Combining file-based with RDMS-based scientific workflows using the LinkAhead-crawler

Henrik tom Wörden, Florian Spreckelsen, Stefan Luther,
Ulrich Parlitz, Alexander Schlemmer

2024-03-06

Article

Open Access Article

Mapping Hierarchical File Structures to Semantic Data Models for Efficient Data Integration into Research Data Management Systems

by Henrik tom Wörden [1], Florian Spreckelsen [1], Stefan Luther [2,3,4,5], Ulrich Parlitz [2,3,4] and Alexander Schlemmer [2,4,*]

1 Indiscale GmbH, 37083 Göttingen, Germany
2 Max Planck Institute for Dynamics and Self-Organization, 37077 Göttingen, Germany
3 Institute for the Dynamics of Complex Systems, Georg-August-Universität, 37077 Göttingen, Germany
4 German Center for Cardiovascular Research (DZHK), Partner Site Göttingen, 37075 Göttingen, Germany
5 Institute of Pharmacology and Toxicology, University Medical Center Göttingen, 37075 Göttingen, Germany
* Author to whom correspondence should be addressed.

Data **2024**, *9*(2), 24; **https://doi.org/10.3390/data9020024**

Submission received: 15 August 2023 / Revised: 8 December 2023 / Accepted: 18 December 2023 / Published: 26 January 2024

(This article belongs to the Section **Information Systems and Data Management**)

tom Wörden et. al., *Data* 2024, 9, 24.
https://doi.org/10.3390/data9020024

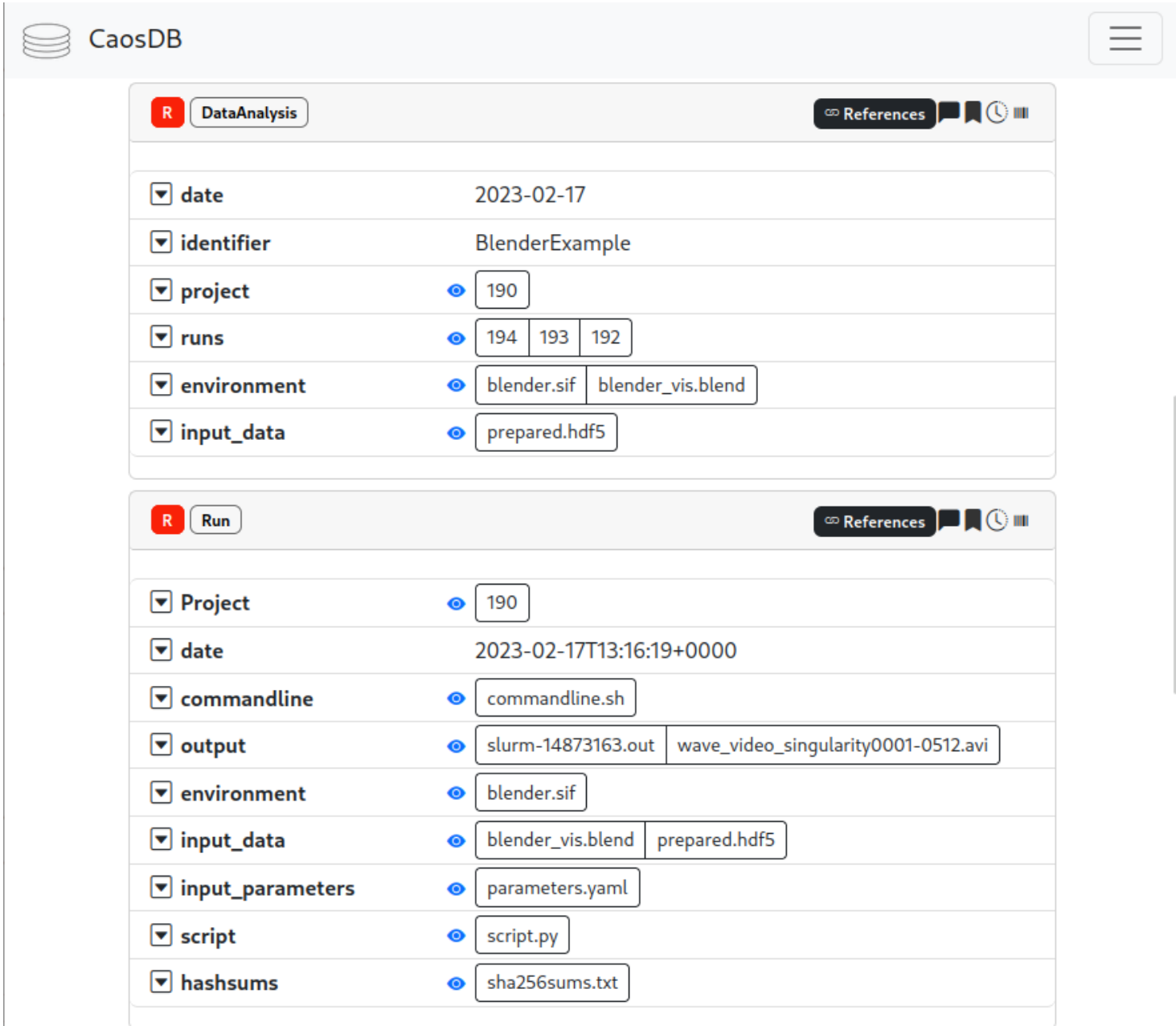
Research Question

- Data integration (e.g. getting data from the disk into databases / RDMS) time-consuming, especially with heterogeneous file layouts and data formats. How can we simplify that?

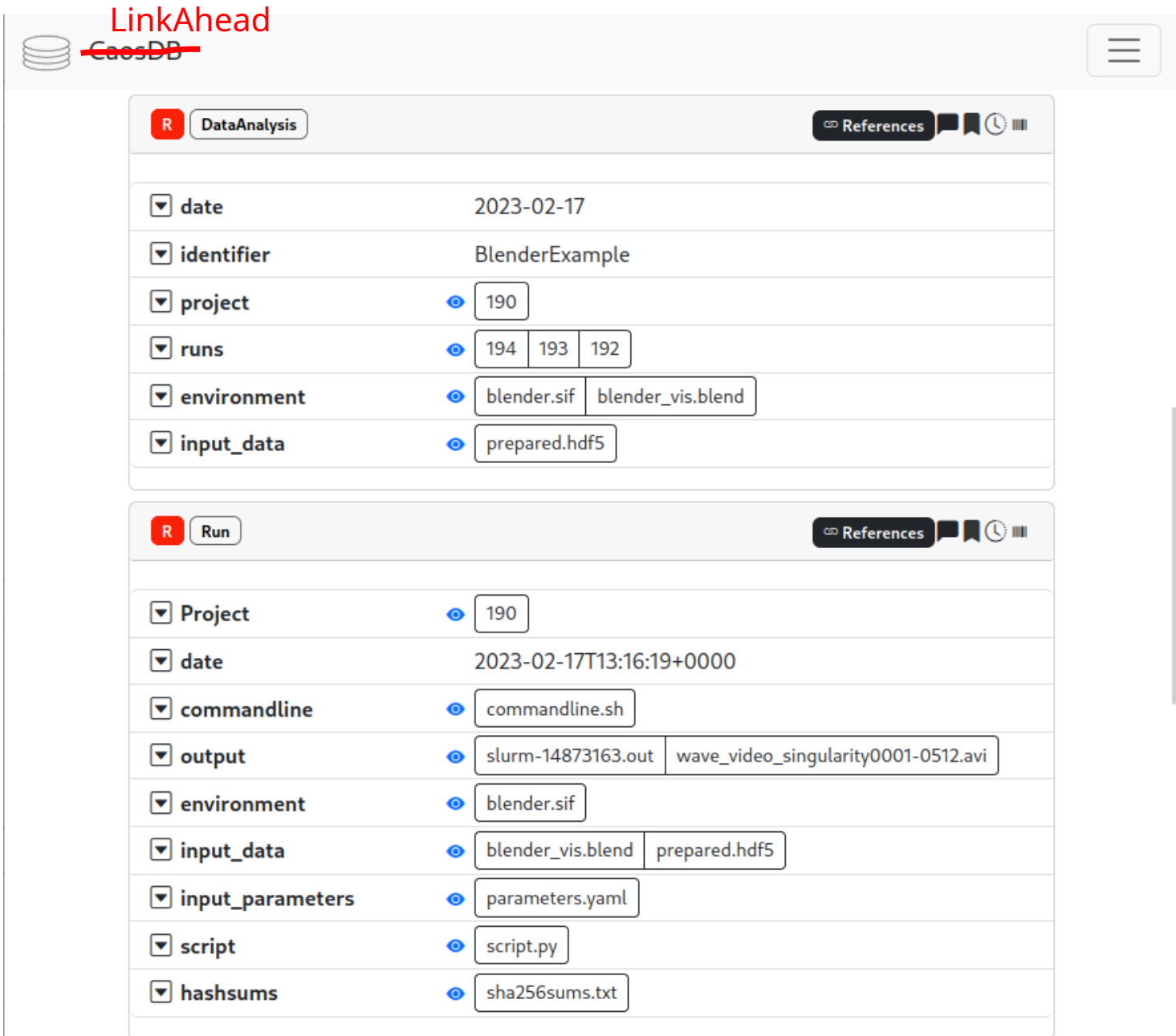
Research Question

- Data integration (e.g. getting data from the disk into databases / RDMS) time-consuming, especially with heterogeneous file layouts and data formats. How can we simplify that?
- How can we use the RDMS simultaneously to traditional file systems (for interoperability)?

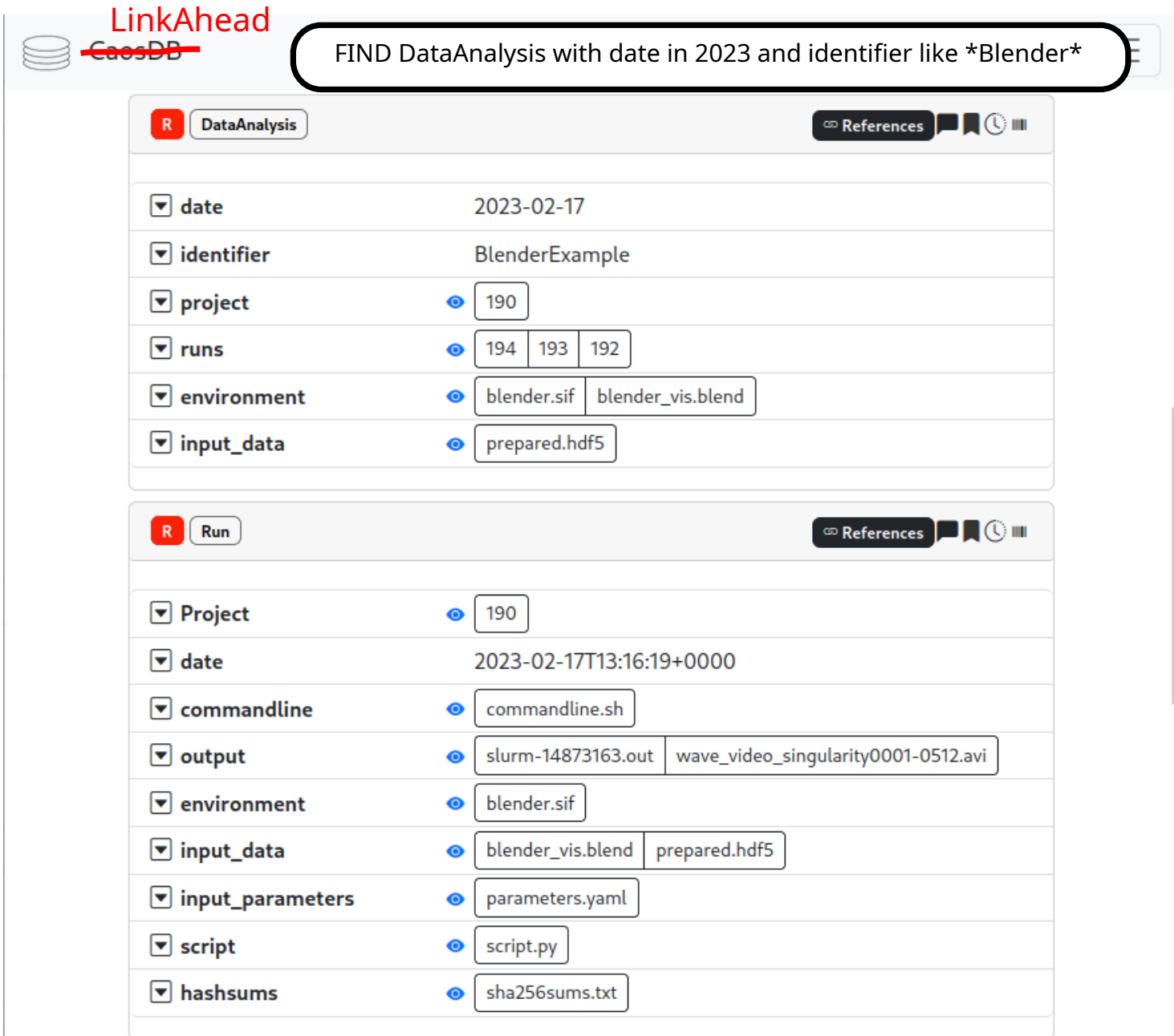
LinkAhead (formerly: CaosDB)



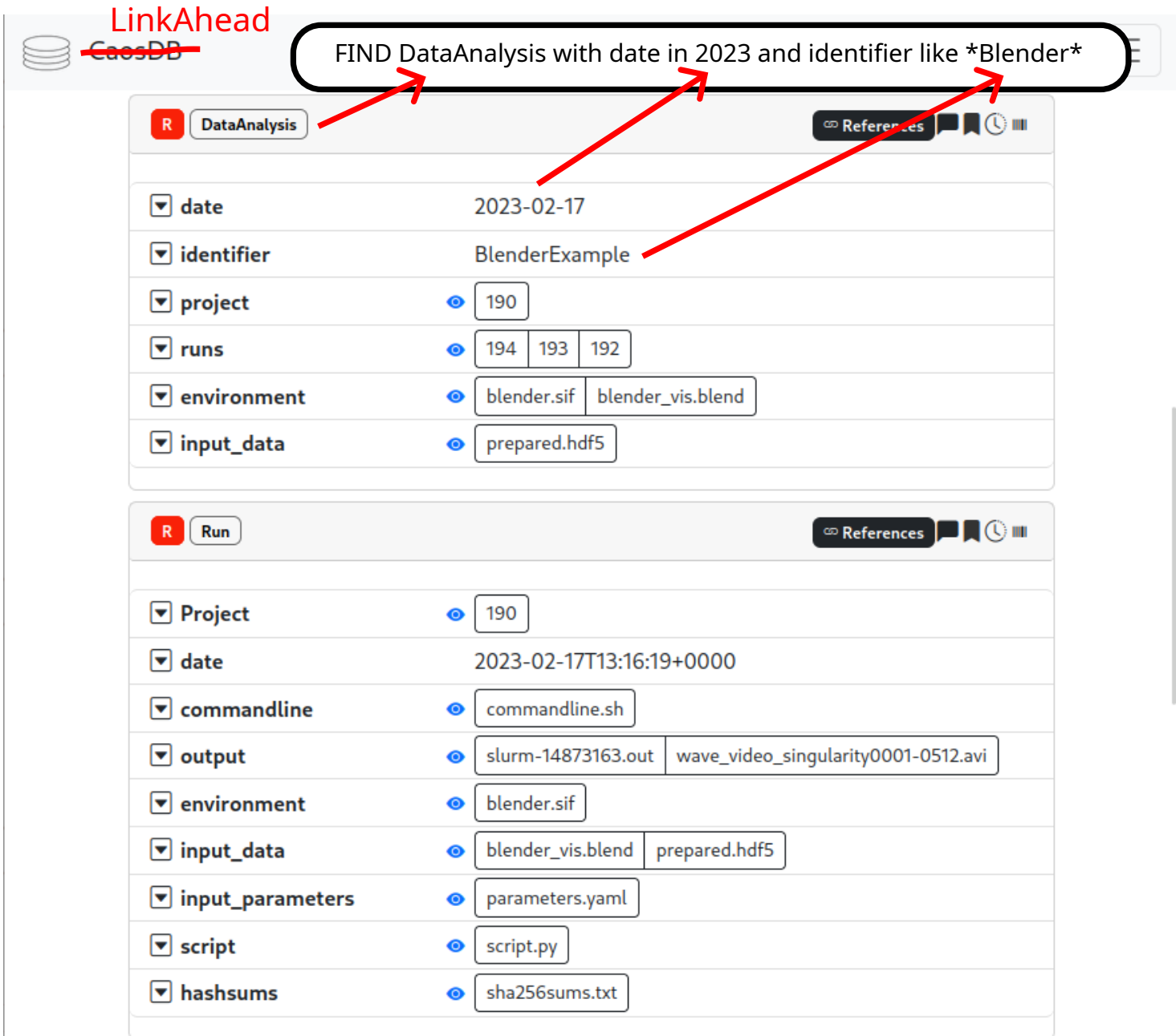
LinkAhead (formerly: CaosDB)



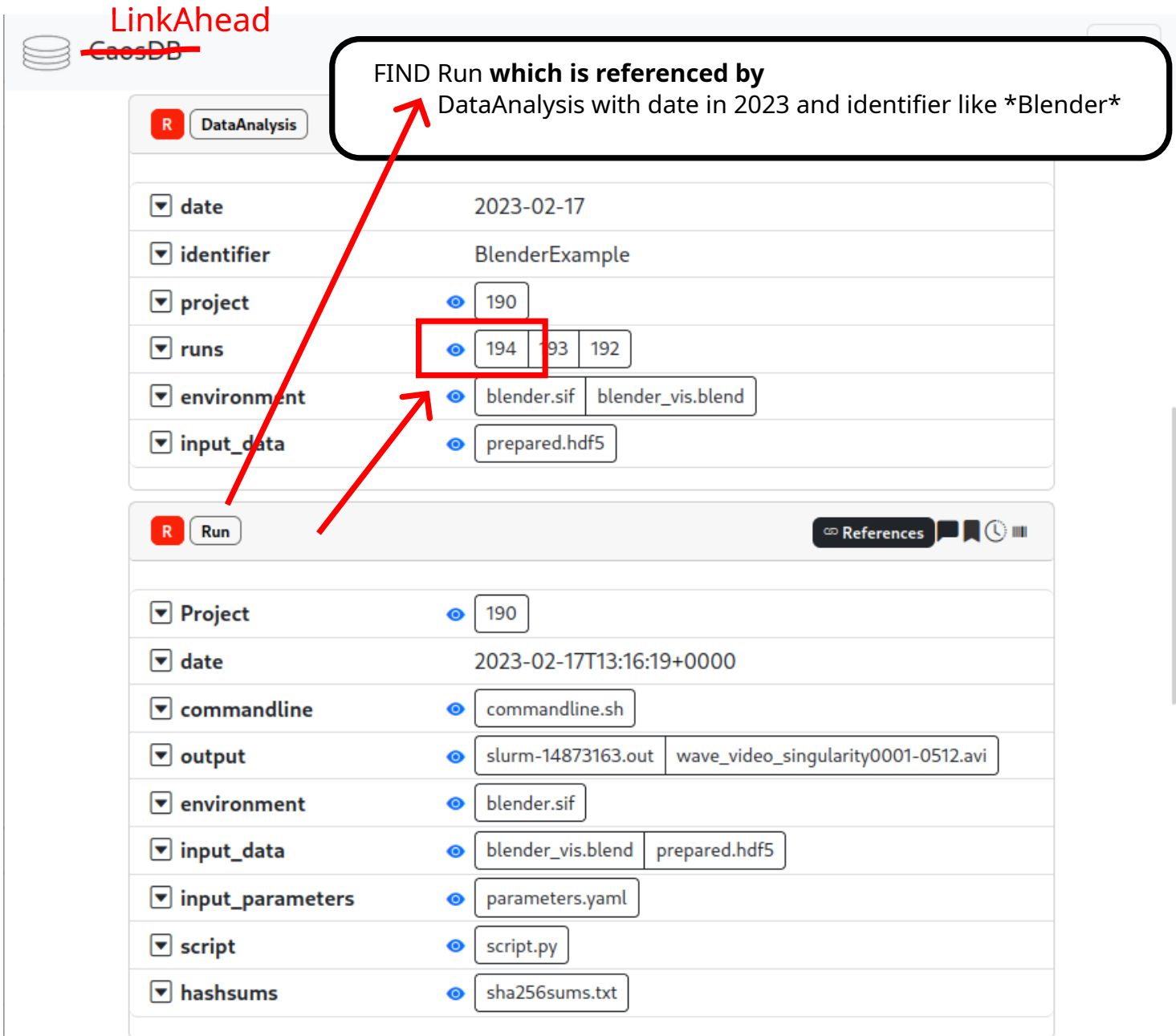
LinkAhead (formerly: CaosDB)



LinkAhead (formerly: CaosDB)



LinkAhead (formerly: CaosDB)



LinkAhead (formerly: CaosDB)

- Semantic Research Data Management System (RDMS)
  Fitschen et.al., *Data* 2019, 10.3390/data4020083
- Developed at the Max Planck Institute for Dynamics and Self-Organization (Göttingen) since 2010
- Open Source project since 2018: gitlab.com/linkahead
- Commercial support available by IndiScale GmbH[1] (since 2019)
- Currently approximately 20-30 instances in very different scientific disciplines
- Main features (very briefly): Flexible semantic data model, highly modular data crawler, semantic search

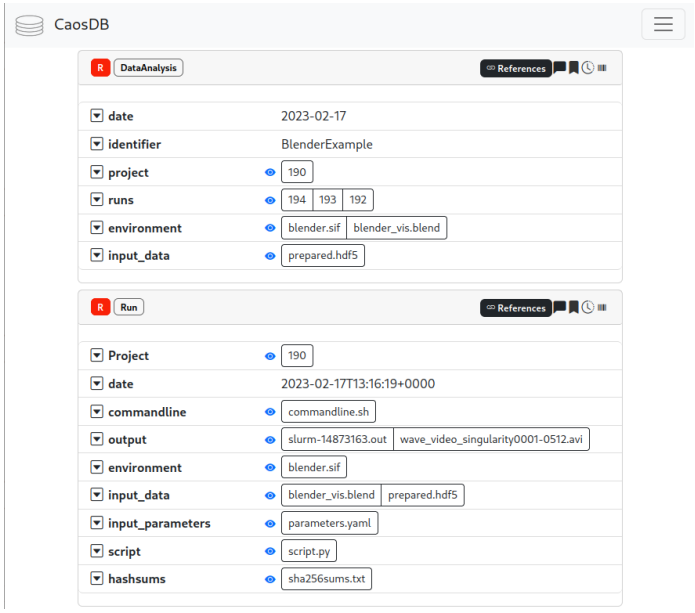
[1] A.S. is a co-founder of IndiScale.

Data Integration

```
example/
├ runs/
│   └ 2023-02-17T13_16/
│       ├ prepared.hdf5
│       ├ sha256sums.txt
│       ├ parameters.yaml
│       ├ commandline.sh
│       └ slurm-14873163.out
├ script.py
└ blender.sif
```

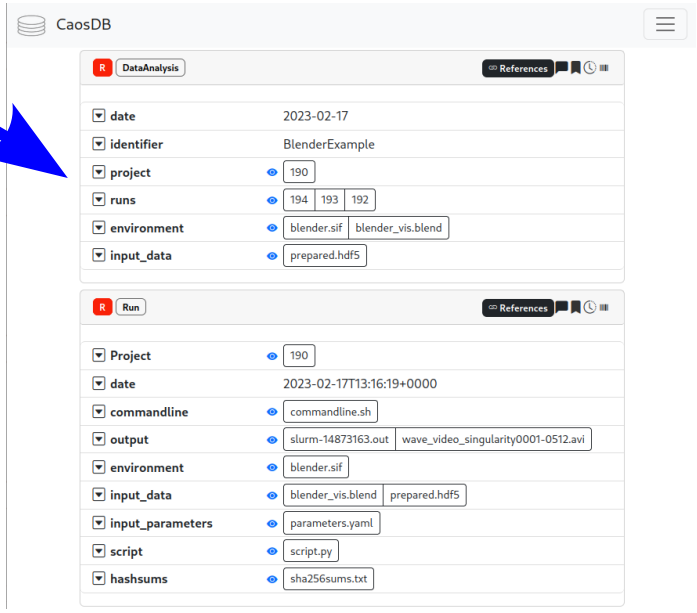
Data Integration

```
example/
├ runs/
│   └ 2023-02-17T13_16/
│       ├ prepared.hdf5
│       ├ sha256sums.txt
│       ├ parameters.yaml
│       ├ commandline.sh
│       └ slurm-14873163.out
├ script.py
└ blender.sif
```



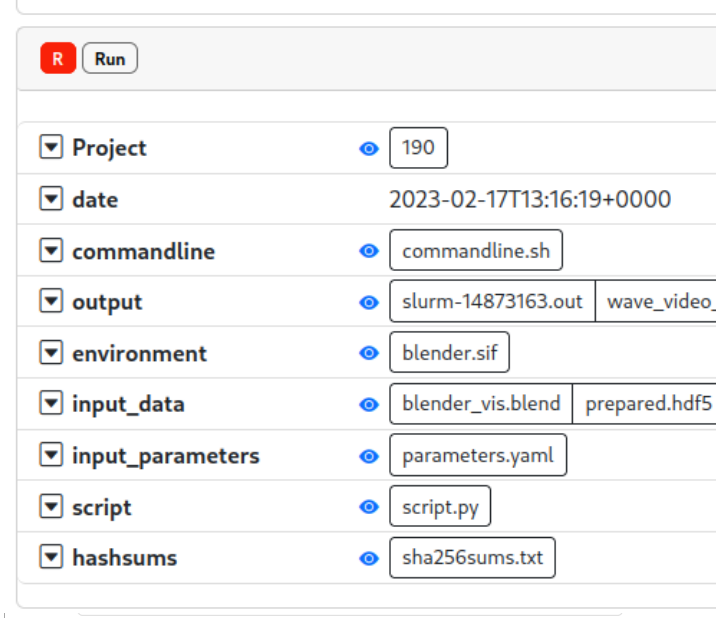
Data Integration

```
example/
├ runs/
│   └ 2023-02-17T13_16/
│       ├ prepared.hdf5
│       ├ sha256sums.txt
│       ├ parameters.yaml
│       ├ commandline.sh
│       └ slurm-14873163.out
├ script.py
└ blender.sif
```

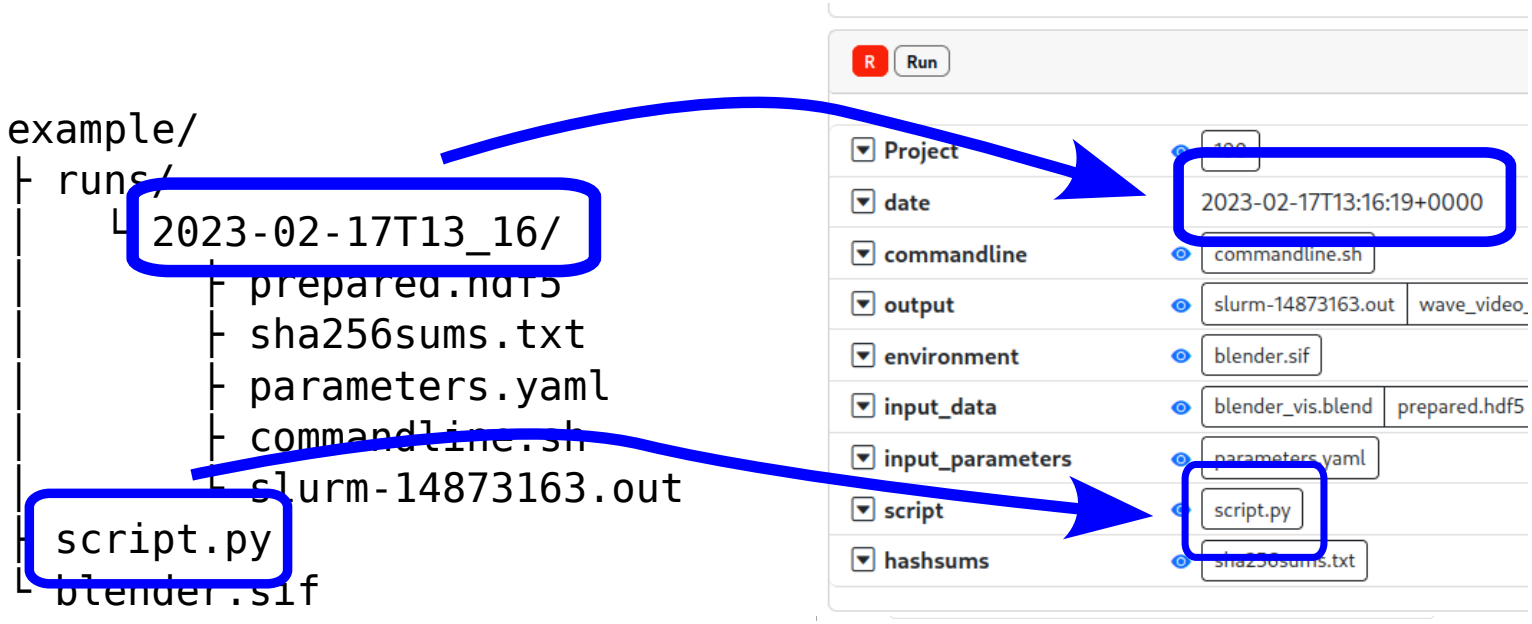


Data Integration

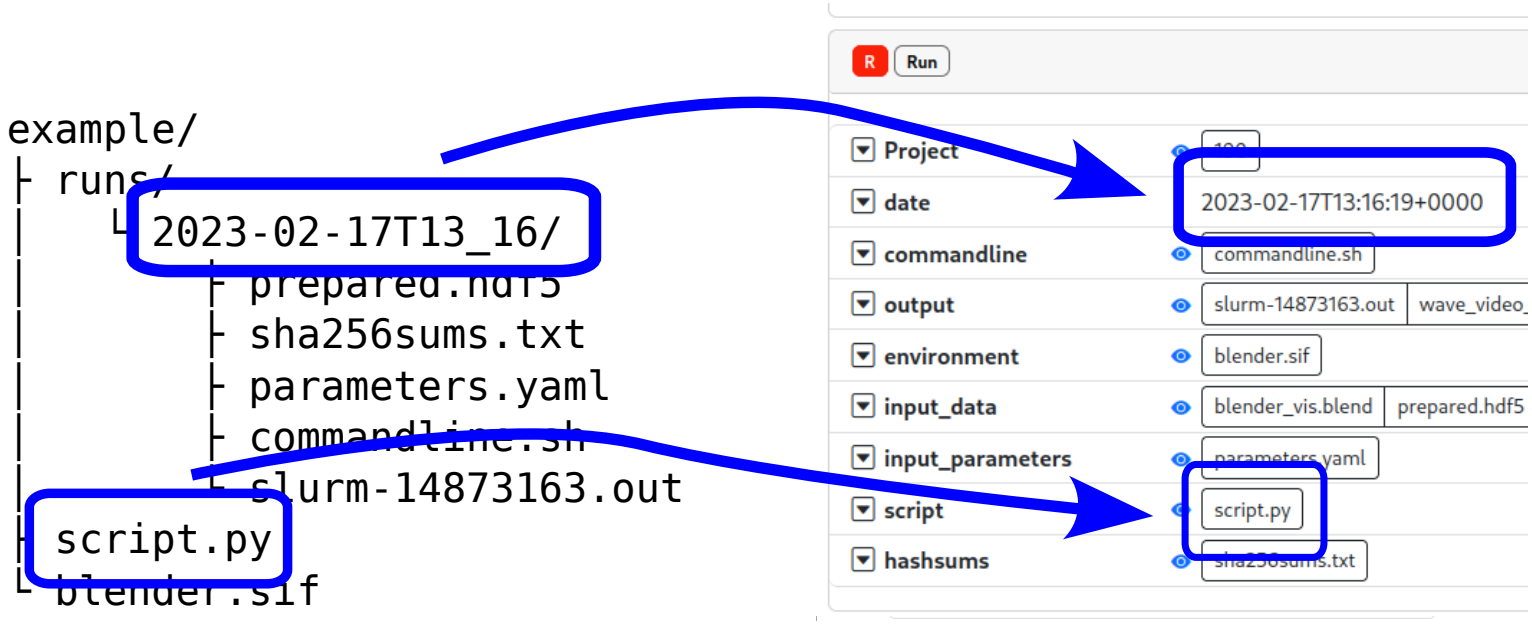
```
example/
├ runs/
│   └ 2023-02-17T13_16/
│       ├ prepared.hdf5
│       ├ sha256sums.txt
│       ├ parameters.yaml
│       ├ commandline.sh
│       └ slurm-14873163.out
├ script.py
└ blender.sif
```



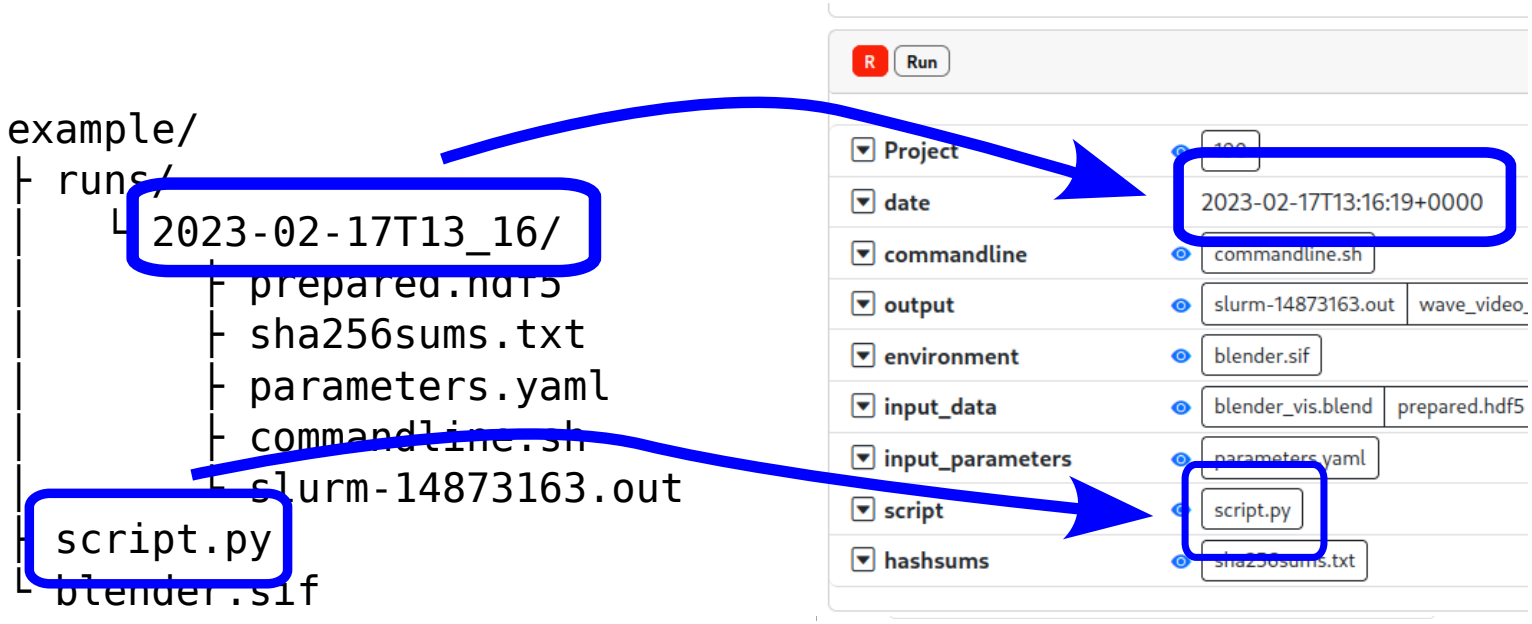
Data Integration



Data Integration

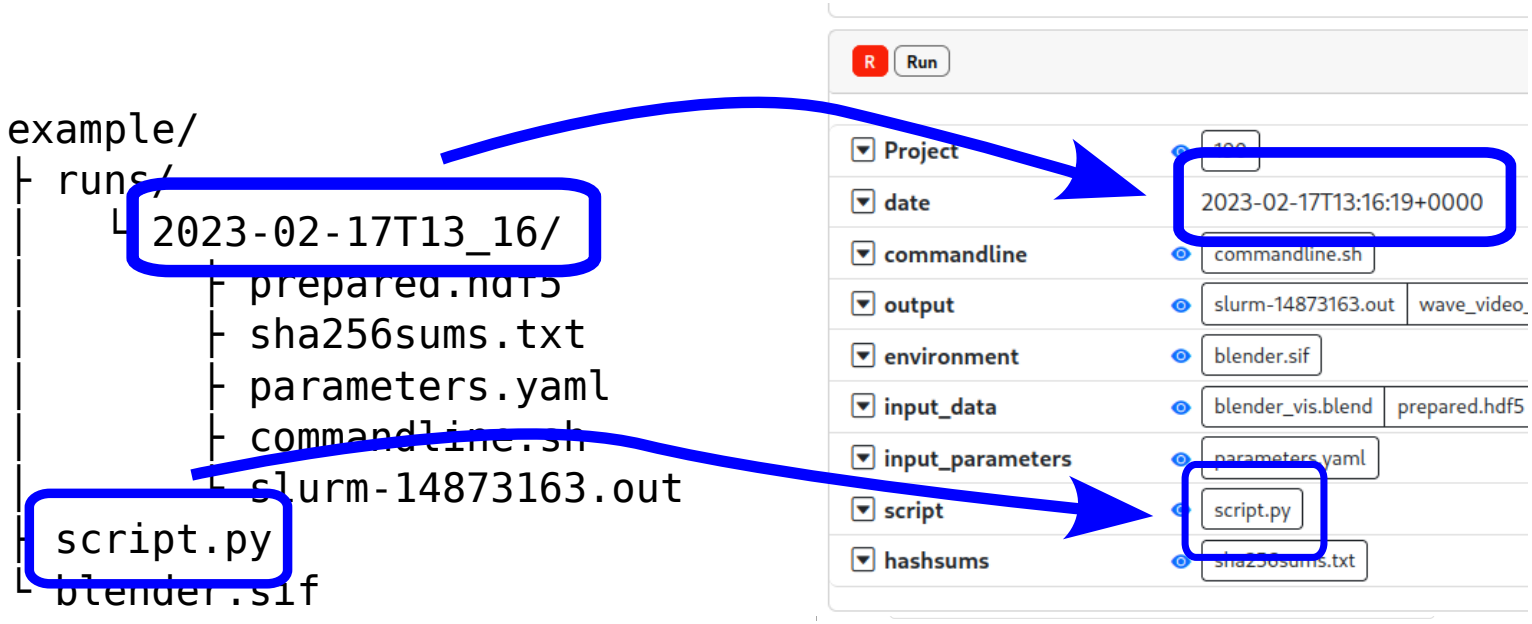


Data Integration



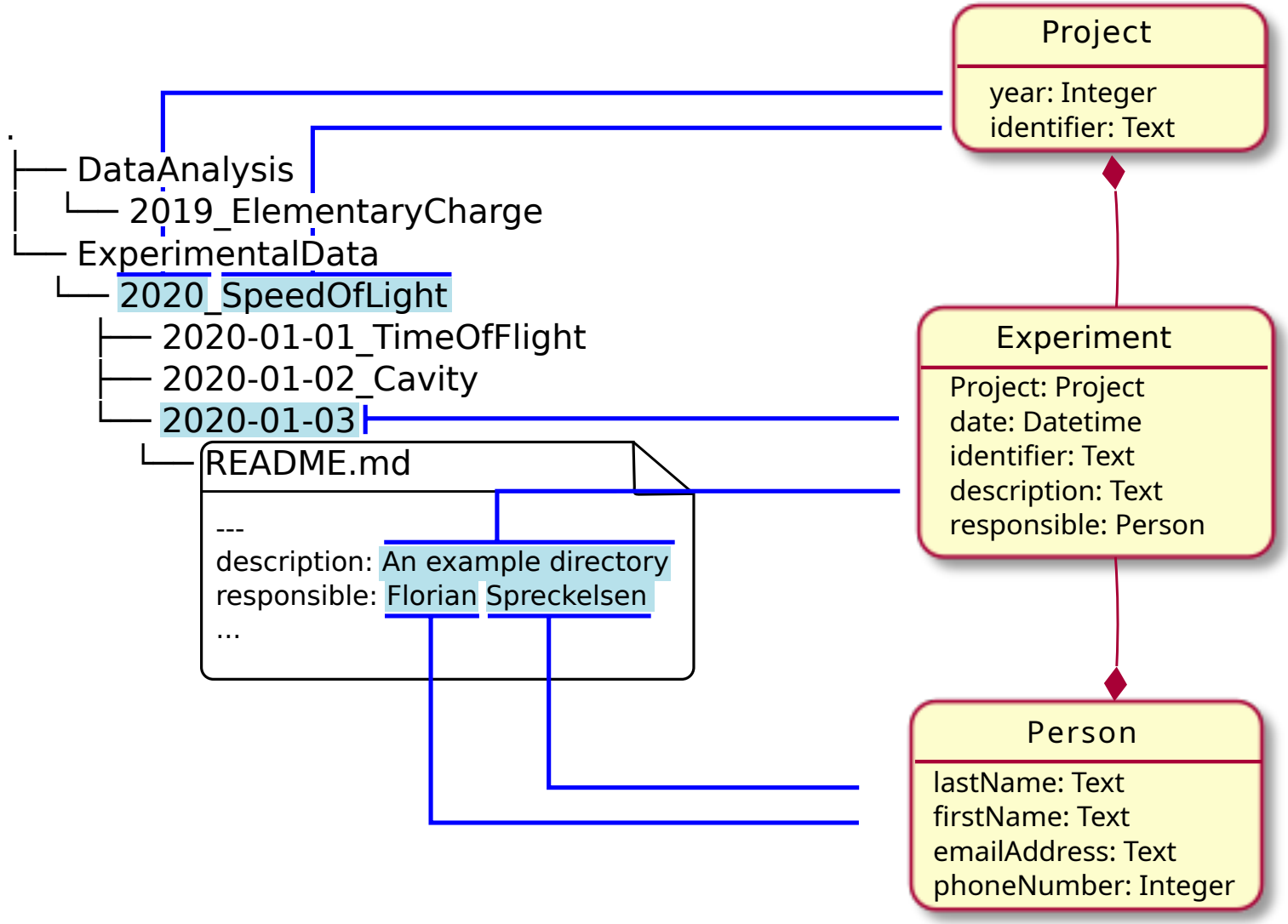
- Extract (meta) data from file names, directory names, and file contents

Data Integration



- Extract (meta) data from file names, directory names, and file contents
- Map these data onto a semantic data model (and store it in CaosDB/LinkAhead)

Example 2



tom Wörden et. al., *Data* 2024, 9, 24.
https://doi.org/10.3390/data9020024

Crawler Design

- Modular file crawler (Python, AGPLv3)

Crawler Design

- Modular file crawler (Python, AGPLv3)
- Adaptable to file structure using YAML specification

Crawler Design

- Modular file crawler (Python, AGPLv3)
- Adaptable to file structure using YAML specification
- Extendable with Python modules (custom data formats, . . .)

Crawler Design

- Modular file crawler (Python, AGPLv3)
- Adaptable to file structure using YAML specification
- Extendable with Python modules (custom data formats, . . .)
- Synchronization into RDMS (instead of data ingest): Files and RDMS can be used simultaneously!

YAML specification

```yaml
ExperimentalData_Dir:
  type: Directory
  match: "ExperimentalData"
  subtree:
    Project_Dir:
    type: Directory
    match: (?P<year>[0-9]{4,4})_(?P<name>.*)
    records:
      Project:
        year: $year
        name: $name
# (...)
```

YAML specification

```yaml
ExperimentalData_Dir:
  type: Directory
  match: "ExperimentalData"
  subtree:
    Project_Dir:
    type: Directory
    match: (?P<year>[0-9]{4,4})_(?P<name>.*)
    records:
      Project:
        year: $year
        name: $name
# (...)
```

Converter matching folders with name
ExperimentalData

YAML specification

```yaml
ExperimentalData_Dir:
  type: Directory
  match: "ExperimentalData"
  subtree:
    Project_Dir:
    type: Directory
    match: (?P<year>[0-9]{4,4})_(?P<name>.*)
    records:
      Project:
        year: $year
        name: $name
# (...)
```

Converter matching folders with name containing numbers

Converter for subfolders
using a regexp, e.g.: **2024_deRSE**

YAML specification

```yaml
ExperimentalData_Dir:
  type: Directory
  match: "ExperimentalData"
  subtree:
    Project_Dir:
    type: Directory
    match: (?P<year>[0-9]{4,4})_(?P<name>.*)
    records:
      Project:
        year: $year
        name: $name
# (...)
```

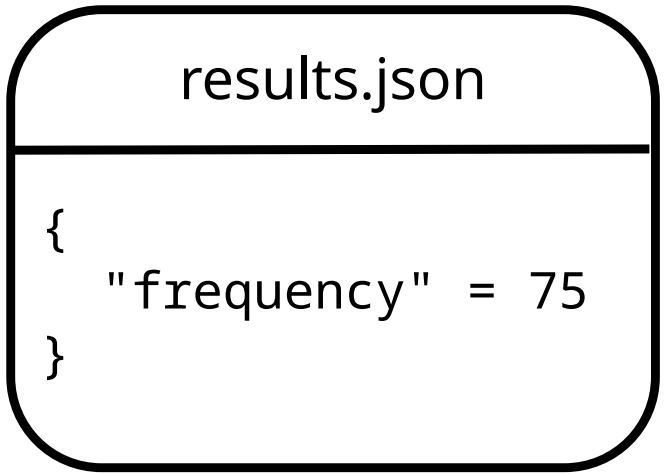
For each subfolder, a record of type **Project** is created, setting 2 properties **year** and **name** from the regexp.

Extract contents from files

```
# (...)
DataFile:
  type: JSONFile
  match: results\.json
  subtree:
    frequency:
      type: DictTextElement
      match_name: frequency
      match_value: (?P<frequency>[0-9]+)
      records:
        DataAnalysis:
          frequency: $frequency
# (...)
```

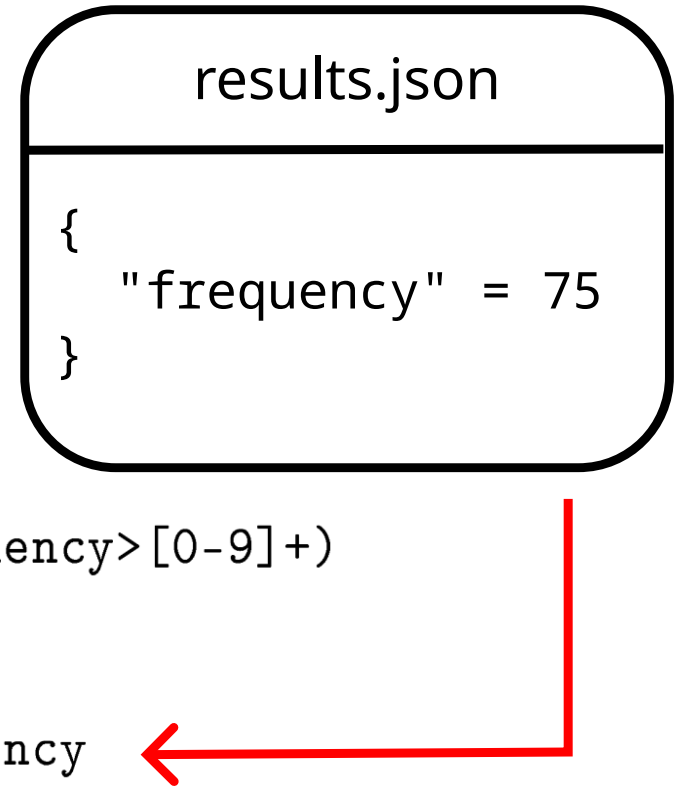
Extract contents from files

```
# (...)
DataFile:
  type: JSONFile
  match: results\.json
  subtree:
    frequency:
      type: DictTextElement
      match_name: frequency
      match_value: (?P<frequency>[0-9]+)
      records:
        DataAnalysis:
          frequency: $frequency
# (...)
```



Extract contents from files

```
# (...)
DataFile:
  type: JSONFile
  match: results\.json
  subtree:
    frequency:
      type: DictTextElement
      match_name: frequency
      match_value: (?P<frequency>[0-9]+)
      records:
        DataAnalysis:
          frequency: $frequency
# (...)
```



Create a record of type **DataAnalysis**
with property **frequency** = 75

Thank You! - Summary

- Crawler that continuously synchronizes (currently one-way) information from file system to semantic RDMS
- YAML based, extendable specification for adapting crawler to different file system layouts and data formats
- tom Wörden et. al., *Data* 2024, 9, 24. `https://doi.org/10.3390/data9020024`
- gitlab.com/linkahead AND gitlab.com/linkahead/linkahead-crawler