Snowflake Builders Blog: …

Snowflake Row-Level Security Overview

Rajiv Gupta Follow

6 min read · Aug 27, 2021

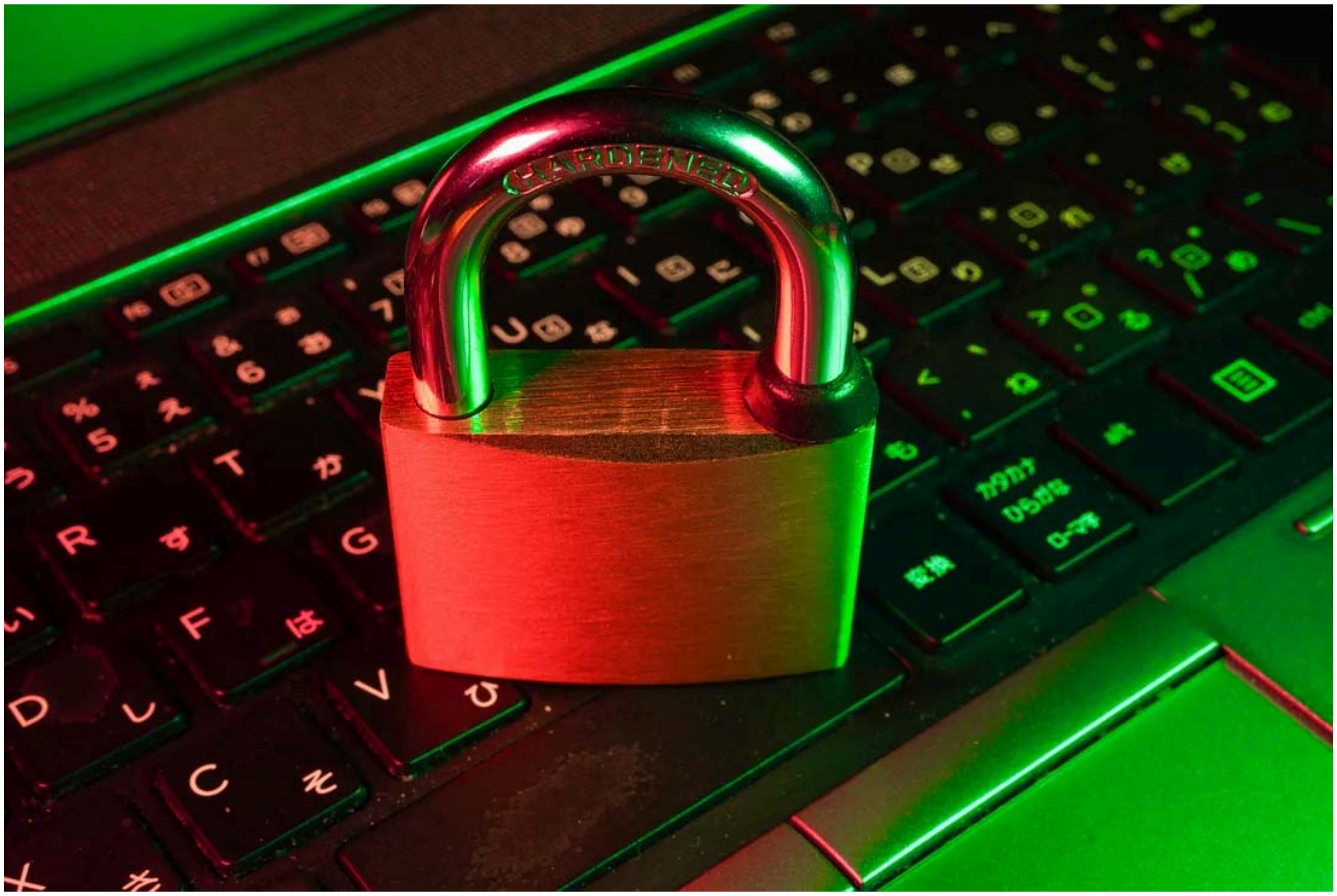


Photo by FLY:D on Unsplash

In this blog, we are going to discuss **Snowflake Row Level Security**. This feature falls under “**Protect your data**” category. This feature is totally based on what data needs to visible for a particular person or group of person.

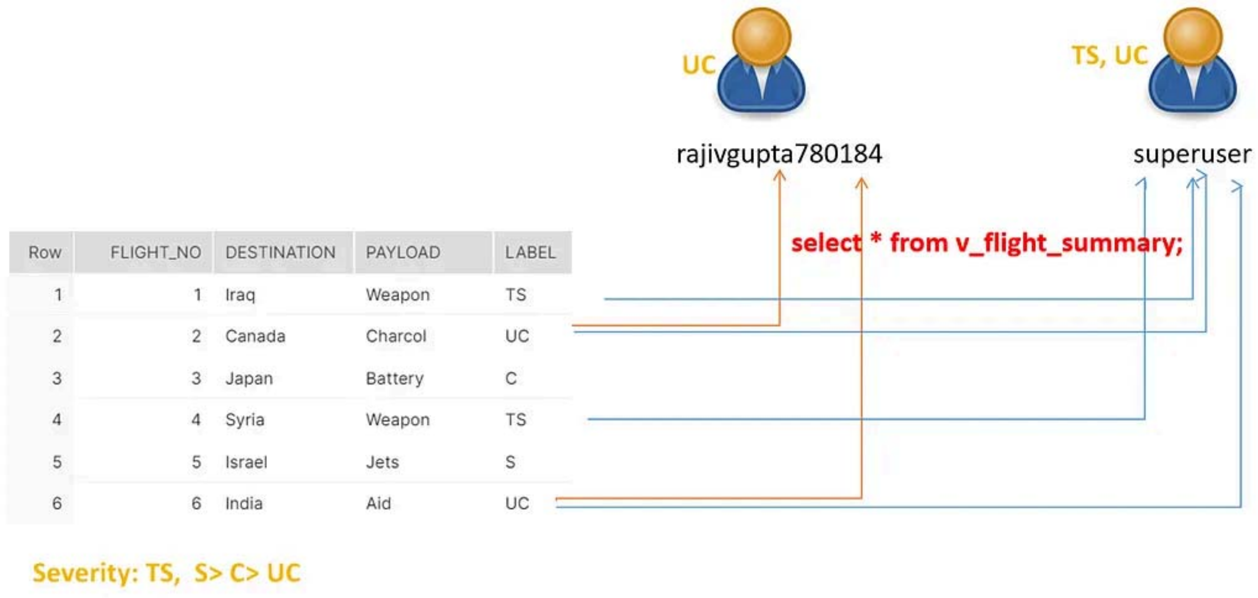
· · ·

What is Row Level Security?

Row-level security, or row-based security, is a data access control concept in which access to data or row in a table is limited according to certain restrictions, and various users, groups or roles may have different permissions on certain rows, based on identities within the rows. Given the right conditions, row-based security can be an important form of data protection control.

Let’s understand Row Level Security with an example of fictitious Flight Payload Summary table.

Give below the diagram, you can see that Flight Payload Summary tables has 6 rows with different labels tagged to them like TS, S, C, UC. Also, if you look at severity, **‘TS’** is a very confidential label and can’t be accessed by anybody. Whereas, **‘S’** is higher(after TS) on role hierarchy followed by **‘C’** and then **‘UC’** being the lowest priority.



Row	FLIGHT_NO	DESTINATION	PAYLOAD	LABEL
1	1	Iraq	Weapon	TS
2	2	Canada	Charcol	UC
3	3	Japan	Battery	C
4	4	Syria	Weapon	TS
5	5	Israel	Jets	S
6	6	India	Aid	UC

In the above image, you can see that we had 2 user, i.e. **rajivgupta780184** is the default user and newly created user viz. **superuser**.

Now we want to implement the row level data security in such a way that **superuser should be able to see all data that has labelled TS & UC** and

rajivgupta780184 user should only view data having label UC being the least privilege user.

So Final goal is when you are connected with **rajivgupta780184** you should see **2 rows** with UC label and when you are connected with **superuser** you should see **4 rows** with label UC & TS.

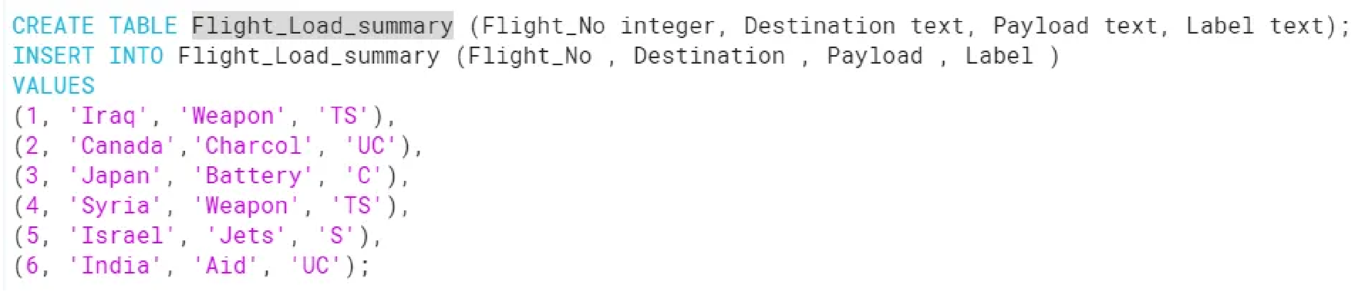
Above, can be achieved by implementing row level security in Snowflake. Row level security helps organization achieve data access control and allow only privilege user to view the data and disregards the other request from same source objects without changing data.

. . .

What are the challenges with Row Level Security ?

Row-based security includes some inherent challenges, and there is no direct solution that solves everything. Rather, there are a series of decisions that need to be made according to the reasons for enforcing row-level security and the risk calculations:

To understand the example below, let’s create a sample table called Flight_Load_summary, holding fictitious flight payload data for different country with different payload:



```
CREATE TABLE Flight_Load_summary (Flight_No integer, Destination text, Payload text, Label text);
INSERT INTO Flight_Load_summary (Flight_No , Destination , Payload , Label )
VALUES
(1, 'Iraq', 'Weapon', 'TS'),
(2, 'Canada','Charcol', 'UC'),
(3, 'Japan', 'Battery', 'C'),
(4, 'Syria', 'Weapon', 'TS'),
(5, 'Israel', 'Jets', 'S'),
(6, 'India', 'Aid', 'UC');
```

Now, that you see above table and its data, you can easily do explicitly row filtering based on Label column. Say if a particular user was supposed to see data based on their position,role & severity tagged to their job. User can do that by putting explicit filter like below:

> *Select * from Flight_Load_summary where Label='UC';*

This process may get a bit more complicated, as users may have access to several labels. In this case, users should be able to filter by any label they have access to.

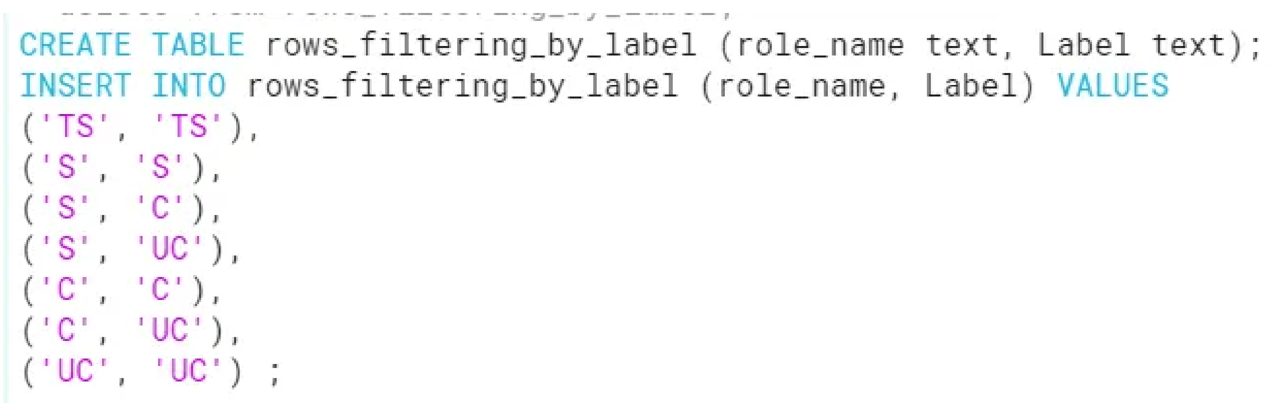
Implementing an **implicit row-level security** system limits the results a user receives from the database according to certain access control settings. In this case, the user will query the table without placing any filtering over the Label column, but the filtering will then be added, either by an abstraction layer of a secure view or by query rewriting.

As per our example, we can create a table containing the row-level security definitions per role as well as a secure view, abstracting data access to the Flight_Load_summary table.

. . .

How we can implement Row Level Security for single condition?

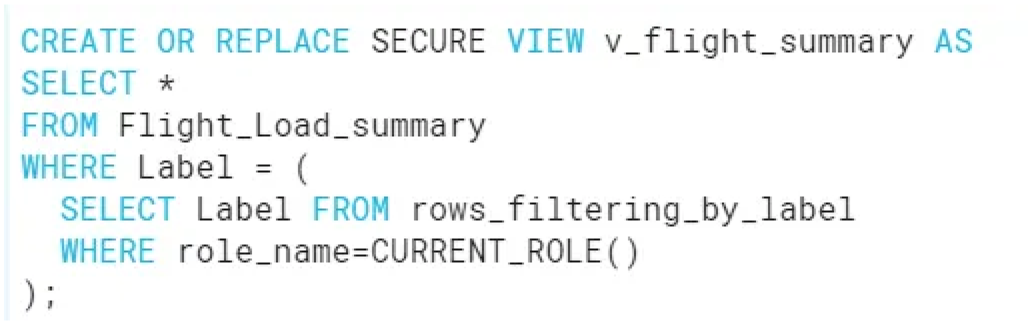
Step1: Create a row level security configuration table which holds access control settings based on severity.



```
CREATE TABLE rows_filtering_by_label (role_name text, Label text);
INSERT INTO rows_filtering_by_label (role_name, Label) VALUES
('TS', 'TS'),
('S', 'S'),
('S', 'C'),
('S', 'UC'),
('C', 'C'),
('C', 'UC'),
('UC', 'UC') ;
```

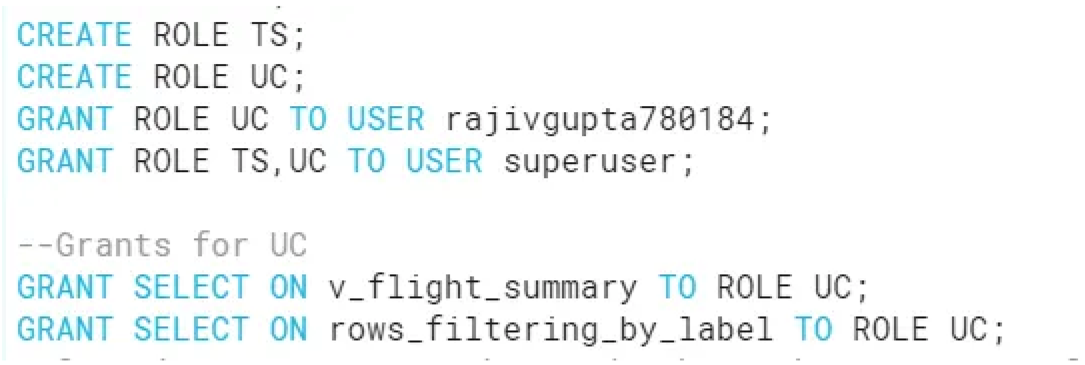
As you can see in this sample, we are creating a table that will contain the mapping of roles to label and implement the control settings based on severity. As per above configuration, TS role will only see TS data, as its restricted details. Whereas, **'S'** is higher(after TS) on role hierarchy followed by **'C'** and then **'UC'** being the lowest priority.

Step 2: Let's now create abstract view which read data from flight load summary table in conjunction with control settings table & Current user role(using the CURRENT_ROLE() function).

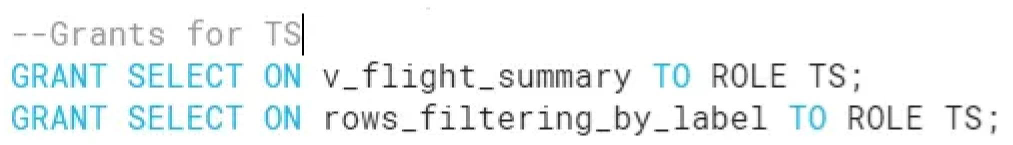


```
CREATE OR REPLACE SECURE VIEW v_flight_summary AS
SELECT *
FROM Flight_Load_summary
WHERE Label = (
  SELECT Label FROM rows_filtering_by_label
  WHERE role_name=CURRENT_ROLE()
);
```

Step 3: Grant access to above created object to newly created role.



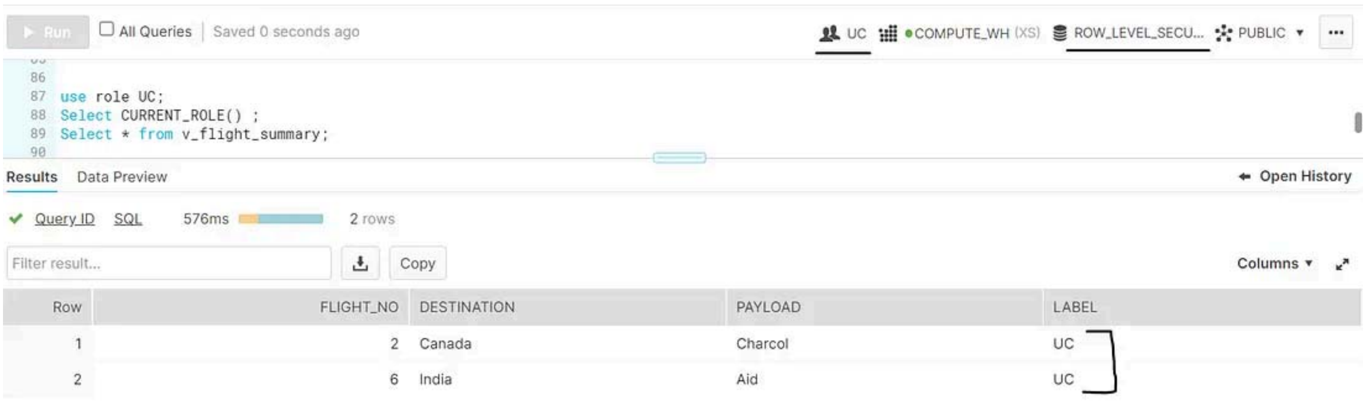
```
CREATE ROLE TS;
CREATE ROLE UC;
GRANT ROLE UC TO USER rajivgupta780184;
GRANT ROLE TS,UC TO USER superuser;

--Grants for UC
GRANT SELECT ON v_flight_summary TO ROLE UC;
GRANT SELECT ON rows_filtering_by_label TO ROLE UC;
```



```
--Grants for TS
GRANT SELECT ON v_flight_summary TO ROLE TS;
GRANT SELECT ON rows_filtering_by_label TO ROLE TS;
```

In above steps, we are creating 2 roles TS & UC and assigning the select privilege on view & control settings table to both the role. Here we are not giving any grants on base table to any of role.

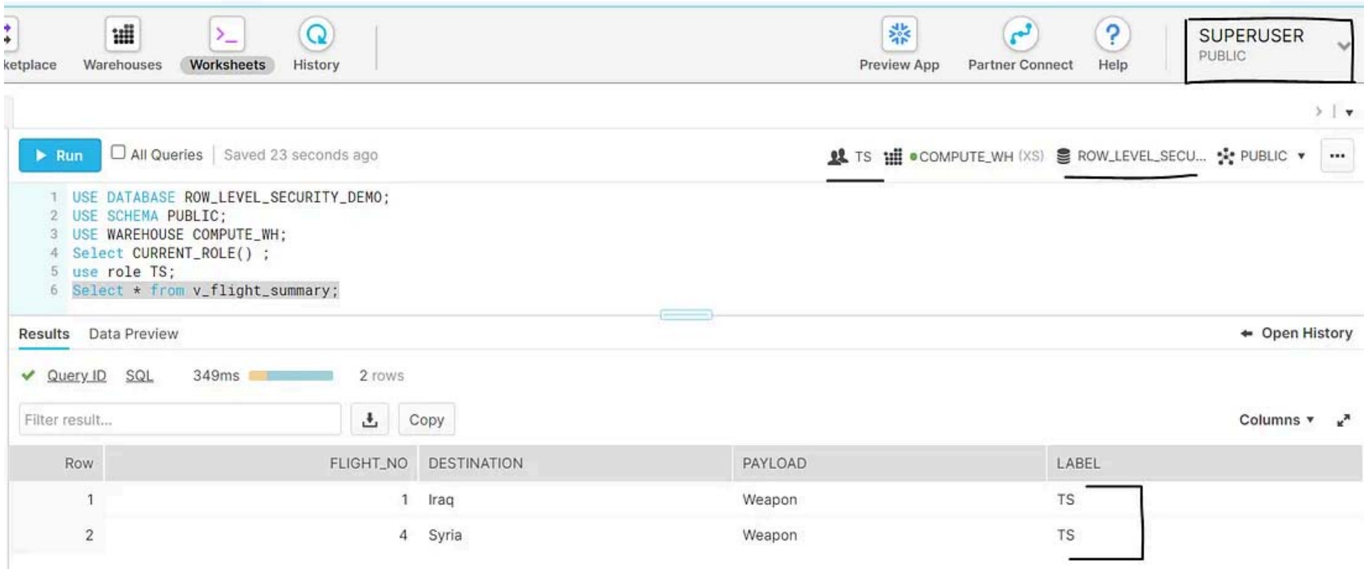
Now, when we use the role “UC”, we only obtain results for label UC:



```
use role UC;
Select CURRENT_ROLE() ;
Select * from v_flight_summary;
```

Row	FLIGHT_NO	DESTINATION	PAYLOAD	LABEL
1	2	Canada	Charcol	UC
2	6	India	Aid	UC

Although we had assigned multiple role to superuser(in our case TS & UC) but still supper user is limited to limited data view based on current role.



```
USE DATABASE ROW_LEVEL_SECURITY_DEMO;
USE SCHEMA PUBLIC;
USE WAREHOUSE COMPUTE_WH;
Select CURRENT_ROLE() ;
use role TS;
Select * from v_flight_summary;
```

Query ID SQL 349ms 2 rows

Row	FLIGHT_NO	DESTINATION	PAYLOAD	LABEL
1	1	Iraq	Weapon	TS
2	4	Syria	Weapon	TS

The above scenario is good if we want to see data based on single condition where user has 1 role assigned. What if we have multiple role assigned, and we are privilege to see more than 1 label at our designation.

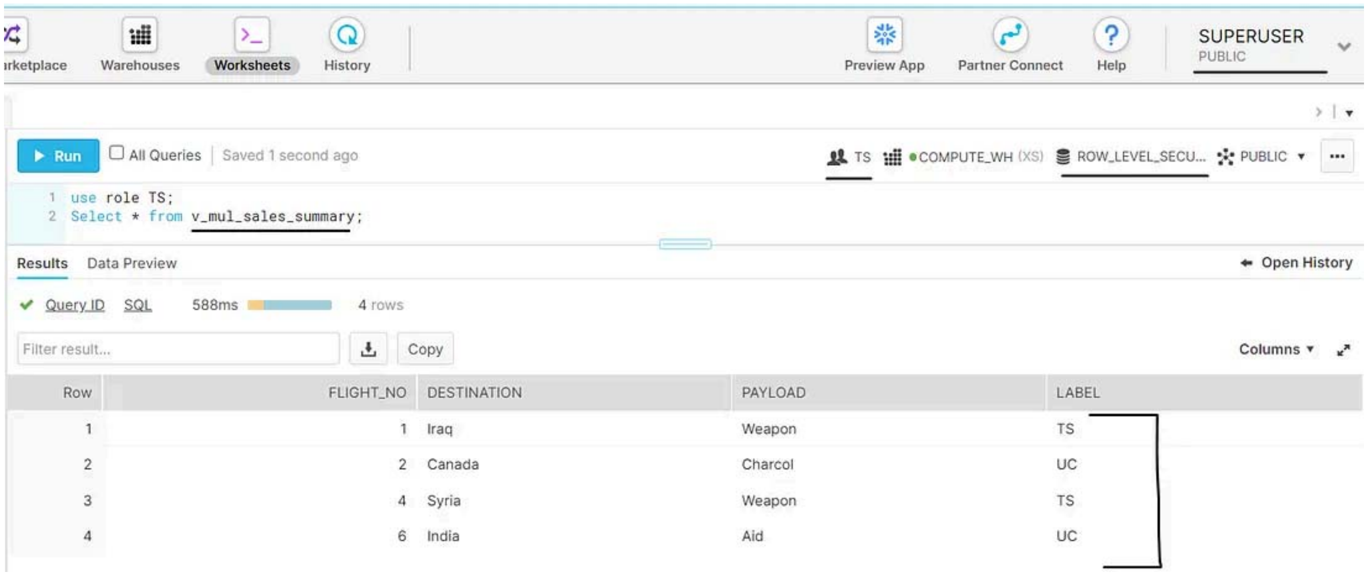
. . .

How we can implement Row Level Security for multiple condition?

If the requirement is to allow access based on multiple roles (in our case each role adds one or more “label” which we will be able to view), we can do so by using the CURRENT_AVAILABLE_ROLES() function, which (as its name implies) returns a JSON array of all available roles to the current user. Let’s modify our code accordingly:

```
CREATE OR REPLACE SECURE VIEW v_mul_sales_summary AS
SELECT *
FROM Flight_Load_summary
WHERE Label IN (
  SELECT label FROM rows_filtering_by_label
  WHERE role_name IN (SELECT value FROM TABLE(flatten(input => parse_json(CURRENT_AVAILABLE_ROLES()))))
);
```

I had modified the view name purposely to keep two different versions of the object. As you can see that, instead of checking for a match for our current role, we look for any matches for any roles the user is granted. In addition, this code allows multiple roles to be tested against each other.



```
use role TS;
Select * from v_mul_sales_summary;
```

Row	FLIGHT_NO	DESTINATION	PAYLOAD	LABEL
1	1	Iraq	Weapon	TS
2	2	Canada	Charcol	UC
3	4	Syria	Weapon	TS
4	6	India	Aid	UC

. . .

Demo will be fascinating….

. . .

Key Observation:

There are quite few limitations to this approach.

- The above setup assumes one "label" or "filter" per role, but this is not always the case. Sometimes, a certain role may have access to multiple different "labels" (or other filters). Similarly, sometimes the filtering is done over multiple different columns or may introduce other complexities this method does not account for.
- When using this method, it is important to note that the users have select permission over the mapping table, meaning that they have visibility into the control settings table.

- Managing many tables with numerous roles may be challenging in terms of both architecture and maintenance.
- In environments with multiple data stores, this process may be challenging to manage at a large scale.
- Considering the above challenges, Snowflake has come-up with new feature **"Data Governance: Row Access Policy"**. In my next blog, I will show how we can cover the above limitation.

. . .

Hope this blog help you to get insight on Snowflake Row Level Security. If you are interested in learning more details about Row Level Security, you can refer to Snowflake documentation. Feel free to ask a question in the comment section if you have any doubts regarding this. Give a clap if you like the blog. Stay connected to see many more such cool stuff. Thanks for your support.

You Can Find me:

Follow me on Medium: https://rajivgupta780184.medium.com/

Follow me on Twitter: https://twitter.com/RAJIVGUPTA780

Connect with me in LinkedIn: https://www.linkedin.com/in/rajiv-gupta-618b0228/

Subscribe to my YouTube Channel:
https://www.youtube.com/channel/UC8Fwkdf2d6-hnNvcrzovktg

#Keep learning #Keep Sharing #Everyday Learning.

References:-

- https://www.snowflake.com/

Snowflake · Row Level Security · Data Governance · Security · Rajiv Gupta

Published in Snowflake Builders Blog: Data Engineers, App Developers, AI, & Data Science

Following

10K followers · Last published 15 hours ago

Best practices, tips & tricks from Snowflake experts and community

Written by Rajiv Gupta

Follow

1.6K followers · 18 following

Snowflake Data Super Hero, Director Of Technology at Kipi.ai , Snowflake SME

Responses (1)

Eng Omar Essam

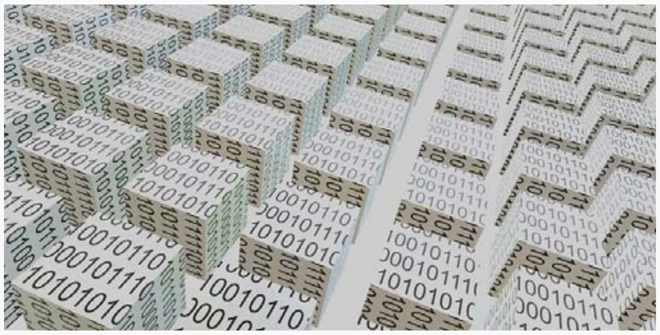
What are your thoughts?

Nidhi lalwani
Aug 28, 2021

How to achieve column level security in snowflake?

Reply

More from Rajiv Gupta and Snowflake Builders Blog: Data Engineers, App Developers, AI, & Data Science



In FAUN.dev() 🐾 by Rajiv Gupta

Automating Data Governance with Snowflake's Sensitive Data...

AGENTIC BUSINESS INSIGHT
Agent Apps Snowflake Intelligence
Agent APIs Cortex Agents
STATE OF THE ART RETRIEVAL
Unstructured Data Retrieval Cortex Search
Structured Data Retrieval Cortex Analyst
SCALABLE AI PROCESSING
Multimodal AI-powered SQL Cortex AISQL
Document Processing Document AI, Parse, Embed
MODELS
OpenAI
ANTHROPIC
Meta
MISTRAL AI
deepseek
snowflake
GOVERNANCE
RBAC
Guardrails
Evaluations
Monitoring
AI Gateway
DATA
DATA INGESTION
Fully managed extraction, load, preprocess, and
ORACLE

In Snowflake Builders Blog: Data Engine... by Ume...

Build Snowflake Cost Savings and Performance Agent in 5 minutes

In the age of data democratization, balancing accessibility with privacy is no longer option…

Sep 29 5

Snowflake Cortex Agent, building agentic AI solutions is no longer about complexity—it's…

Sep 30 44 3

In Snowflake Builders Blog: Data Enginee… by Sai…

Agent Instruction Best Practices for Snowflake Intelligence

Prototyping AI agents is easy. However, successfully launching reliable agents to…

Sep 25 54 3

In Snowflake Builders Blog: Data Engine… by Raji…

Using Contacts in Snowflake: A Strategic Guide to Schema-Level…

In modern data ecosystems, clarity in ownership and support is critical. Snowflake'…

Oct 10 2

See all from Rajiv Gupta

See all from Snowflake Builders Blog: Data Engineers, App Developers, AI, & Data Science

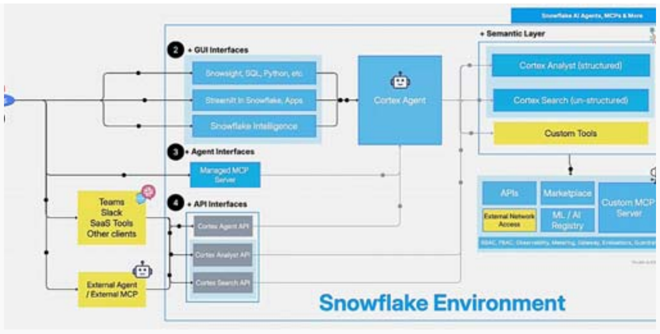
Recommended from Medium

John Ryan

Snowflake Gen-2 Warehouses: Faster Performance or Just Highe…

Introduction

Oct 1 6

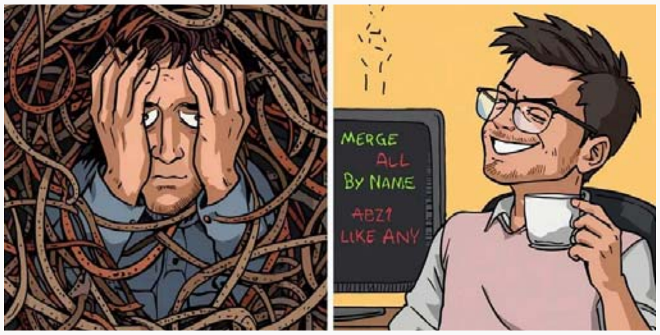


In Fru.dev by Fru

5 Strategic Pillars to Understanding Snowflake's AI…

A Complete Architecture Overview

Sep 16 23



Vishal Kaushal

Top 10 Snowflake SQL Functions Every Data Professional Should…

Are you spending too much time writing and maintaining long SQL queries in Snowflake?…

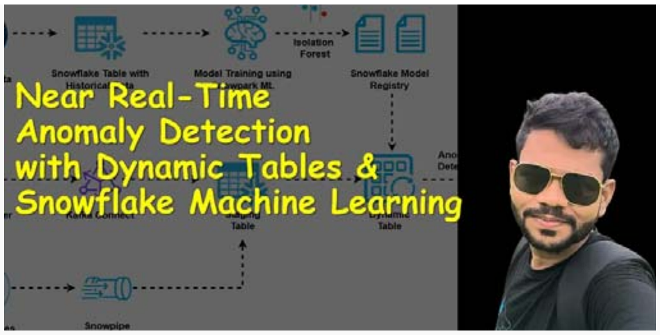
Oct 16 34 1

Karthik Goutam

Enterprise Data Pipeline: Jira Atlassian Cloud to Snowflake ETL…

Abstract

Jun 9 4


Near Real-Time Anomaly Detection with Dynamic Tables & Snowflake Machine Learning
Snowflake Table with Historical Data
Model Training using Snowpark ML
Isolation Forest
Snowflake Model Registry
Kafka Connect
Staging Table
Dynamic Table
Snowpipe

In Data Engineer Things by Satadru

Build a real-time Anomaly Detection pipeline using Dynamic…

Detecting anomalies in real time from high-throughput streams is key for informing on…

Oct 4 3

Kamalakannan R

Snowflake

— use role sysadmin; —use warehouse compute_wh; —create or replace database…

Aug 20

See more recommendations