Security in data systems

8. oktober 2025 10:04

Security is a fundamental part of the data system. Joe Reis and Matt Housley place it as an undercurrent in their data engineering cycle and that makes sense. With every solution that is built security has to be considered, both internally and externally. The internal security is about making sure the people in the organization that are going to use the data only is able to see that which is relevant and allowed for them to see. It's like making sure certain doors or equipment is locked of. The guiding light is the principle of *least privilege* and that really means users and systems should be granted only the minimum access rights necessary to perform their designated tasks.

Many modern data tools have some built-in security permissions management. For example in PowerBI you can grant different access rights on workspaces, semantic models and reports.

In databricks you can manage users in the user management setup and permissions can be controlled on different data assets in the unity catalog. The same goes for surrounding tools like DevOps, Tabular Editor, SQL databases and so on. It's quite an enourmous task to manage security and to fully practice the principle of least privilege. To help us to it more efficiently, we can use security groups such as AD groups. This makes it easier because then adding or removing a person from that group automatically makes that person have the rights and permissions of that group.

Row-level security on datasets - a practical example

Row-level security refers to ways of adjusting the rows from a dataset which are visible to the user. For example it could be that a user should only see data from her/his region, store, country and so on.

In Aarsleff we also use row-level security and it's based on the organizational placement of the person. This setup is shown below.

1. Philosophy

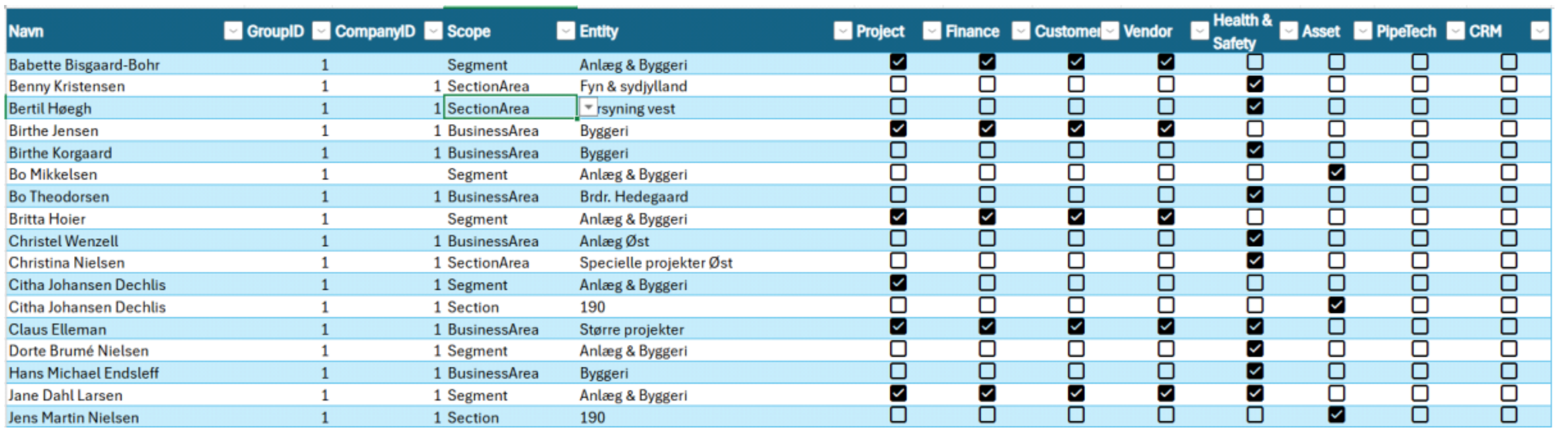
Security is a fundamental part of the data lifecycle at **Aarsleff**. Every data solution we build must consider access control as a first-class concern. To achieve this, we maintain a structured framework that ensures row-level security (RLS) is consistently applied across environments and data solutions.

2. Source of Security Configuration

The security configuration originates from a **local spreadsheet**. This file defines the **scope of access** for different solutions and users.

- Each row specifies which **access type** (e.g., Asset, Finance) and which **scope levels** (Company, Department, etc.) are available to a given user.
- If a user does not have access at a particular scope level, the system defaults the value to **"No Access."**

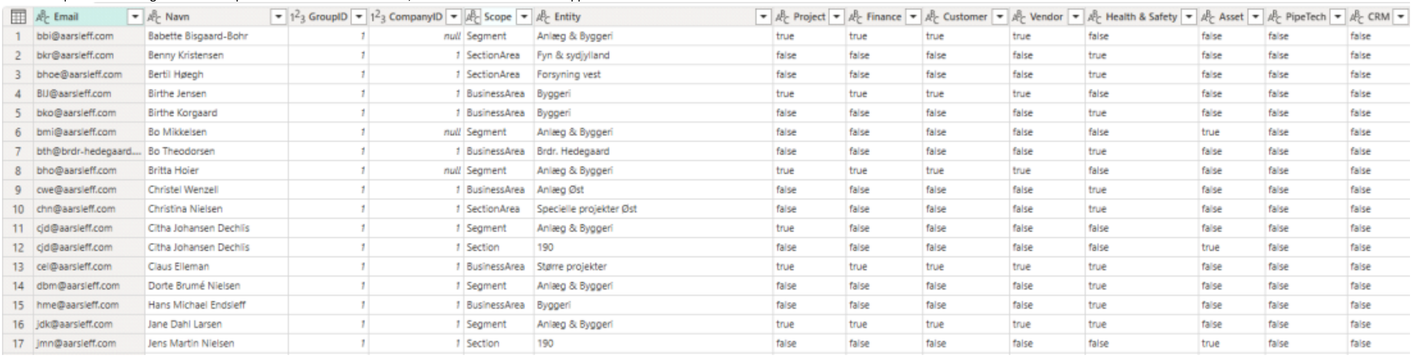
This spreadsheet effectively acts as a **proxy list of entitlements**, centralizing control of row-level permissions.



Navn	GroupID	CompanyID	Scope	Entity	Project	Finance	Custome	Vendor	Health & Safety	Asset	PipeTech	CRM
Babette Bisgaard-Bohr	1		Segment	Anlæg & Byggeri	☑	☑	☑	☑	☐	☐	☐	☐
Benny Kristensen	1	1	SectionArea	Fyn & sydjylland	☐	☐	☐	☐	☑	☐	☐	☐
Bertil Høegh	1	1	SectionArea	rsyning vest	☐	☐	☐	☐	☑	☐	☐	☐
Birthe Jensen	1	1	BusinessArea	Byggeri	☑	☑	☑	☑	☐	☐	☐	☐
Birthe Korgaard	1	1	BusinessArea	Byggeri	☐	☐	☐	☐	☑	☐	☐	☐
Bo Mikkelsen	1		Segment	Anlæg & Byggeri	☐	☐	☐	☐	☐	☑	☐	☐
Bo Theodorsen	1	1	BusinessArea	Brdr. Hedegaard	☐	☐	☐	☐	☑	☐	☐	☐
Britta Hoier	1		Segment	Anlæg & Byggeri	☑	☑	☑	☑	☐	☐	☐	☐
Christel Wenzell	1	1	BusinessArea	Anlæg Øst	☐	☐	☐	☐	☑	☐	☐	☐
Christina Nielsen	1	1	SectionArea	Specielle projekter Øst	☐	☐	☐	☐	☑	☐	☐	☐
Citha Johansen Dechlis	1	1	Segment	Anlæg & Byggeri	☑	☐	☐	☐	☐	☐	☐	☐
Citha Johansen Dechlis	1	1	Section	190	☐	☐	☐	☐	☐	☑	☐	☐
Claus Elleman	1	1	BusinessArea	Større projekter	☑	☑	☑	☑	☑	☐	☐	☐
Dorte Brumé Nielsen	1	1	Segment	Anlæg & Byggeri	☐	☐	☐	☐	☑	☐	☐	☐
Hans Michael Endsleff	1	1	BusinessArea	Byggeri	☐	☐	☐	☐	☑	☐	☐	☐
Jane Dahl Larsen	1	1	Segment	Anlæg & Byggeri	☑	☑	☑	☑	☑	☐	☐	☐
Jens Martin Nielsen	1	1	Section	190	☐	☐	☐	☐	☐	☑	☐	☐

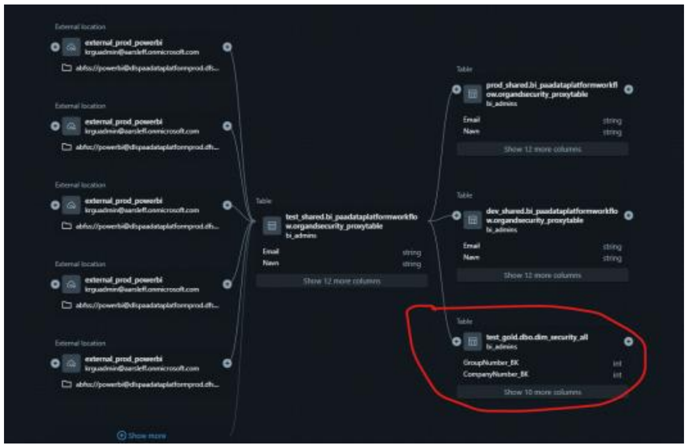
3. Data Ingestion and Lineage

1. The spreadsheet is ingested into the platform via **Power BI Dataflow**, where transformations are applied.



	Email	Navn	GroupID	CompanyID	Scope	Entity	Project	Finance	Customer	Vendor	Health & Safety	Asset	PipeTech	CRM
1	bbi@aarsleff.com	Babette Bisgaard-Bohr	1	null	Segment	Anlæg & Byggeri	true	true	true	true	false	false	false	false
2	bkr@aarsleff.com	Benny Kristensen	1	1	SectionArea	Fyn & sydjylland	false	false	false	false	true	false	false	false
3	bhoe@aarsleff.com	Bertil Høegh	1	1	SectionArea	Forsyning vest	false	false	false	false	true	false	false	false
4	BIJ@aarsleff.com	Birthe Jensen	1	1	BusinessArea	Byggeri	true	true	true	true	false	false	false	false
5	bko@aarsleff.com	Birthe Korgaard	1	1	BusinessArea	Byggeri	false	false	false	false	true	false	false	false
6	bmi@aarsleff.com	Bo Mikkelsen	1	null	Segment	Anlæg & Byggeri	false	false	false	false	false	true	false	false
7	bth@brdr-hedegaard....	Bo Theodorsen	1	1	BusinessArea	Brdr. Hedegaard	false	false	false	false	true	false	false	false
8	bho@aarsleff.com	Britta Hoier	1	null	Segment	Anlæg & Byggeri	true	true	true	true	false	false	false	false
9	cwe@aarsleff.com	Christel Wenzell	1	1	BusinessArea	Anlæg Øst	false	false	false	false	true	false	false	false
10	chn@aarsleff.com	Christina Nielsen	1	1	SectionArea	Specielle projekter Øst	false	false	false	false	true	false	false	false
11	cjd@aarsleff.com	Citha Johansen Dechlis	1	1	Segment	Anlæg & Byggeri	true	false	false	false	false	false	false	false
12	cjd@aarsleff.com	Citha Johansen Dechlis	1	1	Section	190	false	false	false	false	false	true	false	false
13	cel@aarsleff.com	Claus Elleman	1	1	BusinessArea	Større projekter	true	true	true	true	true	false	false	false
14	dbm@aarsleff.com	Dorte Brumé Nielsen	1	1	Segment	Anlæg & Byggeri	false	false	false	false	true	false	false	false
15	hme@aarsleff.com	Hans Michael Endsleff	1	1	BusinessArea	Byggeri	false	false	false	false	true	false	false	false
16	jdk@aarsleff.com	Jane Dahl Larsen	1	1	Segment	Anlæg & Byggeri	true	true	true	true	true	false	false	false
17	jmn@aarsleff.com	Jens Martin Nielsen	1	1	Section	190	false	false	false	false	false	true	false	false

1. The transformed data is stored in the **data lake (Power BI container, prod environment).**
2. From here:
   - The file is replicated across environments (dev/test/prod).
   - A **Delta table** is created for each environment.
   - The central security dataset is materialized as the **dim_security_all** table.


External location
external_prod_powerbi
krguadmin@aarsleff.onmicrosoft.com
abfss://powerbi@dlspaadataplatformprod.dfs...
External location
external_prod_powerbi
krguadmin@aarsleff.onmicrosoft.com
abfss://powerbi@dlspaadataplatformprod.dfs...
External location
external_prod_powerbi
krguadmin@aarsleff.onmicrosoft.com
abfss://powerbi@dlspaadataplatformprod.dfs...
External location
external_prod_powerbi
krguadmin@aarsleff.onmicrosoft.com
abfss://powerbi@dlspaadataplatformprod.dfs...
External location
external_prod_powerbi
krguadmin@aarsleff.onmicrosoft.com
abfss://powerbi@dlspaadataplatformprod.dfs...
Show more
Table
test_shared.bi_paadataplatformworkflow.organdsecurity_proxytable
bi_admins
Email string
Navn string
Show 12 more columns
Table
prod_shared.bi_paadataplatformworkflow.organdsecurity_proxytable
bi_admins
Email string
Navn string
Show 12 more columns
Table
dev_shared.bi_paadataplatformworkflow.organdsecurity_proxytable
bi_admins
Email string
Navn string
Show 12 more columns
Table
test_gold.dbo.dim_security_all
bi_admins
GroupNumber_BK int
CompanyNumber_BK int
Show 10 more columns

4. Structure of dim_security_all

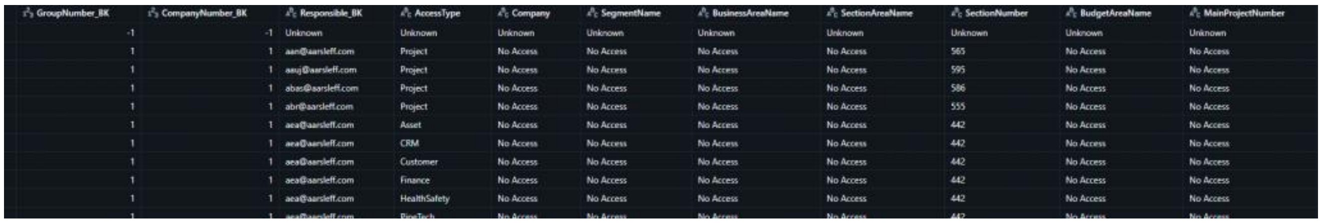
The dim_security_all table contains:

- **Access types** (e.g., Asset, Finance).
- **Scope levels** (columns from Company onward).
- **Access values** per user (actual values or "No Access").

For each data solution, dim_security_all is **cloned and trimmed** down to the relevant access types.

- Example: A user with access to both Asset and Finance will have two rows: one for Asset, one for Finance.

This modular design allows reuse of the security model across multiple solutions.



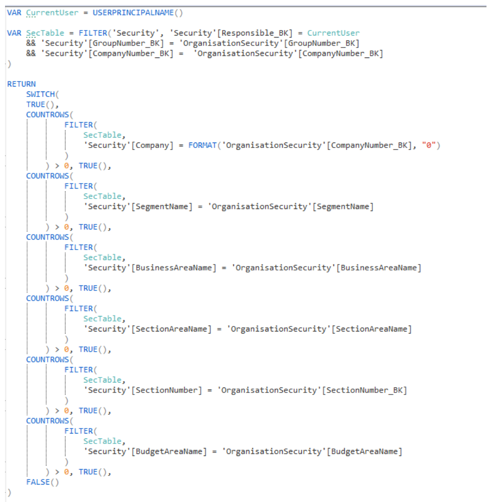
GroupNumber_BK	CompanyNumber_BK	Responsible_BK	AccessType	Company	SegmentName	BusinessAreaName	SectionAreaName	SectionNumber	BudgetAreaName	MainProjectNumber
-1	-1	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
1	1	aan@aarsleff.com	Project	No Access	No Access	No Access	No Access	565	No Access	No Access
1	1	aauj@aarsleff.com	Project	No Access	No Access	No Access	No Access	595	No Access	No Access
1	1	abas@aarsleff.com	Project	No Access	No Access	No Access	No Access	586	No Access	No Access
1	1	abr@aarsleff.com	Project	No Access	No Access	No Access	No Access	555	No Access	No Access
1	1	aea@aarsleff.com	Asset	No Access	No Access	No Access	No Access	442	No Access	No Access
1	1	aea@aarsleff.com	CRM	No Access	No Access	No Access	No Access	442	No Access	No Access
1	1	aea@aarsleff.com	Customer	No Access	No Access	No Access	No Access	442	No Access	No Access
1	1	aea@aarsleff.com	Finance	No Access	No Access	No Access	No Access	442	No Access	No Access
1	1	aea@aarsleff.com	HealthSafety	No Access	No Access	No Access	No Access	442	No Access	No Access
1	1	aea@aarsleff.com	PipeTech	No Access	No Access	No Access	No Access	442	No Access	No Access

5. Applying Row-Level Security

Row-level security is enforced in **Power BI** using:

- The **service principal of the logged-in user.**
- During login, the user is identified, and a **filtered subtable** of dim_security_all is created for that user.
- Filters are applied iteratively on the relevant scope columns until only the allowed **organizational units** remain.

The end result: Each user sees only the data they are entitled to, based on the central security table.



```
VAR CurrentUser = USERPRINCIPALNAME()

VAR SecTable = FILTER('Security', 'Security'[Responsible_BK] = CurrentUser
    && 'Security'[GroupNumber_BK] = 'OrganisationSecurity'[GroupNumber_BK]
    && 'Security'[CompanyNumber_BK] =  'OrganisationSecurity'[CompanyNumber_BK]
)

RETURN
    SWITCH(
    TRUE(),
    COUNTROWS(
            FILTER(
                SecTable,
                'Security'[Company] = FORMAT('OrganisationSecurity'[CompanyNumber_BK], "0")
            )
        ) > 0, TRUE(),
    COUNTROWS(
            FILTER(
                SecTable,
                'Security'[SegmentName] = 'OrganisationSecurity'[SegmentName]
            )
        ) > 0, TRUE(),
    COUNTROWS(
            FILTER(
                SecTable,
                'Security'[BusinessAreaName] = 'OrganisationSecurity'[BusinessAreaName]
            )
        ) > 0, TRUE(),
    COUNTROWS(
            FILTER(
                SecTable,
                'Security'[SectionAreaName] = 'OrganisationSecurity'[SectionAreaName]
            )
        ) > 0, TRUE(),
    COUNTROWS(
            FILTER(
                SecTable,
                'Security'[SectionNumber] = 'OrganisationSecurity'[SectionNumber_BK]
            )
        ) > 0, TRUE(),
    COUNTROWS(
            FILTER(
                SecTable,
                'Security'[BudgetAreaName] = 'OrganisationSecurity'[BudgetAreaName]
            )
        ) > 0, TRUE(),
    FALSE()
)
```

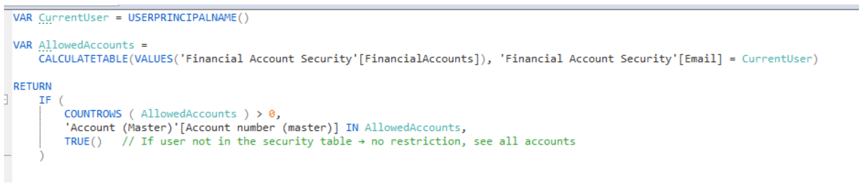
6. Adding additional security

One of the tasks i've had is that in some cases it might be that a person would benefit from viewing some data, but their organizational placement is only really for one domain. We have an example like that. For example we had a person that is mostly responsible for the asset domain, but it would also be meaningful to view other domains like finance. But to view this domain with the organizational placement would allow that person to see something that the person is not necessarily meant to be able to see.

So what we wanted to add was some option for using assigning specific accounts that the user would be able to see. For example to view the accounts that are related to assets. To do that we needed to

1) Create another security table, one that have the users email and the accounts that this user should be able to view.
2) Add that table to the data model
3) Create a new table permission for the read role on the master accounts table

The logic for filtering down on the master account table is as follows:



```
VAR CurrentUser = USERPRINCIPALNAME()

VAR AllowedAccounts =
    CALCULATETABLE(VALUES('Financial Account Security'[FinancialAccounts]), 'Financial Account Security'[Email] = CurrentUser)

RETURN
    IF (
        COUNTROWS ( AllowedAccounts ) > 0,
        'Account (Master)'[Account number (master)] IN AllowedAccounts,
        TRUE()   // If user not in the security table → no restriction, see all accounts
    )
```

This is one way of handling security on a row-level and in this case with the use of some manual spreadsheets that can control which datasets, at which scope and potentially also some financial accounts.