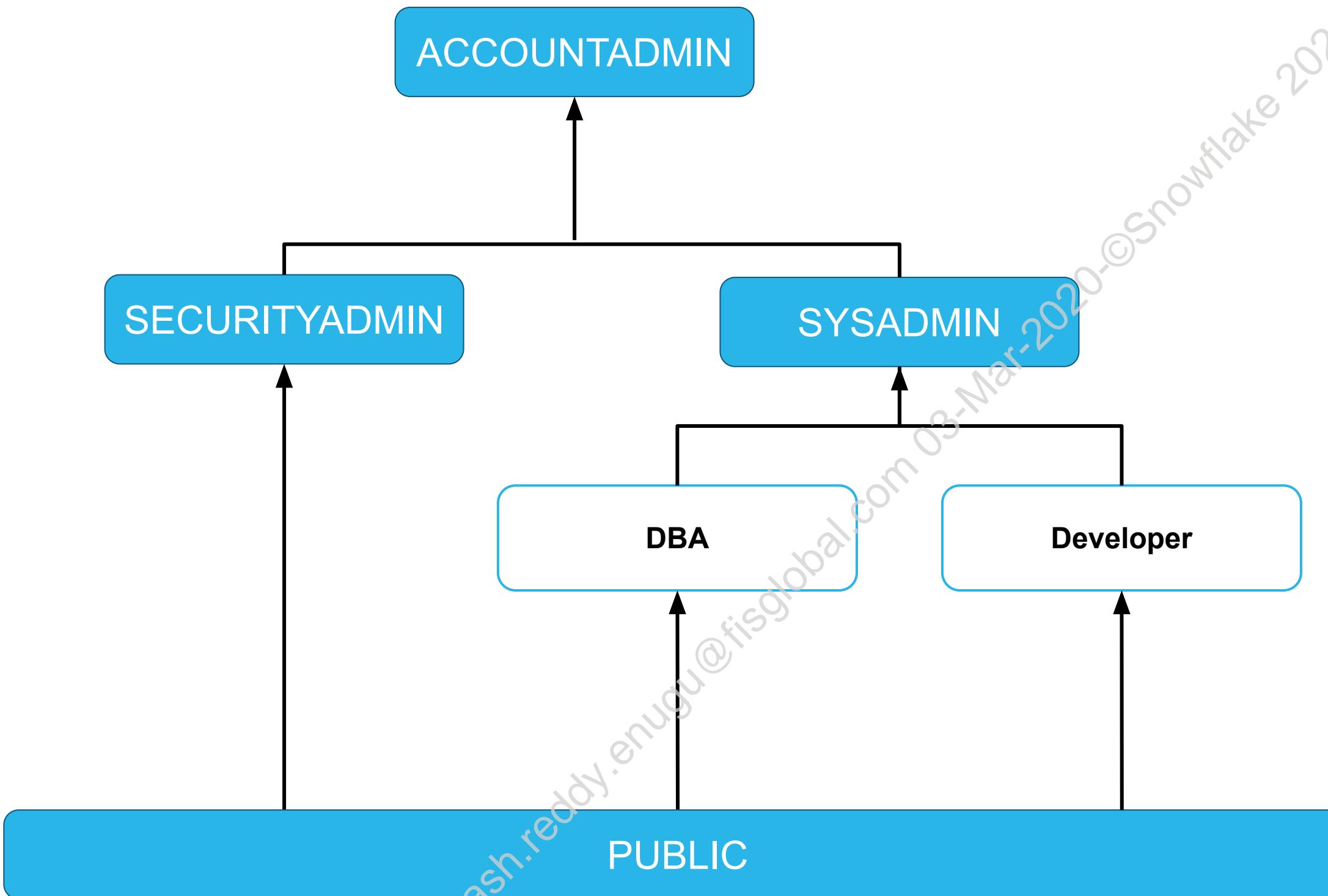
# SCENARIO

## FUN WITH ROLES AND PRIVILEGES

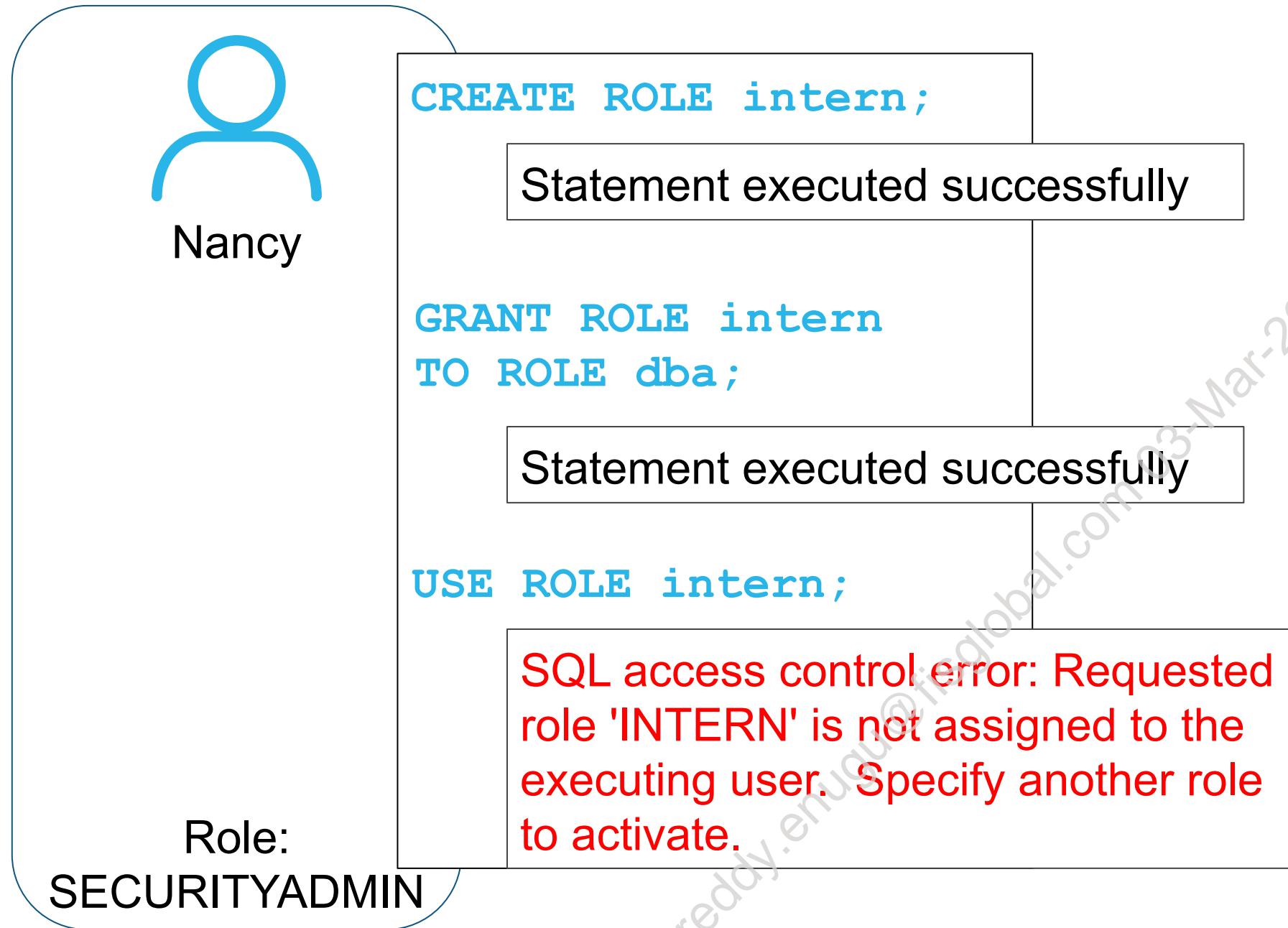
- Nancy is new to Company\_A, who is using Snowflake
- Their previous Snowflake administrator (Roberto) quit to open a gelato shop in Antarctica, and cannot be reached
- Before leaving, Roberto assigned Nancy to the roles SECURITYADMIN and SYSADMIN, and left her a coupon for free gelato
- Management has asked Nancy to create a place where new DBA interns can play with their data warehouse without doing any real harm



# COMPANY\_A CURRENT CONFIGURATION

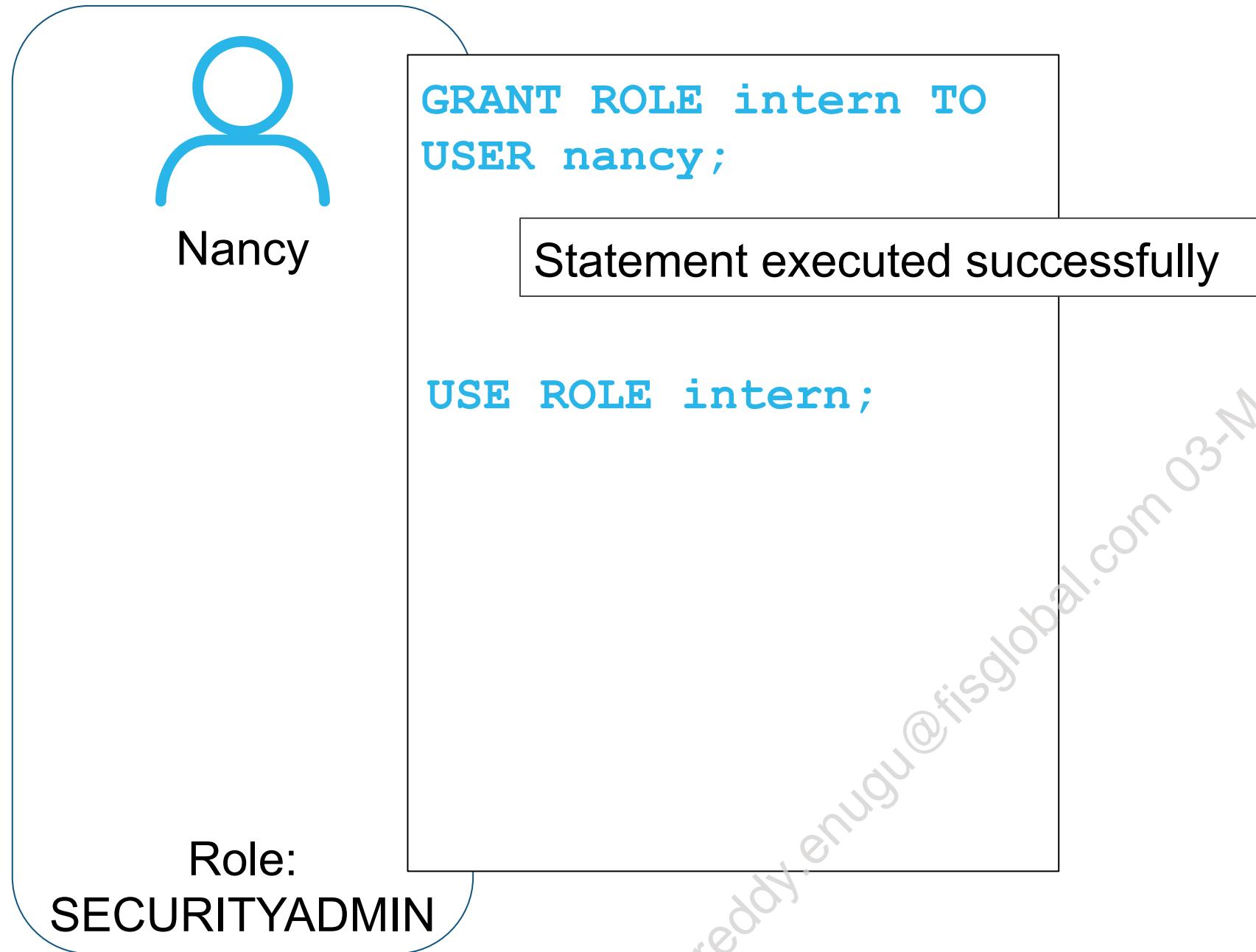


# COMPANY\_A SCENARIO

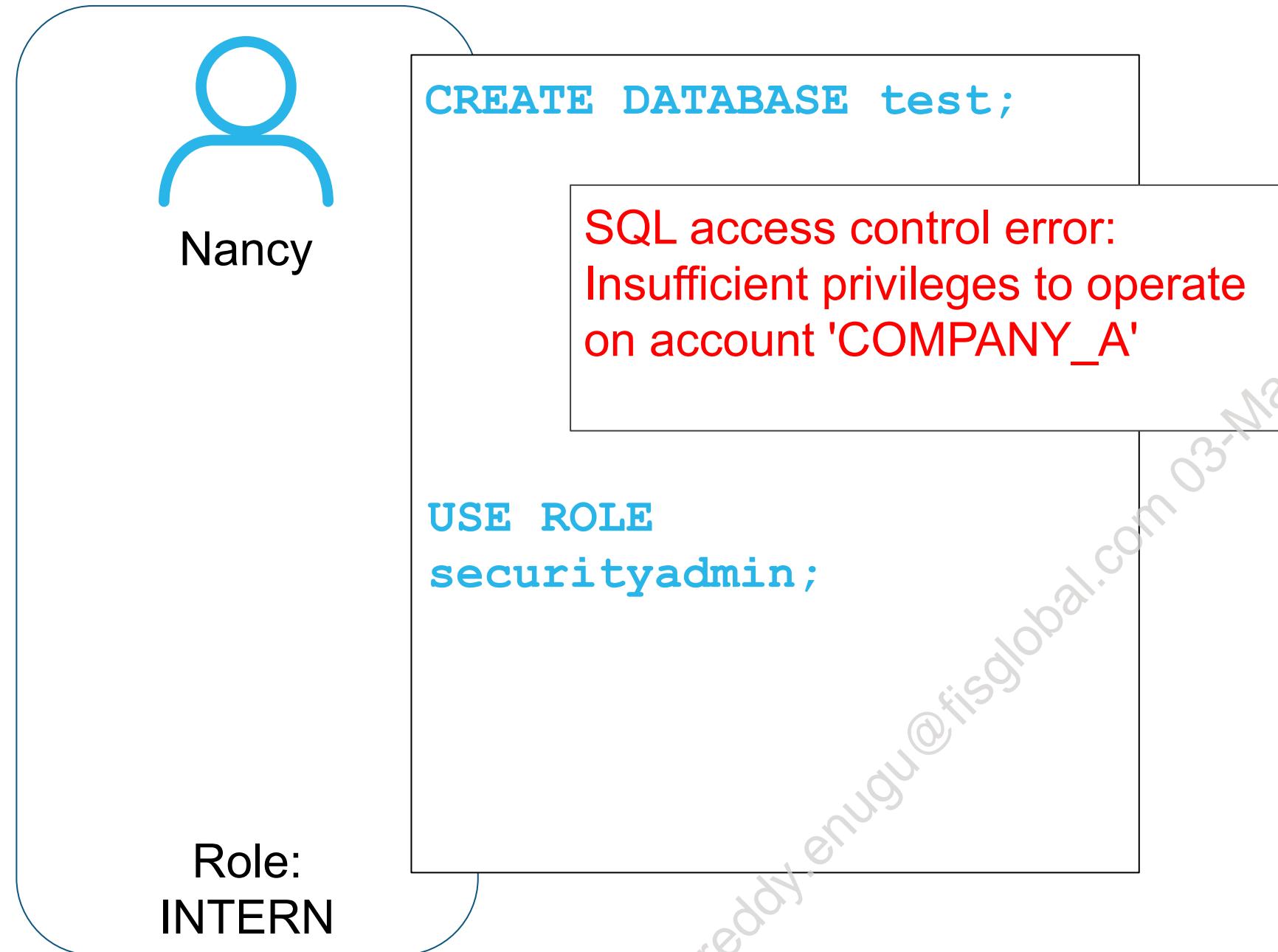


Before anyone can use a role, they must be granted those privileges.

# COMPANY\_A SCENARIO

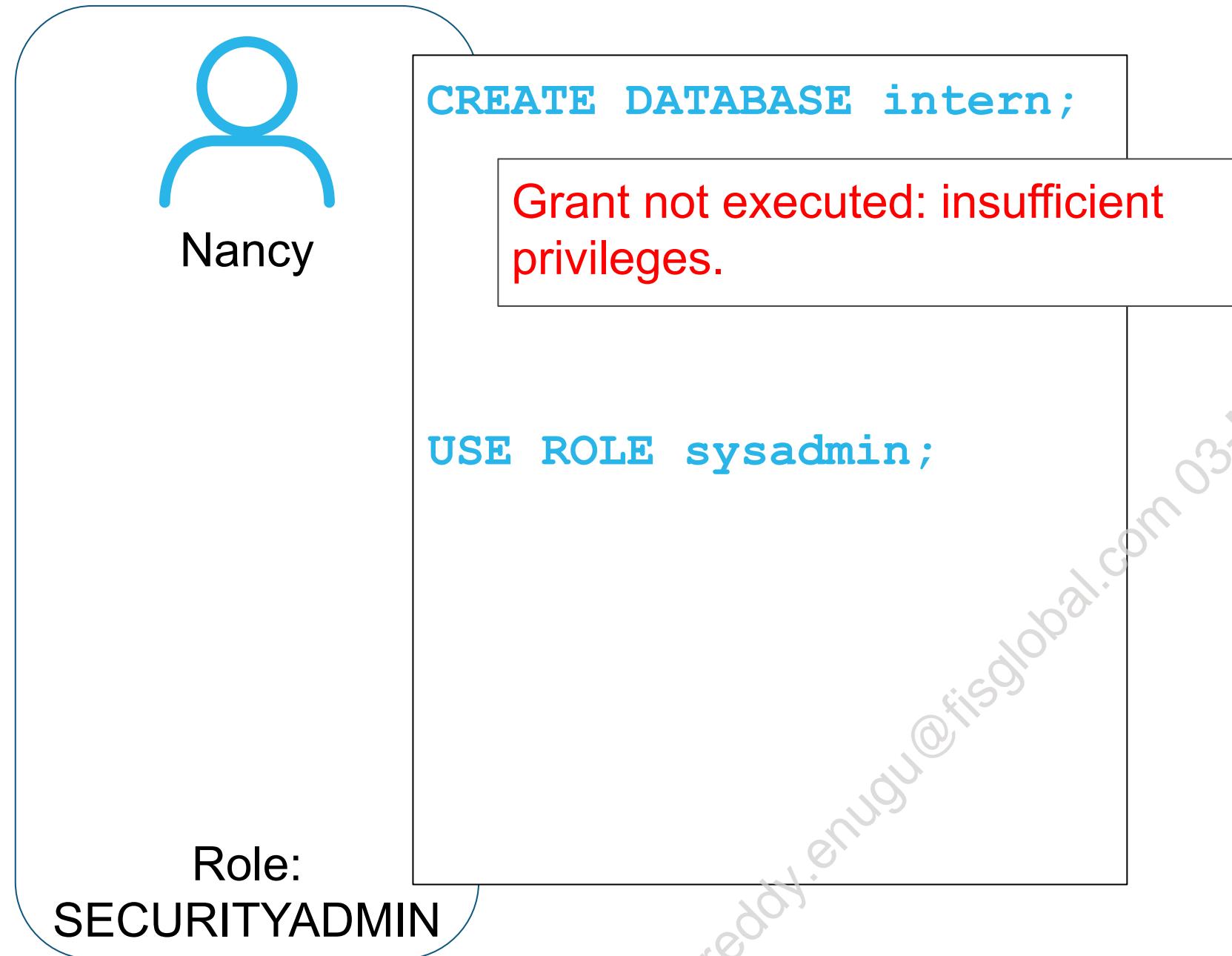


# COMPANY\_A SCENARIO



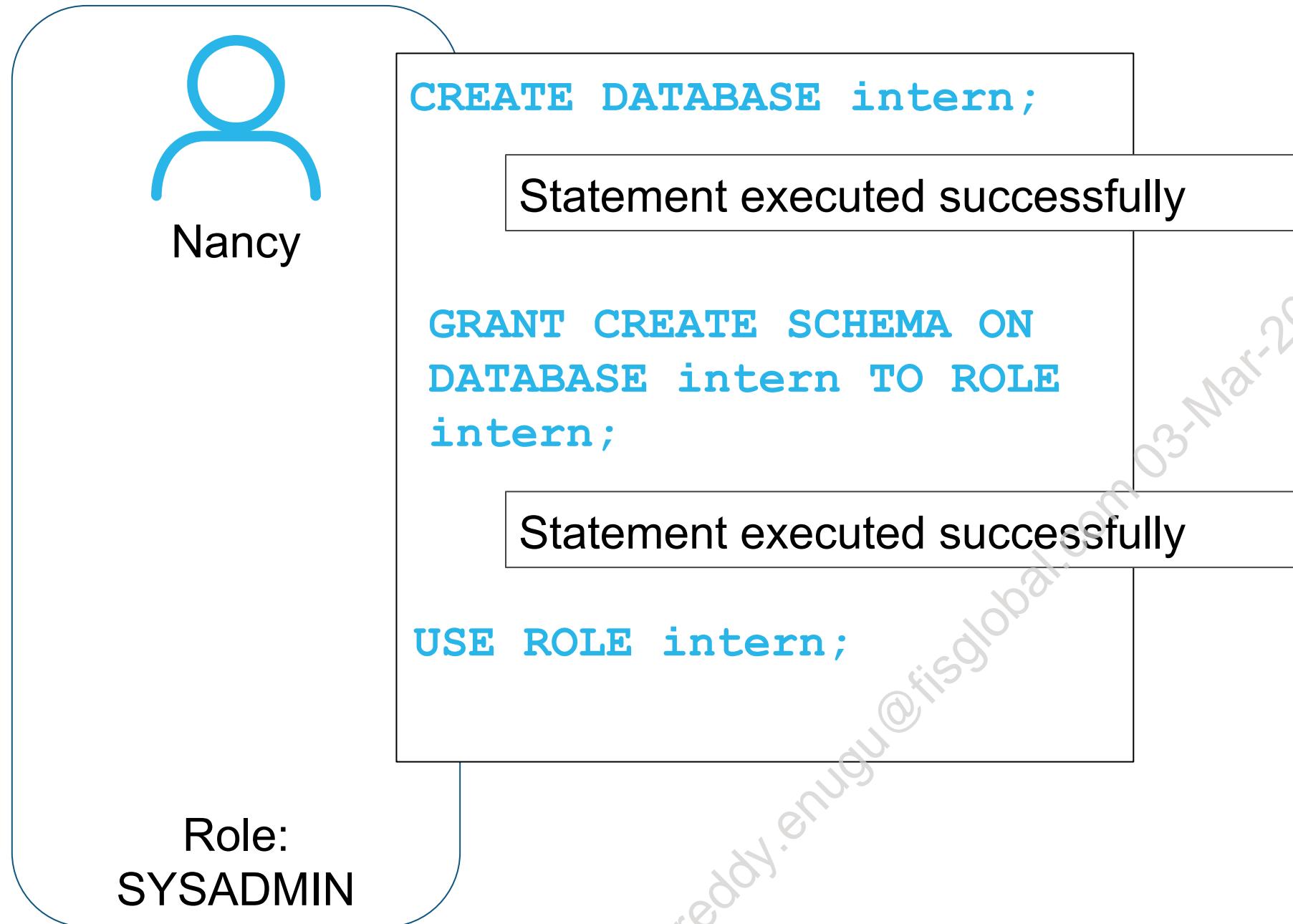
The ROLE intern has not been granted any privileges. As SYSADMIN, Nancy can create databases. But as the role INTERN, she cannot.

# COMPANY\_A SCENARIO

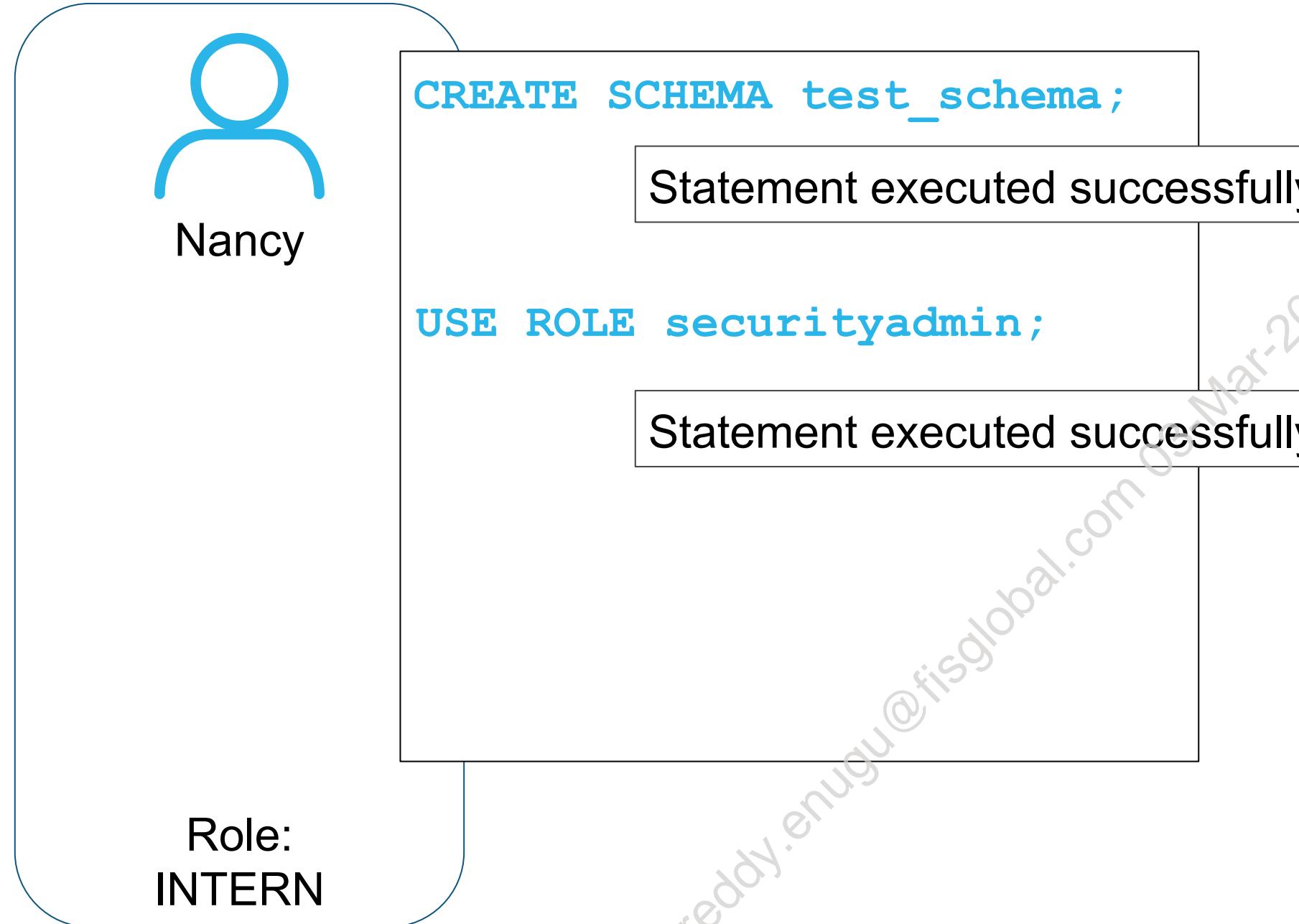


The role SECURITYADMIN does not have the ability to create objects - only users and roles.

# COMPANY\_A SCENARIO



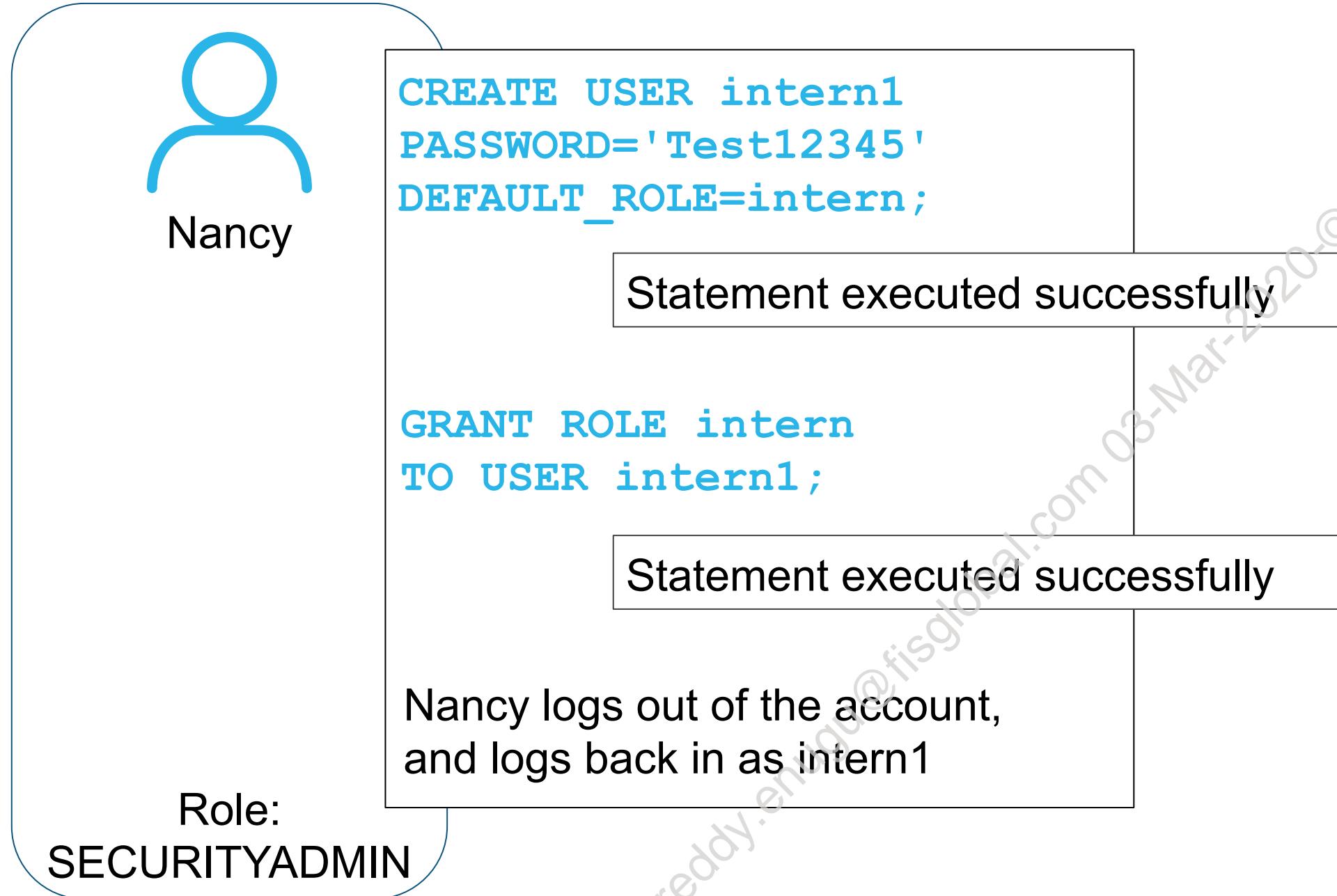
# COMPANY\_A SCENARIO



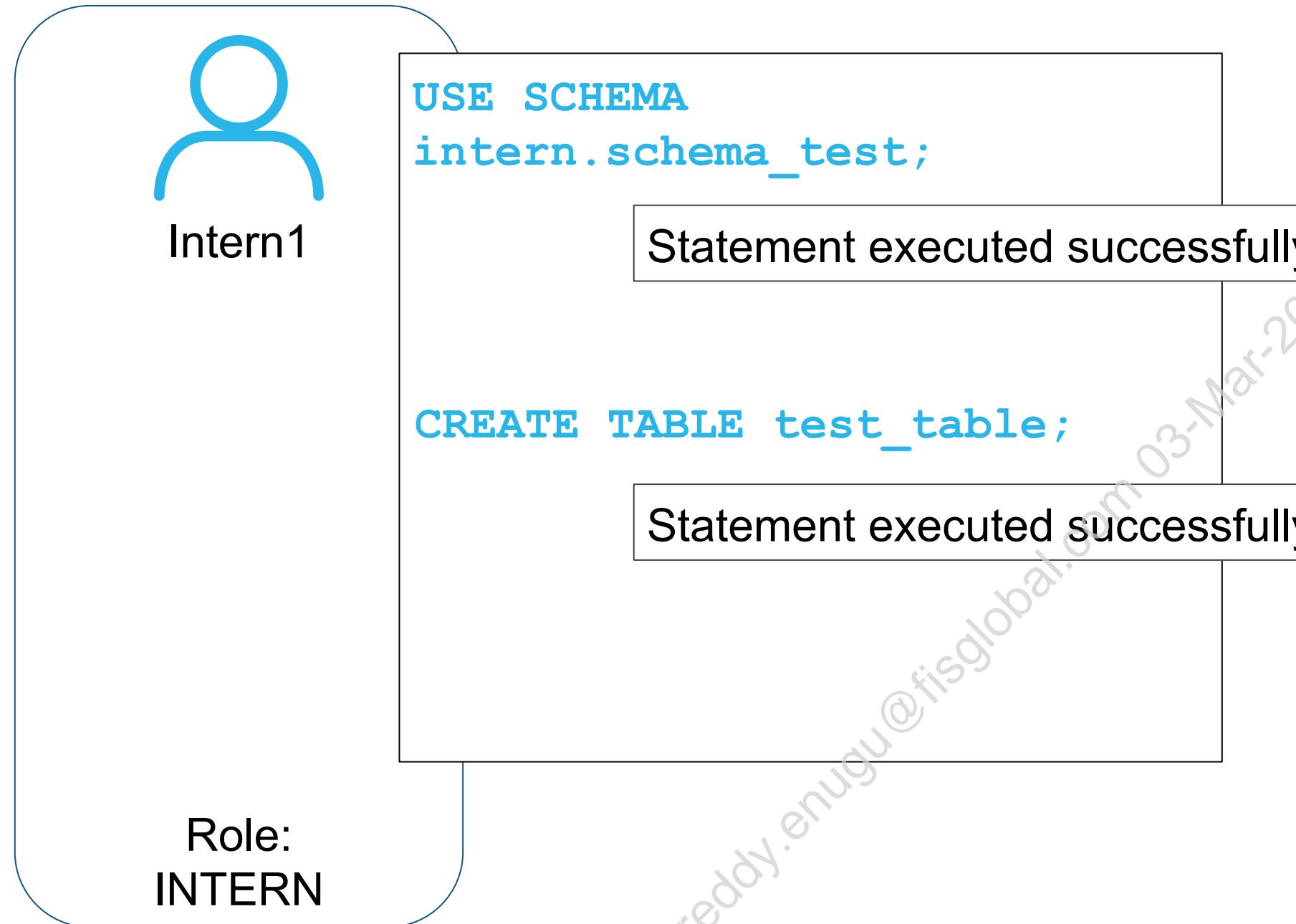
This is not an intern becoming a SECURITYADMIN - this is Nancy using a role she has been granted.

Other users in the INTERN role would not be able to do this.

# COMPANY\_A SCENARIO



# COMPANY\_A SCENARIO



Objects are owned by ROLES, not users. Since the SCHEMA was created by the ROLE intern, anyone using that role is an OWNER, and can do anything with that schema.

# Module 7: Access Control and User Management

## Exercise 7.1: Managing Roles

20 minutes

### Tasks:

- Creating Parent & Child Roles
- Granting Privileges on Roles
- Dropping Roles



# Semi-Structured Data

## Module 8

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Overview
- Semi-Structured Data Types
- Query Semi-Structured Data

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# Overview

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# NATIVE SUPPORT

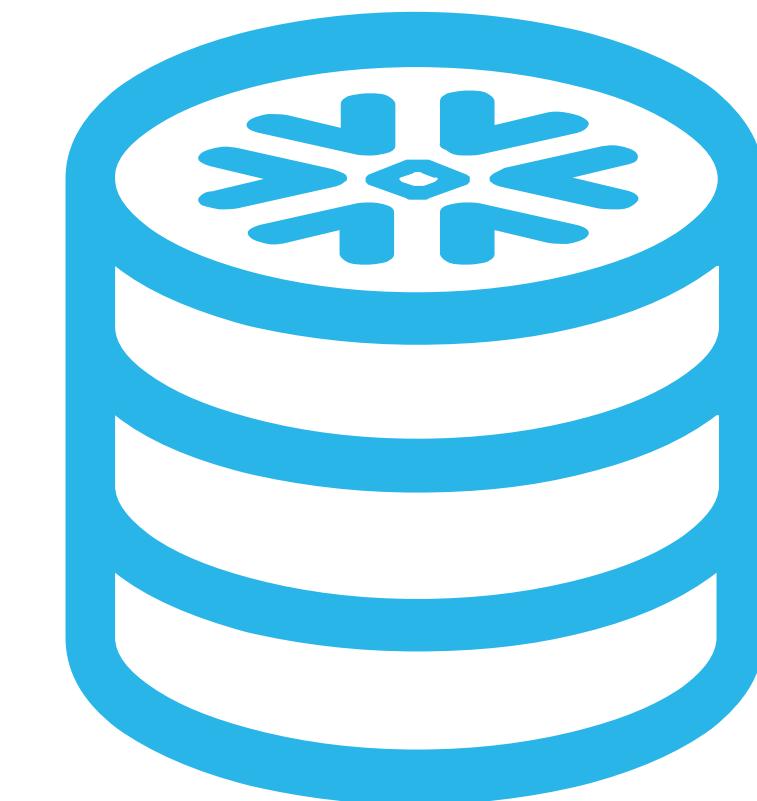
- Native support for all your data
  - Data Lakes with machine-generated data
  - Sensor IoT type data
  - Semi-Structured origin formats (JSON, Avro, ORC, Parquet, or XML)
- First level new data types
  - Statistics are gathered and maintained, including subfields for quicker query access and effective partition pruning for performance
- Support for all SQL operations (joins, group by, order by, ...)
  - Native syntax for accessing data even in semi-structured fields



# LOADING SEMI-STRUCTURED DATA



CREATE FILE FORMAT  
CREATE TABLE  
COPY INTO TABLE



# JSON FILE FORMAT OPTIONS

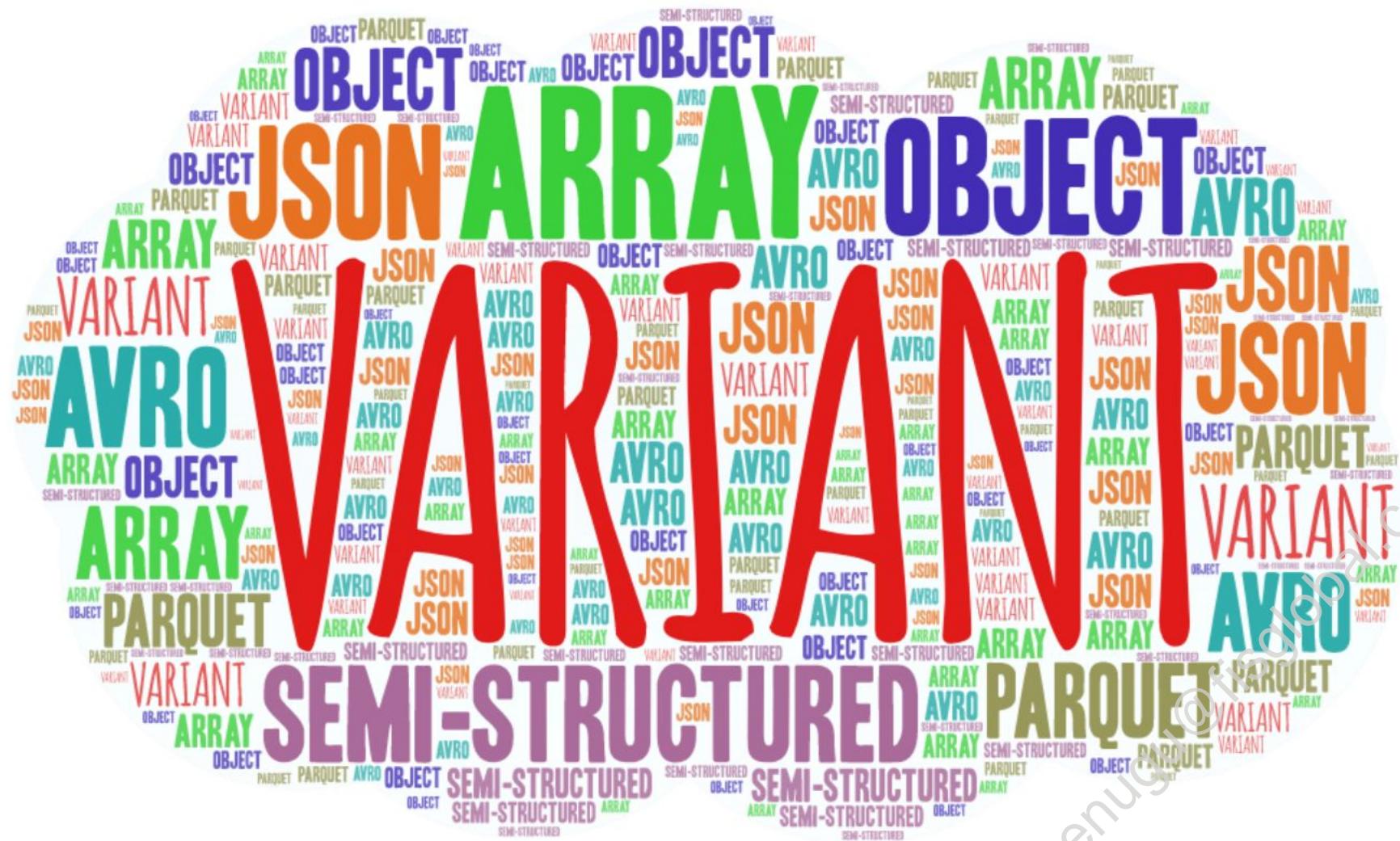
| Option               | Default                    | Details                                                                                                                                                      |
|----------------------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMPRESSION          | Load: Auto<br>Unload: GZIP | If source files is compressed with Brotli, must specify (all others can be auto-detected)                                                                    |
| FILE_EXTENSION       | .JSON                      | Only used for unloading. Specifies file extension to use                                                                                                     |
| ENABLE_OCTAL         | FALSE                      | Only used for loading. If TRUE, enables parsing octal numbers                                                                                                |
| ALLOW_DUPLICATE      | FALSE                      | Only used for loading. If TRUE, allows duplicate object field names (only the last one will be preserved)                                                    |
| STRIP_OUTER_ARRAY    | FALSE                      | Only used for loading. If TRUE, JSON parser will remove outer brackets [ ]                                                                                   |
| STRIP_NULL_VALUES    | FALSE                      | Only used for loading. If TRUE, JSON parser will remove object fields or array elements containing NULL                                                      |
| IGNORE_UTF8_ERRORS   | FALSE                      | Only used for loading. If TRUE, invalid UTF-8 sequences silently replaced with U+FFFD                                                                        |
| SKIP_BYTE_ORDER_MARK | TRUE                       | Only used for loading. If TRUE, BOM is skipped if present in data file. Otherwise, BOM could cause error, or could be merged into first column in the table. |

# Semi-Structured Data Types

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# SEMI-STRUCTURED DATA TYPES



- VARIANT – holds values of standard SQL type, arrays, and objects
  - Non-native values (e.g., dates and timestamps) are stored as strings in a variant column  
Operations could be slower than when stored in a relational column as the corresponding data type
- OBJECT – a collection of key-value pairs
  - The value is a VARIANT
- ARRAY – Arrays of varying sizes
  - The value is a VARIANT
- Snowflake collects metadata on these columns for use by optimizer for pruning

# VARIANT DATA TYPE EXAMPLE

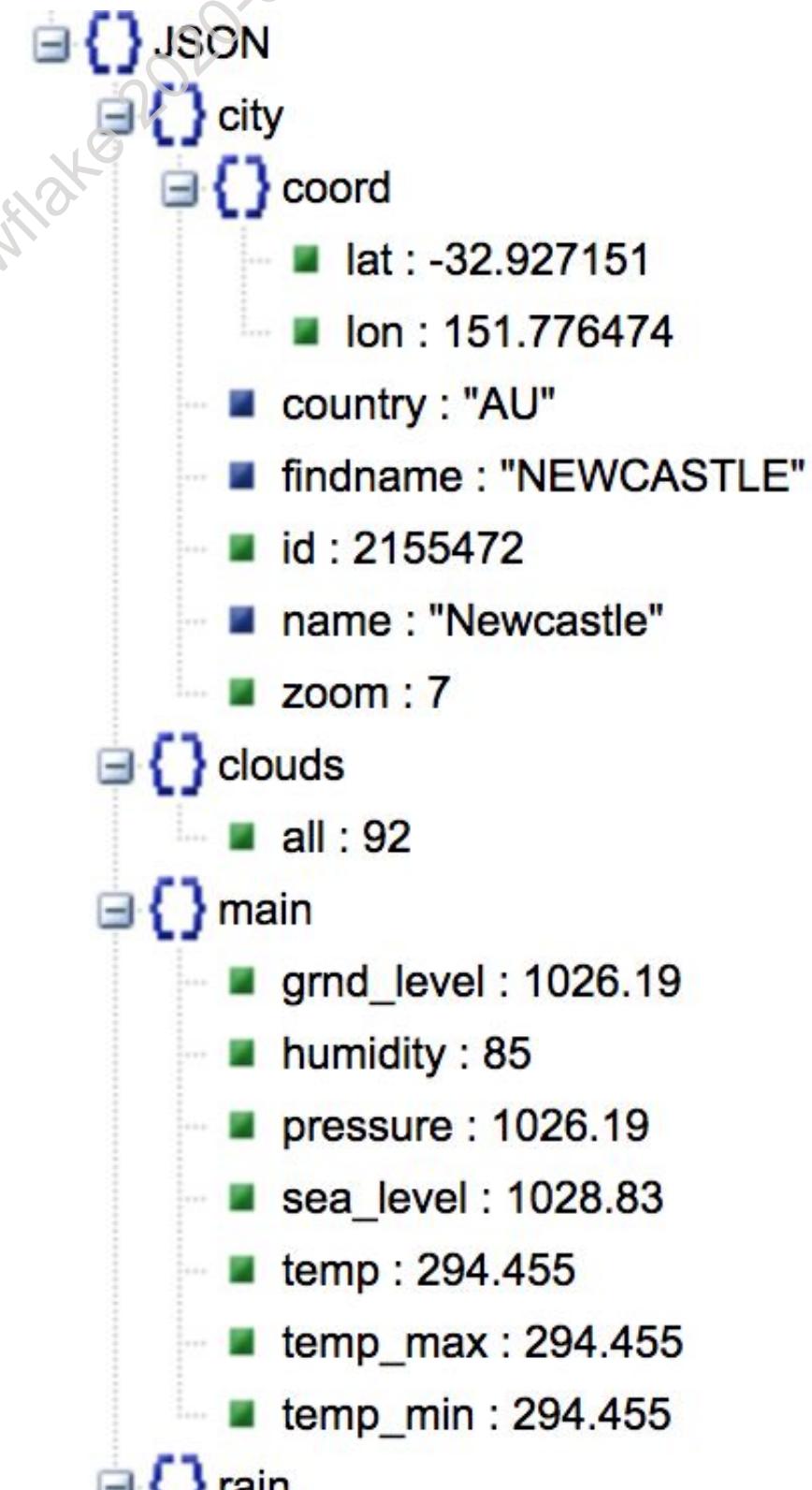
## SAMPLE WEATHER DATA SET IN SNOWFLAKE

 **WEATHER**

▼ Tables

- DAILY\_14\_TOTAL
- DAILY\_16\_TOTAL
- HOURLY\_14\_TOTAL
- HOURLY\_16\_TOTAL
- WEATHER\_14\_TOTAL

```
CREATE TABLE
WEATHER_16_TOTAL
(V VARIANT,
T TIMESTAMP_NTZ(9))
```



# Module 8: Semi-Structured Data

## Exercise 8.1: Load Semi-Structured Data

15 minutes

### Tasks:

- Load Semi-Structured Parquet Data
- Load Semi-Structured JSON Data



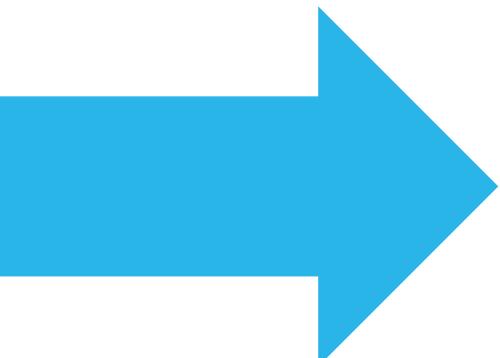
# Querying Semi-Structured Data

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# QUERY RICH HIERARCHICAL STRUCTURES IN SQL

```
humidity": 0, "pressure": 1018.74, "rain": 1.39, "speed": 2.3, "te...
 2.3, "temp": { "day": 303.92, "eve": 301.33, "max": 303.92, "min": 299.28, "morn...
 ed": 0.86, "temp": { "day": 301.55, "eve": 301.36, "max": 302.3, "min": 297.93, "r...
 91, "temp": { "day": 302.64, "eve": 301.77, "max": 304.03, "min": 300.7, "morn...
 ': 1.5, "temp": { "day": 302.69, "eve": 299.63, "max": 302.69, "min": 298.11, "mo...
 1.22, "temp": { "day": 304.08, "eve": 300.31, "max": 304.08, "min": 299.07, "mo...
 speed": 4.06, "temp": { "day": 297.2, "eve": 299.56, "max": 299.86, "min": 296.65...
 np": { "day": 300.7, "eve": 300.7, "max": 300.7, "min": 300.7, "morn": 300.7, "nig...
 1.44, "temp": { "day": 302.57, "eve": 299.61, "max": 302.57, "min": 297.78, "mo...
 ': 1.03, "temp": { "day": 303.41, "eve": 299.69, "max": 303.41, "min": 298.83, "m...
 1.53, "temp": { "day": 304.41, "eve": 300.29, "max": 304.41, "min": 298.32, "mo...
```



|  | MIN      | MAX      |
|--|----------|----------|
|  | 294.1700 | 305.2000 |
|  | 294.1700 | 305.2000 |
|  | 295.0300 | 303.5100 |
|  | 295.0300 | 303.5100 |
|  | 295.3200 | 300.2800 |
|  | 295.3200 | 300.2800 |
|  | 295.4600 | 304.5400 |
|  | 295.4600 | 304.5400 |
|  | 295.5400 | 299.6000 |
|  | 295.5400 | 299.6000 |



# QUERY RICH HIERARCHICAL STRUCTURES IN SQL

```
SELECT
 value:temp:min::number(10, 4) AS min,
 value:temp:max::number(10, 4) AS max
FROM
 daily_14_total dt,
 LATERAL FLATTEN(input=> dt.v:data)
WHERE
 dt.t = (SELECT MAX(t) FROM daily_14_total)
ORDER BY min
LIMIT 10;
```



# SUPPORTS ALL SQL OPERATIONS

- JOINS
- GROUP BY
- SORT



# DATA FUNCTIONS

## Array Creation and Manipulation

ARRAY\_AGG  
ARRAY\_APPEND  
ARRAY\_CAT  
ARRAY\_COMPACT  
ARRAY\_CONSTRUCT  
ARRAY\_CONSTRUCT\_COMPACT  
ARRAY\_CONTAINS  
ARRAY\_INSERT  
ARRAY\_POSITION  
ARRAY\_PREPEND  
ARRAY\_SIZE  
ARRAY\_SLICE  
ARRAY\_TO\_STRING  
ARRAYS\_OVERLAP

## Object Creation and Manipulation

OBJECT\_AGG  
OBJECT\_CONSTRUCT  
OBJECT\_DELETE  
OBJECT\_INSERT

## JSON and XML Parsing

CHECK\_JSON  
CHECK\_XML  
PARSE\_JSON  
PARSE\_XML  
STRIP\_NULL\_VALUE

## Extraction

FLATTEN  
GET  
GET\_PATH ,:  
XMLGET

## Conversion

TO\_ARRAY  
TO\_JSON  
TO\_OBJECT  
TO\_VARIANT  
TO\_XML

## Casting

AS\_<object\_type>  
AS\_ARRAY  
AS\_BINARY  
AS\_CHAR , AS\_VARCHAR  
AS\_DATE  
AS\_DECIMAL , AS\_NUMBER  
AS\_DOUBLE , AS\_REAL  
AS\_INTEGER  
AS\_OBJECT  
AS\_TIME  
AS\_TIMESTAMP\_\*

## Type Checking

IS\_<object\_type>  
IS\_ARRAY  
IS\_BOOLEAN  
IS\_BINARY  
IS\_CHAR , IS\_VARCHAR  
IS\_DATE , IS\_DATE\_VALUE  
IS\_DECIMAL  
IS\_DOUBLE , IS\_REAL  
IS\_INTEGER  
IS\_NULL\_VALUE  
IS\_OBJECT  
IS\_TIME  
IS\_TIMESTAMP\_\*



# ACCESSING VALUES

Can access value in a VARIANT column for a particular key

## Colon - column:key

```
SELECT value:humidity
AS pct_humidity
FROM
my_json_table
```

...

| PCT_HUMIDITY |
|--------------|
| =====        |
| 50           |

## Brackets - column['key']

```
SELECT value['humidity']
AS pct_humidity
FROM
my_json_table
```

...

| PCT_HUMIDITY |
|--------------|
| =====        |
| 50           |



# CASTING DATA FROM VARIANTS

- VARIANTS are often just strings, arrays or numbers.
- Without CASTing, they remain VARIANT object types
- Cast them to SQL data types using the :: operator, followed by the type you want to convert them to

```
SELECT
 value:humidity::number(10,2)
AS pct_humidity
FROM
 daily_14_total dt,
 LATERAL FLATTEN(input=> dt.v:data)
LIMIT 3;
```

| Row | PCT_HUMIDITY |
|-----|--------------|
| 1   | 82.00        |
| 2   | 80.00        |
| 3   | 78.00        |



# FLATTENING DATA IN ARRAYS

- VARIANTS may contain nested elements (arrays that contain the data)
- FLATTEN() extracts data from nested arrays
- Almost always used with a LATERAL join (to refer to columns from other tables, views, or table function)
- LATERAL FLATTEN (input=> *expression* [ *options* ] )

***input =>*** The expression or column that will be unseated into rows.  
The data must be of type VARIANT, OBJECT, or ARRAY.



# FLATTENING DATA

- FLATTEN returns a fixed set of columns

```
SELECT * FROM
TABLE (FLATTEN(input => PARSE_JSON('[1,2,3]')));
```

| SEQ | KEY  | PATH | INDEX | VALUE | THIS    |
|-----|------|------|-------|-------|---------|
| 1   | NULL | [0]  | 0     | 1     | [1,2,3] |
| 1   | NULL | [1]  | 1     | 2     | [1,2,3] |
| 1   | NULL | [2]  | 2     | 3     | [1,2,3] |

- Columns of source table for FLATTEN are also accessible



# COLUMNS RETURNED BY FLATTEN

|              |                                                                                                                                                                             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SEQ</b>   | <ul style="list-style-type: none"><li>• Unique sequence number associated with input record</li><li>• Not guaranteed to be gap-free or ordered any particular way</li></ul> |
| <b>KEY</b>   | For maps or objects, this column contains the key to the exploded value                                                                                                     |
| <b>PATH</b>  | Path to the element within a data structure that needs to be flattened                                                                                                      |
| <b>INDEX</b> | Index of the element, if it is an array; otherwise NULL                                                                                                                     |
| <b>VALUE</b> | Value of the element of the flattened array or object                                                                                                                       |
| <b>THIS</b>  | The element being flattened (useful in recursive flattening)                                                                                                                |



# Module 8: Semi-Structured Data

**DEMO:** View, Query, and Flatten Semi-Structured Data

**10 minutes**

- WEATHER data in SNOWFLAKE\_SAMPLE\_DATA



# Module 8: Semi-Structured Data

## Exercise 8.2: Explore Sample Weather Data

30 minutes

### Tasks:

- Review the Data
- Extract & Transform the Data
- Try some queries on your own



## Module 9

# Snowflake Caching

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Overview
- Query Result Cache
- Metadata Cache
- Data Cache in Compute Cluster

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

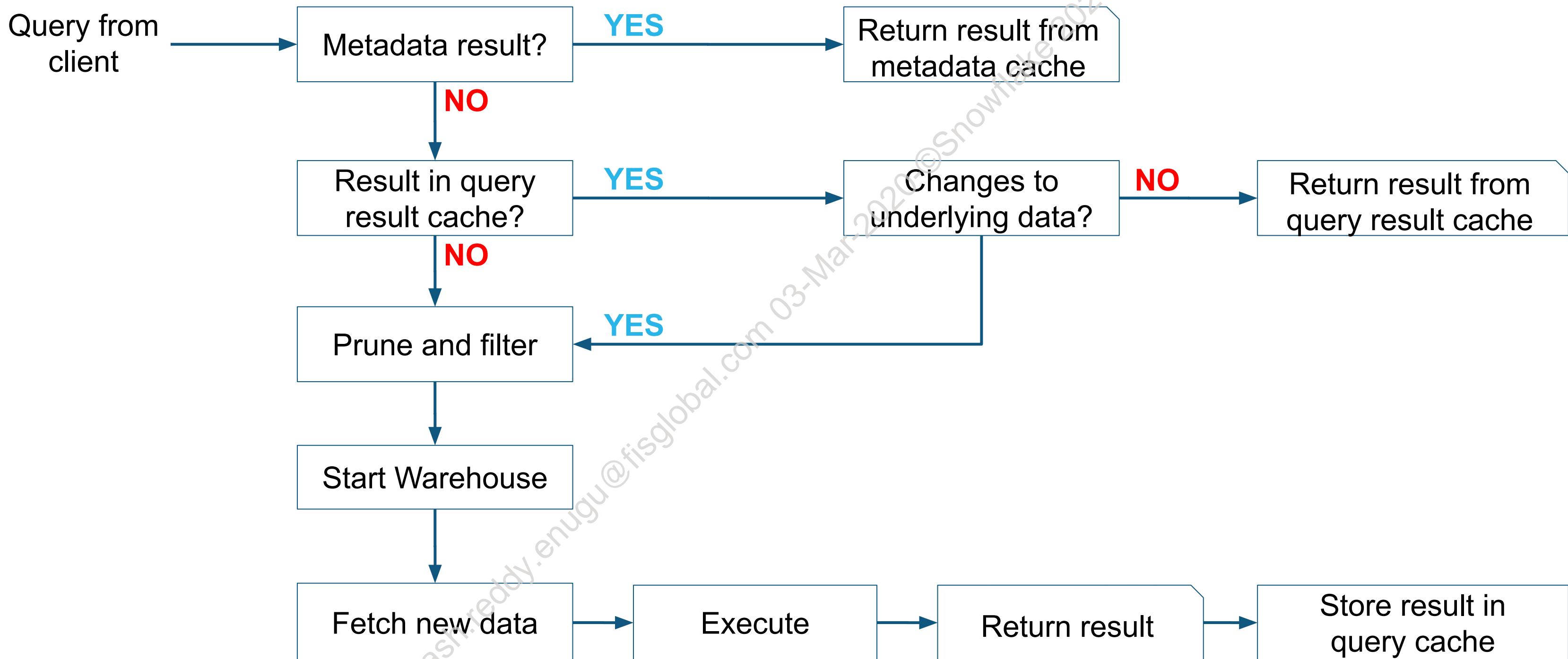


# Overview

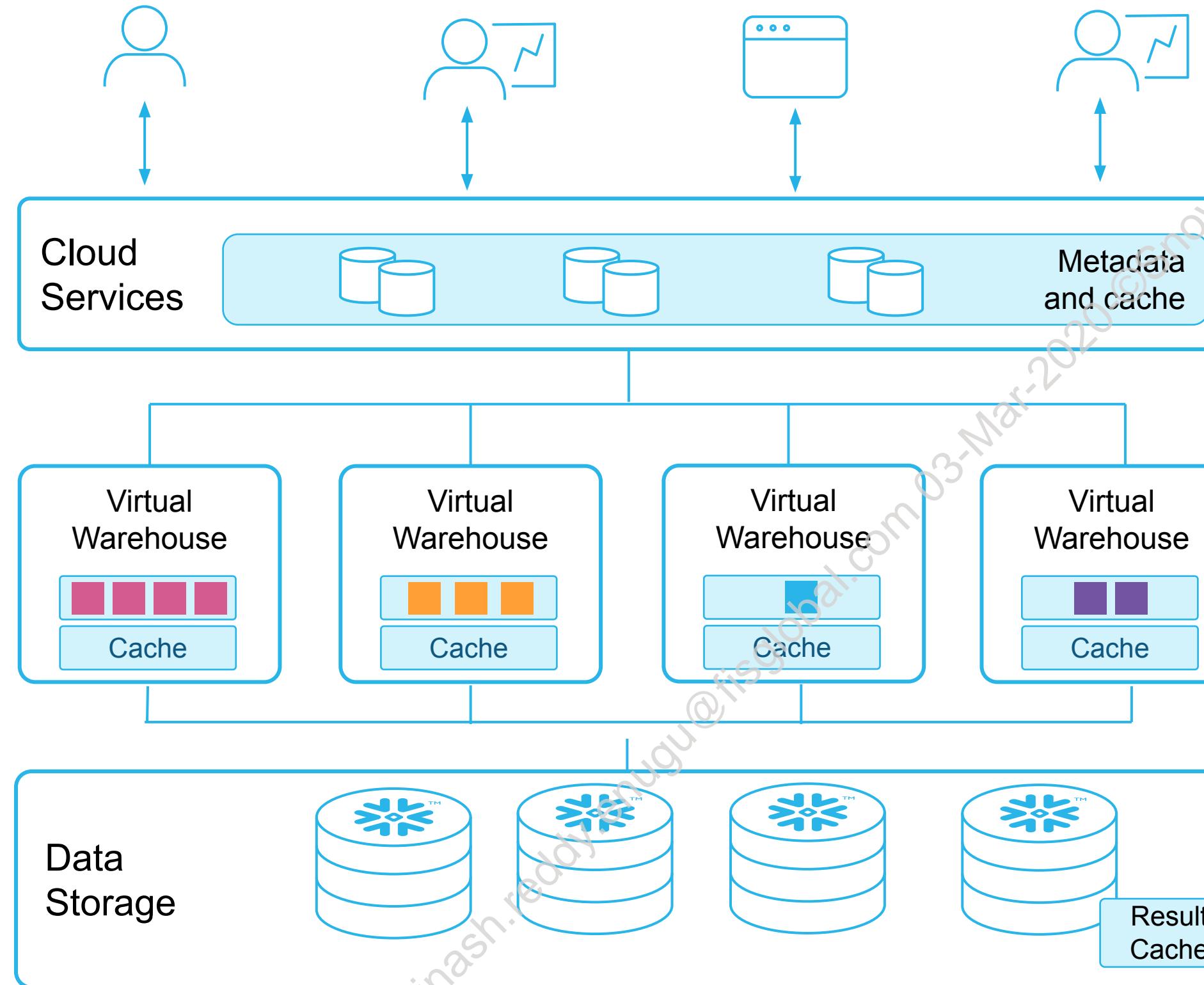
avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# LIFE CYCLE OF A QUERY



# CACHING IN SNOWFLAKE



## METADATA CACHE

Metadata cached for fast access during query processing

## WAREHOUSE DATA CACHE

Active working set transparently cached on virtual warehouse SSD

## QUERY RESULT CACHE

Results sets cached for reuse without requiring compute (e.g., static dashboard queries)



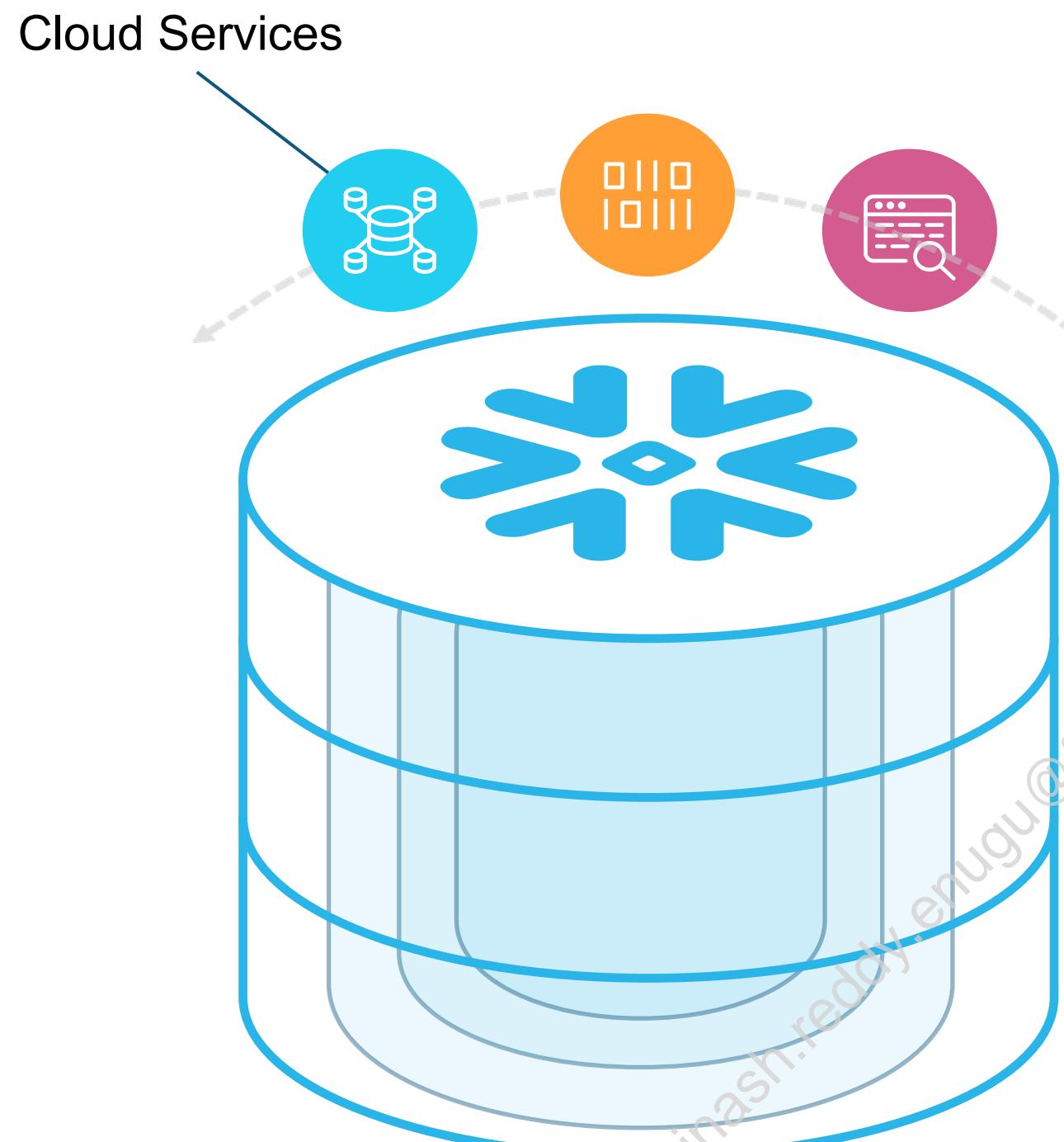
# Query Result Cache

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# QUERY RESULT CACHE

## Persisted Query Results



- Query results are stored and managed by the Cloud Services layer
- Used if the identical query is run, and base tables have not changed
- Available to all users in the same role where the query was run
- Remains in QR cache for 24 hours



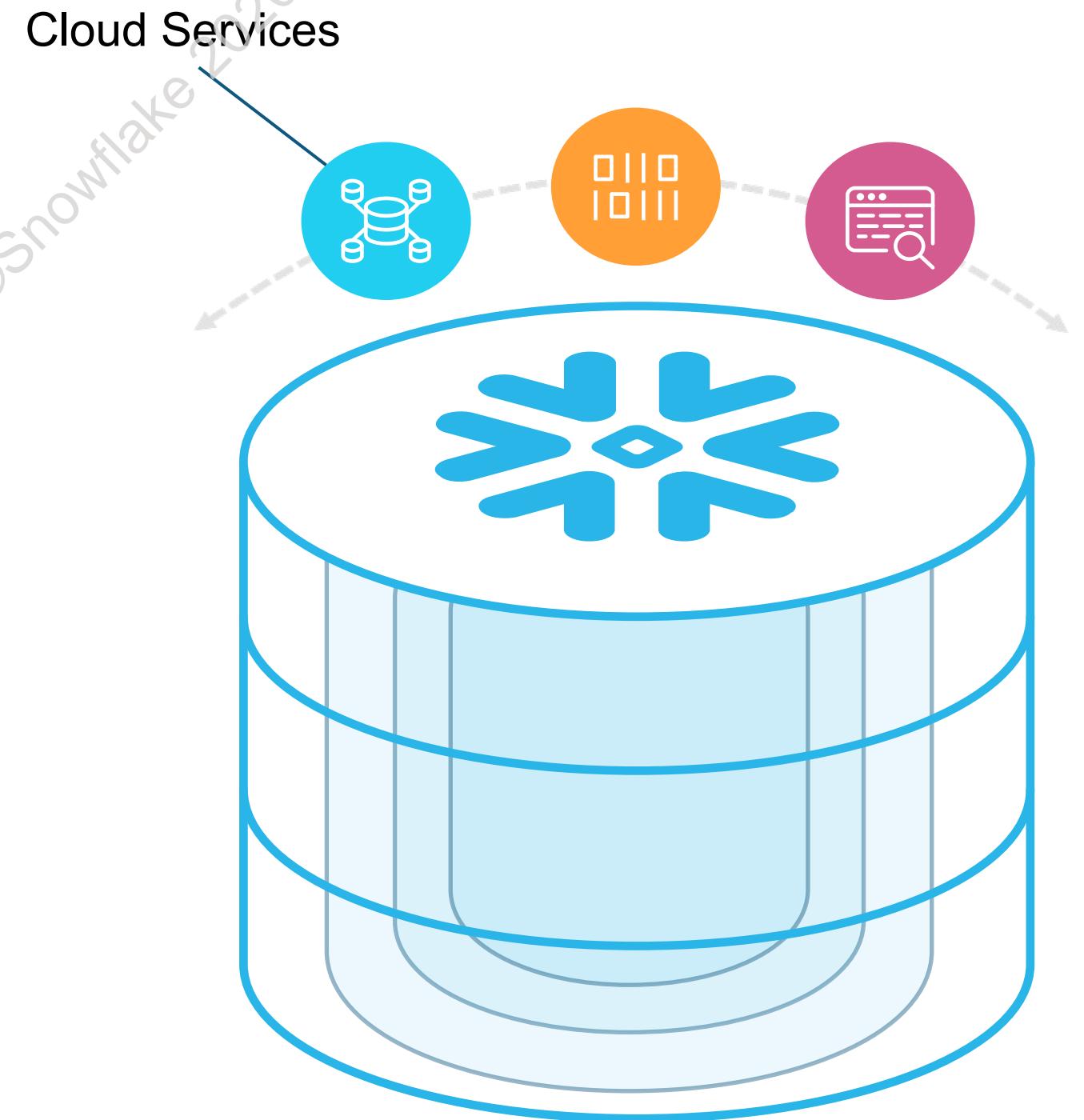
# QUERY RESULT CACHE

## Use Cases

- Static dashboards
- Queries with significant computing
  - Semi-structured data analysis
  - Aggregates
- Queries that are run frequently or are complex

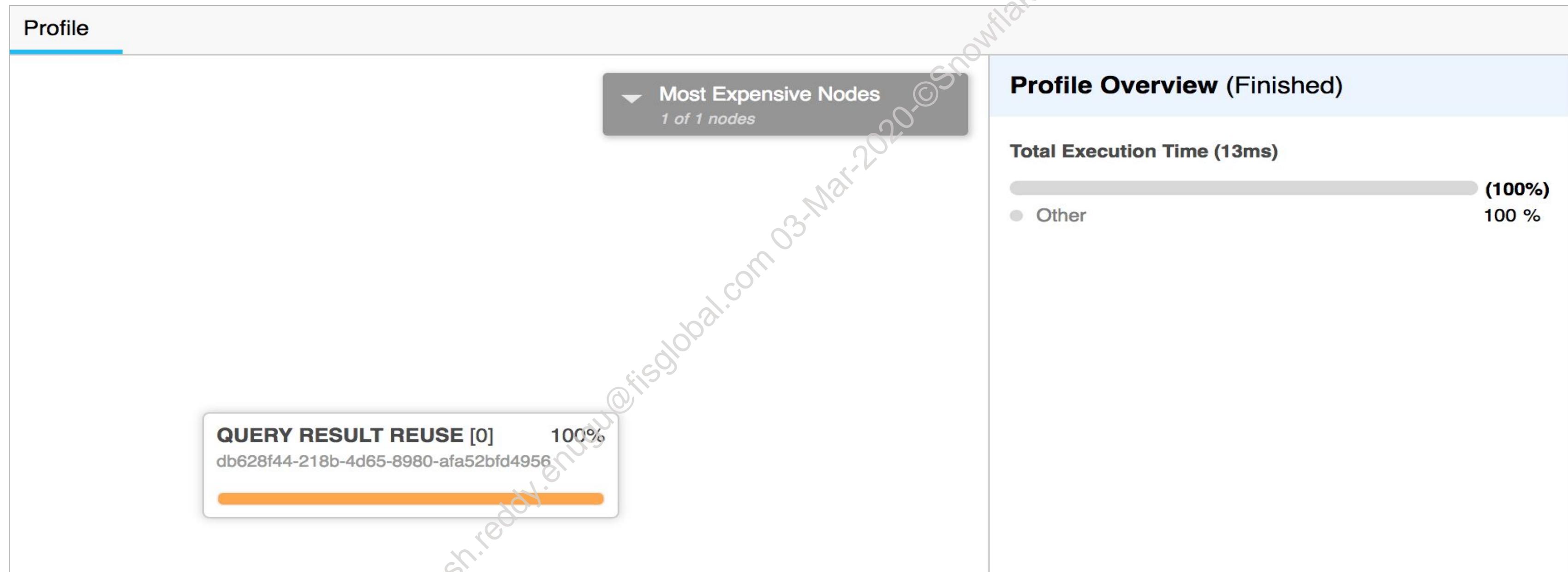
## Benefits

- Fast
- No virtual warehouse is used
- Will never give you stale results



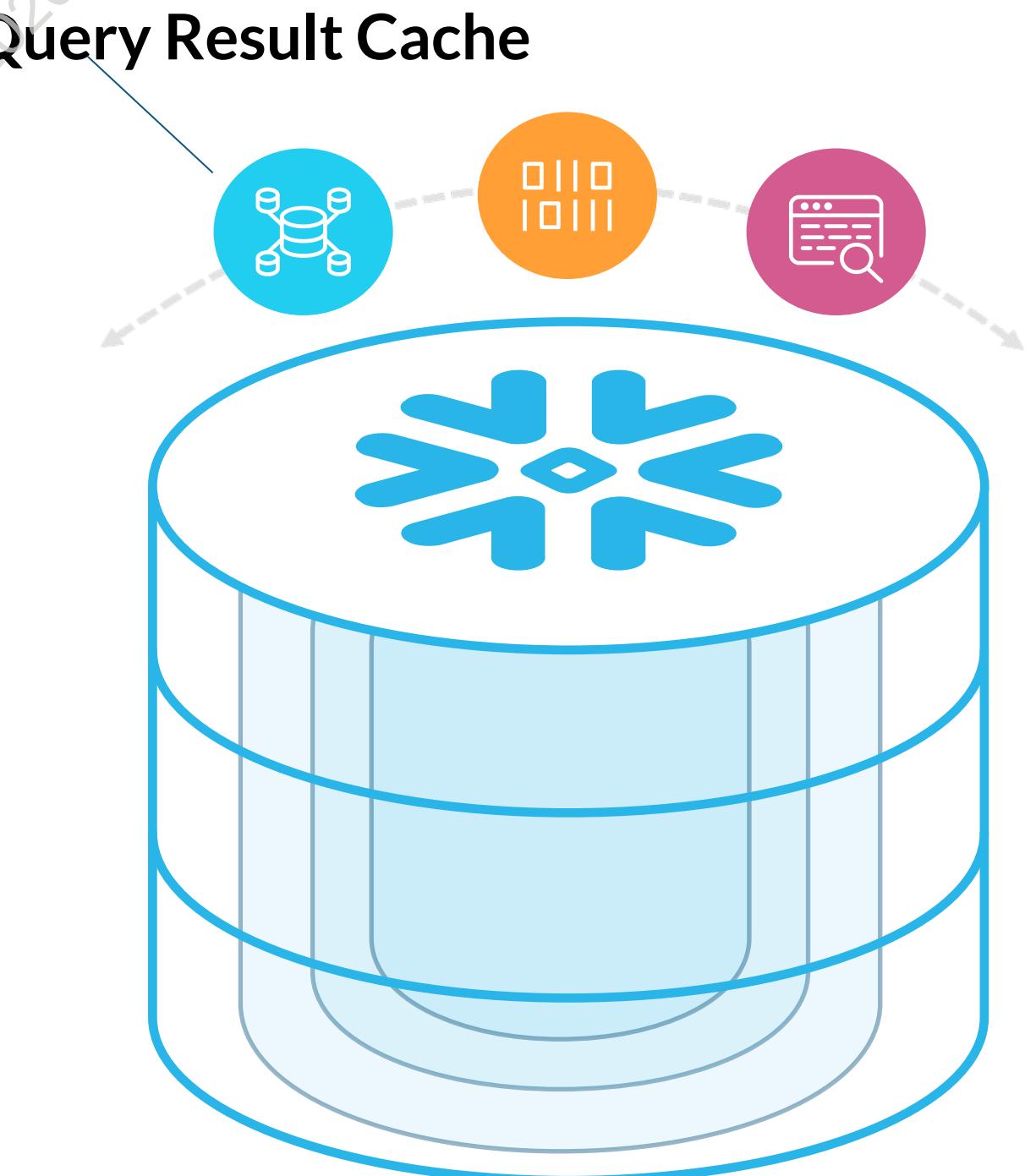
# QUERY RESULT CACHE EXAMPLE

Query profile confirms re-use of query result cache



# HOW IT WORKS

- **Result sets are stored**
- **Result sets are cached** for 24 hours; counter resets each time matching query is re-used
- **Result reuse** can be controlled by `USE_CACHED_RESULT` parameter at account/user/session level
- **Eligibility requirements** for query to use result set cache:
  - Exact same SQL query (\* except maybe whitespace)
  - Result must be deterministic (eg. no random function)
  - Changes CAN be made to source table(s), but only if NONE affect any micro-partitions relevant to query

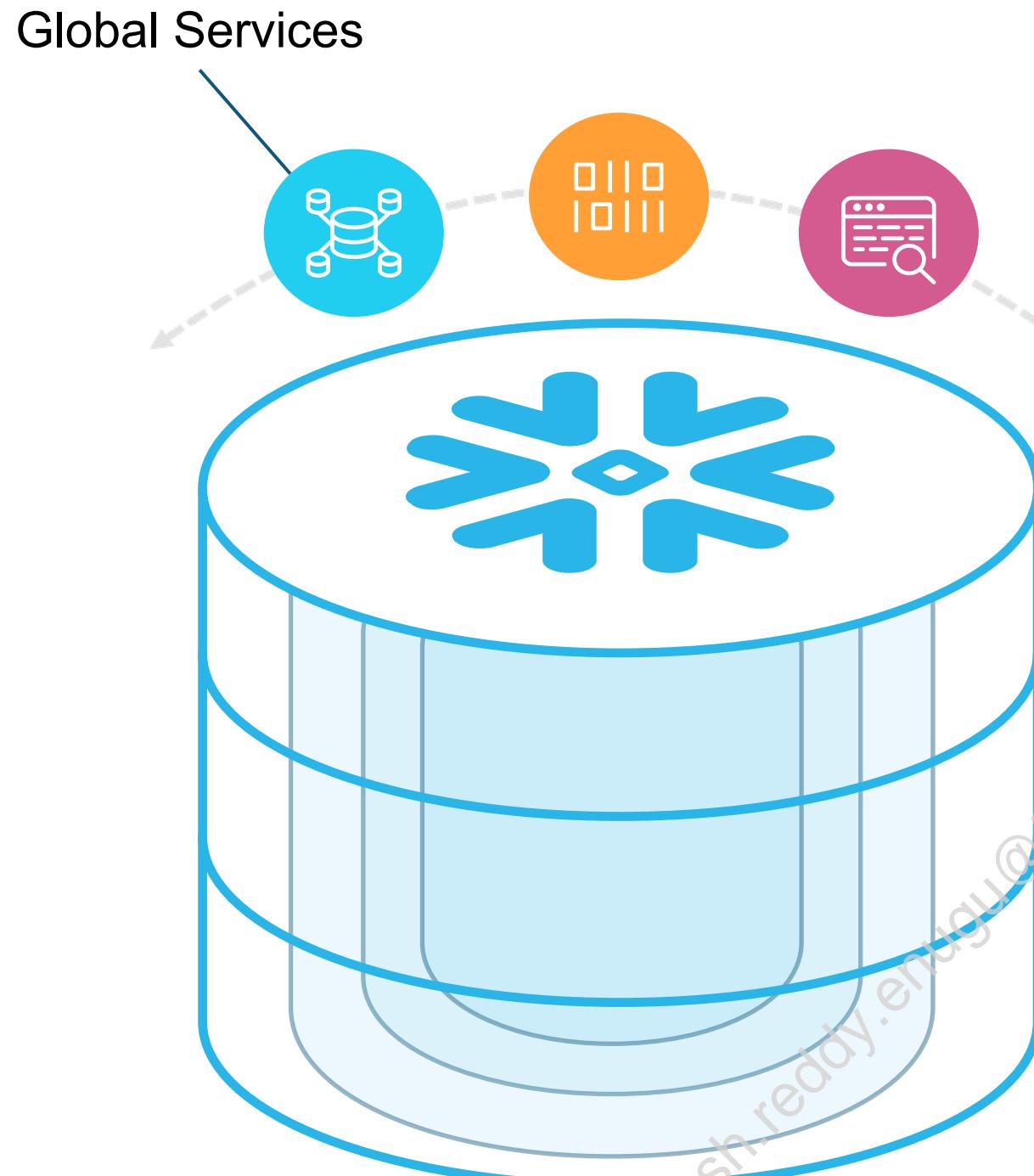


# Metadata Cache

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



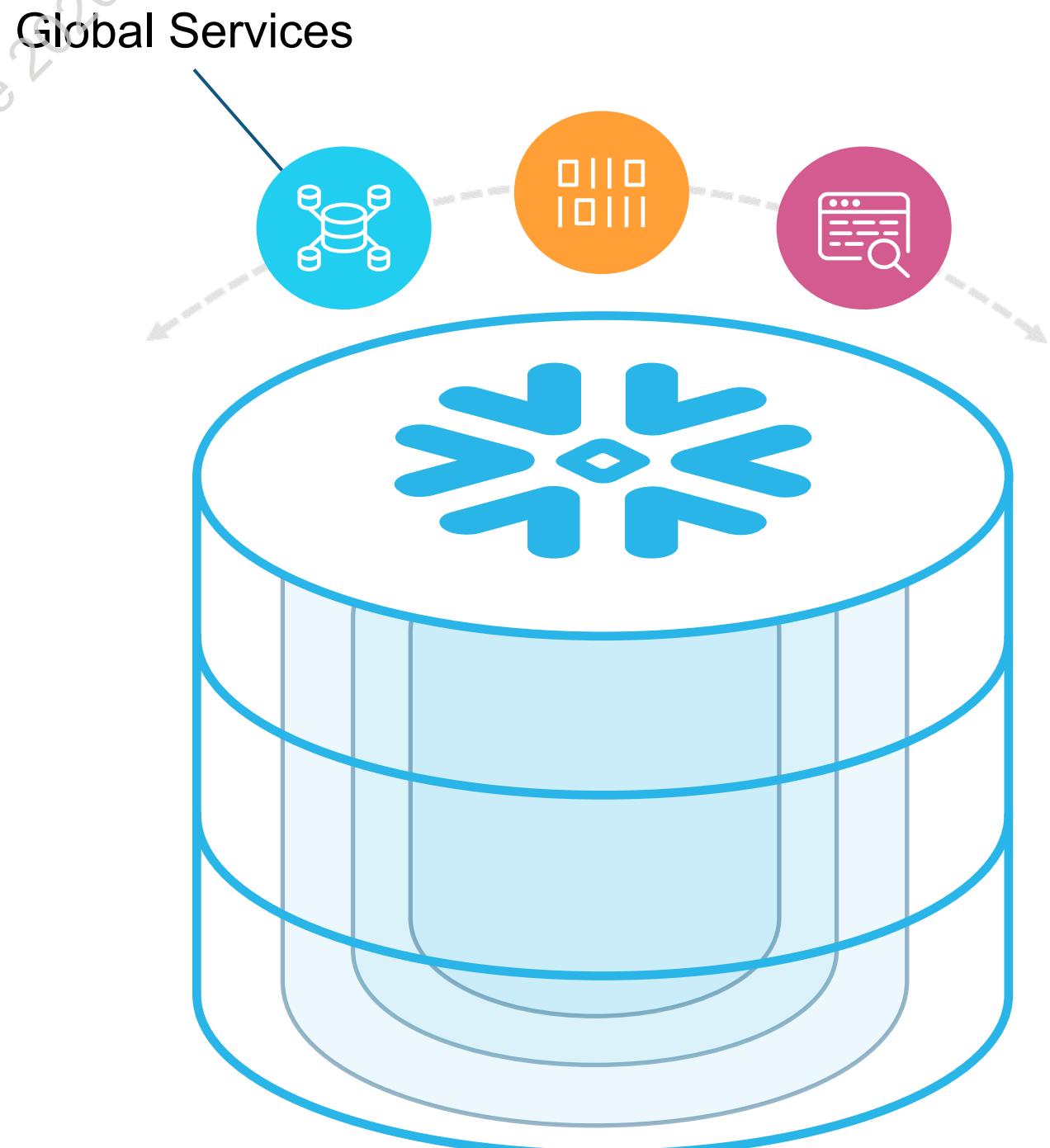
# METADATA CACHE



- Metadata stored in the cloud services layer
- Partition-level metadata
  - Row count
- Partition-column-level metadata
  - MIN/MAX values
  - Number of DISTINCT values
  - Number of NULL values
- Table versions and references to physical files (.fdn)

# MICRO-PARTITION METADATA CACHE

- Used automatically by SQL optimizer to speed up query compilation
- Used for queries that can be answered completely by metadata
- Used for SHOW and other commands
- Fast
- Free compute (no virtual warehouse used)



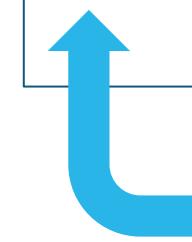
# METADATA CACHE USE EXAMPLES

Queries answered by metadata and cache - NO compute needed!

- SHOW commands
- Queries on INFORMATION SCHEMA
- Aggregate queries such as COUNT, MIN, MAX – milliseconds response time!

```
USE DATABASE
SNOWFLAKE_SAMPLE_DATA;
USE SCHEMA TPCH_SF100;

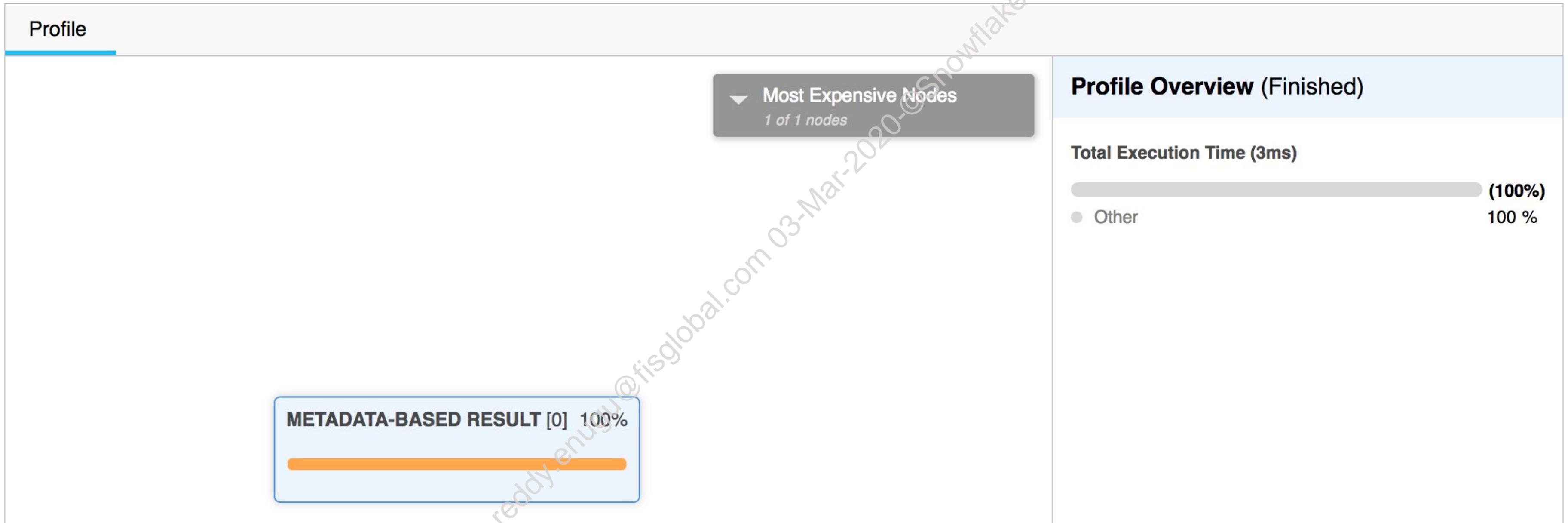
SELECT
 MIN(L_SHIPDATE),
 MAX(L_SHIPDATE)
FROM LINEITEM;
```



**Snowflake's metadata allows some queries to be serviced as metadata only operations. No compute required!**



# METADATA CACHE - QUERY PROFILE

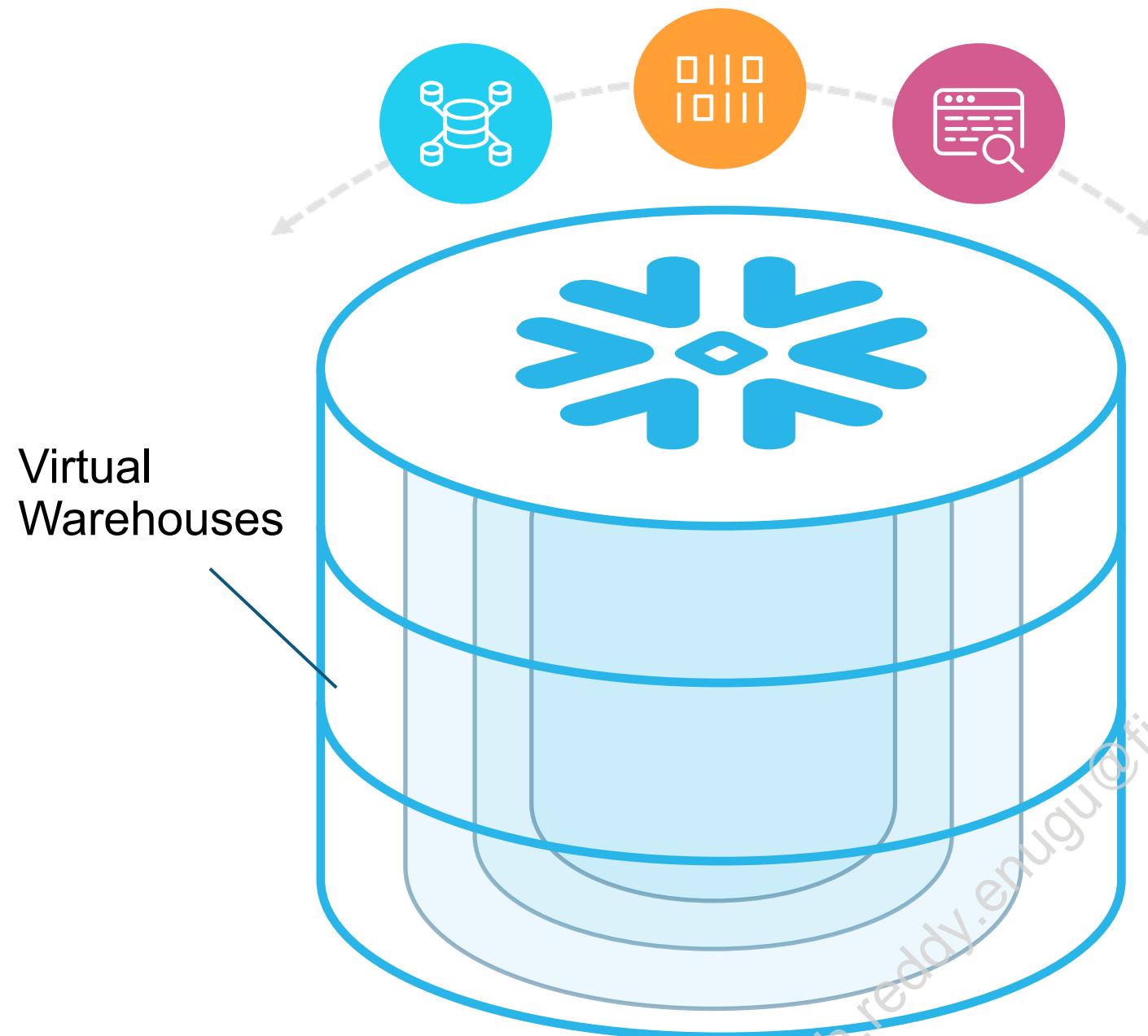


# Data Cache in Compute Cluster

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# DATA CACHE



- Stores file headers and column data from queries
  - Stores the data, not the result
- Stored to SSD in warehouse
- When a similar query is run, Snowflake will use as much data from the SSD cache as possible
- Available for all queries run on the same virtual warehouse

# DATA CACHE

## Query Example:

```
SELECT network FROM rating WHERE (data_stream = 'Live');
```

## What is in the Query Data Cache?

- Only the selected column: **network**
- Only the partitions satisfying predicates: **data\_stream = 'Live'**

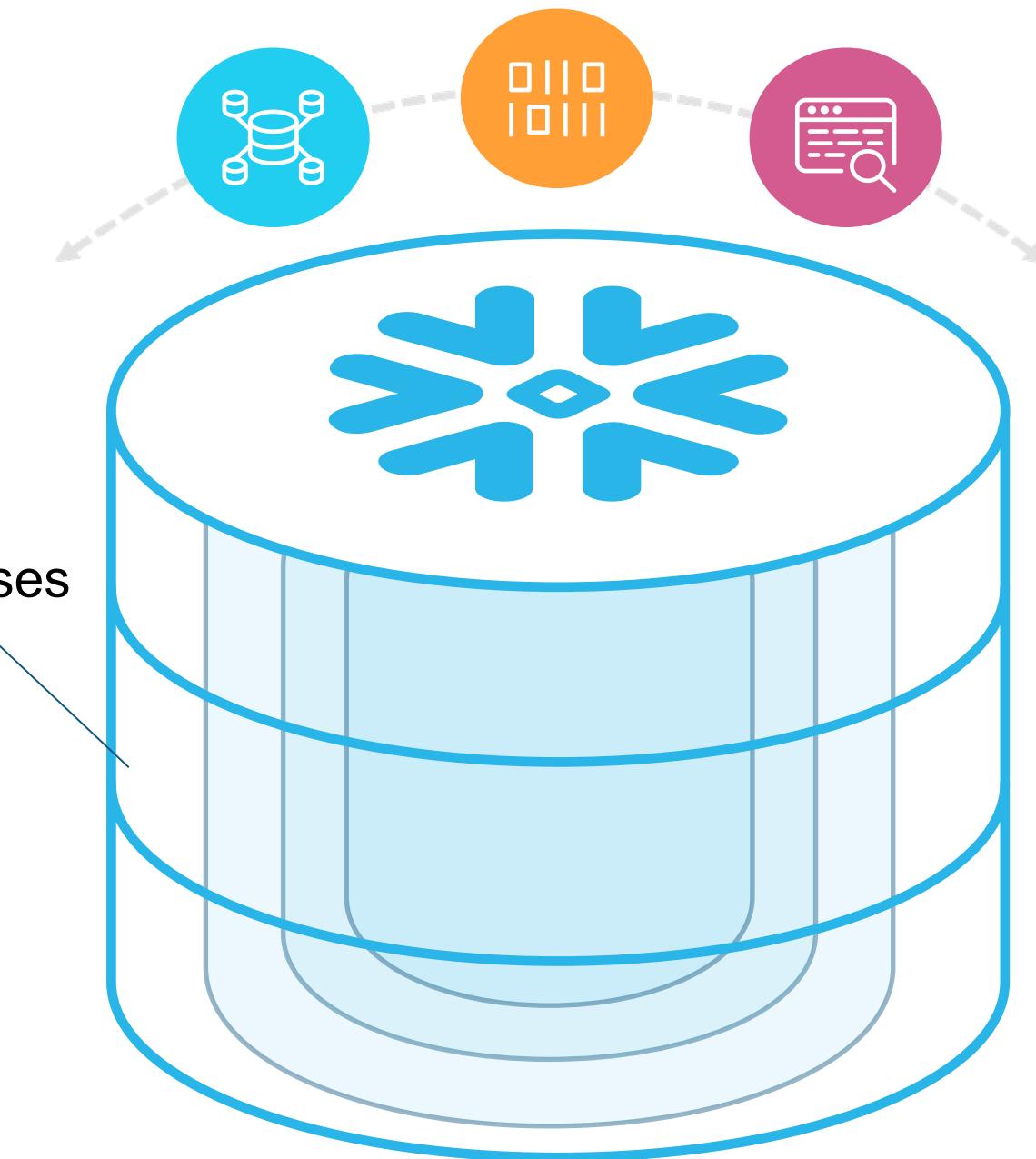
## Best Practice:

Group and execute similar queries on the same virtual warehouse to maximize query data cache reuse, for performance and cost optimization

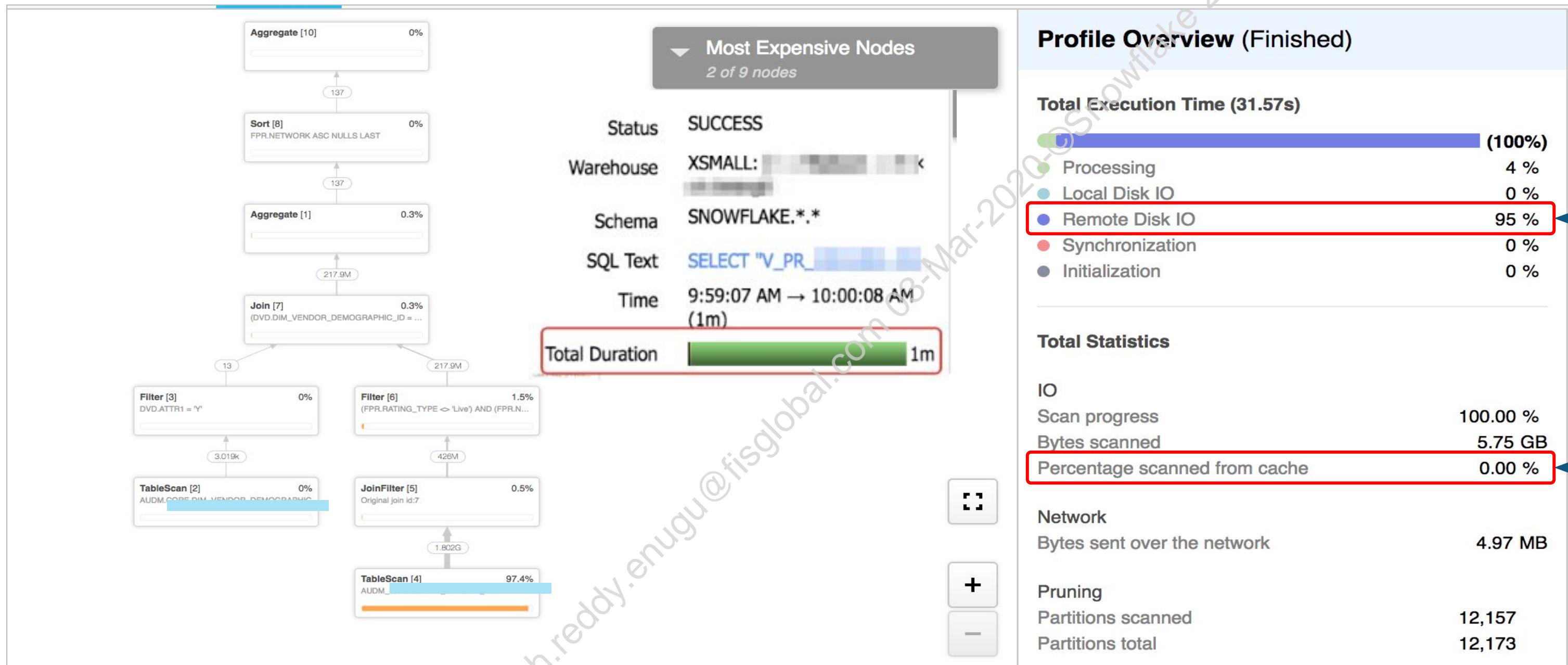


# HOW DATA CACHE WORKS

- When a query is run, file headers and column data retrieved are stored on SSD
- Warehouse will first read any locally available data, then read remainder from remote cloud storage
- Data is flushed out in a LRU (Least Recently Used) fashion, when cache fills



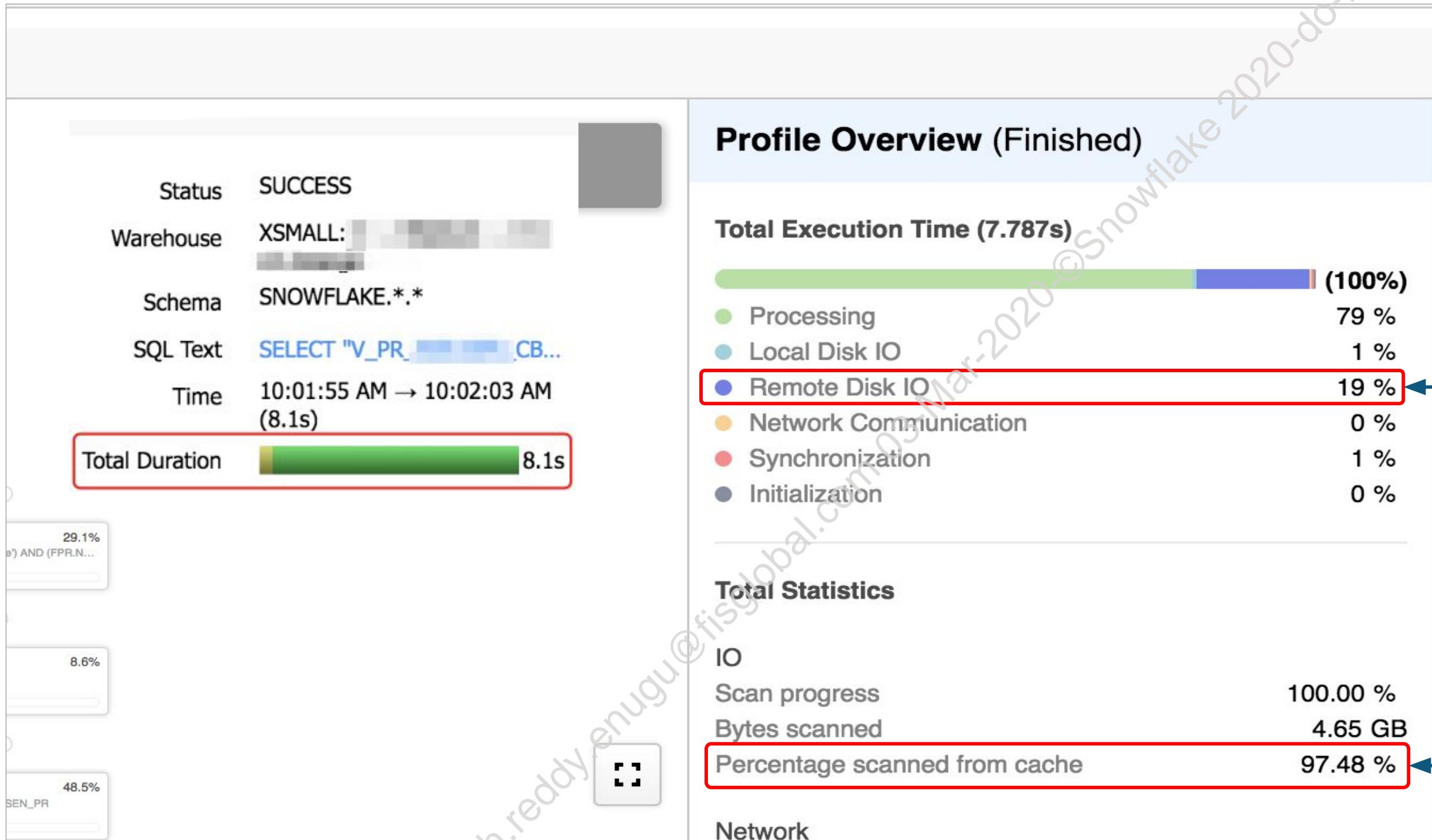
# DATA CACHE: COLD EXAMPLE



95 % of the query cost was spent on remote IO

0% of data was scanned from cache.

# DATA CACHE: WARM EXAMPLE



# SUMMARY OF CACHE OPTIONS

|                        | Metadata Cache                                                | Query Result Cache                                                  | Data Cache                                                                             |
|------------------------|---------------------------------------------------------------|---------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| Where is it stored?    | Cloud services layer                                          | Cloud services layer                                                | Warehouse SSD                                                                          |
| What does it store?    | Metadata and statistics for micro-partitions and tables       | Results set for each query                                          | Data used in query                                                                     |
| When is it used?       | For commands like MIN, MAX, COUNT<br><br>Does not use compute | Identical query is run again<br><br>Base table data has not changed | Query using some or all of the same data is run<br><br>Base table data has not changed |
| How long does it last? | Continuously updated                                          | 24 hours<br><br>Clock reset after every run                         | While warehouse exists<br><br>Rotated out on a LRU basis                               |



# Module 9: Snowflake Caching

## Exercise 9.1: Explore Snowflake Caching

30 minutes

### Tasks:

- Explore Metadata cache
- Explore Data cache
- Explore Query result cache



# Module 10

# Continuous Data Protection

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Time Travel
- Fail Safe
- Cloning

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# Time Travel

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# PROBLEM

## RECOVERING FROM MISTAKES



Tables are dropped

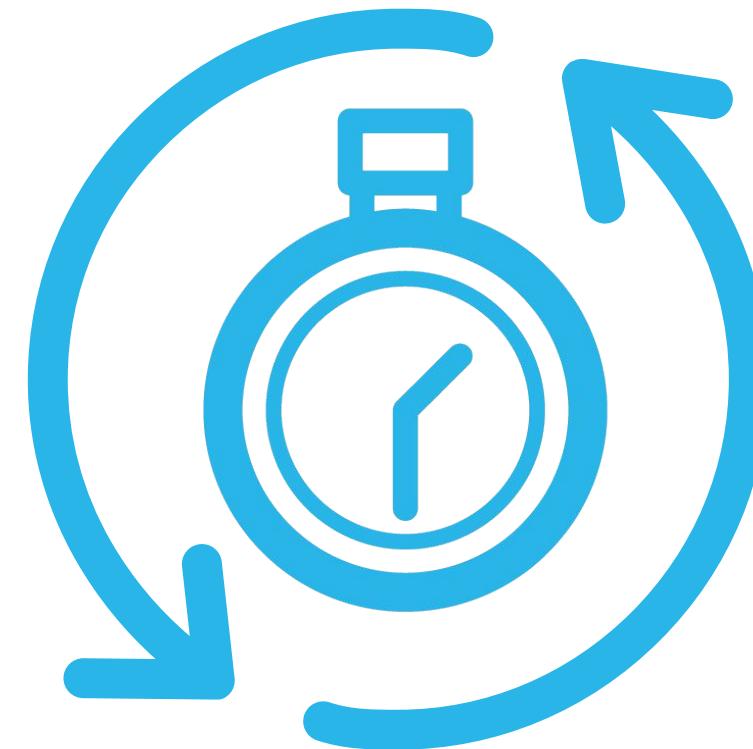
Rows are deleted

Schemas are edited

- User errors
- System errors
- Backup itself is time-consuming task
- Specialized skill and management overhead

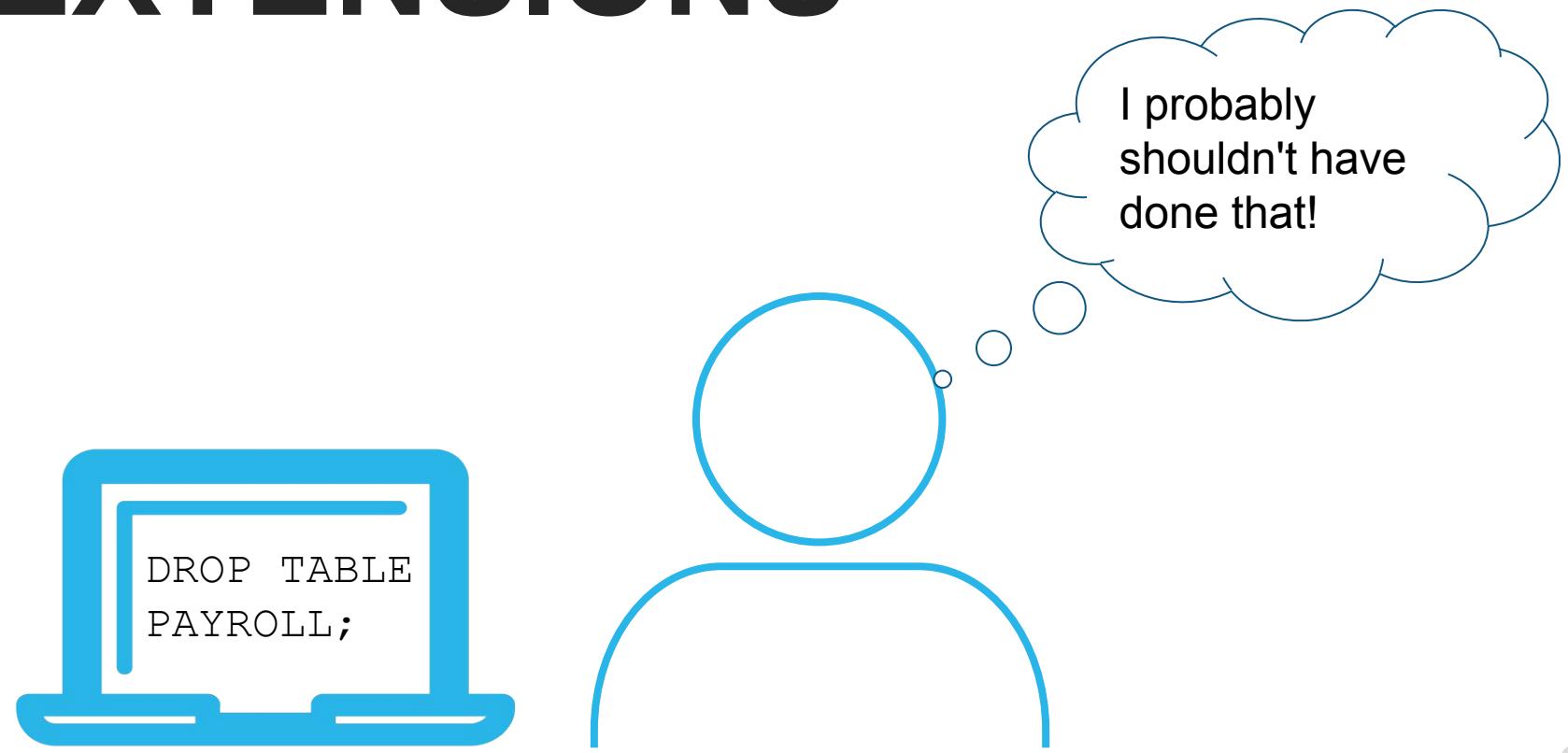


# SOLUTION: TIME TRAVEL



- Access historical data at any point within a defined retention period
- UNDO common mistakes
  - Protect against accidental or intentional modification, removal, or corruption
    - Fix drops, deletes, edits
- Backup/Restore from time or ID

# TIME TRAVEL SQL EXTENSIONS



**UNDROP** <object>

**CREATE** <object> **CLONE**... **AT | BEFORE**

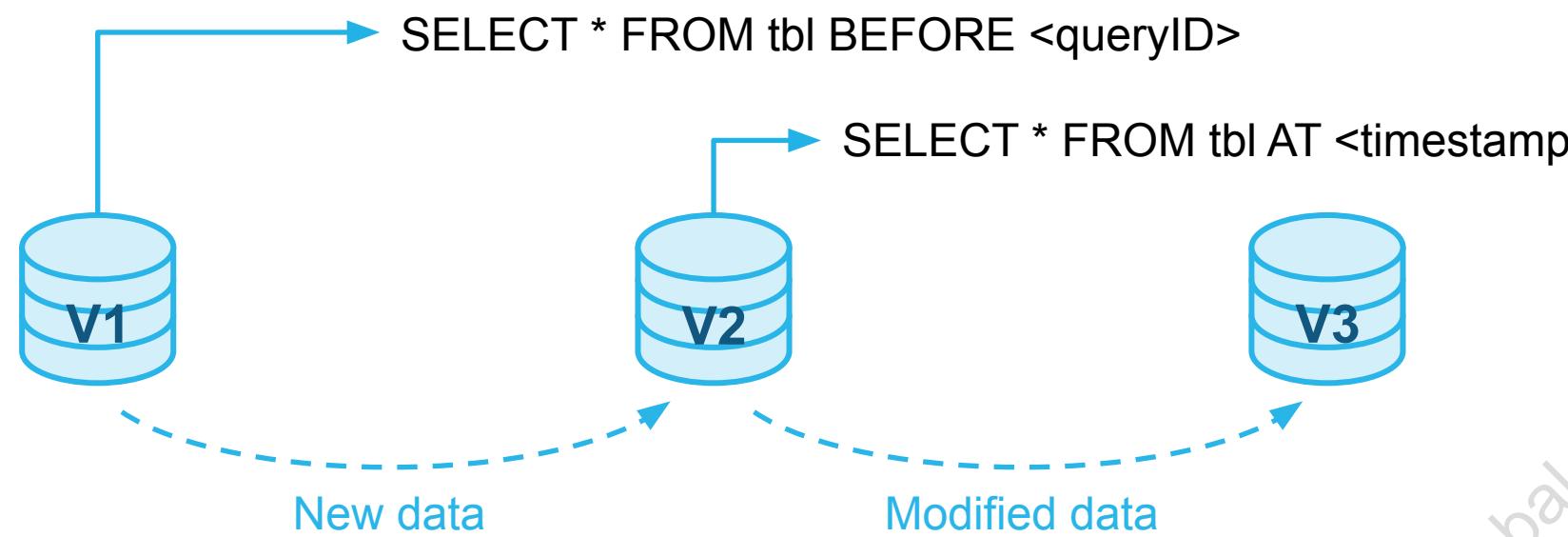
**SELECT**... **FROM** <table> **AT | BEFORE**

**DATA\_RETENTION\_TIME\_IN\_DAYS**

- Instantly bring back deleted tables, schemas, and databases
- Restore or duplicate data from key points in the past:
  - Point-in-time
  - Prior to a specific query ID
- Set retention times at the table, schema, database, or account level



# TIME TRAVEL



- Available for databases, schemas, and tables
- Configuration retention option:
  - `DATA_RETENTION_TIME_IN_DAYS`  
Disable by setting retention to 0
- SQL extensions:
  - `AT | BEFORE` - querying clause
  - `UNDROP` - recovery



# CREATE TABLE WITH TIME TRAVEL

- Automatic with default retention period:

```
CREATE TABLE my_table (c1 int);
```

- Customizable retention period:

```
CREATE TABLE my_table (c1 int)
SET DATA_RETENTION_TIME_IN_DAYS=90;
```

```
ALTER TABLE my_table
SET DATA_RETENTION_TIME_IN_DAYS=30;
```



# QUERYING WITH TIME TRAVEL

## Query Clauses to support Time Travel Actions

- AT or BEFORE

```
SELECT * FROM my_table1
AT (TIMESTAMP => 'Mon, 01 May 2015 16:20:00 -0700' ::timestamp);
```

```
SELECT * FROM my_table1
BEFORE (STATEMENT => '8e5d0ca9-005e-44e6-b858-a8f5b37c5726');
```



# DML EXAMPLES

- Cloning Historical Objects

```
CREATE TABLE restored_table CLONE my_table1
 AT (TIMESTAMP => 'Mon, 09 May 2015 01:01:00 +0300' ::timestamp);
```

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

- Restoring Objects

```
UNDROP TABLE/SCHEMA/DATABASE
```

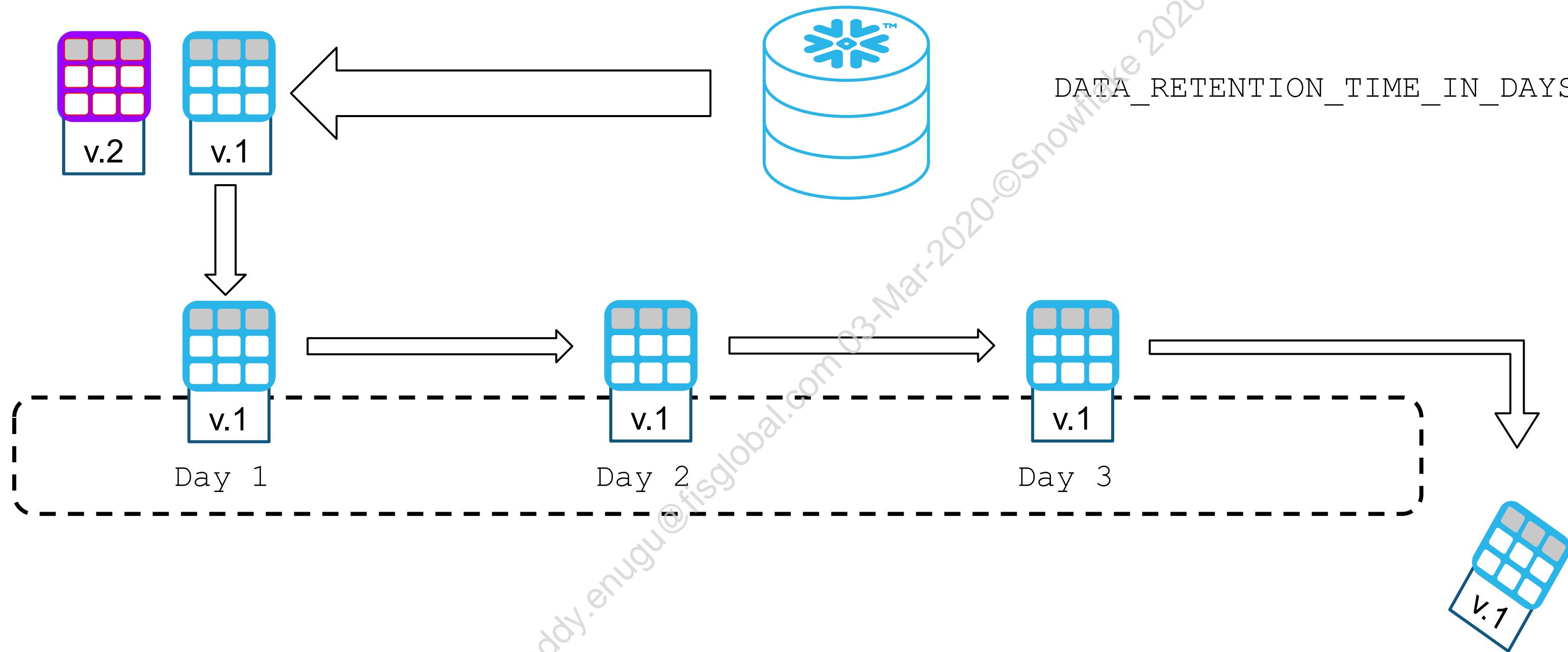


# HOW DOES IT WORK?

- Micro-partitions!
- Micro-partitions are immutable
- When data is changed, new versions of the micro-partitions are created
- We keep the older version for the specified retention time



# TIME TRAVEL OVERVIEW

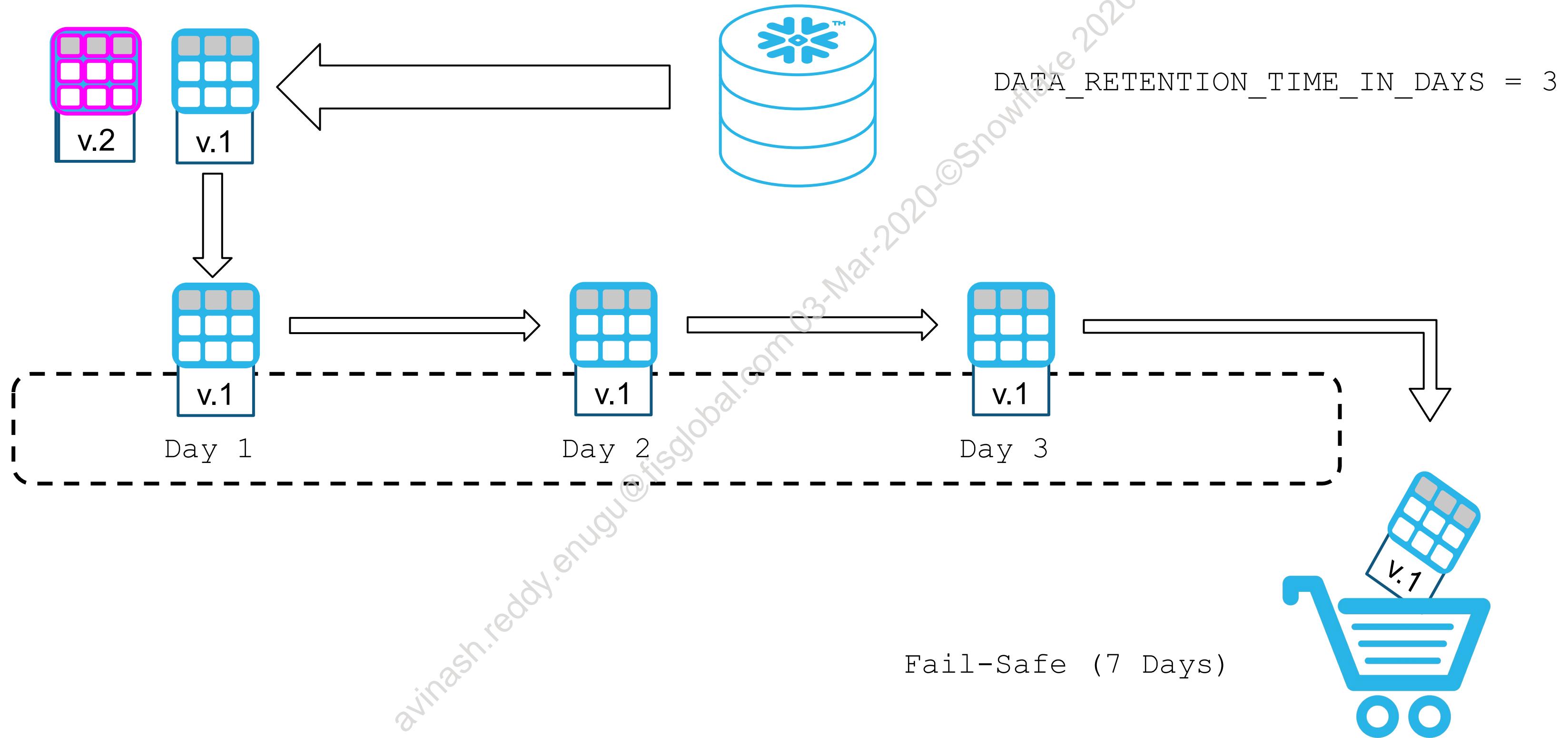


# **Fail-Safe**

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# FAIL-SAFE OVERVIEW



# FAIL-SAFE STORAGE

- Non-configurable, 7-day retention for historical data after Time Travel expiration
- Only accessible by Snowflake personnel
- Admins can view fail-safe use in the Snowflake Web UI under **Account > Billing & Usage**
- Not supported for temporary or transient tables



# DATA PROTECTION OPTIONS BY EDITION

| Snowflake Edition | Time Travel<br>(1 day) | Time Travel<br>(90 days max) | Fail-safe<br>(7 days) |
|-------------------|------------------------|------------------------------|-----------------------|
| Standard          | ✓                      |                              | ✓                     |
| Premium           | ✓                      |                              | ✓                     |
| Enterprise        |                        | ✓                            | ✓                     |
| Business Critical |                        | ✓                            | ✓                     |
| VPS               |                        | ✓                            | ✓                     |



# Module 10: Continuous Data Protection

## Exercise 10.1: Exploring Time Travel

20 minutes

### Tasks:

- UNDROP a table
- Recover data from a specific point in time
- Time travel with object naming constraints

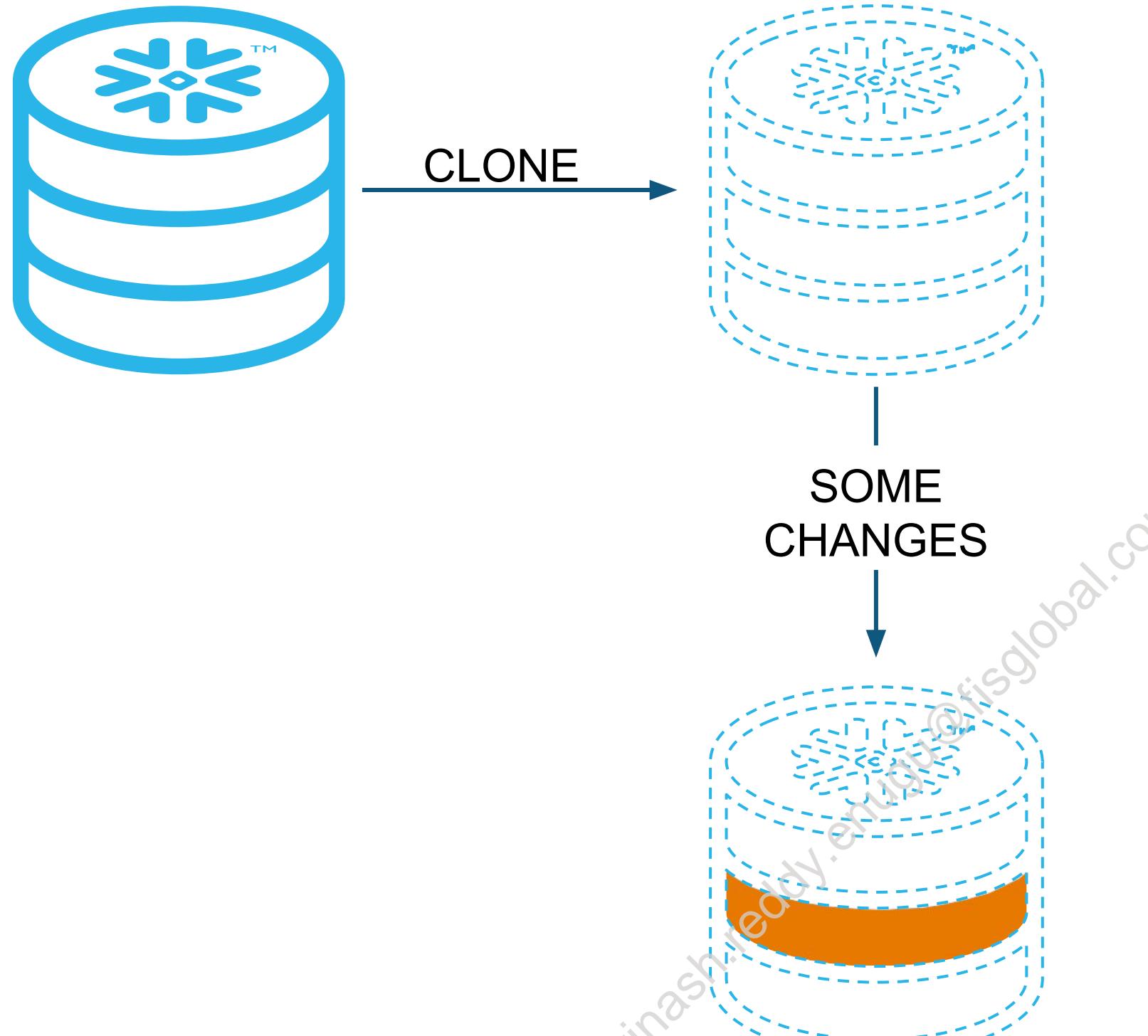


# Cloning

avinash.reddy.enugu@fisglobal.com 03-May-2020 ©Snowflake 2020-do-not-copy



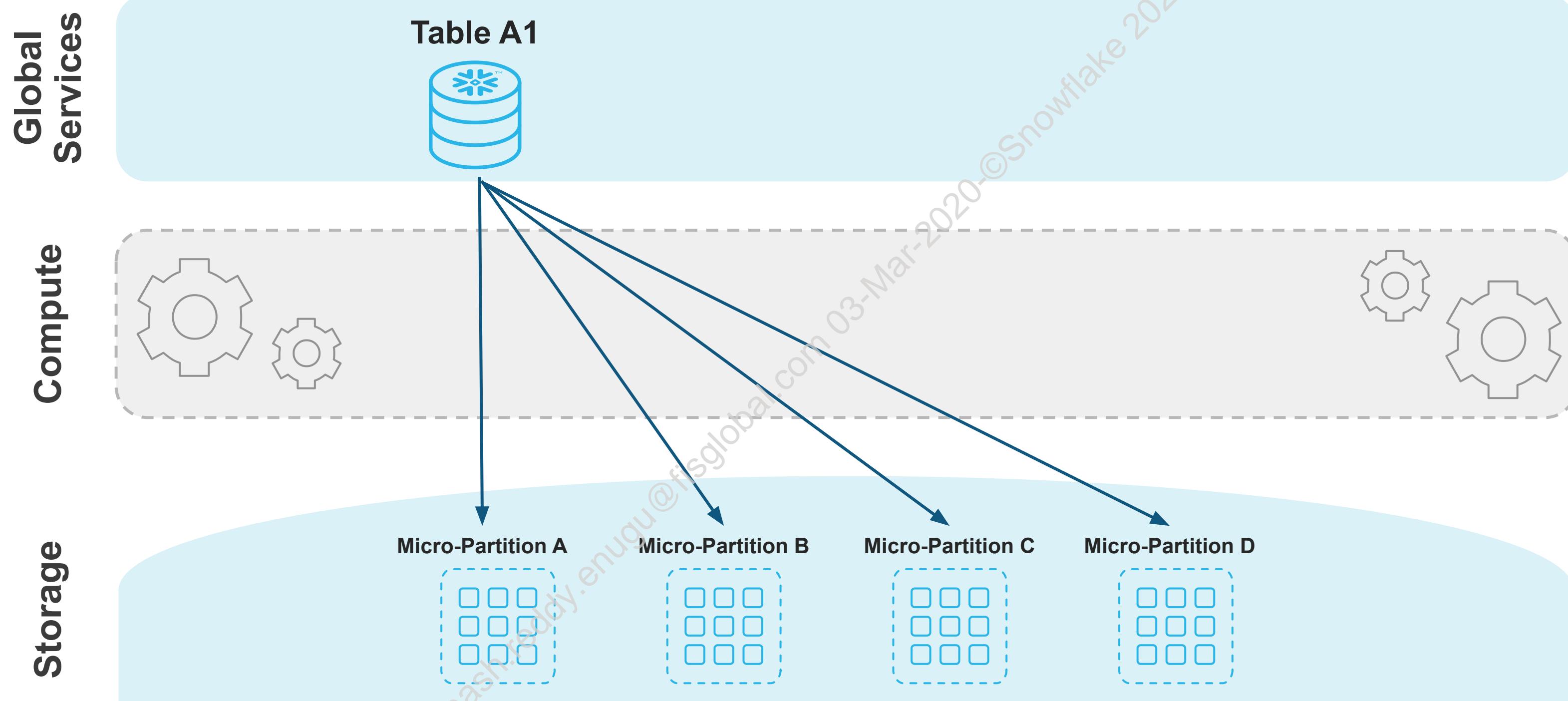
# ZERO-COPY CLONING



- Quickly take a "snapshot" of any table, schema, database (clones can be cloned)
- When the clone is created:
  - All micro-partitions in both tables are fully shared
  - Micro-partition storage is owned by the oldest table, clone references them
- No additional storage costs until changes are made to the original or the clone
- Often used to quickly spin up Dev or Test environments,
- Effective “backup” option as well

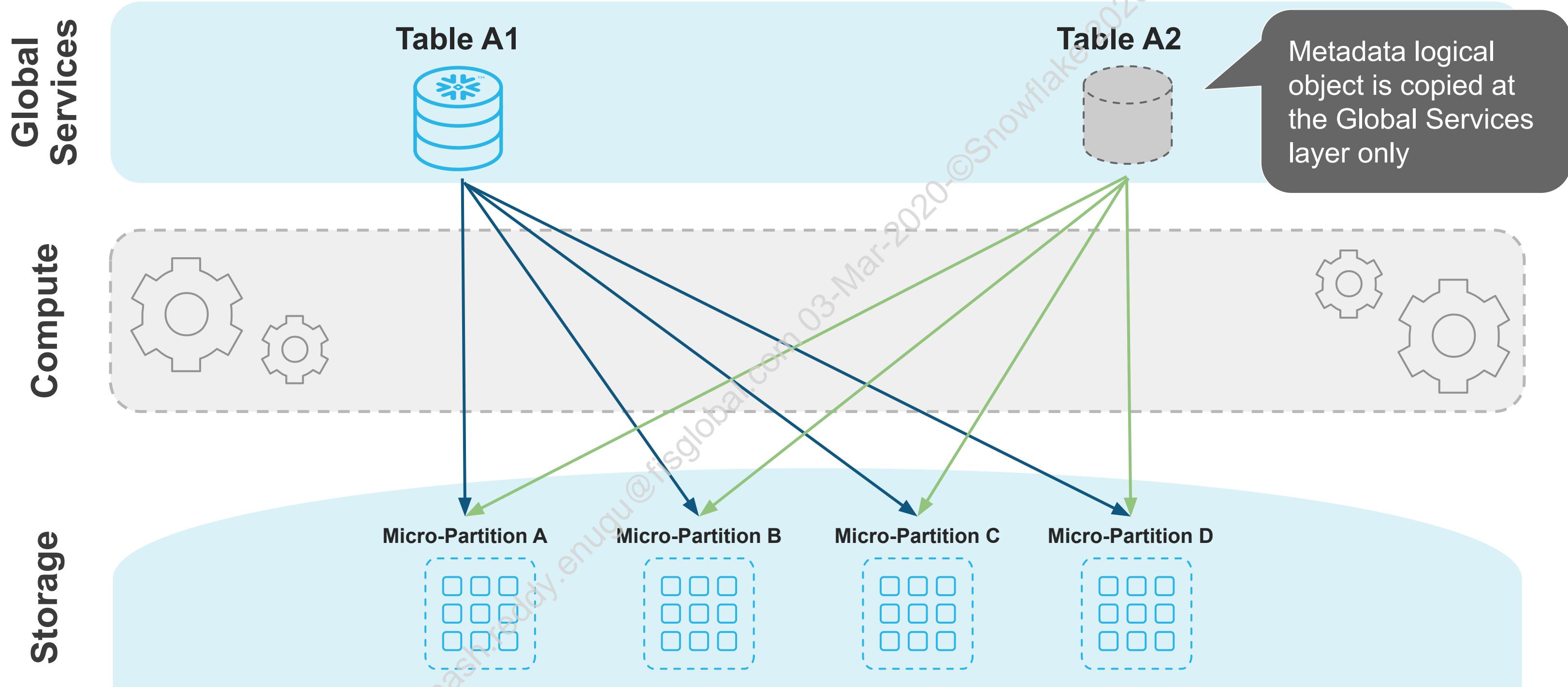
# CLONING EXAMPLE

## BEFORE CLONE



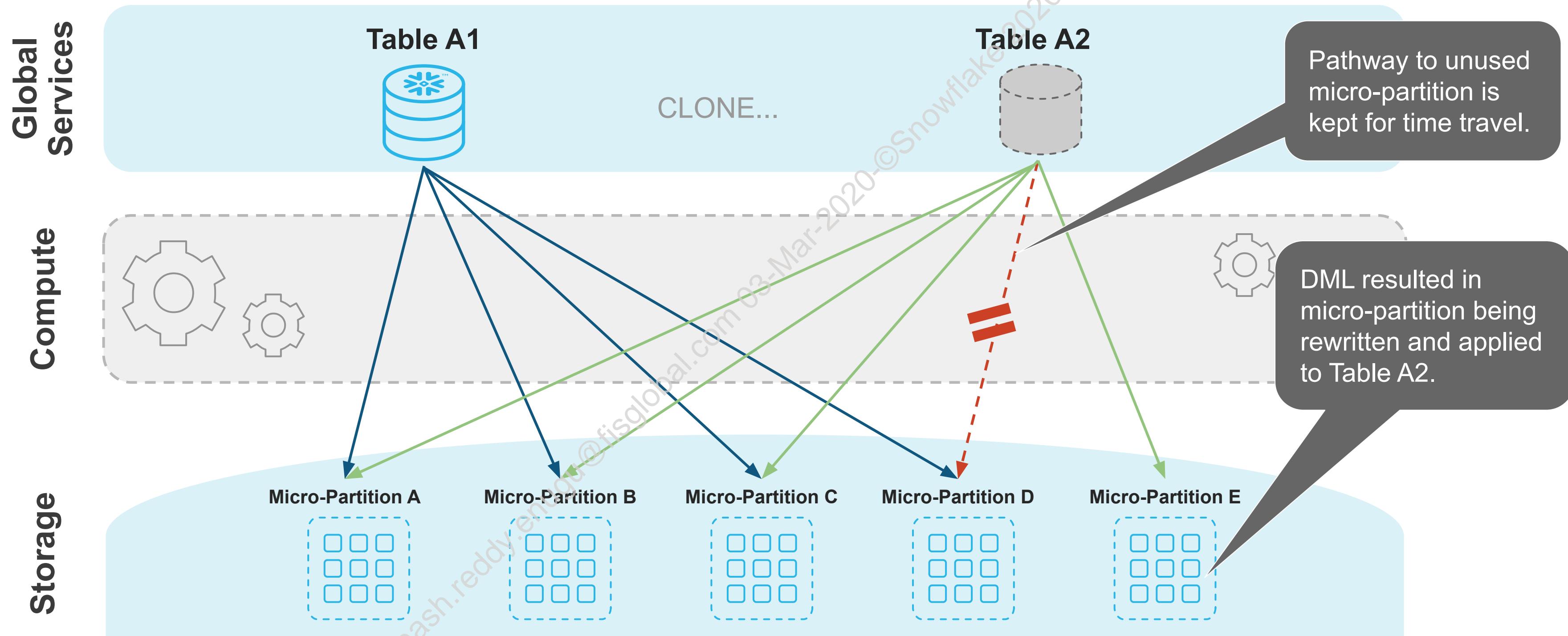
# CLONING EXAMPLE

TABLE A1 IS CLONED TO TABLE A2



# CLONING EXAMPLE

## POST-DML



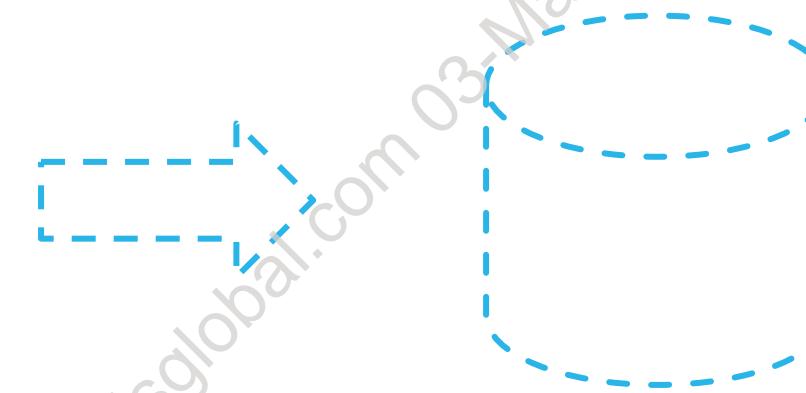
# HOW DOES IT WORK?

Clone the database

```
CREATE DATABASE test_db
CLONE prod_db;
```



PROD\_DB



TEST\_DB

# Module 10: Continuous Data Protection

## Exercise 10.2: Cloning Tables

10 minutes

### Tasks:

- Clone database objects



## Module 11

# Data Sharing

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Data Sharing Overview
- Sharing Data

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

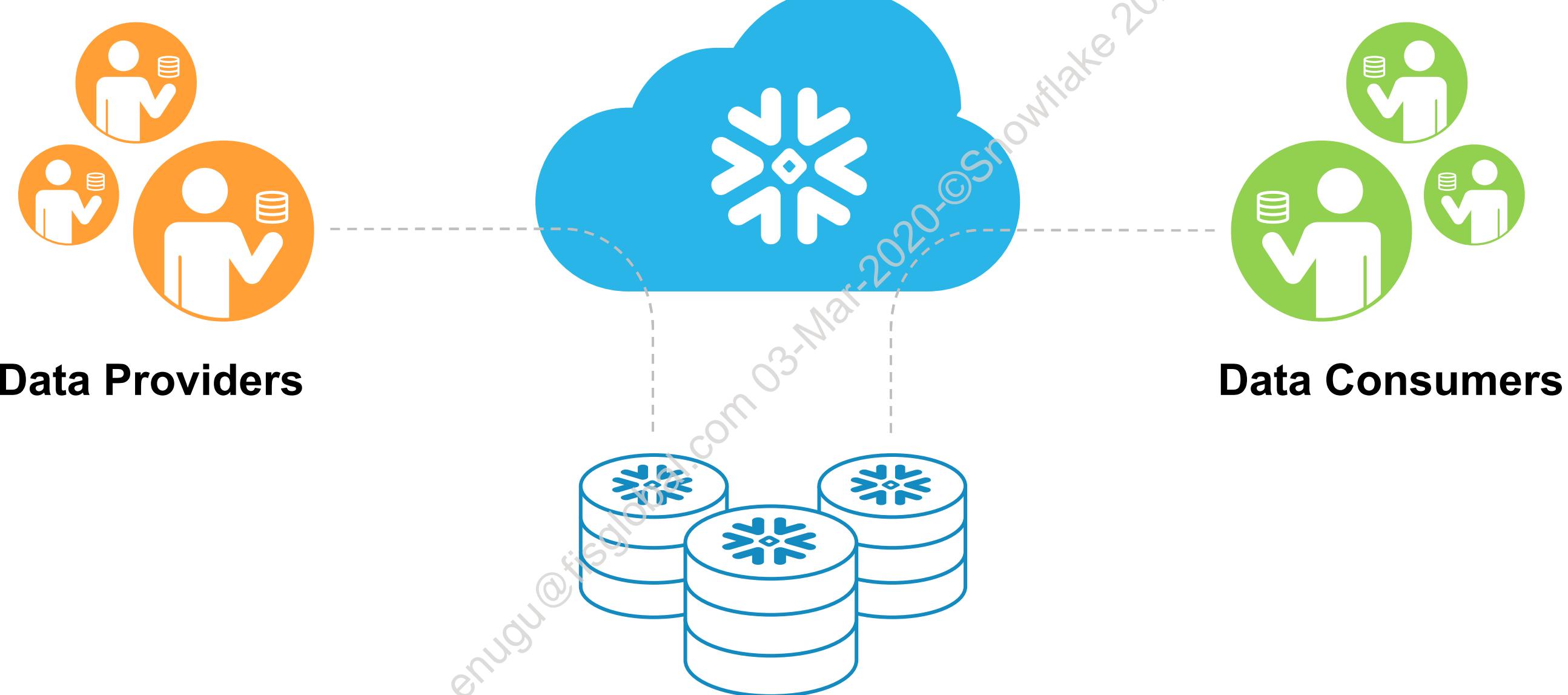


# Data Sharing Overview

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# A BETTER WAY TO SHARE DATA



## No Data Movement

Share with unlimited number of consumers, without duplicating storage.

## Live Access

Data consumers immediately see all updates

## Ready to Use

Consumers can immediately start querying



# DATA SHARING ACCOUNTS



**Data Providers**

Share data with others



**Data Consumers**

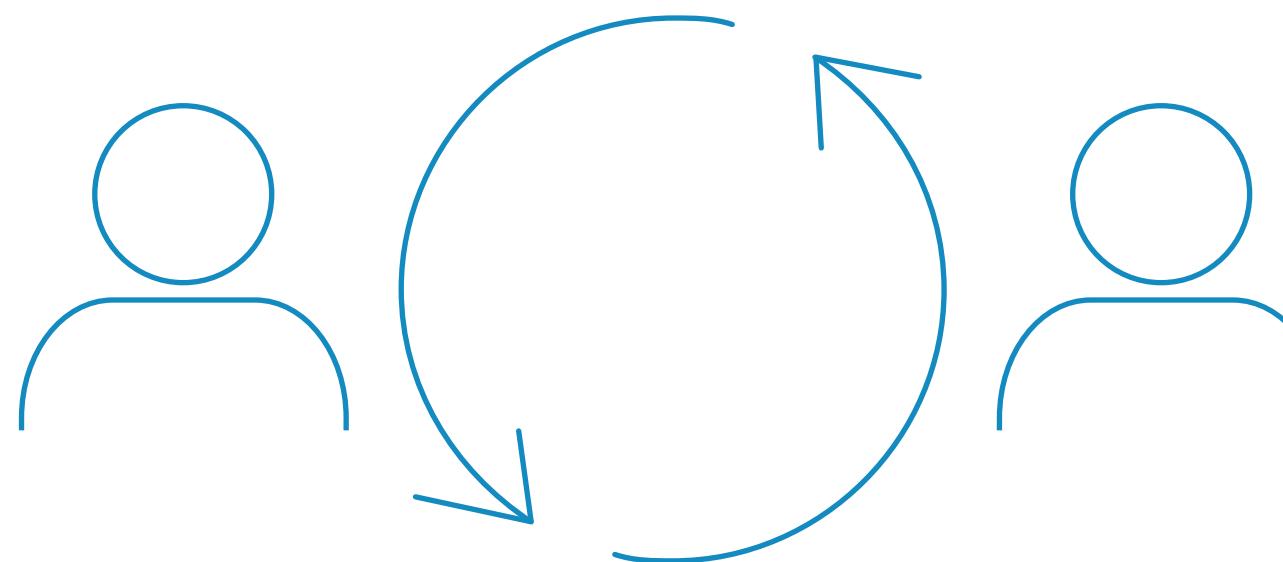
Accesses shared data with  
their own Snowflake  
account



**Reader Accounts**

Query data using  
compute from data  
provider's account

# INSTANT, LIVE DATA SHARING



- Share with an unlimited number of consumers, without duplicating storage
- Data consumers immediately see all updates
- Consumers can immediately start querying
- Reader accounts enable sharing with non-Snowflake customer



# DATA PROVIDERS



**Data Providers**

- Snowflake accounts that creates shares and makes them available for others to consume
- Unlimited number of shares can be created and an unlimited number of accounts can be added to a share
- Grants provide granular access control to selected objects, including at the row level (using filters)



# DATA CONSUMERS



**Data Consumers**

- Snowflake account that accesses a share from another account
- Can make a copy of the shared data in their account
  - One copy per share
- Can consume an unlimited number of shares
- Are charged for their own compute on that share

# READER ACCOUNTS



## Reader Accounts

- Consumer who does not already have a Snowflake account
- Accounts are created by Providers
- Compute credits funded by the provider
- Consumer cannot write data to the account
- Allows consumer to experiment with data sharing without a Snowflake contract

# Module 11: Data Sharing

**DEMO: Data Sharing**

**15 minutes**

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy





Data Provider  
(ProvXyz)

# CREATE A SHARE

```
USE ROLE ACCOUNTADMIN;

CREATE SHARE share1; --empty share

GRANT USAGE ON DATABASE sales TO SHARE share1; -- add database
GRANT USAGE ON SCHEMA sales.east TO SHARE share1; -- add schema
GRANT SELECT ON TABLE east.accts TO SHARE share1; -- add table

ALTER SHARE share1 ADD ACCOUNTS=Cons123; -- add account
```





Data Consumer  
(Consz123)

# CONSUME A SHARE

```
USE ROLE ACCOUNTADMIN;
```

```
CREATE DATABASE ProvXyz_accts
FROM SHARE ProvXyz.share1; --read-only shared database
```

```
USE DATABASE ProvXyz_accts; --switch to read-only database
```

```
SELECT * FROM east.accts; --same as querying any other
 database in your account
```



# SHARING SECURE VIEWS

WITH ACCESS CONTROLS



Data Provider

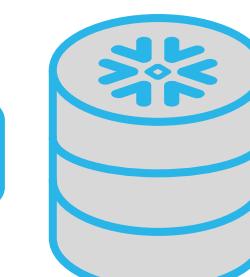
| partner_data |       |            |
|--------------|-------|------------|
| id           | sales | commission |
| 100          | 2,350 | 11.75      |
| 100          | 1,975 | 49.375     |
| 200          | 3,459 | 8.6475     |
| 200          | 9,156 | 68.67      |

| acct_map |           |
|----------|-----------|
| id       | acct_name |
| 100      | ABC       |
| 200      | XYZ       |

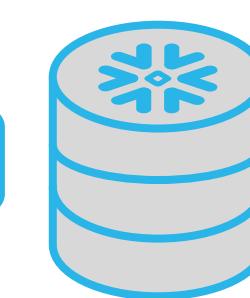


Data Consumers

| shared_data |       |            |
|-------------|-------|------------|
| id          | sales | commission |
| 100         | 2,350 | 11.75      |
| 100         | 1,975 | 49.375     |



Account = ABC



Account = XYZ



```

CREATE SECURE VIEW shared_data
AS SELECT sd.*
 FROM partner data pd
 JOIN acct_map am ON pd.id = am.id
 AND am.acct_name = CURRENT_ACCOUNT();
GRANT SELECT
ON shared_data TO SHARE shr_sales;

```



# CONSIDERATIONS FOR PROVIDERS

- Create shares with a role that has CREATE SHARES privileges (such as the ACCOUNTADMIN role)
- Consumer accounts must be in the same region
- Large tables (>1TB) should have a clustering key defined
- Only Secure Views and Secure UDFs are supported in shares at this time
- New or modified data in a share are immediately available to all consumers
- You must grant usage on new objects created in a database in a share in order for them to be available to consumers



**Data Providers**



# CONSIDERATIONS FOR CONSUMERS

- Administer shares with a role that has IMPORT SHARES privileges (such as the ACCOUNTADMIN role)
- Share can only be consumed once per account
- Shared databases are read-only
- Shared data can be copied into a table in the consumer account (but cannot be cloned)
- You cannot forward (re-share) shared databases
- Time travel is not supported for shared databases



Data Consumers

# CONSIDERATIONS FOR READER ACCOUNTS

- Requires additional configuration after creating the share and the reader account
  - Create database from the share
  - Configure Users, Roles, Security for access to data in the reader account
  - Create Virtual Warehouses for Reader use (paid for by the Data Provider)
  - Set default role and VWH for logins
- Per the above...often the provider logs into the reader account and configures everything for the reader and then simply provides account url, login info
- Provider responsible for support for the reader accounts



Data Reader

# Data Sharing

## DEMO: Reader Accounts and Secure Views

20 minutes

- Create a share
- Create a reader account with secure views
- Demonstrate secure views with different accounts



# Module 12

# Performance and Concurrency

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Review of Micro-Partitions
- Data Clustering
- Other Performance Tips
- Virtual Warehouse Scaling

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

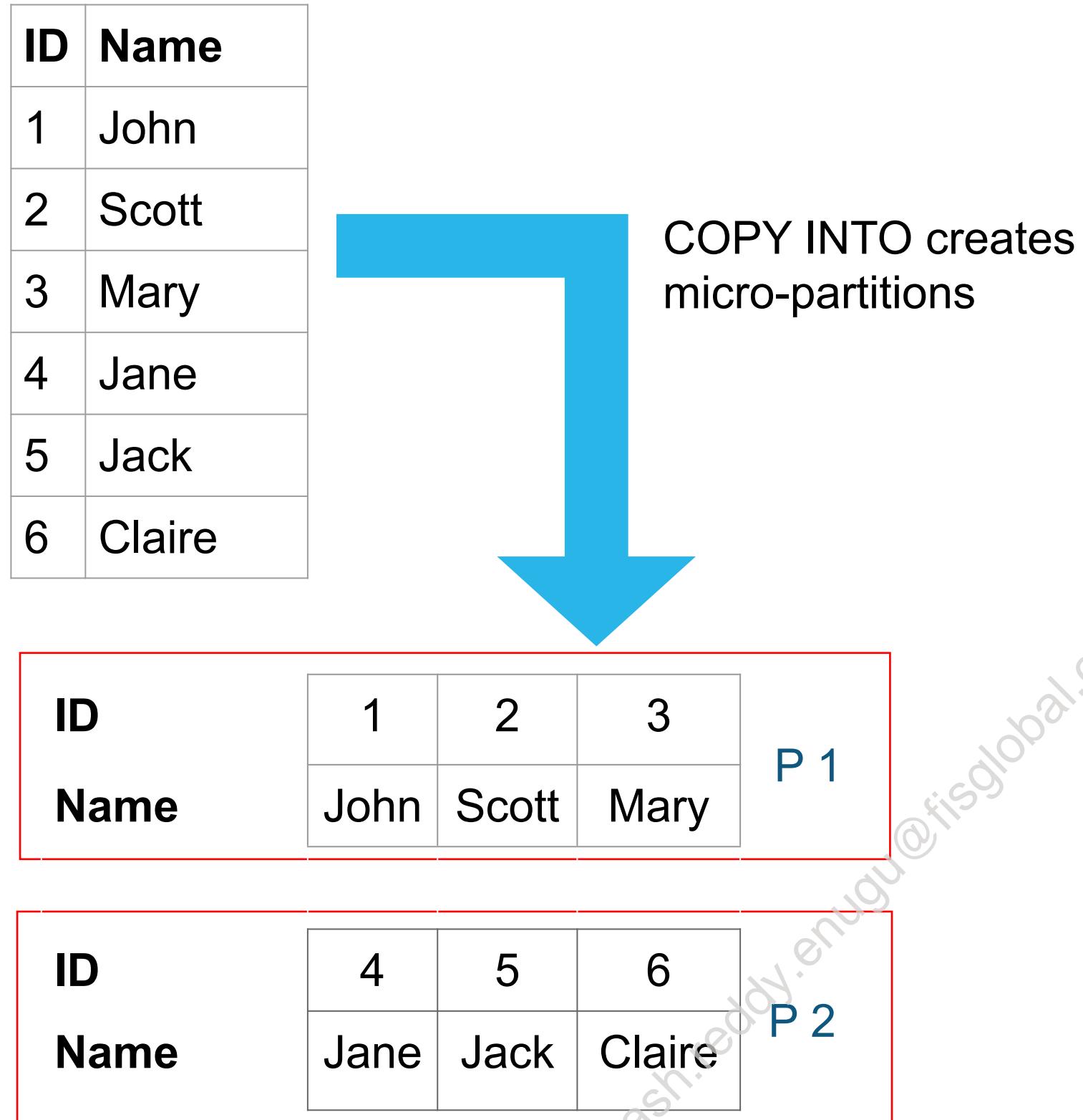


# Review of Micro-Partitions

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MICRO-PARTITIONING



- Physical data files that comprise Snowflake's logical tables
- Automatically-created, contiguous units of storage
- Partitioned based on ingestion order
- Immutable - updates create new Micro-partition versions



# PRUNING DURING A QUERY

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



# PRUNING EXAMPLE

| Statistics                                           |           |
|------------------------------------------------------|-----------|
| IO                                                   |           |
| Scan progress                                        | 100.00 %  |
| Bytes scanned                                        | 230.00 MB |
| Percentage scanned from cache                        | 0.00 %    |
| Network                                              |           |
| Bytes sent over the network                          | 2.86 GB   |
| Pruning                                              |           |
| Partitions scanned                                   | 1,840     |
| Partitions total                                     | 722,313   |
| Attributes                                           |           |
| Full table name                                      |           |
| SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.<br>STORE_SALES |           |
| Columns (4 / 104)                                    |           |



## TPCDS 100TB STORE\_SALES table

- 300 billion records
- Automatically partitioned  
Into 722,313 partition files
- Query reads only 1840  
Partition files from table

# Data Clustering

avinash.reddy.enugu@fisglobal.com 02-Mar-2020 ©Snowflake 2020-do-not-copy



# NATURAL DATA CLUSTERING

- As data is loaded into tables, micro-partitions are created based ingestion date
- Ingestion date may highly correlate with one or more columns
  - Tables with a sequential field
  - Tables with a date field

| ID | NICKNAME  | YOB  |
|----|-----------|------|
| 1  | PRISCILLA | 1962 |
| 2  | DAWG      | 2001 |
| 3  | JOKER     | 1997 |
| 4  | PRINCESS  | 1998 |
| 5  | SNEEZY    | 1983 |

| DATE                 | ORDER_NUMBER   | ITEM_TYPE        |
|----------------------|----------------|------------------|
| 22-Jul-2019 23:29:07 | 2019_072111356 | Office Supplies  |
| 22-Jul-2019 23:32:00 | 2019_072111357 | Office Furniture |
| 22-Jul-2019 23:44:56 | 2019_072111358 | Services         |
| 22-Jul-2019 23:59:01 | 2019_072111359 | Office Supplies  |
| 23-Jul-2019 00:01:07 | 2019_072111360 | Printers         |



# WHAT DETERMINES NATURAL CLUSTERING?

- A single data load reads source data and writes it into some number of micro-partitions
- The source data organization determines what range of values are represented in the micro-partitions
- Examples:
  - Source data contains rows from a specific day: that day's data will be in contiguous micro-partitions
  - Source data contains rows for a month or year: those micro-partitions will have high/low values for dates within that month or year
  - Source data is ordered alphabetically on a "last name" column: contiguous micro-partitions will be in alphabetical order



# CLUSTERED TABLES AND CLUSTERING KEYS

- All tables have a “natural” clustering based on ingestion time
- Natural clustering can be overridden by specifically designating clustering “keys”
  - Clustering keys can be columns or expressions
  - Use 1-3 keys maximum, in order of low to high cardinality. More keys is not “better”
- After clustering, like data (by key) is co-loaded in the same micro-partitions
- Snowflake keeps the clustering order updated in the background, billed on a per-second basis



# WHAT TABLES SHOULD BE CLUSTERED?

- **Clustering keys are not for every table!**
  - Automatic clustering consumes credits
  - Reclustering also increases storage costs
  - Can be less expensive to add cluster key after loading the table
  - Automatic Clustering can be disabled
- **Good candidates for clustering keys:**
  - **Tables in the multi-terabyte range** experience the most benefit from clustering (particularly if DML is performed regularly on these tables)
  - Tables that change infrequently are less expensive to recluster
  - Tables whose query performance degrades noticeably over time



# CLUSTERING COMMAND SAMPLES

```
CREATE TABLE t1 (c1 date, c2 string, c3 number)
CLUSTER BY (c1, c2);
```

```
CREATE TABLE t2 (c1 timestamp, c2 string, c3 number)
CLUSTER BY (TO_DATE(c1), SUBSTRING(c2, 0, 10));
```

```
ALTER TABLE t1
CLUSTER BY (c1, c3);
```

```
ALTER TABLE t2
CLUSTER BY (SUBSTRING(c2, 5, 10), TO_DATE(c1));
```



# Module 12: Performance & Concurrency

## Exercise 12.1: Natural Clustering and Clustering Keys

20 minutes

In this lab, you will compare clustering information and performance on a table that is naturally clustered, one with clustering keys, and one that has been specifically ordered by the column you are querying.



# Other Performance Tips

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# TYPICAL DATABASE ORDER OF EXECUTION



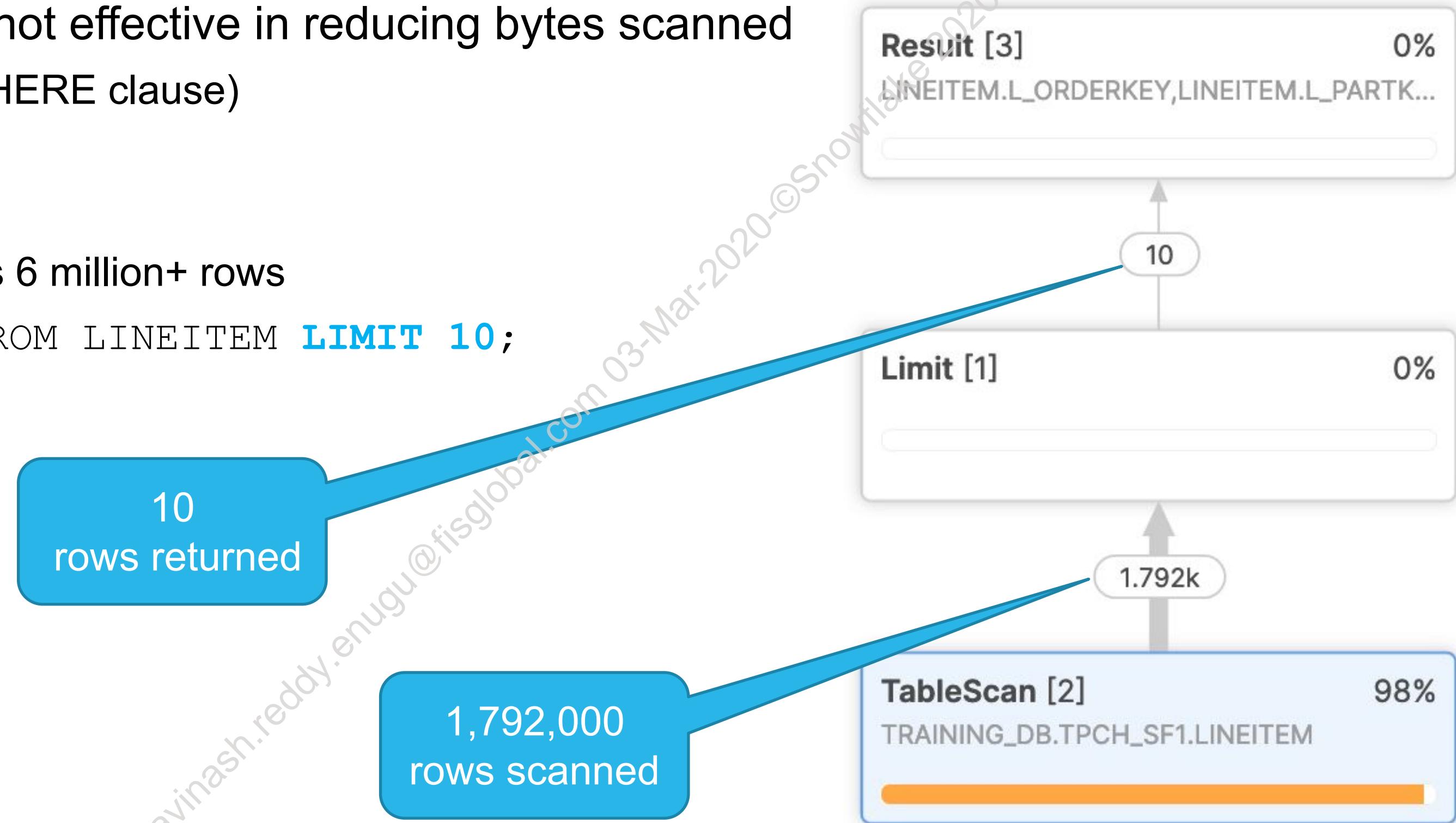
# TOP PERFORMANCE TIP

- **ROW OPERATIONS ARE PERFORMED BEFORE GROUP OPERATIONS**
- **CHECK ROW OPERATIONS FIRST**
  - First check FROM and WHERE clauses
  - Then check GROUP BY and HAVING clauses
- **BEST PRACTICES: USE APPROPRIATE FILTERS AND APPLY EARLY**



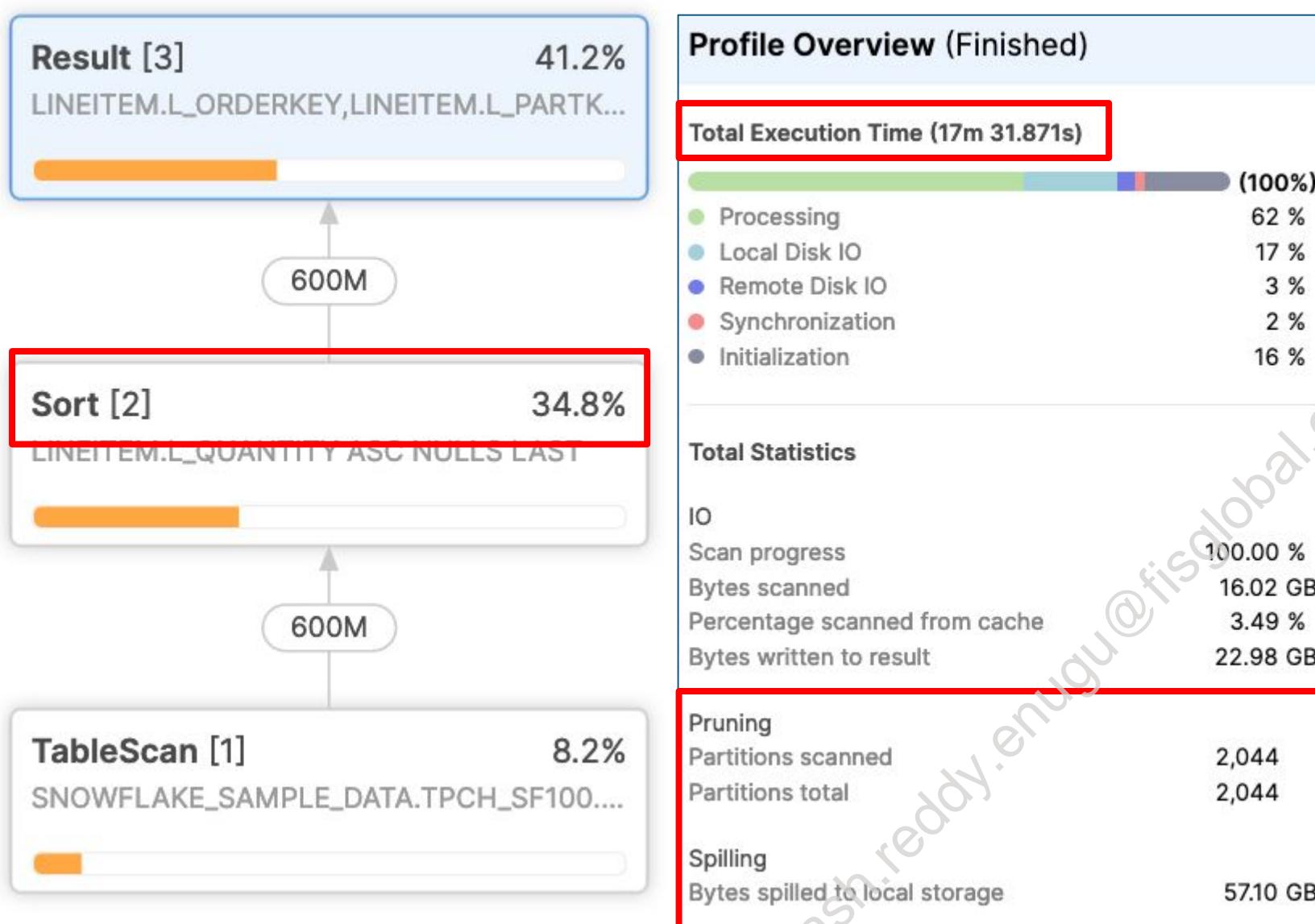
# PITFALLS OF USING LIMIT

- LIMIT alone is not effective in reducing bytes scanned
  - Add filters (WHERE clause)
- Example:
  - LINEITEM has 6 million+ rows
  - SELECT \* FROM LINEITEM **LIMIT 10;**

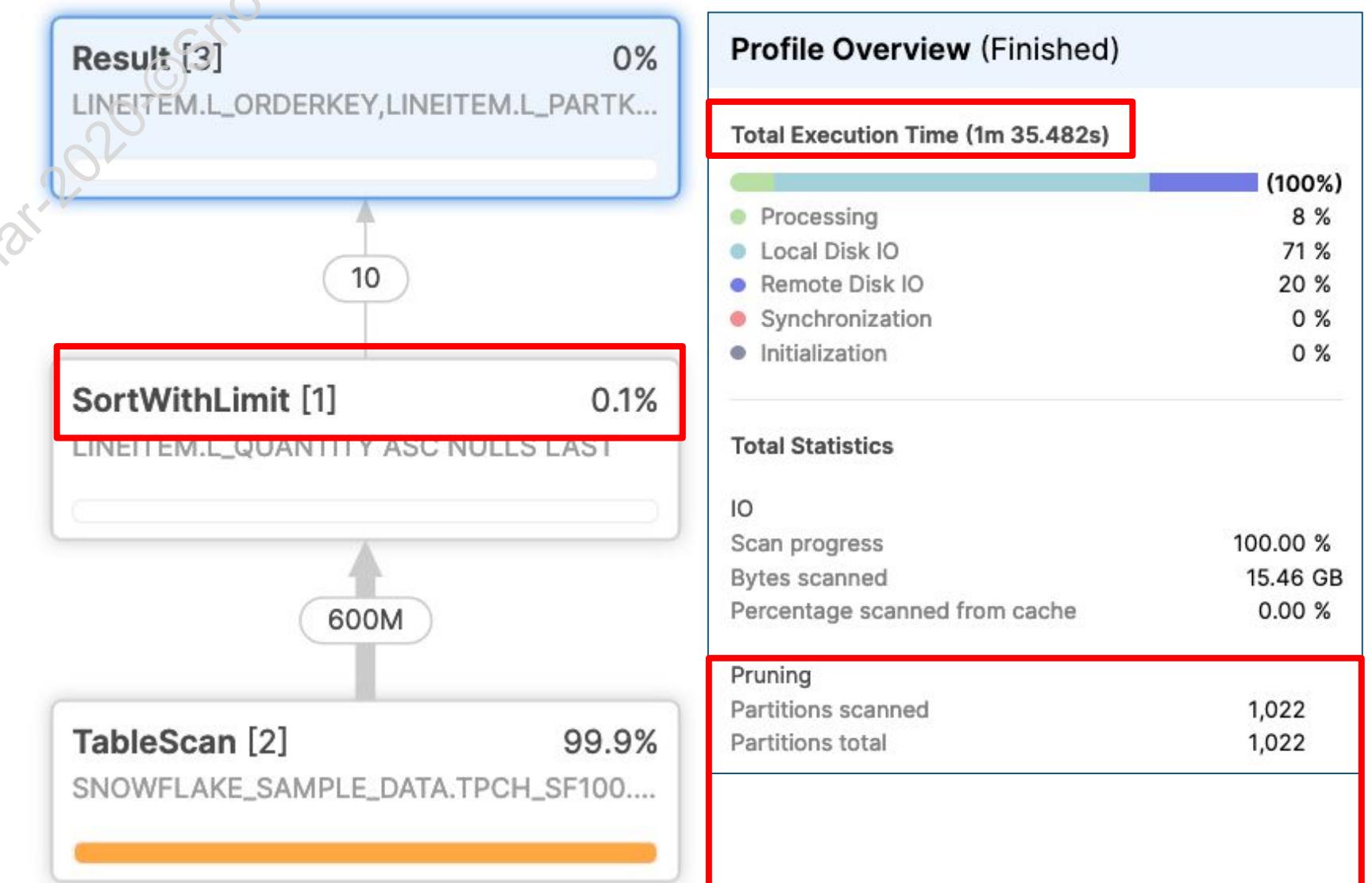


# ADD LIMIT TO LARGE ORDER BY

```
SELECT * FROM LINEITEM
ORDER BY L_QUANTITY;
```



```
SELECT * FROM LINEITEM
ORDER BY L_QUANTITY LIMIT 10;
```



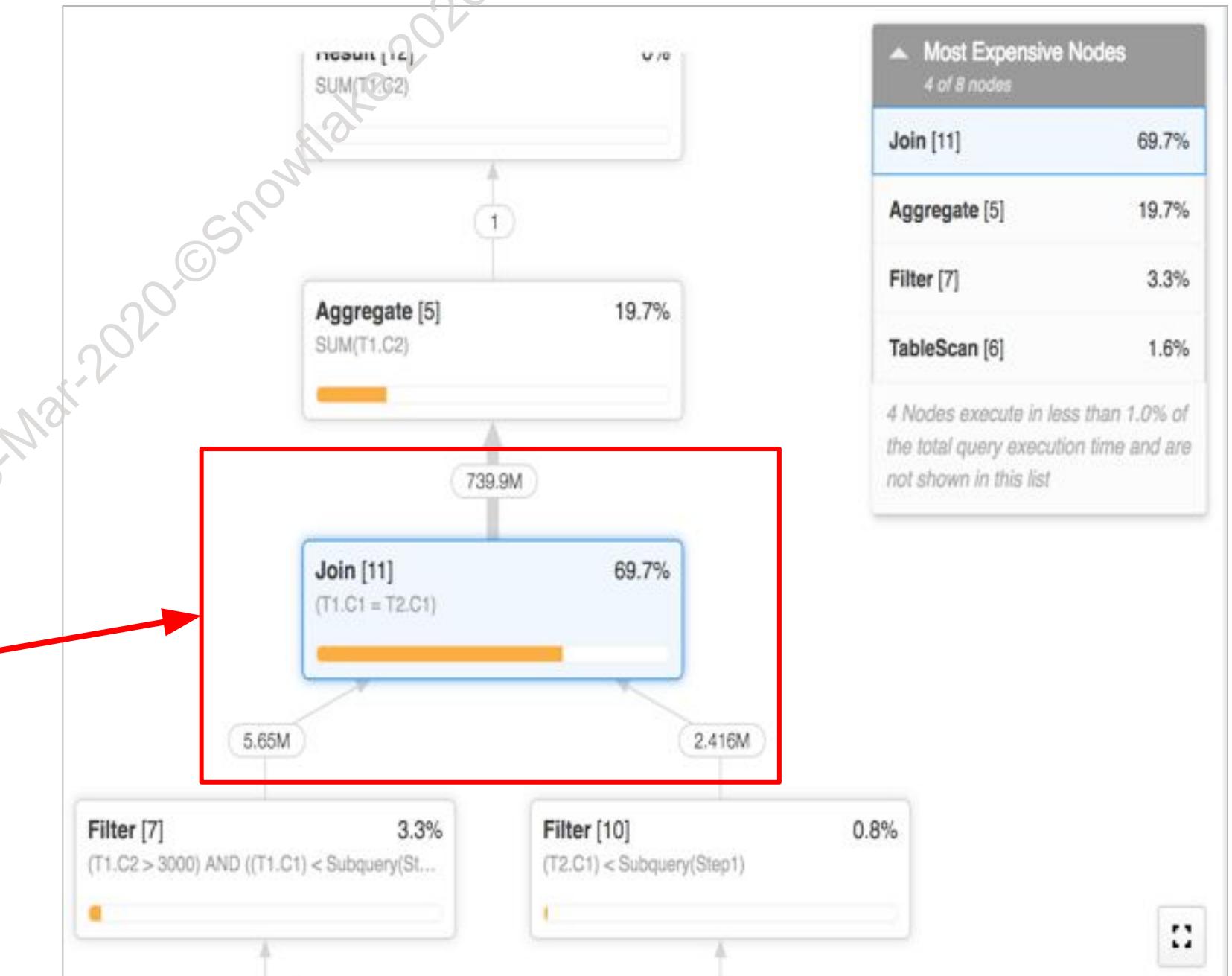
# JOIN ON UNIQUE KEYS

## Best Practices

- Ensure keys are distinct
- Understand relationships between your tables before joining
- Avoid many-to-many join
- Avoid unintentional cross join

## Troubleshooting Scenario

- Joining on non-unique keys can explode your data output (***join explosion***)
  - Each row in table1 matches multiple rows in table 2



# SNOWFLAKE BUILT-IN OPTIMIZATIONS

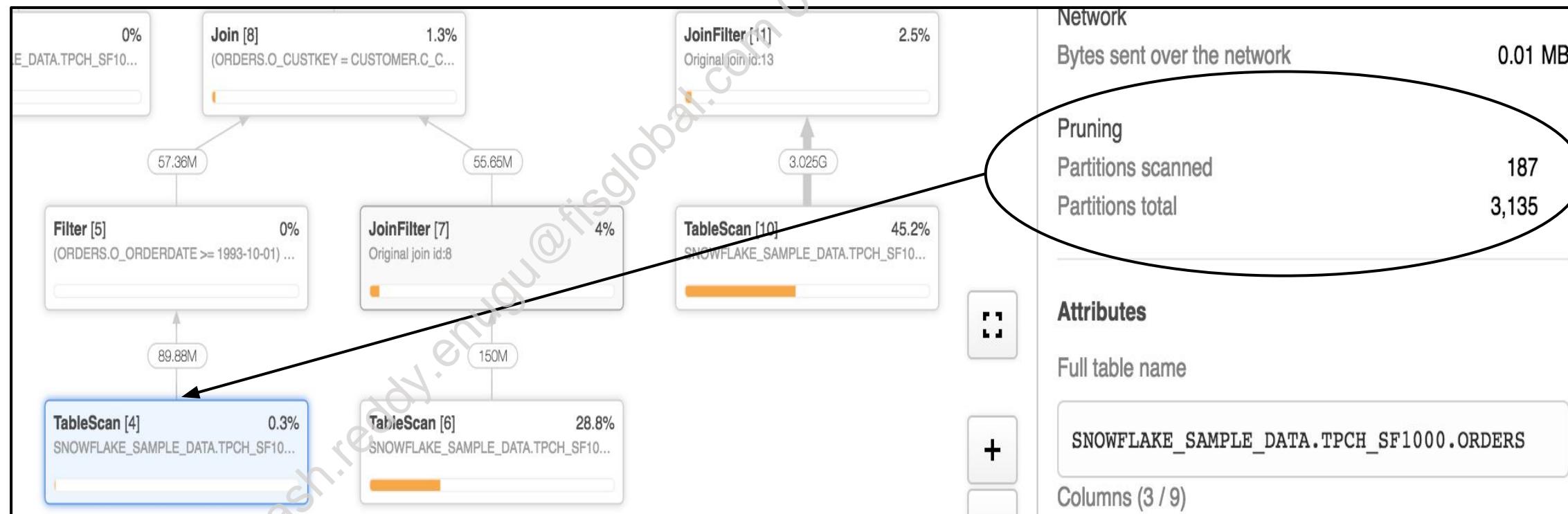
- Snowflake provides patented micro-partition pruning to optimize query performance:
  - Static partition pruning based on columns in WHERE clause
  - Dynamic partition pruning based on JOIN columns of a query
- Best practices to assist SQL optimizer:
  - Apply appropriate filters as early as possible in the query
  - For naturally clustered tables, apply appropriate predicate columns (e.g. date columns) that have a high correlation to ingestion order



# PARTITION PRUNING

Effective pruning: query's filter column matches table's clustering order

```
SELECT <items>
FROM order
WHERE o_orderdate >= to_date('1993-10-01')
AND o_orderdate < dateadd(month, 3, to_date('1993-10-01'));
```



# PREDICATES AND PERFORMANCE

- Built-in functions and UDFs are very useful, but they can impact performance when used in a query predicate

```
SELECT l_orderkey
FROM lineitem l, orders o
WHERE l_orderkey=o_orderkey AND
LOG(10, l_extendprice) > 4.5 AND
LOG(10, o_totalprice - l_tax)> 4.5
```

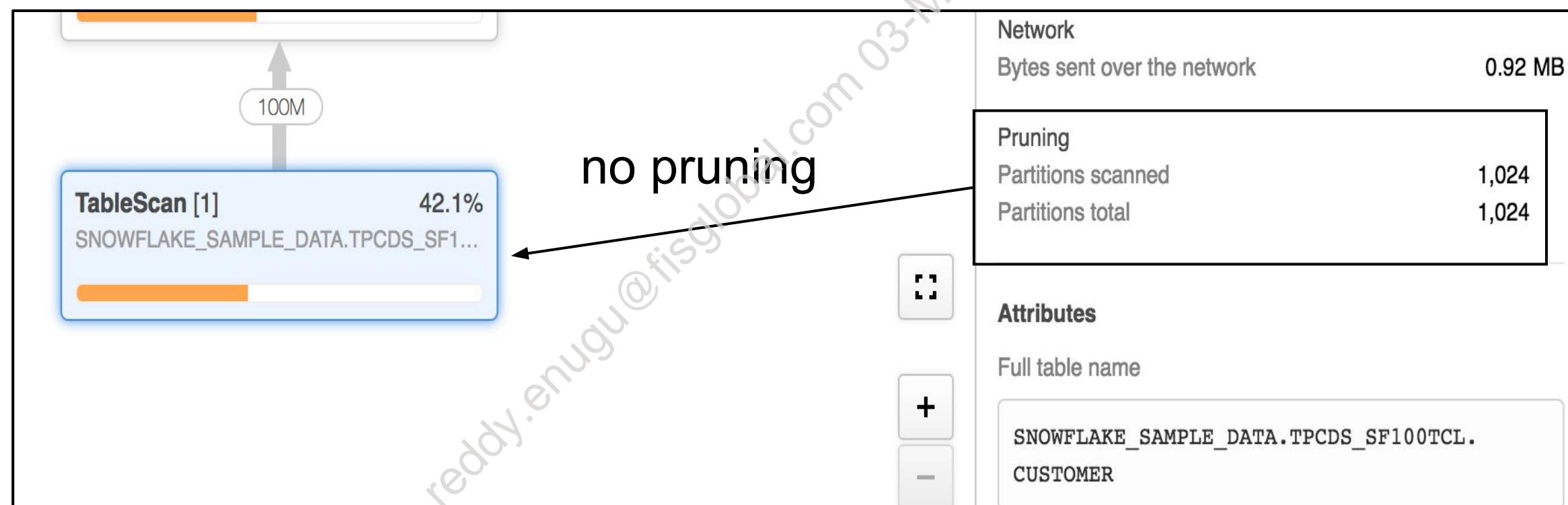
- Consider materializing the intermediate result using a temporary table



# PITFALLS OF NON-PERFORMING PREDICATES

Some WHERE predicates provide no opportunity to optimize pruning

```
SELECT c_customer_id, c_last_name
FROM tpcds.customer
WHERE UPPER (c_last_name) LIKE '%KROLL%'
```



# UNCORRELATED VS CORRELATED SUBQUERIES

- Uncorrelated scalar subqueries (with no external column references)

```
SELECT p.name, p.annual_wage, p.ctry FROM pay AS p
WHERE p.annual_wage <
 (SELECT per_capita_gdp FROM intl_gdp
WHERE country = 'Brazil');
```

- Correlated scalar subqueries in WHERE clause

```
SELECT p.name, p.annual_wage, p.ctry FROM pay AS p
WHERE p.annual_wage <
 (SELECT MAX(per_capita_gdp) FROM intl_gdp i
WHERE p.ctry = i.country);
```



# GROUP BY WITH FEW DISTINCT VALUES

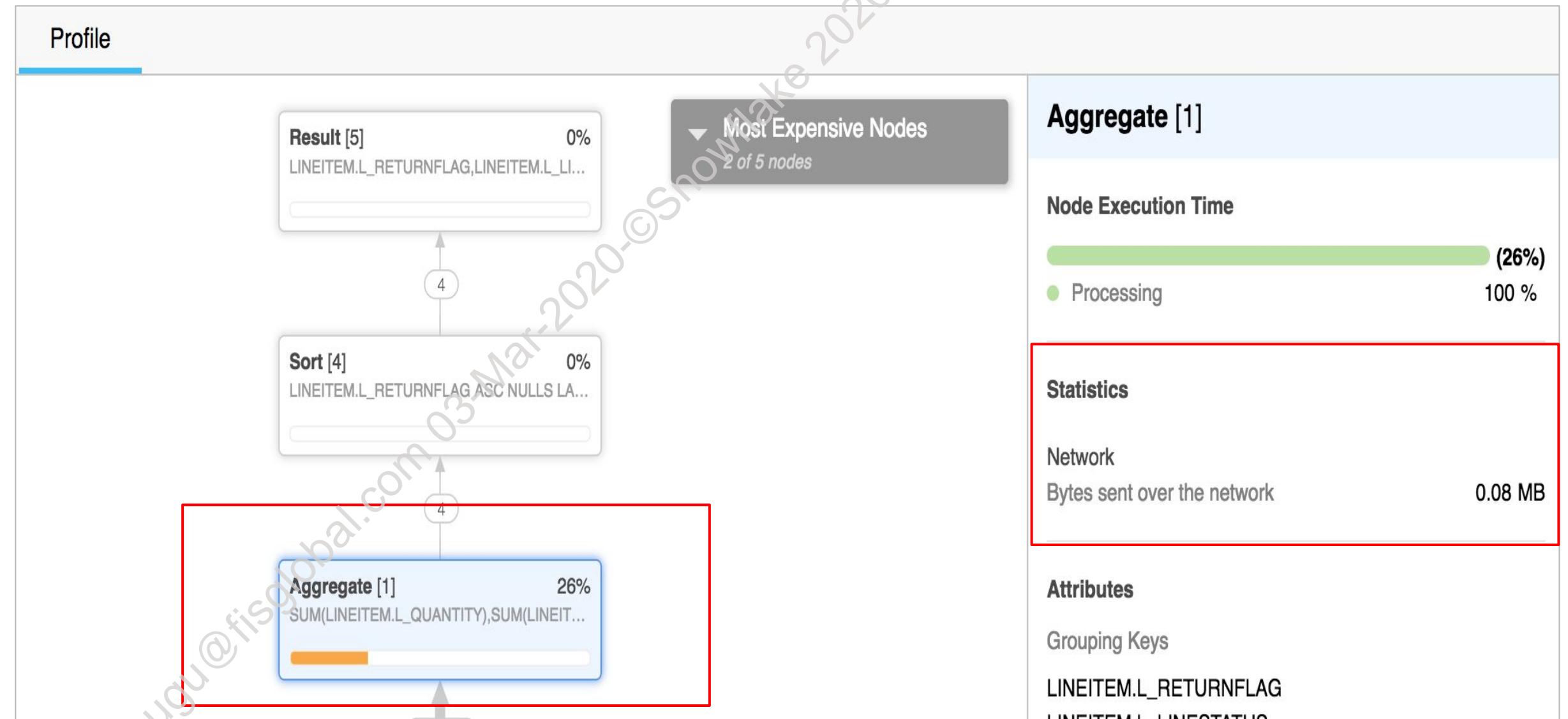
SELECT

```
 l_returnflag,
 l_linenstatus,
 SUM(l_quantity)
```

FROM lineitem

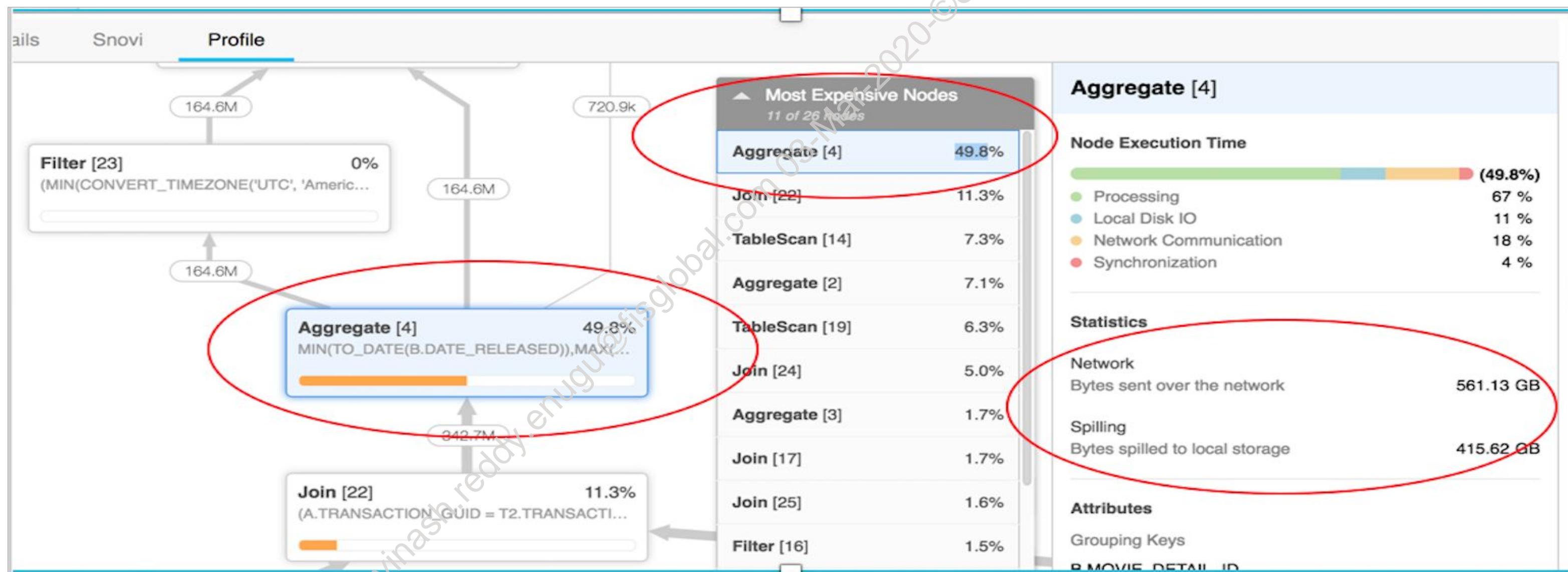
**GROUP BY**

```
 l_returnflag,
 l_linenstatus;
```



# GROUP BY WITH MANY DISTINCT VALUES

- Memory-intensive
- Spilling to disk, high network data
- Sub-optimal performance



# ORDER BY

```
select
 o_orderpriority,
 count(*) as order_count
from
 orders
where
 o_orderdate >= to_date('1993-07-01')
 and o_orderdate < dateadd(month, 3, to_date('1993-07-01'))
 and exists (
 select
 *
 from
 lineitem
 where
 l_orderkey = o_orderkey
 and l_commitdate < l_receiptdate
 order by l_orderkey
)
group by
 o_orderpriority
order by
 o_orderpriority;
```

**Wasted Compute**

**Best Practice**

- Orders values in ascending or descending order on a specified column
- Use ORDER BY in top level SELECT only
  - ORDER BY in subqueries does not impact the result, and slows performance



# Module 12: Performance & Concurrency

## Exercise 12.2: Review the Query Profile

15 minutes

### Tasks:

- Run a query
- Review the query profile



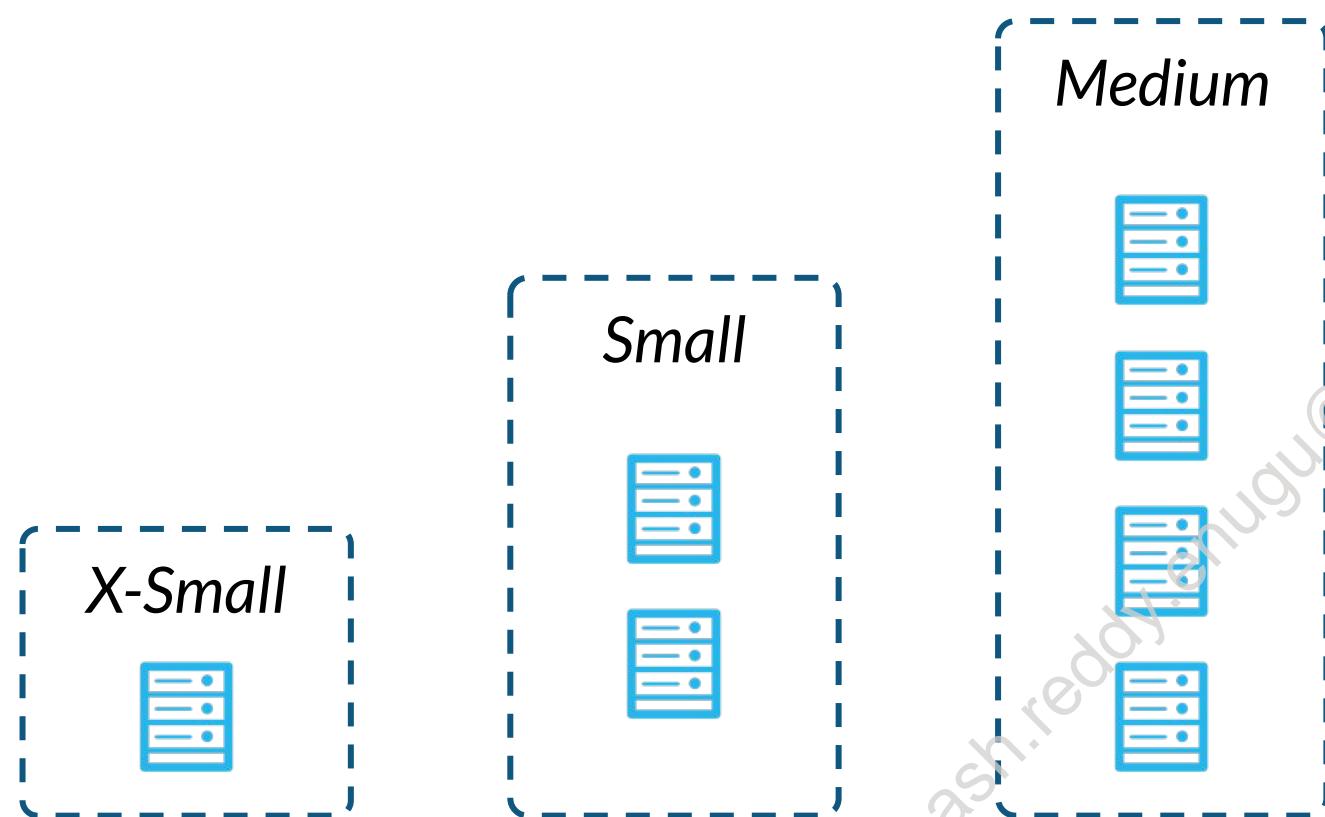
# Virtual Warehouse Scaling

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

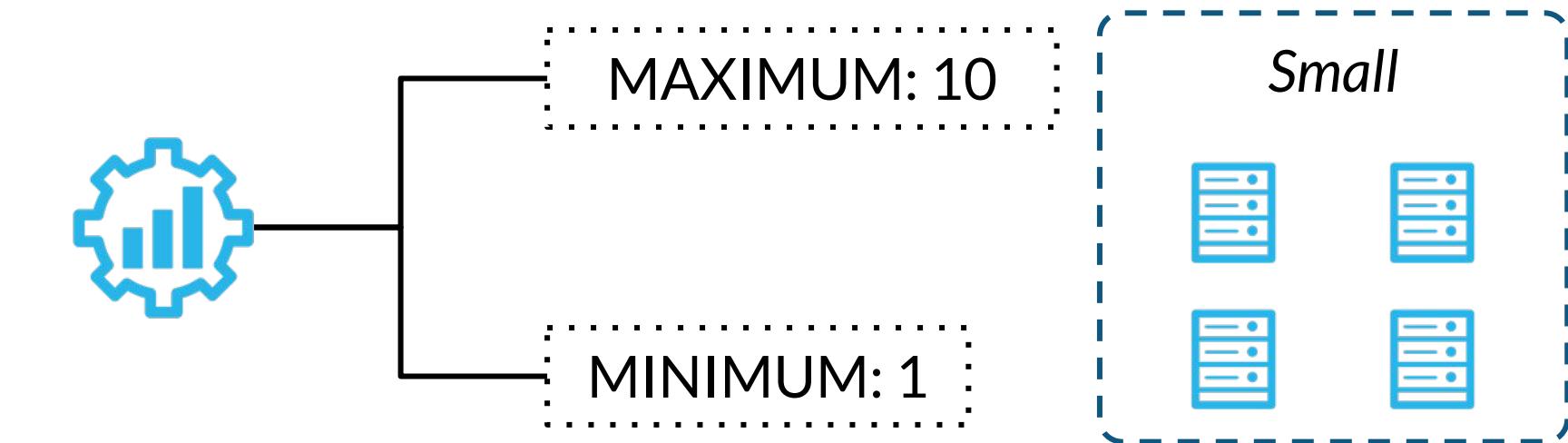


# SCALING UP VS. SCALING OUT

Resizing a warehouse to handle complex/process-intensive queries



Adding more clusters to your current Virtual Warehouse *without* changing the size of the Warehouse to handle concurrency issues.



# SERVERS PER CLUSTER

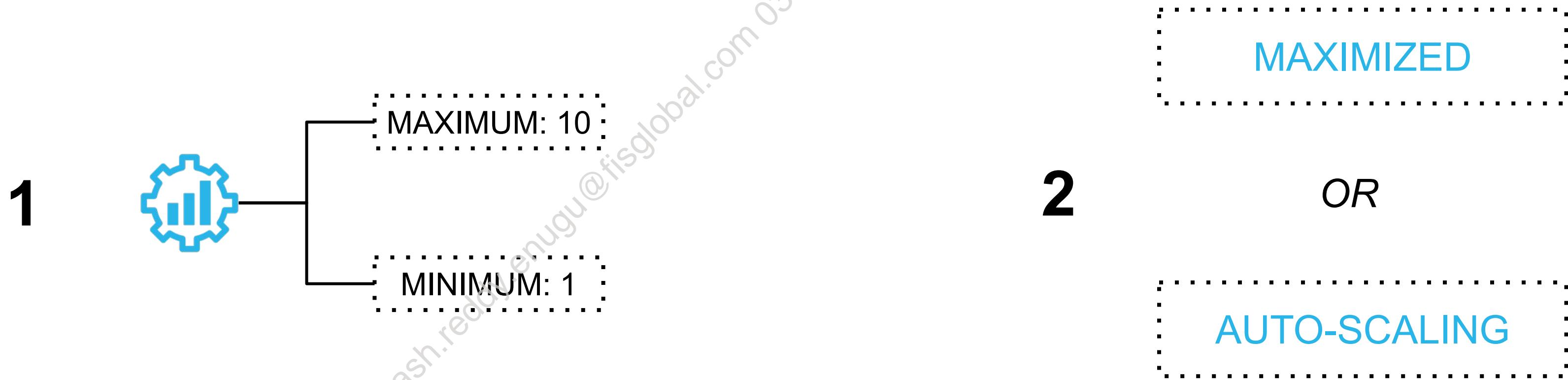
| Warehouse Size | Servers | Clusters |
|----------------|---------|----------|
| X-Small        | 1       | 1        |
| Small          | 2       | 1        |
| Medium         | 4       | 1        |
| Large          | 8       | 1        |
| X-Large        | 16      | 1        |
| 2X-Large       | 32      | 1        |
| 3X-Large       | 64      | 1        |
| 4X-Large       | 128     | 1        |

- As queries are submitted, required resources are calculated and reserved
- If there are insufficient resources, that query is *queued* until other queries finish and release resources



# SCALING OUT: MULTI-CLUSTER WAREHOUSES

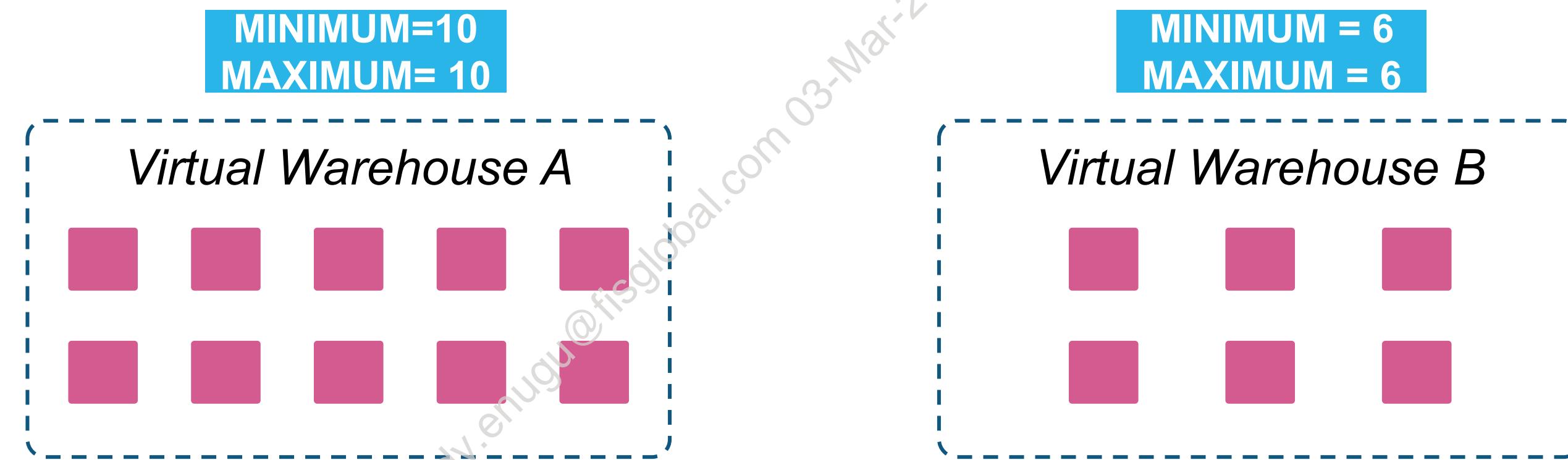
- Multi-cluster warehouses are designed specifically for handling queuing and performance issues related to large numbers of concurrent users and/or queries.
- Multi-cluster warehouses can help automate this process if your number of users/queries tend to fluctuate.



# SCALING OUT: MAXIMIZED

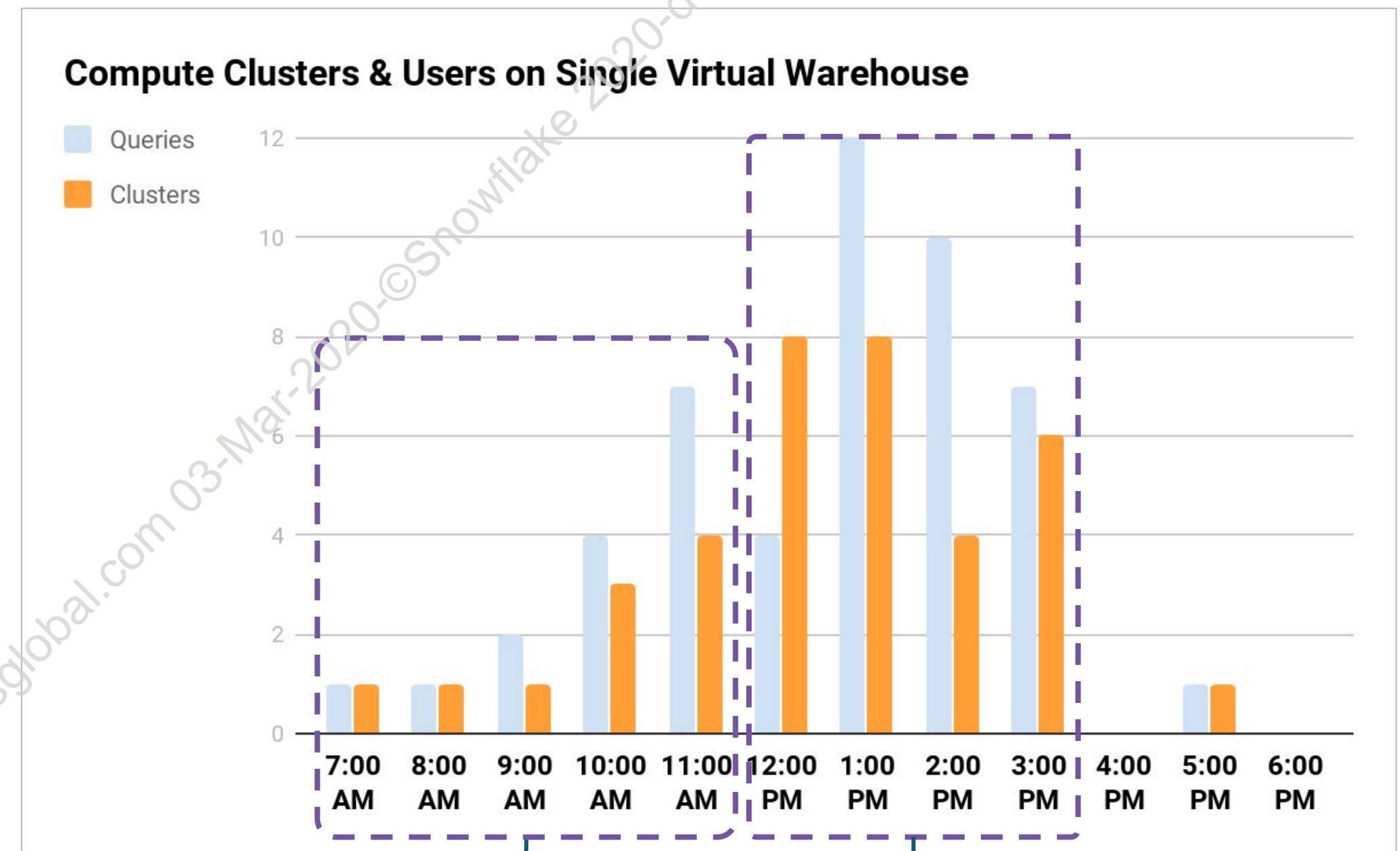
## Maximized:

When the warehouse is started, Snowflake starts all the clusters so that maximum resources are available *while* the warehouse is running.



# SCALING OUT: AUTO-SCALING

Allows Snowflake to start and stop clusters as needed to dynamically manage the load on the warehouse

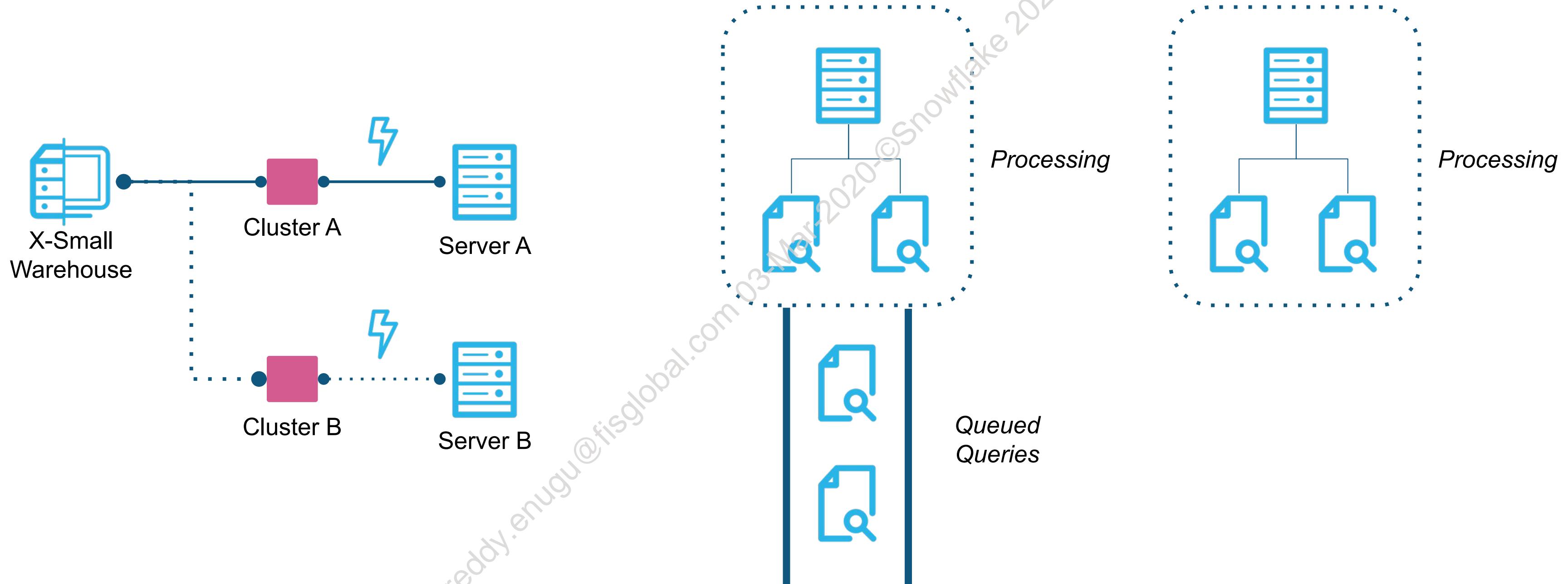


# AUTO-SCALING: SCALING POLICY

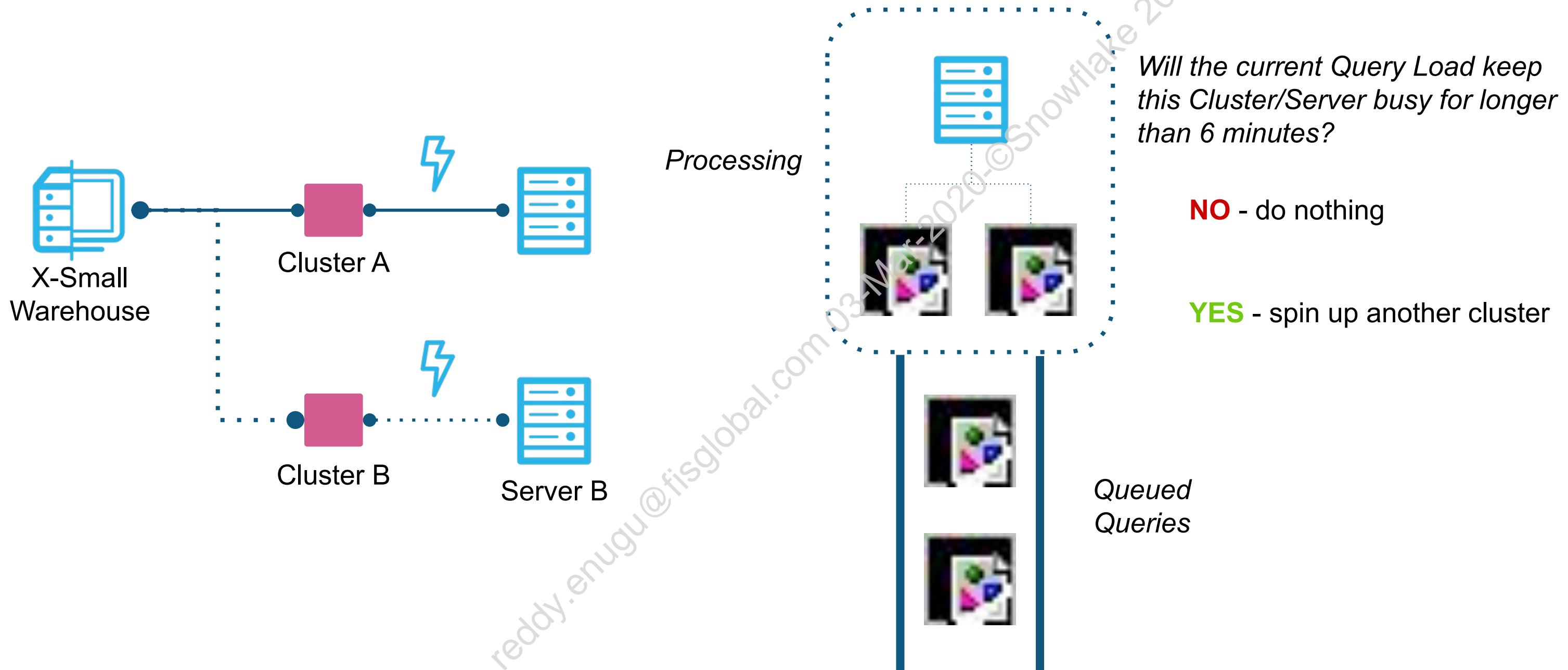
| Policy   | Description                                                                                                                                                        | Cluster Starts...                                                                                                                            | Cluster Shuts down...                                                                                                                                                                           |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Standard | Minimizes queuing and starts additional clusters.                                                                                                                  | Immediately when either a query is queued or the system detects that there's one more query than the currently-running clusters can execute. | After 2-3 consecutive checks (performed at 1 minute intervals) determine that the load on the least-loaded cluster could be redistributed to the others without spinning up the cluster again.  |
| Economy  | Favors keeping running clusters fully-loaded rather than starting additional clusters; could result in jobs being queued rather than starting additional Clusters. | Only if the system estimates there's enough query load to keep the cluster busy for at least 6 minutes.                                      | After 5-6 consecutive checks (performed at 1 minute intervals), determine that the load on the least-loaded cluster could be redistributed to the others without spinning up the cluster again. |



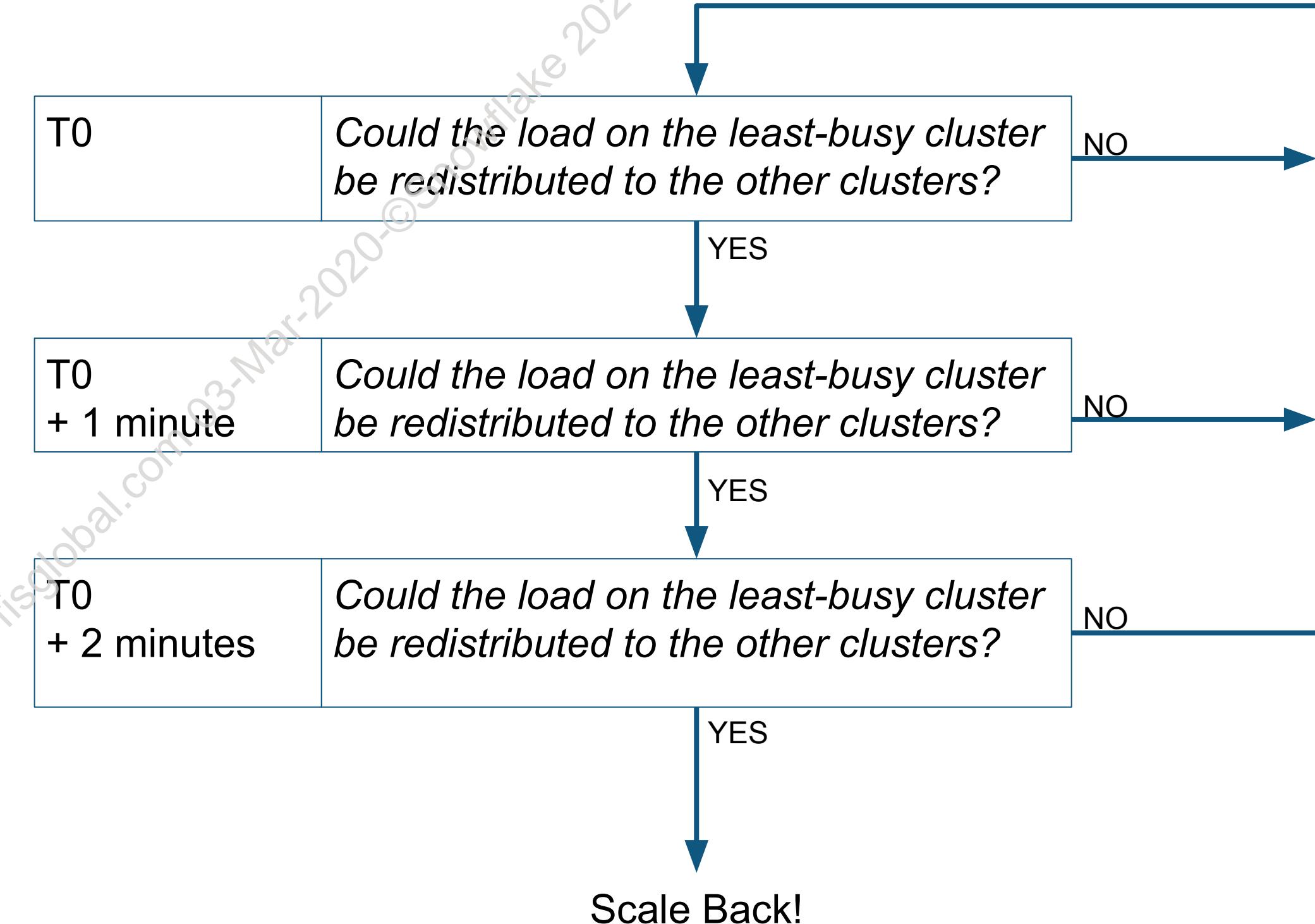
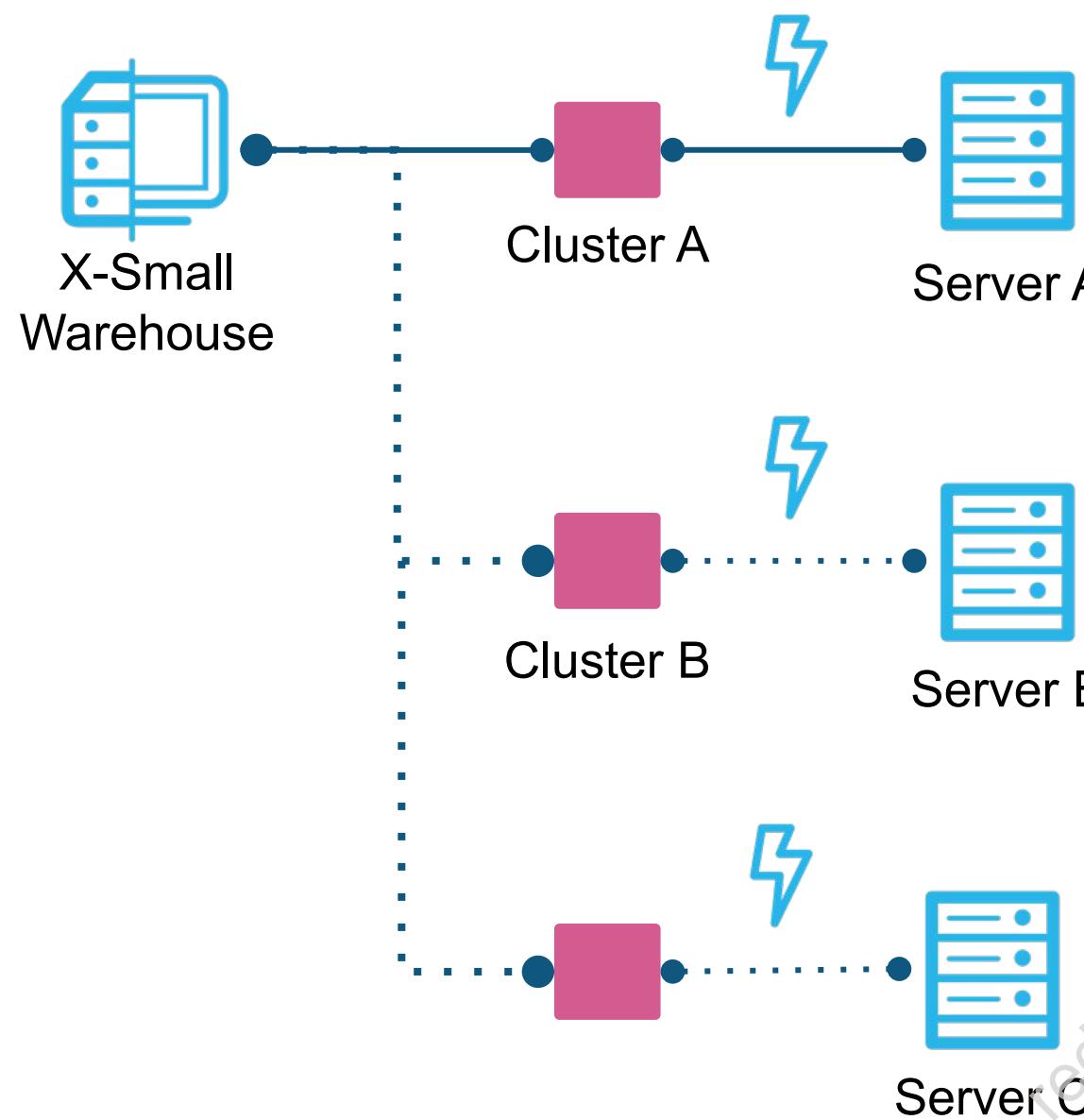
# SCALING OUT POLICY: STANDARD



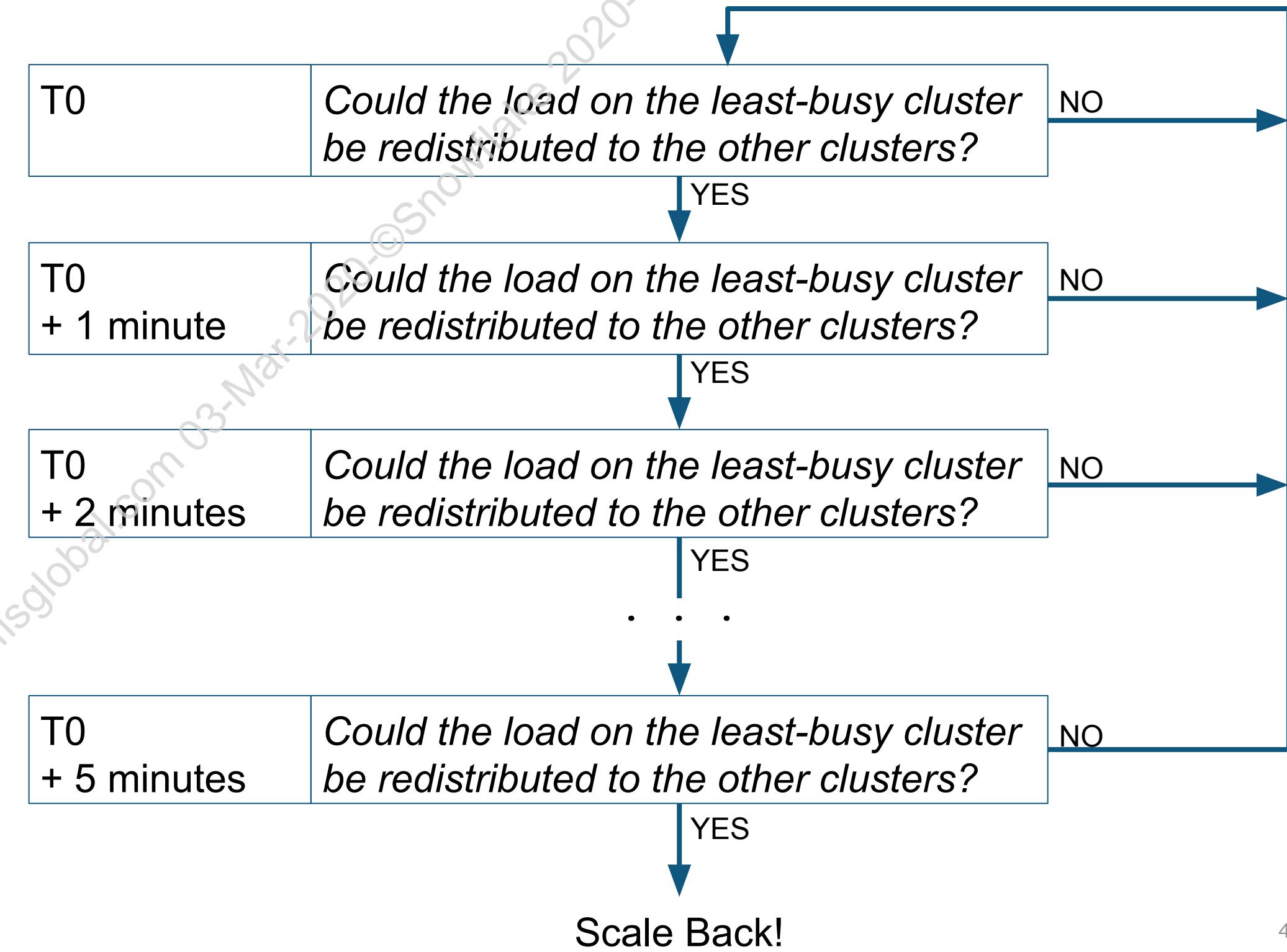
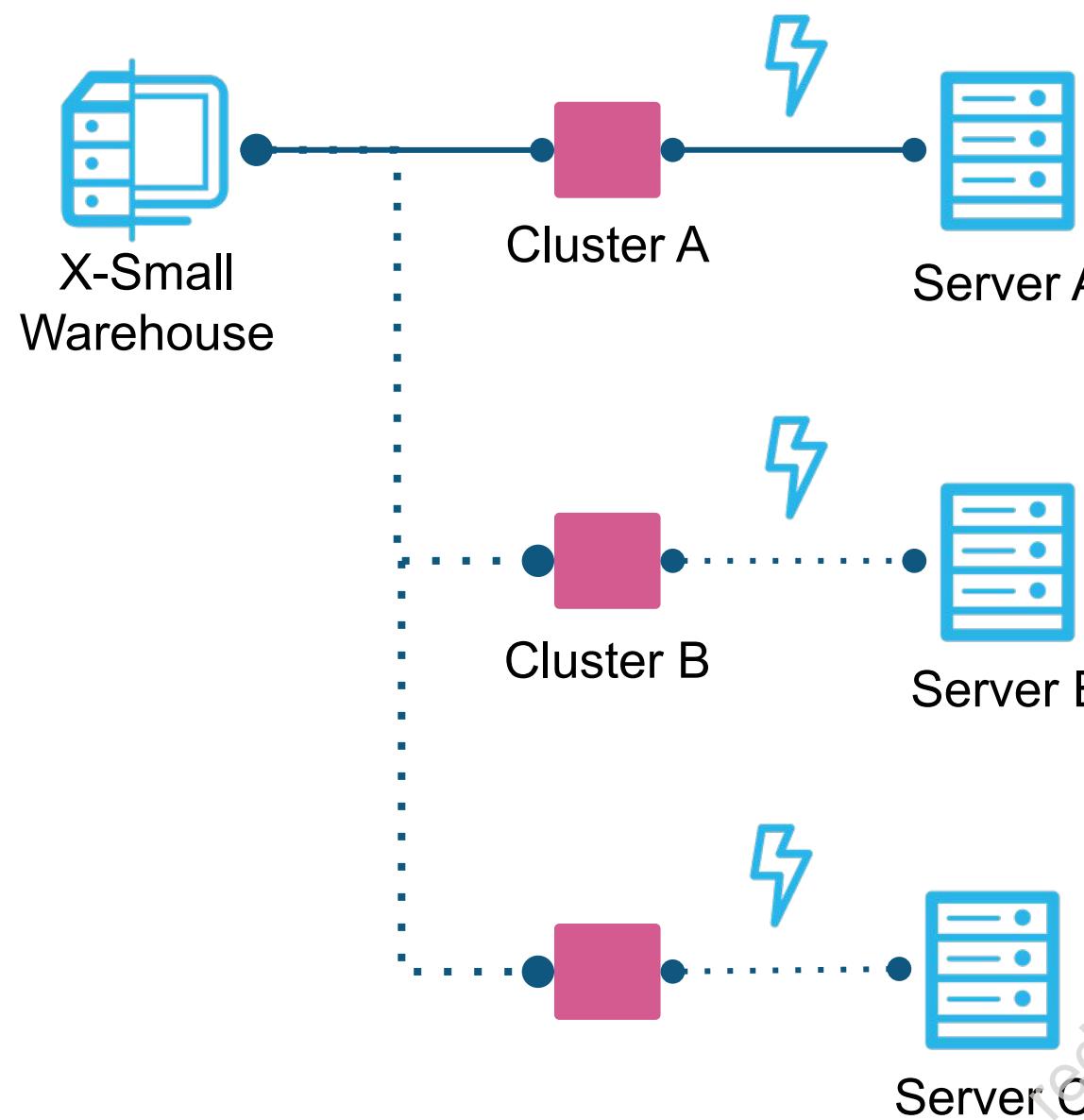
# SCALING OUT POLICY: ECONOMY



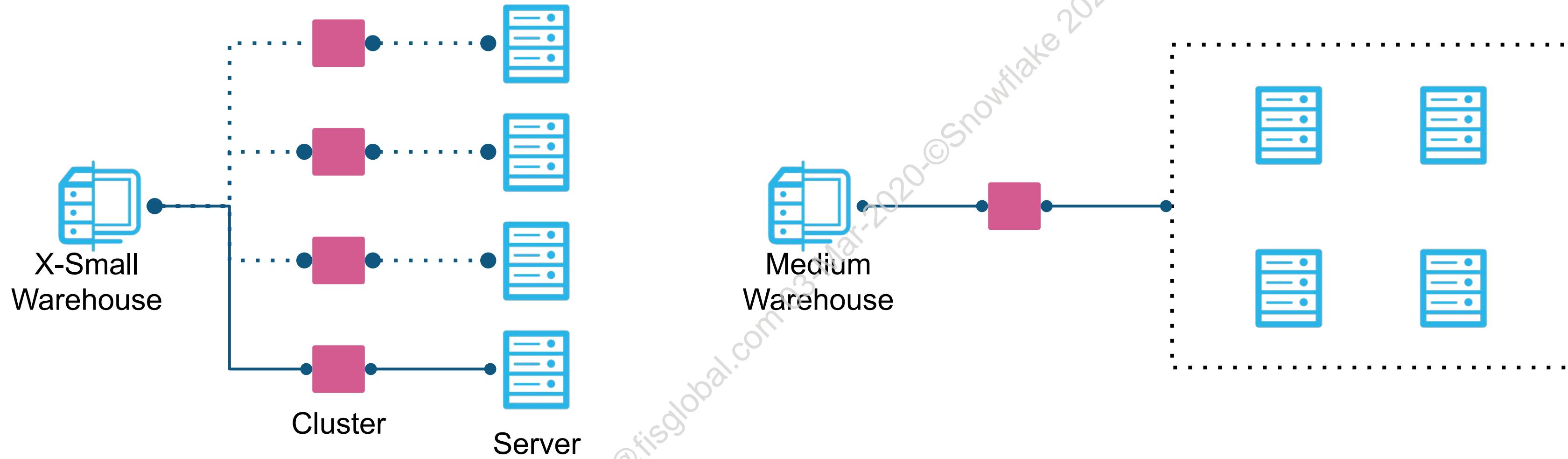
# SCALING BACK POLICY: STANDARD



# SCALING BACK POLICY: ECONOMY



# ARE THESE WAREHOUSES EQUIVALENT?



## Auto-Scale, Multi-Cluster X-Small Warehouse

- Minimum of 1
- Maximum of 4

## Medium Warehouse

- Comes with 1 cluster of 4 servers by default.



# ALTERING QUERY BEHAVIOR

STATEMENT\_QUEUED\_TIMEOUT\_IN\_SECONDS

- How long a queued query will wait before being cancelled by the system
- Can be set for Account/Session/Role or Virtual Warehouse
- Default: 0

STATEMENT\_TIMEOUT\_IN\_SECONDS

- How long a query can run before being cancelled by the system
- Set for Account/Session/Role or Virtual Warehouse
- ***Default: 2 days → Deployment best practice → lower at the account level to your preference***



# Module 11: Performance & Concurrency

## Exercise 11.3: Determine Appropriate Warehouse Sizes

60 minutes

### Tasks:

- JOIN using an XSmall warehouse
- JOIN using a Medium warehouse
- Determine appropriate warehouse size



# LAB RECAP

In Exercise 11.3, you ran the same query against several warehouse sizes.

Which warehouse size would you recommend, and why?

| Warehouse Size | Query Time | Credits |
|----------------|------------|---------|
| Small          | 2m 27s     | .0882   |
| Medium         | 1m 00s     | .0660   |
| Large          | 0m 30s     | .0660   |
| XLarge         | 0m 16s     | .0704   |
| XXLarge        | 0m 10s     | .0890   |



# Module 13

# Account & Resource Management

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Controlling Costs
- Resource Monitors
- INFORMATION\_SCHEMA
- SNOWFLAKE Database

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# Controlling Costs

avinash.reddy.enugu@fisglobal.com 02-Mar-2020 ©Snowflake 2020-do-not-copy



# PERFORMANCE AND USAGE CONSIDERATIONS

Granularity type

## STORAGE



- Database
- Table
- Stage
- Account

## COMPUTE



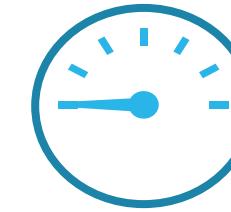
- Compute cluster aka Virtual warehouse
- Query/Job workload
- Serverless (Snowpipe, Materialized View, Auto Clustering)

## COST



- Compute Cluster (Virtual Warehouse)
- Storage
- Account

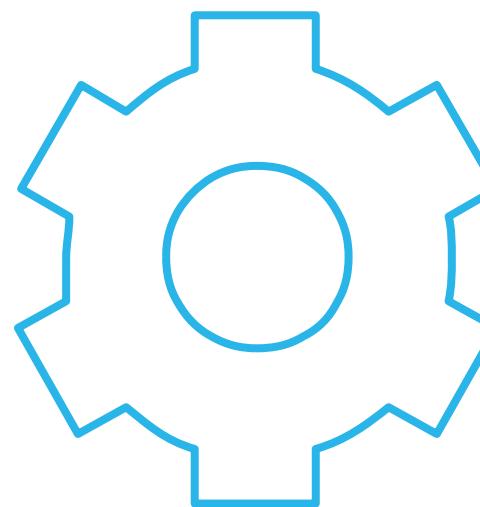
## PERFORMANCE



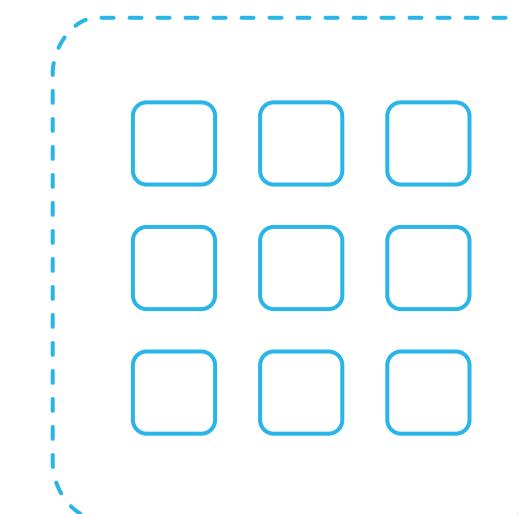
- Execution time of Query/Job workload
- Compilation time

# COMPUTE (CREDIT) AND STORAGE USE

- Goal of Snowflake pricing is to enable flexibility and scalability to data warehousing at a low cost and in the simplest possible way
- Costs are associated based on usage of two distinct functions



**Compute**  
(Virtual Warehouses)

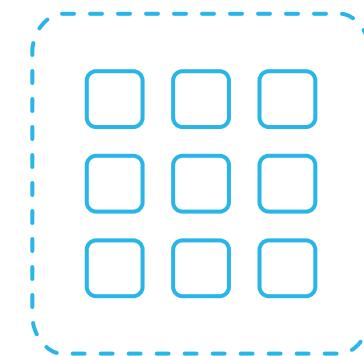


**Storage**  
(Internal Stages +  
Active, Time Travel, Fail-Safe Data)

# STORAGE USAGE



**Internal Stages**  
(User, Table, Named)



**Database Tables**  
(Active, Time Travel, Fail-Safe)

- Billed monthly
- Calculated based on daily bytes of all data stored each day in account
- Cost varies based on type of account, cloud provider, and region
- Usage can be viewed via the UI, SHOW commands, or Information Schema/Account Usage



# USER-MANAGED COMPUTE

| Warehouse Size | Servers / Cluster | Credits / Hour | Credits / Second |
|----------------|-------------------|----------------|------------------|
| X-Small        | 1                 | 1              | 0.0003           |
| Small          | 2                 | 2              | 0.0006           |
| Medium         | 4                 | 4              | 0.0011           |
| Large          | 8                 | 8              | 0.0022           |
| X-Large        | 16                | 16             | 0.0044           |
| 2X-Large       | 32                | 32             | 0.0089           |
| 3X-Large       | 64                | 64             | 0.0178           |
| 4X-Large       | 128               | 128            | 0.0356           |

- Only billed when running
- Billed per-second, with a 60-second minimum
- Can be limited and controlled via resource monitors
- Usage can be viewed via the UI or Information Schema/Account Usage



# COST CONTROL

- Consider impact of data cache when setting auto-suspend values
- Multi-cluster warehouses and auto-scaling
- Time travel settings
  - Unlimited storage comes with a storage cost
- Privileges for creating warehouses
- Configure Reader accounts
- Resource Monitors



# AUTOMATIC CLUSTERING

- Snowflake internally manages the state of clustered tables as well as resources used for clustering
- Dynamically allocates resources as needed
- Efficient and effective reclustering
- Automatic clustering is transparent, does not block DML statements against tables that are being re-clustered



# MATERIALIZED VIEW MAINTENANCE

- Materialized view maintained automatically as background process
- Can consume significant resources resulting in increased credit usage
- Compute charges based on volume of data changes + number of MVs created on each base table

```
SELECT * FROM
TABLE (INFORMATION_SCHEMA.MATERIALIZED_VIEW_REFRESH_HISTORY());
```

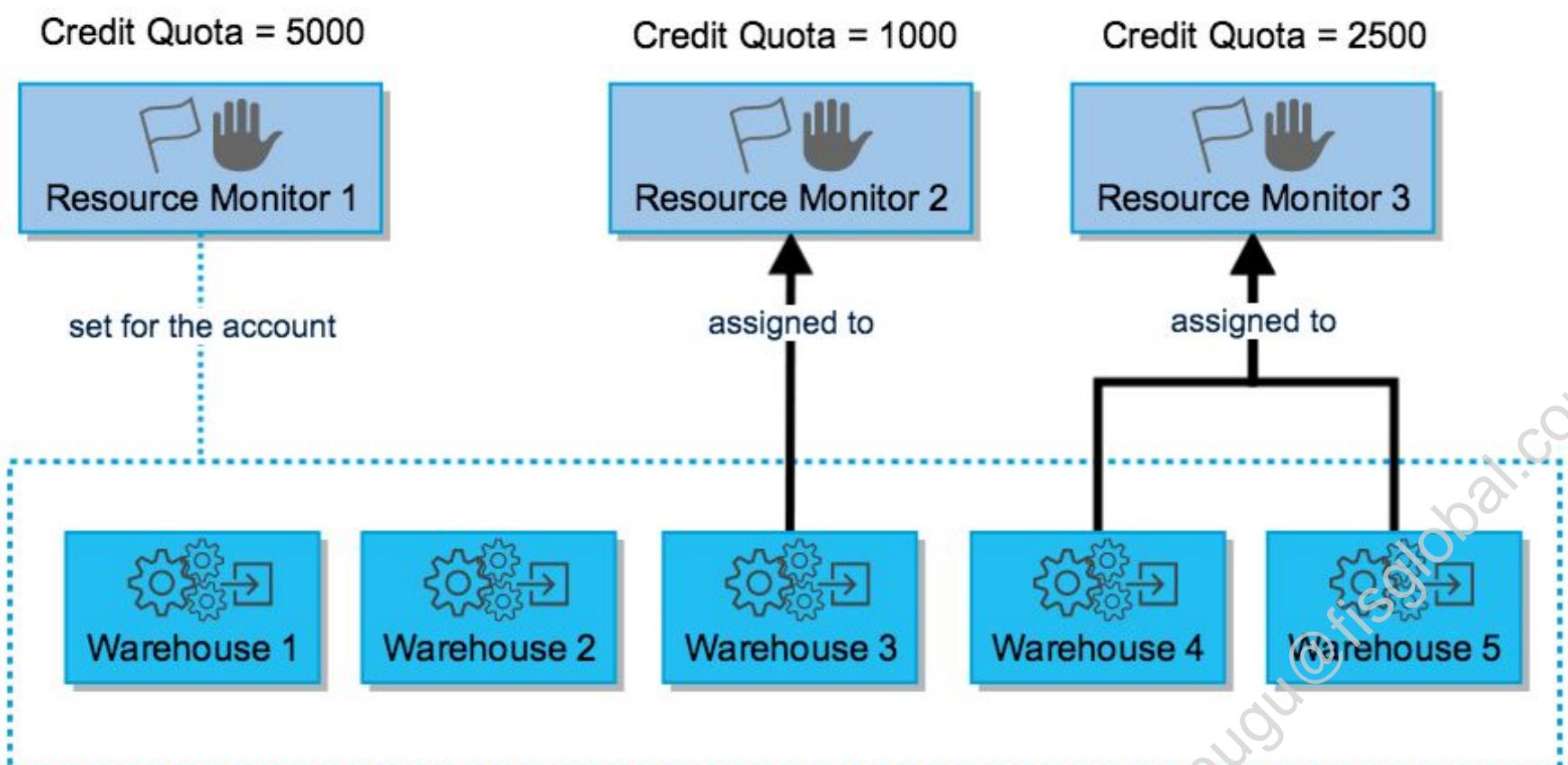


# Resource Monitors

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# RESOURCE MONITORS



- Help control costs and avoid unexpected warehouse credit usage
- Set at the account or warehouse level
- Can trigger various actions
  - Sending alert notifications
  - Suspending the warehouse(s)



# RESOURCE MONITORS - DETAILS

- ACCOUNTADMIN privileges are required to set up Resource Monitors
- Resource Monitor attributes:
  - Credit quota
  - Level (account, warehouse, or warehouses)
  - Schedule (start, end, frequency)
  - Action (notify, suspend, suspend immediate)
- Not intended to strictly control usage to the second
  - Quota may be exceeded before action is triggered
- **Must enable notifications in role Preferences**



# CREATE RESOURCE MONITORS

- Through the UI
- With SQL

```
CREATE RESOURCE MONITOR
ALTER RESOURCE MONITOR
SHOW RESOURCE MONITOR
DROP RESOURCE MONITOR
```



# Resource Management and Monitoring

## Demonstration: Set Up Resource Monitors

10 minutes



© 2019 Snowflake Computing Inc. All Rights Reserved

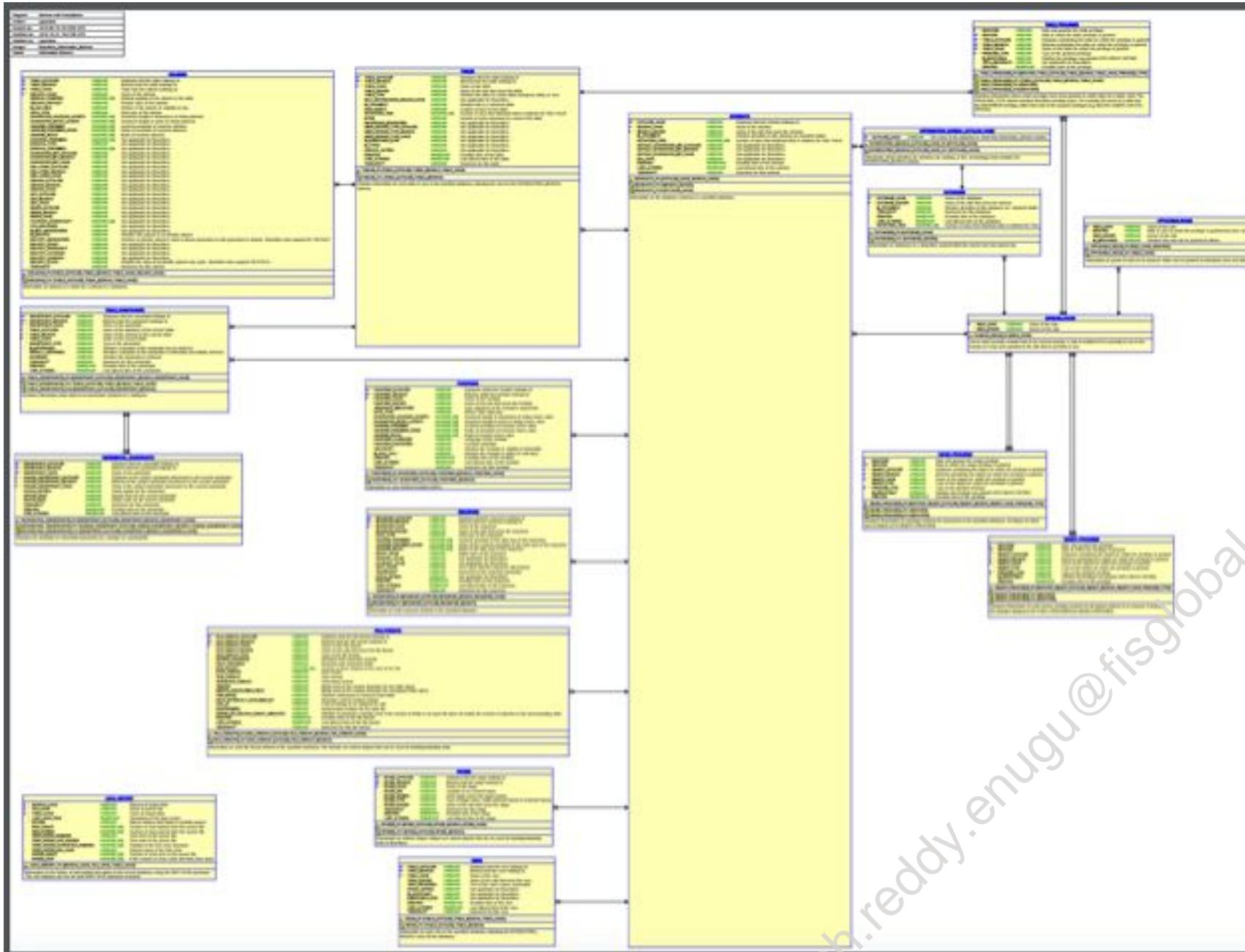
avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

# **INFORMATION\_SCHEMA**

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# INFORMATION\_SCHEMA

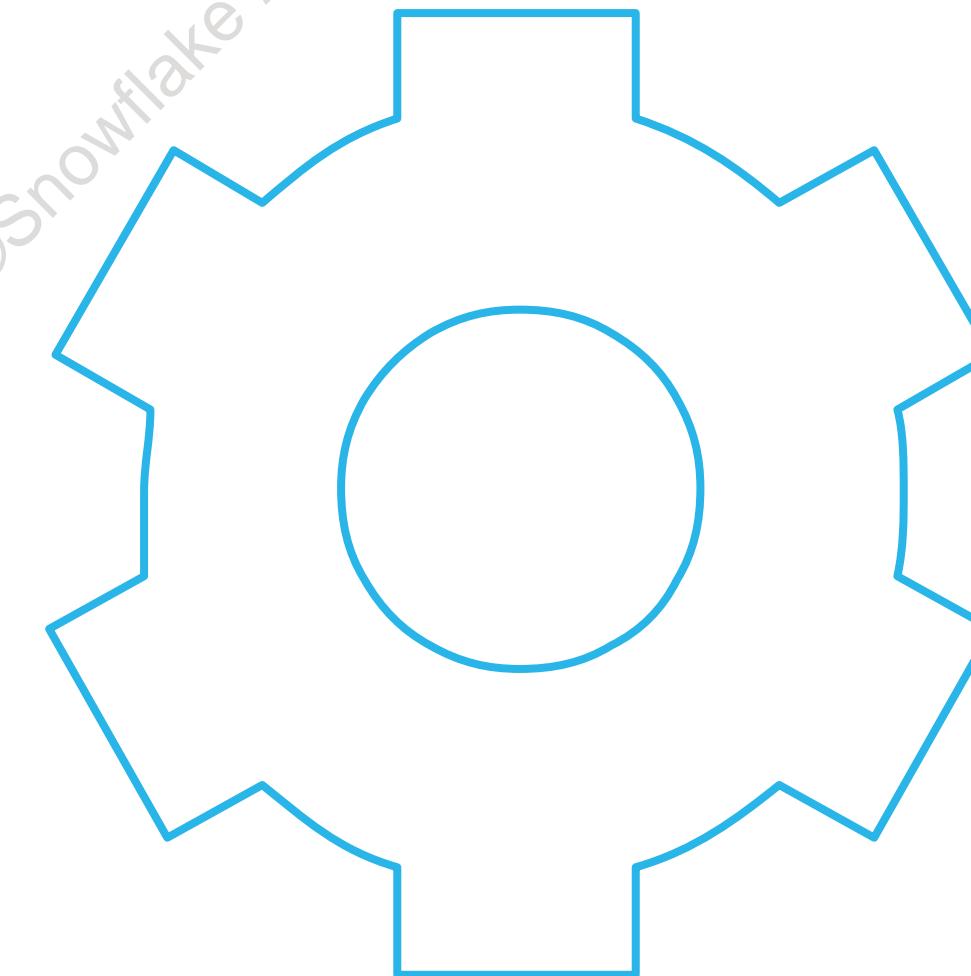


- Set of system-defined views and table functions that provide metadata information about objects
- Based on the SQL-92 ANSI Information Schema + additional views and functions specific to Snowflake
- Queries return only objects to which the current role has been granted access



# INFORMATION\_SCHEMA

- Built-in, read-only schema
- Exists for each database
- System-defined views and table functions
  - Views for all objects in the database
  - Views for account-level objects
  - Table functions for historical and usage data
- Provide extensive “logging” metadata information
- Requires a warehouse to query



# SOME VIEWS IN INFORMATION\_SCHEMA

- APPLICABLE\_ROLES
- COLUMNS
- DATABASES
- ENABLED\_ROLES
- FILE FORMATS
- FUNCTIONS
- LOAD\_HISTORY
- PIPES
- PROCEDURES
- SEQUENCES
- STAGES
- TABLE\_PRIVILEGES
- TABLE\_STORAGE\_METRICS
- TABLES
- USAGE\_PRIVILEGES
- VIEWS



# TABLE FUNCTIONS IN INFORMATION\_SCHEMA

See documentation for a full list

- AUTOMATIC\_CLUSTERING\_HISTORY
- DATABASE\_STORGE\_USAGE\_HISTORY
- LOGIN\_HISTORY
- LOGIN\_HISTORY\_BY\_USER
- MATERIALIZED\_VIEW\_REFRESH\_HISTORY
- QUERY\_HISTORY
- WAREHOUSE\_LOAD\_HISTORY



# INFORMATION\_SCHEMA EXAMPLES

List object privileges

```
SELECT object_type, object_name, privilege_type, grantor, grantee
FROM INFORMATION_SCHEMA.OBJECT_PRIVILEGES
ORDER BY object_type, object_name;
```

| OBJECT_TYPE | OBJECT_NAME | PRIVILEGE_TYPE | GRANTOR       | GRANTEE       |
|-------------|-------------|----------------|---------------|---------------|
| DATABASE    | DBHOL       | OWNERSHIP      | SYSADMIN      | SYSADMIN      |
| DATABASE    | MY_DB       | OWNERSHIP      | TRAINING_ROLE | TRAINING_ROLE |
| DATABASE    | SNOWFLAKE   | USAGE          | ACCOUNTADMIN  | PUBLIC        |
| SCHEMA      | PUBLIC      | OWNERSHIP      | TRAINING_ROLE | TRAINING_ROLE |
| TABLE       | TEST        | OWNERSHIP      | TRAINING_ROLE | TRAINING_ROLE |



# INFORMATION\_SCHEMA EXAMPLES

List total table size, by table, in megabytes (MB)

```
SELECT table_schema, table_name, table_owner, table_type,
 ROUND(bytes/1024/1024,3) AS mb
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA NOT IN ('INFORMATION_SCHEMA');
```

| TABLE_SCHEMA | TABLE_NAME    | TABLE_OWNER   | TABLE_TYPE      | MB      |
|--------------|---------------|---------------|-----------------|---------|
| PUBLIC       | BITS          | TRAINING_ROLE | BASE TABLE      | 0.001   |
| PUBLIC       | CUSTOMER      | TRAINING_ROLE | BASE TABLE      | 103.143 |
| DEV          | MY_TEMP_TABLE | DBA           | LOCAL TEMPORARY | 1048.34 |



# INFORMATION\_SCHEMA EXAMPLES

List privileges granted on databases:

```
SELECT object_name AS database_name, privilege_type, grantee AS granted_to,
 grantor AS granted_by
FROM INFORMATION_SCHEMA.OBJECT_PRIVILEGES
WHERE object_type='DATABASE' AND privilege_type NOT IN ('OWNERSHIP')
ORDER BY database_name;
```

| DATABASE_NAME         | PRIVILEGE_TYPE | GRANTED_TO    | GRANTED_BY   |
|-----------------------|----------------|---------------|--------------|
| MY_DB                 | USAGE          | TRAINING_ROLE | SYSADMIN     |
| PAYROLL               | MONITOR        | DB_MONITOR    | SYSADMIN     |
| SNOWFLAKE             | USAGE          | PUBLIC        | ACCOUNTADMIN |
| SNOWFLAKE_SAMPLE_DATA | USAGE          | PUBLIC        | ACCOUNTADMIN |
| TRAINING_DB           | USAGE          | TRAINING_ROLE | SYSADMIN     |



# **SNOWFLAKE Database**

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# SNOWFLAKE DATABASE

The screenshot shows the Snowflake web interface. At the top, there's a blue header with the Snowflake logo and navigation links for 'Databases' and 'Shares'. Below the header, the path 'Databases > SNOWFLAKE' is displayed. A navigation bar at the top of the main content area includes tabs for 'Tables', 'Views', 'Schemas' (which is underlined in blue), and 'Stages'. Below this are buttons for 'Create...', 'Clone...', 'Alter...', and 'Drop...'. The main content area displays a table titled 'Schemas' with columns 'Schema' and 'Creation Time'. The table contains four rows:

| Schema               | Creation Time      |
|----------------------|--------------------|
| INFORMATION_SCHEMA   | 2:06:05 PM         |
| READER_ACCOUNT_USAGE | 8/15/18 9:14:17 AM |
| ACCOUNT_USAGE        | 8/15/18 9:14:16 AM |

- Shared by Snowflake to all accounts
  - Example of secure data sharing
- Contains 3 schemas:
  - ACCOUNT\_USAGE
  - INFORMATION\_SCHEMA
  - READER\_ACCOUNT\_USAGE
- Contains historical usage data and logs specific to your account

# SNOWFLAKE DATABASE

- Shared by Snowflake as a secure view
- By default, only ACCOUNTADMIN can access the SNOWFLAKE database and schemas
- Privileges can be granted to other roles

```
GRANT IMPORTED PRIVILEGES ON DATABASE SNOWFLAKE TO ROLE <role>;
```



# ACCOUNT\_USAGE

The screenshot shows the Snowflake web interface. In the top navigation bar, the 'snowflake' logo is on the left, followed by 'Databases', 'Shares', and 'Data' tabs. Below the navigation, the path 'Databases > SNOWFLAKE' is displayed. Underneath, there are tabs for 'Tables', 'Views' (which is underlined in blue), 'Schemas', and 'Stages'. A row of buttons includes '+ Create...', 'Drop...', and 'Transfer Ownership'. The main area displays a table with the following data:

| View Name              | Schema        |
|------------------------|---------------|
| FUNCTIONS              | ACCOUNT_USAGE |
| LOGIN_HISTORY          | ACCOUNT_USAGE |
| LOAD_HISTORY           | ACCOUNT_USAGE |
| COPY_HISTORY           | ACCOUNT_USAGE |
| WAREHOUSE_LOAD_HISTORY | ACCOUNT_USAGE |

- Contains views that display object metadata and usage metrics
- Generally mirrors the corresponding views and table functions in INFORMATION\_SCHEMA, with some differences



# READER\_ACCOUNT\_USAGE

The screenshot shows the Snowflake web interface. At the top, there's a navigation bar with the Snowflake logo, followed by tabs for Databases, Shares, and Data Exchange. Below this, the path 'Databases > SNOWFLAKE' is shown. A sub-navigation bar includes 'Tables', 'Views' (which is underlined in blue), 'Schemas', 'Stages', and 'File F'. Under 'Views', there are buttons for 'Create...', 'Drop...', and 'Transfer Ownership'. The main area displays a table with the following data:

| View Name                 | Schema               |
|---------------------------|----------------------|
| LOGIN_HISTORY             | READER_ACCOUNT_USAGE |
| QUERY_HISTORY             | READER_ACCOUNT_USAGE |
| RESOURCE_MONITORS         | READER_ACCOUNT_USAGE |
| STORAGE_USAGE             | READER_ACCOUNT_USAGE |
| WAREHOUSE_METERING_HIS... | READER_ACCOUNT_USAGE |

- Contains views that apply to reader accounts
- Small subset of views in ACCOUNT\_USAGE

# INFORMATION\_SCHEMA

The screenshot shows the Snowflake web interface. At the top, there is a navigation bar with the Snowflake logo, followed by three icons: Databases, Shares, and Data Exchange. Below the navigation bar, the path 'Databases > SNOWFLAKE' is displayed. Underneath this, there is a horizontal menu with tabs: Tables, Views (which is underlined), Schemas, Stages, and File Format. Below the menu, there are three buttons: '+ Create...', 'Drop...', and 'Transfer Ownership'. The main content area displays a table with the following data:

| View Name        | Schema ▾           |
|------------------|--------------------|
| APPLICABLE_ROLES | INFORMATION_SCHEMA |
| COLUMNS          | INFORMATION_SCHEMA |
| DATABASES        | INFORMATION_SCHEMA |
| ENABLED_ROLES    | INFORMATION_SCHEMA |
| EXTERNAL_TABLES  | INFORMATION_SCHEMA |

- Schema automatically created in all databases
- Does not serve a purpose in shared databases, and can be disregarded

# ACCOUNT\_USAGE VS. INFORMATION\_SCHEMA

| ACCOUNT_USAGE                                     | INFORMATION_SCHEMA<br>(IN INDIVIDUAL DATABASES)                 |
|---------------------------------------------------|-----------------------------------------------------------------|
| Includes dropped objects                          | Does not include dropped objects                                |
| 45 minutes to 3 hours latency<br>(varies by view) | No latency                                                      |
| Data retained for 1 year                          | Data retained up to 6 months<br>(varies by view/table function) |



# ACCOUNT\_USAGE EXAMPLES

Login History for a Specific User Over the Last Week:

```
SELECT event_id, event_timestamp, event_type, user_name, error_code, error_message
FROM SNOWFLAKE.ACCOUNT_USAGE.LOGIN_HISTORY
WHERE user_name = 'Anne_Jordan'
 AND EVENT_TIMESTAMP >= DATEADD('DAYS', -8, CURRENT_DATE())
 AND EVENT_TIMESTAMP < CURRENT_DATE()
ORDER BY EVENT_TIMESTAMP;
```

| EVENT_ID        | EVENT_TIMESTAMP               | EVENT_TYPE | USER_NAME   | ERROR_CODE | ERROR_MESSAGE    |
|-----------------|-------------------------------|------------|-------------|------------|------------------|
| 124077310283306 | 2019-09-05 10:58:39.777 -0700 | LOGIN      | ANNE_JORDAN | 390100     | INCORRECT_USE... |
| 124077310289407 | 2019-09-12 08:21:43.103 -0700 | LOGIN      | ANNE_JORDAN | NULL       | NULL             |



# ACCOUNT\_USAGE EXAMPLES

Query History for a Specific Warehouse Over the Last Month:

```
SELECT *
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE WAREHOUSE_NAME = 'DEMO_WH'
 AND START_TIME >= DATEADD('DAYS', -31, CURRENT_DATE())
 AND START_TIME < CURRENT_DATE()
ORDER BY START_TIME;
```

| QUERY_ID      | QUERY_TEXT           | DATABASE_ID | DATABASE_NAME | SCHEMA_ID |
|---------------|----------------------|-------------|---------------|-----------|
| 018eb0f9-0... | SHOW WAREHOUSES;     | NULL        | NULL          | NULL      |
| 018eb0fa-0... | GRANT CREATE TABL... | 239         | FINANCE       | NULL      |



# Resource Management & Monitoring

## Exercise 13.1: Monitor Billing & Usage Information

40 minutes

### Tasks:

- View warehouse usage
- View billing and usage
- Monitor storage
- Query INFORMATION\_SCHEMA and ACCOUNT\_USAGE



## Module 14

# Preview Features

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# MODULE AGENDA

- Preview Features Overview
- Specific Preview Features
- Other Roadmap Features

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# Preview Features Overview

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# WHAT IS A PREVIEW FEATURE?

- Implemented and tested in Snowflake
- Full usability and corner-case handling may not be complete
- Use of preview features **are not warranted** against defects that may produce undesired results
  - Behavior may change between Preview and Release
- Not recommended for production use



# BEST PRACTICES FOR PREVIEW FEATURES

- Do not use on production data
- Do not use on production data
- Do not use on production data
- Contact support for additional information if you really want to use it on production data



# PREVIEW AVAILABILITY

- Private Preview / On Request
  - Typically invite-only at first
  - Generally in the early stages of preview
  - Work with your account team if access is required
- Public Preview / Open
  - Enabled by default for all accounts
  - Free to use, though still carry some risk and are not for full production use
- How do I know what's available in public preview?

<https://docs.snowflake.net/manuals/release-notes/preview-features.html>



# FEATURES CURRENTLY IN PREVIEW

| Feature                                               | Availability  |
|-------------------------------------------------------|---------------|
| Database replication and failover                     | Open          |
| Unloading data to Parquet format with LZO compression | Open          |
| Tri-Secret Secure (customer-managed keys)             | On Request    |
| Streams                                               | Open          |
| Tasks                                                 | Open          |
| External tables                                       | Open          |
| Hive metastore integration                            | Open          |
| MATCH_BY_COLUMN_NAME                                  | By Invitation |



# FEATURES CURRENTLY IN PREVIEW

| Feature                                                               | Availability |
|-----------------------------------------------------------------------|--------------|
| Google Cloud Storage external stages for loading/unloading data       | Open         |
| Credential-less access to external stages (AWS and Google)            | Open         |
| Auto-ingest                                                           | Open         |
| Snowflake connector for Kafka                                         | Open         |
| SYSTEM\$WHITELIST                                                     | Open         |
| Support for AWS IAM policy credential to access external stages in S3 | Open         |
| Support for XML                                                       | Open         |



# Specific Preview Features

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# XML SUPPORT



- XML support is in public preview
- Available functions:
  - CHECK\_XML
  - PARSE\_XML
  - TO\_XML
  - XMLGET

# MATCH\_BY\_COLUMN\_NAME

- In Private Preview
- Load semi-structured data without using a COPY transformation
- Available for:
  - JSON
  - Avro
  - ORC
  - Parquet
- Column represented in data must **exactly match** column name



# MATCH\_BY\_COLUMN\_NAME

## 1. Create target table

```
CREATE TABLE customers ("name" VARIANT, "address" VARCHAR,
"company" VARCHAR, "email" VARCHAR, "phone" VARCHAR);
```

## 2. Load the data

```
COPY INTO customers
FROM @my_stage/customers.json
FILE_FORMAT = (TYPE = JSON STRIP_OUTER_ARRAY = TRUE)
MATCH_BY_COLUMN_NAME = CASE_INSENSITIVE;
```



# TASKS

- Scheduled execution of SQL statements
- Subset of cron syntax
- Currently run in user-defined warehouses
- Will eventually run serverless
  - Using Snowflake-provided compute



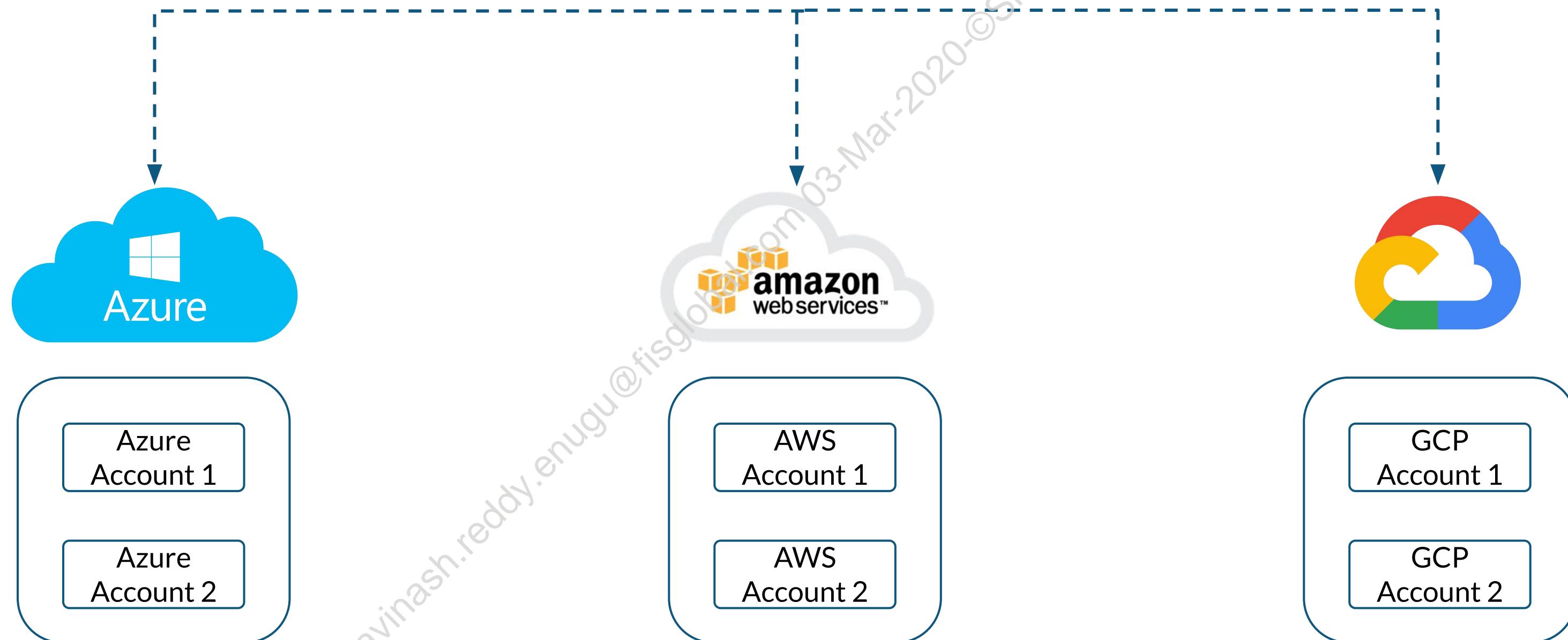
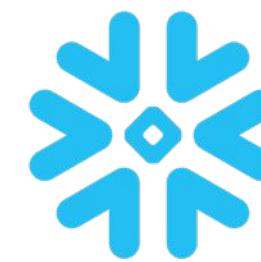
# SOME USE CASES FOR TASKS

- Generate periodic reports
  - Insert or merge rows into a report table
- Combine with table streams for continuous ELT workflows for Change Data Capture (CDC)



# GLOBAL SNOWFLAKE

Snowflake Organization



# Class Exercise

## Complete the Course Survey

**20 minutes**

- Take 20 minutes before we move on to the last section to fill out your course surveys



# Recap

avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy



# Review Questions

What are examples of some Snowflake objects?

Database

Table

User

Stored Procedure

Role

All of the Above



# Review Questions

What are examples of some Snowflake objects?

Database

Table

User

Stored Procedure

Role

All of the Above



## Review Questions

In which Snowflake layer does a virtual warehouse reside?

The Data Layer

The Compute Layer

The Services Layer



# Review Questions

In which Snowflake layer does a virtual warehouse reside?

The Data Layer

**The Compute Layer**

The Services Layer



# Review Questions

In what increment are you billed for compute usage?

By the second, with a one-minute minimum

By the second, with a five-minute minimum

By the minute, with a five-minute minimum

By the query

By the hour



# Review Questions

In what increment are you billed for compute usage?

By the second, with a one-minute minimum

By the second, with a five-minute minimum

By the minute, with a five-minute minimum

By the query

By the hour



# Review Questions

In what increment are you billed for storage?

By the Megabyte

By the Gigabyte

By the Terabyte

By the Yottabyte



avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

# Review Questions

In what increment are you billed for storage?

By the Megabyte

By the Gigabyte

By the Terabyte

By the Petabyte



avinash.reddy.enugu@fisglobal.com 03-Mar-2020 ©Snowflake 2020-do-not-copy

# Review Questions

## What is a micro-partition?

How the files need to be broken up before loading them into Snowflake

How Snowflake breaks up the files when unloading data

Contiguous units of storage in Snowflake

How Semi-Structured data is viewed in Snowflake



# Review Questions

## What is a micro-partition?

How the files need to be broken up before loading them into Snowflake

How Snowflake breaks up the files when unloading data

Contiguous units of storage in Snowflake

How Semi-Structured data is viewed in Snowflake



## Review Questions

Which of the following is NOT a Snowflake architecture layer

Storage

Compute

Hardware

Services



# Review Questions

Which of the following is NOT a Snowflake architecture layer

Storage

Compute

Hardware

Services



# Review Questions

What are some of the things that the cloud services layer is responsible for?

Security

Optimizer

Metadata

Writing well-crafted SQL statements



# Review Questions

What are some of the things that the cloud services layer is responsible for?

Security

Optimizer

Metadata

Writing well-crafted SQL statements



# Review Questions

For what reason would you scale a virtual warehouse up?

More complex queries

When you need more storage for data

Concurrency

If you need more room in a worksheet



# Review Questions

For what reason would you scale a virtual warehouse up?

More complex queries

When you need more storage for data

Concurrency

If you need more room in a worksheet



# Review Questions

For what reason would you scale a virtual warehouse out?

To group your data differently

Concurrency

More complex queries



# Review Questions

For what reason would you scale a virtual warehouse out?

To group your data differently

Concurrency

More complex queries



# Review Questions

How many clusters are in the standard MEDIUM warehouse?

4

1

128

42



# Review Questions

How many **clusters** are in the standard MEDIUM warehouse?

4

1

128

42





snowflake®

THANK YOU



avinash.reddy.enugu@fisglobal.com  
03-Mar-2020 ©Snowflake 2020-do-not-copy