Data Analytics

Data Lake

Data Factory

Lakehouse

Data Mesh

BURYING DATA WAREHOUSE - RIP?

https://www.linkedin.com/pulse/buying-data-warehouse-rip-bill-inmon/?trackingId=W3T2j93RSHGWmFE280r3IQ%3D%3D

Data Analytics

Data Lake

Data Factory

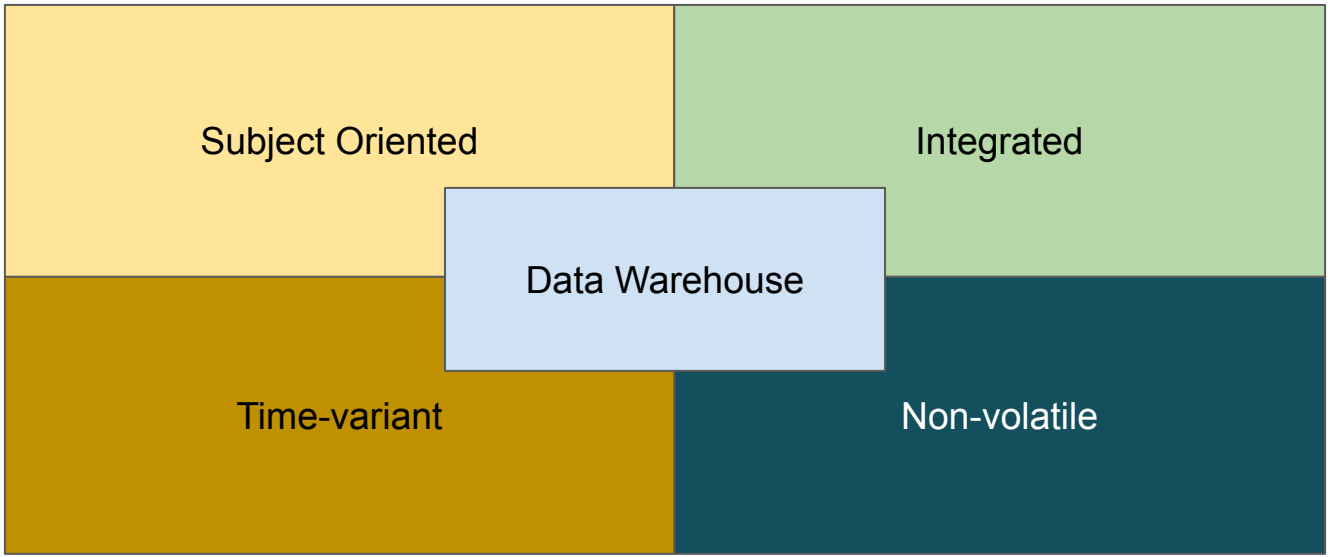
Lakehouse

Data Mesh

BURYING DATA WAREHOUSE?
Resilient Information Processing

https://www.linkedin.com/pulse/buying-data-warehouse-rip-bill-inmon/?trackingId=W3T2j93RSHGWmFE280r3IQ%3D%3D

Data Warehouse Requirements - Technology agnostic



“The primary purpose of a data warehouse is to transform data from an application state into into an integrated corporate state”
Bill Inom, the father of data warehousing

“A data warehouse is a copy of transaction data specifically structured for query and analysis.”
Ralph Kimball, Dimensional Modeling

Inspired by https://youtu.be/HnFHUdFv9AU

A Persistent Staging …

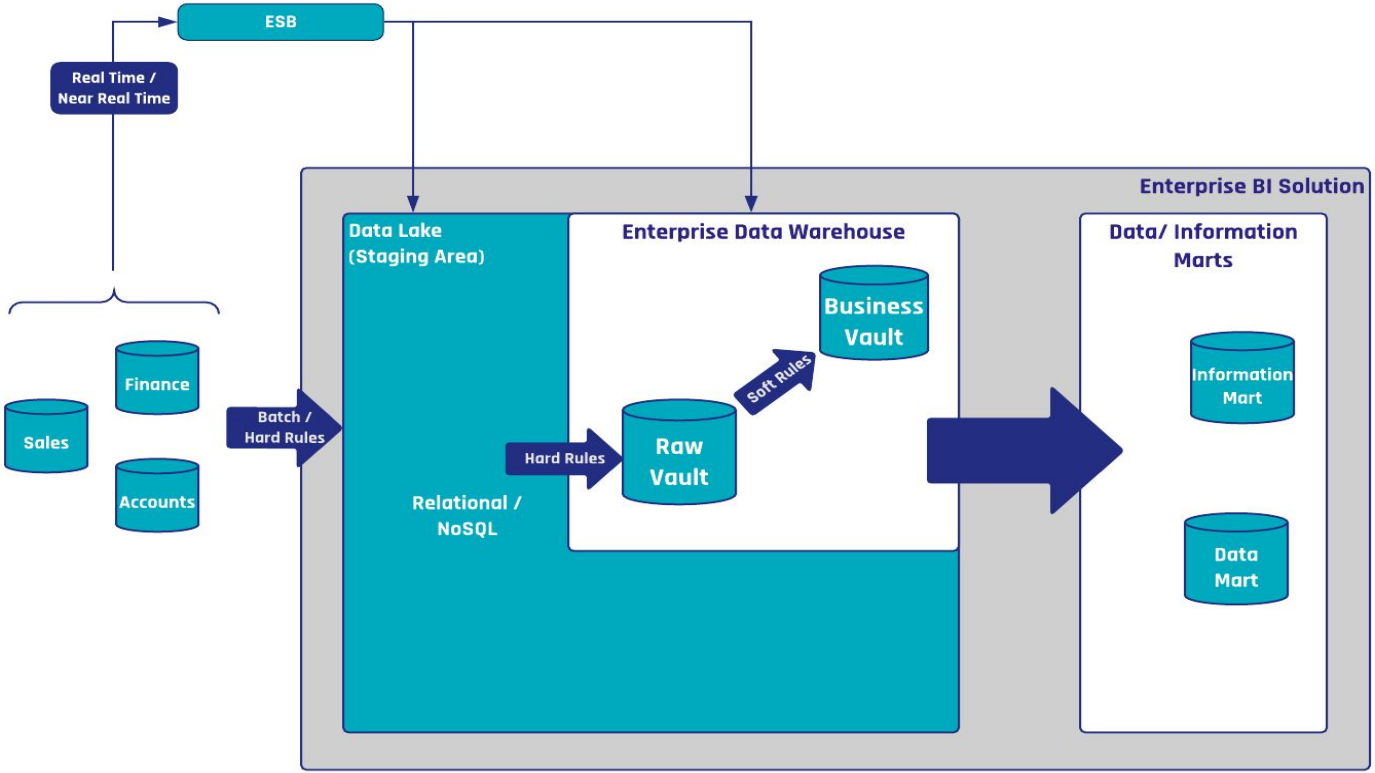
- Staging Area that is not wiped out after each load
- Keeps full history of all data deltas from all source deliveries to your warehouse

Benefits

- Reloading downstream layer from PSA (Datavault)
- Faster build time than Data Vault
  - PSA is a “Push of a button” automation
  - Data Vault needs understanding of the business model
- Source data can be available for other use cases even before you know the business value
- PSA is implemented with changed rows only

Ist die PSA ein fester Bestandteil der Datavault Datenarchitektur?

Data Lake as PSA https://www.scalefree.com/scalefree-newsletter/efficient-data-lake-structure/



Relational versus Data Lake

Persistent Stage in DBMS

Schema on **Write**
Structured Schema kept up to date with ALTER statements

Costly Storage
Database can be massive and with indexes the cost is significant

Structured Reload data
Easy integration to reload the next layer in the Data Warehouse

Data Lake - Files

Schema on **Read**
Rapid landing of data but requires extensive tagging to ensure that it is usable across the enterprise

Unlimited Storage
Inexpensive storage unlimited data and easily scalable

Schema applied reload
Reload needs to accommodate schema changes over time

Inspired by https://youtu.be/NzfBAutN80s

PSA Considerations for Azure Data Lake

Persistent Stage in DBMS

Schema on **Write**
Structured Schema kept up to date with ALTER statements

Modern DBMS can do schema on Read using JSON

Costly Storage
Database can be massive and with indexes the cost is signifi

Separation of Compute and Storage in modern DBMS can help

Structured Reload data
Easy integration to reload the next layer in the Data Warehouse

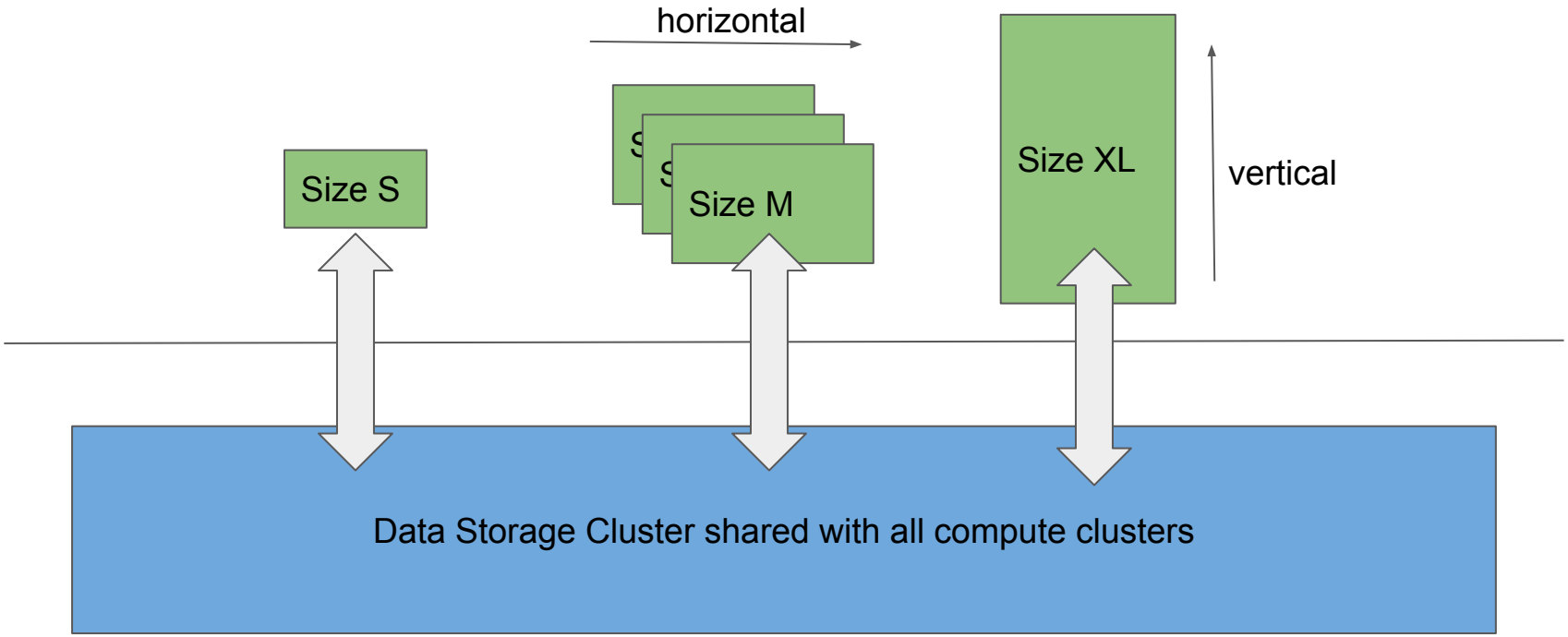
Data Lake - Files

Schema on **Read**
Rapid landing of data but requires extensive tagging to ensure that it is usable across the enterprise

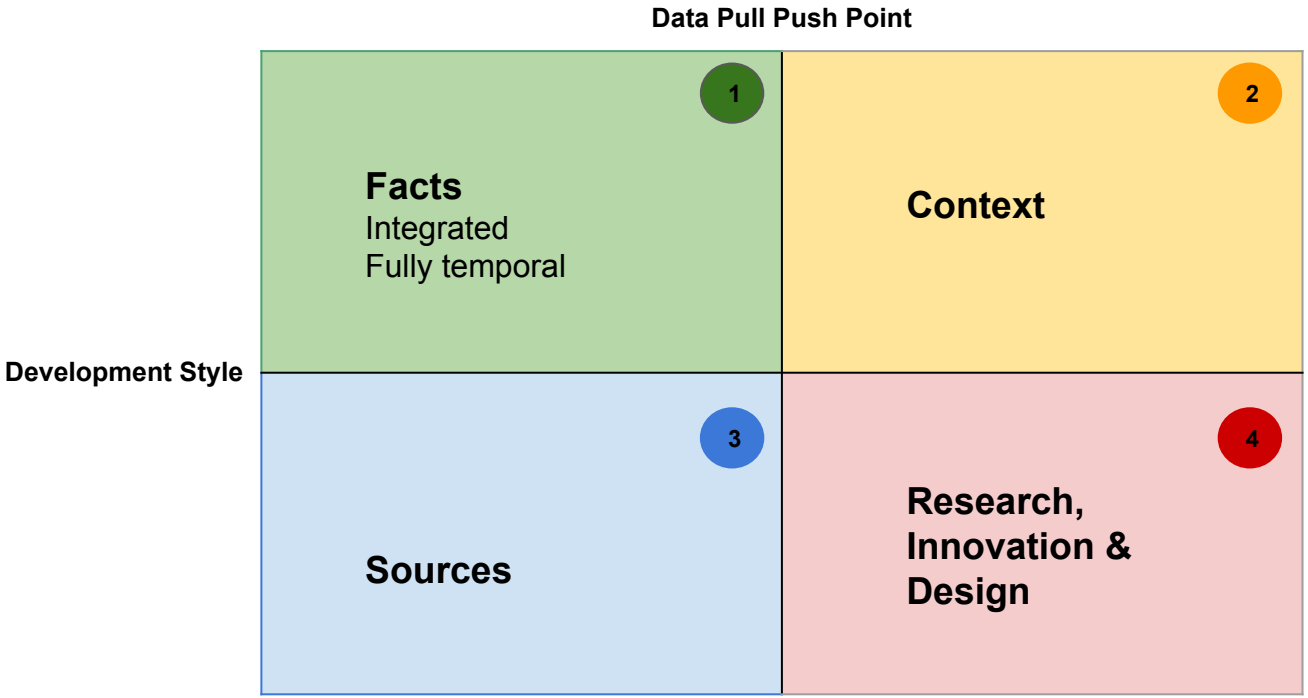
Unlimited Storage
Inexpensive storage unlimited data and easily scalable

Schema applied reload
Reload needs to accommodate schema changes over time

Decoupling Compute and Storage

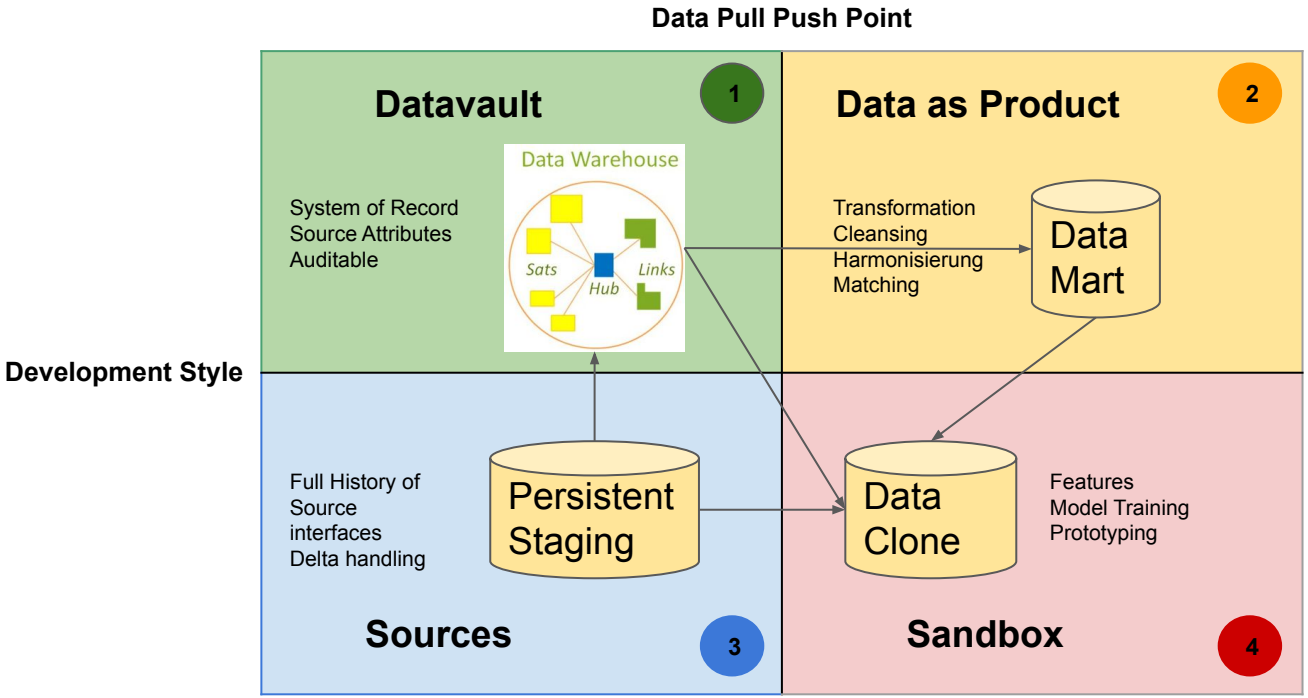


Data Management Quadrant

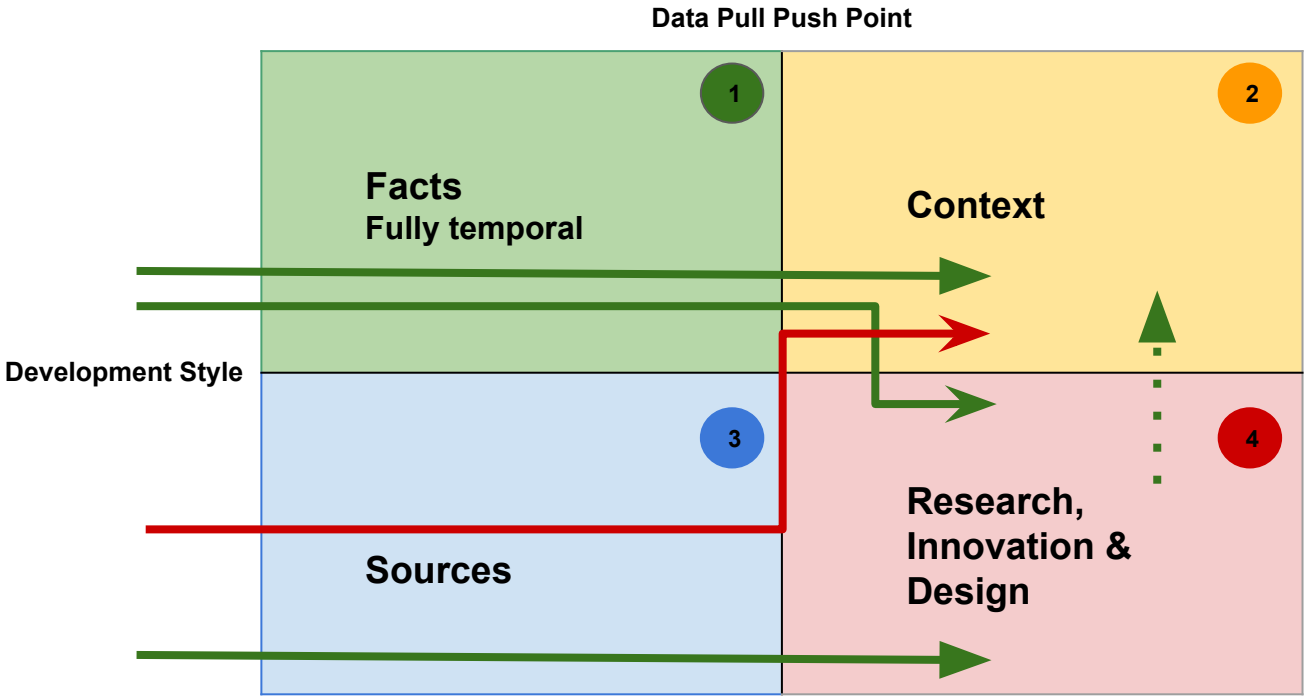


https://alligatorcompany.gitlab.io/acs-docs/datamanagement/

Data Management Quadrant - Layer / Komponenten



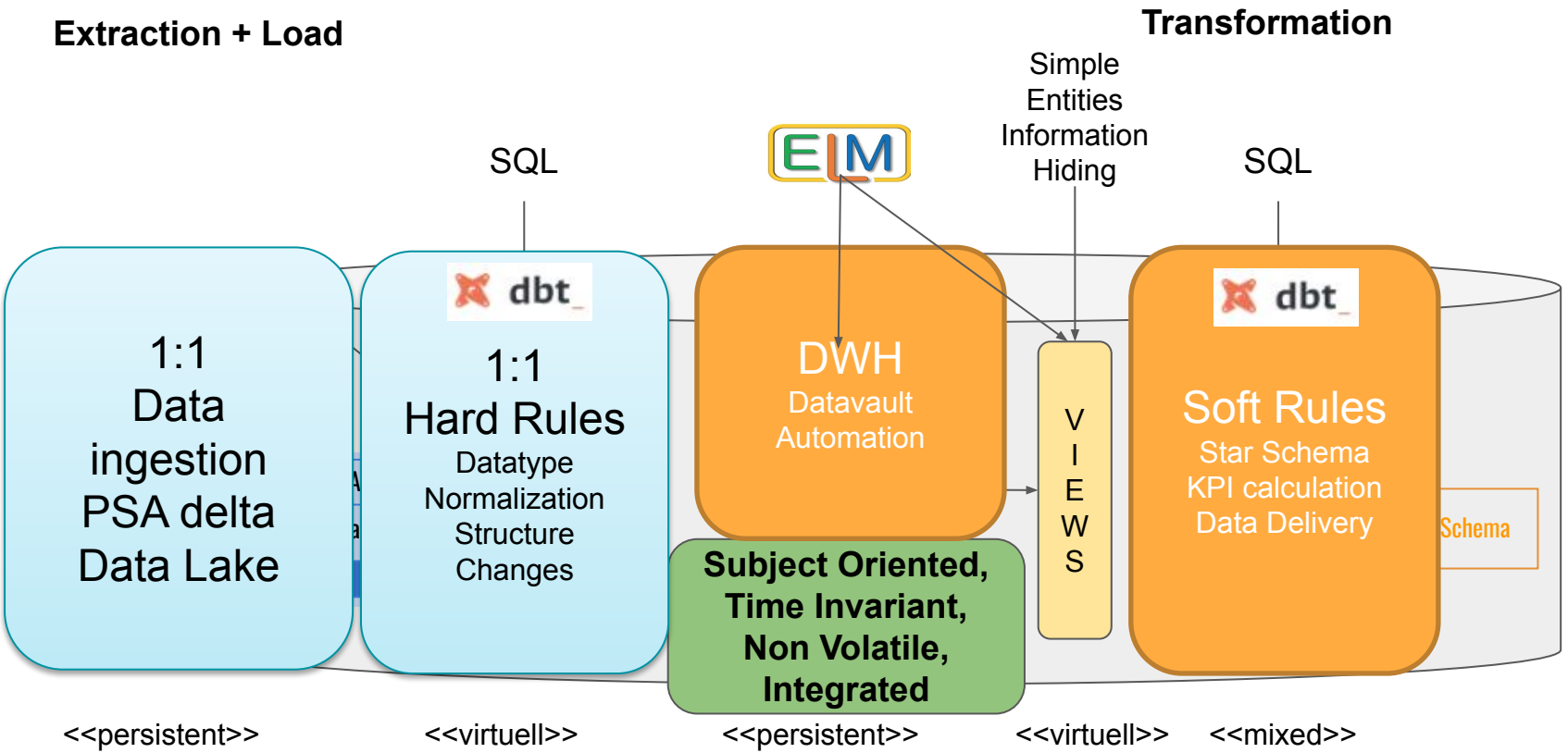
Data Management Quadrant - Governance



Quadrant 4 → 2
Operationalisierung
- Data as Product
- MLOps

Combine Datavault Automation and Analytics Engineering

https://alligatorcompany.gitlab.io/acs-docs/



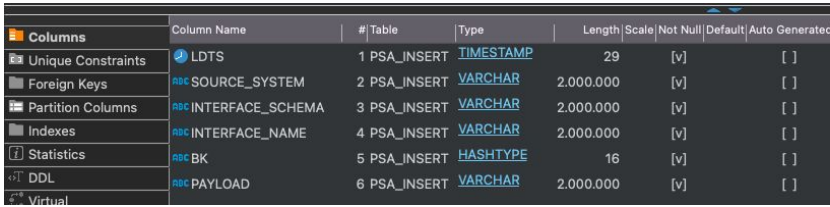
PSA - Schema On Read with DBT on Exasol

- **Generic PSA table stores JSON documents (PAYLOAD) per LDTS, Source, Schema, Relation and BK (technical primary key of the interface)**
- **Loading source interfaces via JDBC**

```
sql = "SELECT * FROM ( IMPORT FROM JDBC AT %s STATEMENT 'select * from %s.%s')" % (ctx.source_system, ctx.interface_schema, ctx.interface_name)
stmt = C.execute(sql)
for row in stmt:
        ctx.emit(ctx.source_system.lower(), ctx.interface_schema.lower(), ctx.interface_name.lower(), json.dumps({k.lower():v for k, v in row.items()}))
C.close()
```

- **Loading Delta via incremental DBT model** https://gitlab.com/alligatorcompany/acs-samples/fifadwh/-/raw/master/00_src/psa/models/psa_insert.sql
- **Schema on Read through DBT views:**

```
create or replace view ZZZ_Q3_SRC_fifa.stg_fifa_none_player as
with
sor_view_delta as (
   select
      cast( json_value( payload, '$.id') as bigint) as "ID",
      cast( json_value( payload, '$.player_api_id') as bigint) as "PLAYER_API_ID",
…
      cast( json_value( payload, '$.weight') as bigint) as "WEIGHT"
   from yyy_q3_psa.psa_insert
   where source_name = 'fifa' and interface_schema='None' and interface_name = 'player'
      and ldts = (
         select max(ldts)
         from yyy_q3_psa.psa_insert
         where source_name = 'fifa' and interface_schema='none' and interface_name = 'player'
      ))
select * from sor_view_delta;
```



Ausblick - Data Science in Cloud DBMS

Python und R in DB → supported von Exasol und Snowflake

Dbt-ml-preprocessing (https://github.com/omnata-labs/dbt-ml-preprocessing)

DBT in 1.3 wird Python Modelle im DAG zulassen https://github.com/dbt-labs/dbt-core/discussions/5261

Wiederverwendung der Datenintegration für Data Science

Verprobung von dbt python modules in Exasol

Performance und Kostenanalyse - Schema On Read in Cloud DBMS