Reaching maximum automation

Generating semantic models as part of your platform workloads

Conference Partners

Data Point Prague 2025

Learning objectives

Semantic Link	Automate	Opinionated view
Know exactly what Semantic Link (labs) is, how you can use it in your benefit to power your solutions.	Understand till which extend your semantic model generation can be automated using Semantic Link (labs)	An opinionated view on who should be developing the semantic model, will it be the data engineer of the data analyst?

Marc Lelijveld

Technical Evangelist | Solution Architect
Macaw Netherlands

Microsoft® Most Valuable Professional

@marclelijveld.bsky.social

linkedin.com/in/MarcLelijveld

Data-Marc.com

DutchFabricUsergroup.com

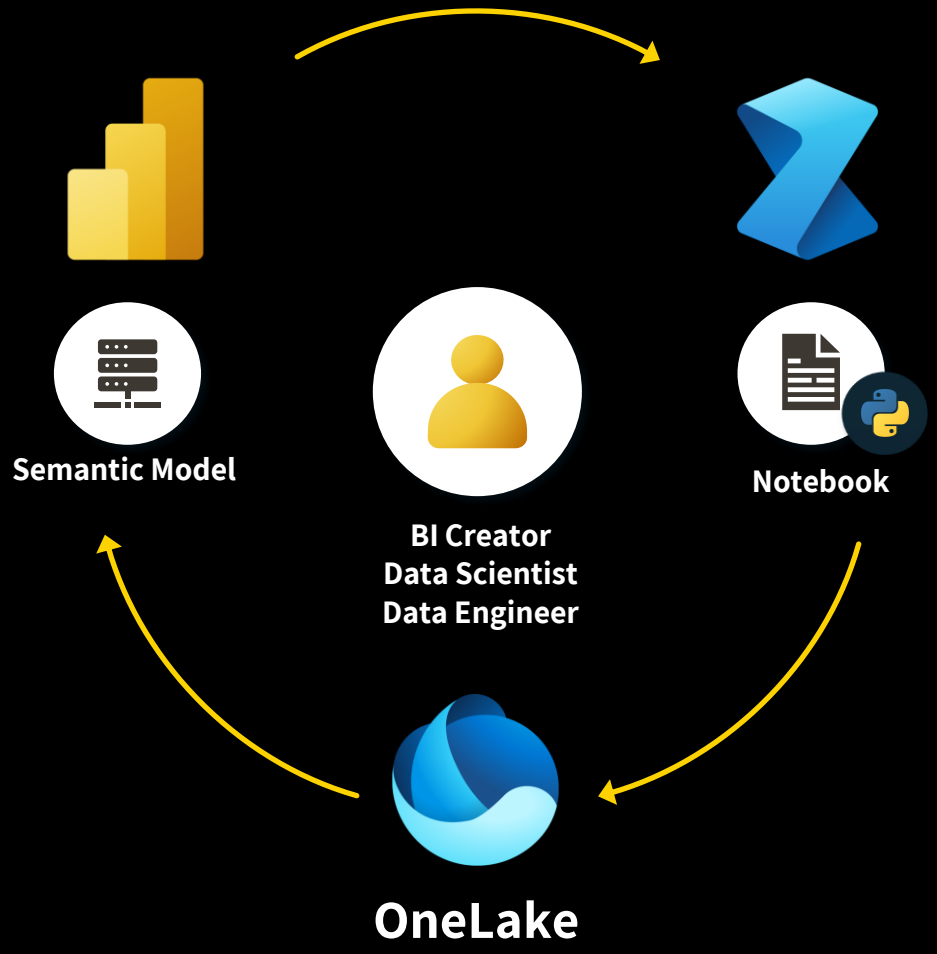
FAVORITE STUFF:

Setting the scene

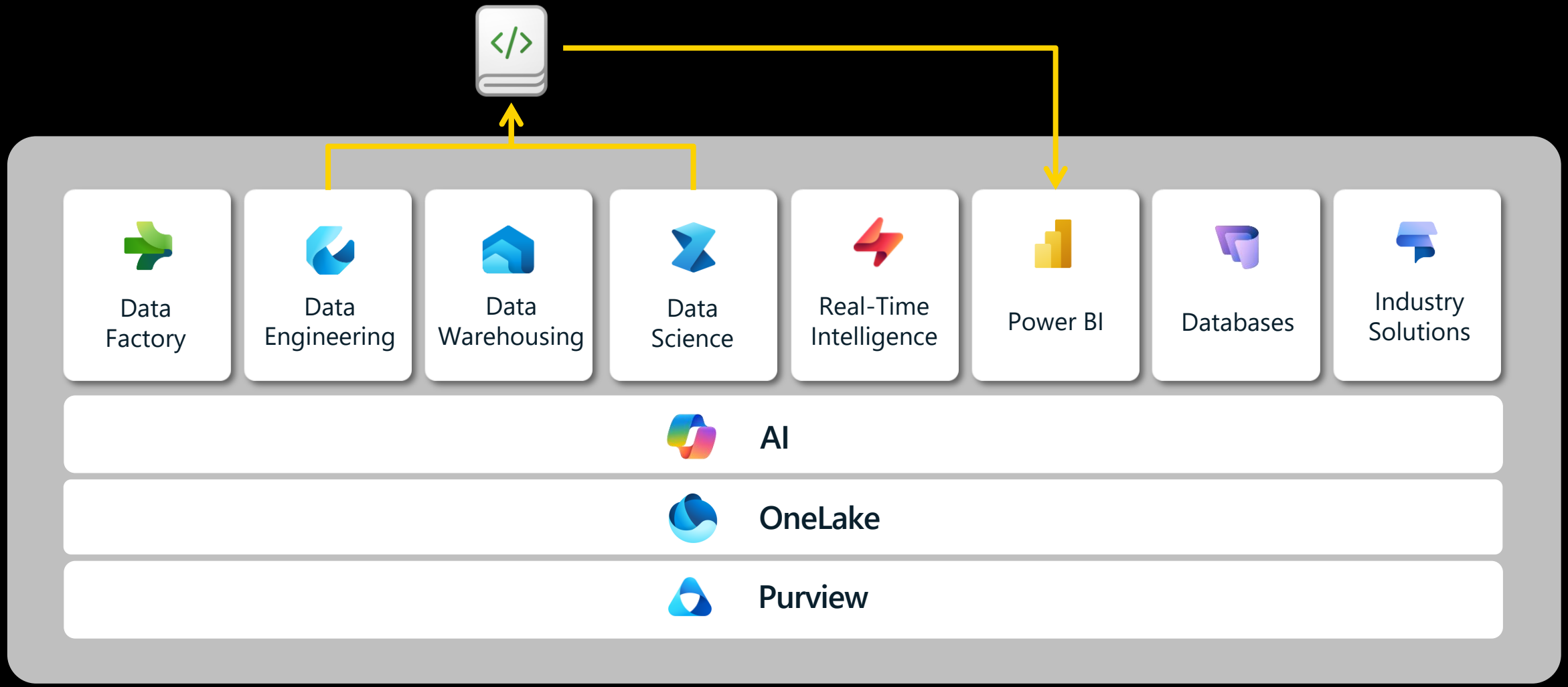
What is Semantic Link?

Semantic Link is a feature in Microsoft Fabric that allows you to connect from Data Science / Engineering **Notebooks** to Power BI Semantic Models.

This feature **only** exists and works in Microsoft Fabric.



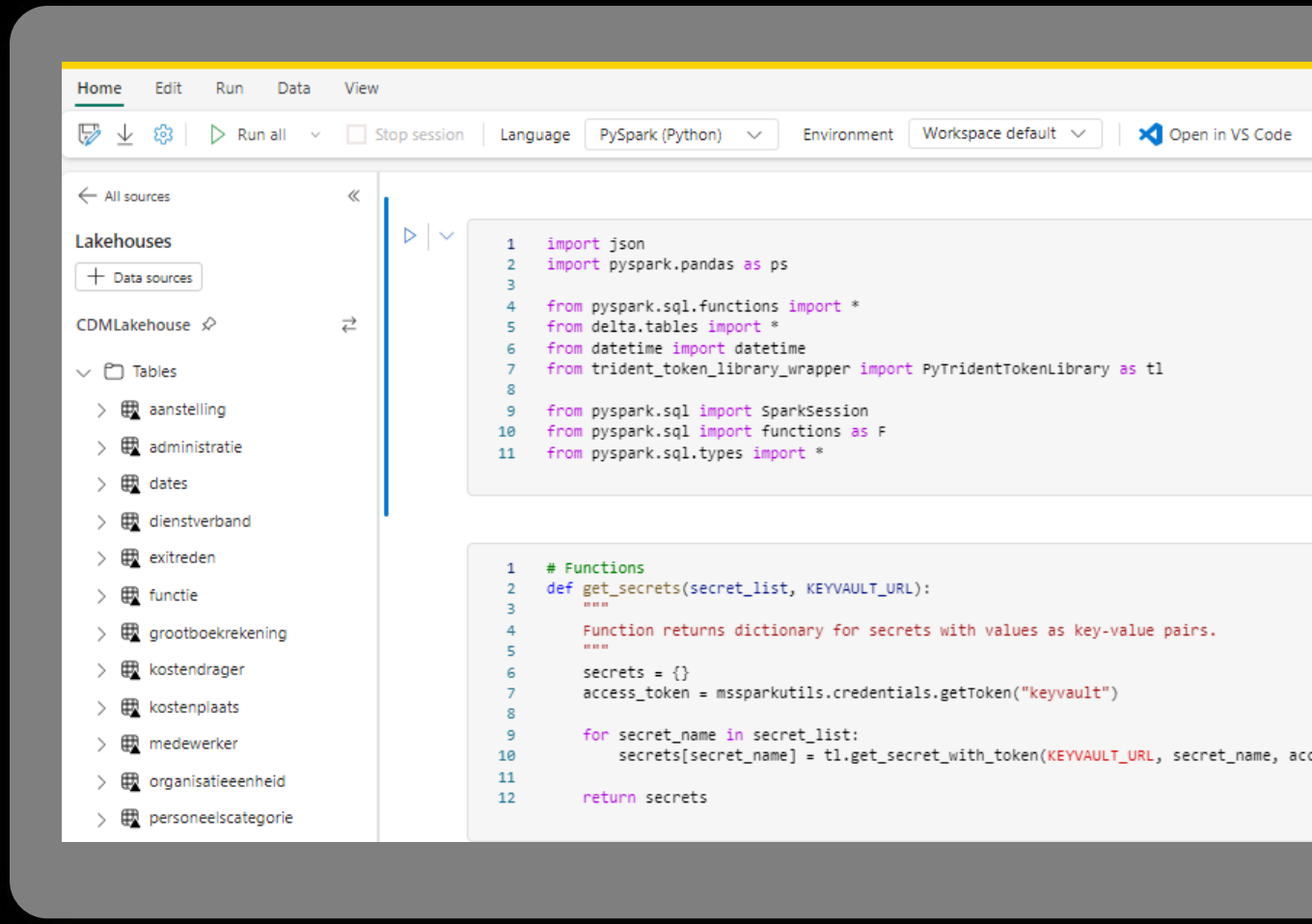
Where does Semantic Link fit in?



What are notebooks?

- Code first
- Web-based interface
- Cell based code blocks
- Runs on nodes (part of Fabric capacity)
- Often used languages are Python, Spark & Markdown

- Used by data engineers for data ingest, prep and transformations
- Used by data scientist for experiments and models



What is Semantic Link?

- Python library for interacting with semantic models
- Used inside of a Fabric notebook
- Pre-installed in the Fabric runtime
- Supported by Microsoft
- List_* functions to view semantic model metadata
- Functions to execute DAX, TMSL, XMLA, DMVs
- Basic API wrapper functions (list_capacities, list_workspaces, list_reports, list_items)
- And more…

Semantic Link – Use cases

Power BI general

- Documenting Power BI items
- Move Power BI items across workspaces
- Detect broken reports
- Rebind reports
- Set a report theme
- Migration of report-level measures to the semantic model
- Tenant Settings tracking

Semantic Models

- Best Practice Analyzer
- Vertipaq Analyzer
- Semantic model edits (TOM)
- Metadata translations
- Semantic model refresh
- Visualize a refresh
- Semantic model backups
- Run DAX with impersonation
- Manage Query Scale Out

Direct Lake

- Migration to Direct Lake
- Check Direct Lake guardrails
- Warm the cache for Direct Lake
- Analyze Delta tables for Direct Lake
- Fallback to DirectQuery diagnostics
- Update connection of a Direct Lake semantic model

Capacities

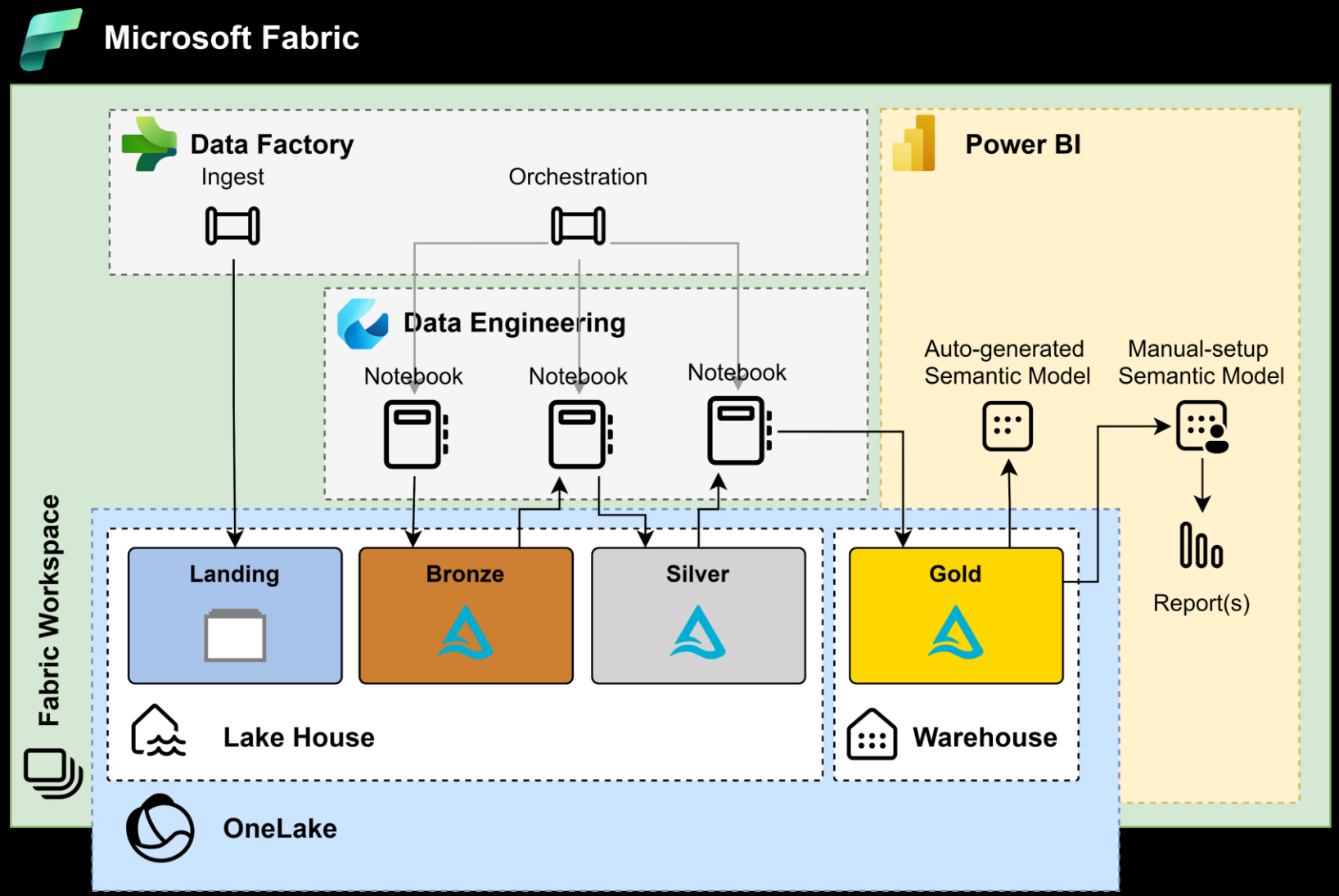
- Migration from P SKUs to F SKUs
- Migration from FT SKUs to F SKUs
- Capacity management

And more…

Demo – First exploration of Semantic Link

The debate

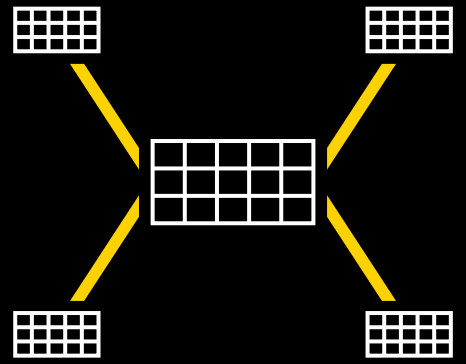
Typical analytics solution



Applied (best) practices

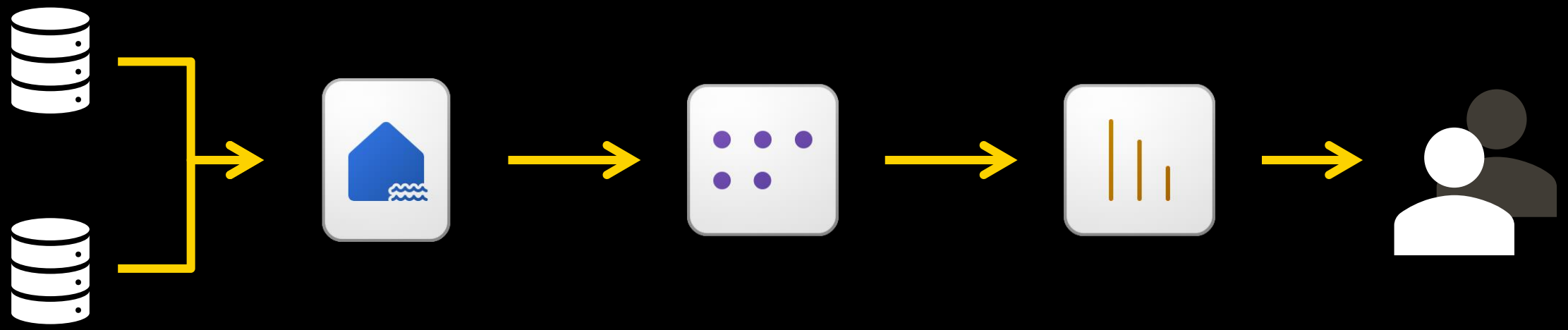
Structured platform striving for a single source of truth

Transformation done upstream



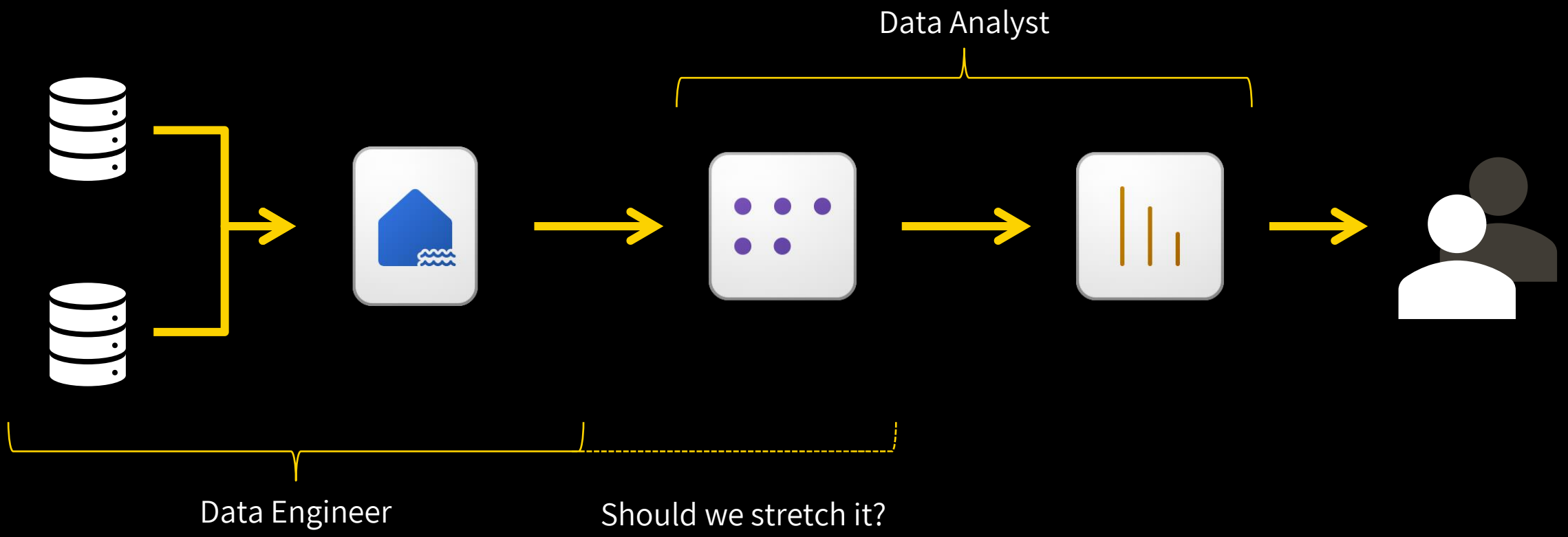
Semantics and business logic added

Typical analytics solution - simplified



But who builds the Semantic Model?

Typical analytics solution - simplified



Can we automate Semantic Model creation?

From basic to advanced

- Semantic Link mainly focusses on (meta)data extraction
- Propagation of semantic information

But is limited capable of advanced operations (…yet)

Take it one step further with Semantic Link Labs

- Initiatied by Michael Kovalsky (Fabric CAT)
- Extension to Semantic Link
- Programmatic access to Fabric items
- Wrapper functions for easy use of Power BI and Fabric APIs
- Full access to TOM (100+ functions)
- You **DO NOT** need to be a python expert!
- Open-sourced python library on GitHub
  https://github.com/microsoft/semantic-link-labs

Start with creating a Semantic Model first

sempy_labs.create_semantic_model_from_bim(*dataset: str, bim_file: dict, workspace: str | UUID | None = None*)

sempy_labs.create_blank_semantic_model(*dataset: str, compatibility_level: int = 1605, workspace: str | UUID | None = None, overwrite: bool = True*)

sempy_labs.directlake.generate_direct_lake_semantic_model(*dataset: str, lakehouse_tables: str | List[str], workspace: str | UUID | None = None, lakehouse: str | None = None, lakehouse_workspace: str | UUID | None = None, schema: str = 'dbo', overwrite: bool = False, refresh: bool = True*)

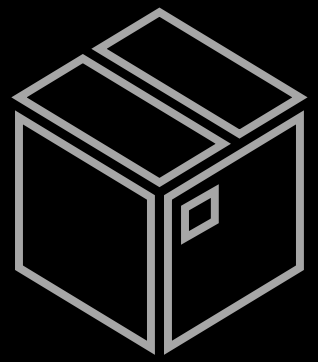
Dynamically generates a Direct Lake semantic model based on tables in a Fabric lakehouse.

Parameters:

- **dataset** (*str*) – Name of the semantic model to be created.
- **lakehouse_tables** (*str | List[str]*) – The table(s) within the Fabric lakehouse to add to the semantic model. All columns from these tables will be added to the semantic model.
- **workspace** (*str | uuid.UUID, default=None*) – The Fabric workspace name or ID in which the semantic model will reside. Defaults to None which resolves to the workspace of the attached lakehouse or if no lakehouse attached, resolves to the workspace of the notebook.
- **lakehouse** (*str, default=None*) – The lakehouse which stores the delta tables

Demo – Generate Semantic Models

Medallion architecture – and its complexity



Landing (optional)
Getting the data in, format as is.

Bronze
Standardizing format, building up history.

Silver
Canonical model, combined, most rich.

Gold
Serving purposes, no transformations.

Often generated and fully automated
Python heavy

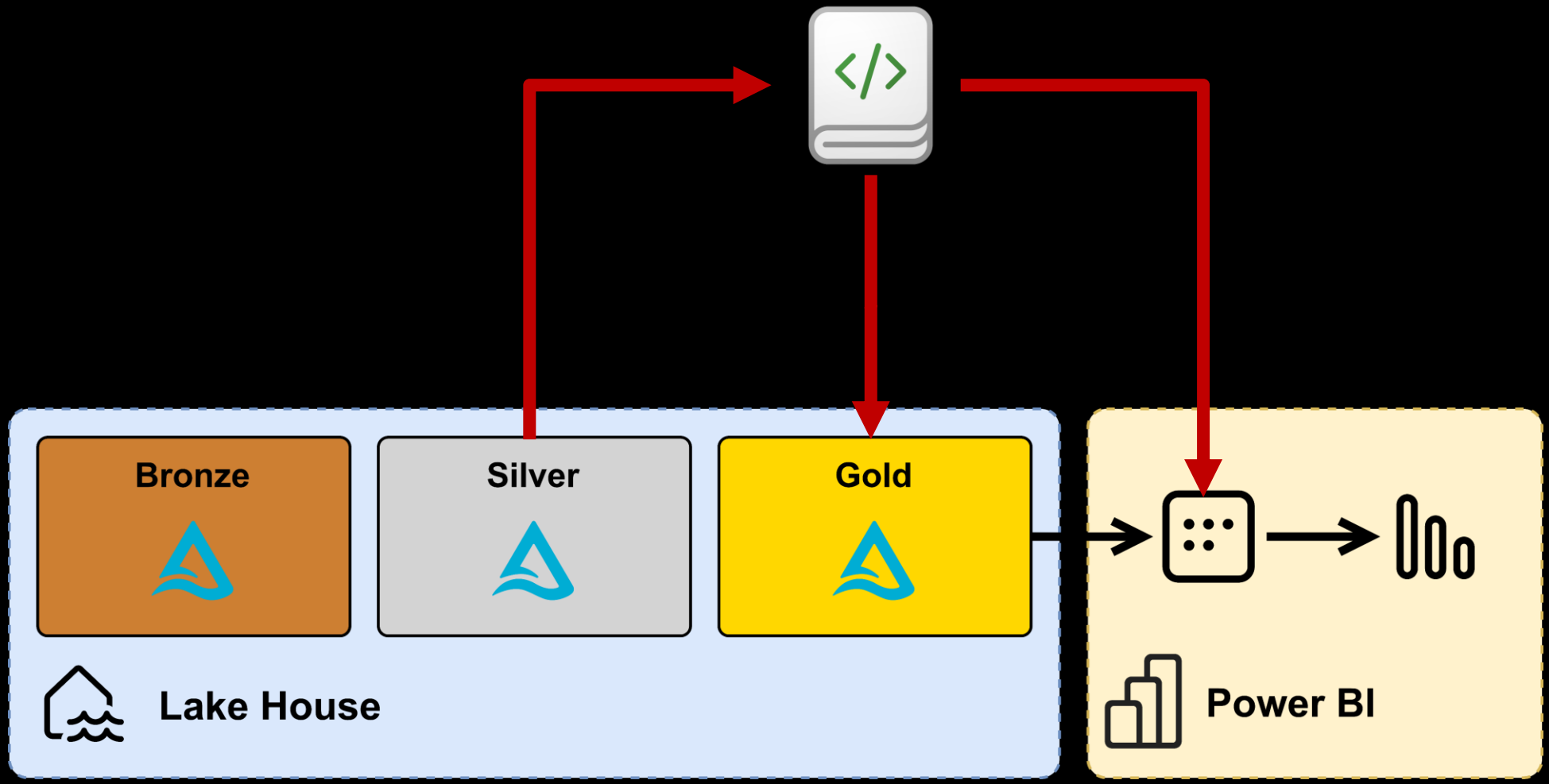
Custom
T-SQL & Python

Adding tables

We have an empty Semantic Model, now we want to add tables to it.

- Gold layer tables are often processed / generated one-by-one
- Add additional steps in gold Notebooks to
  - Add table to Semantic Model(s)
  - Add relationships to semantic model

Positioning



Relationship types

There are **three types of relationships** in data modeling

1 to 1

Every **individual record** in dataset A is mapped to one **individual record** in dataset B

Dataset A
Customer information

Customer ID	Country
AW000111024	United States
AW00019377	Germany

Dataset B
Customer information

Customer ID	Date of Birth
AW000111024	9 april 1990
AW00019377	9 april 1983

1 to many

One record of dataset A is mapped to **multiple records** in dataset B

Dataset A
Customer information

Customer ID	Country
AW000111024	United States
AW00019377	Germany

Dataset B
Sales Information

Customer ID	Product	Order Quantity
AW00011024	CA-1098	2
AW00019377	BC-M005	1
AW00019377	CA-1098	1
AW00019377	FE-6654	1
AW00019377	HL-U509-B	1
AW00019377	TI-M602	1
AW00019377	TT-M928	1
AW00019377	WB-H098	1

Many to many

Multiple records of dataset A are mapped to **multiple records** in dataset B

Dataset A
Customer information

Customer ID	Store ID	Store Type
AW00011024	ON-1	Online
AW00011024	ST-1	Regional City Store
AW00011024	ST-2	Regional City Store
AW00019377	ON-1	Online
AW00019377	ST-1	Regional City Store
AW00019377	ST-2	Regional City Store

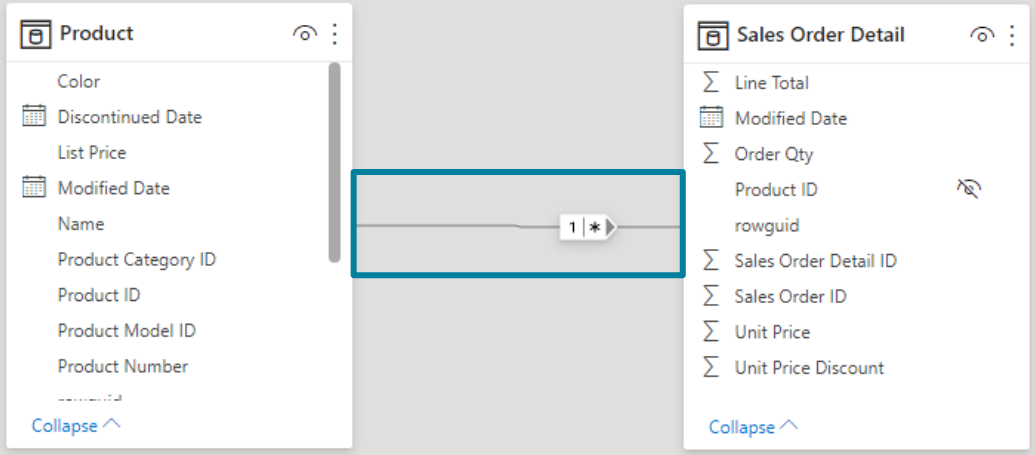
Dataset B
Sales Information

Customer ID	Product	Order Quantity
AW00011024	CA-1098	2
AW00011024	TT-M928	5
AW00019377	BC-M005	1
AW00019377	CA-1098	1
AW00019377	FE-6654	1
AW00019377	HL-U509-B	1
AW00019377	TI-M602	1
AW00019377	TT-M928	1
AW00019377	WB-H098	1

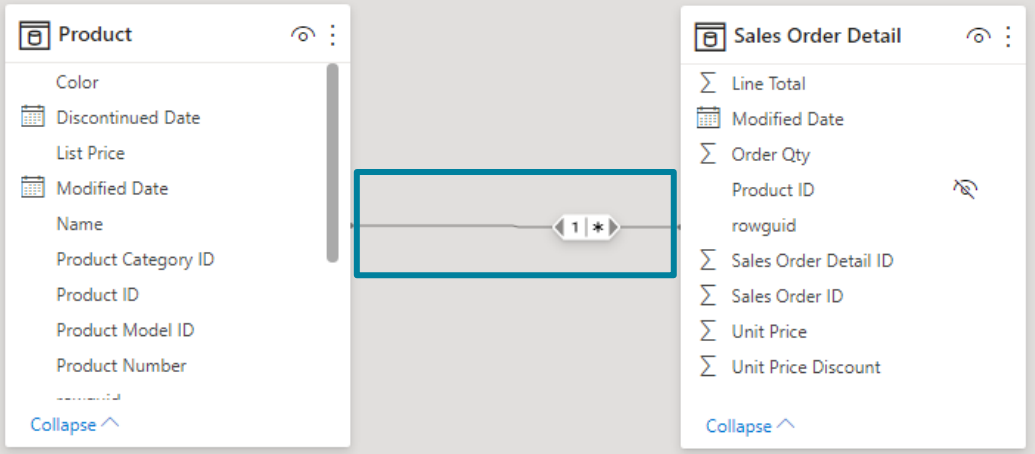
Relationship direction

There are **two types** of relationship directions

Singular



Bi-directional



Semantic Link Labs – Generate Relationships

Looks simple at first:

sempy_labs.create_relationship_name(*from_table: str, from_column: str, to_table: str, to_column: str*) → str

Formats a relationship's table/columns into a fully qualified name.

Parameters:
- **from_table** (*str*) – The name of the table on the 'from' side of the relationship.
- **from_column** (*str*) – The name of the column on the 'from' side of the relationship.
- **to_table** (*str*) – The name of the table on the 'to' side of the relationship.
- **to_column** (*str*) – The name of the column on the 'to' side of the relationship.

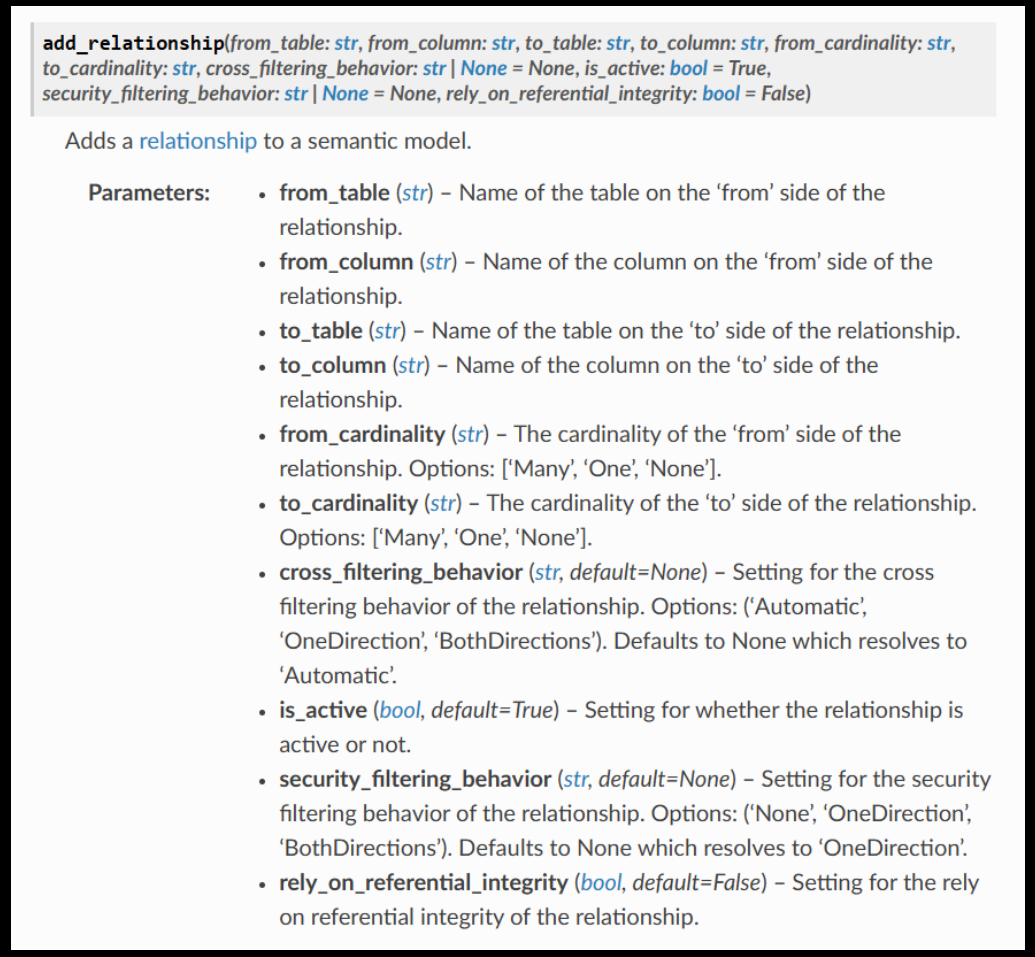
Returns: The fully qualified relationship name.

Return type: str

Semantic Link Labs – Generate Relationships

More complex that hoped for…

- Function used, just returns a text string. Nothing more, nothing less…
- Adding relationships requires to use the **TOM package** (Tabular Object Model) *(You end up using .NET and C#)*



add_relationship(*from_table: str, from_column: str, to_table: str, to_column: str, from_cardinality: str, to_cardinality: str, cross_filtering_behavior: str | None = None, is_active: bool = True, security_filtering_behavior: str | None = None, rely_on_referential_integrity: bool = False*)

Adds a relationship to a semantic model.

Parameters:

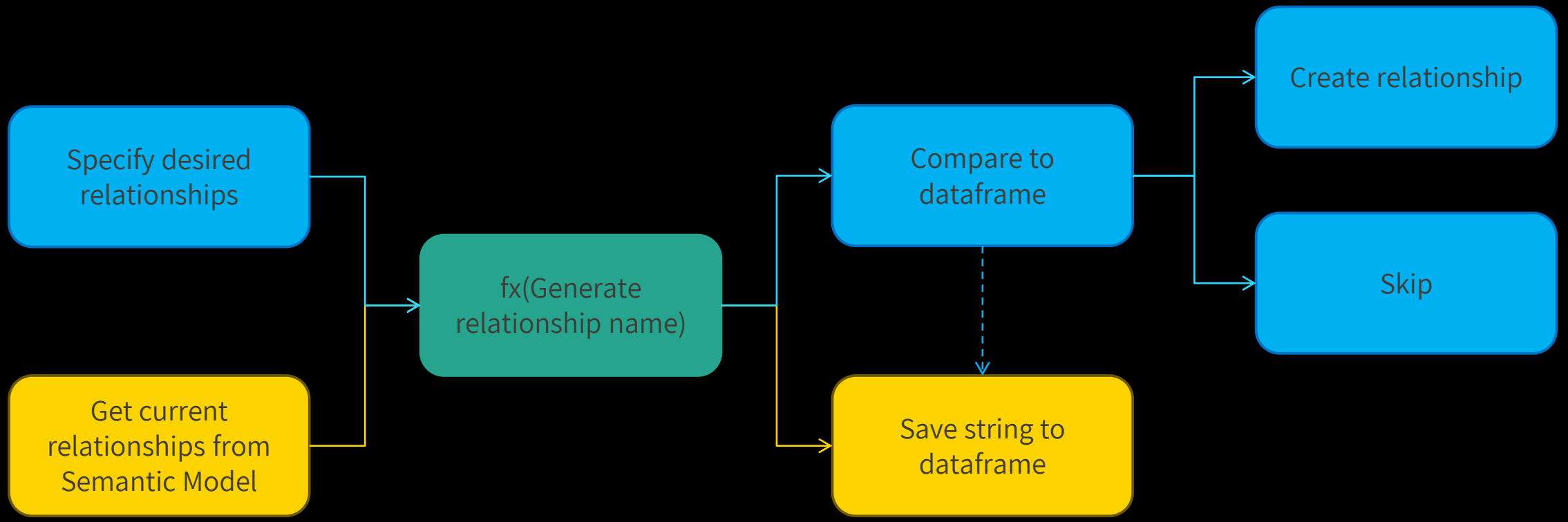
- **from_table** (*str*) – Name of the table on the 'from' side of the relationship.
- **from_column** (*str*) – Name of the column on the 'from' side of the relationship.
- **to_table** (*str*) – Name of the table on the 'to' side of the relationship.
- **to_column** (*str*) – Name of the column on the 'to' side of the relationship.
- **from_cardinality** (*str*) – The cardinality of the 'from' side of the relationship. Options: ['Many', 'One', 'None'].
- **to_cardinality** (*str*) – The cardinality of the 'to' side of the relationship. Options: ['Many', 'One', 'None'].
- **cross_filtering_behavior** (*str, default=None*) – Setting for the cross filtering behavior of the relationship. Options: ('Automatic', 'OneDirection', 'BothDirections'). Defaults to None which resolves to 'Automatic'.
- **is_active** (*bool, default=True*) – Setting for whether the relationship is active or not.
- **security_filtering_behavior** (*str, default=None*) – Setting for the security filtering behavior of the relationship. Options: ('None', 'OneDirection', 'BothDirections'). Defaults to None which resolves to 'OneDirection'.
- **rely_on_referential_integrity** (*bool, default=False*) – Setting for the rely on referential integrity of the relationship.

Semantic Link Labs – Generate Relationships

Adding relationships may returns errors if you try to create a relationship that already exists, due to;

- Semantic model does not allow two active relationships at the same time
- Two the same relationships at the same time between the same tables are not allowed

Generate Relationships flow



Demo – Adding Relationships

Flow of actions

1	2	3	4	5
Create placeholder table (_Measures)	**Generate** Semantic Model	**Add** dimension tables	**Add** fact table(s)	**Add** relationship(s)

But we’re not there yet…

Some things are still missing…

- Measures
- Calculation Groups
- Translations *(probably fine to generate)*
- Perspectives

All can be created and generated using the TOM wrapper, however…

Pretty please hand-over to a Data Analyst, who knows the business well and can apply this logic to the semantic model.

Recap & gotcha!

Semantic Link

- …access your semantic model in a programmatic interface
- …allows you to read (meta)data from your semantic model

Semantic Link Labs

- …is an open-source extension
- …goes the extra mile in automating
- …allows for complex patterns (like the TOM-wrapper)

Automating Semantic Models

- …is possible for sure
- …generation could be done by Data Engineers
- …business logic should be added by Data Analysts (opinionated view)

LET ME KNOW WHAT YOU THINK!

Thanks for attending!

Marc Lelijveld
Technical Evangelist | Architect
Macaw Netherlands

Microsoft® Most Valuable Professional

linkedin.com/in/MarcLelijveld

Data-Marc.com