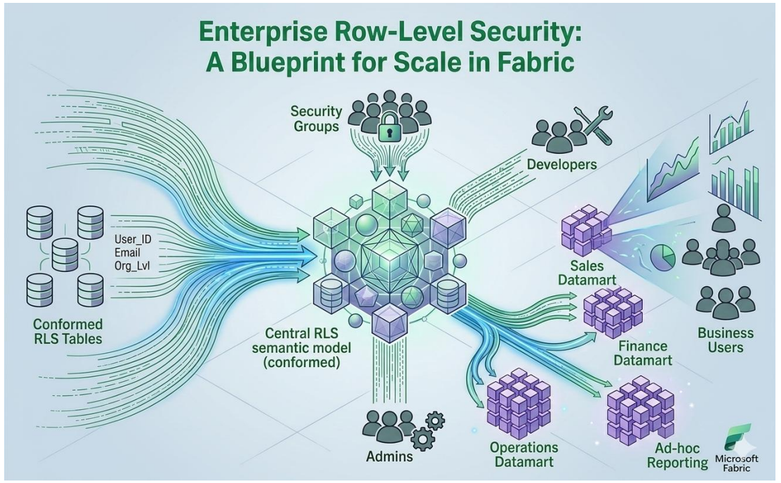
#FABCONSQLCON2026

FABCON
Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON
Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 - 20, 2026



Enterprise Row-Level Security: A Blueprint for Scale in Fabric

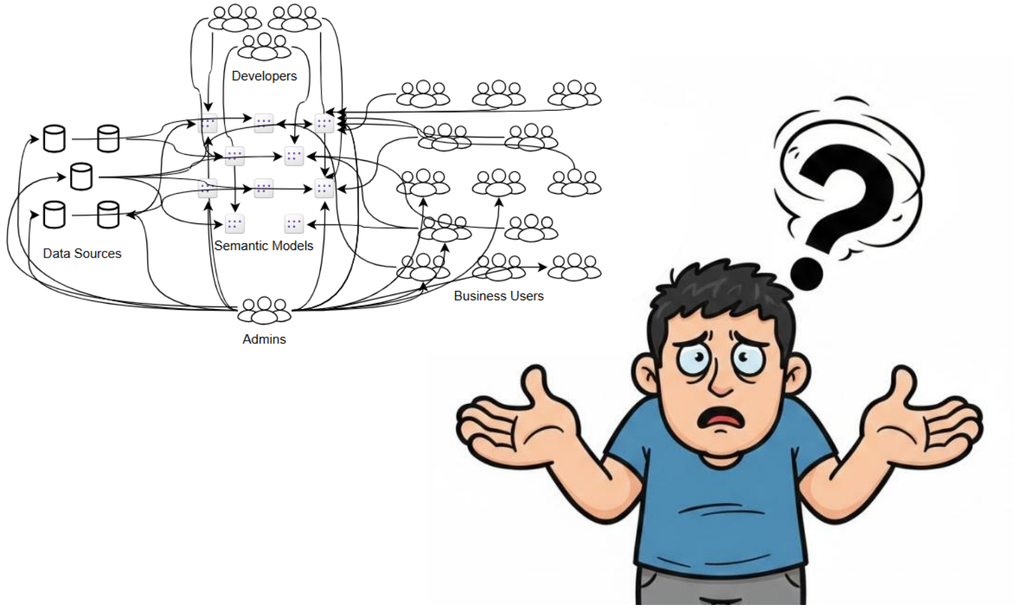
Karl Pover

Senior Solutions Architect

Axis Group

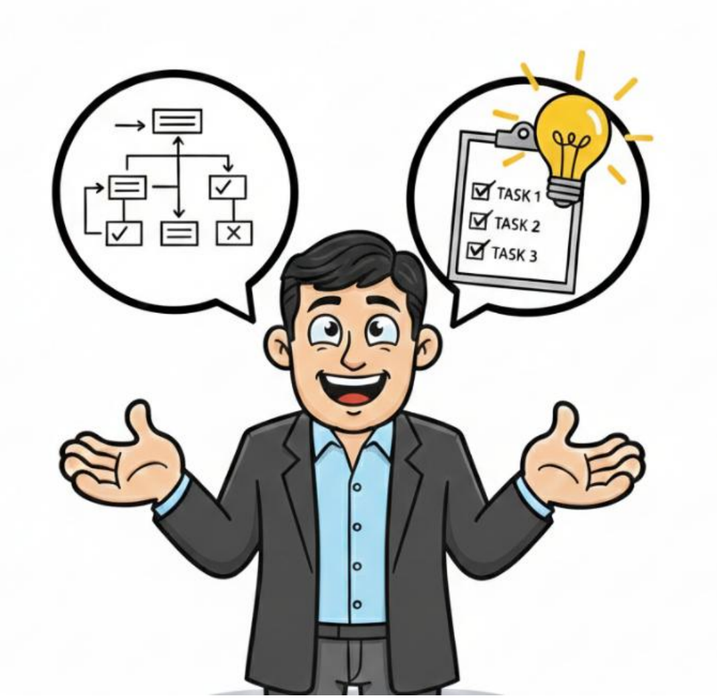
Power BI Row-Level Security (RLS): Current Situation

- Managing hundreds to thousands of users across multiple Semantic Models.
- Governing developers using inconsistent RLS data sources and custom DAX rules.
- Administering dozens of RLS Roles and Security Groups — or worse, individual user assignments.



What You'll Walk Away With

- A proven RLS architecture that scales to 100+ users.
- Practical guidance on sourcing user access data and managing Security Groups.
- Reusable design patterns for RLS Mapping tables and Semantic Model templates.
- A self-service data access lookup report and monitoring recommendations.
- A look ahead at AI-assisted development and testing.



RLS Overview


BUSINESS
USER
USER INTERFACE


BUSINESS USER
Who?


What data can they see?
BUSINESS USER
Who?

Security Groups

- C-suite
- GROUP VPs
- SALES REPS
- PRODUCT MANAGER
- REGIONAL TEAM

Security Groups

- C-suite
- GROUP VPs
- SALES REPS
- PRODUCT MANAGER
- REGIONAL TEAM

Semantic Model

RLS Roles

- GLOBAL
- SALES ORG LVL
- CLIENT LVL
- PRODUCT LVL
- SUPPLIER LVL

Security Groups

- C-suite
- GROUP VPs
- SALES REPS
- PRODUCT MANAGER
- REGIONAL TEAM

RLS Roles

- GLOBAL
- SALES ORG LVL
- CLIENT LVL
- PRODUCT LVL
- SUPPLIER LVL

Semantic Model

RLS DAX EXPRESSION

```
[UPN]  =  USERPRINCIPALNAME()
```

Security Groups

- C-suite
- GROUP VPs
- SALES REPS
- PRODUCT MANAGER
- REGIONAL TEAM

RLS Roles

- GLOBAL
- SALES ORG LVL
- CLIENT LVL
- PRODUCT LVL
- SUPPLIER LVL

Semantic Model

RLS DAX EXPRESSION

```
[UPN]  =  USERPRINCIPALNAME()
```

RLS-Filtered Tables

UPN	Dim Key
jane.doe	1
john.smith	2

Security Groups

- C-suite
- GROUP VPs
- SALES REPS
- PRODUCT MANAGER
- REGIONAL TEAM

RLS Roles

- GLOBAL
- SALES ORG LVL
- CLIENT LVL
- PRODUCT LVL
- SUPPLIER LVL

Semantic Model

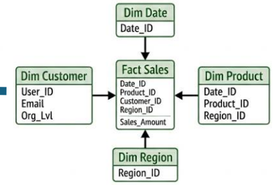
RLS DAX EXPRESSION

```
[UPN]  =  USERPRINCIPALNAME()
```

RLS-Filtered Tables

UPN	Dim Key
jane.doe	1
john.smith	2

Data Model



Security Groups

- C-suite
- GROUP VPs
- SALES REPS
- PRODUCT MANAGER
- REGIONAL TEAM

RLS Roles

- GLOBAL
- SALES ORG LVL
- CLIENT LVL
- PRODUCT LVL
- SUPPLIER LVL

Semantic Model

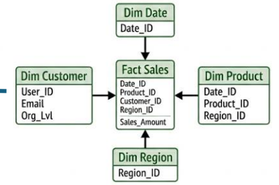
RLS DAX EXPRESSION

```
[UPN]  =  USERPRINCIPALNAME()
```

RLS-Filtered Tables

UPN	Dim Key
jane.doe	1
john.smith	2

Data Model



What Makes RLS “Enterprise”?

Enterprise RLS: Core Principles

REUSABLE

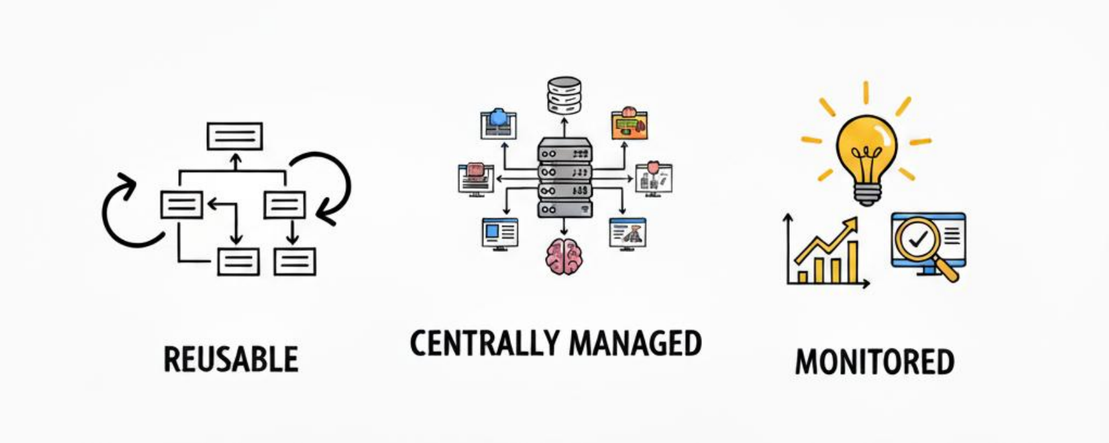
- Artifacts shared across multiple Semantic Models
- Templatized logic for consistent implementation

CENTRALLY MANAGED

- IT-managed process with business approval workflow

MONITORED

- Self-service permission visibility
- Group membership and access tracking



Reusable: What can be standardized?

Conformed Security Groups & RLS Roles

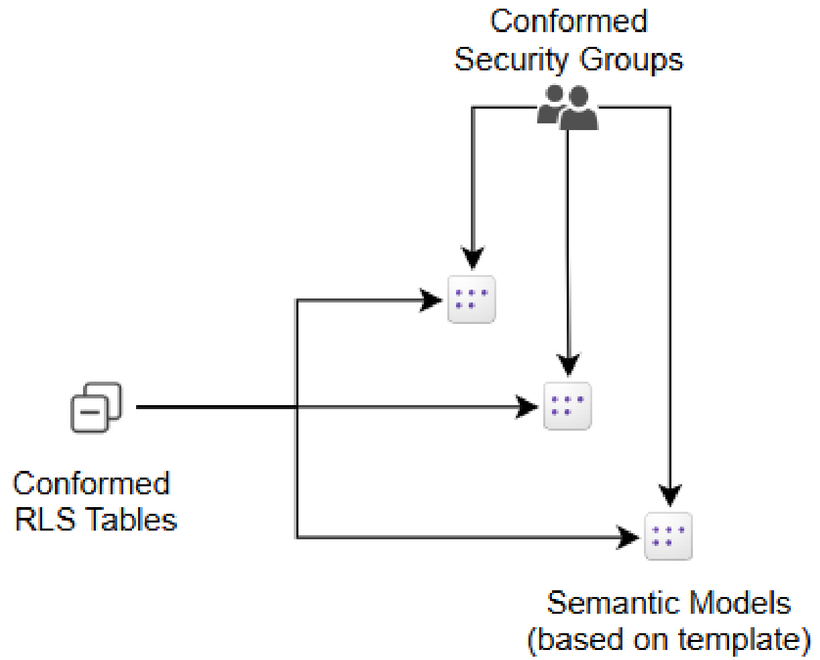
Defined once, applied consistently across all Semantic Models.

Conformed RLS Tables

Centralized upstream as mapping tables in a Fabric Lakehouse or Data Warehouse as a single source of truth.

Power Query and RLS Role Patterns

Common Power Query and DAX rules packaged in a Power BI template for consistent, accelerated development.



Centrally Managed: Who does what?

IT

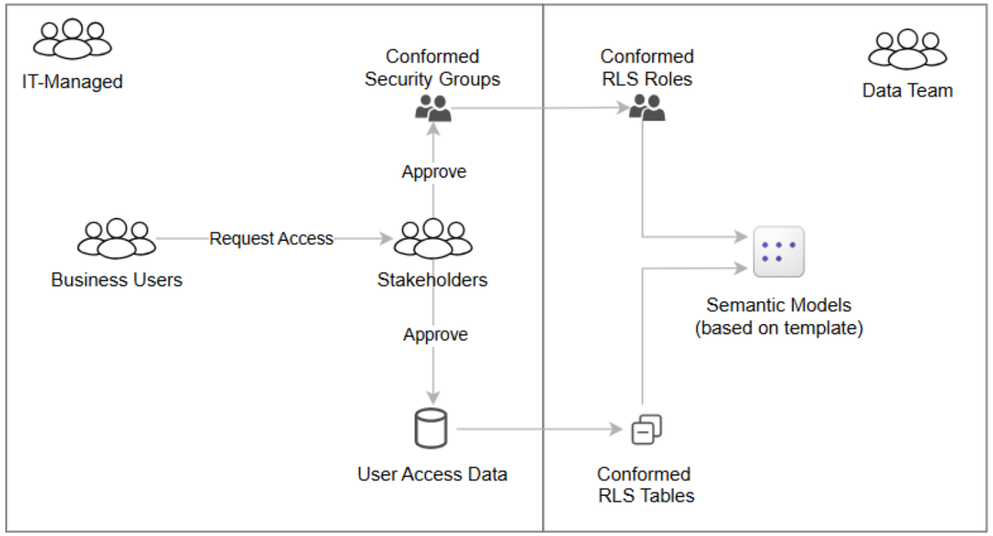
Provisions and maintains Security Groups, administers the formal approval workflow, and manages the User Access data source.*

Data Team

Develops RLS Mapping Views, defines RLS Roles, owns the Semantic Model template, and manages the User Access data source.*

Business

Requests and approves Security Group membership and user access changes.



Monitored: Access Transparency

Self-Service Access Review

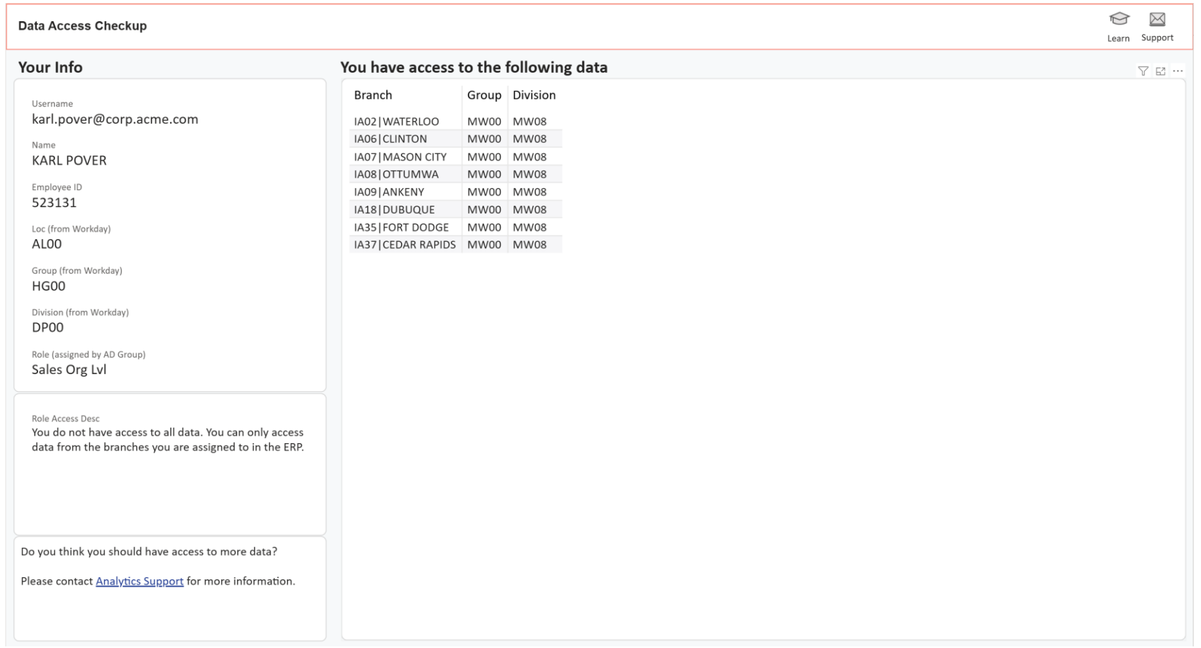
Users can view and verify their own access at any time.

Business Owner & Data Steward Review

Designated owners have full visibility into user access across their data domains.

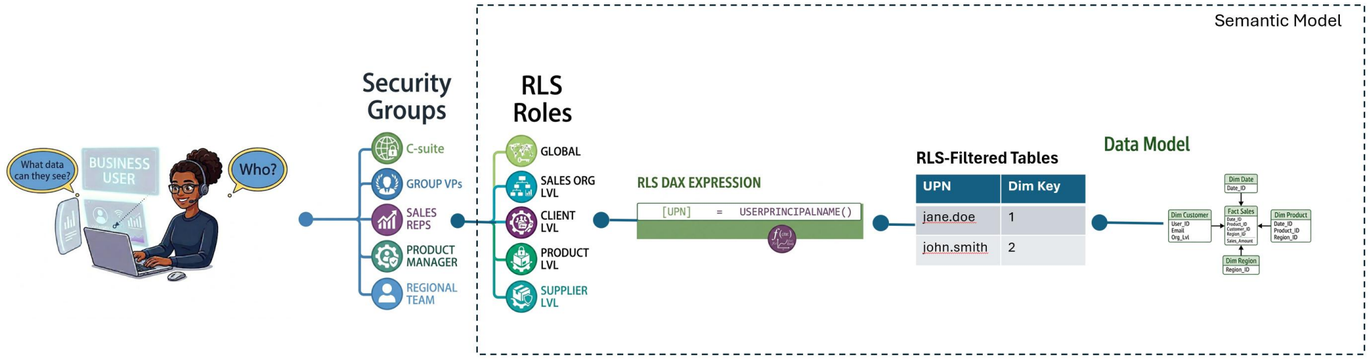
Admin Audit

Administrators periodically audit Security Group membership, flagging users assigned to more than one group.



Architecture Decision Points

Key Decision Points for RLS Architecture



Source User Access Data from Existing Systems

Re-use existing sources to define what data users have access to see.

Single Group Membership per User

One group per user keeps access logic predictable and auditable.

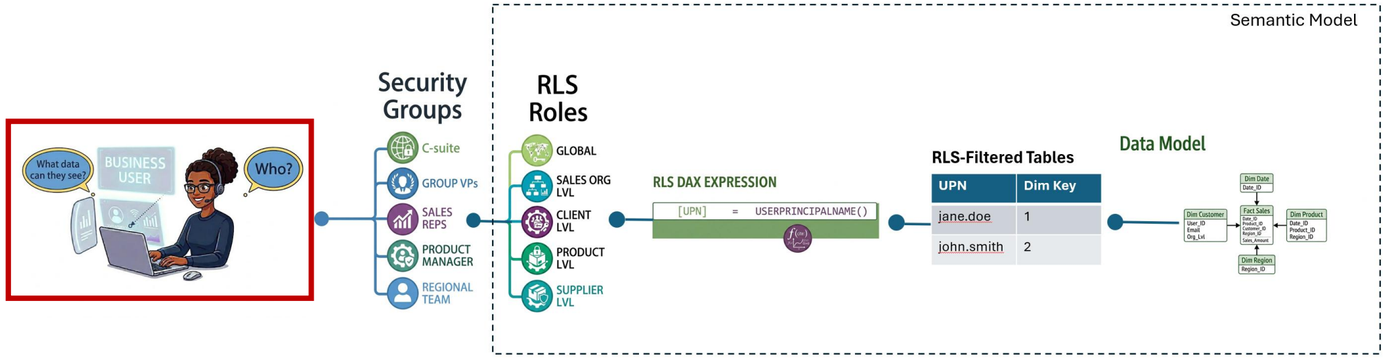
Dedicated Security Group per RLS Role

- One group per role to maintain clear boundaries and avoid permission overlap.
- Options to scope Security Groups to a Workspace or share across multiple, based on governance requirements.
- Global access options to use a dedicated group and RLS Role with no DAX filters or handle global access within existing roles with more complex DAX, choose based on maintainability.

Push RLS Filter Logic Upstream

Move filtering logic into RLS Mapping tables, reducing DAX to [Employee UPN] = USERPRINCIPALNAME().

Business User



Retrieving User Principal Names (UPNs)

UPNs ≠ Email Addresses

Treat them as distinct identifiers to avoid incorrect access mappings.

Use the Microsoft Graph API

Source UPNs directly to ensure accuracy and consistency with user ID.

Join to Employee Data

Correlate UPNs to employee records using email address.

Power Query to extract UPNs in a Dataflow Gen2

```
let
    // Function to get all pages using List.Generate (iterative approach)
    GetAllPagesIterative = (initialUrl as text) as list =>
        let
            // Generate pages using List.Generate
            pages = List.Generate(
                // Initial state
                () => [
                    url = initialUrl,
                    data = {},
                    hasMore = true
                ],

                // Condition to continue
                (state) => state[data] <> null,

                // Next iteration
                (state) =>
                    let
                        response = try Json.Document(Web.Contents(state[url]), 65001) otherwise null,
                        currentData = if response <> null then response[value] else null,
                        nextLink = if response <> null and Record.HasFields(response, "@odata.nextLink")
                                   then response[#"@odata.nextLink"]
                                   else null
                    in [
                        url = nextLink,
                        data = currentData,
                        hasMore = nextLink <> null
                    ],

                // Selector - what to return from each iteration
                (state) =>
                    if state[data] <> null
                    then state[data]
                    else {}
            ),

            // Combine all pages
            allData = List.Combine(pages)
        in
            allData,

    // Initial URL with your query parameters
    initialUrl = "https://graph.microsoft.com/v1.0/users?" &
                 Uri.BuildQueryString([
                     ConsistencyLevel = "eventual",
                     #"$filter" = "endswith(userPrincipalName,'corp.motion-ind.com')",
                     #"$count" = "true",
                     #"$top" = "999"
                 ]),

    // Get all pages using the iterative function
    allData = GetAllPagesIterative(initialUrl),

    // Convert the combined list to a table
    #"Converted to Table" = Table.FromRecords(allData)
in
    #"Converted to Table"
```

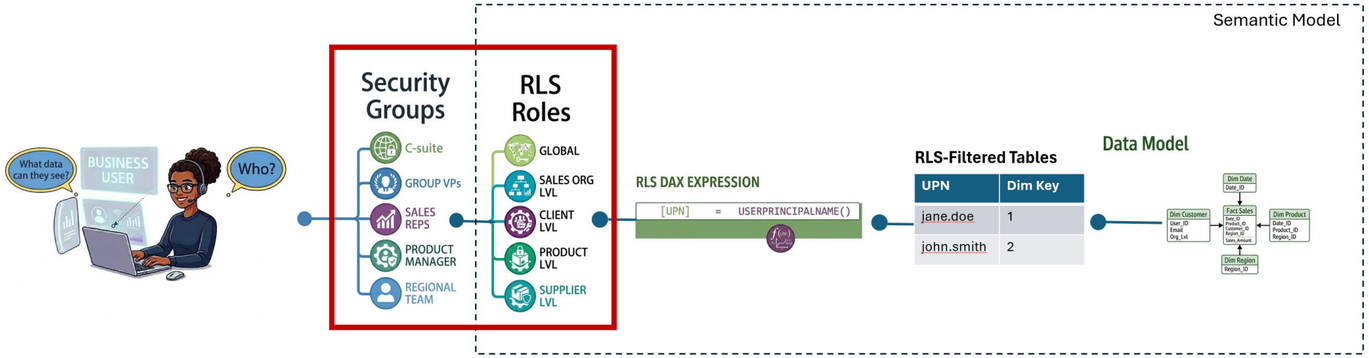
Where should to source security data?

Rating	Source	✅ Pros	❌ Cons
Best	**ERP / CRM Security Tables**	Permissions often already align; established user management process already in place	Source tables may be inaccessible or encrypted
Better	**Employee HR Attributes**	Data already exists; often already in Fabric	May be depended on by other processes; HR unlikely to take on additional responsibility; limited to certain dimensions
Better	**Employee HR Hierarchy**	Same as above; supports managerial visibility into direct reports	Doesn't account for non-managerial roles needing broader access
Better	**Employee Fields on Dimension Tables**	Data already exists; motivates data accuracy; business owns access	Limited to few dimension tables; typically only one employee field per record
Better	**Customer Contact Tables**	Same as above; supports M:M employee-to-customer relationships	Not available on all dimension tables
Good	**CRUD Interface in Fabric**	Fully independent; precisely tailored to reporting needs	Tool and data must be built from scratch
OK	**SharePoint List / Excel**	Independent; quick to set up and modify	Prone to becoming stale; structure easily broken; analytics team likely owns maintenance; data must be entered from scratch

Security data can come from different sources depending on role

Role	Definition	Source
Sales org-level	Filter by branch ID for branch and division managers to review sales and finance metrics.	ERP / CRM Security Tables
Employee-level	Filter by employee ID for managers to review their directs' performance	Employee HR Hierarchy
Customer-level	Filter by customer ID for sales reps to review their customers' sales. Based on Sales Rep field on D_Customer table.	Employee Fields on Dimension Tables
Customer-level	Filter by customer ID for sales support team based on if they are in the customer contact table.	Customer Contact Tables

Security Group Patterns



Why Create New Security Groups *Just for RLS*

✅ DO

- Always assign access through Security Groups — never assign individual users directly to roles.
- One dedicated Security Group per RLS Role.
- Establish and enforce consistent naming conventions across all Security Groups.
- Empower the business to own and manage Security Group membership.

❌ DON'T

- Repurpose or reuse existing Security Groups not originally designed for RLS.

RLS Role	Security Group
Global	PBI-VIEWER-RLS-GLOBAL
Sales Organization Level	PBI-VIEWER-RLS-SALE_ORG-LVL
Customer Level	PBI-VIEWER-RLS-CUST-LVL
Product Level	PBI-VIEWER-RLS-PROD-LVL

Security Group Naming Convention Pattern

Pattern: [Tool Identifier]-[Workspace Name (Optional)]-[Workspace Role]-[RLS Role]

Tool Identifier	Workspace (Optional)	Workspace Role	RLS Role
PBI	WS-SALES	VIEWER	RLS-GLOBAL
FABRIC	WS-FINANCE	APP_AUDIENCE	RLS-SALE_ORG-LVL
	WS-PURCHASING		RLS-CUST-LVL
			RLS-PROD-LVL

Key Points

- Self-documenting - The name alone communicates the tool, Workspace role, and RLS Role immediately.
- Simplifies auditing by making Workspace and RLS role assignments immediately visible.
- *Workspace name is optional — include only when the group is scoped to a specific Workspace.

  Example: PBI-SALES-VIEWER-RLS-GLOBAL vs. PBI-VIEWER-RLS-GLOBAL

RLS Security Groups Patterns

Workspace-Specific RLS Security Groups

Workspace	RLS Role	Security Group
Sales	Global	PBI-WS-SALES-VIEWER-RLS-GLOBAL
	Sales Org Level	PBI-WS-SALES-VIEWER-RLS-SALE_ORG-LVL
	Customer Level	PBI-WS-SALES-VIEWER-RLS-CUST-LVL
	Product Level	PBI-WS-SALES-VIEWER-RLS-PROD-LVL
Finance	Global	PBI-WS-FINANCE-VIEWER-RLS-GLOBAL
	Sales Org Level	PBI-WS-FINANCE-VIEWER-RLS-SALE_ORG-LVL
	Customer Level	PBI-WS-FINANCE-VIEWER-RLS-CUST-LVL

Scenario-Specific RLS Security Groups

RLS Role	Security Group
Global	PBI-VIEWER-RLS-GLOBAL
Sales Organization Level	PBI-VIEWER-RLS-SALE_ORG-LVL
Customer Level	PBI-VIEWER-RLS-CUST-LVL
Product Level	PBI-VIEWER-RLS-PROD-LVL

+ Workspace Role Security Groups

Workspace	Security Group
Sales	PBI-WS-SALES-VIEWER
Finance	PBI-WS-FINANCE-VIEWER

Scenario-Specific Security Groups Scale Better

Workspace-Specific vs. Scenario-Specific RLS Security Groups

	Scenario-Specific RLS Security Groups	Workspace-Specific RLS Security Groups
Access Control	Consistent access across all Workspaces where applied	Precise, limited to a specific Workspace
Flexibility	Less flexible when Workspaces have distinct requirements	**Easier to tailor to unique Workspace requirements**
Scalability	**Scales well; one group assignment applies everywhere**	Harder to scale as Workspaces grow
Group Management	Requires careful governance to prevent access creep	**Simplifies decommissioning — scope is clear**
Administrative Overhead	**Lower — one group assignment covers multiple Workspaces**	Higher — each Workspace needs its own set of groups
Auditability	**Easier to audit at scale; fewer groups to manage**	Contained scope makes auditing straightforward

Hybrid RLS Security Group Patterns

Workspace-Specific **Global** RLS Security Groups

Workspace	RLS Role	Security Group
Sales	Global	PBI-WS-SALES-VIEWER-RLS-GLOBAL
Finance	Global	PBI-WS-FINANCE-VIEWER-RLS-GLOBAL

Scenario-Specific **Non-Global** RLS Security Groups

RLS Role	Security Group
Sales Organization Level	PBI-VIEWER-RLS-SALES_ORG-LVL
Customer Level	PBI-VIEWER-RLS-CUST-LVL
Product Level	PBI-VIEWER-RLS-PROD-LVL

+ Workspace Role Security Groups

Workspace	Security Group
Sales	PBI-WS-SALES-VIEWER
Finance	PBI-WS-FINANCE-VIEWER

Global Access Security Group Patterns

Without Global Security Groups

RLS Role	Security Group
Sales Organization Level	PBI-VIEWER-RLS-SALE_ORG-LVL
Customer Level	PBI-VIEWER-RLS-CUST-LVL
Product Level	PBI-VIEWER-RLS-PROD-LVL

+ No Global RLS Role

+ More Complex DAX RLS Rule in other RLS Roles

```
[Employee UPN] = USERPRINCIPALNAME()
||
NOT ISEMPTY(
  FILTER(zSecurity,
    zSecurity[zUPN] = USERPRINCIPALNAME()
    && zSecurity[zHasGlobalAccess] = TRUE))
```

With Global Security Groups

RLS Role	Security Group
Global	PBI-VIEWER-RLS-GLOBAL
Sales Organization Level	PBI-VIEWER-RLS-SALE_ORG-LVL
Customer Level	PBI-VIEWER-RLS-CUST-LVL
Product Level	PBI-VIEWER-RLS-PROD-LVL

+ Global RLS Role without RLS Rules

+ Simpler DAX RLS Rule in other RLS Roles

```
[Employee UPN] = USERPRINCIPALNAME()
```

Global Security Groups Reduce DAX

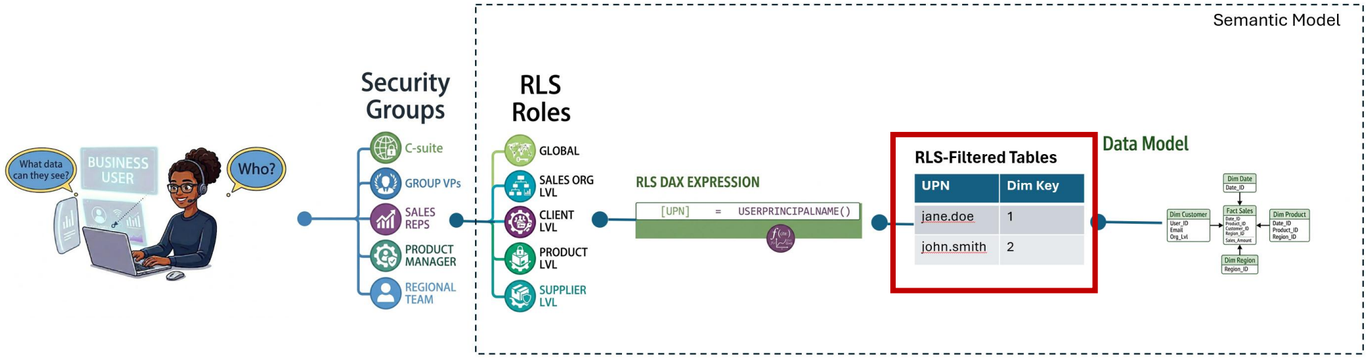
Global Access Handling

	Without Global Security Group	With Global Security Group
Group Management	**Fewer groups to create and manage**	One additional group required
User Maintenance	**No group membership change needed to grant or revoke global access**	Membership change required to grant or revoke global access
Access Management	**Global and non-global access managed in one unified location**	Global access isolated in a dedicated group
DAX Complexity	More complex DAX required to handle global access exceptions	**DAX reduced to a single, simple expression**
Query Performance	Additional compute required to resolve global access filters	**No additional compute — access resolved through group membership**

Formalize Business-Driven Security Group Membership

- Leverage tools such as SailPoint, ServiceNow, or MS Flow via the Graph API to automate Security Group Membership where:
  - Business owns and is accountable for all membership changes.
  - Formal approval workflows are required for all additions, modifications, and removals.
  - Regular audits are done to identify and resolve overlapping group assignments.
  - All changes tracked over time to maintain a clear audit trail.

RLS Mapping Table Design



What is a RLS Mapping Table?

- Connects users (UPN) to dimensional data, defining who can see what.
- Exists upstream as Views in a Fabric Lakehouse or Data Warehouse, serving as a single source of truth independent of any Semantic Model.
- Reused across multiple Semantic Models, eliminating redundant access logic.
- Leverages model relationships to handle filtering, keeping DAX rules simple and minimal.

UPN	Branch Key
john.doe@corp.acme.com	92928
john.doe@corp.acme.com	39282
john.doe@corp.acme.com	93832
jane.smith@corp.acme.com	92928

RLS Mapping Table Build Pattern

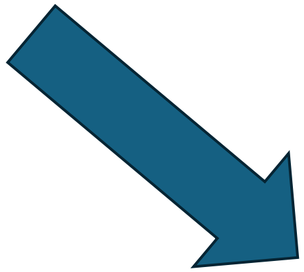
Source Sales Org Security Table

Employee ID	Security Level	Division Key	Branch Key
000001	Division	003829	
000002	Branch	003829	92928

RLS Mapping Table Build Pattern – Expand Hierarchies

Source Sales Org Security Table

Employee ID	Security Level	Division Key	Branch Key
000001	Division	003829	
000002	Branch	003829	92928



Expanded Sales Org Security Table

Employee ID	UPN	Security Level	Division Key	Branch Key
000001	john.doe@corp.acme.com	Division	003829	92928
000001	john.doe@corp.acme.com	Division	003829	39282
000001	john.doe@corp.acme.com	Division	003829	93832
000002	jane.smith@corp.acme.com	Branch	003829	92928

RLS Mapping Table Build Pattern – Trim to Mapping Table

Source Sales Org Security Table

Employee ID	Security Level	Division Key	Branch Key
000001	Division	003829	
000002	Branch	003829	92928

Expanded Sales Org Security Table

Employee ID	UPN	Security Level	Division Key	Branch Key
000001	john.doe@corp.acme.com	Division	003829	92928
000001	john.doe@corp.acme.com	Division	003829	39282
000001	john.doe@corp.acme.com	Division	003829	93832
000002	jane.smith@corp.acme.com	Branch	003829	92928

RLS_Sales_Org_Lvl

UPN	Branch Key
john.doe@corp.acme.com	92928
john.doe@corp.acme.com	39282
john.doe@corp.acme.com	93832
jane.smith@corp.acme.com	92928

RLS Mapping Table Build Pattern Key Points

- Create in the Fabric Lakehouse or Data Warehouse, changes propagate automatically to all dependent Semantic Models.
- Explode hierarchies to the lowest grain, producing one flat table per dimension.
- Map business identifiers to Semantic Model surrogate keys and employee identifiers to UPNs.
- Reduce each table to only two fields: **UPN** and the **dimensional key**.
- *Warning:* Avoid transactional-level RLS tables - use one RLS Mapping table per dimension when multiple dimensions require filtering.

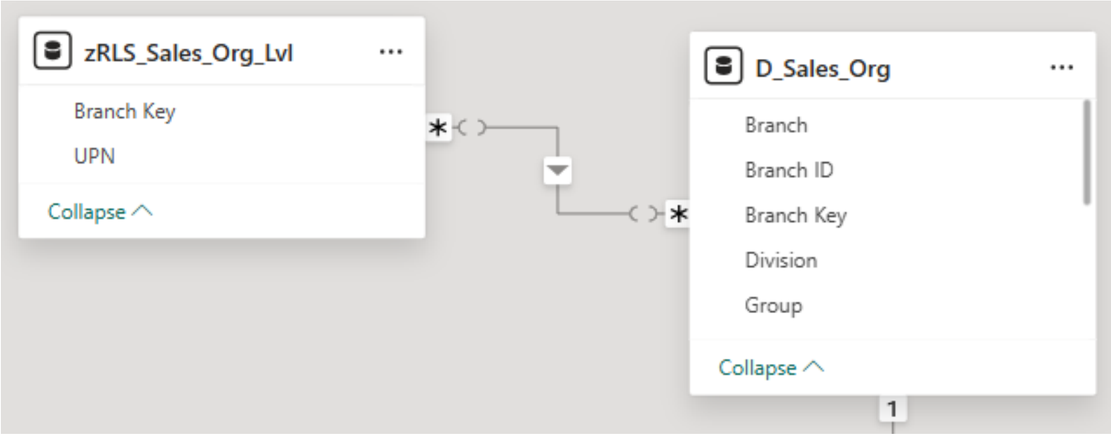
Minimal Power Query to Load RLS Mapping Table

- Apply an inner join to fact table to pre-filter the RLS Mapping table to only relevant rows, reducing Semantic Model size.
- Ensure Power Query is foldable.
- If using custom SQL, use *Value.NativeQuery()* with folding explicitly enabled.

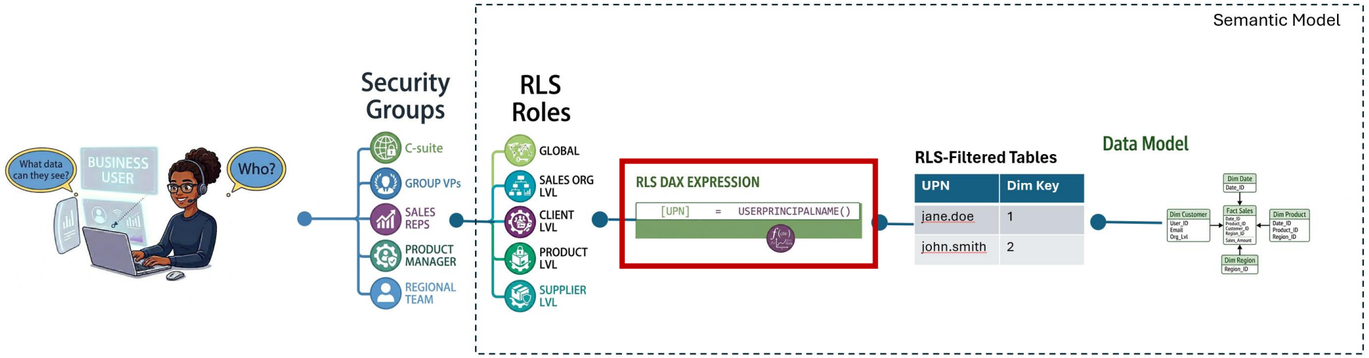
```
let
        Source = Sql.Database(Param_Lakehouse,"Enterprise_LH"), dbo_RLS_Sales_Org_Lvl = Value.NativeQuery(Source,
                "SELECT RLS.*
                FROM RLS_Sale_Org_Lvl RLS
                INNER JOIN (
                        SELECT DISTINCT Branch_Key
                        FROM dbo.Fact_Sale) FACT
                ON FACT.Branch_Key = RLS.Branch_Key",
                null, [EnableFolding = true])
in
        dbo_RLS_Sales_Org_Lvl
```

Create Single Direction Join from RLS Table to Dimension

- Link each RLS Mapping table to its corresponding dimension table - UI filters will only surface values the user is authorized to see.
- Define relationships as single-direction, many-to-many (M:M) away from RLS table.
- Prefix all RLS table names with z (e.g., *zRLS_Sales_Org_Lvl*) to designate them as system tables and push them to the bottom of the table list.



DAX RLS Rules

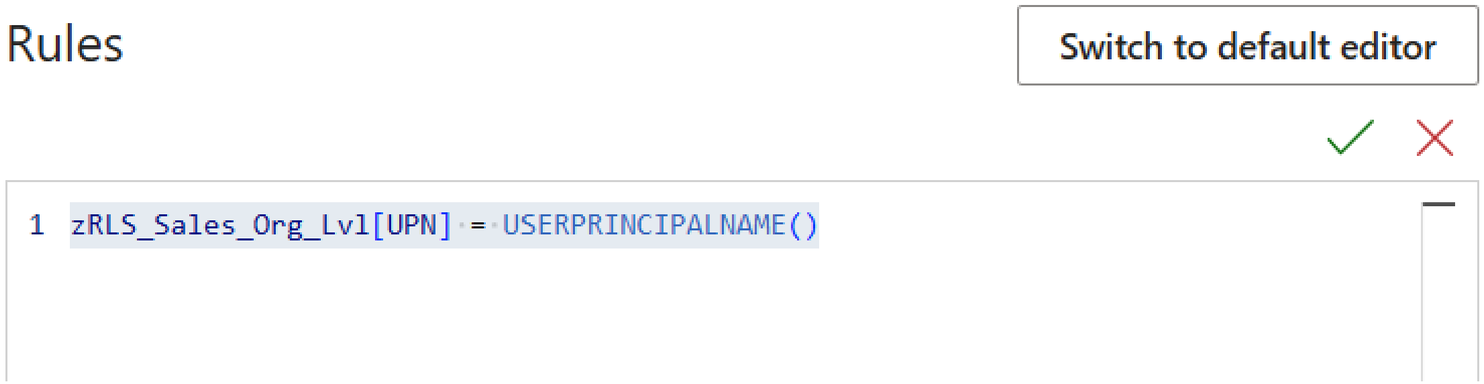


Keep DAX RLS Rules Simple

Core Principle: Push security logic upstream, DAX should be the last line of defense, not the first.

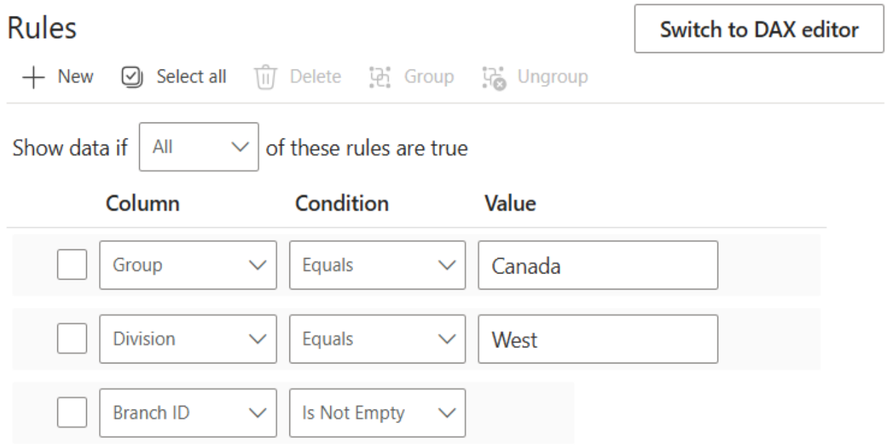
Dynamic RLS Better Practice

- Reduce DAX to a single, simple expression: *[UPN] = USERPRINCIPALNAME()*
- Let table relationships handle filtering, keeping DAX rules minimal and maintainable.

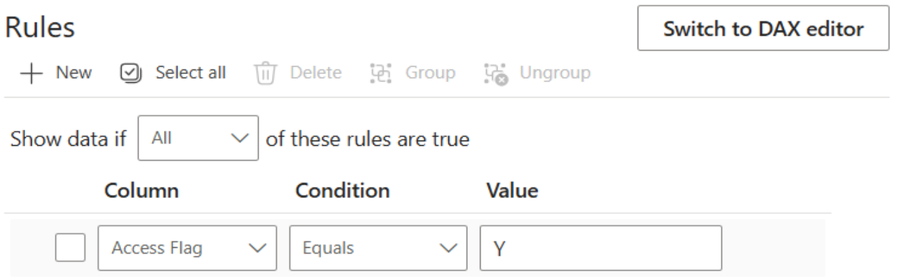


Simplify Static RLS Rules

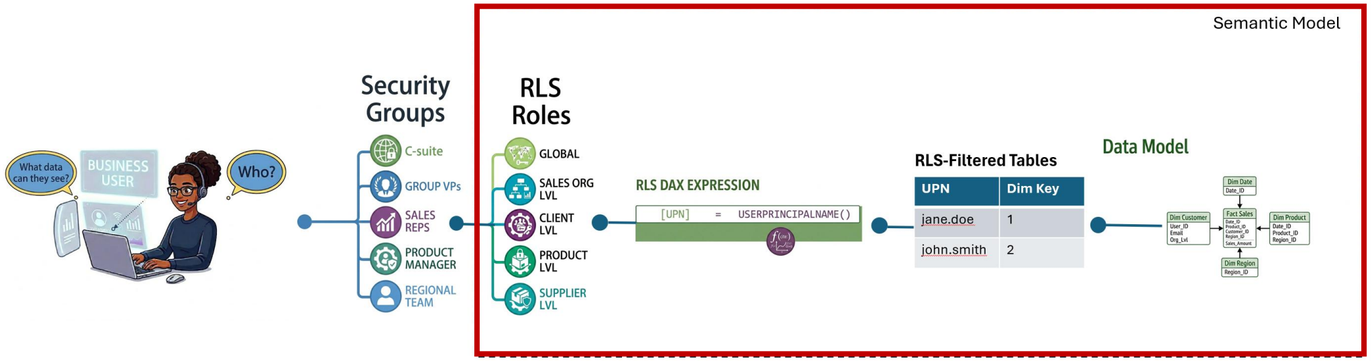
❌ **Don't Do This** — embedding business logic directly in DAX creates brittle, hard-to-maintain rules.



✅ **Do This Instead** — push any conditional logic upstream and resolve it in a single [Access Flag] field in the source table, reducing DAX to a simple expression.



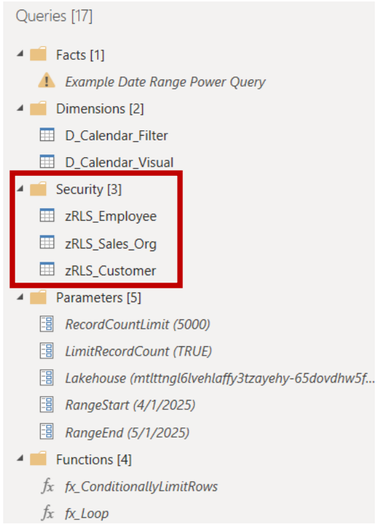
Governance & Templates



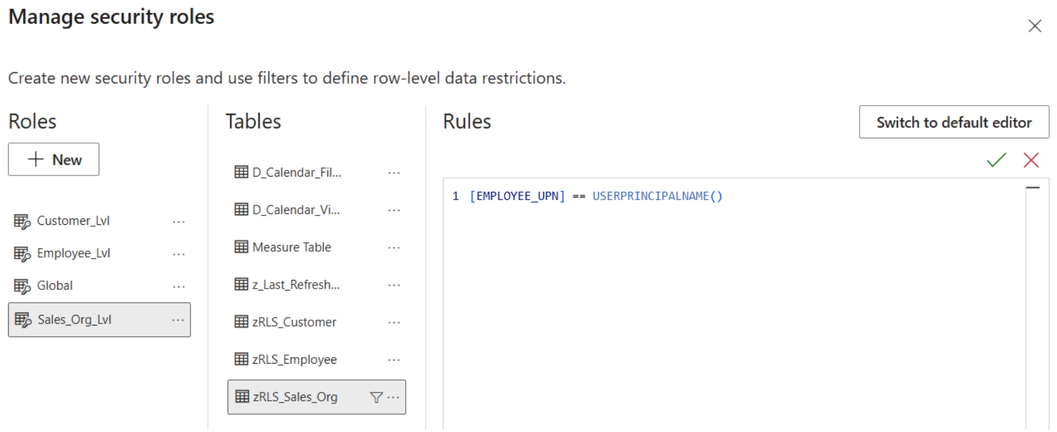
Standardize Your Implementation

Pre-build components in an organizational Power BI template (.pbit)

Power Query to load RLS tables



RLS Roles with DAX expressions



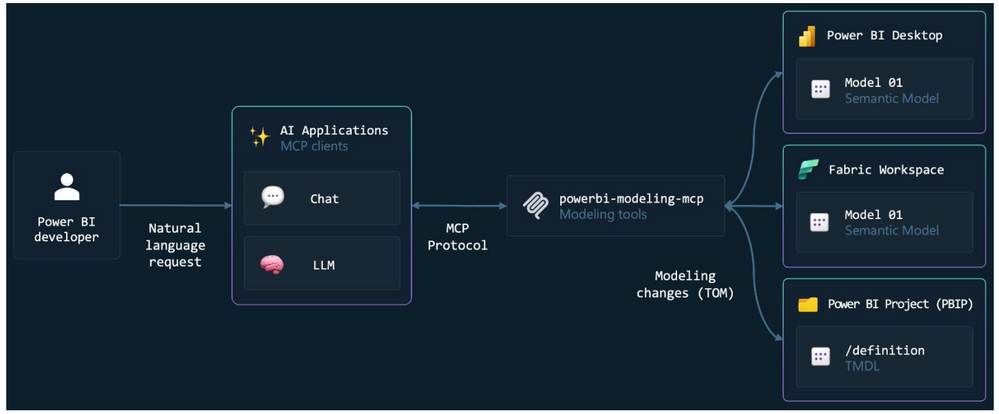
Add RLS to Development Checklist

Concept	Developer	Reviewer	Comments
RLS			
Semantic Model uses conformed RLS tables. Inner join is used to maintain relational integrity between the conformed RLS table and the data model.	✅	✅	Note any exceptions and reasons.
Semantic Model uses conformed RLS roles.	✅	✅	
RLS table's relationship in data model filters Semantic Model as expected.	✅	✅	
DAX RLS rules are as simple as possible. Logic is pushed as far upstream as possible.	✅	✅	
Impersonated user testing is completed for each role.	✅	✅	

Use GitHub Copilot to Develop and QA RLS

Power BI Modeling MCP Server

- Developers and AI agents can interact with Power BI models for development, testing,and documentation generation.
- Use in combination with GitHub Copilot in Visual Studio Code to make managing RLS between multiple Semantic Models easier.
- Find out more at https://github.com/microsoft/powerbi-modeling-mcp/blob/main/README.md.



GitHub Copilot Example

Example prompt over an existing Semantic Model with RLS best practices configured.

Create a detailed artifact that captures the following information about Row Level Security (RLS) in the Semantic Model:

- *RLS Roles: List all roles defined in the model.*
- *DAX Rules: For each role, include the DAX RLS rule(s) defined.*
- *Filtered Tables: List the tables filtered by each role.*
- *Power Query Definitions: For each filtered table, include the Power Query definition or M expression used to populate the table.*
- *Data Sources: Specify the data source for each filtered table.*
- *Relationships: For each filtered table, list its relationships with other tables, including:*
  - *The related table(s).*
  - *The columns used in the relationship.*
  - *The relationship type (e.g., one-to-many, many-to-one).*

Format the artifact in markdown to allow easy comparison with another Semantic Model.

Example result we use as context to test that other Semantic Models have the same RLS patterns.

```
# Row Level Security (RLS) Artifact

## RLS Roles
- **Employee Lvl**: Filters `zRLS_Employee`
- **Global**: No specific table restrictions
- **Sales Org Lvl**: Filters `zRLS_Sales_Org`
- **Customer Lvl**: Filters `zRLS_Customer`

## DAX Rules
- **Employee Lvl**: `zRLS_Employee[UPN] = USERPRINCIPALNAME()`
- **Global**: No DAX rule defined
- **Sales Org Lvl**: `zRLS_Sales_Org`: `zRLS_Sales_Org[UPN] = USERPRINCIPALNAME()`
- **Customer Lvl**: `zRLS_Customer[UPN] = USERPRINCIPALNAME()`

## Filtered Tables

### zRLS_Sales_Org
**Power Query Definition:**
```m
let
    Source = Sql.Database(Lakehouse&".datawarehouse.fabric.microsoft.com", "Enterpri
    dbo_RLS_Location = Value.NativeQuery(Source,
```
```

Monitoring

Monitoring Focus: Security Group Membership

Single Group Membership per User

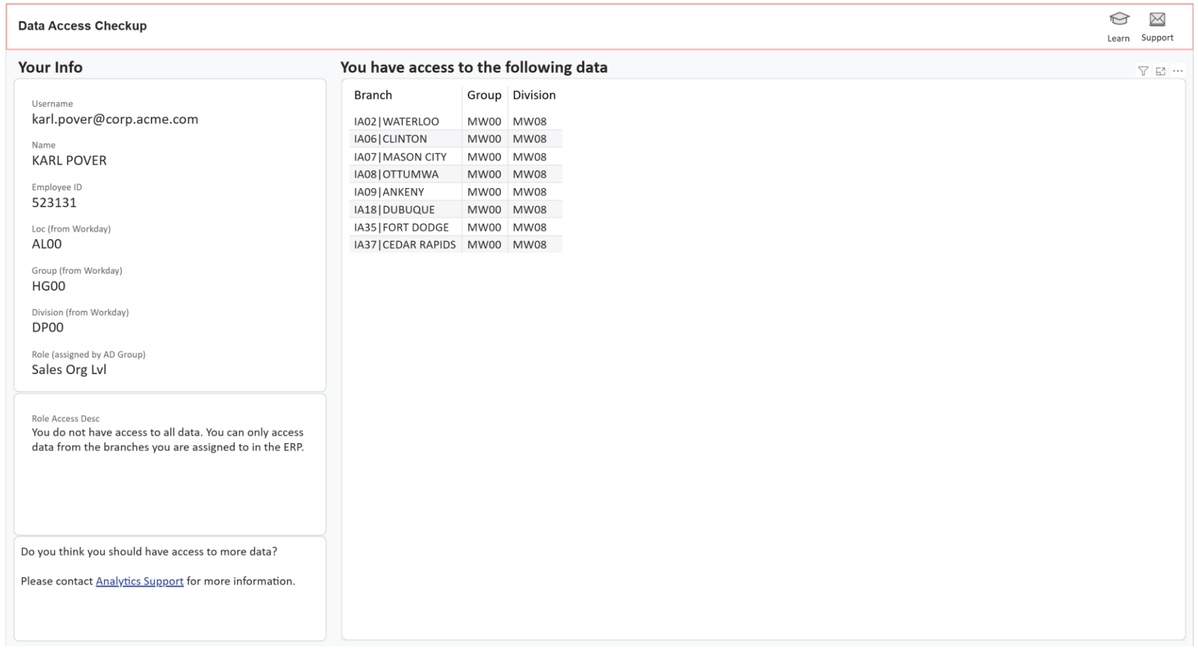
Monitor for overlapping assignments, as users in more than one RLS Security Group can cause increased resource usage and unexpected filtering behavior.

Security Group & User Access Data Mismatch

Flag members with no corresponding User Access record, and users with User Access records but no Security Group membership — both result in blocked access.

Self-Service Monitoring

- Provides each user with a personal access summary and answer the question, *"What data can I see?"*
- Empowers users to validate and troubleshoot their own access, reducing support tickets and dependency on IT.
- Doubles as a developer testing tool, with user impersonation to verify RLS rules are behaving as expected.



Key Takeaways

The Enterprise RLS Blueprint

- Find a formal source for user access data
- Create RLS-dedicated Security Groups per RLS Role
- Assign users to a single Security Group
- Formalize process to grant users access
- Build reusable RLS mapping tables upstream in Fabric Lakehouse or Data Warehouse
- Keep DAX simple: [UPN] = USERPRINCIPALNAME()
- Template and govern your approach with Copilot
- Provide self-service RLS access monitoring for business
- Audit Security Groups to maintain optimal performance

What Success Looks Like

Ad-hoc RLS	Enterprise RLS
Users request RLS access to each Semantic Model.	Users make a single request for RLS access to every Semantic Model.
BI developers change user access data in Excel files.	Security is based on an IT-managed business-driven process.
BI developers must go through several Semantic Models to update RLS DAX Rules.	Updates are done upstream in a single, shared RLS mapping table in a Fabric Lakehouse.
Users ask BI developers to confirm their access.	Users use a Power BI report to look up their own access.
Power BI Admins need to spend time to know which Security Groups line up to which RLS roles.	Admins easily assign and audit Security Groups in RLS roles based on naming convention.
BI developers spend time developing new RLS for each Semantic Model.	BI developers re-use RLS elements in templates.
Interactive CU usage spikes due to users in multiple roles.	Admins proactively remove users in multiple Security Groups.

Questions?

Let's discuss

- Your specific RLS challenges
- Implementation questions

Contact:

- Email: kpover@axisgroup.com
- LinkedIn: https://www.linkedin.com/in/karlpover/

Thank you!

Sound off.
The mic is all yours.
Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

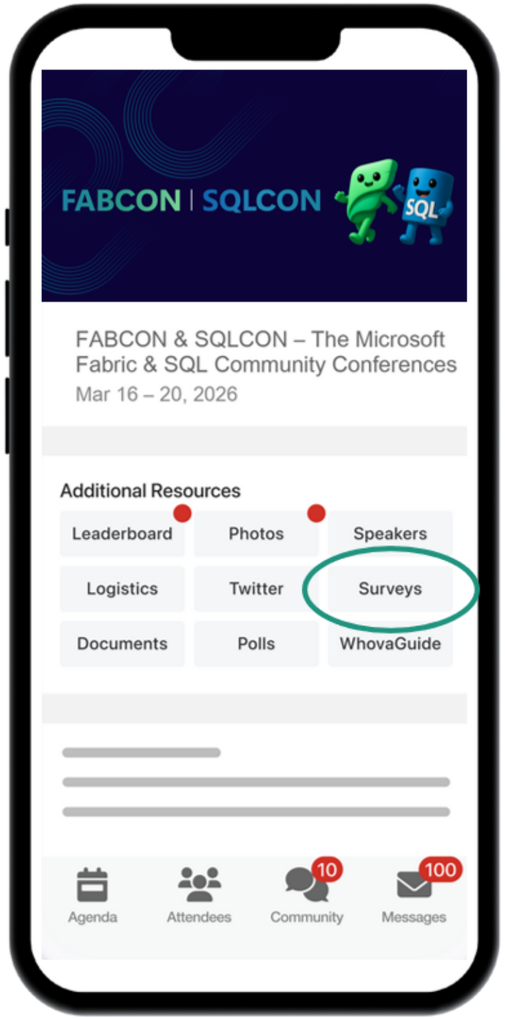
Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel

How was the session?

Complete Session Surveys in Whova for your chance to WIN PRIZES!



Get Two Fabric Certifications for FREE

Attendees of FABCON can take the Fabric Analytics Engineer or Fabric Data Engineer exam for free. Be part of the 2 fastest growing role-based certifications in Microsoft history.

Request your voucher by March 23, 2026.

https://aka.ms/fabcon/cert100