Quickly Visualize Snowflake’s Roles, Grants, and Privileges

Snowflake’s Control Framework

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

All objects in Snowflake are secured; in order to access these objects the user needs to have:

- The required Role
- And appropriate action privilege (e.g. Usage, Modify, Monitor Usage, etc.)

By default Snowflake offers the following default roles:

- AccountAdmin
- SecurityAdmin
- SysAdmin
- Public

These are a good starting point, but won’t be sufficient for most implementations, so we end up defining custom roles. Snowflake’s recommendation is to create a hierarchy of custom roles with the top-most custom role assigned to the system role SYSADMIN. That way the system administrators will be able to manage all warehouses and databases while maintaining management of user and roles restricted to users granted the SECURITYADMIN or ACCOUNTADMIN roles.

Role Hierarchy

A role hierarchy is created via the Grant statement, for example:

```
GRANT ROLE DEVELOPER TO ROLE DBA;
```

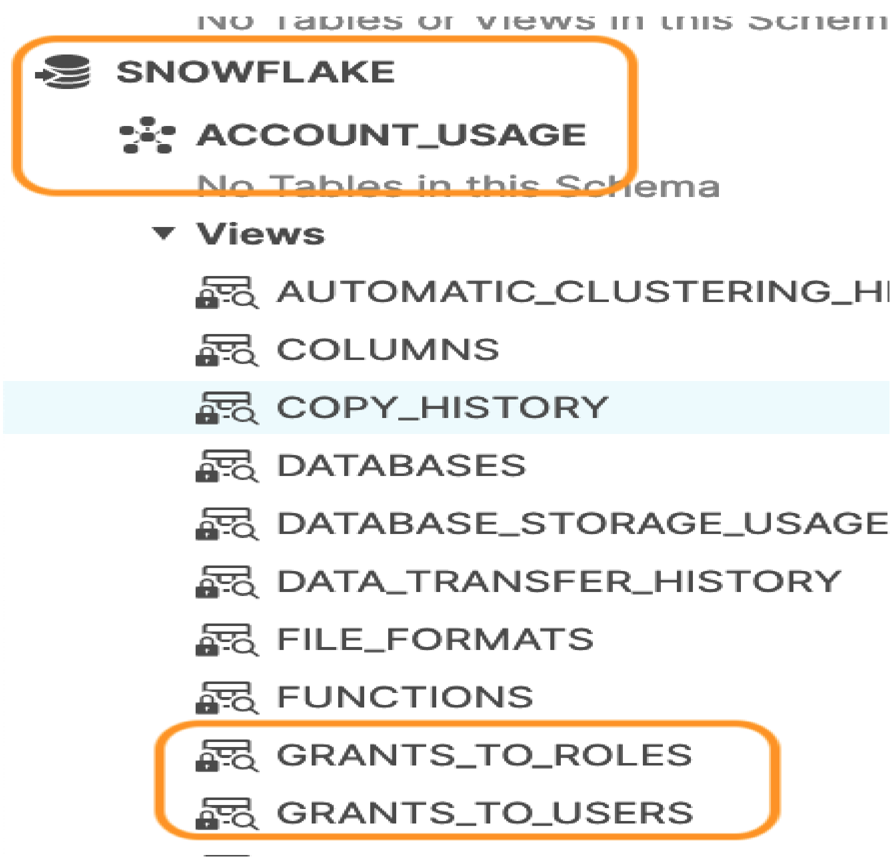
Information of the roles and privilege can be obtained via:

- Show Grants
- Show Roles

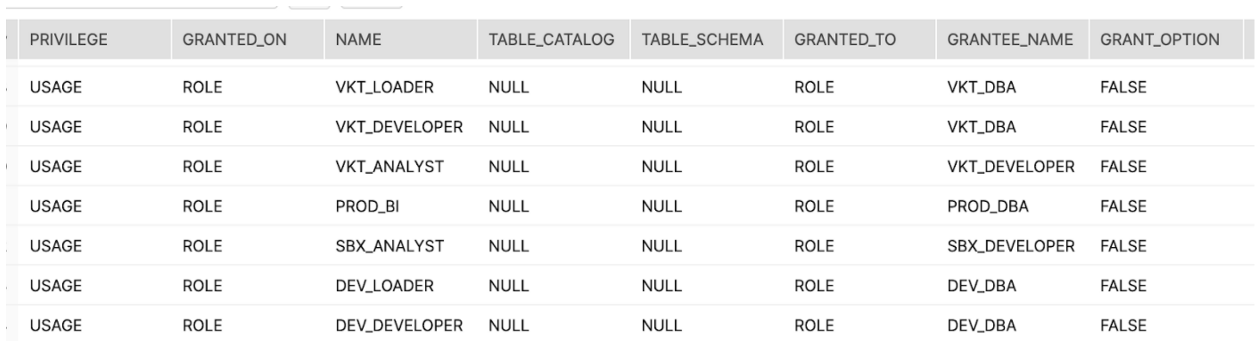
As mentioned, things have changed in the Snowflake world since I last talked about capturing users, roles, and grants into a table in a previous post; Snowflake now provides views which deliver these functions:

Views presenting relations between roles, grants, and users:

- GRANTS TO ROLES
- GRANTS TO USERS



The data is now readily queryable and results are available as table results:



PRIVILEGE	GRANTED_ON	NAME	TABLE_CATALOG	TABLE_SCHEMA	GRANTED_TO	GRANTEE_NAME	GRANT_OPTION
USAGE	ROLE	VKT_LOADER	NULL	NULL	ROLE	VKT_DBA	FALSE
USAGE	ROLE	VKT_DEVELOPER	NULL	NULL	ROLE	VKT_DBA	FALSE
USAGE	ROLE	VKT_ANALYST	NULL	NULL	ROLE	VKT_DEVELOPER	FALSE
USAGE	ROLE	PROD_BI	NULL	NULL	ROLE	PROD_DBA	FALSE
USAGE	ROLE	SBX_ANALYST	NULL	NULL	ROLE	SBX_DEVELOPER	FALSE
USAGE	ROLE	DEV_LOADER	NULL	NULL	ROLE	DEV_DBA	FALSE
USAGE	ROLE	DEV_DEVELOPER	NULL	NULL	ROLE	DEV_DBA	FALSE

Visualization of the Role Hierarchy

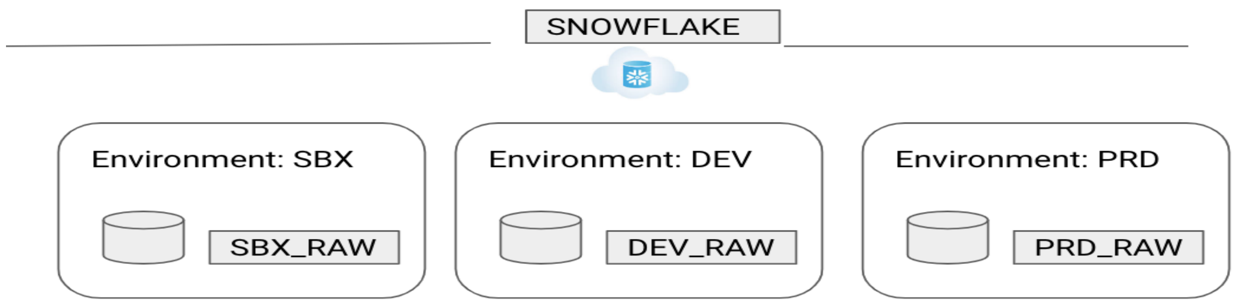
There is a natural hierarchy to the data; a quick way of understanding the data is to present it in the form of a Graph Network.

I am not a UX/UI expert, but I would prefer the graph network to clearly lay out the roles and objects as nodes and the privileges for those objects as lineage.

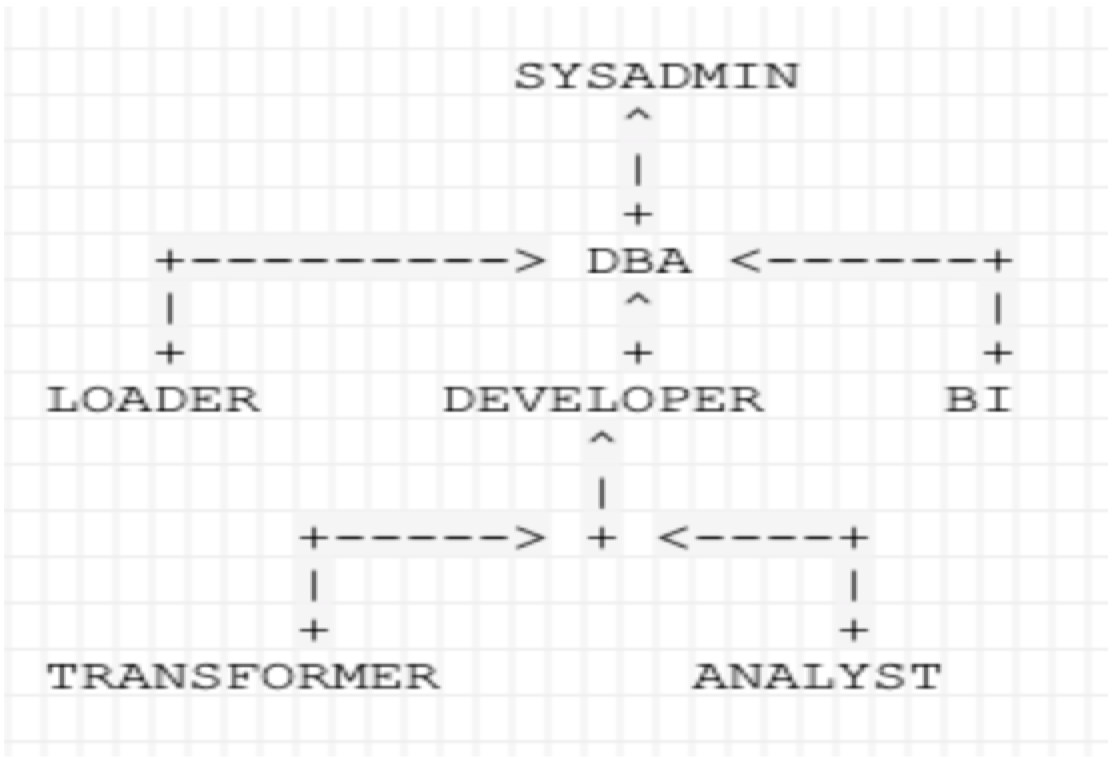
After working my way through Sanskey diagrams as well as Chord diagrams, I finally landed on using Dagre-d3 which is a D3-based renderer for Dagre. Dagre is a JavaScript library that makes it easy to layout directed graphs on the client-side.

My Sample Scenario

In the scenario below, I have a single Snowflake account with multiple environment configurations. Each environment (DEV, SBX, PRD) is prefixed in the name of the objects: role DEV_DBA, database SBX_RAW.



I defined the following custom role hierarchy:



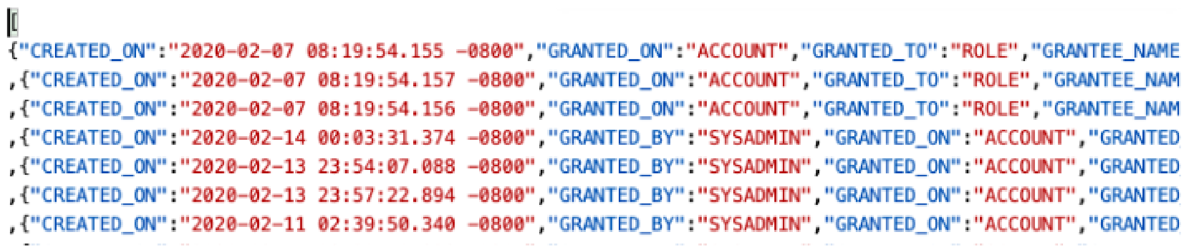
Other objects like database, warehouse, etc. also get defined. Roles are assigned to appropriate privileges.

Data extraction

As mentioned, the roles and grants are available via the GRANTS_TO_ROLE view. While there are many ways of extracting and presenting the data, I adopted a simple approach. For now, I've used the query below to get the roles as JSON documents and store them locally in a file: [data/grants_to_roles.json.]()

```
select to_json(OBJECT_CONSTRUCT(*))
from snowflake.account_usage.grants_to_roles
where deleted_on is null
order by privilege;
```

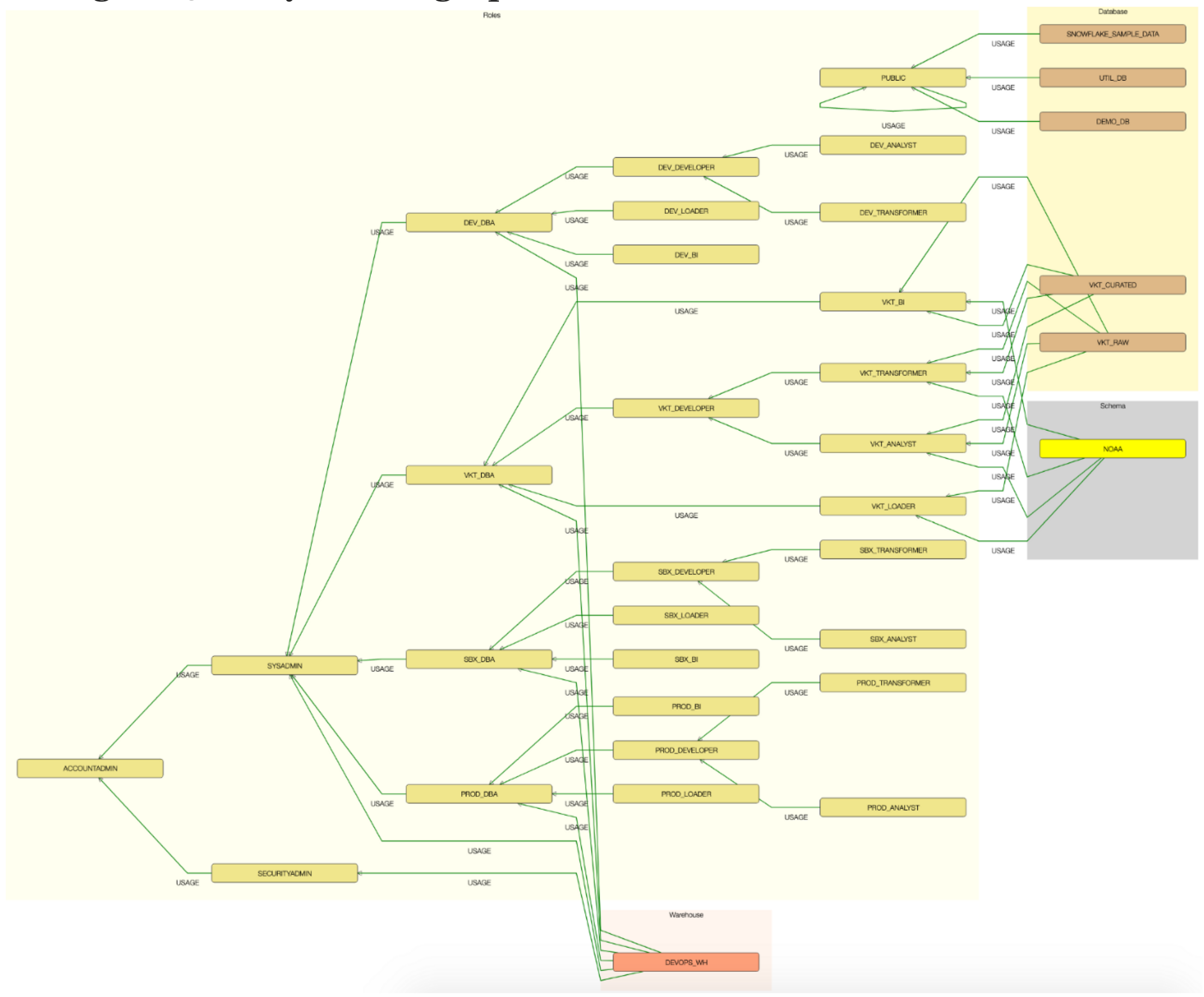
I quickly did a manual edit to set the rows into a JSON array:



```
[
{"CREATED_ON":"2020-02-07 08:19:54.155 -0800","GRANTED_ON":"ACCOUNT","GRANTED_TO":"ROLE","GRANTEE_NAME
,{"CREATED_ON":"2020-02-07 08:19:54.157 -0800","GRANTED_ON":"ACCOUNT","GRANTED_TO":"ROLE","GRANTEE_NAM
,{"CREATED_ON":"2020-02-07 08:19:54.156 -0800","GRANTED_ON":"ACCOUNT","GRANTED_TO":"ROLE","GRANTEE_NAM
,{"CREATED_ON":"2020-02-14 00:03:31.374 -0800","GRANTED_BY":"SYSADMIN","GRANTED_ON":"ACCOUNT","GRANTED
,{"CREATED_ON":"2020-02-13 23:54:07.088 -0800","GRANTED_BY":"SYSADMIN","GRANTED_ON":"ACCOUNT","GRANTED
,{"CREATED_ON":"2020-02-13 23:57:22.894 -0800","GRANTED_BY":"SYSADMIN","GRANTED_ON":"ACCOUNT","GRANTED
,{"CREATED_ON":"2020-02-11 02:39:50.340 -0800","GRANTED_BY":"SYSADMIN","GRANTED_ON":"ACCOUNT","GRANTED
```

Graphing with Dagre-d3

This is a prototype so I've used the default settings and configuration of Dagre-d3 to layout the graph.



The prototype achieves the following:

- Lays out the roles and their hierarchy in a tree.
- Maps privileges as edges so you can see how the role is connected to an object (warehouse, databases, schema).
- Filters out the ownership privilege because it caused too much noise in the graph.
- Groups the objects into a bounding box.

Going Beyond the Prototype

This demo is a simple prototype. Should you want to go beyond zoom-in and zoom-out responsiveness in the graph, you could enhance the code to adopt the following functions:

- When a specific object is chosen, highlight all the neighboring nodes
- Filter
- Tooltips
- Append with GRANTS_TO_USERS view dataset