



# DWH Automation Challenge

---

Automating Willibald with dbt and  
datavault4dbt

# Who are we

---



Jan Binge has gained over 25 years of experience in the field of IT, out of which he has spent more than a decade as a freelance consultant specializing in "data warehouse design". Following his certification as a Datavault Practitioner in 2014, he has directed his attention towards modeling and developing data warehouse systems while also emphasizing the automation of creation processes.

[jan@binge.de](mailto:jan@binge.de)

[linkedin.com/in/jan-binge](https://linkedin.com/in/jan-binge)



Andreas Haas has been working as a consultant in the business intelligence sector for over 20 years. During this time, he has successfully implemented data warehouse projects in various industries, mainly in the roles of data warehouse architect, data engineer and in project management. As a certified Data Vault 2.0 Practitioner, large metadata-driven Data Vault implementations are the main focus of his work.

[andreas@haas-erlangen.com](mailto:andreas@haas-erlangen.com)

[linkedin.com/in/haasandreas/](https://linkedin.com/in/haasandreas/)

# What is dbt

- dbt (data build tool) is a compiler
- focus is on data transformation - mainly using SQL
- each table or view (called model) is defined in a separate text-file
- dependency management (orchestration)
- automation and standardization with jinja2-macros
- no DML necessary, materialisations can be configured
- build in testing / documentation – capabilities
- open source (on-premise called dbt core)
- also available as SaaS (called dbt cloud) including
  - scheduling
  - API-calls
  - Integrated IDE
- dbt does not load data from other sources
- dbt has no scheduling feature



<https://www.getdbt.com/product/what-is-dbt/>

# Why we use dbt

---

- the integration in GIT provides a key-feature to a standardized development and deployment process.
- vibrant large community using, discussing ,extending the product (packages)
- easy to get up to speed
- imposed standardisation by using macros
- many target databases supported
- one platform for all data teams:
  - Data Engineer – Teams
  - Data Analytics – Teams
  - Data Science – Teams
- scalable projects (multiple parallel teams – data mesh)



# Structure of dbt - project

## DDVUG-WILLIBALD-SAMEN-DBT

- > .venv
- > analyses
- > dbt\_packages
- > logs
- > macros
- ✓ models
  - > dwh\_00\_meta
  - > dwh\_01\_ext
  - > dwh\_02\_load
  - > dwh\_03\_err
  - > dwh\_03\_stage
  - > dwh\_04\_rv
  - > dwh\_05\_sn
  - > dwh\_06\_bv
  - > dwh\_07\_inmt
  - ! sources.yml

We structured the models within the dbt-project in subfolders according to the layers.

Within the layers, we defined a subfolder for each business object including all models related to it (including tests)

- ✓ dwh\_04\_rv
  - > \_not\_dataspot\_sourced
  - > associationpartner
  - ✓ customer
    - customer\_associationpartner\_l.sql
    - customer\_associationpartner\_ws\_es.sql
    - customer\_associationpartner\_ws\_sts.sql
    - customer\_h.sql
    - customer\_ws\_la\_ms.sql
    - customer\_ws\_s.sql
    - customer\_ws\_sts.sql
    - ! test\_customer\_associationpartner\_l.yaml
    - ! test\_customer\_h.yaml
    - ! test\_customer\_ws\_s.yaml
  - > delivery
  - > deliveryadress
  - > deliveryservice
  - > order
  - > position
  - > product
  - > productcategory
  - > reference

# What is datavault4dbt



<https://github.com/ScalefreeCOM/datavault4dbt/wiki>

Datavault4dbt is an Open-Source package from Scalefree which includes a set of macros to create a fully compliant data vault 2.0 - data warehouse

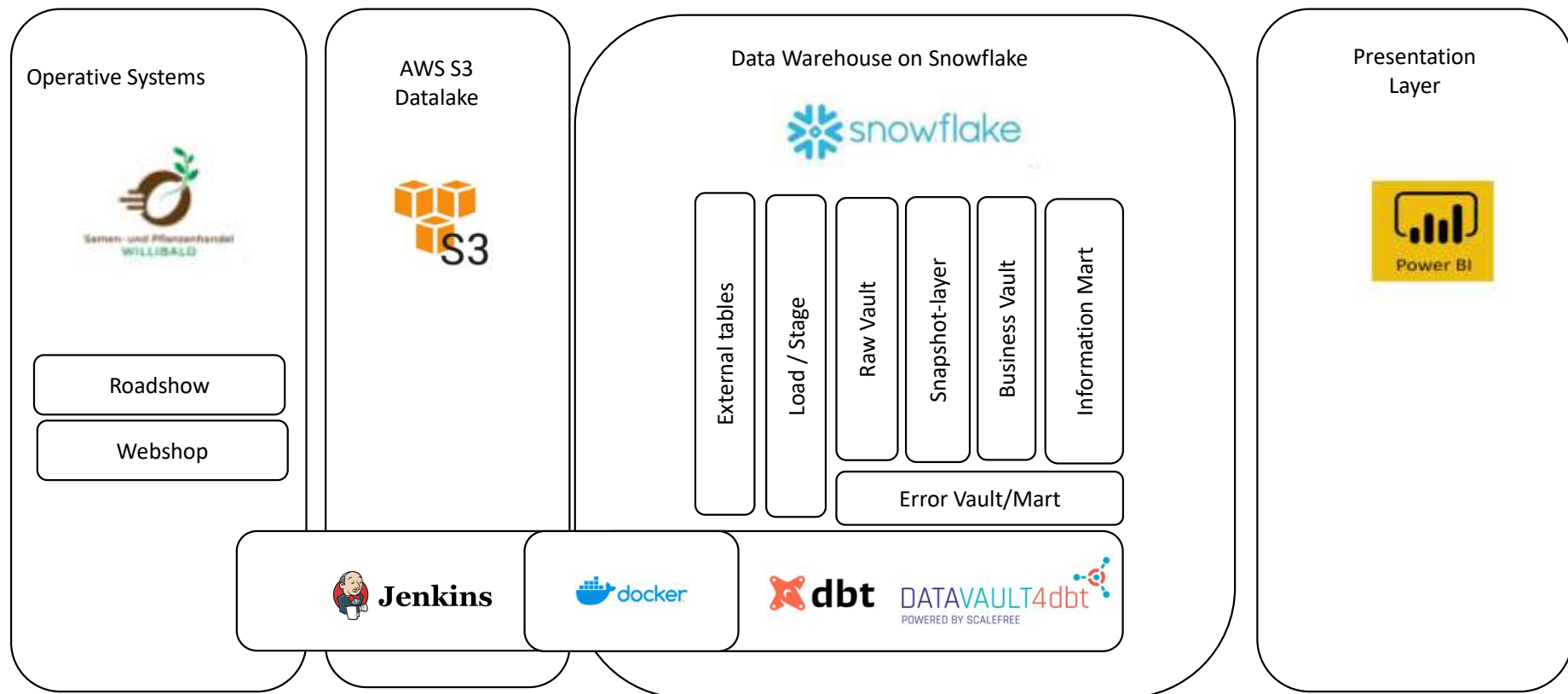
```
131 {% for source_model in source_models %}
132
133     {% set source_number = source_model.id | string -%}
134
135     {% if ns.has_rsrc_static_defined -%}
136         {% set rsrc_statics = ns.source_models_rsrc_dict.id -%}
137     {% endif -%}
138
139     {% if 'hk_column' not in source_model.keys() %}
140         {% set hk_column = hashkey -%}
141     {% else -%}
142         {% set hk_column = source_model['hk_column'] -%}
143     {% endif %}
144
145     src_new_{{ source_number }} AS (
146
147         SELECT
148             {{ hk_column }} AS {{ hashkey }},
149             {% for bk in source_model['bk_columns'] -%}
150                 {{ bk }},
151             {% endfor -%}
152
153             {{ src_ldts }},
154             {{ src_rsrc }}
155         FROM {{ ref(source_model.name) }} src
```

excerpt of the hub macro

```
4     {{ config(materialized='incremental') }}
5
6     {% set yaml_metadata -%}
7     source_models:
8         stg_webshop_lieferdienst:
9             bk_columns: 'deliveryservice_bk'
10             rsrc_static: '*/webshop/lieferdienst/*'
11         stg_webshop_lieferung:
12             bk_columns: 'deliveryservice_bk'
13             rsrc_static: '*/webshop/lieferung/*'
14     hashkey: hk_deliveryservice_h
15     business_keys:
16         - 'deliveryservice_bk'
17     {% endset -%}
18
19     {% set metadata_dict = fromyaml(yaml_metadata) %}
20
21     {{ datavault4dbt.hub(source_models=metadata_dict["source_models"],
22                         hashkey=metadata_dict["hashkey"],
23                         business_keys=metadata_dict["business_keys"]) }}
```

„dbt-model“ with all necessary parameters calling the hub macro

# Automation Challenge dbt-Setup



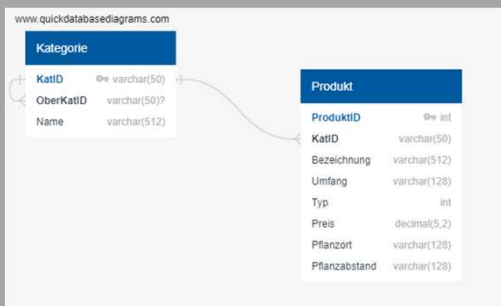
# Evaluation criteria

---

- (1) Hierarchical Link
- (2) Multi-active Satellite
- (3) Validity in Relationships
- (4) m:n Tables
- (5) Early Integration
- (6) (Historized) reference table
- (7) Duplicate Data
- (8) Rows without business keys
- (9) Changes of Attributes
- (10) Deletion of Business Keys
- (11) Invalid foreign Keys
- (12) Deletion of Orders
- (13) Change of Dimensions
- (14) Key Figures
- (15) Business Rules
- (16) Data Lineage
- (17) Error Handling
- (18) Orchestration
- (19) Deployment
- (20) Scheduling
- (21) Supported Databases
- (22) Installation Requirements
- Yedi-tests
- High Water Marking

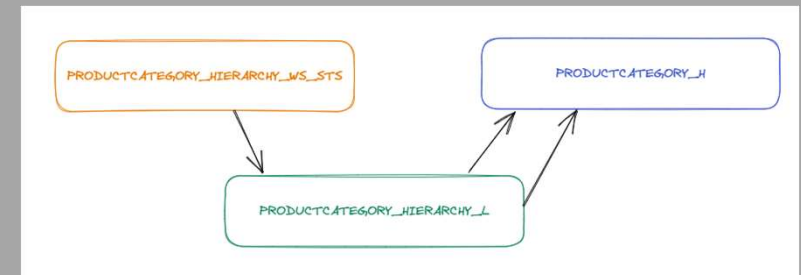


# Evaluation criterium 1: Hierarchical Link



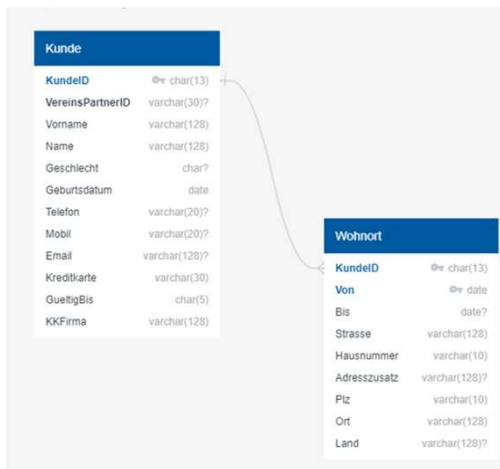
There is a table Kategorie in the webshop, which includes a parent-child relationship between KatID and OberKatID with several levels

REPORTING_DATE	PRODUCTCATEGORY_L3	PRODUCTCATEGORY_L2	PRODUCTCATEGORY_L1
12. März 2022	Amaranth	Mittelzehrer	Gemüse
14. März 2022	Amaranth	Mittelzehrer	Gemüse
19. März 2022	Amaranth	Gemüse	Mittelzehrer
26. März 2022	Amaranth	Mittelzehrer	Gemüse



Implemented as a hierarchical link productcategory\_hierarchy\_l (standard macro in datavault4dbt) and a status-satellite, to identify potential changes

# Evaluation criterium 2: multi-active satellite



In this case, several valid rows are delivered for each customer, the multiactive-key is **von**. There is a standard-macro for multi-active Satellites in datavault4dbt.

```
multi_active_config:
  multi_active_key:
    - von
  main_hashkey_column: hk_customer_h
```

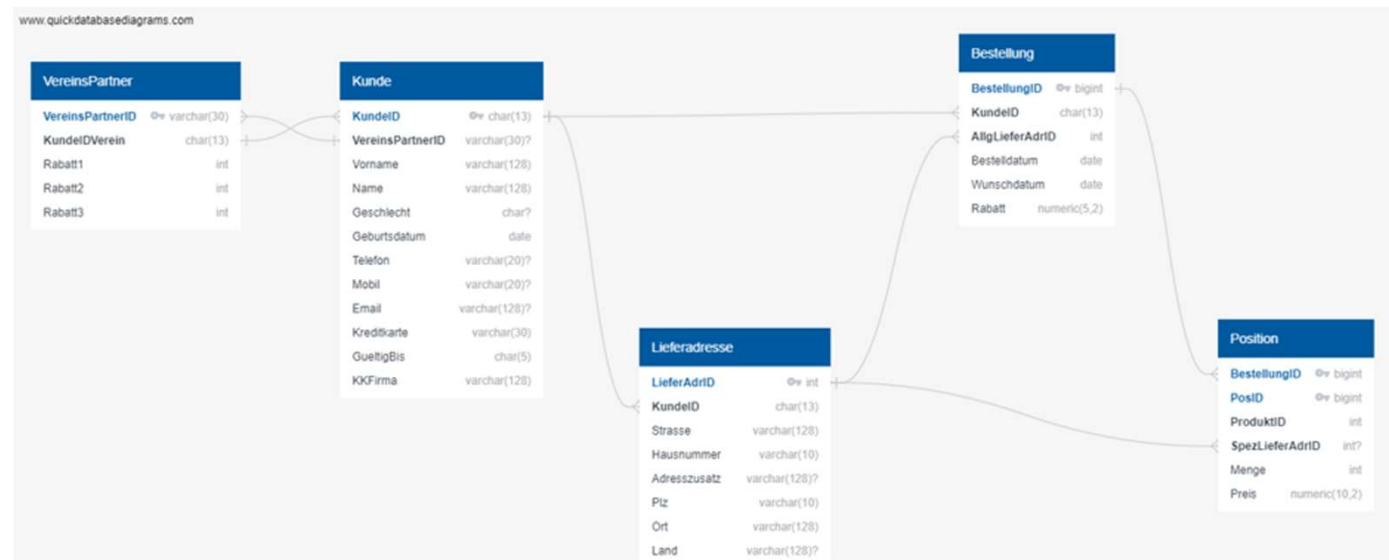
Each delivered set gets the same Hash\_Diff. That way a change in one entry leads to including the complete set new.

LDTS	ABC CUSTOMER_BK	VON	BIS	ABC STRASSE	ABC HAUSNUMMER	ABC HD_LIVINGADDRESS_WS_MS
2022-03-26 08:00:00.000	344	2008-04-27	2010-08-04	Niederseelbach	83	2211e62f74e463eb2dc97a1cbc047013
2022-03-26 08:00:00.000	344	2010-08-05	[NULL]	Kötenicher Straße	179	2211e62f74e463eb2dc97a1cbc047013
2022-03-19 08:00:00.000	344	1988-11-16	1992-08-25	Niederseelbach	83	135fb29e790e3ad67bb49f0d4e401d38
2022-03-19 08:00:00.000	344	1992-08-26	[NULL]	Kötenicher Straße	179	135fb29e790e3ad67bb49f0d4e401d38

# Evaluation criterium 3: Validity in Relationships

The relationship between ORDER (Bestellung) and POSITION (Position) cannot change. The key situation makes every change a deletion and a new creation.

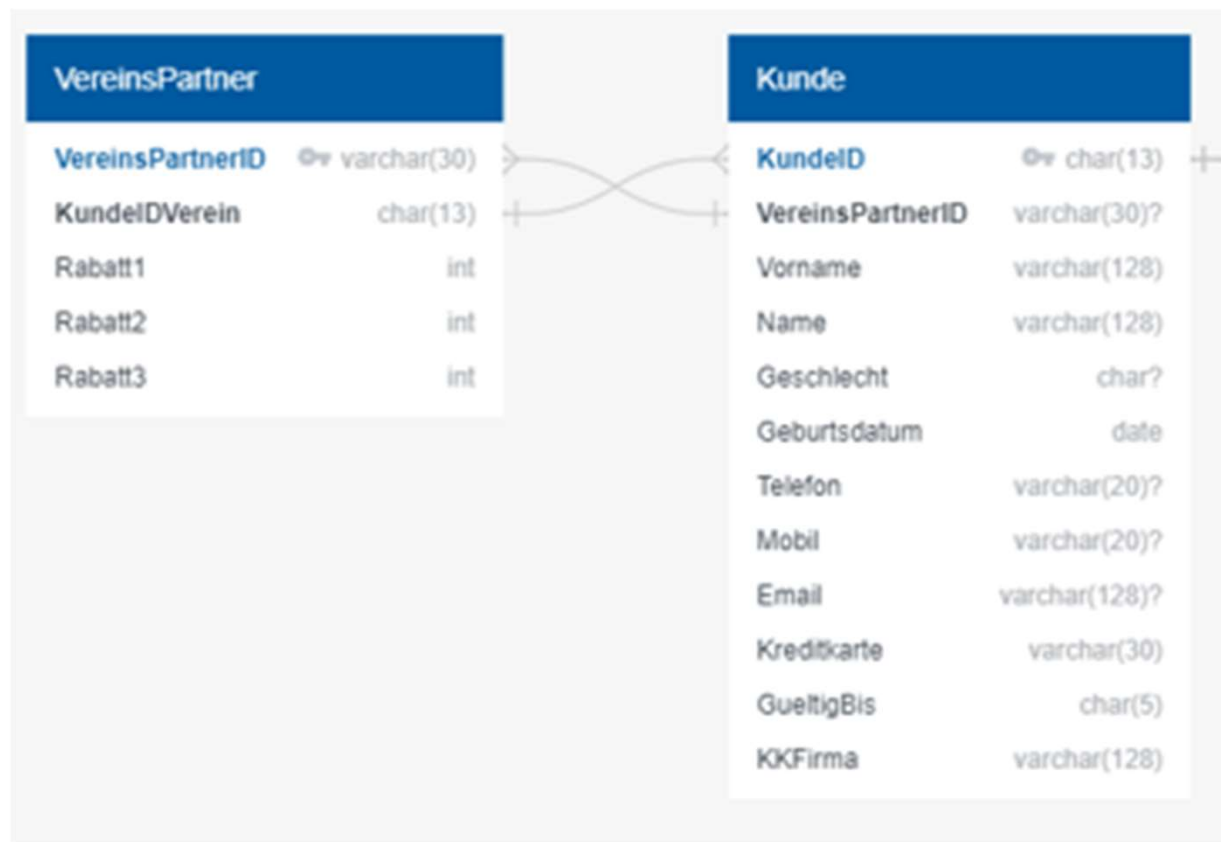
All other relationships can change. The test cases are all implemented on the foreign key in CUSTOMER (Kunde) to ASSOCIATION PARTNER (Vereinspartner).



The following situations occur here:

- the foreign key is optional and therefore also NULL
- The foreign key changes between ASSOCIATION PARTNERS
- The foreign key changes from "valid" to "invalid" - and some cases then even back to "valid" again

# Evaluation criterium 3: Validity in Relationships



# Evaluation criterium 3 (cont): Validity in Relationships

## Identifying Relationship and Driving Keys

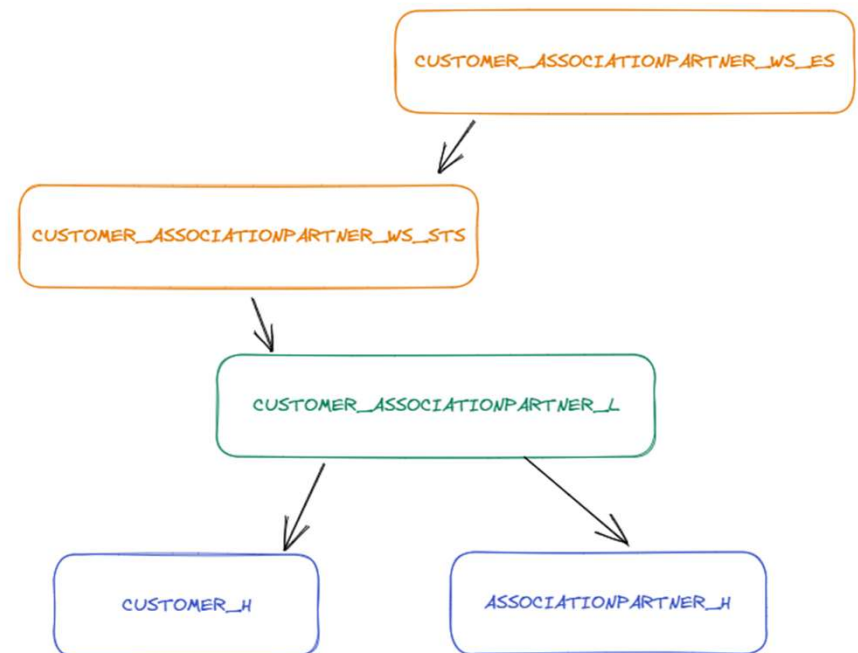
To track and identify changes of relationships, we defined **status satellites** (postfixed with `_sts`), which can be used for either links or hubs.

They contain a cdc-attribute (either "I" – insert or "D" – delete)

The load logic depends on the data-delivery (full-load, partial-load, cdc-delivery).

The idea here is, that further downstream no knowledge regarding the load-logic is necessary.

In case there is a driving-key for a link, an effectivity satellite view is set on top of the status-satellite and the link (postfixed with `_es`), which enddates all entries no longer valid.



# Evaluation criterium 4: m:n Tables

We decided to define a **non-historized-link** (postfixed with **\_nhl**) for that case.

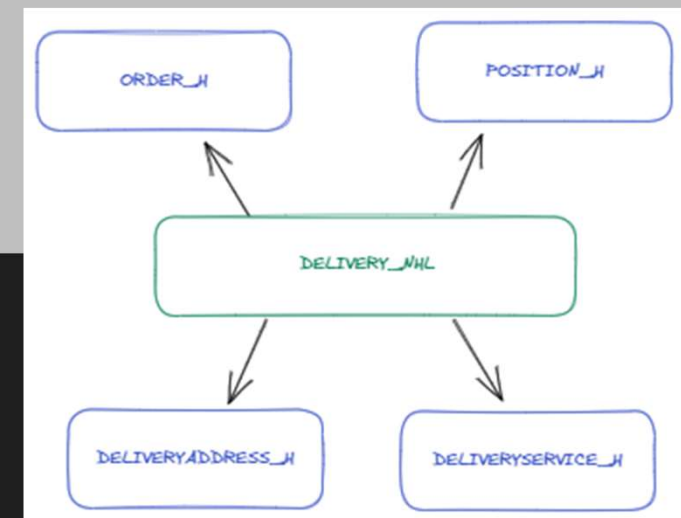
Solving this topic using a keyed instance or a dependent child link would be possible as well.

```
{{ config(materialized='incremental') }}
{% set yaml_metadata -%}
source_models: stg_webshop_lieferung
link_hashkey: hk_delivery_l
foreign_hashkeys:
  - 'hk_deliveryaddress_h'
  - 'hk_deliveryservice_h'
  - 'hk_order_h'
  - 'hk_position_h'
payload:
  - lieferdatum
{% endset -%}

{% set metadata_dict = fromyaml(yaml_metadata) -%}

{% set link_hashkey = metadata_dict['link_hashkey'] -%}
{% set foreign_hashkeys = metadata_dict['foreign_hashkeys'] -%}
{% set payload = metadata_dict['payload'] -%}
{% set source_models = metadata_dict['source_models'] -%}

{{ datavault4dbt.nh_link(link_hashkey=link_hashkey,
                        foreign_hashkeys=foreign_hashkeys,
                        payload=payload,
                        source_models=source_models) }}
```



# Evaluation criterium 5: Early Integration

There are orders from the webshop and roadshow. The orders and positions of both source-systems are loaded into the Raw Vault Hubs order\_h and position\_h. Describing data is saved in separate satellites for each source-system.

Each source-table loading into position\_h needs to be defined as source model using the standard datavault4dbt hub-macro. In the roadshow dataset, there is no positionID available, that's why we defined it concatenating bestellungid and produktid on stage-level.

```
{{ config(materialized='incremental') }}

{% set yaml_metadata -%}
source_models:
  stg_roadshow_bestellung:
    bk_columns: 'position_bk'
  stg_webshop_lieferung:
    bk_columns: 'position_bk'
  stg_webshop_position:
    bk_columns: 'position_bk'
hashkey: hk_position_h
business_keys:
  - 'position_bk'
{% endset -%}

{% set metadata_dict = fromyaml(yaml_metadata) %}

{{ datavault4dbt.hub(source_models=metadata_dict["source_models"],
                    hashkey=metadata_dict["hashkey"],
                    business_keys=metadata_dict["business_keys"]) }}
```

„dbt-model“ with all necessary parameters calling the macro hub

# Evaluation criterium 5: Early Integration

```
models > dwh_03_stage > bestellung > stg_roadshow_bestellung.sql
50
51 derived_columns:
52   position_bk:      You, 3 months ago * . . .
53     value: cast(BESTELLUNGID || '_' || produktid || '_' || cast(row_number() over (partition by ldts, bestellungid, produktid order by menge, preis) as varchar) as varchar)
54     datatype: 'VARCHAR'
```

Defining a position business key for roadshow on stage-level

```
{{ config(materialized='incremental') }}

{% set yaml_metadata -%}
source_models:
  stg_roadshow_bestellung:
    bk_columns: 'position_bk'
  stg_webshop_lieferung:
    bk_columns: 'position_bk'
  stg_webshop_position:
    bk_columns: 'position_bk'
hashkey: hk_position_h
business_keys:
  - 'position_bk'
{% endset -%}

{% set metadata_dict = fromyaml(yaml_metadata) %}

{{ datavault4dbt.hub(source_models=metadata_dict["source_models"],
                    hashkey=metadata_dict["hashkey"],
                    business_keys=metadata_dict["business_keys"]) }}
```

dbt-model with all necessary parameters calling the hub macro



# Evaluation criterium 6: (Historized) reference table

The data for delivery-adherence is being delivered twice (first day and third day of delivery) with changes to the data in-between.

These changes should be reflected in the calculation of the delivery adherence.

In datavault4dbt two macros are provided to create a **reference-table** and a **historized reference-table**: ref\_hub and ref\_sat\_v0.

They create a DV-structure that is quite similar to the standard hub and satellite with the difference of using natural keys instead of hash-keys

	1	HD_CATEGORY_DELIVERYADHERENCE_MISC_RS	LDTs	COUNT_DAYS_FROM	COUNT_DAYS_TO	NAME
1	(unknown)	00000000000000000000000000000000	0001-01-01 00:00:01.000000000	(unknown)	(unknown)	(unknown)
2	Fehler	04704691de48419ca96a1d8c2d872e4f	2022-03-12 08:00:00.000000000	xxx	xxx	Auftrag zu lange aktiv
3	deutlich zu früh	d345bced1646f57a446242b960747a7f	2022-03-19 08:00:00.000000000	-10	-4	bis zu 10 Tagen zu früh
4	deutlich zu spät	d74dc013071b428095fe96e610bb601e	2022-03-12 08:00:00.000000000	4	10	4 bis 10 Tage zu spät
5	deutlich zu spät	7a0aceb8aa50a547e0f01d205fbb756b	2022-03-19 08:00:00.000000000	4	10	bis zu 10 Tage zu spät
6	irrelevant	8f26ba5e80b2c0930859dda92f9f10f8	2022-03-12 08:00:00.000000000	zzz	zzz	Abverkauf, keine Lieferung
7	pünktlich	74a67a1514dc9901febddd44a2804af57	2022-03-12 08:00:00.000000000	0	1	pünktlich
8	pünktlich	802a8225b26c6d626164dce0c2dac8c2	2022-03-19 08:00:00.000000000	-1	1	pünktlich
9	viel zu früh	e54471f58309ca3d504d332fb1a77c08	2022-03-12 08:00:00.000000000	-1000000	-5	mehr als 5 Tage früher
10	viel zu früh	c352025d0819a1224561ffe7a0f80b6b	2022-03-19 08:00:00.000000000	-1000000	-10	mehr als 10 Tage zu früh

# Evaluation criterium 9: Changes of Attributes (A-B-A changes in customer data)

A very simple test case, the data in the customer (KundeID 107) is changed to a value in delivery 2 and get the values from delivery 1 again in delivery 3.

This type of change is handled correctly by the datavault4dbt-macro sat\_v0.

REPORTING_DATE	CUSTOMER_BK	EMAIL	GEBURTSDATUM	GUELTIGBIS	GESCHLECHT	KKFIRMA	KREDITKARTE	MOBIL
12. März 2022	107	waltraudthier@web.none	Montag, 31. Dezember 1951	12/10	w	American Express	0000 6425 0800 2000	0165/4543863
19. März 2022	107	walterthier@web.none	Montag, 31. Dezember 1951	12/10	m	American Express	0000 6425 0800 2000	0165/4543863
26. März 2022	107	waltraudthier@web.none	Montag, 31. Dezember 1951	12/10	w	American Express	0000 6425 0800 2000	0165/4543863

# Evaluation criterium 10: Deletions of Business Keys (Deletions in customer data)

---

In the willibald-data business-keys are being deleted and reappearing at a later date. (e.g., CustomerID '70' appearing in the webshop-data on 14. and 28. but not on 21.)

REPORTING_DATE	CUSTOMER_BK	NAME	VORNAME
14. März 2022	70	Göpfert	Jörn
28. März 2022	70	Göpfert	Jörn

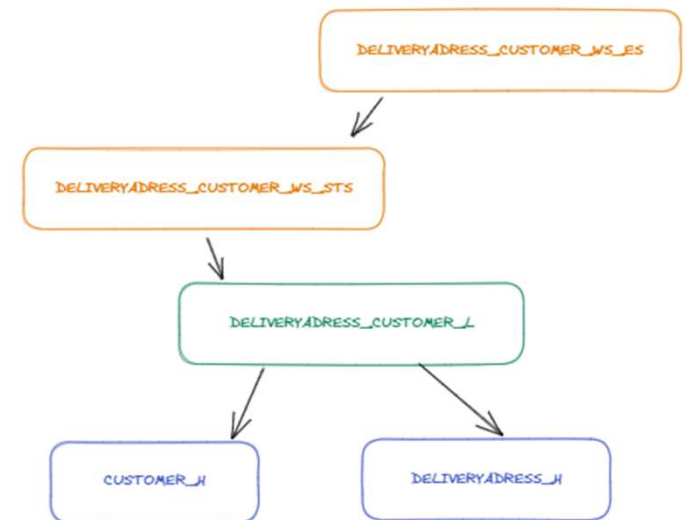
This issue has been solved by using status-tracking-satellites (\_sts) as explained in criterium #3.

# Evaluation criterium 11: Invalid foreign keys (Lieferadresse has unknown Kunde)

---

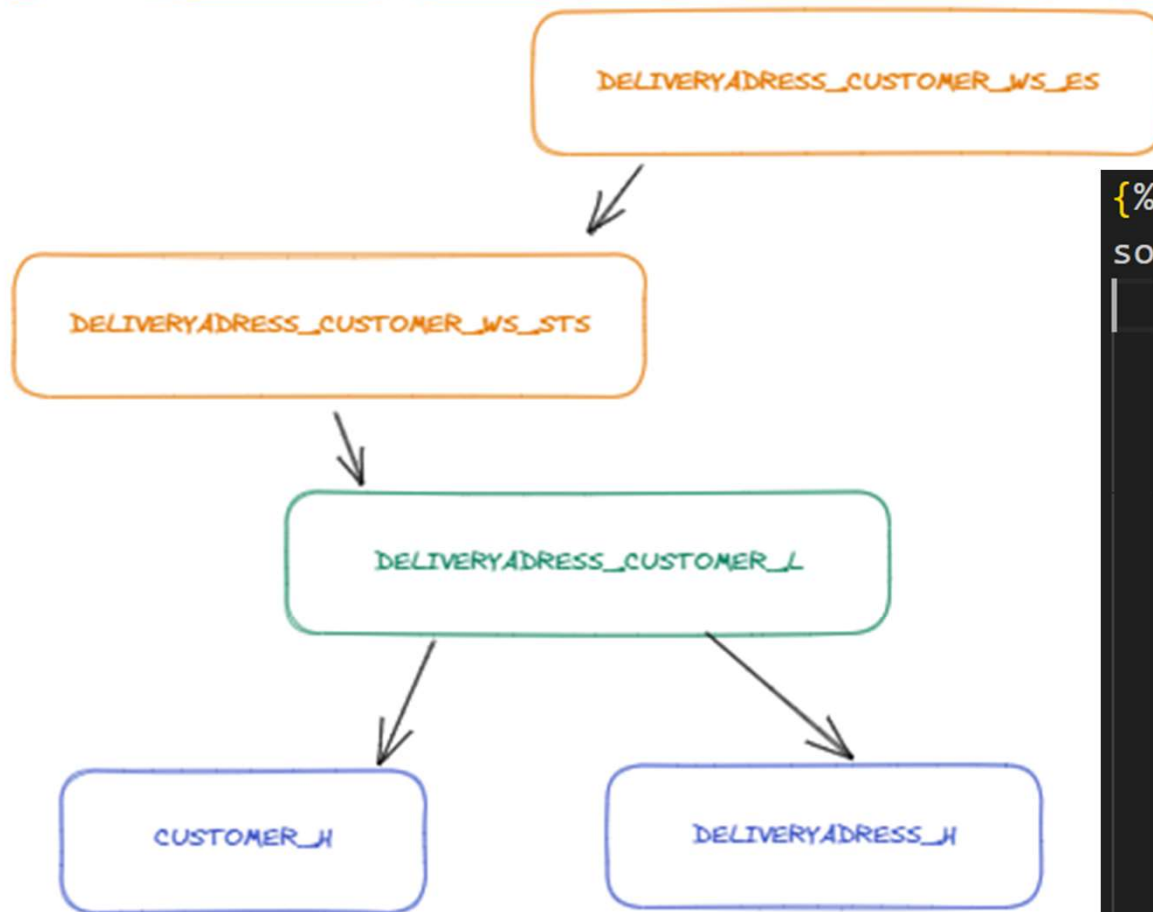
The first delivery contains delivery addresses for which there is no record with the same customer ID (KundeID) in CUSTOMER (Kunde - KundeIDs 999, 998 and 997).

As part of the standard Raw Vault implementation, one of the sources of CUSTOMER\_H is also the customer-column in delivery-address. That way the three relevant rows are correctly loaded into all the entities shown on the right.



# Evaluation criterium 11: Invalid foreign keys (Lieferadresse has unknown Kunde)

---



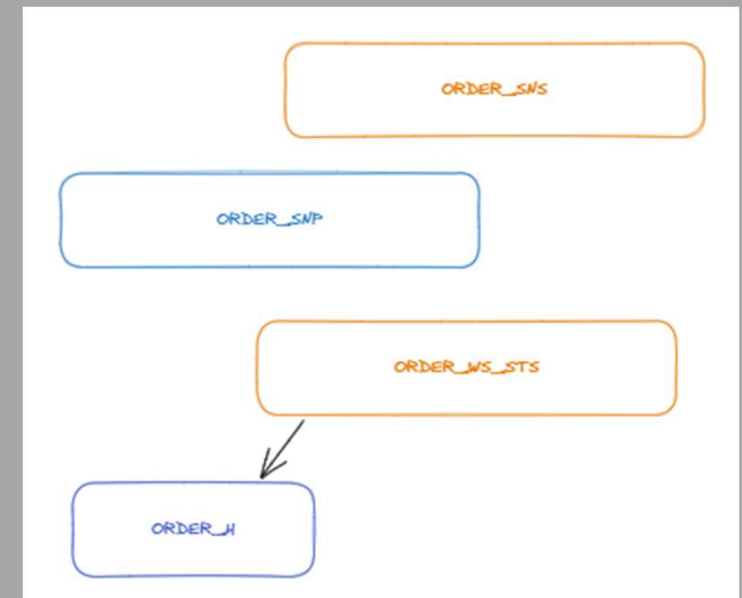
```
{%- set yaml_metadata -%}  
source_models:  
  - name: stg_webshop_bestellung  
    hk_column: hk_customer_h  
    bk_columns: 'customer_bk'  
    rsrc_static: '*/webshop/bestellung/*'  
  - name: stg_webshop_kunde  
    hk_column: hk_customer_h  
    bk_columns: 'customer_bk'  
    rsrc_static: '*/webshop/kunde/*'  
  - name: stg_webshop_lieferadresse  
    hk_column: hk_customer_h  
    bk_columns: 'customer_bk'  
    rsrc_static: '*/webshop/lieferadresse/*'
```

# Evaluation criterium 12: Deletion of Orders

The orders are relevant for counting and are deleted during the dates of deliveries.  
Between period 1 and 2 the orderIDs (BestellungID) 99, 220 and 465.  
Between periods 2 and 3 the orderIDs (BestellungID) 1470 and 1288.

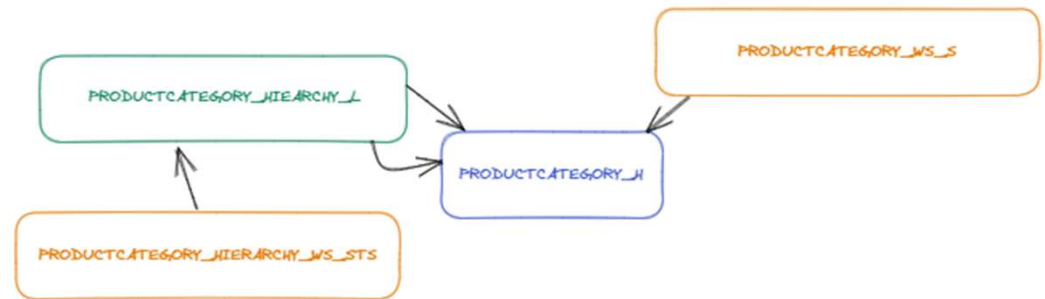
Within the status-satellite ORDER\_WS\_STS, a new record with cdc=D is added for orderID=99, when loading period 2.  
This leads to this order not being part of the „snapshot-satellite“ order\_sns, for each reporting\_date after period 2.

REPORTING_DATE	ORDER_BK
14. März 2022	220
14. März 2022	465
14. März 2022	99
21. März 2022	1288
21. März 2022	1470



# Evaluation criterium 13: Change of Dimensions

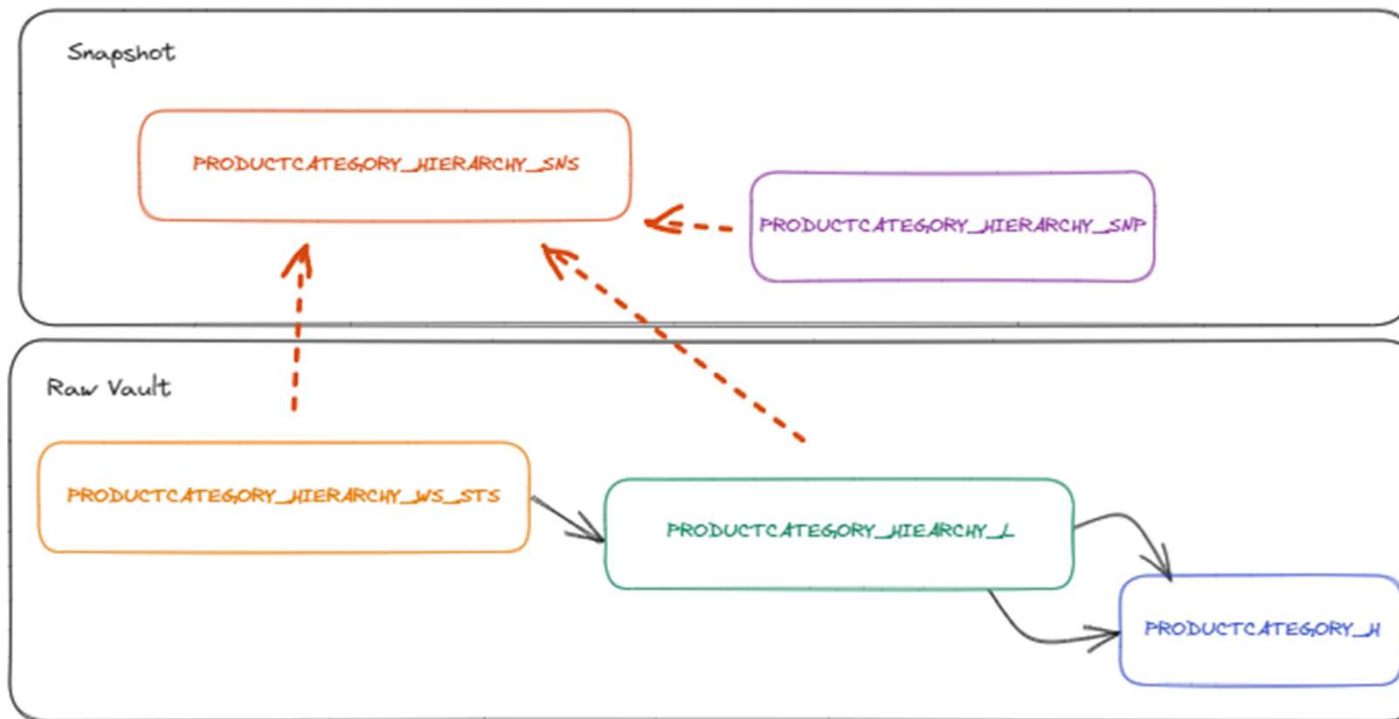
The productcategory-hub is sourced from the columns KatID and ObereKatID from the same source-table. The link defines the hierarchy (hierarchical-link). This hierarchy completely changes from period 1 to period 2.



REPORTING_DATE	PRODUCTCATEGORY_L3	PRODUCTCATEGORY_L2	PRODUCTCATEGORY_L1	AMOUNT
12. März 2022	Amaranth	Mittelzehrer	Gemüse	685,00
19. März 2022	Amaranth	Gemüse	Mittelzehrer	1.306,00
26. März 2022	Amaranth	Mittelzehrer	Gemüse	1.348,00
Gesamt				3.339,00

# Data Warehouse – Snapshot-Layer

---



It consists of Snapshot-PIT for every hub and link and Snapshot-views on top joining the pit with all its attached satellites including status and effectivity satellites

This is the base for Business Vault

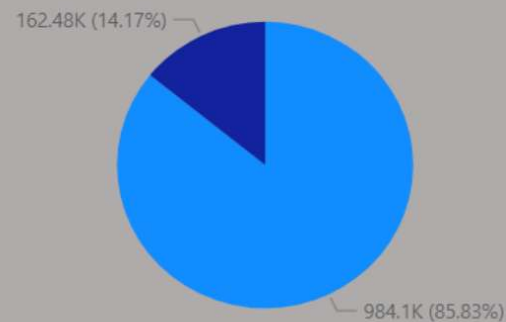


# Evaluation criterium 14: Key Figures

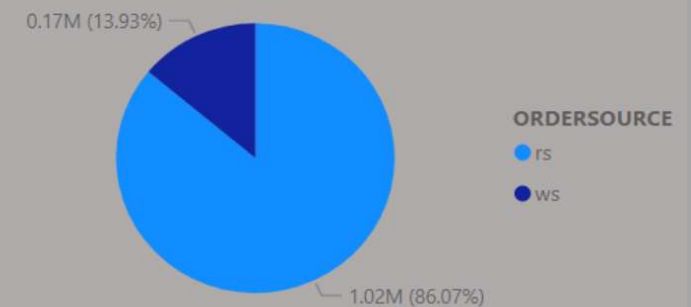
We created a PowerBI Report to give access to the data:

REPORTING_DATE	ORDERSOURCE	GROSS_PROFIT	REVENUE
3/7/2022 11:59:59 PM	rs	114,542.05	109,913.20
3/14/2022 11:59:59 PM	rs	229,069.70	219,821.54
3/14/2022 11:59:59 PM	ws	32,722.25	32,053.58
3/21/2022 11:59:59 PM	rs	340,511.65	327,182.84
3/21/2022 11:59:59 PM	ws	61,783.35	60,514.53
3/28/2022 11:59:59 PM	rs	340,511.65	327,182.84
3/28/2022 11:59:59 PM	ws	71,384.55	69,909.71

REVENUE by ORDERSOURCE



GROSS\_PROFIT by ORDERSOURCE



# Evaluation criterium 15: Business rules implementation

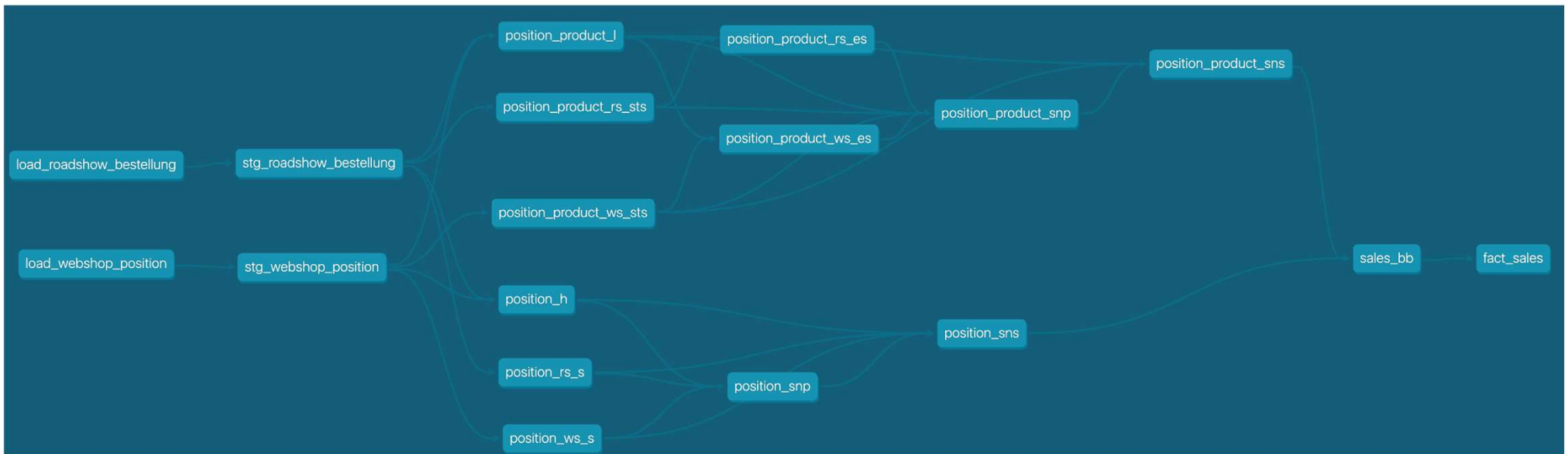
---

- `order_customer_bb`:  
Adds records for roadshow orders, where customers are missing in the order-table, but can be identified using the creditcard-information to the already existing records (in the Raw Vault stored in `order_customer_l`)
- `sales_bb`:  
Contains all the logic necessary for the fact-table
- a `*_bs` (business satellite) view is specified for each dimension, if any business-logic needs to be applied

# Evaluation criterium 16: Data Lineage

dbt provides data lineage on table-level based on the references within the code

```
from {{ ref("customer_sns") }} s
```



Columnar lineage is not available.

# Evaluation criterium 17: Error Handling

## Implemented in macro for load-view

**Step 1:** Get attributes with names as VARCHAR

**Step 2:** try\_to-Cast to defined datatype

**Step 3:** If result of cast is NULL and RAW-value was NOT NULL...an error has occurred: the type-check failed

**Step 4:** All type-checks, duplicate-checks and empty-key-checks are connected to one column: **is\_check\_ok**

```
1 with
2 raw_data AS
3 (
4     SELECT
5         TRIM(replace(right(filenamedate,19),'.csv',''))::STRING as ldts_raw
6         , TRIM(filenamedate::STRING) as rsrc_raw
7         , TRIM(trim(reverse(substring(reverse(replace(filenamedate,'.csv','')), 17,8))::varchar)::STRING) as edts_in_raw
8         , TRIM(value::STRING) as raw_data_raw
9         , TRIM(metadata$file_row_number::STRING) as row_number_raw
10        , TRIM(value:c1::STRING) as bestellungid_raw
11        , REPLACE(TRIM(value:c4::STRING) , ',' , ' ') as spezlieferadrid_raw
12        , TRIM(value:c6::STRING) as preis_raw
13    FROM DWS.DWH_01_EXT.EXT_WEBSHOP_POSITION
14 )
15 SELECT
16     TRY_TO_TIMESTAMP(ldts_raw, 'YYYYMMDD_HH24MISS') as ldts
17     , rsrc_raw as rsrc
18     , TRY_TO_DATE(edts_in_raw, 'YYYYMMDD') as edts_in
19     , raw_data_raw as raw_data
20     , TRY_TO_NUMBER(row_number_raw) as row_number
21     , bestellungid_raw as bestellungid
22     , TRY_TO_NUMBER(spezlieferadrid_raw) as spezlieferadrid
23     , TRY_TO_TIMESTAMP(ldts_raw, 'YYYYMMDD_HH24MISS') IS NOT NULL OR ldts_raw IS NULL as is_ldts_type_ok
24     , TRY_TO_NUMBER(row_number_raw) IS NOT NULL OR row_number_raw IS NULL as is_row_number_type_ok
25     , TRY_TO_NUMBER(spezlieferadrid_raw) IS NOT NULL OR spezlieferadrid_raw IS NULL as is_spezlieferadrid_type_ok
26     , ROW_NUMBER() OVER (PARTITION BY ldts,bestellungid,PosID ORDER BY ldts,bestellungid,PosID) = 1 AS is_dub_check_ok
27     , COALESCE(bestellungid_raw, '') <> '' as is_bestellungid_key_check_ok
28     , is_ldts_type_ok AND is_edts_in_type_ok AND is_row_number_type_ok AND [...] is_check_ok
29     , TO_VARIANT(ARRAY_EXCEPT((REPLACE(IFF(NOT is_ldts_type_ok,'ldts:' || COALESCE(TO_VARCHAR(ldts_raw), '') || ',' , '')) || [...] ) , [...] ) ) ) chk_all_msg
30 FROM raw_data
```

## Evaluation criterium 17: Error Handling

---

```
, REPLACE(TRIM(value:c10::STRING) , ',', '.') as preis_raw
```

```
, TRY_TO_NUMBER(preis_raw, 20,8) as preis
```

```
, TRY_TO_NUMBER(preis_raw, 20,8) IS NOT NULL OR preis_raw IS NULL as is_preis_type_ok
```

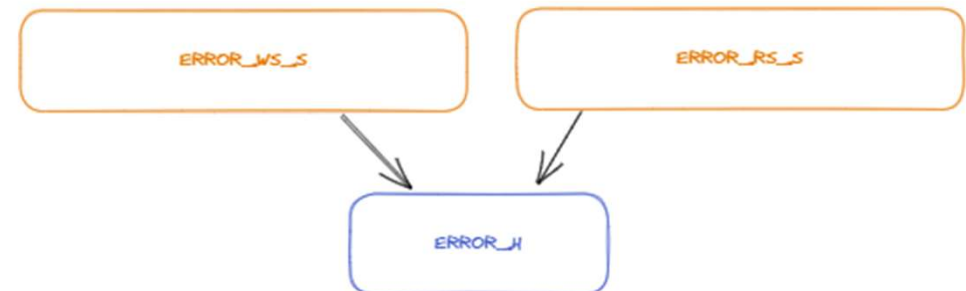
```
, is_ldts_type_ok  
AND [...] is_preis_type_ok  
AND is_rabatt_type_ok [...]  
AND is_produktid_key_check_ok is_check_ok
```

# Evaluation criterium 17 (cont): Error Handling

Based on the column `is_check_ok`, erroneous data can easily be excluded from the further standard process and loaded into the error-vault.

We defined one Error-Satellite for each source-system

- one column with all the RAW\_DATA as json
- one column with a summary of all the failed checks



ABC HK_ERR	ABC HD_ERROR_S	ABC RSRC	LDTS	ABC RAW_DATA	ABC CHK_ALL_MSG
ccfab07212b8d9	473573925348e17434f9029bf43b15bf	ldts/webshop/produkt/p	2022-03-12 08:00:00.000	{"c1":"21","c2":"GMiCHILI ","c3":"Chili „Vietnam -	[{"dub_check":"ldts,produktid"}]
1024342fa04781	d6b87cdd56e915602e19f87144caa336	ldts/webshop/produkt/p	2022-03-12 08:00:00.000	{"c1":"20","c2":"GMiCHILI ","c3":"Chili, Jalapeno „I	[{"dub_check":"ldts,produktid"}]
48cdf3c66440f	6f6417f0bf8fc24d6a7cdd0058c3e68	ldts/webshop/lieferdiens	2022-03-19 08:00:00.000	{"c10":"Niedersachsen","c2":"Schalalala OHG","c3	[{"dub_check":"ldts,LieferDienstID"},"key_check":"LieferDienstID"}]
d68849e38209c	ffe8dbc1c84e3bde8166b67dae9e6dd	ldts/webshop/lieferdiens	2022-03-19 08:00:00.000	{"c2":"Femen AG","c3":"06257/279376925","c4":"N	[{"key_check":"LieferDienstID"}]

# Evaluation criterium 18: Orchestration

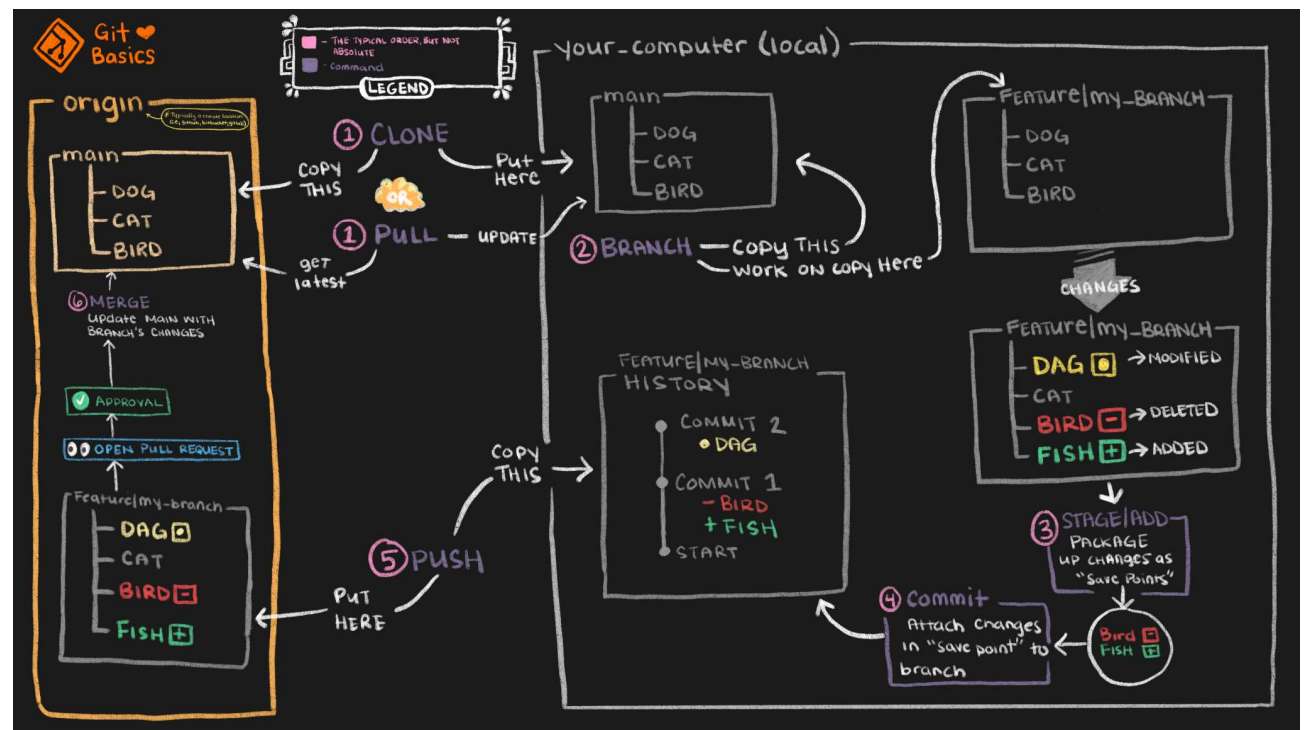
---

- dbt uses lineage-information to calculate the sequence to build all dependent objects. There is no need to create specific loading-chains.
- To refresh the whole model only the command `dbt build` has to be issued
- If only one source has new data it could be useful to refresh only this source, and all dependend models by issuing the command `dbt build -s <modelname>+`
- The dbt build - command will only add new data. If all data (existing in the landing zone) should be loaded, the parameter `--full-refresh` can be added
- To make the loading from the landing-zone more efficient, we added a high-watermark-table to the model. In there the latest load-date of every source is saved and only newer data is read.

# Evaluation criterium 19: Deployment

All of the programming-artefacts in dbt are plain text files dbt supports github, gitlab, bitbucket, Azure DevOps etc.

deployment follows the git workflow



<https://www.getdbt.com/analytics-engineering/transformation/git-workflow/>

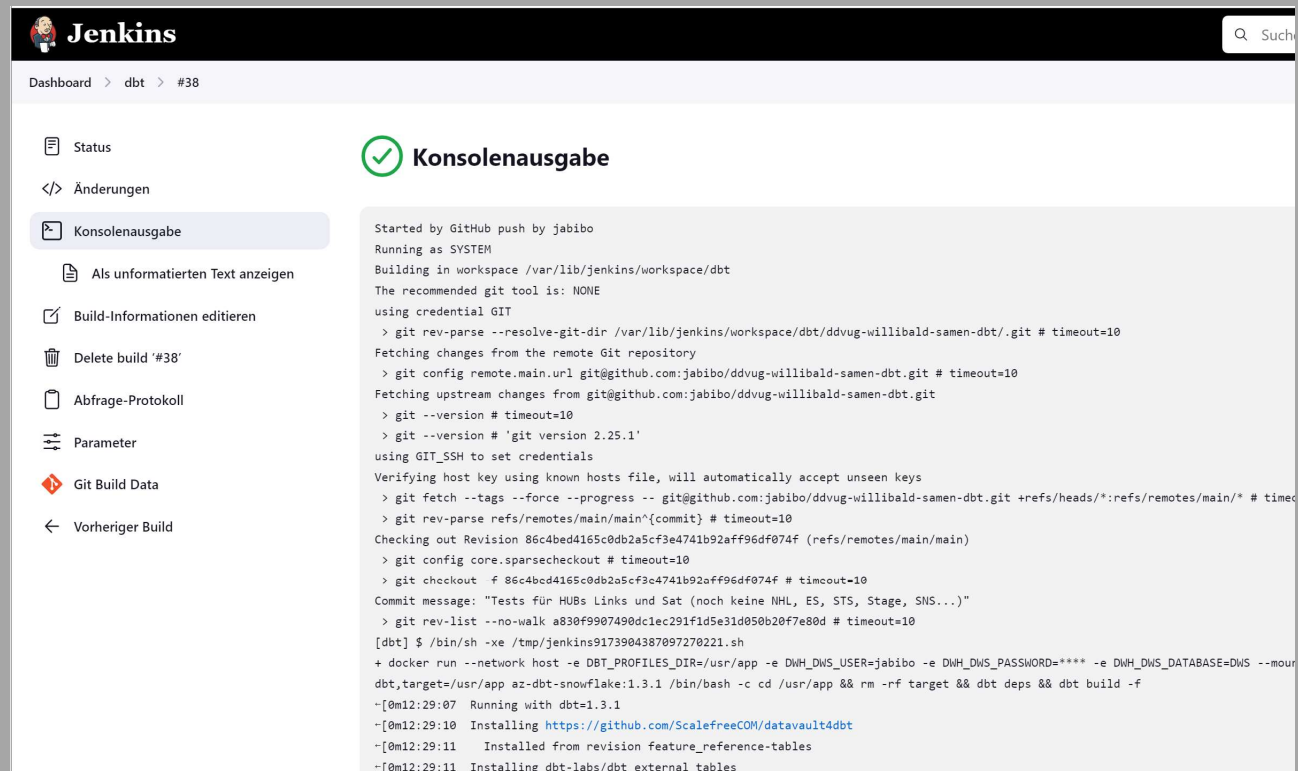


# Evaluation criterium 20: Scheduling

In our project we use Jenkins to run dbt in a docker container.

As the Willibald-data is provided by a github-repo. We also use Jenkins to get the data and store it in the S3-Bucket.

dbt doesn't need a lot of resources because most of the work is done by the target-database-server.



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo and a search bar. Below the navigation bar, the breadcrumb trail reads 'Dashboard > dbt > #38'. The left sidebar contains a list of actions: 'Status', 'Änderungen', 'Konsolenausgabe' (selected), 'Als unformatierten Text anzeigen', 'Build-Informationen editieren', 'Delete build '#38'', 'Abfrage-Protokoll', 'Parameter', 'Git Build Data', and 'Vorheriger Build'. The main content area is titled 'Konsolenausgabe' with a green checkmark icon. It displays the console output of a build, which starts with 'Started by GitHub push by jabibo' and 'Running as SYSTEM'. The output shows the build directory is '/var/lib/jenkins/workspace/dbt' and the recommended git tool is 'NONE'. It then shows the git command 'git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/dbt/ddvug-willibald-samen-dbt/.git # timeout=10' and the output 'Fetching changes from the remote Git repository'. The build continues with 'git config remote.main.url git@github.com:jabibo/ddvug-willibald-samen-dbt.git # timeout=10' and 'Fetching upstream changes from git@github.com:jabibo/ddvug-willibald-samen-dbt.git'. The build then shows 'git --version # timeout=10' and 'git --version # 'git version 2.25.1''. The build then shows 'using GIT\_SSH to set credentials' and 'Verifying host key using known hosts file, will automatically accept unseen keys'. The build then shows 'git fetch --tags --force --progress -- git@github.com:jabibo/ddvug-willibald-samen-dbt.git +refs/heads/\*:refs/remotes/main/\* # timeout=10' and 'git rev-parse refs/remotes/main/main^{commit} # timeout=10'. The build then shows 'Checking out Revision 86c4bed4165c0db2a5cf3e4741b92aff96df074f (refs/remotes/main/main)' and 'git config core.sparsecheckout # timeout=10'. The build then shows 'git checkout f 86c4bed4165c0db2a5cf3e4741b92aff96df074f # timeout=10' and 'Commit message: "Tests für HUBs Links und Sat (noch keine NHL, ES, STS, Stage, SNS...)"'. The build then shows 'git rev-list --no-walk a830f9907490dc1ec291f1d5e31d050b20f7e80d # timeout=10' and '[dbt] \$ /bin/sh -xe /tmp/jenkins9173904387097270221.sh'. The build then shows '+ docker run --network host -e DBT\_PROFILES\_DIR=/usr/app -e DWH\_DWS\_USER=jabibo -e DWH\_DWS\_PASSWORD=\*\*\*\* -e DWH\_DWS\_DATABASE=DWS --mount dbt,target=/usr/app az-dbt-snowflake:1.3.1 /bin/bash -c cd /usr/app && rm -rf target && dbt deps && dbt build -f'. The build then shows '-[0m12:29:07 Running with dbt=1.3.1' and '-[0m12:29:10 Installing https://github.com/ScalefreeCOM/datavault4dbt'. The build then shows '-[0m12:29:11 Installed from revision feature\_reference-tables' and '-[0m12:29:11 Installing dbt-labs/dbt\_external\_tables'.

# Evaluation criterium 21: Supported Databases (as of April 2023)

\*<https://docs.getdbt.com/docs/supported-data-platforms>

(there are even more created from the community)

\*\*<https://github.com/ScalefreeCOM/datavault4dbt>

(The work-in-progress (wip) should be available sometimes this year)

target	dbt*	datavault4dbt**
AlloyDB	x	
Azure Synapse	x	(wip)
SQL-Server	(community)	(wip)
BigQuery	x	x
Databricks	x	(wip)
Dremio	x	
Postgres	x	(wip)
Exasol	(community)	x
Redshift	x	(wip)
Snowflake	x	x
Spark	x	
Starburst & Trino	x	
Greenplum	(community)	(wip)

# Evaluation criterium 22:

## Installation Requirements

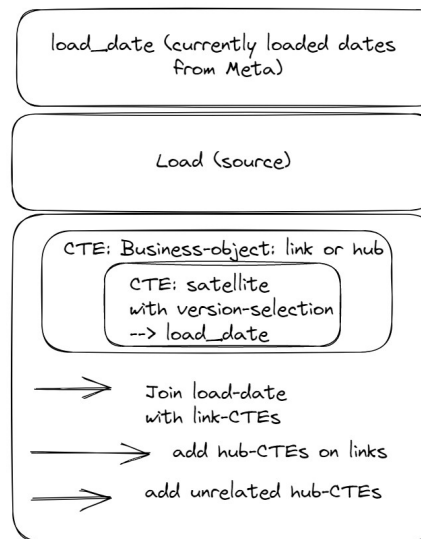
Prerequisites for installing dbt are python (3.7+) and a git-client.

We also installed Visual Code on our windows machines to be able to edit the text-files more comfortably.

All other installations are then done using pip or dbt (except for jenkins and docker).

# Evaluation criterium supplemental 1: Yedi-Tests

Yedi-tests can be created using the  
yedi\_test-macro located under  
macros/tables.



```
{{ config(enabled=True) }}
{% set yaml_metadata -%}
source_model_source: 'load_roadshow_bestellung'
load_type: partial
source_model_target:
| order_h: Jan Binge, last week + ...
  business_object:
    - order: bestellungid
  satellites:
    order_rs_rts:
position_h:
  business_object:
    - position: bestellungid
    - position: produktid
  satellites:
    position_rs_s:
      columns:
        - bestellungid
        - gueltigbis
        - kreditkarte
        - rabatt
    position_rs_sts:
order_associationpartner_1:
  business_object:
    - order: hk_order_h
    - associationpartner: hk_associationpartner_h
  satellites:
    order_associationpartner_rs_sts:
{% endset -%}

{% set metadata_dict = fromyaml(yaml_metadata) %}

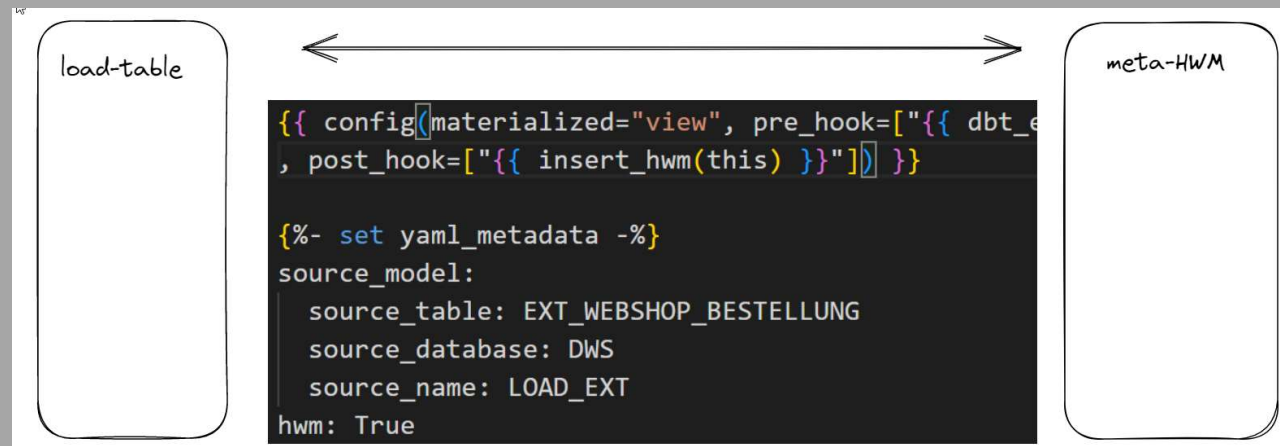
{{ yedi_test(source_model_source=metadata_dict['source_model_source'],
            source_model_target=metadata_dict['source_model_target'],
            load_type=metadata_dict['load_type']) }}
```

Due to some weird behaviour from the snowflake-optimizer not all Yedi-Tests may run. If you experience long running SQLs (> 10 secs) it could be the case, that the execution plan has not been built optimal. Please reach out to snowflake support and refer to case-no: 00511229 – snowflake will then change something in the configuration of the account.

# Evaluation criterium supplemental 2: High water marking

We created a high-water marking – mechanism to make the access to the data lake more efficient

- There is hwm-switch and a "post-hook" in every loading-model.
- If the hwm-switch is set to true it will (in incremental-load) only read from the last "High-Water-Mark" (highest Idts) which is been set by the post-hook the last time data has been inserted
- In full-load the "High-Water-Mark" will be ignored at reading – but it will be set after loading.
- The hwm-table is located in the DWH\_00\_META-schema
- There is also a hwm-mechanism integrated in datavault4dbt – which we didn't use.



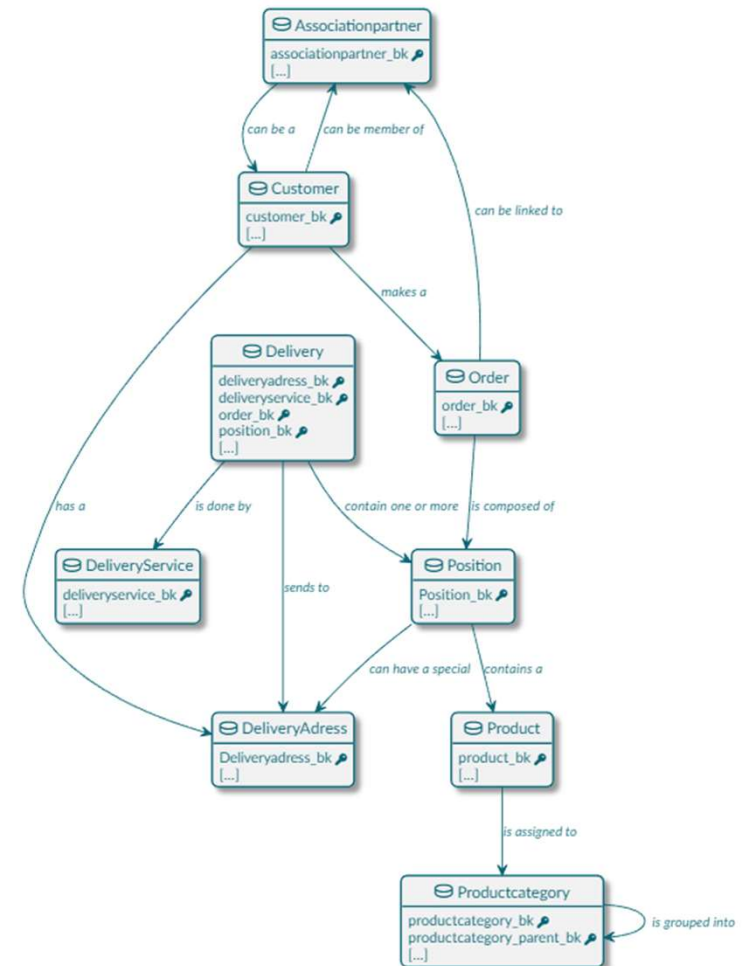
# Intermediate result

---

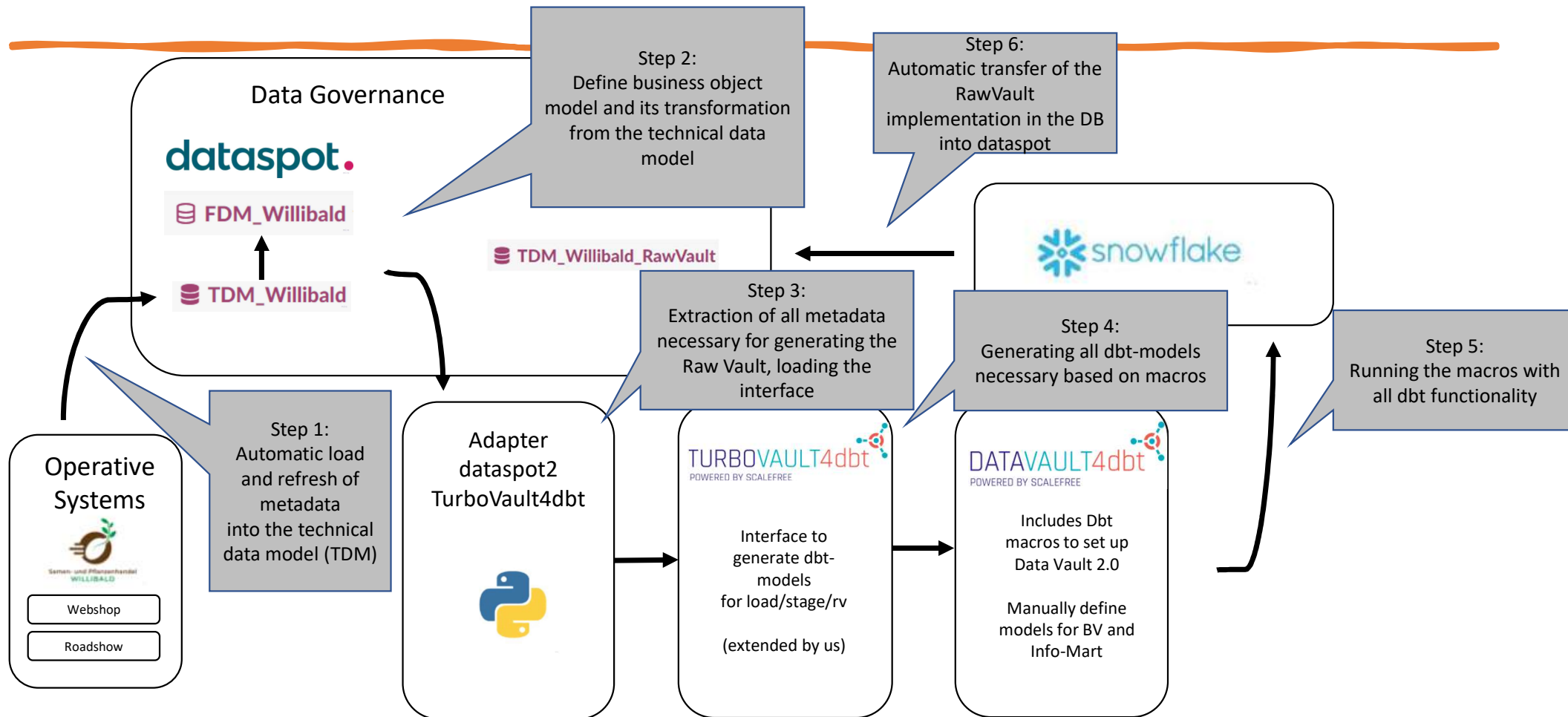
- Fully data vault 2.0 compliant data warehouse **solving all the issues** defined in the challenge
- dbt follows a **technical approach**, allowing a professional developing process – but there is a **risk** of creating a gap between the business and the data vault implementation.
- A data warehouse setup should promote close **cooperation** and transparency between the business departments and the data engineers.
- To address this issue, we added a **graphical data governance tool** to the toolstack and integrated it into our technical solution.



- #1 Data Governance Tool in BARC's The Data Management Survey 23
- based on Data Excellence Framework©
- easily configurable (additional metadata)
- dataspot. is built by a very passionate, collaborative team, it is fun to work with

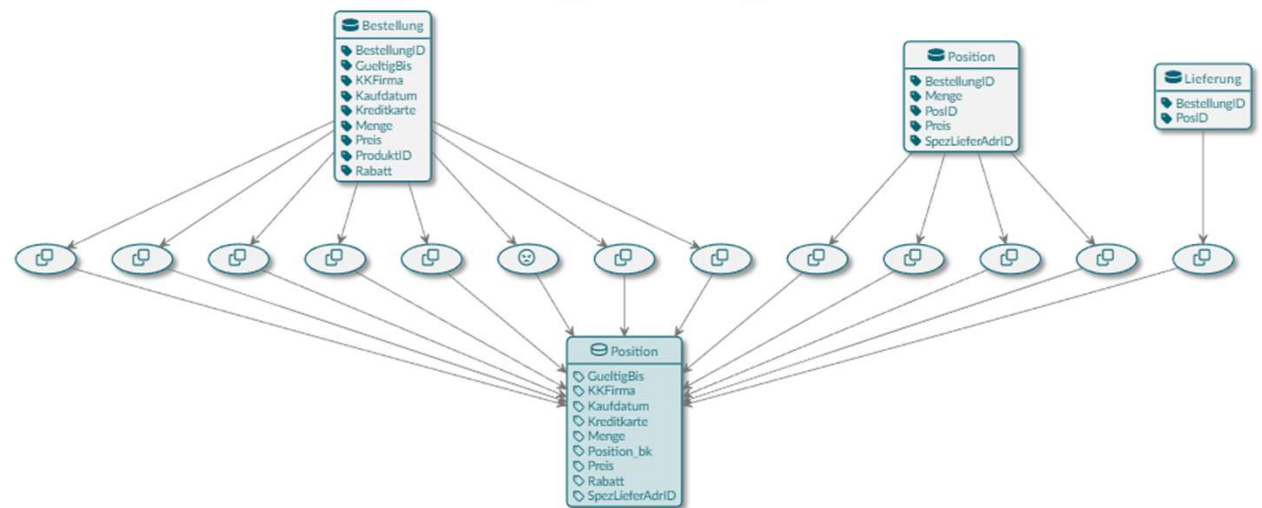
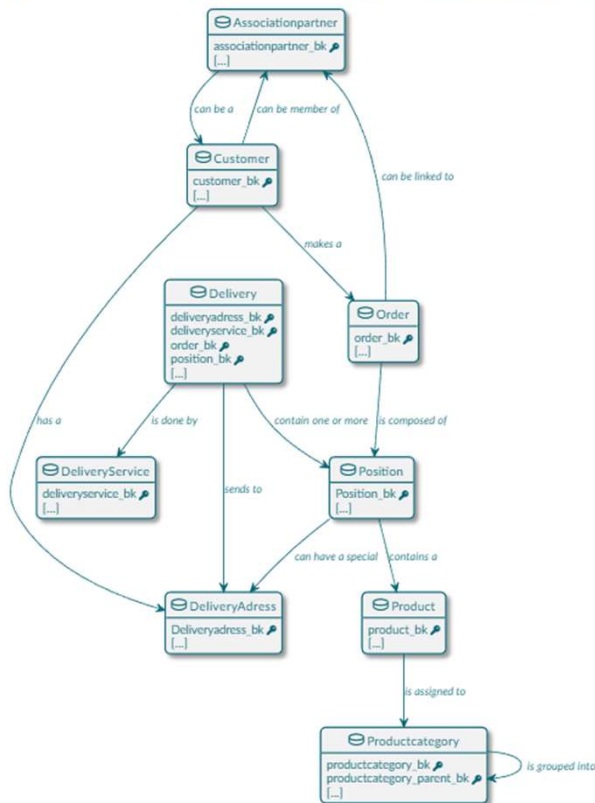


# Our technical setup – metadata flow

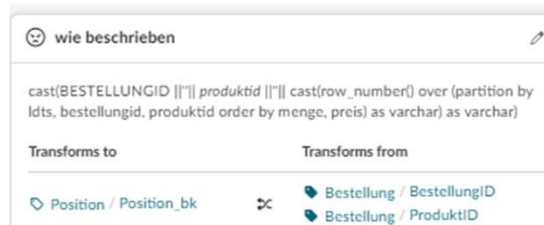




# Business Object Model (FDM\_Willibald)

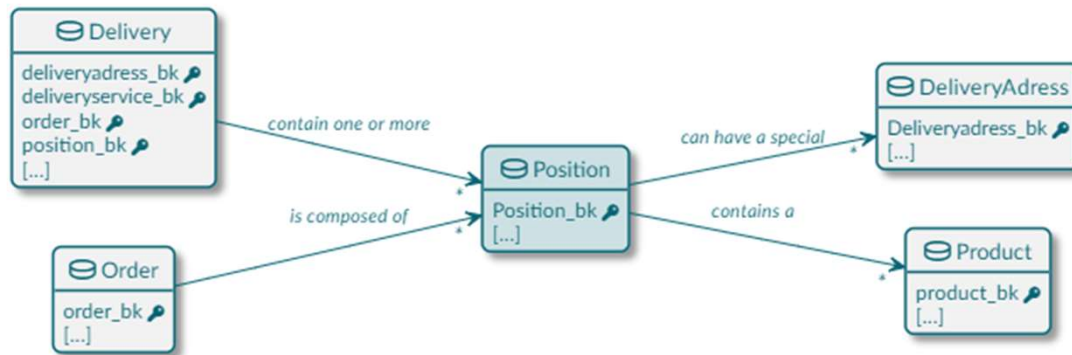


Transformation from technical data model into Business Object Model

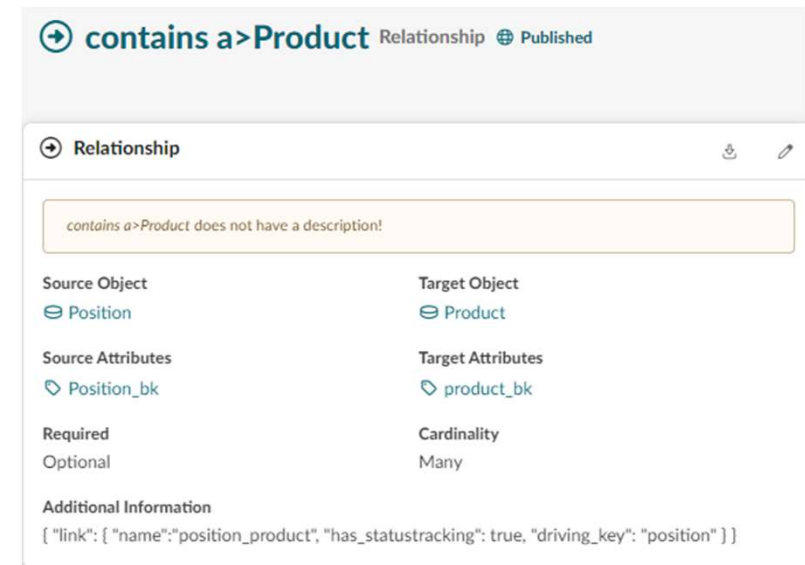


More complex transformation

# Business Object Model (FDM\_Willibald)



Relationships within the business object model are set up as links in the raw vault

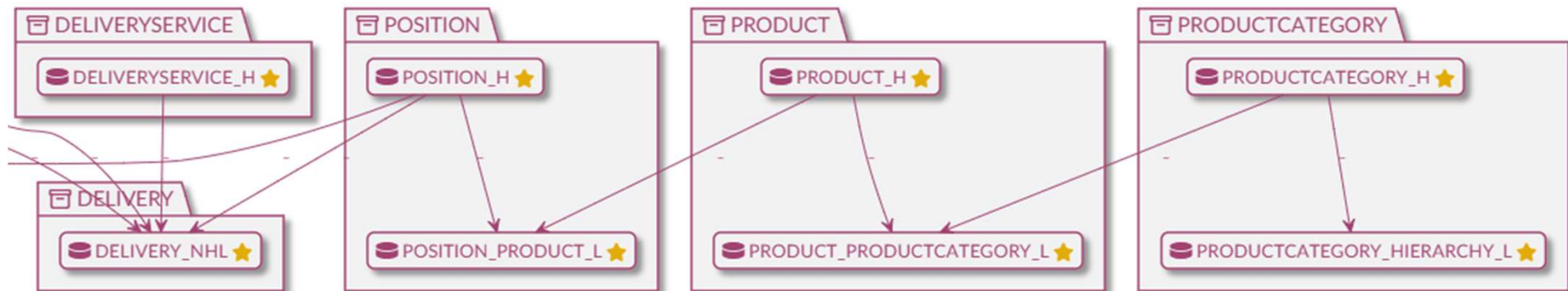


All necessary information is defined within the model

# Raw Vault in dataspot.

(TDM\_Willibald\_RawVault)

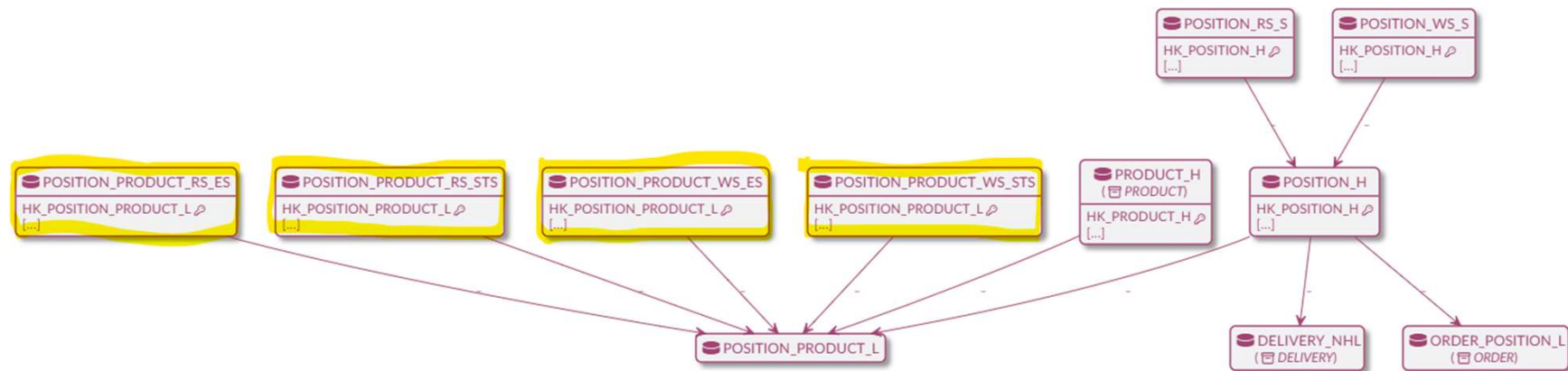
---



Each Business-Object and its related models is defined in its own collection, only showing the hubs and links (the spine) in the overview diagram.

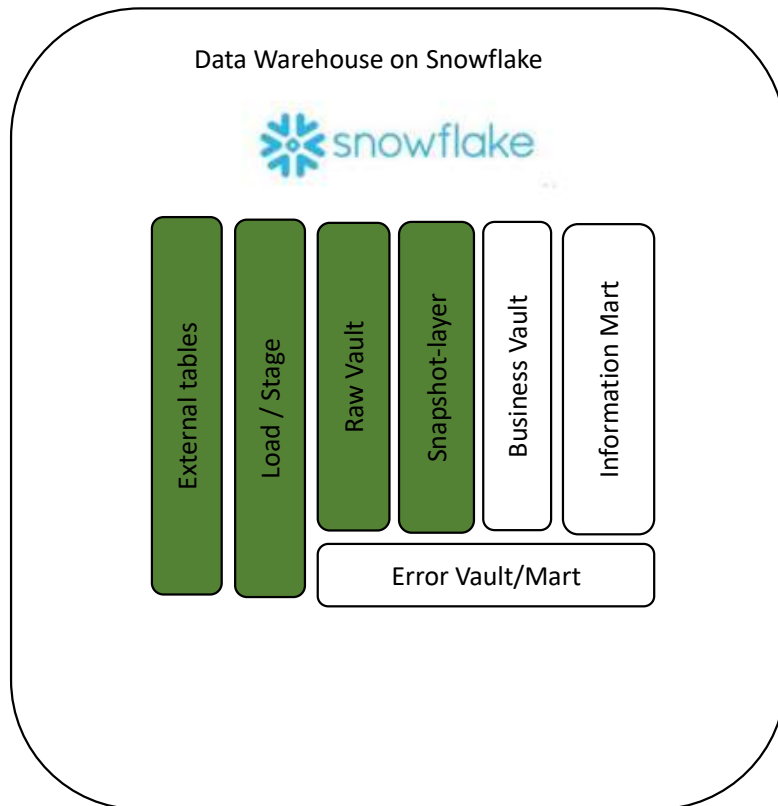
# Raw Vault in dataspot.

(TDM\_Willibald\_RawVault)



When extending Position, all entities within the collection (and entities connected to it) are shown. Based on the additional information defined in the relationship "position contains a product", the marked entities and all its necessary load and stage entities are automatically generated.

# Models automatically generated



95 % of all models and tests in the marked layers are automatically generated based on the metadata in dataspot.

All generated models are marked with a comment like:

```
{# template link Version:0.1.0 #}  
{# automatically generated based on dataspot#}
```

# Summary

---

- dbt and datavault4dbt can be used to implement a fully data vault 2.0 compliant data warehouse **solving all the issues** defined in the challenge
- By integrating the solution with a business-centric data governance tool like **dataspot** we were able to reach a **new level of maturity** concerning the automation process, forcing all stakeholders to closely work together, achieving a common understanding by design :-).
- This setup is heading towards our idea of an **ideal data warehouse solution**
- If you want to contact us:
  - [andreas@haas-erlangen.com](mailto:andreas@haas-erlangen.com)
  - [jan@binge.de](mailto:jan@binge.de)

# Contribution

---

