Paper 11701-2016

An End-to-End, Automation Framework for Implementing Identity-Driven Row-Level Security Using SAS® Visual Analytics

Paul Johnson, Sopra Steria; Dileep Pournami, RBS;
Ekaitz Goienola, SAS UK Professional Services

ABSTRACT

At the Royal Bank of Scotland (RBS), business intelligence users require sophisticated security permissions both at object level and data (row) level in order to comply with data security, audit, and regulatory requirements. When we rolled out SAS® Visual Analytics to our main stakeholder groups, this was identified as a key requirement as data is no longer restricted to the desktop but is increasingly available on mobile devices such as tablets and smart phones. Implementing row-level security (RLS) controls, in addition to standard security measures such as authentication, is a most effective final layer in your data authorization process. RLS procedures in leading relational database management systems (RDBMSs) and business intelligence (BI) software are fairly commonplace, but with the emergence of big data and in-memory visualization tools such as SAS® Visual Analytics, those RLS procedures now need to be extended to the memory interface. Identity-driven row-level security is a specific RLS technique that enables the same report query to retrieve different sets of data in accordance with the varying security privileges afforded to the respective users.

This paper discusses an automated framework approach for applying identity-driven RLS controls on SAS® Visual Analytics and the plans to implement a generic end-to-end RLS framework extended to the Teradata data warehouse.

INTRODUCTION

RBS have been early adopters of SAS® Visual Analytics (VA) ever since it was released. Adoption has steadily increased over the years, but it was at the tail end of 2014 that a renewed interest by a wider community in the bank triggered the need to modernise the lab environment and go fully Production.

Spring of 2015 saw an upgrade to the underlying infrastructure and a software upgrade to the SAS® Visual Analytics 7.1 release. Uptake has risen exponentially since.

Fast forward to summer of 2015. The environment hosting SAS® Visual Analytics services circa 1,300 end-users and analysts; and plans are underway to build a larger, strategic platform to support over 10,000 end-users.

With an increased interest in the use of SAS® Visual Analytics across different franchises in the bank, securing the data and reports being developed is of paramount importance.

Security is initially tackled by designing the platform to adhere to multi-tenancy design principles. As at summer 2015, the multi-tenancy design is still in its infancy, with dedicated SAS® LASR™ Analytic Servers being deployed for each of the bank franchises, which leaves access to underlying data open to all users of a given franchise.

The solution is, Row-Level Security in SAS® Visual Analytics.

IDENTITY-DRIVEN ROW-LEVEL SECURITY

Identity-driven RLS may be defined as when the same query returns a different set of rows, as determined by the varying data access privileges afforded to each user of an application or data warehouse. The SAS® VA implementation of Identity-driven RLS is implemented through the following features:

- Tools to populate and synchronize identity groups in the SAS® Metadata repository.
- A metadata interface that facilitates the specification of a conditional grant read permission on a LASR™ table as a clause that serves to compare RLS column values with identity group values.
- The automatic resolution of identity values in the conditional grant permission clause.
- The execution of the conditional grant permission clause at run-time, suppressing non-matching rows where access is unauthorized.

OBJECTIVES

The high level objectives of the RBS implementation were to:

- Secure LASR™ data at the row-level based in accordance with the strategic security data model.
- Adhere to security guidelines.
- Comply with audit, risk and security regulations.
- Minimize any user, support or administration impact.
- Avoid duplication at the data and reporting layer.

CASE STUDY AT RBS – MAJOR UK BANK

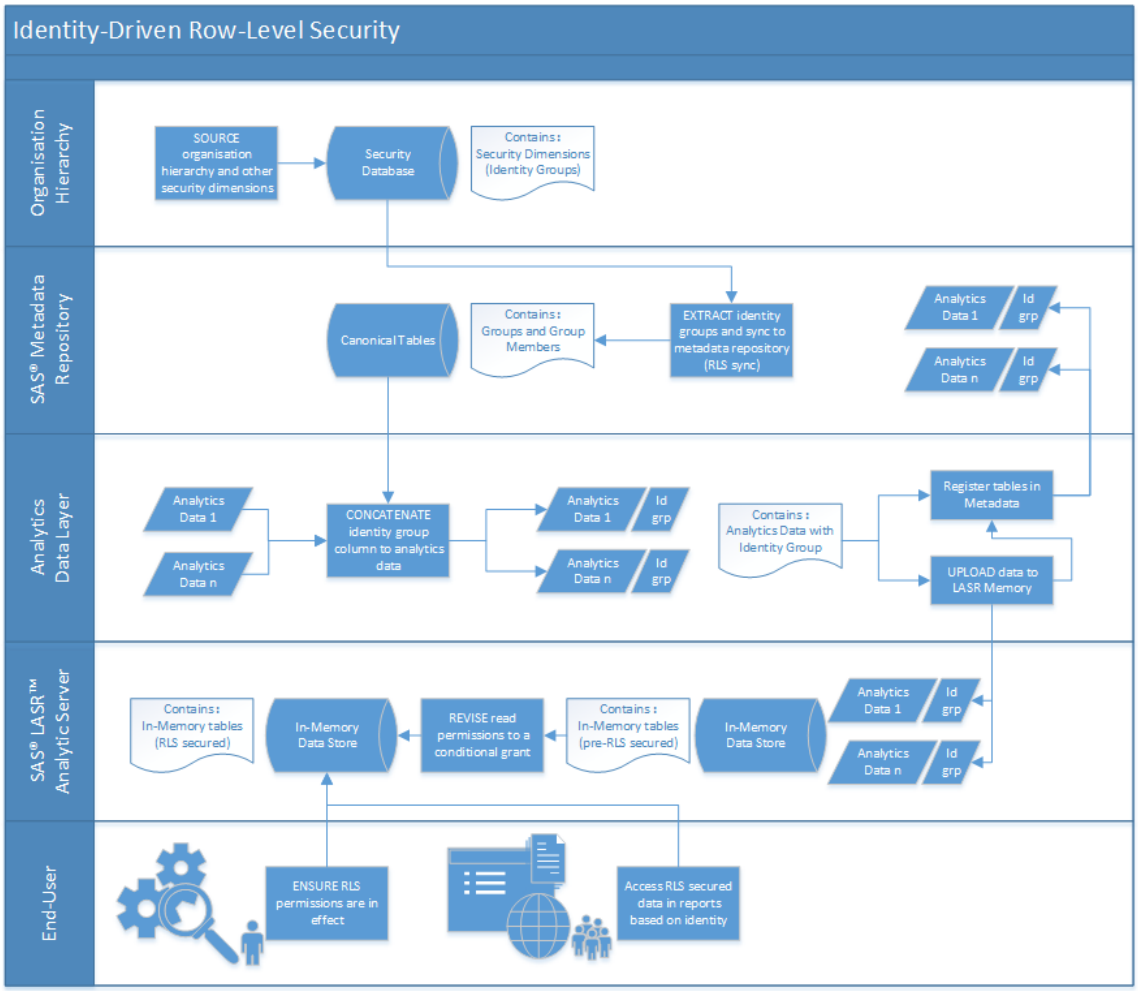


Figure 1. RLS Framework – SECURE

APPROACH

The approach taken at RBS was to evaluate the potential options to implement Identity-Driven RLS within VA, with Star schema views and Metadata identity groups being the main contenders. The metadata identity group option compared best and so was selected for the proof of concept prototype. To address capacity and performance concerns the prototype was scaled up to comprise approximately 30,000 identity groups, which performed well during stress testing. As the prototype satisfied the key objectives, the RLS framework designs were completed with a strong emphasis on flexibility and extensibility for bank-wide rollout. Extracts from the security database were compiled, supplemented by lookup tables that mapped to the target LASR™ tables to be secured.

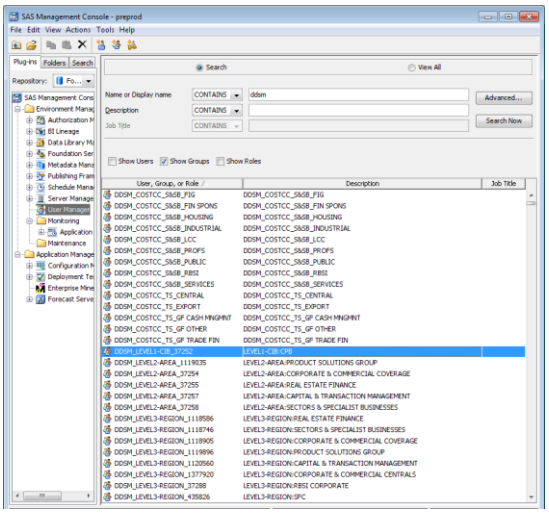
The flexibility of the RLS framework was then demonstrated through its ability to accommodate additional user group information maintained external to the security database. This provides the means to extend or modify the permissions inherent with the security database for specific use cases. Finally the RLS framework was then fully integrated into the metadata security layer, and scheduled to run as part of the JML batch overnight job suite.

RLS FRAMEWORK - S E C U R E

The RLS framework, a.k.a. “SECURE” represents the principal steps required to implement Identity-Driven RLS within a SAS® Visual Analytics environment.

SOURCE

The key prerequisites for implementing RLS are to establish where and how to source the identity groups needed to secure the data. If you’re contemplating an identity-driven RLS implementation based on **SAS.IdentityGroups** then both User and Group information needs to be present within the source data. Ideally the source data will be an existing security model that contains a user and group dimension. In theory any dimensional column that maps to a user identity may be a potential candidate for the group values. The group dimension can for example be: department names, organizational nodes, geographical regions or even portfolio codes. The optimal security model will ideally comprise all the dimensional columns required to secure your data. The most important consideration though, is that the security dimension must map to the LASR™ table(s) targeted to be RLS secured. Should your existing metadata groups already reflect your required security dimensions then this step and the extract step to follow, are of course not required.



Display 1. SAS® Management Console User Manager Plug-In showing RLS Groups

EXTRACT

Once the user and group values have been identified the next step is to *extract* them from the source data and transform into group and group membership tables. The group and group membership information must adhere to the canonical model maintained within the SAS® metadata repository (see diagram below). The SAS® bulk load, auto-call macros (%MDU), serve quite well for this purpose. However the extract from source, will largely comprise bespoke SAS® or SQL code.

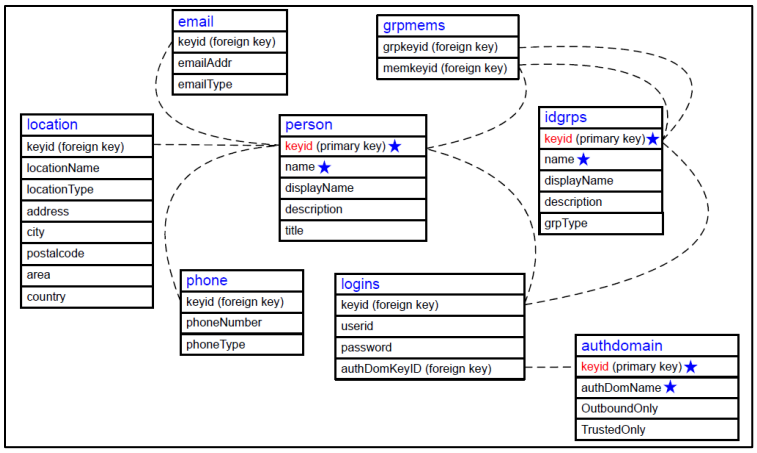


Figure 2. SAS® Canonical Tables[1]

The extract process comprises the following steps:

(1) An extract of the target SAS® Metadata repository can be performed using the %MDUEXTR macro as shown in the sample code excerpt below.

(2) The source and target RLS groups are then compared using the %MDUCMP macro which determines the Joiners, Movers and Leavers (JML), in terms of identity groups. The resultant group tables in the JML library will have a _add, _update and _delete suffix respectively. Note that the **SAS.IdentityGroups** solution, featured in this paper, is only reliant on the group (IDGRPS) and group membership (GRPMEMS) tables. The remaining canonical tables may be null populated and excluded via the filter dataset (*work.exclude*) specified in the exceptions parameter of the macro.

(3) The %MDUCHGV macro is used to verify the integrity of the JML data. Any errors detected will be written to the errors dataset specified (*work.errors*).

(4) The value of the mduchgv_errors macro variable (*not shown*) may be checked in order conditionally execute the %MDUCHGLB macro, which syncs the RLS identity groups in the metadata repository.

```
(1) %mduextr(libref=target);

        proc sql ;
         create table work.exclude (filter char(100), tablename char(40) );
         insert into work.exclude
          values('1=1','email_info')
          values('1=1','location_info')
          values('1=1','logins_info')
          values('1=1','person_info')
          values('1=1','phone_info')
          values('1=1','email')
          values('1=1','location')
          values('1=1','logins')
          values('1=1','phone')
          values('1=1','groupmemgroups_info')
          values('1=1','groupmempersons_info')
          values("grpkeyid not contains &GRP_PREFIX",'grpmems')
          values("keyid not contains &GRP_PREFIX",'idgrps')      ;
        quit;

(2)%mducmp(master=source,target=target,change=JML,exceptions=work.exclude,externonly=1,authdomcompare=NAME);
(3)%mduchgv(change=JML,target=target,temp=work,errorsds=work.errors);
(4)%mduchglb(change=JML,temp=work,failedobjs=work.failedobjs,extidtag=RLSImport,blksize=100);
```

Output 1. Sample Code using Bulk Load Macros to Synchronize RLS Metadata Groups

A backup of the metadata repository is strongly recommended beforehand, since any metadata synchronization failures could potentially impact the integrity of the metadata repository. Another important consideration is to keep the RLS identity groups separate from the existing metadata groups, particularly if the latter are also imported via the bulk load macros. One way to accomplish this is to prefix the RLS groups with an identifier. The RLS groups can then be segregated for processing using a corresponding filter in the dataset specified via the exceptions parameter in %MDUCMP macro. For RLS purposes you may discard any superfluous users in the group membership table, which do NOT already exist in the metadata repository. Finally, checks should be performed to verify the RLS metadata sync process was successful. The following output shows a simple metadata identity group verification check.

```
15          %let prefix=DDSM;
16             DATA _null_ ;
17                 length id $30 groups 4;
18                 id='';
19                 groups=1;
20                 groups=METADATA_GETNOBJ("omsobj:IdentityGroup?@Name contains '&prefix'",groups,id);
21                 putlog "The number of RLS Metadata Identity Groups prefixed with &prefix is: " groups comma.;
22             RUN;
The number of RLS Metadata Identity Groups prefixed with DDSM is: 28,792
```

Output 2. SAS® Log Showing a Sample Query to Verify the Number of RLS Metadata Groups

CONCATENATE

The *concatenate* step marks the start of the RLS enablement process. At this stage an RLS column (RC) is concatenated to the LASR™ table, should it not already exist. The RC should be defined as a character variable as the results returned by the **SAS.IdentityGroups** attribute will be a character string series of identity groups. Furthermore the RC may not be created within the VA interface e.g. as a calculated or duplicate item. Most importantly the RC should be populated with values that exactly match the RLS identity source group values loaded into the metadata, via the RLS Sync process. Any rows loaded with non matching values will not be displayed in VA, once a conditional read statement has been applied further on in the process. It is therefore imperative that the appropriate data quality checks are performed during this process to prevent undesirable results.

UPLOAD

Once the RC is populated on the LASR™ table then the *upload* process can be performed to render the table to the LASR™ server memory. At this point, it is important to note that ALL the data (rows) uploaded are still accessible through the LASR™ server engine, as they are not yet RLS secured. Should this be an issue then you may consider initially loading the table with test data or null records. However once the conditional read permissions are applied, as per 'revise' step below, then the table will be RLS secured for this and any subsequent LASR™ loads thereafter.

REVISE

The last step in the RLS enablement process is to *revise* the read permissions for the RLS table by applying a conditional grant statement. This can be done via the authorization property in the VA administration interface. The **SAS.IdentityGroups** system attribute is not however available for selection from the Visual tab, so you will have to manually type the conditional grant statement in the adjacent Text tab. The conditional grant statement can alternatively be applied via the java batch metadata interface using the **sas-set-metadata-access** program (*see output below*). This method requires the standard metadata parameters to be supplied along with the credentials from a user in the SAS® Metadata administrators group.

```
sasinst1@vstcbis02d:$ cd /opt/sas/software/SASPlatformObjectFramework/9.4/tools
sasinst1@vstcbis02d:$ ./sas-set-metadata-access -profile admin "/Explore/CPB/CPB An
alytics/LASR Data/TDW_RLS_BESPOKE(Table)" -grant "SASUSERS(UserGroup)":Read -condit
ion "DDSM_RLS_CCHIER IN ('SUB::SAS.IdentityGroups')"
```

Output 3. Applying Conditional Grant Statement via SAS® Batch Metadata Interface

ENSURE

With the RLS Sync and enablement steps completed, all that remains is to *ensure* that the RLS permissions are enacted correctly. LASR™ table permissions, including conditional grant statements can be examined directly within the VA administration interface. The batch metadata interface offers alternative ways to ensure metadata permissions have been correctly assigned. The **sas-show-metadata-access** program is particularly useful for displaying explicit or effective permissions for a user or group (*see output below*). User entitlement reports (UER) may also assist in the verification of the RLS permissions. RLS UER reports can for example be generated, by joining the GRPMEMS table to an RLS LASR™ table, to establish the rows each user can see. Data quality problems and identity integrity issues are typically highlighted from these reports should they not be detected in the RLS Sync process.

```
sasinst1@vstcbis02d:$ ./sas-show-metadata-access -profile admin "/Explore/CPB/CPB A
nalytics/LASR Data/TDW_RLS_BESPOKE(Table)" -onlyGroup "SASUSERS"
```

```
-grant "SASUSERS(UserGroup)":Read
  -condition "DDSM_RLS_CCHIER IN ('SUB::SAS.IdentityGroups')"
```

Output 4. Verifying RLS Permissions for a Group or User

CHALLENGES ENCOUNTERED

There were some challenges encountered during the implementation at RBS.

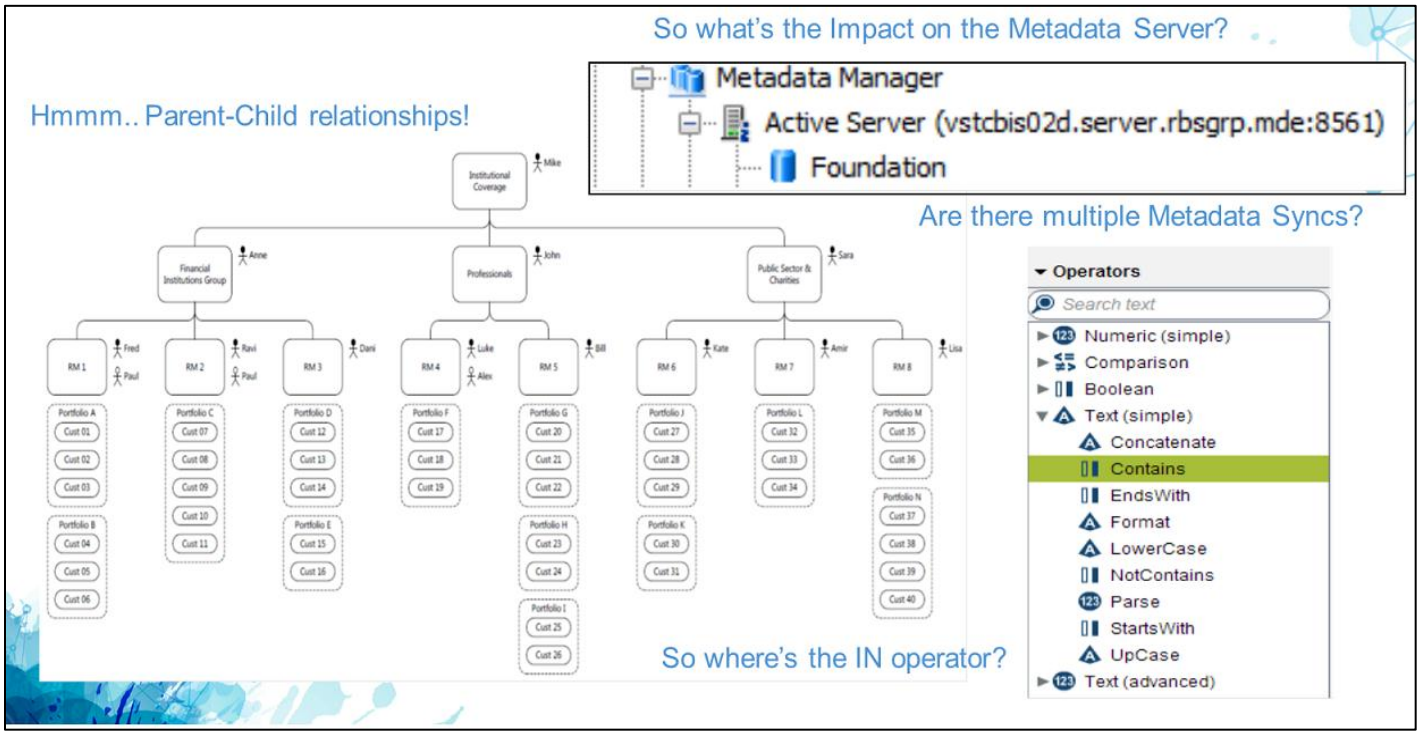
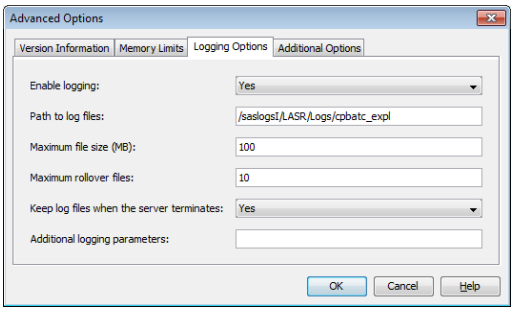


Figure 3. RLS Challenges

Firstly the security database comprised an Oracle schema, which contained Parent-Child hierarchical relationships modeling the multiple levels and cascading nodes within the bank structure. The initial prototype mapped the hierarchical relationships to nested metadata groups and we found performance to be substandard at scale. These were replaced with explicit group member entries, which did lead to a much larger GRPMEMS table but resolved the performance issues observed. Incidentally, the performance of LASR™ tables pre- and post-RLS enablement may be measured by temporarily activating the logging option on the LASR™ server and analyzing the resultant logs (*see logging display below*). Stress tests of the LASR™ server using 30,000 Identity groups have been performed, with no adverse effects recorded.



Display 2. Monitoring Performance via LASR™ Server logging

SAS® users and groups are on-boarded to the platform via a similar metadata synchronization process using the bulk load macros. Exclusion filters were thus needed in both the RLS and on-boarding sync jobs to prevent synchronization issues between the two. The "**IN"** operand was preferred to "**Contains"** in the RLS clauses, despite it not being accessible within the VA administration interface in versions 6.4 to 7.1. However by popular demand, it has now been re-instated in version 7.3.

CURRENT STATUS & BENEFITS

At RBS, there are currently 5 security dimensions comprising 3,500 metadata identity groups securing over 15M rows and 30 GB of LASR™ data through RLS restrictions.

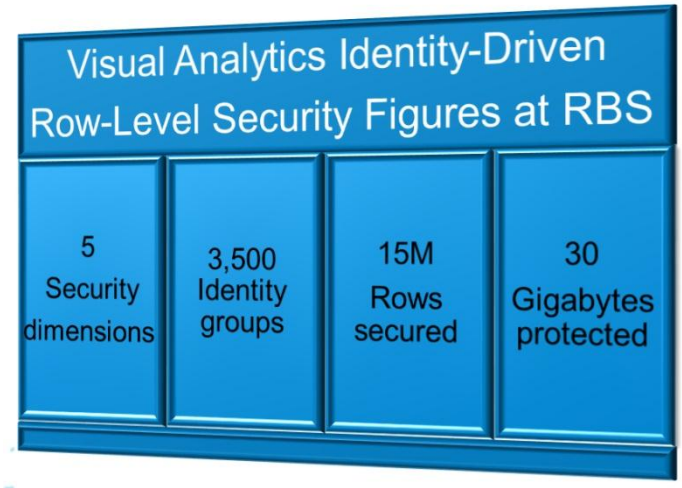


Figure 4. RLS current status

A summary of the benefits realized from the VA RLS implementation at RBS are:

- ✓ An enhanced authorization layer.

- ✓ Less reliance on the SAS® Metadata ACT controls.
- ✓ Consistency with the strategic security model.
- ✓ Higher stakeholder assurance.
- ✓ Minimal data and report duplication in the LASR™ Analytic Server environment.
- ✓ Low support, maintenance and administration overhead.
- ✓ Risk and Audit compliance.

MAKING THE FRAMEWORK END-TO-END

The RLS model in SAS® VA serves well, however we now need to look at integrating this across the wider data and analytics architecture. VA is where most of our end users access data, but it is after all a memory cache. The real data is persisted elsewhere, and as such, any security model we implement will need to accommodate this fact. Plans to integrate the SAS® VA RLS model to the data warehouse security model are now being advanced. In RBS, Teradata hosts the Enterprise Data Warehouse (EDW) and hence provides the key source of data for VA visualizations.

User Access on Teradata EDW is controlled mainly through role based access control on the semantic layer views. These are logical views built on top of the underlying physical data model. There are several role groups that give users access to the logical views on EDW. e.g.: Finance MI, HR etc. These role groups, along with other roles for a user are all stored in the enterprise directory as described in the diagram.

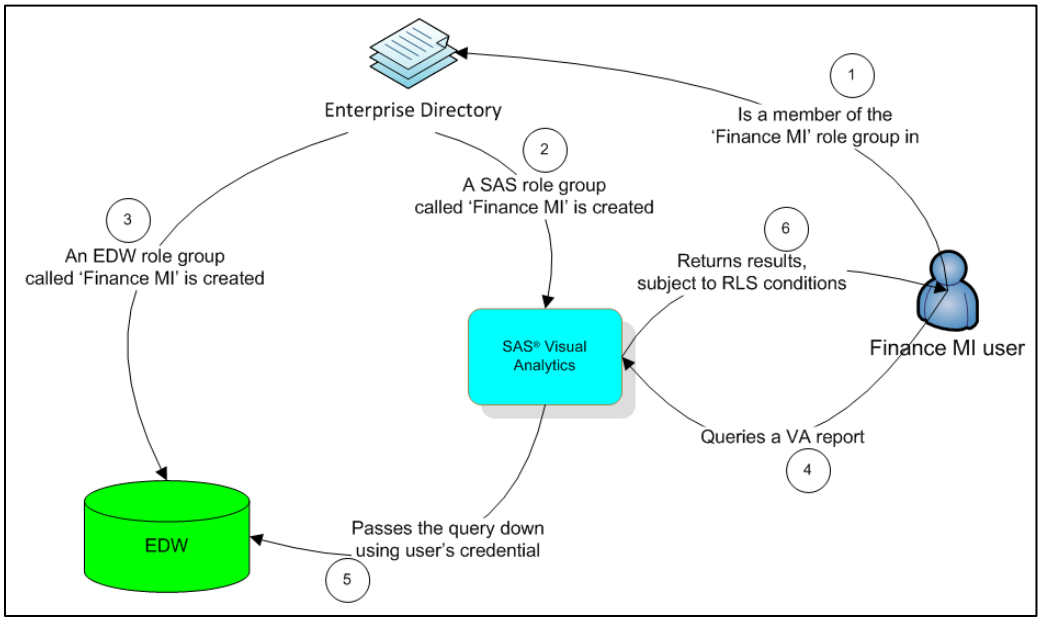


Figure 5. Enterprise RLS Framework

A user could interact with EDW using SAS® VA in the following two scenarios:

1. When a user interactively uses VA to upload a data set from Teradata, SAS® VA uses the end user's credentials to pass the query down to Teradata (see point 5 in the diagram). Hence security is applied at the point of data extract. This applies when other SAS® clients are used to query the Teradata data warehouse as well. If the user is a member of the appropriate role group, they are able to import the data into LASR. Otherwise, they get an access denied error.

2. For a visualization, that uses LASR™ data (in-memory), this scenario is a bit more complex (see points 4 and 6 in Figure 5). This scenario requires the creation of reciprocal Role Based Access Control (RBAC) groups in Teradata and SAS® Metadata with controlled access to both. e.g.: a group called 'Finance MI' will be created in SAS® Metadata, assigned to a folder, where the corresponding Finance MI semantic views will be registered and cached (LASR™ data). The equivalent group will also exist in Teradata. Since we use Microsoft Active Directory (AD) to maintain this authorization information centrally, we will drive the creation of SAS® and Teradata role groups from AD, thus ensuring they are always in sync.

ROW-LEVEL SECURITY IN EDW

Since the adoption of EDW for reporting is still in its infancy across the RBS enterprise, there are as yet no specific RLS requirements for the evolving data repository. Users, who have access to EDW, currently undergo a comprehensive screening process in order to be permitted access, which is only granted within the constraints of RBAC. However, we will discuss our approach to integrate RLS in EDW adopting a consistent framework to the SAS® VA implementation, for when such requirements materialise.

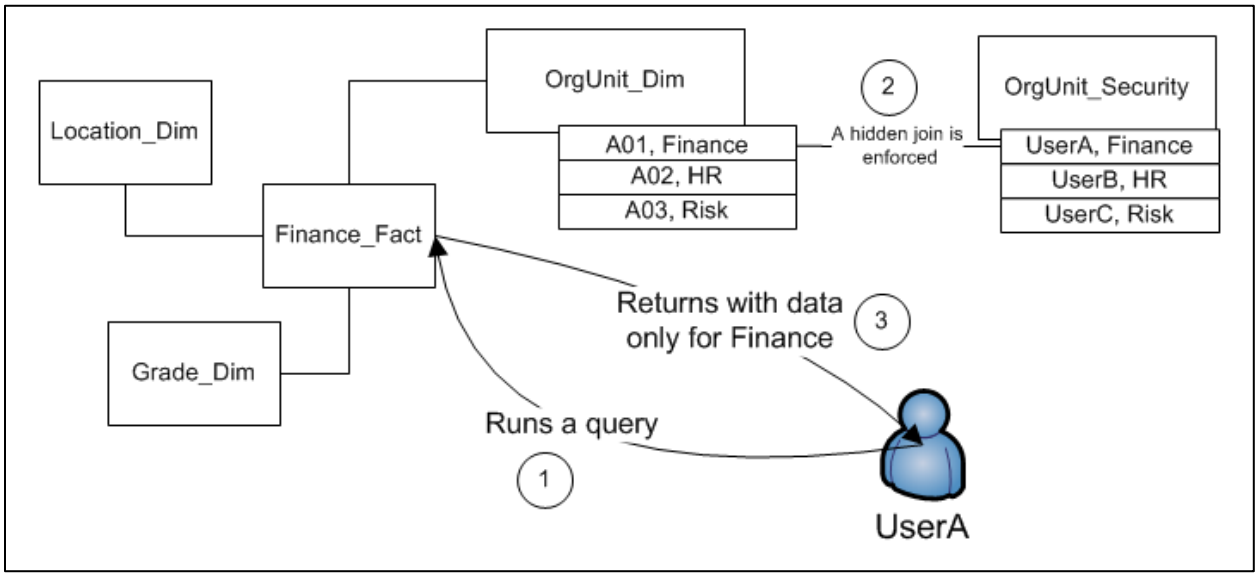


Figure 6. RLS in EDW

As illustrated in the diagram, a typical implementation of RDBMS RLS involves one or more security tables storing the access attributes of a user or a group, associated with dimensions in a star schema. The semantic layer view definition will then include a forced and hidden join (point 2) to these security tables. This join dynamically adds a where predicate based on the user's ID or group. Thus, UserA, who is part of Finance, gets to see only the Finance records in the data model (point 3).

The security tables will therefore serve a dual-purpose, to secure access to data at the row-level, both within EDW (*semantic view*) and further downstream in the SAS® LASR™ Analytic Server (*identity groups*). This is depicted in Figure 7 below.

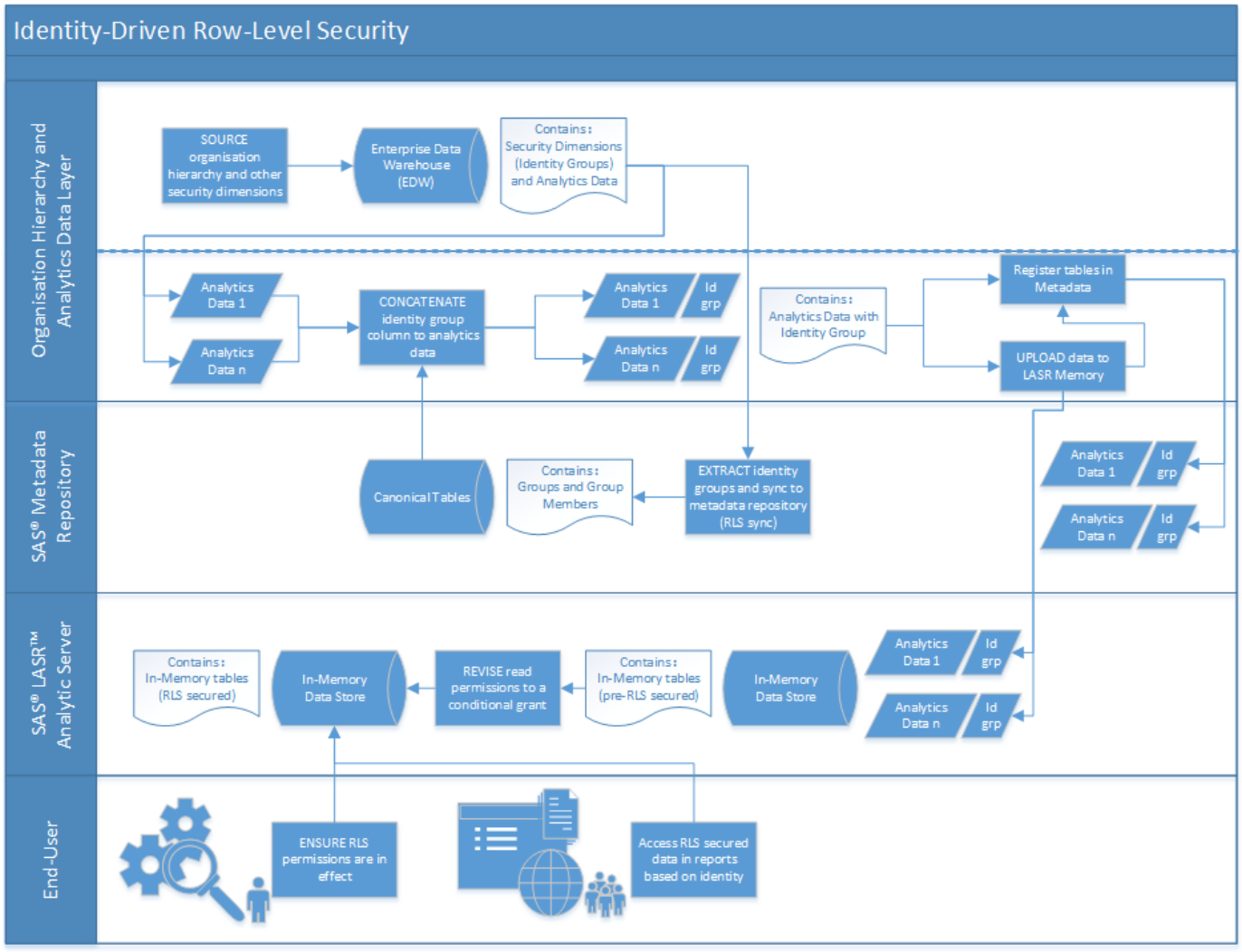


Figure 7. End-To-End Identity-Driven Row-Level Security

FUTURE ROAD MAP AND CONSIDERATIONS

Summer of 2016 will see SAS® Visual Analytics being rolled out to 10,000 users at the Royal Bank of Scotland. This platform will feature SAS® Visual Analytics 7.3, SAS® Visual Statistics 7.3 and a myriad of other SAS® products and solutions deployed in one of the largest platforms of its kind in the world.

Bank-wide Genie, as the platform is known, will be servicing thousands of BI consumers across several departments in the bank. The RLS model described in this paper will be scaled to meet the demands of the new business teams leveraging the power of SAS® Visual Analytics.

With time, our aspiration to integrate the RLS model with the Teradata EDW will start to take shape.

The prospect of integrating RLS in SAS® Visual Analytics with EDW raises an important point for us and for SAS® R&D. In order to implement the above approach, a more effective method would be to use 'the security table' to drive RLS permissions, as opposed to creating SAS® metadata groups. This is currently not possible in VA, but would be a really useful feature, that would deliver the following benefits:

- No support overhead of maintaining additional SAS® metadata groups
- A simpler more transparent RLS process

- Security information is maintained in a single place - where the data is persisted
- Any revisions to the security dimensions would be dynamically reflected in the LASR™ Server

This will be raised as a feature request to SAS R&D.

CONCLUSION

In conclusion, a large scale implementation of Identity-Driven Row-Level Security Using SAS® Visual Analytics benefits from a robust and automated process as well as a mature business operating model.

Prospective users of RLS are encouraged to pay special attention to two key takeaways:

- A well-established security database describing the organizational hierarchy and other security dimensions is desired
- RLS implementations at scale benefit from the use of the SAS® bulk-load auto-call macros, java batch tools and clear segregation between user identities in the platform and those managed and maintained by the RLS process itself

The solution described in this paper makes full use of the current functionality available in SAS® Visual Analytics, to meet the row-level security requirements. The vision to make it an end-to-end framework and roll it out bank-wide, will soon be realized.

ACKNOWLEDGMENTS

The solution architect would like to call out the following people at RBS for their support and encouragement throughout the RLS implementation: James Tattley, Andrew Starr, Jon Tonner, Tony Nealon and Dylan Leong. A special thanks is also extended to the co-authors of this publication.

RECOMMENDED READING

- *SAS(R) Visual Analytics 7.1: Administration Guide*
- *SAS(R) 9.4 Intelligence Platform: Security Administration Guide, Second Edition*

CONTACT INFORMATION

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

Paul Johnson
Sopra Steria
4th Floor
30 Old Broad Street
London, EC2N 1HT
United Kingdom
paul.johnson@soprasteria.com

Dileep Pournami
Data & Analytics, Royal Bank of Scotland
1st Floor Business House E Gogarburn,
Edinburgh, EH12 1HQ
United Kingdom
dileep.pournami@rbs.co.uk

Ekaitz Goienola
SAS UK&I Professional Services
Wittington House, Henley Road
Marlow, Buckinghamshire, SL7 2EB
United Kingdom
ekaitz.goienola@sas.com
