DBRepo: A Data Repository System for Research Data in Databases

IEEE Big Data 2024

Martin Weise & Andreas Rauber

ISE, TU Wien, Austria

December 15th – 18th in Washington D.C., USA

Motivation

Data-driven research, but how do we store research data in databases?

store
process
share
preserve
access
...

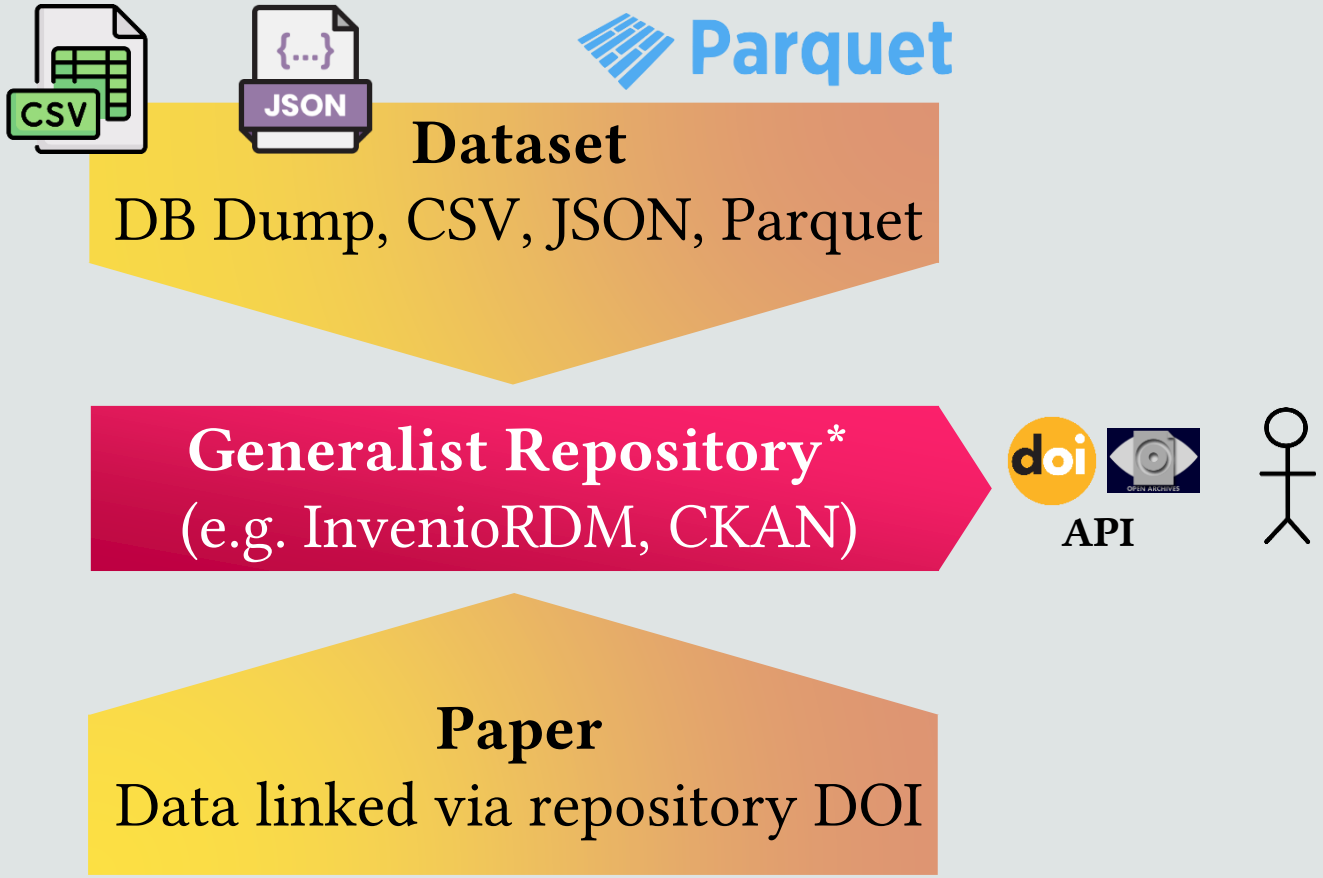
Having in mind:

- Data **structure** (filter, query)
- Data siloed in **research units**
- Data analysis happens **somewhere else** (VREs, TREs[1])
- Data **evolves over time** (knowledge *gained*, knowledge *invalidated*)
- Data needs to be **trustworthy** (reproducibility)
- Data **integration** tasks complexity (rich metadata needed)

Without burdening the researchers with yet *another* extra homework!

[1] System implementing the 5 safes framework of Desai et al., we use it synonymously with SPEs, SREs, SDS'

Naïve Approach



* from a technical perspective

Naïve Approach

Researcher exports dataset as **file** (database dump, CSV, JSON, Parquet)

Creates challenges:

- Data **curation** after repository deposit
- Data **preservation** (EOL decades later)

Shifts problems towards “re-users”:

- Allocate **storage** for download and extraction
- Install database engine (same version)
- Import dataset to database
- Formulate **queries** to explore data & create subset for own research

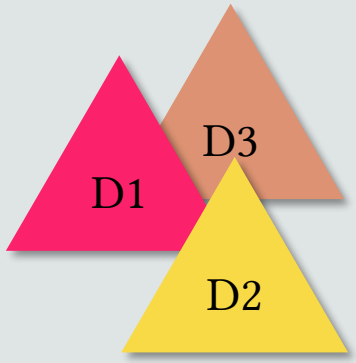
Problem

Secondary Use cannot mask the **problem severity** anymore:

Assume: researcher found reusable data and wants to publish her journal paper with **subsets** used as **supplementary** material… how?

What happens when the original data is modified and released to a **new version**? Release 100's similar dumps?

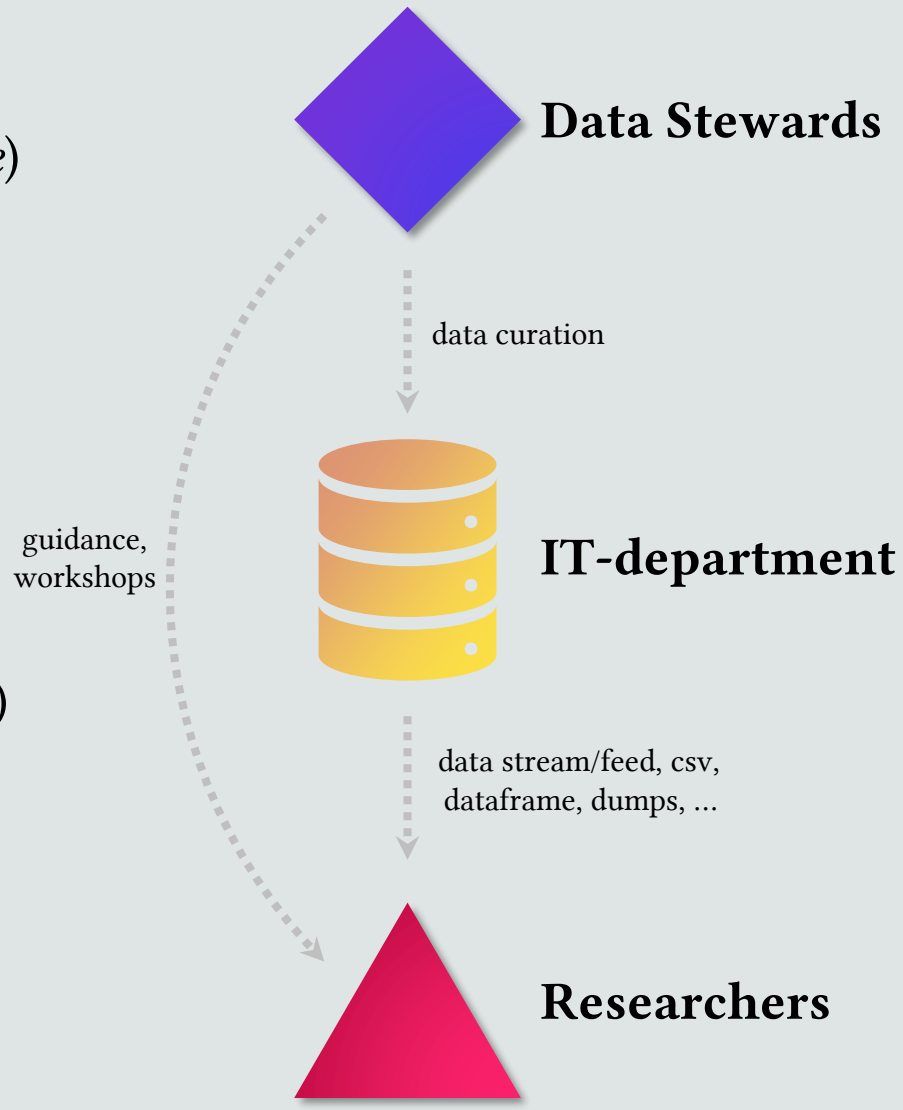
- **Delays** the time when research can happen
- Requires **compute infrastructure**
- Requires **technical knowledge** (import, subset)



Just release as similar dumps?

Proposition

- Separation of concerns (*right side*)
- Research database repository infrastructure (DBRepo)
  - microservice architecture
  - available
  - scalable
  - FAIR
  - interfaces (UI, API, Python)
- Analytics process for research data in databases



Background: Data Versioning

Temporal features in SQL:2011 standard, **sparse adoption** by RDBMS'

- Adding **invisible time window** (validity of tuples)
- Built-in **reproducibility** (adding time dimension)

			(invisible timestamps)	
ID	**Sensor**	**Temp**	**Valid From**	**Valid To**
1	A	23.1	t1	
2	B	25.8	t2	

SET `Temp` = 22.1

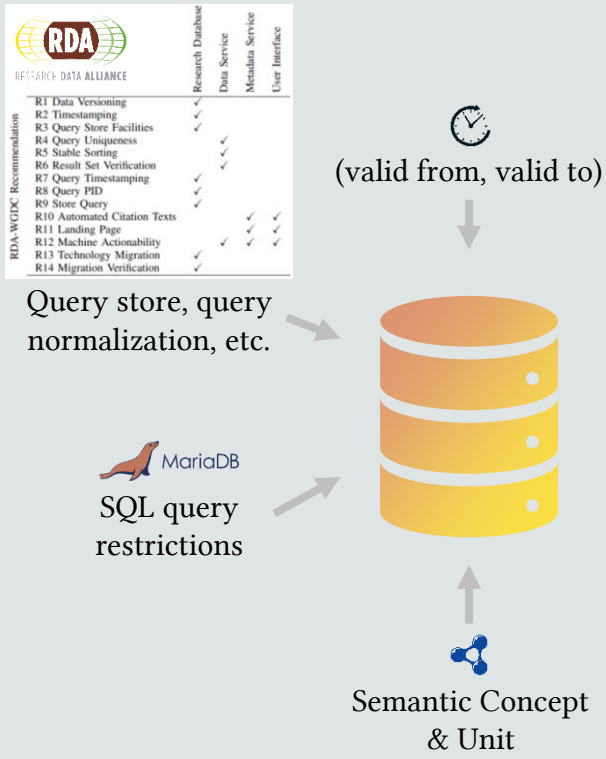
WHERE ID=1

				(invisible timestamps)
ID	**Sensor**	**Temp**	**Valid From**	**Valid To**
1	A	23.1	t1	t3
2	B	25.8	t2	
1	A	22.1	t3	

System Design: Research Database

Extra requirements to make database suitable for research

- **Data versioning**
- RDA-WGDC recommendations (e.g. query store)[doi:10.1162/99608f92.be565013]
- SQL query **restrictions**[1] (reproducibility of floating-point operations, vendor-independent)
- Semantic **concept** & **unit of measurement** (machine-understandable context)



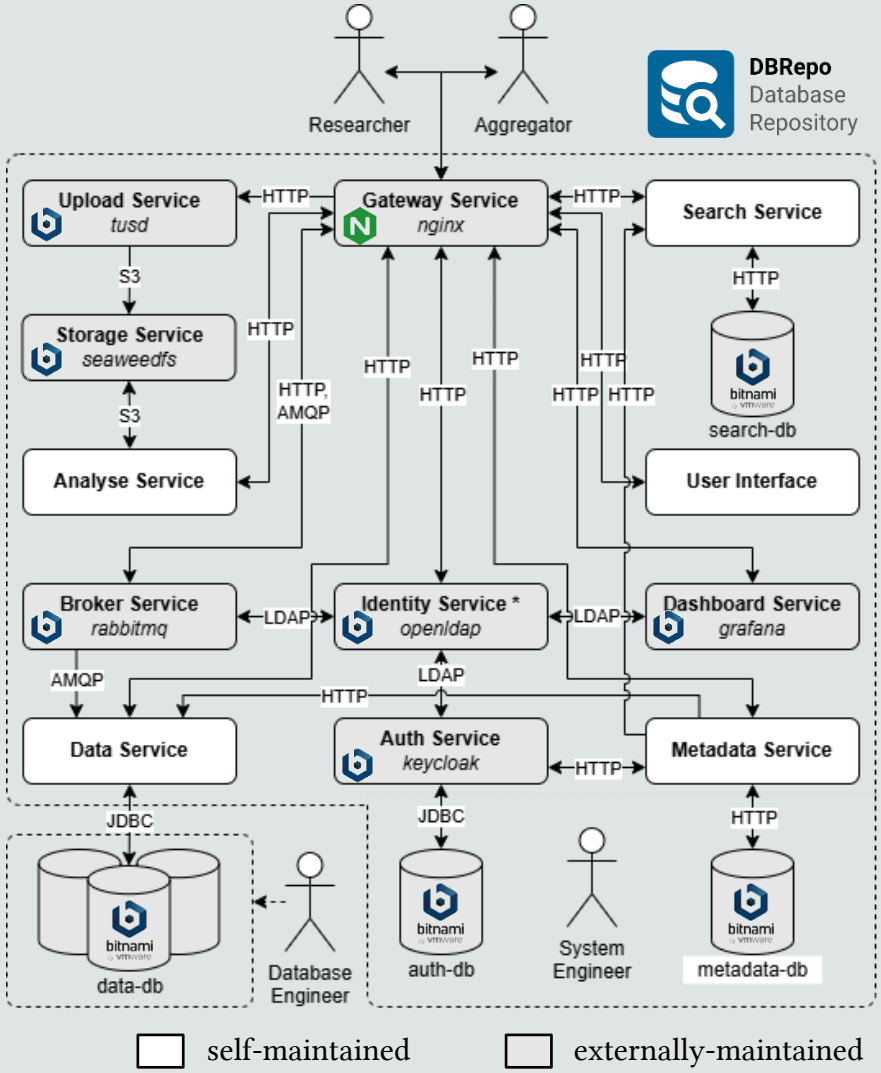
A subset DOI cites a normalized SQL query at a specific timestamp, instead of materialized (big) data

[1] AVG, BIT_AND, BIT_OR, BIT_XOR, COUNT, COUNTDISTINCT, GROUP_CONCAT, JSON_ARRAYAGG, JSON_OBJECTAGG, MAX, MIN, STD, STDDEV, STDDEV_POP, STDDEV_SAMP, SUM, VARIANCE, VAR_POP, VAR_SAMP

System Design: Architecture

Microservices

- stateless
- scalable
- exchangeable
  - **Storage Service** (S3)
    e.g. *minIO*
  - **Identity Service** (LDAP)
    e.g. own idp, *SAML 2.0*
  - **Search DB**
    e.g. *ElasticSearch*
- external community maintained open-source images



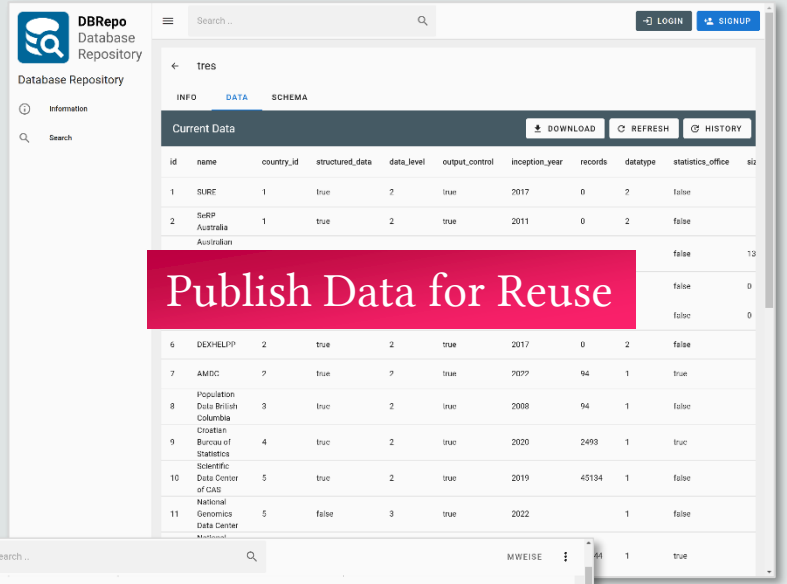
https://www.ifs.tuwien.ac.at/infrastructures/dbrepo/1.5/

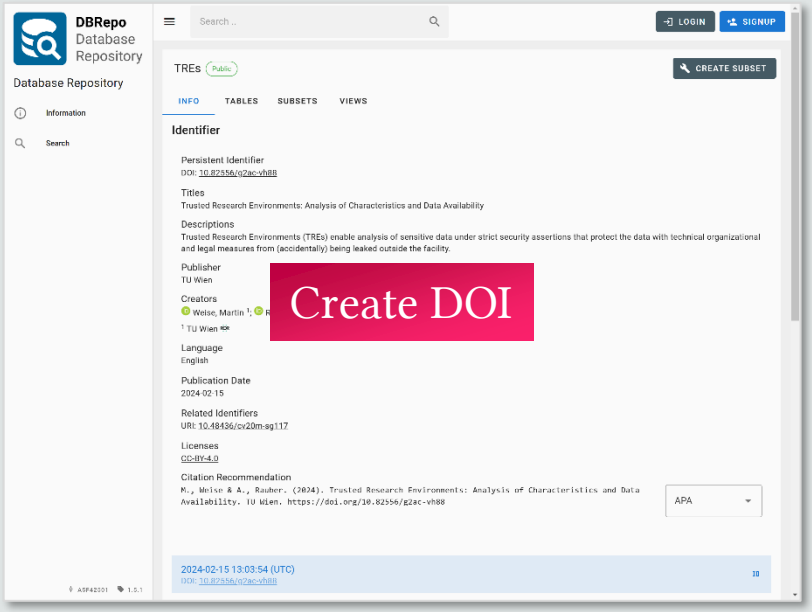
System Design : Interfaces for Research

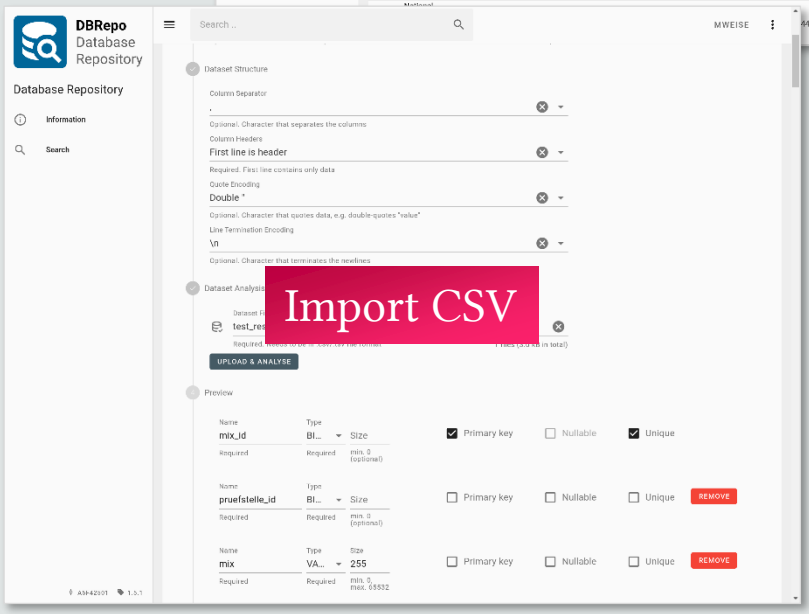
Database knowledge differs across disciplines, inclusive as possible

Novice uses graphical interface (UI)

- CRUD operations on data
- CSV upload/-download







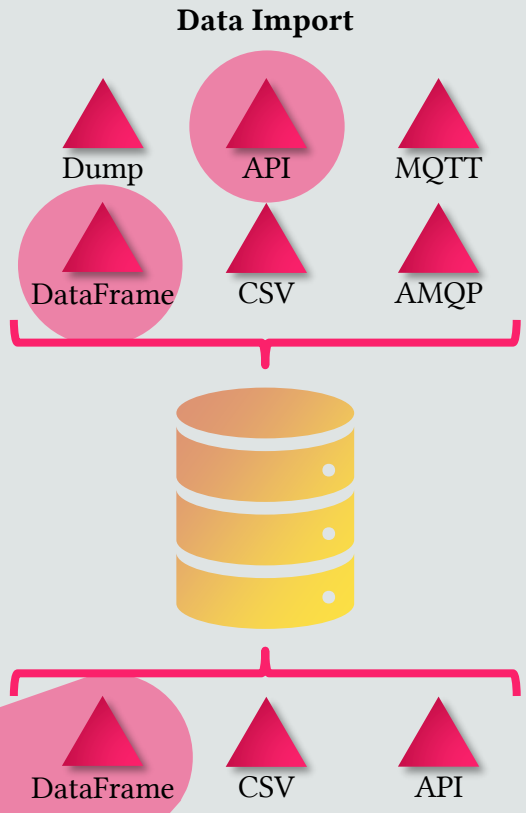
System Design : Interfaces for Research

Database knowledge differs across disciplines, inclusive as possible

Novice uses graphical interface (UI)

- CRUD operations on data
- CSV upload/-download
- Same power as API / Python library

Expert uses API / Python library



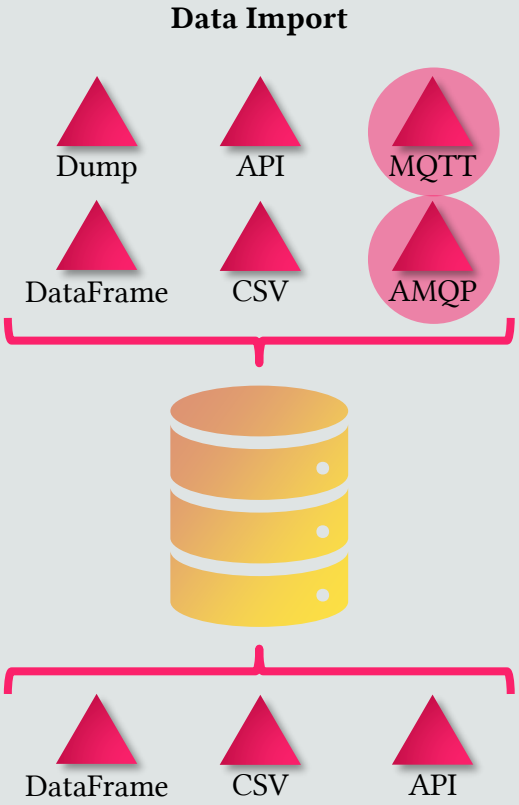

```python
client = RestClient(endpoint="https://dbrepo1.ec.tuwien.ac.at")
df = client.get_subset_data(database_id=27, subset_id=10, page=0, size=10_000, df=True)
for group in df.groupby(by="block_id"):
    block_id = group[0]
    df2 = group[1]
    data = pd.DataFrame({"intensity": df2[df2["position"] == 0]["value"].values,
                         "binding_energy": df2[df2["position"] == 1]["value"].values})
```

Data Access / -Export	User Interface	HTTP API	AMQP API	Python Library
Operations				
M1 Import .csv dataset	✓	✓		✓
M2a Create data tuple	✓	✓	✓	✓
M2b Read/Update/Delete data tuple	✓	✓		✓
M3 Create/Read/Delete database view	✓	✓		✓
M3 Create/Read/Delete database table	✓	✓		✓
M4 Create database subset	✓	✓		✓

Use Case: Teaching Industry 4.0

TU Wien & UTeM[doi:10.1109/icodse56892.2022.9971958]

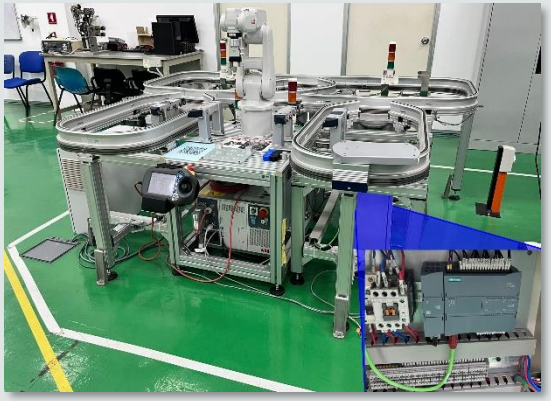
- Discrete multi-variant series production in **small quantities** (from design to assembly)
- **Digital skills** of manufacturing workforce
- Necessity of **data infrastructure**



EMCO Maxxturn 45
+ **SIEMENS** PLC

Power
MQTT 40t/s

Reconfig. Man. Sys.
+ **SIEMENS** PLC



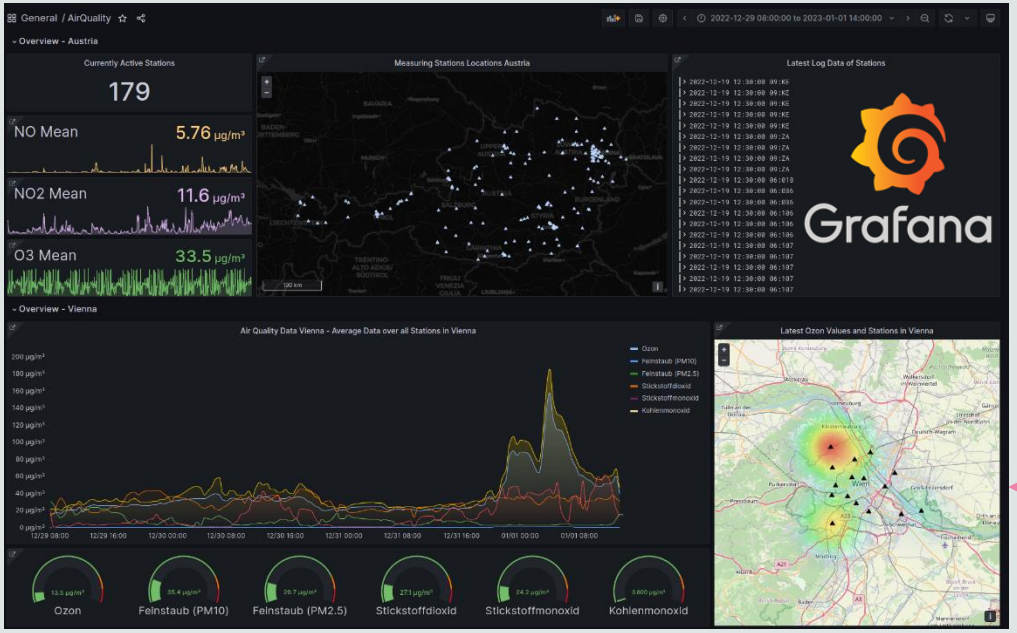
RabbitMQ

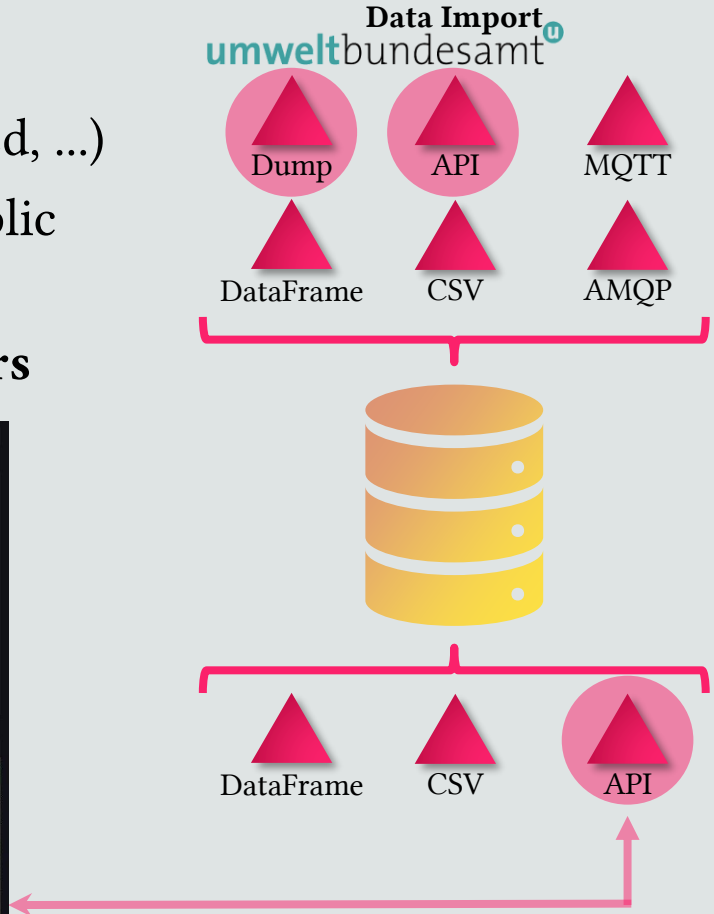
List of 11 use cases on the project website: https://www.ifs.tuwien.ac.at/infrastructures/dbrepo/1.5/examples/power/

Use Case: Decades of Air Quality Data

Monitoring climate environment in Austria

- 178 sensor stations (NO_x, CO, SO_2, Hg, Cd, ...)
- **Scrape data** public data, history not public
- Cooperation: **historic data** from 1990 (in 30min intervals) available → **34 years**





[1] https://dbrepo1.ec.tuwien.ac.at/dashboard/d/FLB9eAv4z/airquality

Position in the Repository Landscape

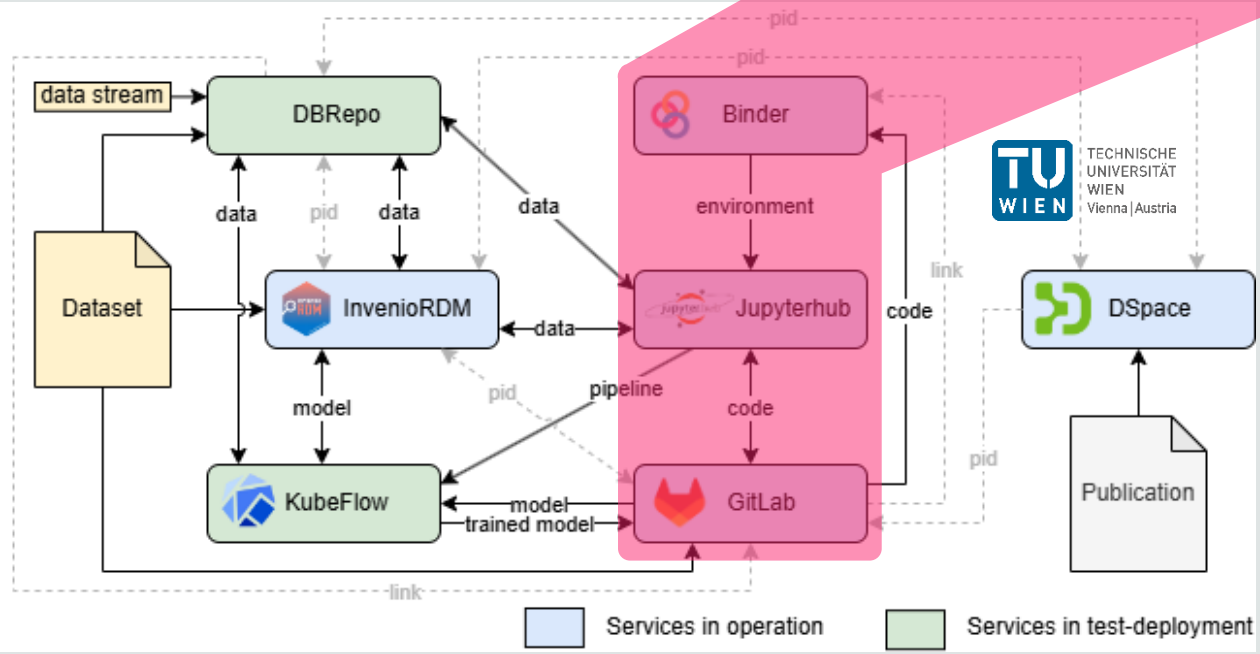
e.g. Accountability (reproducibility) in ML in RDM lectures

- Environment, preferably VRE
- Data (model, training, split)

Researcher benefits from this approach: identify erroneous data, publish update & contact cited publications to re-run computation

Reuse ML-data from DBRepo

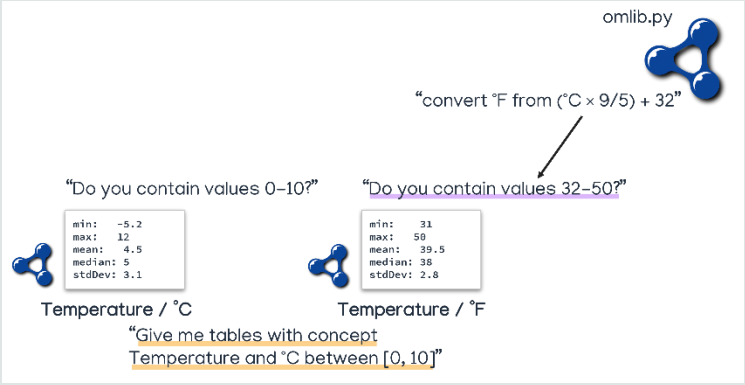
```
from dbrepo.RestClient import RestClient
import torch

client = RestClient("https://dbrepo.example.com")
df = client.get_table_data(48, 1516, df=True)
torch_tensor = torch.tensor(df['targets'].values)
...
```



Future Work

- Support for **low-level ontologies** e.g. organizational[1] (fallback to large ontologies i.e. WikiData)

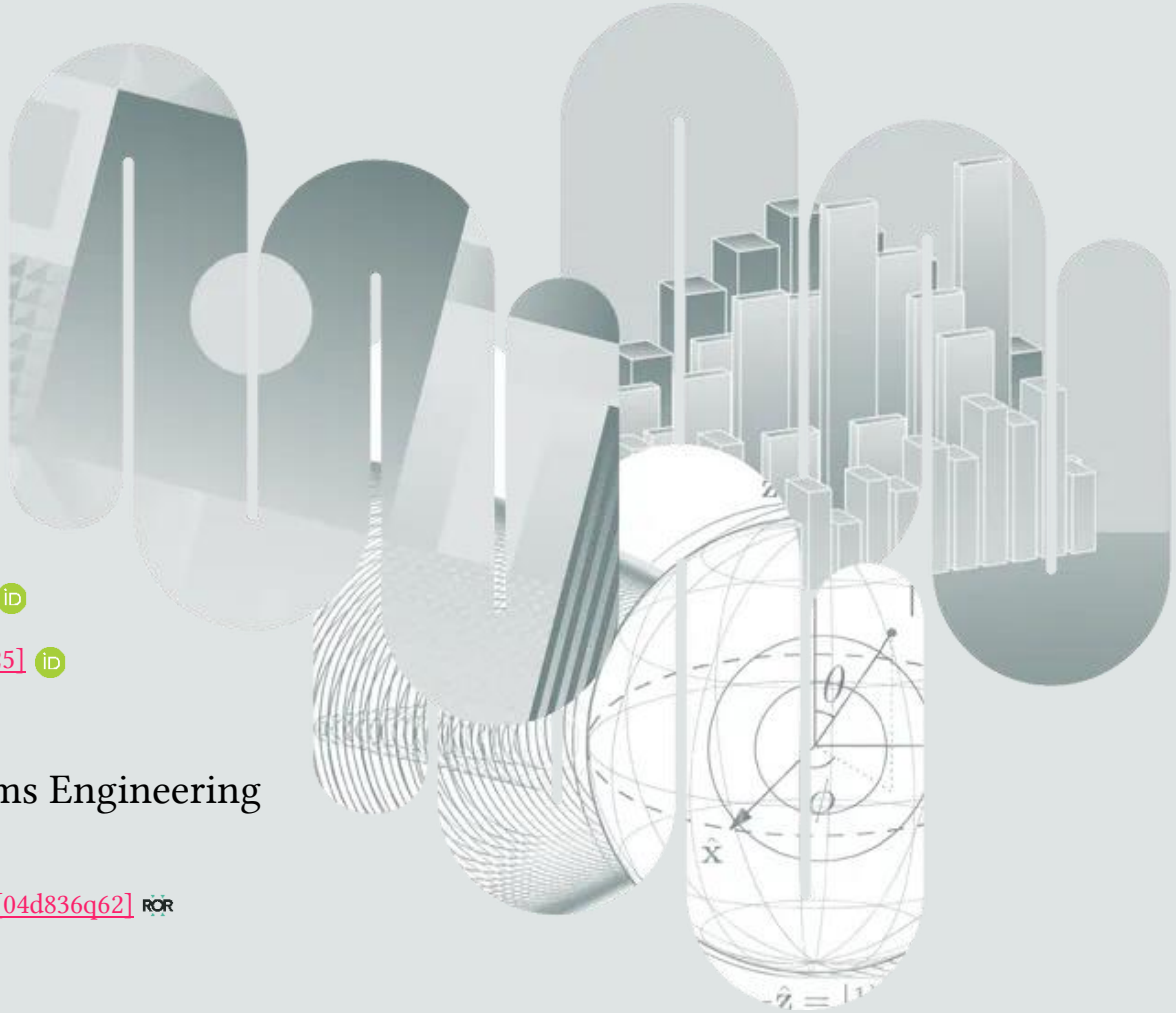
- **Auto-suggest** semantic concept & unit of measurement based on label prototype[doi:10.34726/7280] using BGE-M3 Embeddings[doi:10.48550/arXiv.2402.03216]
- Adoption of CODATA DRUM ontology and tools for **unit-independent search**[doi:10.5281/zenodo.14173690]



- Towards a system where machines can retrieve desired data without human intervention for e.g. systematic literature reviews

[1] https://www.auto.tuwien.ac.at/downloads/thinkhome/ontology/

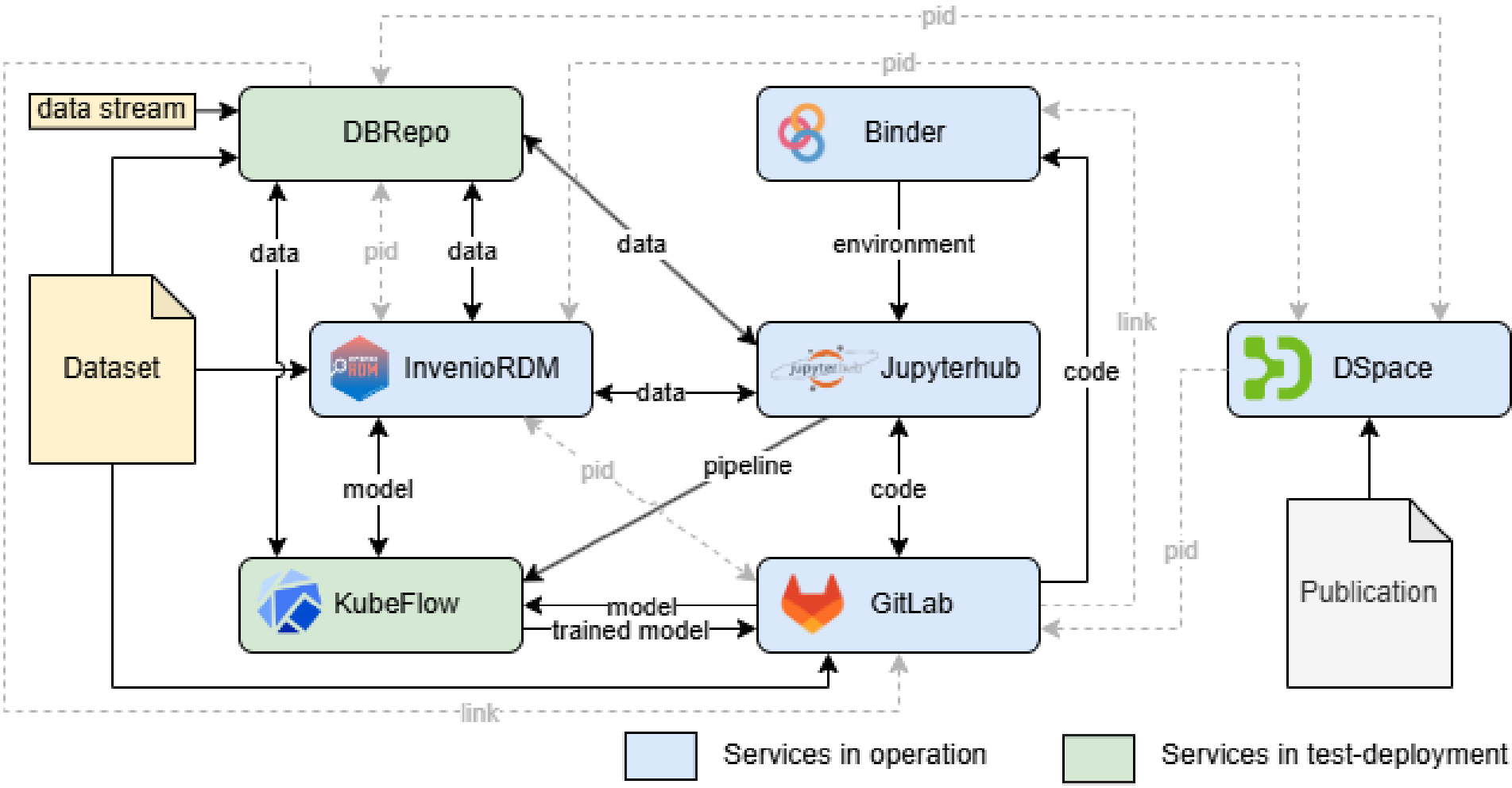
Contact



Martin Weise [0000-0003-4216-302X]
Andreas Rauber [0000-0002-9272-6225]

Institute of Information Systems Engineering
Data Science Research Unit
Technische Universität Wien [04d836q62]

Correspondence: martin.weise@tuwien.ac.at


pid
pid
data stream
DBRepo
Binder
data
pid
data
data
environment
Dataset
InvenioRDM
data
Jupyterhub
code
link
DSpace
model
pid
pipeline
code
KubeFlow
model
trained model
GitLab
pid
Publication
link
Services in operation
Services in test-deployment