Design of Automation RBAC Application

## **Change history**

|  |  |  |
| --- | --- | --- |
| **Author** | **Date** | **Change** |
| Ihor Karbovskyy | 15-May-19 | Initial version Alpha |
| Ihor Karbovskyy | 22-May-19 | Updated logic in SP to work with added Pivot table |
| Ihor Karbovskyy | 1-Aug-19 | Added Schema processing |
| Ihor Karbovskyy | 8-Aug-19 | Added Schema Access Manager |

## Purpose:

The outcome of the process is expected to create a script in a Snowflake that will create ROLES, GRANTS what will be referenced to Users structured in the Groups or MS ADFS Groups. Roles will/may have hierarchy levels, access permissions to the data, based on preset RBAC access matrix (HUB-RAM) discussed, designed and approved by customer.

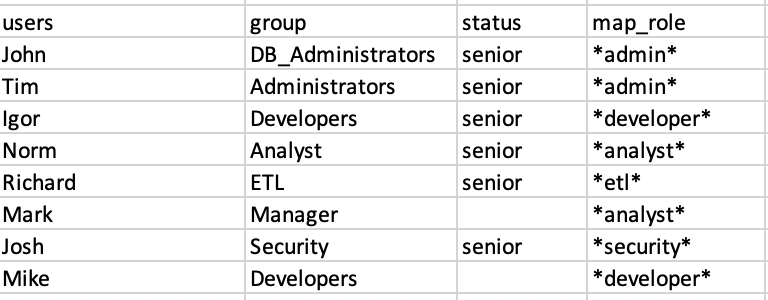
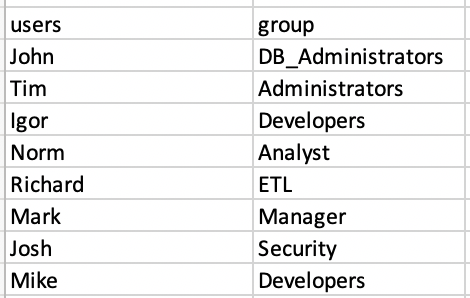
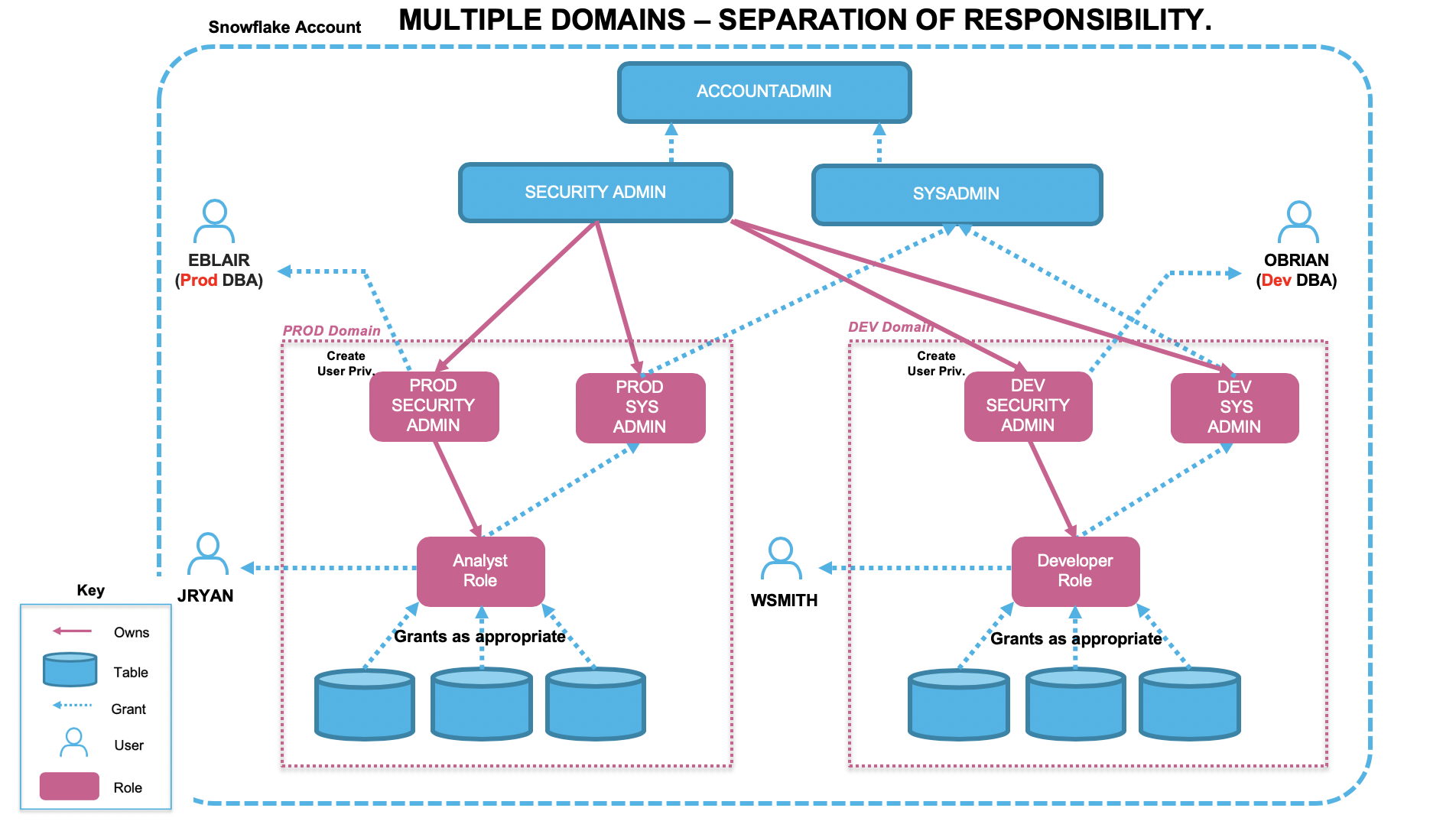
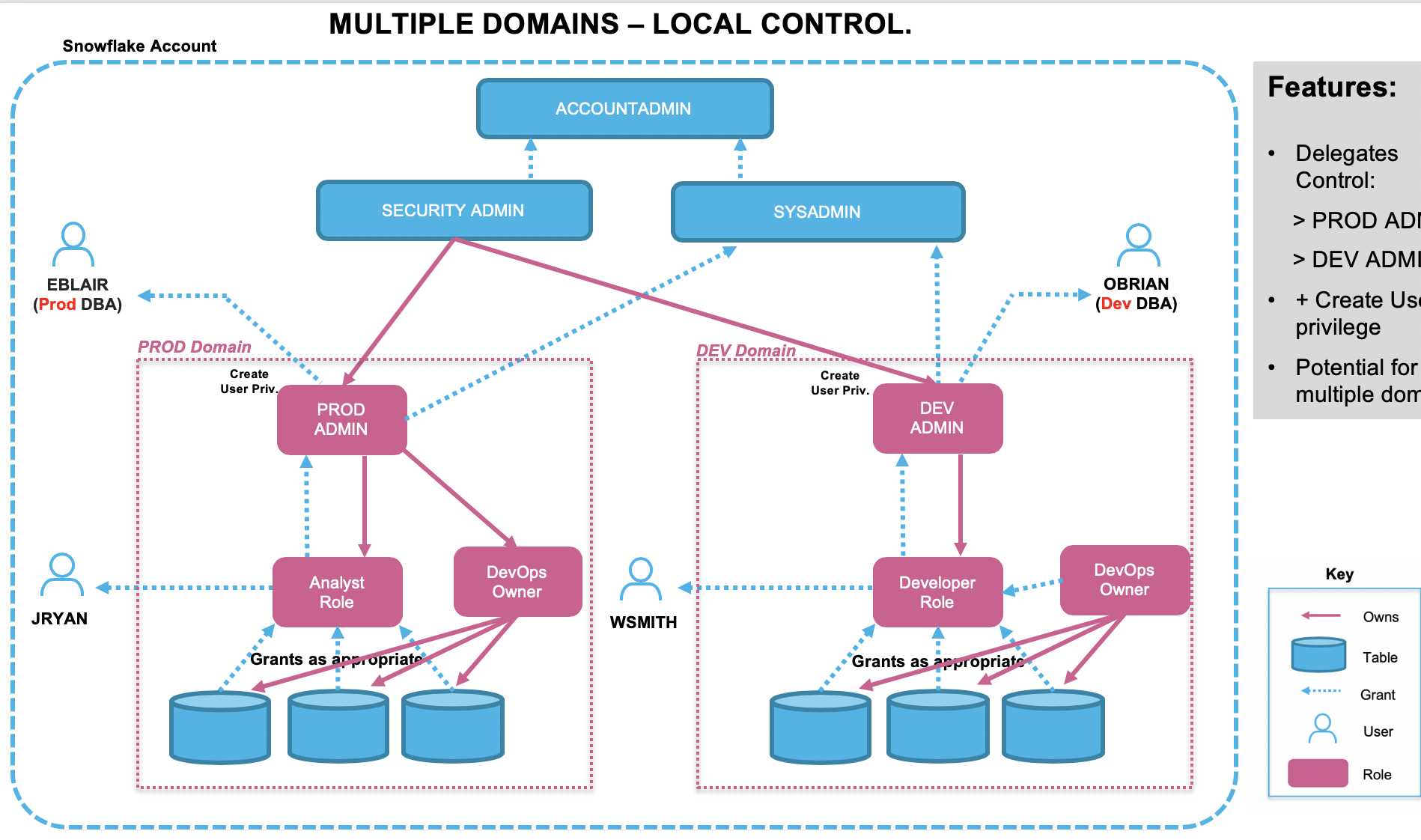
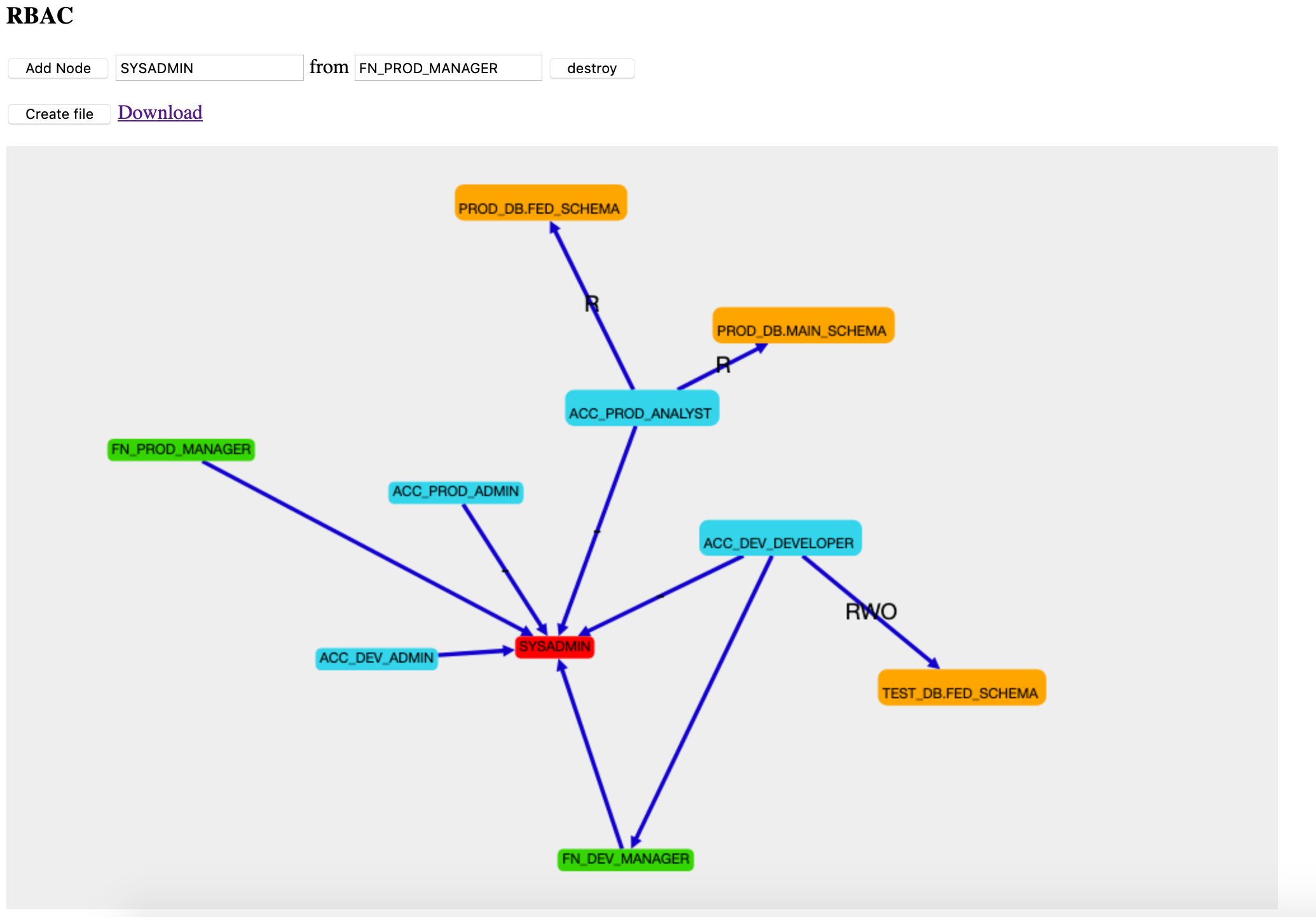
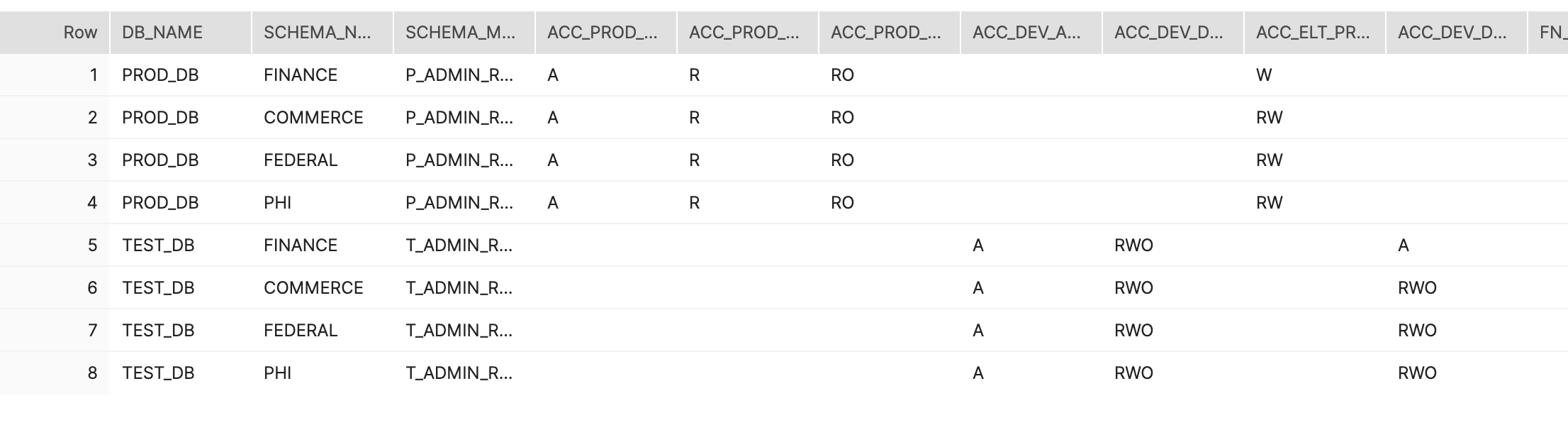
With an application it will be possible to set predefined Role model option, as Simple Domain model, Multi Domain model, pre-build for Financial, Medical, Government, etc., with additional support roles as ETL, Reportes, Shares. The models will be pre built tables/data with preset Role access diagrams. These Models can be tuned/tweaked later for customer needs.

The application will allow to connect to MS ADFS or other users-groups authentication services and will pre create users group as roles list. The list will be set as functional roles for the application. Application will check regularly for changes in the client AD/ list and follow the trigger the action.

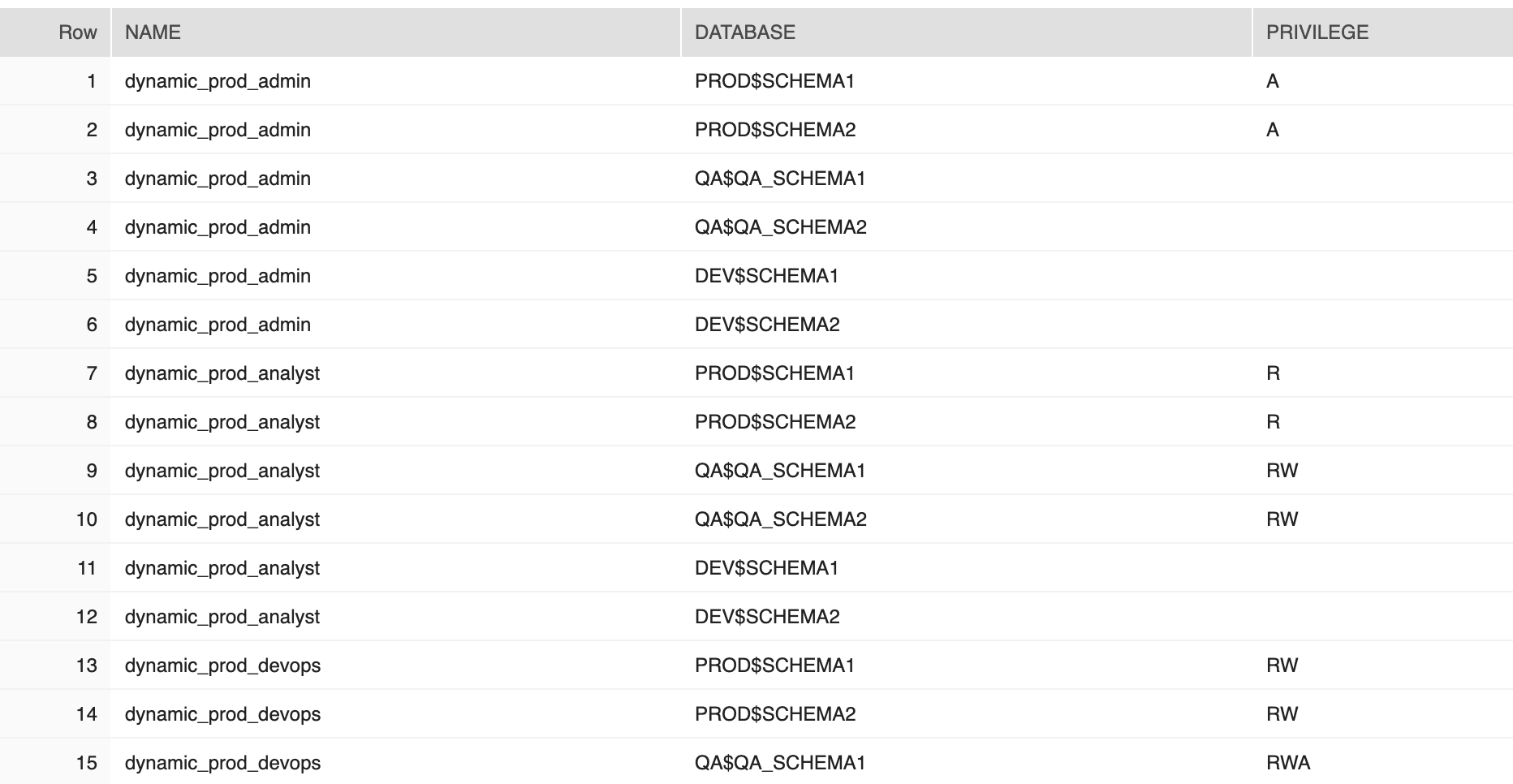
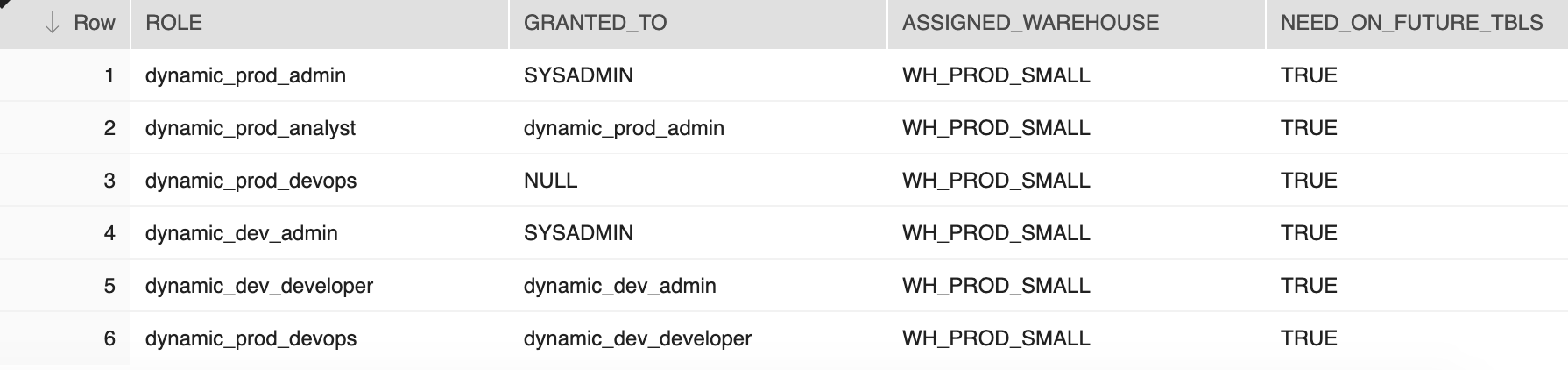
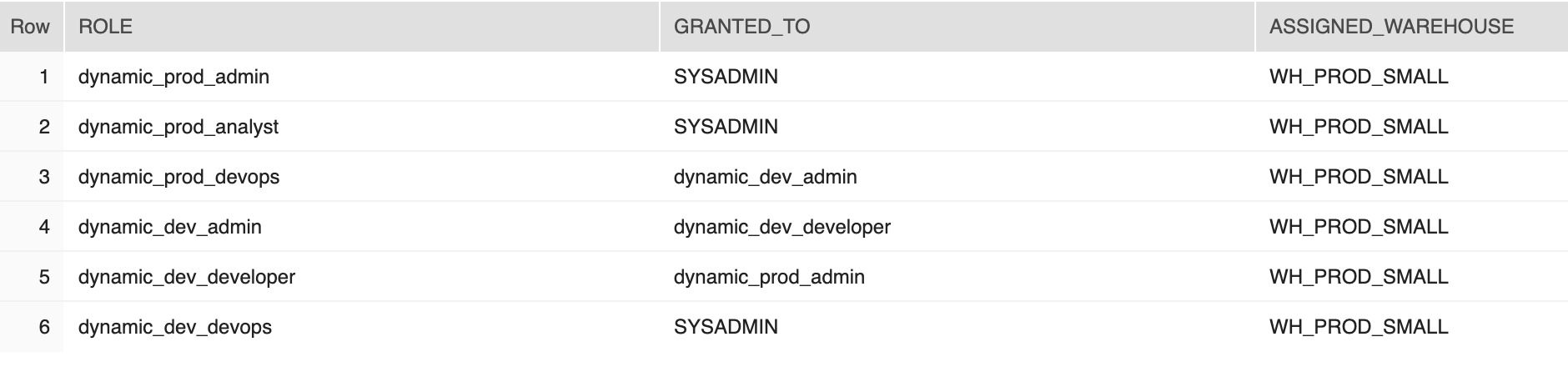
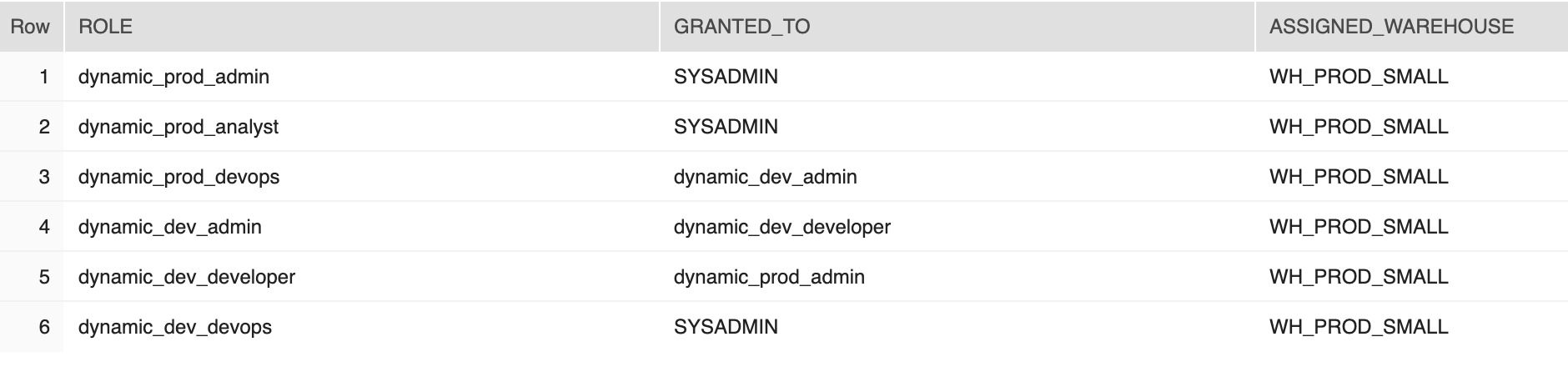
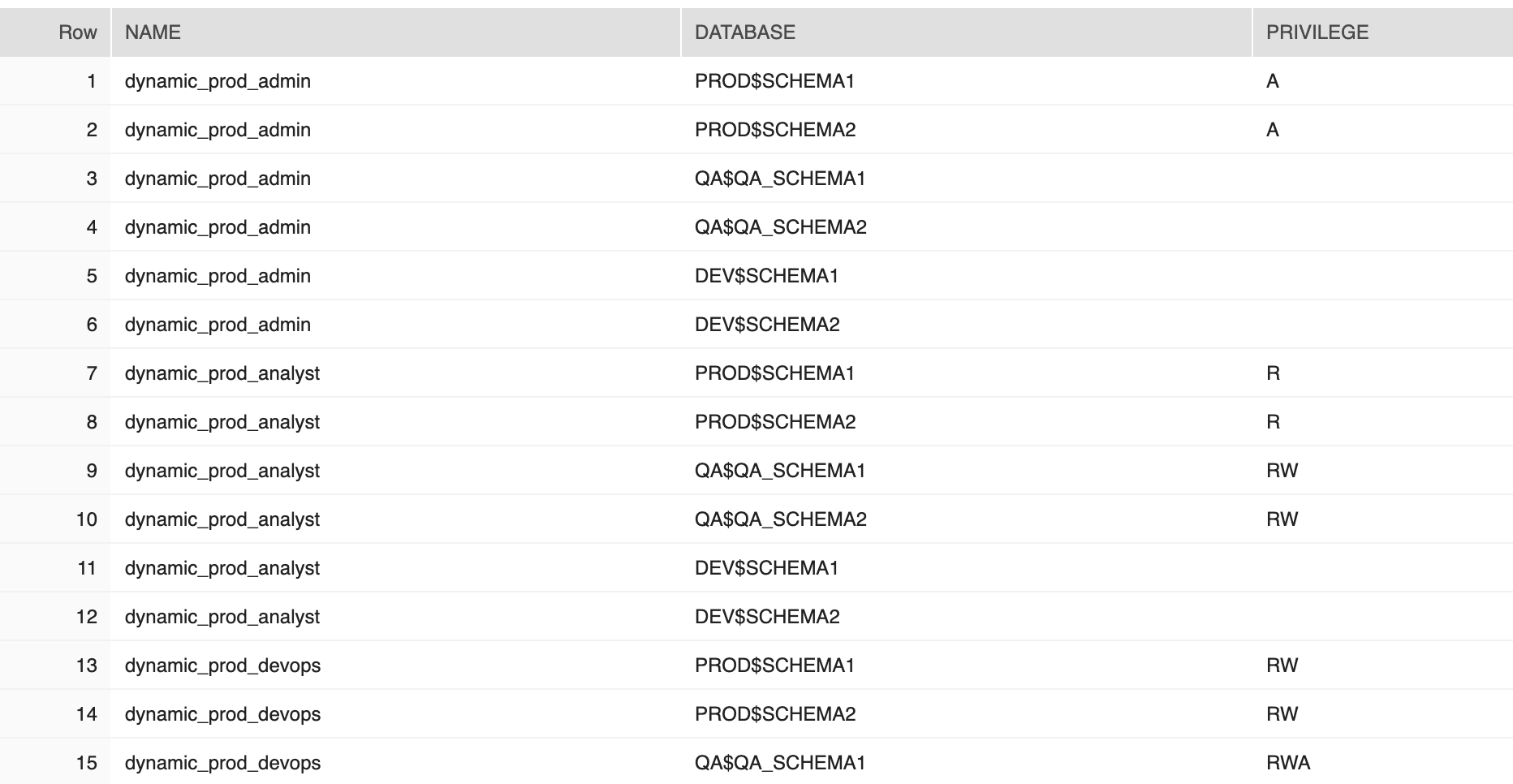
The structure for Access and Functional roles with roles hierarchy need to be prepared for this time. As a first step, draw graph diagram for Roles and RBAC-objects connections with the application, will help SA/customer visualize the relations between the RBAC objects. The Auto-RBAC application will create roles, grants from provided RAM table and support tables saved from three HTML pages.

## RBAC Recommendations

PS Security Group may create few RAM options with different and predefined Roles, Access level, Data warehouse Sizes to satisfy some options for various customers. Create RAM option with specific Role names and other parameters for Financial, Health, Government, Industrial, Commercial based by previous engagements and tune, tweak on client sites.

1. Customer needs to provide a list of users/roles separated into logical groups. The list may be part of MS AD, or some similar. 
2. In the first step, discuss with the customer about the desired ROLE hierarchy, ROLE access level. Drawing into a diagram and adding new data to User-Group table. For example two options for now.RBAC\_DB\_ACCESS\_LOCAL
3. Create a graph for Role and RBAC object relations:  
   
4. Add Roles from selected OPTION and set privileges to the Roles. Option 1 - Local. Tables with options maybe pre-created for different scenarios explained in the Option diagrams. (We can Clone the selected option to process.)S - Security; A - DB and Schema full access - no read-write; R - Read; W - Write; O - Operate Access to Modify Objects in schema.  
   Combinations - S, A, R, W, RW, RW(O).   
    **S**(ecurity) - should be used only as a separate setting, ‘grant CREATE USER, CREATE ROLE, MANAGE GRANTS to role ...’  
    **A**(dmin) - NO READ-WRITE on tables, will have ALL privileges on databases and schemas, plus ‘grant CREATE DATABASE, CREATE WAREHOUSE, CREATE SHARE, IMPORT SHARE ...’  
    **R** - ‘grant select on all tables in schema ’  
    **W** - ‘GRANT INSERT, UPDATE, DELETE, REFERENCES ON ALL TABLES IN SCHEMA ...’

**RW** - ‘GRANT ALL ON ALL TABLES IN SCHEMA ...’   
 RW**O** - in combination with ‘RW’ - ‘O’ has ‘grant USAGE on database…’ and ‘GRANT ALL ON SCHEMA ...’

1. Unpivot the previous table to the following format.  
   
2. Grouping together data for ROLES and Granting level, Warehouses.
3. 1. Role level will show level of hierarchy, relation to the Column ‘GRANTED\_TO’
   2. Assigned default Warehouse.
   3. Currently, defaulted to TRUE for Future Grans. Check if the role need access to the tables created in the future.
4. The application/main stored procedure will execute the process based on ACCESS\_CODE/PRIVILEGE from the Matrix table to create roles and object relations:
   1. The main SP will run over the first table for ROLE hierarchy. 
   2. Within the SP a corresponding data for each Role line will be invoked from the second table to temp access table. 
   3. ??The app will check for New Rows on ‘Created’ column and check on ‘Extra’ column to trigger action.
5. Execute Main stored procedure ‘SP\_RBAC\_PROCESSOR(’ calls internally ‘SP\_ROLE\_ASSIGNER’ -   
   call SP\_RBAC\_PROCESSOR();
   1. SP body:

create or replace procedure SP\_RBAC\_PROCESSOR()

returns string

language javascript

strict

execute as caller

as

$$

var EXECUTE\_Script = false;

var SQL\_ACCESS = "select role, granted\_to, ASSIGNED\_WAREHOUSE, true from ROLE\_RBAC\_MAIN\_PY";

var result\_set\_main = snowflake.execute ({sqlText: SQL\_ACCESS});

var forReturnt = "";

var retValueGrant = "";

var forReturntAll = "";

var create\_role = "";

var retCrtRole = "";

while (result\_set\_main.next()) {

var sp\_caller = "call SP\_ROLE\_ASSIGNER";

var ROLE\_NAME = result\_set\_main.getColumnValue(1);

var GRANTED\_TO = result\_set\_main.getColumnValue(2);

var ASSIGNED\_WAREHOUSE = result\_set\_main.getColumnValue(3);

var NEED\_ON\_FUTURE\_TBLS = result\_set\_main.getColumnValue(4);

create\_role = "CREATE ROLE IF NOT EXISTS " + ROLE\_NAME + ";";

retCrtRole = retCrtRole + "\n" + create\_role;

//var result\_set = snowflake.execute ({sqlText: create\_role});

if (GRANTED\_TO != '') //if no role to grant, just Functional Role

{

//Granting to Role

var sql\_cmd\_role\_grantedto = "GRANT ROLE " + ROLE\_NAME + " TO ROLE " + GRANTED\_TO + ";";

retValueGrant = retValueGrant + "\n" + sql\_cmd\_role\_grantedto;

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_role\_grantedto});

}

sp\_caller = sp\_caller + "('" + ROLE\_NAME + "','" + ASSIGNED\_WAREHOUSE + "'," + NEED\_ON\_FUTURE\_TBLS + ");"

forReturnt = forReturnt + "\n" + sp\_caller;

//Call SP from here -->

/\* \*/

var result\_set1 = snowflake.execute ({sqlText: sp\_caller});

while (result\_set1.next())

{

forReturntAll = forReturntAll + "\n" + result\_set1.getColumnValue(1);

}

}

return "--Create Roles as !Securityadmin!: \n USE ROLE SECURITYADMIN; \n " + retCrtRole + "\n \n --Grant role to Parent Role " + retValueGrant + "\n \n -- RBAC script: \n" + forReturntAll;

$$;

* 1. Returned Result:

1. SP - ‘SP\_ROLE\_ASSIGNER’ - called from the main SP(one tem stub in code for Schema in DB):

create or replace procedure SP\_ROLE\_ASSIGNER

(ROLE\_NAME VARCHAR, WAREHOUSE\_NAME VARCHAR, GRANT\_ONFUTURE BOOLEAN)

returns string

language javascript

strict

execute as caller

as

$$

//Parameter that allows CMD Execution, 'false' => just text, no execution

var EXECUTE\_Script = false

var retValueFrom = "";

var SQL\_ACCESS =

"select ROLES, DB\_NAME, SCHEMA\_NAME, SCHEMA\_MNGR,PRIVILEGE from RBAC\_DB\_ACCESS\_MNG unpivot(privilege for ROLES in (" + ROLE\_NAME + ")) WHERE PRIVILEGE <> ''";

var result\_set\_access = snowflake.execute({ sqlText: SQL\_ACCESS });

var retValueFromIf = "";

var schema = "";

while (result\_set\_access.next()) {

var ROLE\_NAME = result\_set\_access.getColumnValue(1);

var DB\_NAME = result\_set\_access.getColumnValue(2);

var SCHEMA\_NAME = result\_set\_access.getColumnValue(3);

var ROLE\_MNGD\_SCHEMA = result\_set\_access.getColumnValue(4).toUpperCase();

var ACCESS\_CODE = result\_set\_access.getColumnValue(5).toUpperCase();

retValueFrom = retValueFrom + "\n" + "--New Schema " + SCHEMA\_NAME;

var checkAccessCode = ACCESS\_CODE.charAt(0);//check if the first letter A or S

if (checkAccessCode == 'S') {

//retValueFromIf = "Security Only";

var sql\_cmd\_for\_skrt =

"USE ROLE SECURITYADMIN; \n GRANT CREATE USER, CREATE ROLE, MANAGE GRANTS TO ROLE " + ROLE\_NAME + ";";

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_skrt});

retValueFrom = sql\_cmd\_for\_skrt + "\n";

} else if (checkAccessCode == 'A') {

retValueFromIf = "Sys Only";

//grant CREATE DATABASE on account to sysadmin;

var sql\_cmd\_for\_acc =

"USE ROLE SYSADMIN; \n " +

"GRANT CREATE DATABASE, CREATE WAREHOUSE ON ACCOUNT TO " + ROLE\_NAME + ";";

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_acc});

var sql\_cmd\_for\_db =

"GRANT ALL ON DATABASE " + DB\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_db});

var sql\_cmd\_for\_schema =

"GRANT ALL ON ALL SCHEMAS IN DATABASE " + DB\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_schema});

retValueFrom = retValueFrom + "\n" + sql\_cmd\_for\_acc + "\n" + sql\_cmd\_for\_db + "\n" + sql\_cmd\_for\_schema + "\n";

}

if ((checkAccessCode == 'A' || checkAccessCode == 'S')) //Checks, it is NOT Admin nor Security

{

//Admin or Security

}

else

{

//retValueFromIf = retValueFromIf + "::RWO+not Sec or Sys ";

var sql\_cmd\_for\_db =

"USE ROLE SYSADMIN;-- Or role assigned to perform this \n " +

"GRANT USAGE ON DATABASE " + DB\_NAME + " TO ROLE " + ROLE\_NAME + " ;"; //?? ONLY USAGE on DB or need CREATE SCHEMA ???

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_db});

if (ROLE\_MNGD\_SCHEMA.length != 0)

{

var sql\_cmd\_mngd\_schema\_role\_set = 'USE ROLE ' + ROLE\_MNGD\_SCHEMA + ';'

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_mngd\_schema\_role\_set});

}

if (ACCESS\_CODE.includes("O")) //in a case 'RWO' Operate: Create, Delete, Modify

{

var sql\_cmd\_for\_schema =

"GRANT ALL ON SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_schema});

retValueFrom = retValueFrom + "\n" + sql\_cmd\_for\_db + "\n" + sql\_cmd\_mngd\_schema\_role\_set + "\n" + sql\_cmd\_for\_schema + "\n";

}

else

{

// add Granting on schema level...

var sql\_cmd\_for\_schema =

"GRANT USAGE ON SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_schema});

retValueFrom = retValueFrom + "\n" + sql\_cmd\_for\_db + "\n" + sql\_cmd\_mngd\_schema\_role\_set + "\n" + sql\_cmd\_for\_schema + "\n";

}

{

// Process ACCESS for Tables

var sql\_cmd\_for\_tables = "";

var sql\_cmd\_grant\_on\_future = "";

var ACCESS\_CODE\_stripped = ACCESS\_CODE.replace("O", "");//remove 'Operation'

switch (ACCESS\_CODE\_stripped) {

case "R":

// code block

sql\_cmd\_for\_tables =

"GRANT SELECT ON ALL TABLES IN SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (GRANT\_ONFUTURE == true)

sql\_cmd\_grant\_on\_future =

"GRANT SELECT ON FUTURE TABLES IN SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

break;

case "RW":

// code block

sql\_cmd\_for\_tables =

"GRANT ALL ON ALL TABLES IN SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (GRANT\_ONFUTURE == true)

sql\_cmd\_grant\_on\_future =

"GRANT ALL ON FUTURE TABLES IN SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

break;

case "W":

// code block

sql\_cmd\_for\_tables =

"GRANT INSERT, UPDATE, DELETE, REFERENCES ON ALL TABLES IN SCHEMA " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

if (GRANT\_ONFUTURE == true)

sql\_cmd\_grant\_on\_future =

"GRANT INSERT, UPDATE, DELETE on future tables in schema " + DB\_NAME + "." + SCHEMA\_NAME + " TO ROLE " + ROLE\_NAME + ";";

break;

default:

// code block

var sql\_cmd\_for\_tables = "";//Generate an error

sql\_cmd\_grant\_on\_future = "";

}

//Execute SQL to grant ACCESS on Databse.SCHEMA.TABLES

if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_tables});

//retValueFrom = retValueFrom + "\n" + sql\_cmd\_usedb + "\n" + sql\_cmd\_for\_tables;

retValueFrom = retValueFrom + "\n" + sql\_cmd\_for\_tables + "\n" + sql\_cmd\_grant\_on\_future + "\n";

}

}

}

// Assign a Warehouse to role // ?? Operate, ...???

var sql\_cmd\_for\_warehouse =

"USE ROLE SYSADMIN; \n " +

"GRANT USAGE, OPERATE ON WAREHOUSES " + WAREHOUSE\_NAME + " TO ROLE " + ROLE\_NAME + ";";

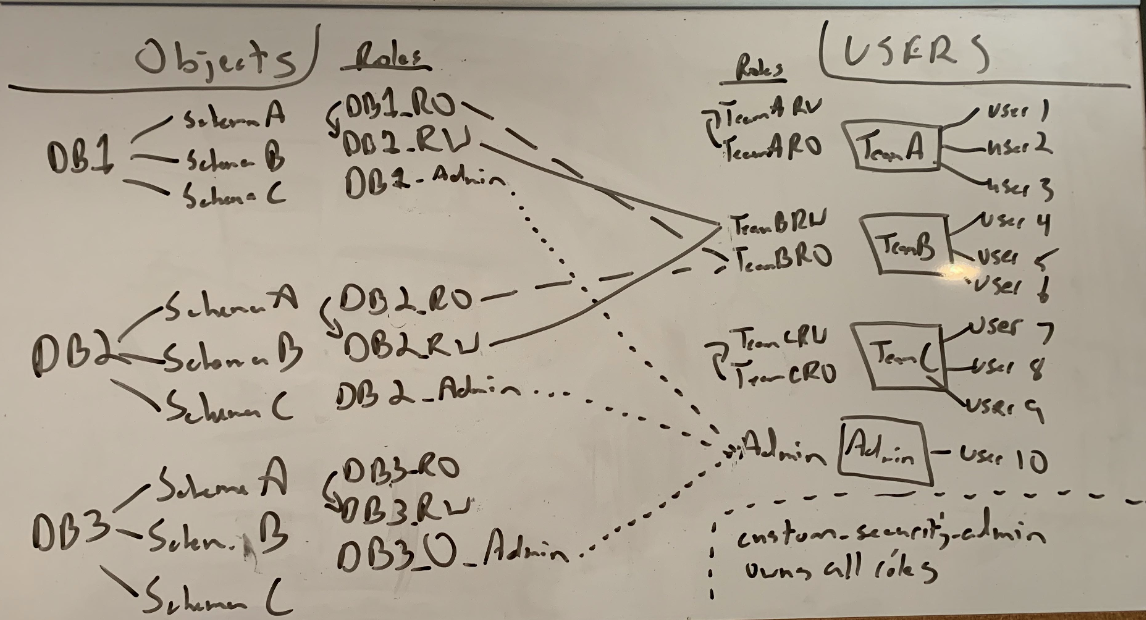
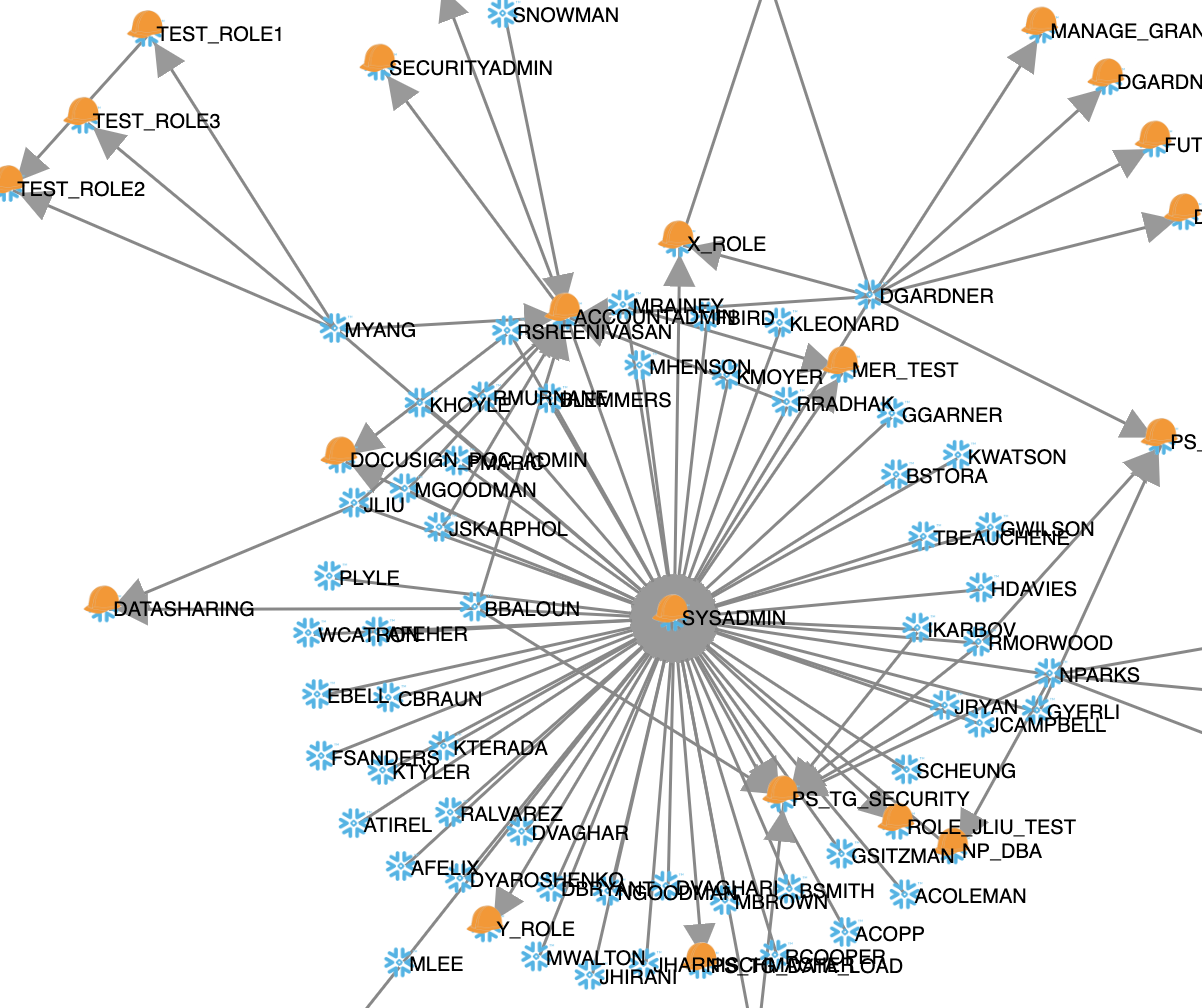
if (EXECUTE\_Script) snowflake.execute ({sqlText: sql\_cmd\_for\_warehouse});

retValueFrom = retValueFrom + "\n" + sql\_cmd\_for\_warehouse + "\n";

return "--New Role :::\n" + retValueFrom;

$$

;

1. The application will create Users from prepared table with Groups and some additional parameters to specify some needs described and approved in the diagram. 
2. Currently we can generate the visualization representation on User-Role relation for the customer. 
3. As a next step we can create a graph to show Roles to Objects relation, Role to DB/Schema/Table…, compare option, ?