Best Practices for Row Level Security in Tableau with Entitlements Tables

Bryant Howell, Senior Sales Consultant
Miranda Osterheld, Senior Product Consultant
Fearghal Gunning, Sales Consultant
Ron Dugger, Senior Sales Consultant

Row Level Security introduction & overview

Row Level Security (RLS) in Tableau refers to restricting the rows of data a certain user can see in a given workbook or data source at the time they view the data. It contrasts with permissions within Tableau Server (or Tableau Online), which are used to control access to content and feature functionality.

For example, permissions control whether a user can or cannot see (or filter, subscribe to, comment on, etc.) a workbook, while RLS allows two users viewing the same Workbook to see the same dashboards populated with individualized sets of data.

For live connections and multi-table extracts, the RLS workflow can be generalized as the following:

1. User is identified after secure authentication
2. The set of data entitlements for the user is retrieved from all possible data entitlements
3. The data is filtered by that set of data entitlements
4. The filtered data is returned to the user

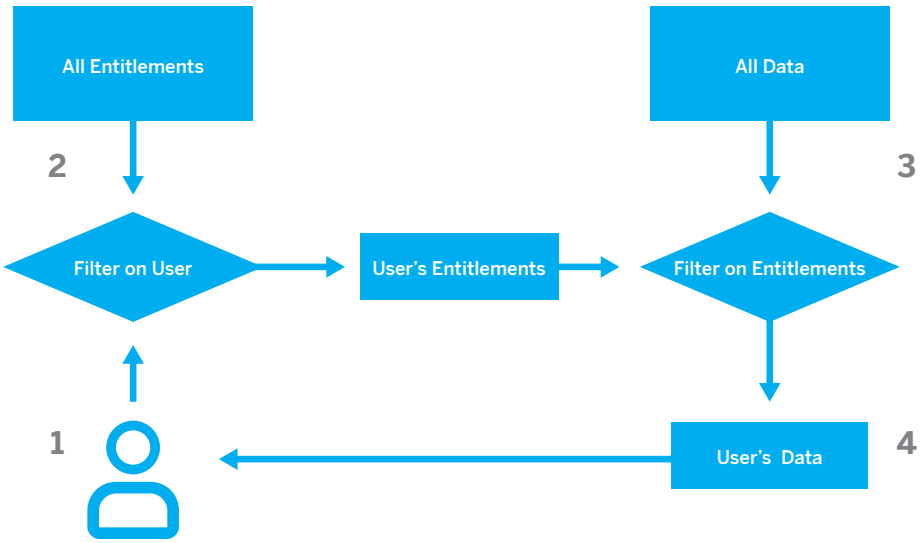

Figure 1 High level process flow for RLS.

A quick note on terminology

Throughout this document, the words "**data source**" in lowercase refer to an actual system for storing data, such as a RBDMS, a RESTful web service, or a text file. "**Tableau Data Source**" in uppercase refers to the description within Tableau of connections to data sources, along with other metadata information. A Tableau Data Source can be part of a .twb file (workbook) or saved separated as a .tds file, which be referred to as a "**TDS**" throughout the document. A Tableau Data Source can also be published to a Tableau Server, in which case it is a "**Published Data Source.**" A "**Calculated Field**" in uppercase refers to calculations created within Tableau.

Row Level Security filtering in Tableau

The best practice recommendation for RLS in Tableau is achieved via the following:

1. A Calculated Field in Tableau Desktop that utilizes a user function
2. A data source filter on that RLS Calculated Field in Tableau Desktop
3. The creator publishes the Tableau Data Source to Tableau Server, separate from the workbook
4. All workbooks connect to a Published Data Source with RLS data source filters

The recommended techniques for achieving RLS in Tableau Server depend on:

1. Secure single sign-on (SSO) of a **distinct username** representing a user
2. The username exists on the Tableau Server with appropriate permissions prior to being logged in
3. The underlying data source has data entitlements that can be linked in some way to the Tableau username

Synchronizing usernames and establishing permissions in the Tableau Server can be automated using Active Directory, LDAP, or the Tableau Server REST API.

If Published Data Sources are not used, the RLS calculated fields can be edited removing all RLS protections. There is a risk that a user may change the RLS Calculated Field(s) in a web authoring session or when they open the workbook in Tableau Desktop. For this reason, Published Data Sources are always recommended unless Web Edit and Download will always be disabled.

What if my data source system already has a Row Level Security mechanism built in?

You are in luck! For many data source types, once a user is logged into Tableau Server, Tableau can pass that username on securely so that all queries automatically use existing RLS rules on those systems. However, this does require using Live Connections rather than Tableau Extracts.

These techniques are covered in the section below called "Utilizing built-in RLS in a data source."

A quick note about OLAP Cubes

OLAP Cube connections in Tableau do not have the equivalent of a data source filter, which is required for the entitlements table-based RLS method in Tableau, or access to the USERNAME() function. For these reasons, Kerberos and constrained delegation is a recommended approach to RLS with OLAP data sources, which allows Tableau to leverage user filtering that has already been implemented on the OLAP Server side.

If the users viewing the dashboard will not be part of the domain, then the manual approach to creating user filters is possible. However, because the User Filter Set generated cannot be added as a data source filter, and will instead exist on the filters shelf, it is important that Web Editing and Download Workbook functionality is not permissible for any published views utilizing this method.

What if the recommended mechanism of Row Level Security does not meet my needs?

If your setup does not involve individual users logging into Tableau or standard supported data sources, other more advanced methods may need to be utilized. Examples of more advanced setups include connecting to stored procedures as a data source, needing to pass programmatic filters at report load time without logging in individual users, connecting to RESTful APIs as data sources, etc.; RLS Methods that address these more advanced scenarios are not covered in this document.

What if I don't have any sort of data source RLS or entitlements in place yet?

If you're just getting started with trying to build out a row level secured data system, a great first step is to read this document on how to set up Data Entitlements in your data source systems, and implement a method that makes sense for you.

Additionally, Tableau Desktop has a "Create User Filter" option in the Server menu, which allows you to manually assign users and groups to values within a particular field. While this method is secure, most organizations may find this option limited and not scalable, because the result is a hardcoded set that can then be added as a filter. Any sort of update will require going back into Desktop and editing the set manually.

There is a matrix at the end of this document that provides a side-by-side comparison of each of the approaches described in the following sections.

Contents

Utilizing built-in RLS in a data source

As mentioned in the introduction, many data sources have mechanisms for RLS built in. If your organization has already put effort into building out RLS in a data source, you may be able to use one of the following techniques to take advantage of your existing RLS. In order to leverage the data source's security models, live connections are required. Additionally, these techniques are likely not available in Tableau Online; the Tableau username for Online is a unique email address which is not typically the user identity on the data source side.

It is not necessarily easier or better to implement a built-in RLS model vs. building it with Tableau in mind; these techniques are generally leveraged when an organization has already invested in these technologies and they want to take advantage of the investment.

Examples of this would be using Impersonate or Run As for Microsoft SQL Server, constrained delegation using Kerberos or SAML for HANA, or Initial SQL to use VPD in Oracle. Because they require that users with RLS applied also exist as users within the database, generally these techniques aren't leveraged if the data is going to be accessed by users external to the organization. The main benefit of this feature is that it allows administrators to implement and control their data security policy in one place: their databases.

Impersonation (Microsoft SQL Server)

Microsoft SQL Server (and a few related systems) can be configured so that users of the database only have access to views with RLS filters built in, either utilizing Security Junction Tables or views built by the DBA. Tableau can take advantage of this using a concept called "impersonation."

When publishing a Tableau Data Source containing an MS SQL Server connection to Tableau Server, there are two authentication options available to take advantage of impersonation. The menu you see will depend on whether you logged into the SQL Server using network authentication or if you entered in username/password credentials.

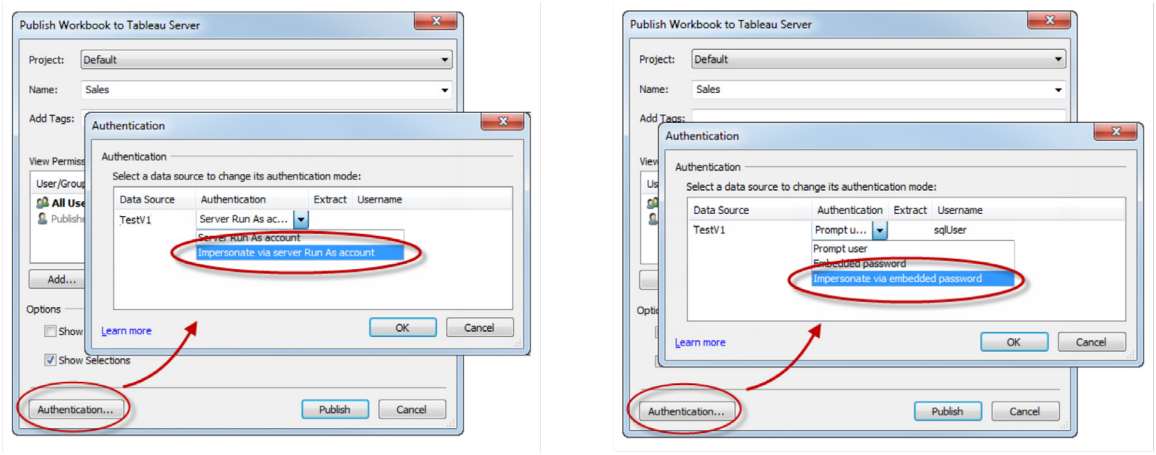

Figure 2 Authentication with impersonation.

To enable RLS filtering for any user who can access the Published Data Source in Tableau Server, either the AD Run-As Account or the embedded SQL server credentials must have permission to EXECUTE AS for all of the Tableau users in the database that will be accessing the dashboard or data source. All Tableau users must exist in the database server as users, with SELECT rights for the Views you are trying to connect to (and have RLS applied to). See the comprehensive list of requirements in the Tableau Online Help article.

Kerberos & constrained delegation

Constrained delegation within Tableau Server using Kerberos operates similarly to impersonation in that it allows Tableau Server to use the Kerberos credentials of the view of a workbook or view to execute a query on behalf of the viewer, so if RLS is set up on the data source, the viewer of the workbook will only see their data.

To see the comprehensive list of data sources where Kerberos delegation is supported, see the Tableau Online Help article. Active Directory is required; the computer where Tableau Server is installed must be joined to the Active Directory domain. The authentication method specified when publishing the data source is viewer credentials.

It is worth noting that Kerberos can be leveraged for RLS when using Microsoft Analysis Services.

SAML delegation & SAP HANA

If Tableau Server is configured to use SAML delegation with SAP HANA to provide a single sign-on experience, the viewer credentials are used to execute the query as that user, which will operate within whatever security is applied on the user level. The functionality operates much like impersonation and Kerberos Delegation, except the SAML token specifies the user. The authentication method specified when publishing the data source is Viewer Credentials.

Initial SQL to force a user-specific session (Oracle VPD)

Initial SQL allows you to specify a SQL command that is run when the connection is made to the database for the purpose of setting up temporary tables to use during the session or to set up a custom data environment.

For Oracle VPD, you can set up a session specific to a user by running a particular stored procedure or function to set the context of the database connection to match the Tableau user's username:

```
begin

DBMS_SESSION.SET_IDENTIFIER([TableauServerUser]);

end;
```

The same high-level requirements hold true for using this for RLS as with impersonation; the DBA must set up VPD and all of the associated users to exist on the database.

On MS SQL Server, you could force an EXECUTE as command (however, this is similar to what Tableau does with impersonation already):

```
EXECUTE AS USER = [TableauServerUser] WITH NO REVERT;
```

Implementing RLS in a Relational Database

The recommended method of RLS in Tableau uses a join between the data view and the entitlements view. This is recommended because the same technique can be used with live connections and multi-table extracts (covered in depth later).

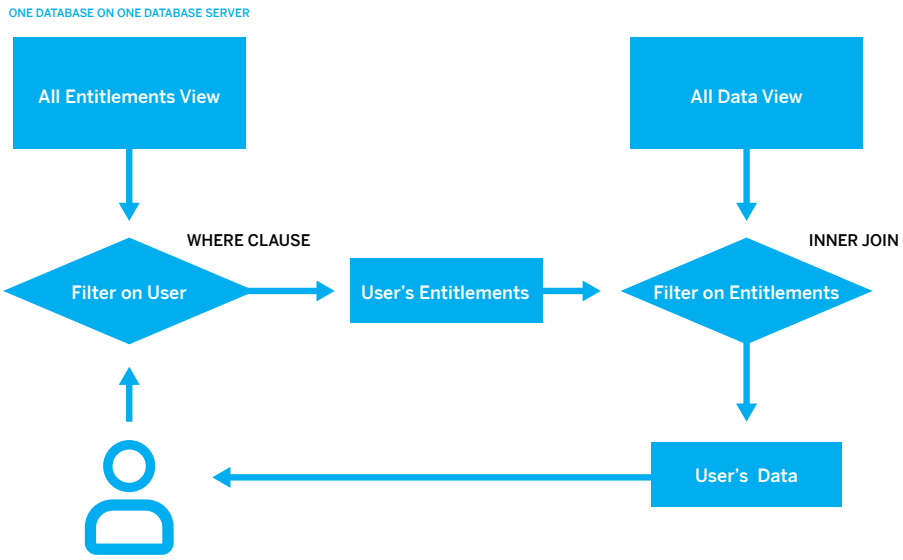

Figure 3 Recommended Row Level Security flow in Tableau.

The workflow is implemented by joining an entitlements view, which is filtered down to just a particular user's entitlements based on a WHERE clause. In Tableau, the WHERE clause is created via a data source filter. Because the entitlements view is joined to the data view on attribute columns of the data view, the inner join reduces the rows to only those that the user is entitled to see. The inner join can be on all of the fields that make up a unique entitlement (for example, Theater_ID, Region_ID, SubRegion_ID, etc), or the join could be between a compound or a synthetic/surrogate key if it also exists in the fact table. The actual design details and their performance characteristics will depend on the data system and will require testing. For example, using a single key could potentially improve the performance because the join is then only executing on one column, but correctly indexing all of the columns may give equal performance when other factors are taken into consideration.

Security entitlements in a relational database

This section will cover techniques, considerations, and general recommendations for deploying and maintaining the entitlements model in a relational database (or text or Excel files configured in a similar manner). Actual application of these concepts for usage in Tableau (in both live and extract connections) will be covered in the next section. Please note that the concepts discussed assume some prior knowledge of database administration, and these are simply conceptual examples for the purpose of illustrating the possibilities these types of models will enable. It is up to your organization to apply these concepts to your own security requirements.

Generally, an entitlement is the conceptual building block that will eventually tie these users to the rows they should be allowed to see. An entitlement is a single unique combination of attributes that the data will be filtered on.

Entitlement ID	Category	Sub-Category	Region
FUR-BOO-N	Furniture	Bookshelves	North
FUR-BOO-S	Furniture	Bookshelves	South
FUR-CHA-N	Furniture	Chairs	North

Figure 4 Any unique combination of Category, Sub-Category, and Region is an entitlement.

In relational databases, entitlements are typically stored in a set of normalized tables, that are eventually brought together in a query, view, or denormalized table in order to use the set of entitlements for filtering. Denormalizing is recommended from a performance standpoint; joins are expensive operations and denormalizing often makes for much faster analytical queries, since all the data is in the table itself.

Additionally, it's uncommon for entitlements to be assigned directly to a user; for example, most entitlement tables do not look like Figure 5.

Region	User
North	rdugger
East	mosterheld
West	fgunning
South	bhowell

Figure 5 Generally, entitlements are not represented by so simple a model.

More often, combinations of entitlements are represented as "roles", which are then linked to users in a many-to-many mapping table. An example set of entitlements common to many organizations are sales territories:

EMEA-AFRICA-SA-E-I-ENA-1

USCA-CA-CA2-E-F-SNA-2

LAC-MX-MX1-C-I-TER-3

Sales territories will used to exemplify the concepts throughout this section.

Generally, there are two primary models for representing entitlements:

1. Full mapping to the deepest level of granularity:
    a. Entitlements are defined fully for every column.
    b. There is one row in the mapping table for every possible entitlement the user has.
    c. This model requires fewer join clauses.

2. Sparse entitlements:
    a. Entitlements are defined for every level of hierarchy, with NULL used to represent an "all" state.
    b. There is a single row in the mapping table for a particular level in the entitlement hierarchy.
    c. This model vastly reduces the number of entitlement rows for users at high levels in a hierarchy.

Full mapping to deepest level of granularity

The full mapping entitlement table example is shown below in Figure 6.

entitlement_id	region_id	sub_region_id	country_id	country_name
APAC-JPN-JP	APAC	JPN	JP	Japan
APAC-SEA-KH	APAC	SEA	KH	Cambodia
APAC-SEA-ID	APAC	SEA	ID	Indonesia
APAC-SEA-MY	APAC	SEA	MY	Malaysia
APAC-SEA-MM	APAC	SEA	MM	Myanmar (Burma)
APAC-SEA-PH	APAC	SEA	PH	Philippines
APAC-SEA-TH	APAC	SEA	TH	Thailand
APAC-SEA-VN	APAC	SEA	VN	Vietnam
APAC-SK-KR	APAC	SK	KR	South Korea

Figure 6 Deepest granularity denormalized table of potential entitlements.

For secure user filtering to work in Tableau, Tableau must be able to get the set of entitlements based on the Username as recorded in Tableau Server or Online. However, it is rare for there to be a direct linking of entitlements for each user in a data source. Instead, most systems relate a set of entitlements to a "role", then the user is separately linked to a role (or roles). This allows for easily changing or removing a user from the role, while still maintaining a record of the role (and its entitlements).

In this structure, you would have a roles table (Figure 7), which defines the absolute set of roles:

role_id	role_name
RD-APAC	APAC Regional Director
SRD-APAC-SEA	Sub-Regional Director, South East Asia
CD-APAC-JPN-JP	Country Director, Japan
CD-APAC-SK-KR	Country Director, South Korea
CD-APAC-SEA-TH	Country Director, Thailand

Figure 7 A roles table.

There is a separate table that relates each Role ID to the Relevant Entitlement IDs. This is a many-to-many table which can contain many instances of roles and entitlements.

The deepest granularity role-to-entitlement mapping table example is shown in Figure 8.

role_name	rol_id
RD-APAC	APAC-JPN-JP
RD-APAC	APAC-SEA-KH
RD-APAC	APAC-SEA-ID
RD-APAC	APAC-SEA-MY
RD-APAC	APAC-SEA-MM
RD-APAC	APAC-SEA-PH
RD-APAC	APAC-SEA-TH
RD-APAC	APAC-SEA-VN
RD-APAC	APAC-SK-KR
SRD-APAC-SEA	APAC-SEA-KH
SRD-APAC-SEA	APAC-SEA-ID
SRD-APAC-SEA	APAC-SEA-MY
SRD-APAC-SEA	APAC-SEA-MM
SRD-APAC-SEA	APAC-SEA-PH
SRD-APAC-SEA	APAC-SEA-TH
SRD-APAC-SEA	APAC-SEA-VN
CD-APAC-JPN-JP	APAC-JPN-JP
CD-APAC-SK-KR	APAC-SK-KR
CD-APAC-SEA-TH	APAC-SEA-TH

Figure 8 A deepest granularity mapping table, roles to entitlements.

Sparse entitlements

Alternatively, when a sparse entitlements table (Figure 9) is deployed, there are entries for every level in the hierarchy, not just for the entries at the most granular level. Null values in columns represent having access to all the data at and below that level.

Role_ID	Theatre_ID	Region_ID	Sub_Region_ID	District_ID
ALL	NULL	NULL	NULL	NULL
EMEA	EMEA	EMEA	NULL	NULL
EMEA-AFRICA	EMEA	EMEA	AFRICA	NULL
EMEA-AFRICA-SA	EMEA	EMEA	AFRICA	SA
AMER	AMER	NULL	NULL	NULL
USCA	AMER	USCA	NULL	NULL
USCA-CA	AMER	USCA	CA	NULL
USCA-CA-CA2	AMER	USCA	CA	CA2

Figure 9 A sparse entitlements table.

The sparse entitlements mapping table (Figure 10) is generally much smaller. It also makes it easy to find specific levels within the hierarchy (and the people who occupy those levels). The sparse entitlements method requires fewer rows in the roles-to-entitlements table, but at the cost of more complexity in joining and filtering.

role_id	entitlement_id
RD-APAC	APAC
SRD-APAC-SEA	APAC-SEA
CD-APAC-JPN-JP	APAC-JPN-JP
CD-APAC-SK-KR	APAC-SK-KR
CD-APAC-SEA-TH	APAC-SEA-TH

Figure 10 The role mapping table for a sparse entitlements model.

Once each role is tied to one or more entitlements, now those roles need to be tied to particular users using another many-to-many mapping table. The value stored in the username column shown below in Figure 11 needs to either match the Username or Full Name in Tableau Server or Online in order to take advantage of the user functions in Tableau Desktop. If necessary, string calculations are available within Tableau Desktop to manipulate the values.

role_id	username
RD-APAC	regional_director@company.com
SRD-APAC-SEA	sub_regional_director@company.com
CD-APAC-JPN-JP	japan_manager@company.com
CD-APAC-SK-KR	south_korea_manager@company.com
CD-APAC-SEA-TH	thailand_manager@company.com

Figure 11 A many-to-many mapping table for roles to users.

Alternatively, a many-to-many mapping table can be created that instead assigns users directly to entitlements (Figure 12) as opposed to going through joining a role table. It will require managing the values more directly in the table but does eliminate a join.

Mapping_ID	Username	Role_ID	Company_Role
10001	d.rector	AMER-USCA-CA-CA1	Director
10002	d.rector	AMER-USCA-CA-CA2	Director
10003	d.rector	AMER-LAC-MX-MX1	Director
10004	a.canadien	AMER-USCA-CA-CA1	Regional Manager
10005	a.canadien	AMER-USCA-CA-CA2	Regional Manager
10006	j.data	AMER-USCA-CA-CA1	Account Manager

Figure 12 A directly mapped user-to-role/entitlement table.

Choosing between the deepest granularity and sparse entitlements

There can be performance benefits if you use the deepest granularity method and everything is hierarchical—you only need to do a single join, on the deepest level of the hierarchy (since the higher levels of the hierarchy are implied by the deepest level).

This only works if all of the attributes at the lowest level are distinct: if you are joining on a unique ID, then you would be clear just to join on that deepest level attribute, but if there is a chance for duplication in the real values (for example, a Central sub-region under both AMER and EMEA), then you'll need to join on all the columns to achieve the effect of a distinct key value.

Lastly, regardless of the model used to represent the entitlements, it is advisable to join all entitlements/mapping tables together into a single denormalized entitlements view. While at first this will cause a "blowup" (highly duplicative) version of the entitlements, the entitlements should be filtered down to just those belonging to a particular user before it gets joined into the data. You will also want this view if you plan on using an extract (covered later).

The next section covers how these concepts are leveraged and implemented within the Tableau Platform.

Recommended mechanism for Row Level Security with live connections

There are several methods for implementing RLS using a live connection. The majority of them are based on the concept of using the user functions in Tableau to match against a column in a database, so that the resulting rows limit the data that can be seen.

A note on terminology: Throughout the following sections, the word "view" is utilized. In this scenario, it means any of the following in a relational database: a table, a view, a materialized view or indexed view; or any other possible arrangement in a database that looks like a single Table when connected to using Tableau.

The next section will talk about how to filter the data by the entitlements using the two models referenced in the previous section, and how to set up your filters (WHERE clauses) for each model in Tableau.

Filtering the data by the entitlements — deepest granularity method

Once the denormalized entitlements-users-roles mapped view exists, an inner join is set up between the view and the data in the Tableau data connection dialog. The data (fact/dimension tables) can remain in a traditional star schema, like in Figure 13:

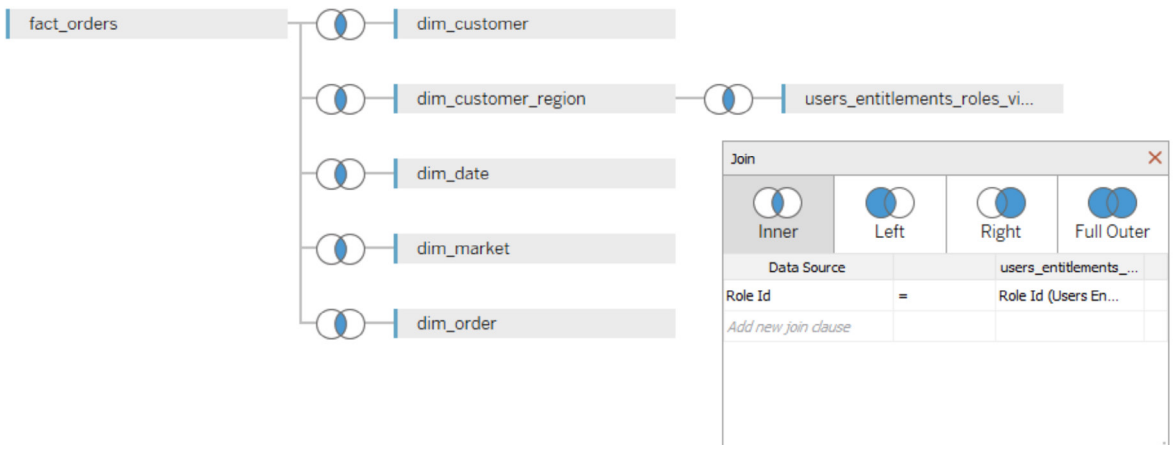

Figure 13 Example join structure in the Tableau Desktop data connection dialog.

Alternatively, the dimension and fact tables can be materialized together into two views as shown in Figure 14. Multi-table extracts will build extract tables to match the joins as depicted in this dialog, so creating the two views will simplify the resulting extract.

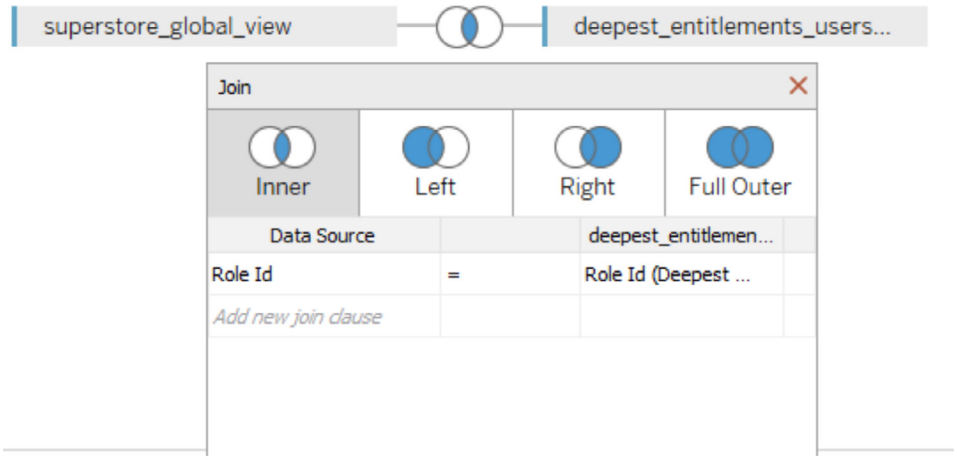

Figure 14 The Superstore Global View brings all of the global superstore star schema together, which is then joined directly to the view that has the entitlements, roles, and users materialized.

The SQL will follow the basic pattern in Figure 15 (although VizQL will produce whatever complex queries it needs).

```
SELECT *

FROM data d

INNER JOIN entitlements e ON

d.attribute_a = e.attribute_a AND

d.attribute_b = e.attribute_b AND ...

WHERE e.username = USERNAME()
```

Figure 15 Example conceptual SQL for the deepest granularity method.

Once you've set up the data, the next step is to set up a Calculated Field that utilizes the user functions to compare to a column in your dataset. For example, Figure 16 shows a simple Boolean comparison of whether the user listed in the Username field column is the same as the username of the person logged into the Tableau Server.

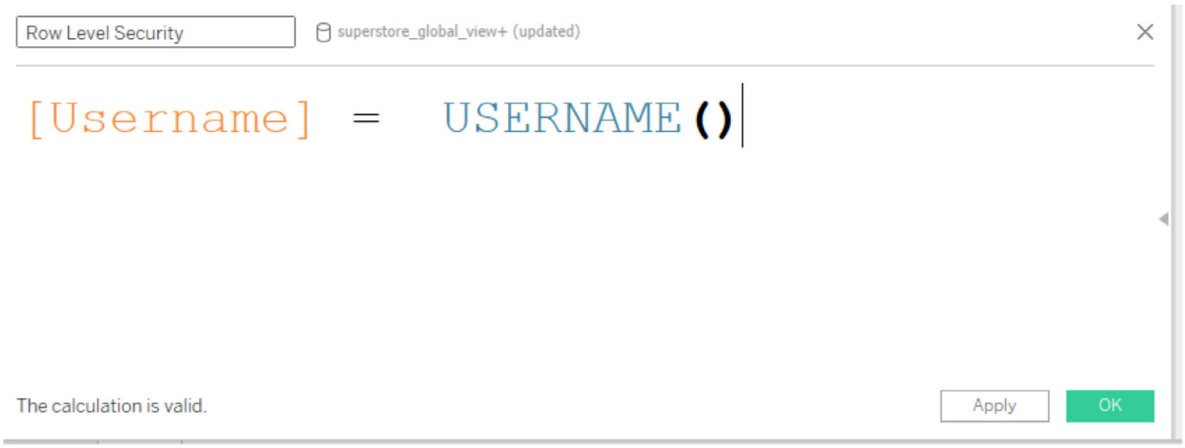

Figure 16 Tableau Calculated Field utilizing the USERNAME() function.

As mentioned in in the introduction, this should be applied as a data source filter (with TRUE selected) and in most cases the Tableau Data Source should be published separately as opposed to being left embedded in the workbook. If the data source is left embedded in the workbook and a user is permissioned to Web Edit or Download the workbook then the RLS is essentially nonexistent since the calculation enforcing it can be easily removed.

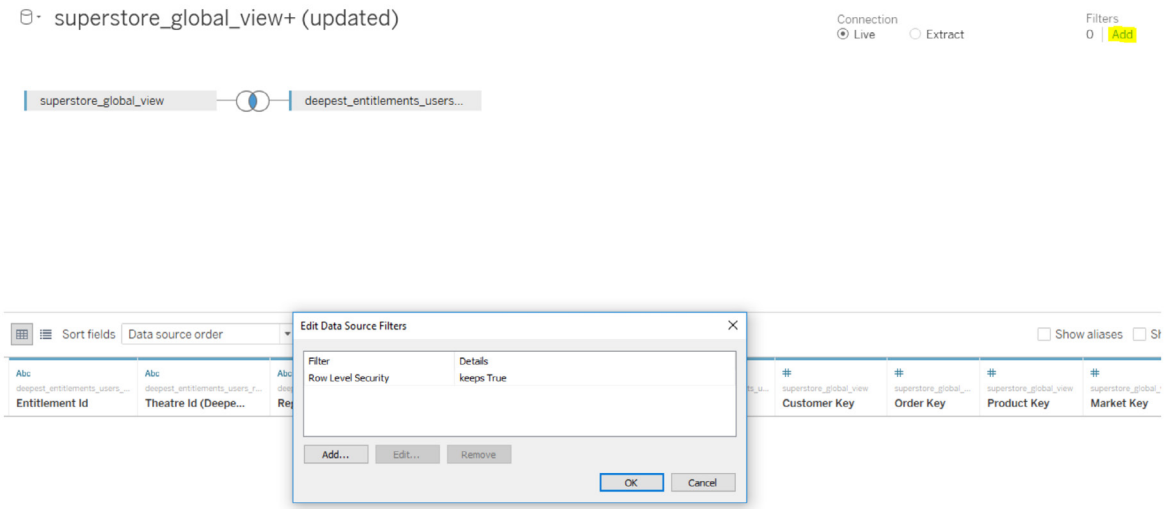

Figure 17 Adding a data source filter.

Filtering the data by the entitlements — sparse entitlements method

If your entitlements more closely resemble the sparse entitlements model, then the SQL to join the data to the entitlements would be a little more complex because of the NULL values. Conceptually, it would look like Figure 18.

```
SELECT *
FROM data d
INNER JOIN entitlements e ON
(e.region_id = d.region_id OR ISNULL(e.region_id) AND
(e.sub_region_id = d.sub_region_id OR ISNULL(e.sub_region_id) AND
(e.country_id = d.country_id OR ISNULL(e.country_id)
```

Figure 18 Example conceptual SQL for the sparse entitlements method.

You could pass SQL resembling this format directly via custom SQL, but to recreate it via Tableau's join dialogue requires join calculations (Figure 19). Because only simple equality join definitions (=, <>, <, >, <=, >=) are available when defining a join in desktop, a join calculation is necessary to essentially create a cross join, where every row in one view is joined to every other view in the other table after the WHERE clause is applied.

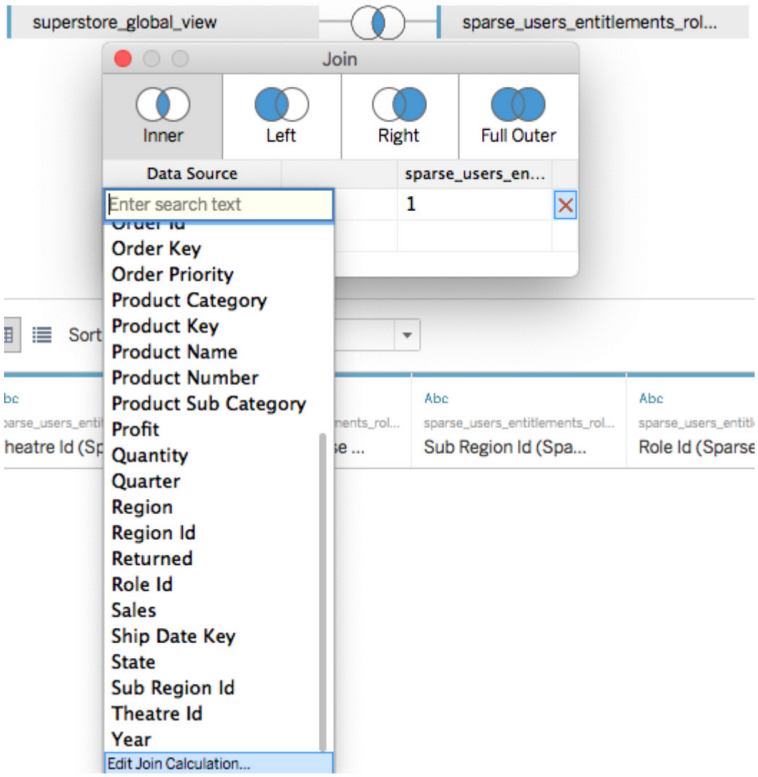

Figure 19 Create a join calculation on either side of the join dialog.

The calculation you create on either side of the join is simply the integer value of 1.

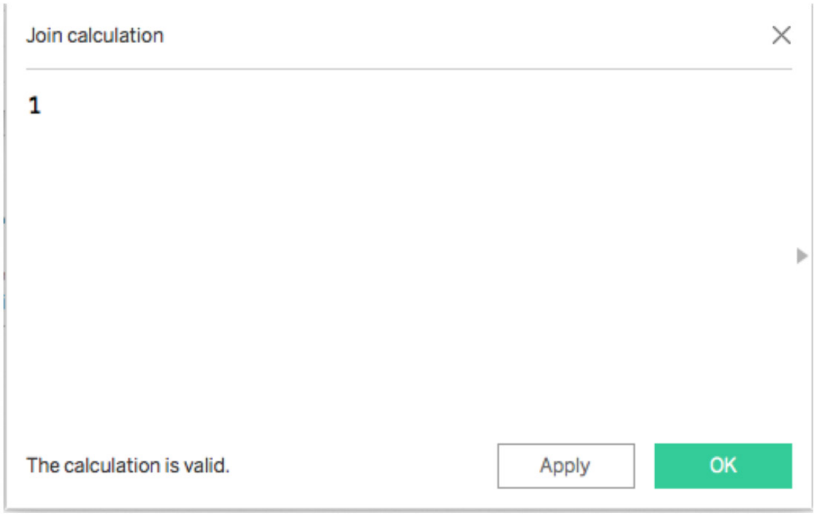

Figure 20 The join calculation with the integer value 1.

Create this "calculation" on both sides and make them equal. Now, just as with the deepest granularity model, you need to go define Calculated Fields to set up our RLS. The first is the username Calculated Field, which will reduce the total rows of entitlements:

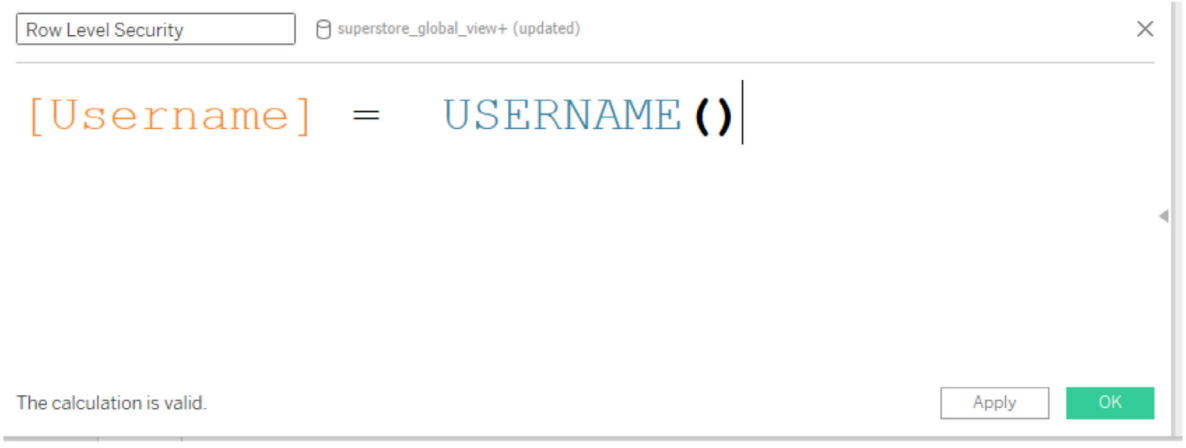

Figure 21 Username Calculated Field.

Then you need a calculation (or individual calculations) to account for the levels in the hierarchy. For example, you could have several calculations that follow this format:

```
[region_id] = [region_id (Entitlements View)] OR ISNULL([region_id
(Entitlements View)]
```

Or you could have a combined calculation for all levels in one:

```
([region_id] = [region_id (Entitlements View)] OR ISNULL([region_id
(Entitlements View)])
```

AND

```
([sub_region_id] = [sub_region_id (Entitlements View)] OR ISNULL([sub_region_
id (Entitlements View)])
```

AND

```
([country_id] = [country_id (Entitlements View)] OR ISNULL([country_id
(Entitlements View)])
```

The ISNULL function provides the ability to match up any entitlement column to all items in the other column.

Just like in the deepest granularity model, these Calculated Fields should be added as data source filters.

"All access" or "deepest granularity" methods for filtering

There is also a common scenario in which there are two access levels within the organization: people who can see everything ("all access") or people with some reasonably definable subset of entitlements ("deepest granularity" above). This is mostly commonly seen for embedded applications—the organization hosting the data can see everything, but each client can only see their own data. In this case, you need a way to circumvent a join for the "all access" users, while maintaining the deepest granularity joins for all other users.

For this technique, you will use Tableau's Groups functionality to give an "override" in the join calculation. This group will be called "Admins" and it will contain users who get to see the entirety of the data.

From the fact view, you will do a left join, then two join conditions. The first join should join on the column that represents the deepest level of granularity (in our example, country_id).

Then create a second join condition, with a join calculation on each side (similar to the method described above in the sparse entitlements technique). On the left side (the fact view), simply set the calculation value to:

```
True
```

Then on the right side (the entitlements view), the calculation should be:

```
IF ISMEMBEROF('Admins') THEN False

ELSE True

END
```

If a user is a member of the Admins group on Tableau Server, then the join becomes a left join on TRUE = FALSE. This means there are no matches at all in the entitlements view, so in a left join situation you get the entirety of the fact view and all NULLs for the columns from the entitlements view (zero duplication). In the case where the user is not part of the "Admins" Tableau group, the column will match up with the values in the entitlements view on the field representing the deepest granularity of the hierarchy (and the TRUE = TRUE doesn't affect anything at all).

Then you go into a sheet, and create a Calculated Field for the user, which you will make a data source filter:

```
[Username] = USERNAME() OR ISNULL([country_id] (Entitlements View)])
```

This will evaluate true for all rows when the group override is working, or it will filter down to only the user's deepest granularity in the hierarchy (country_id in this example).

Note on performance in live connections — processing order of operations

The joins and Calculated Fields used as filters built in Tableau Desktop when you are defining your data source provide direction to Tableau about the relationships and filters to be sent to the data source. When the visualization is viewed in Tableau (Desktop, Server, or Online), Tableau writes an optimized query, which the RDBMS system receives, processes, and then sends results back to Tableau to render the visualization with the resulting data. When the Tableau Data Source is a live connection, rather than an extract, this process happens for every query that is necessary to render a particular viz or dashboard. When a data source is an extract, the process of querying data from the underlying data source only happens at extract creation and refresh time, and all of the individual queries for visualizations are answered by the extract engine from the extract file.

When a query is received by a relational database, there is a query optimizer which translates the incoming SQL into an efficient plan for retrieving the data. Thus, when using a live connection to a data source in Tableau, the performance of the query execution is greatly dependent on the query optimizer in the relational database. Otherwise, the result of many-to-many relationships across tables joined together can result in huge data expansion and duplication during the processing of the query (and possibly in the results if the joins and Calculated Fields are not operating correctly).

In an ideal situation, the query optimizer will utilize database optimizations such as primary and foreign key relationships to process the resulting query in something like the following order:

1. Determine the entitlement rows, filtered on the username.
2. Join that filtered set of entitlement rows to the fact table using available Indexes and key relationships.
3. If a user is entitled to everything, the maximum number of rows = rows in data table.

If the database processes the query in the following order, there will be a "blowup:"

1. Join all of the entitlement rows to the fact table.
2. The maximum possible size = the number of entitled users * rows in the fact table (every user entitled to see that particular row * each row).
3. Filter down to user's rows.

It will be very clear if this second scenario happens—if your queries are taking inordinately long to finish, you get errors, or your DBA sends you a nasty email for ruining performance in the database. Your total data volume will expand exponentially which could cause inordinate system strain on the backend. If the RLS calculations are correct, the eventual result set received by the end user will be correct regardless of which way the data source processes the queries—the only difference will be whether the user sees those results in seconds or in minutes.

The exact same issue is present when building single table extracts, only the issue is twice as bad, because the "blowup" will happen both on the underlying data source and within the resulting extract itself.

Recommended mechanism for Row Level Security with extracts

Multiple table extracts

Starting in Tableau 2018.3, the data engine can create a multi-table extract, which facilitates implementing the recommended RLS pattern described in this document. Using multiple table extracts vastly reduces the time it takes to generate an extract with many-to-many relationships by completely avoiding the requirement to materialize the entirety of a join.

Previously, the materialization of the join had the risk of causing massive data duplication, as every row that was allocated to more than one entitlement or user would be duplicated as a result of the "blowup" caused by the many-to-many relationships. Now the join prior to extracting can be avoided entirely.

The following features are not available to multi-table extracts:

- Number of rows options:
  - Incremental extract (the load times for Hyper extracts are so much faster than TDE that this shouldn't be an issue, even on large data sets)
  - Top
  - Sample
- Aggregation
- Filters

In a single table extract, you can define "extract filters" which are translated into WHERE clauses on the SQL query that retrieves the extract data. These are different from data source filters, which are applied at the time the viz is generated, but before any other filters.

Because multiple table extracts have extract filters disabled, you must do your filtering either in the views or tables you connect to in the data source, or define the filters in custom SQL objects in the Tableau data connection dialog. The extract will have the entire scope of the dataset included, so publishing the data source separately (as opposed to keeping it embedded) is necessary unless permissions will be diligently applied to ensure users are not able to download or edit the workbook in any way.

Design best practices for multiple table extracts

The recommendation for a multi-table extract for the purposes of RLS is to only have two "tables:" a data view and an entitlements view. This will be the simplest storage in the extract and result in the best performance.

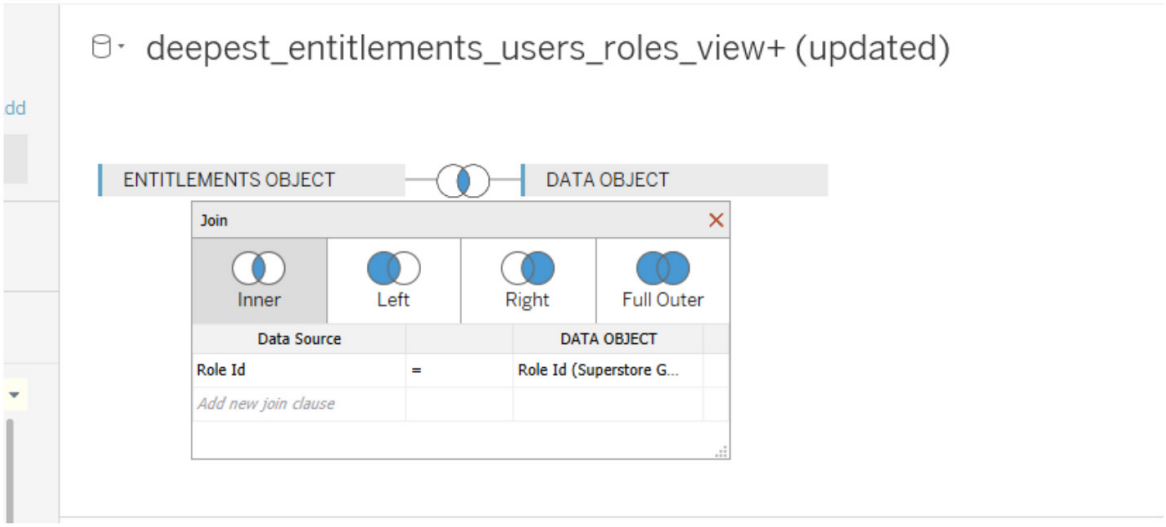

Figure 22 The ideal format for a multi-table extract used for RLS.

The "entitlements object" is a denormalized table, view or custom SQL query of whatever entitlements are necessary to filter the data at the most granular level, which requires:

- A column for Username matching the exact usernames in Tableau Server or Online
- A row for each of the most granular entitlements to the data object

This format is laid out in the "deepest granularity method" explained in the "Security entitlements in a relational database" section; in fact, that format is followed exactly in multi-table extracts, with the caveat of ensuring that only two data objects are being joined, and with any field-specific filtering already applied within the object.

The "data object" is the table, view or custom SQL query that represents the denormalized combination of the fact and necessary dimension tables.

Single table extracts

The following method is only recommending when using a version of Tableau prior to 2018.3.

Single table Tableau extracts take whatever table relationships from the live connection screen and combine them into a single table through one query, the results of which are transformed in a single table in the extract file. This process is technically called "denormalization," and the materialization of the join had the risk of causing massive data duplication, as every row that was allocated to more than one entitlement or user would be duplicated as a result of the many-to-many relationship.

To prevent this data "blowup," the recommended route leverages the CONTAINS() function within Tableau. Note that this technique requires creative data preparation in SQL, but it does not result in any row duplication. Essentially, this technique requires you to include the list of allowed usernames into a column that then gets parsed via the CONTAINS() function:

```
CONTAINS([Security Users Field], USERNAME())
```

```
The Security Users Field contains values like this:
```

```
bhowell|mosterheld|rdugger|...
```

This method obviously has some caveats. It requires that you go from your entitlements in rows to a single column separated correctly using SQL, and that column can only contain so many characters. Partial matches can be trouble, and you need to ensure to use separators that will never be valid in the IDs themselves. Although it is performant within the Tableau Data Engine, as a string calculation it will be very slow for most databases. This will limit the ability to switch back to a live connection. When at all possible, it is best practice to leverage the multiple table extract method.

Of course, there is always the option to take different extracts per "client" or entitlement level, so that only the data appropriate to that person or level is contained within the extract, but this will require processes to appropriately permission and leverage template publication within Tableau Server, generally via the APIs.

In Summary

There are a variety of ways to implement RLS in Tableau, in order to fit the architecture and security requirements of the variety of customers across industries, organization size, technical maturity, and implementations. The method(s) selected will depend on the confluence of a variety of these factors, but the general framework of how RLS works in Tableau remains consistent:

1. User is identified after secure authentication.

2. The set of data entitlements for the user is retrieved from all possible data entitlements.

3. The data is filtered by that set of data entitlements.

4. The filtered data is returned to the user.

In order to lock down this framework, it is advisable to always:

1. Create a Calculated Field in Tableau Desktop utilizing a user function.

2. Create a data source filter on the Calculated Field in Desktop.

3. Publish the data source to Tableau Server.

4. Connect workbooks to that Published Data Source.

It is always wise to think about how a given method will scale with the security needs of your organization, and for that reason, the entitlements table method of RLS is most commonly adopted by Tableau customers. However, depending on the specific needs of the organization, any of the methods described throughout this document is completely viable to ensure that the correct people are enabled with the correct data.

Appendix: Comparison matrix for RLS methods

This technical documentation covers a variety of ways RLS can be implemented within your Tableau environment, each having its own reason it may be preferable to an organization. The below matrix will help you decide which option may be best for you to help you focus your efforts and read the relevant section of this documentation.

	Method	Useful when...	Pros	Cons
RECOMMENDED	**Entitlements table method**	• There is an existing concept of entitlements in the database • The organization is setting up Row Level Security for the first time	• Easy to test, update, maintain, and scale • Works for both live connections and extracts in version 2018.3+	• Requires creating and maintaining entitlements table • Could require selecting and creating appropriate keys to optimize for performance
	CONTAINS () method in extracts	• You're implementing RLS in extracts prior to version 2018.3	• Allows you take advantage of extract efficiencies	• Requires mapping all users to a single column • Difficult to switch back to live connections because of string calculation
	Impersonation	• Every user accessing the data will be exist as a user in your SQL server (Usually, internal deployments)	• Security is handled and maintained in one place—the database	• Requires every person accessing the view to exist as a user within your SQL Server • Only works for Microsoft SQL Server
	Kerberos	• All necessary data sources are set up for Kerberos delegation and RLS is set up on the data source (usually internal deployments)	• The viewer's name appears on the access logs for the data source • Security is handled and maintained in the data source	• Tableau must be configured to use LDAP- Active Directory • Tableau Server must be joined to the AD domain • Every user must exist within your AD domain
	Initial SQL	• The database supports Initial SQL and RLS is set up on the data source side	• Allows the passing of Tableau parameters at load time • Dedicated connection that can't be shared with other users • Users must exist within data source to execute query as user	• Not all data sources support Initial SQL • Potential performance implications because of restricted cache sharing

Based on the information above, it is recommended that customers review the section most relevant to how they will be deploying Tableau Server (and any connected applications). If a preferred configuration is not currently known, the standard Row Level Security method is recommended.

About Tableau

Tableau is a complete, integrated, and enterprise-ready visual analytics platform that helps people and organizations become more data driven. Whether on-premises or in the cloud, on Windows or Linux, Tableau leverages your existing technology investments and scales with you as your data environment shifts and grows. Unleash the power of your most valuable assets: your data and your people.

Additional Resources

Tableau Server Scalability Technical Guide

Disaster Recovery for Tableau Server

Enterprise Analytics Powered by IT

Governed Self-Service Analytics at Scale