Paper 048-2013

How to Automate Security Filters for SAS® OLAP Cubes Using Users Groups Information Available in SAS Management Console

Plinio Faria, Sao Paulo, Brazil

ABSTRACT

In order to limit the data that a user can access in a OLAP Cube, it is required to use MDX conditions and those expressions must be customized for every OLAP Cube because each one has a different structure. This paper will focus on showing how to automate the creation of Multidimensional Expressios (MDX) conditions using the users groups information available in the metadata server and in a way that is possible to change the data subset that the users are allowed to see only changing the user group in SAS Management Console. It will also be demonstrated how to implement OLAP Cube security adding directly the users login to a cube dimension.

INTRODUCTION

In OLAP cubes it is not possible to use permission tables in the same way that is done in relational databases and to apply security roles it is necessary to create MDX expressions to limit user access to data.

In this example, an enterprise is divided in regions and subregions as below in figure 1:

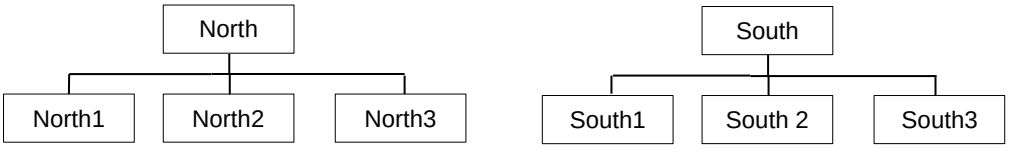


Figure 1. Enterprise regions and subregions

Each region has a SAS User Group defined in SAS Management in order to grant access.

Supposing the North manager takes vacation and the South manager assumes his/her position, taking care of two regions, the first manager will belong to both groups, but only adding the user to a new group will not grant access in the OLAP cube. It will be necessary to access metadata server to retrieve which groups a user belongs to and then apply a customized MDX condition in the permission table.

CREATING AN OLAP PERMISSION TABLE

The MDX expressions are stored in a permission table and to create it follow these steps:

1. Open SAS OLAP Cube Studio
2. Select Tools ➡ Manage Permission Tables ➡ Add
3. Fill in the required fields
4. Select the inventory tab and then select the OLAP Cube

In this example, the permission table was named PERMISSIONS and looks like display 1:

	fullname	olapschema	cube	dimension	items	permission	perm_type	mdx_condition	remove_ace
1	HELENA TOTH	SASMain - OLAP Schema	_SGF_example			ReadMetadata	GC		N
2	HELENA TOTH	SASMain - OLAP Schema	_SGF_example	Regions		Read	GD	{[Regions].[All Regions].[South]}	N

Display 2. Permissions table before changes

GROUPS INFORMATION

The program below will use the built-in SAS macro MDUEXTR to retrieve metadata information and generate the table WORK.USERS_GROUP containing all users groups and the users associated with them:

```
libname MD 'X:\COMP_TEST\SGF\MD';

options metaserver="xxx"
metaport=8561
metauser="sasadm@saspw"
metapass="xxx"
metaprotocol=bridge
metarepository=Foundation;


%mduextr(libref=MD);


data MD.PERSON;
  retain Name Title DisplayName keyid description objid externalkey;
  set MD.PERSON;
  ATTRIB DisplayName LENGTH=$256 FORMAT=$256. INFORMAT=$256. LABEL='Person
DisplayName';
run;

data MD.IDGRPS;
  retain Name DisplayName keyid description grpType objid externalkey;
  set MD.IDGRPS;
  ATTRIB DisplayName LENGTH=$256 FORMAT=$256. INFORMAT=$256. LABEL='Person
DisplayName';
run;

proc sql;
create table users_group as
select distinct
    compress(TRANWRD(upcase(a.USERID),"CORP\",""))  as  UserLogin,
    b.name as fullname,
    c.name as GroupName
FROM MD.LOGINS a,
     MD.PERSON b,
     MD.GROUP_INFO c,
     MD.GRPMEMS d
WHERE a.keyID=b.keyid=d.memkeyid
      AND d.grpkeyid=c.ID
order by b.name;
```

The USERS_GROUP table should will be:

	UserLogin	fullname	GroupName
1	I310461	HELENA TOTH	North
2	I310461	HELENA TOTH	South

Display 2. Table containing the groups that a user belongs to.

CREATING MDX CONDITIONS

Using the table USERS_GROUPS generated in the step above, it will be created the ACCESS table containing customized MDX conditions for the OLAP Cube named SGF_example.

Since the USERS_GROUPS table contains all groups registered in the metadata server, it is necessary to filter only those ones related to information access:

```
data access;
set
users_group;
where upcase(GROUPNAME) in ("NORTH","SOUTH");
run;
```

The MDX conditions will follow a pattern according to the dimension hierarchy:

```
%let mdx_template=[Regions].[All Regions].[;
```

This step will generate a MDX condition for each user, e.g. if the user belongs to “South” group it will be generated the expression:
{[Regions].[All Regions].[South]}

If the user belongs to groups North and South the expression will be:
{[Regions].[All Regions].[North],[Regions].[All Regions].[South]}.

```
data access;
set
access;
OLAPschema="SASMain - OLAP Schema";
dimension="Regions";
cube="_SGF_example";
remove_ace="N";
by fullname notsorted;
retain mdx_condition;
if first.fullname=1 then mdx_condition="{&mdx_template." || strip(GROUPNAME) ||
"]";
if first.fullname=0 then mdx_condition= catx(",",mdx_condition,"&mdx_template." ||
strip(GROUPNAME) || "]");
if last.fullname=1 then mdx_condition=strip(mdx_condition) || "}";
if last.fullname=1;
run;
```

In this part the mdx_condition column will be updated, but previously the users that are not in the table yet it will be inserted.

```
/* Users with no access to cube */
Proc sql;
Create table cube_access as
SELECT a.fullname
FROM access a left join (SELECT fullname FROM SGF.permissions WHERE dimension="") b
                        On a.fullname=b.fullname
WHERE b.fullname is missing;


/* Users with no access to the dimension */
Proc sql;
Create table dimension_access as
SELECT a.fullname
FROM access a left join (SELECT fullname FROM SGF.permissions WHERE
dimension="Regions") b
                        On a.fullname=b.fullname
WHERE b.fullname is missing;

/* Insert access to cubes*/
Proc sql;
DELETE *
FROM SGF.permissions
```

```
WHERE fullname in (SELECT fullname FROM cube_access)
      OR fullname in (SELECT fullname FROM dimension_access);

Proc sql;
Insert into SGF.permissions (fullname, OLAPschema, cube, dimension, items,
permission, perm_type, mdx_condition, remove_ace)
Select fullname, OLAPschema, cube, "","", "Read", "GC", "", remove_ace
FROM ACCESS
WHERE fullname in (SELECT fullname FROM cube_access);
Proc sql;
Insert into SGF.permissions (fullname, OLAPschema, cube, dimension, items,
permission, perm_type, mdx_condition, remove_ace)
Select fullname, OLAPschema, cube, "","", "ReadMetadata", "GC", "", remove_ace
FROM ACCESS
WHERE fullname in (SELECT fullname FROM cube_access);

/* Inserts access to dimension */
Proc sql;
Insert into SGF.permissions (fullname, OLAPschema, cube, dimension, items,
permission, perm_type, mdx_condition, remove_ace)
Select fullname, OLAPschema, cube, "Regions","", "Read", "GD", mdx_condition,
remove_ace
FROM ACCESS
WHERE fullname in (SELECT fullname FROM dimension_access);
```

The permission table will contain the following records:

	fullname	olapschema	cube	dimension	items	permission	perm_type	mdx_condition	remove_ace
1	HELENA TOTH	SASMain - OLAP Sche...	_SGF_example			Read	GC		N
2	HELENA TOTH	SASMain - OLAP Sche...	_SGF_example			ReadMetadata	GC		N
3	HELENA TOTH	SASMain - OLAP Schema	_SGF_example	Regions		Read	GD	{[Regions].[All Regions].[North], [Regions].[All Regions].[South]}	N

Display 3. Permission table after updates

APPLYING MDX CONDITIONS

The program below will update the permission table but it will not take effect until the permissions are applied to the OLAP Cube.

In order to apply the permission using OLAP Cube Studio follow these steps:

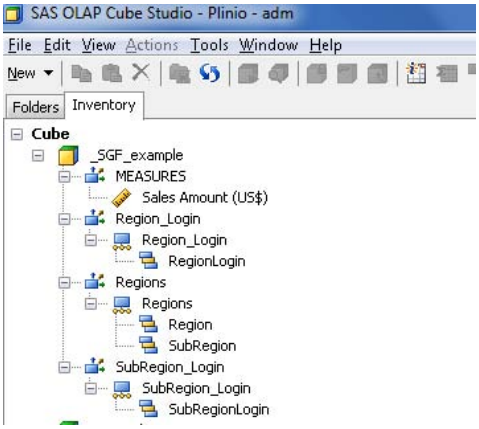
1. Open SAS OLAP Cube Studio
2. Select Tools ➡ Manage Permission Tables ➡ Add
3. Select the permission table and click on Modify
4. Click on Run or click on Export Code to get the code and use it to automate the process of applying conditions

CREATING A DIMENSION THAT CONTAINS USERS LOGINS

When you have many users and each one must access a specific part of OLAP, it is not recommended to create a user group for a single user. In this case you may create a column in the input table from which the OLAP Cube will be generated, specifying which user should see the registry and then create a dimension based in this user login column (see column SubRegionLogin in display 3).

	RegionID	Region	SubRegionID	SubRegion	RegionLogin	SubRegionLogin	SalesAmount
1	1	North	11	North1	F949843@CORP	i310461@CORP	200
2	1	North	12	North2	F949843@CORP	i310461@CORP	300
3	2	South	21	South1	F949843@CORP	i310140@CORP	180
4	2	South	22	South2	F949843@CORP	i310140@CORP	220

Display 3. Input table for OLAP Cube



Display 4. Cube Hierarchy

The login dimension must have a MDX condition applied:

1. Open **SAS OLAP Cube Studio**
2. Expand the cubes dimensions
3. Right click on the login dimension (Region_Login in this example) and select properties
4. Select **Authorization tab**
5. Select the user group and Click on the Grant check box for Read
6. Click on **Add authorization** button
7. Select **Create an advanced MDX expression using the expression builder**
8. Click on **Build Formula** button and add the expression {[Region_Login].[All Region_Login].[SUB::SAS.Userid]}

It will be shown the error message “Formula error - An invalid member name was encountered in the MDX statement” because the variable SAS.Userid is not resolved during the validation process, but the MDX condition will work on queries execution.

CONCLUSION

Implementing and managing OLAP Cubes Security require double access grants and can be easier if the MDX expression are fixed or automated. In this way, access can be granted or denied with a few clicks on SAS Management Console and be done by a user that does not have knowledge on MDX language.

REFERENCES

SAS® 9.2 Language Interfaces to Metadata. SAS Institute Inc. 2009. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Plinio Faria
Rua Luis Coelho, 53 apto 1A
Sao Paulo-SP
Brazil
Phone: + 55 11 99808-3632