Hands-on data management with open-source software: CaosDB

Florian Spreckelsen, Daniel Hornung
IndiScale GmbH

2023-03-23

LinkAhead

IndiScale Data Services

CaosDB

CaosDB: Agile, semantic data management

History

- CaosDB started at MPI-DS (Göttingen) around 2011
- Running stable since ca. 2016, released as open-source (AGPLv3)[1] in 2018
- Increasing user base since 2020
- Commercial support by IndiScale GmbH
  - distribution branded as LinkAhead

Caosdb
an open scientific database

CaosDB: Agile, semantic data management

History

- CaosDB started at MPI-DS (Göttingen) around 2011
- Running stable since ca. 2016, released as open-source (AGPLv3)[1] in 2018
- Increasing user base since 2020
- Commercial support by IndiScale GmbH
  - distribution branded as LinkAhead

Caosdb
an open scientific database

GitLab

[1] https://gitlab.com/caosdb

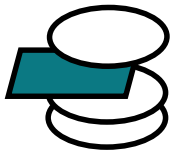
CaosDB: Agile, semantic data management

History

- CaosDB started at MPI-DS (Göttingen) around 2011
- Running stable since ca. 2016, released as open-source (AGPLv3)[1] in 2018
- Increasing user base since 2020
- Commercial support by IndiScale GmbH
  - distribution branded as LinkAhead

Caosdb
an open scientific database

GitLab

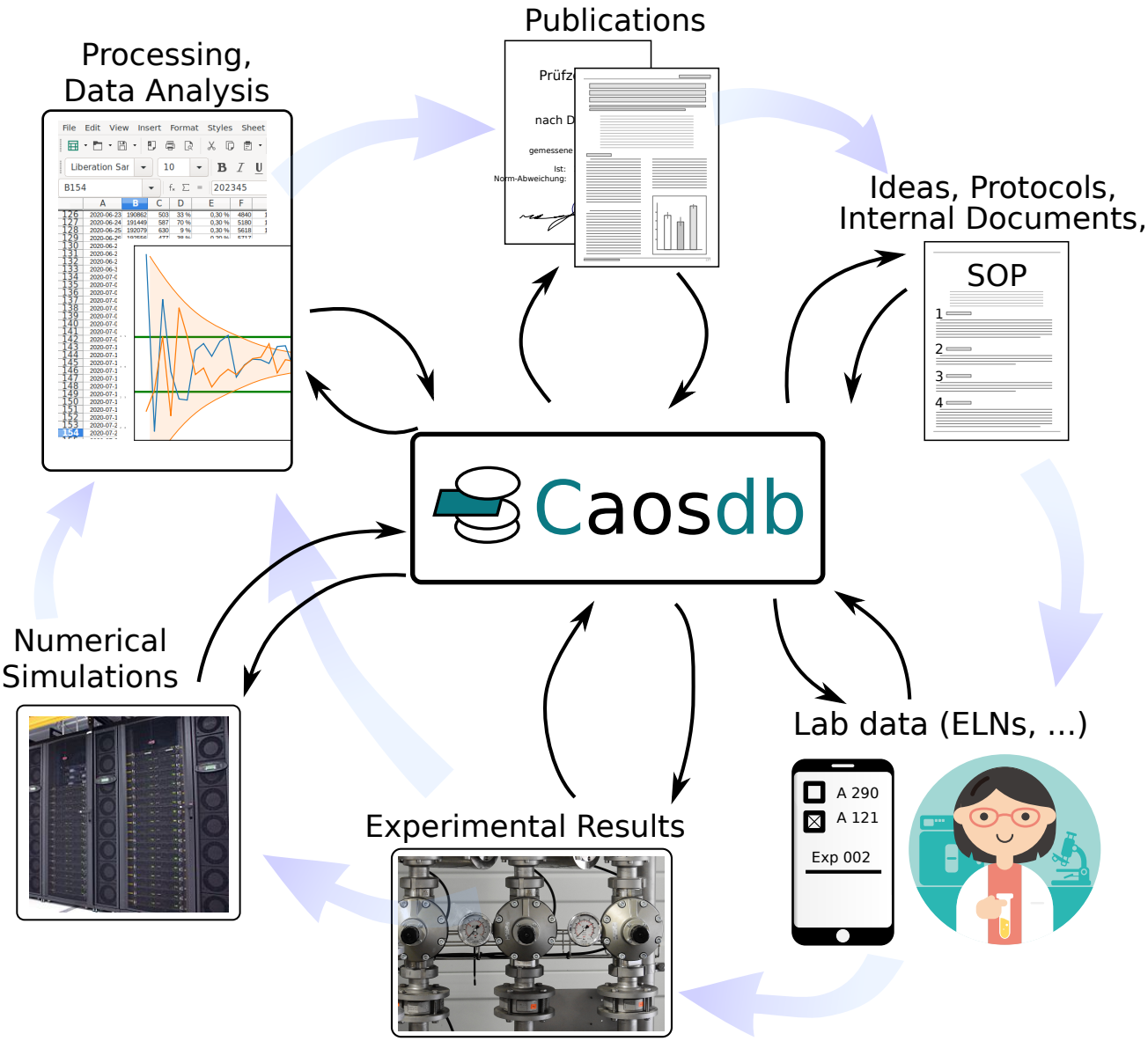


LinkAhead

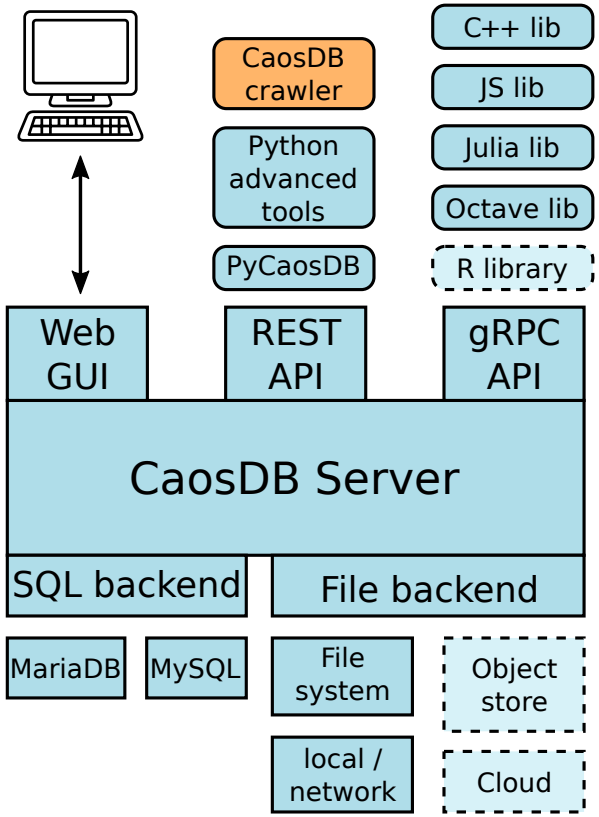
IndiScale
Data Services

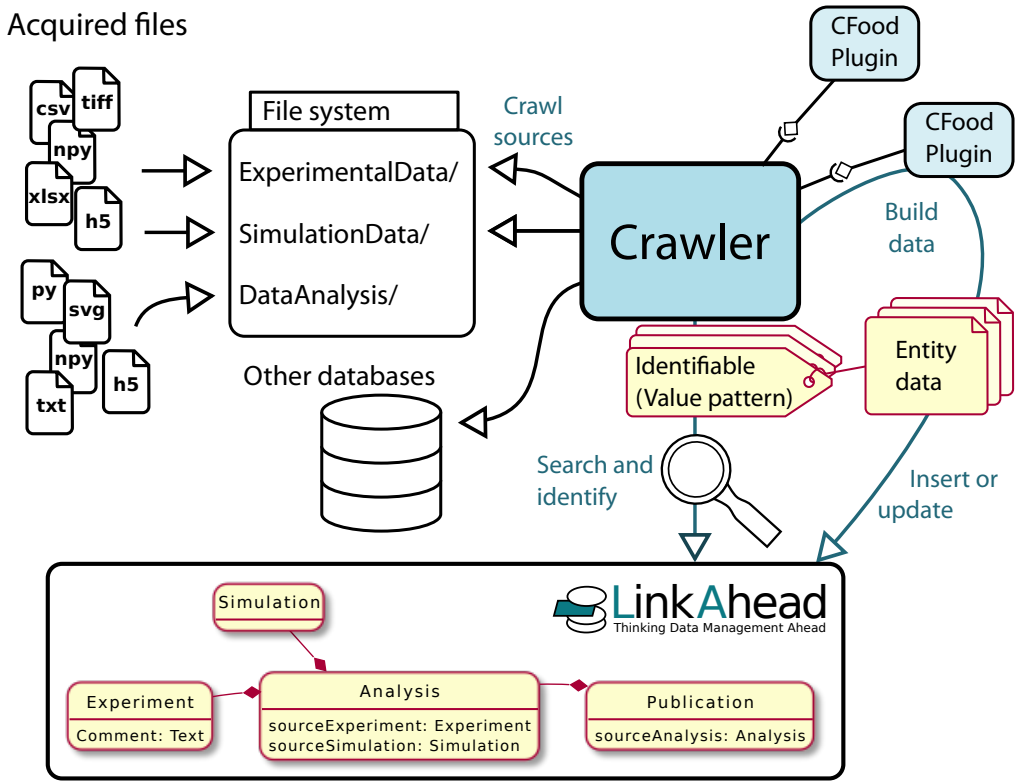
[1] https://gitlab.com/caosdb

Data Life Cycle



Architecture & Crawler LinkAhead



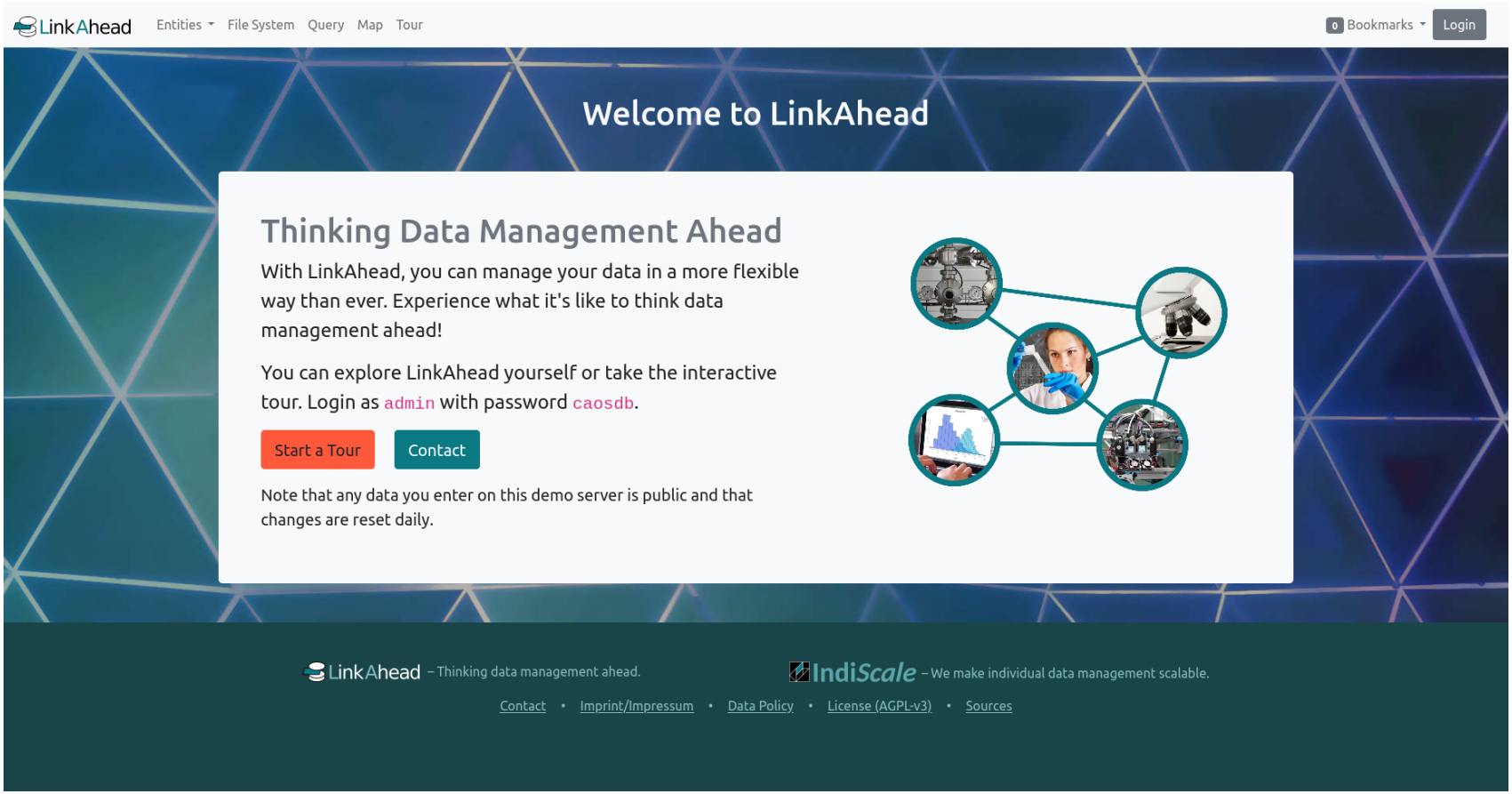


Useful Links

- Documentation: https://docs.indiscale.com/
- Gitlab repository: https://gitlab.com/caosdb/

Live-Demo

https://demo.indiscale.com



The Data Model

Presentation

- Main Menu

Task

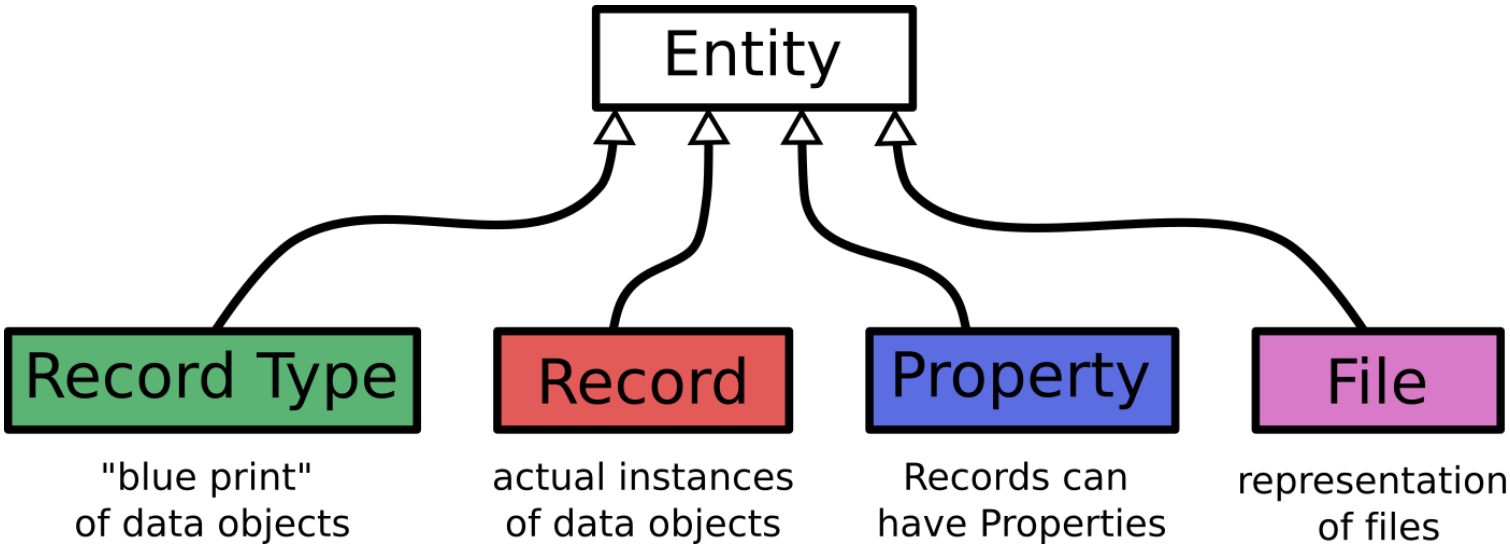
- Try the different buttons of the “Entities” Menu.

Note

- By default only 10 Entities are shown on one page. You can get to other pages with the “Next Page” and “Previous Page” buttons.

⇒ What are the differences between the options of the “Entities” menu?

The Data Model

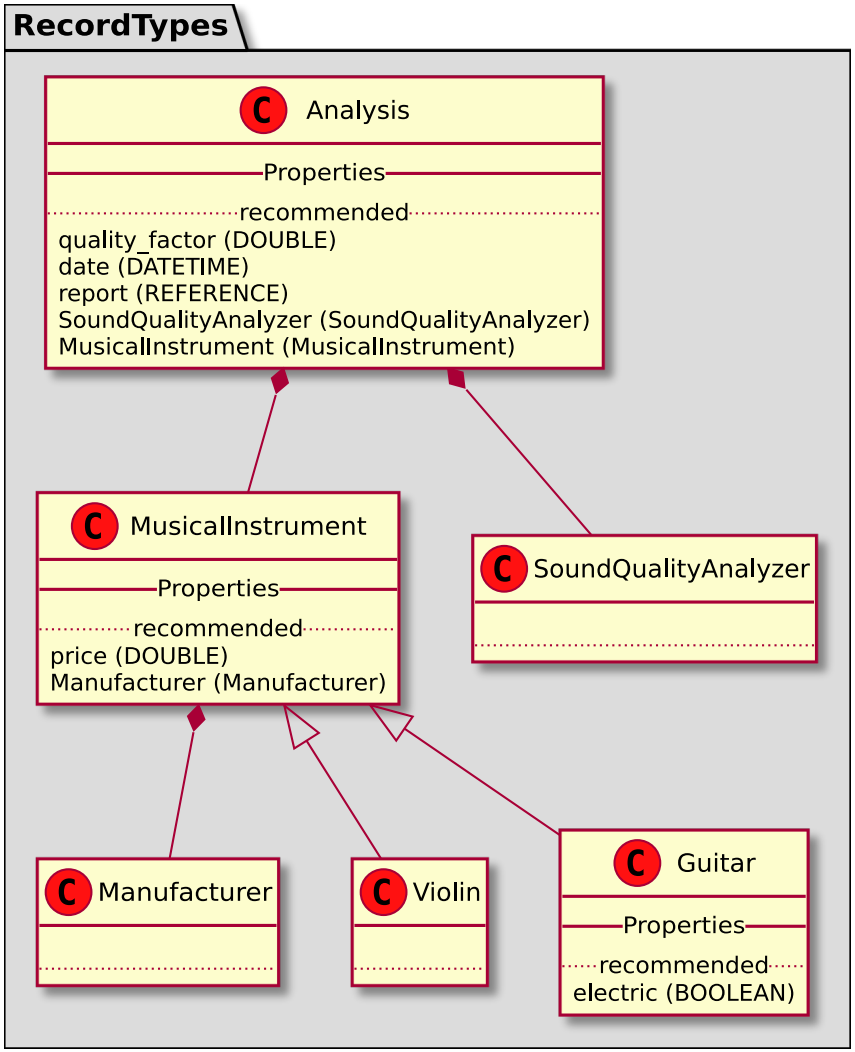


Note

RecordTypes and Properties define the ontology. Records store the actual data and represent things, e.g., a particular measurement.

For more details, read on at docs.indiscale.com/caosdb-server/Data-Model.html.

The Data Model



References in LinkAhead are directed.
If Record A references Record B, then:

- The referencing Record A has a corresponding Property.
- The referenced Record B does not.

Finding "back-references" in the web interface: References

Searching Data I

- The semantic data model allows efficient data access
- CaosDB Query Language (CQL) is used to search data

Presentation

- Simple Queries

Note

- Try to guess what the result will be, before actually executing the query.

Documentation on the query language:
https://docs.indiscale.com/caosdb-server/CaosDB-Query-Language.html

Presented Queries

- `FIND RECORD MusicalInstrument`
- `FIND Guitar`
- `FIND RecordType Guitar`
- `FIND RECORD 'Nice Guitar'`
- `FIND RECORD Nice*`
- `FIND Manufacturer`
- `FIND 'Budget Bikes'`

Searching Data: Using Properties

The power of CQL (CaosDB Query Language): using properties and references to refine search results

Find Queries:

- `FIND <Role> <Name> [Property Filter]`

Presentation

- FIND Queries with Property filter

Note

- Typically, the `Property Filter` has the form `<Property> <Operator> <Value>`, for example `length >= 0.7mm`.

See the CQL-tutorial for details on the possible filters.

Presented Queries

- `FIND RECORD MusicalInstrument WITH Manufacturer`
- `FIND RECORD MusicalInstrument WITH price=48`
- `FIND RECORD WITH price>48`
- `FIND RECORD WHICH HAS A PROPERTY temperature`
- `FIND RECORD WITH Manufacturer LIKE Budget*`
- `FIND RECORD WITH date in 2019`
- `FIND RECORD WHICH REFERENCES A Guitar`

Restricting the details of results

- Using `COUNT` instead of `FIND` will only return the number of entities in the result set

Note

This is often useful when experimenting with queries.

- Using `SELECT [Property Name], ... FROM` instead of `FIND` returns specific information in a table
- A comma-separated list of Property names can be provided behind the `SELECT` keyword.

Presentation

SELECT Queries

Presented Queries

- ```
  SELECT price, electric FROM Guitar
  ```
- ```
  SELECT name,
   quality_factor, report, date, Manufacturer.latitude
   FROM MusicalAnalysis WHICH REFERENCES A
   MusicalInstrument WITH electric=TRUE
  ```

For more, read on at
https://docs.indiscale.com/caosdb-server/CaosDB-Query-Language.html and
https://docs.indiscale.com/caosdb-webui/tutorials/query.html.

Automating LinkAhead: The Python library

This Hacky Hour's Juyter Notebook

Find the Jupyter Notebook for this hacky hour on
https://gitlab.indiscale.com/caosdb/src/notebooks-hacky-hour-dd23

Connect to existing Data Analysis Software

jupyter

LinkAhead

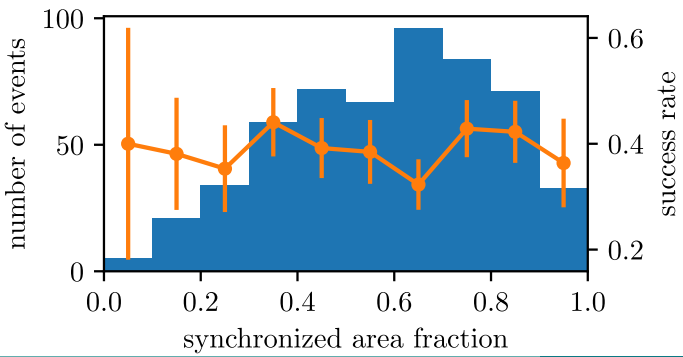
jupyter Introduction to the CaosDB-Python Client (autosaved)

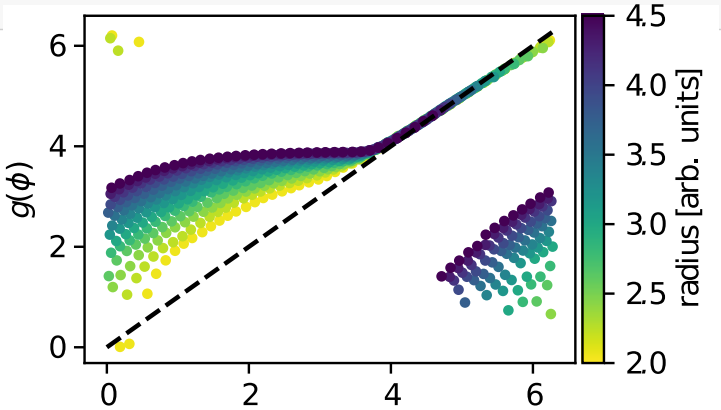
File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Markdown

Data Analysis with Jupyter

In [ ]:
```
import caosdb as db
data = db.execute_query("SELECT quality_factor FROM RECORD Analysis with quality_factor" )
table = to_table(data)
plt.plot(table.x, table.y)
```





Additional information

- Online demo:
  - https://demo.indiscale.com
- Source code and development: https://gitlab.com/caosdb
- CaosDB community chat: #caosdb:matrix.org
- Documentation: https://docs.indiscale.com
- Scientific article, published in ***Data***: https://doi.org/10.3390/data4020083

Summary

- Fully open-source software (AGPLv3)

Summary

- Fully open-source software (AGPLv3)
- Powerful search options

Summary

- Fully open-source software (AGPLv3)
- Powerful search options
- CaosDB can be easily extended or modified

Summary

- Fully open-source software (AGPLv3)
- Powerful search options
- CaosDB can be easily extended or modified
- Integrates into existing workflows

Summary

- Fully open-source software (AGPLv3)
- Powerful search options
- CaosDB can be easily extended or modified
- Integrates into existing workflows
- Stores semantic links together with the data

Thank you for your interest!

Appendix

- Comparison to SPARQL (RDF query language for e.g. WikiData)
- Data model evolution (at the AWI)

Comparison to SPARQL

SPARQL

```
SELECT DISTINCT ?item ?itemLabel ?givenName ?familyName WHERE {
    ?item wdt:P31 wd:Q5;  # Any instance of a human.
          wdt:P27 wd:Q145; # United Kingdom
          wdt:P21 wd:Q6581072; # female
          wdt:P106 wd:Q36180; # writer
          wdt:P569 ?birthday;
          wdt:P570 ?diedon;
          wdt:P734 [rdfs:label ?familyName];
          wdt:P735 [rdfs:label ?givenName].
  FILTER(?birthday > "1870-01-01"^^xsd:dateTime
         && ?diedon < "1950-01-01"^^xsd:dateTime)
  FILTER(regex(?givenName, "M.*") || regex(?familyName, "M.*"))
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
  }
```

Comparison to SPARQL

SPARQL

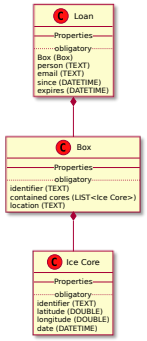
```
SELECT DISTINCT ?item ?itemLabel ?givenName ?familyName WHERE {
    ?item wdt:P31 wd:Q5;  # Any instance of a human.
          wdt:P27 wd:Q145; # United Kingdom
          wdt:P21 wd:Q6581072; # female
          wdt:P106 wd:Q36180; # writer
          wdt:P569 ?birthday;
          wdt:P570 ?diedon;
          wdt:P734 [rdfs:label ?familyName];
          wdt:P735 [rdfs:label ?givenName].
  FILTER(?birthday > "1870-01-01"^^xsd:dateTime
         && ?diedon < "1950-01-01"^^xsd:dateTime)
  FILTER(regex(?givenName, "M.*") || regex(?familyName, "M.*"))
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
  }
```

CaosDB query language

```
SELECT given_name, family_name FROM Writer
  WITH gender=f AND country=UK AND birthday > 1870 AND death < 1950
       AND (given_name LIKE "M*" OR family_name LIKE "M*")
```

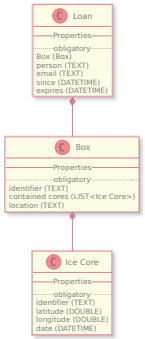
Data model evolution

Data model
ca. 2018

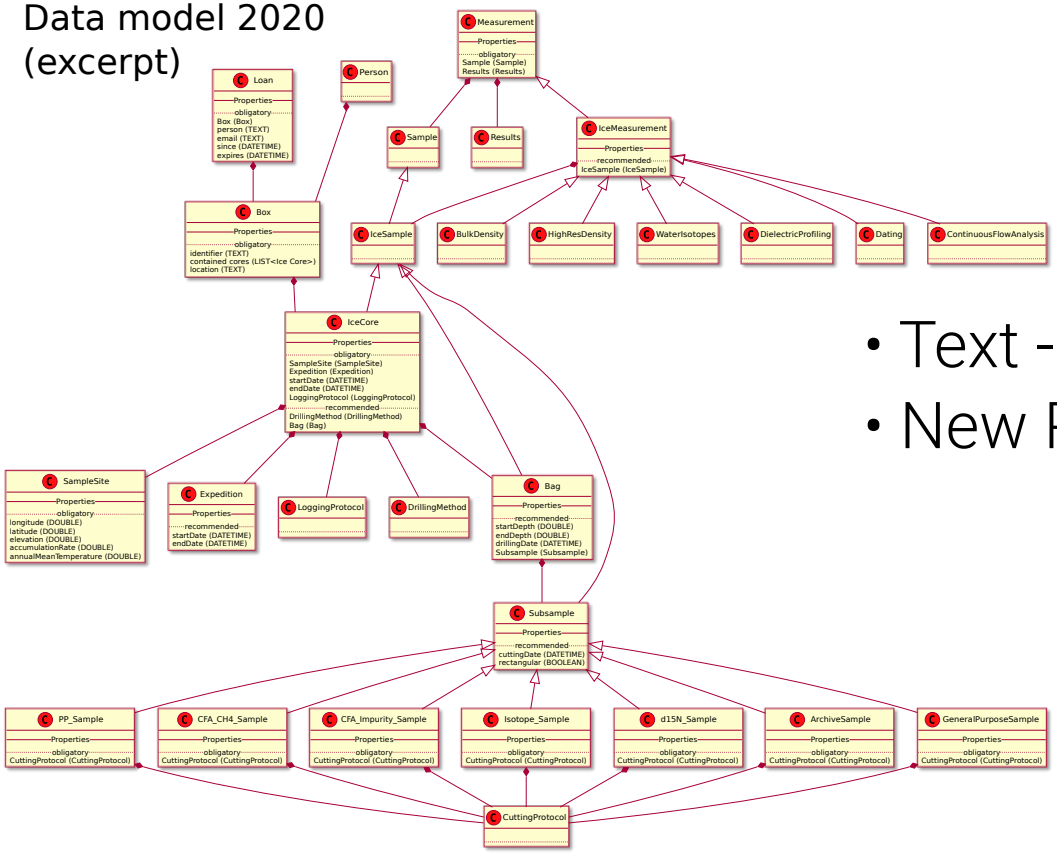


Data model evolution

Data model ca. 2018



Data model 2020 (excerpt)



- Text -> object entities
- New RecordTypes