

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

Open Access Article

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

by  Henrik tom Wörden¹  ,  Florian Spreckelsen¹  ,  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: ChaosDB)

 ChaosDB 

 DataAnalysis    

☒ date	2023-02-17
☒ identifier	BlenderExample
☒ project	 190
☒ runs	 194 193 192
☒ environment	 blender.sif blender_vis.blend
☒ input_data	 prepared.hdf5

 Run    

☒ Project	 190
☒ date	2023-02-17T13:16:19+0000
☒ commandline	 commandline.sh
☒ output	 slurm-14873163.out wave_video_singularity0001-0512.avi
☒ environment	 blender.sif
☒ input_data	 blender_vis.blend prepared.hdf5
☒ input_parameters	 parameters.yaml
☒ script	 script.py
☒ hashsums	 sha256sums.txt



LinkAhead (formerly: ChaosDB)

 **LinkAhead**
~~ChaosDB~~

R

DataAnalysis

References

date	2023-02-17
identifier	BlenderExample
project	190
runs	194 193 192
environment	blender.sif blender_vis.blend
input_data	prepared.hdf5

R

Run

References

Project	190
date	2023-02-17T13:16:19+0000
commandline	commandline.sh
output	slurm-14873163.out wave_video_singularity0001-0512.avi
environment	blender.sif
input_data	blender_vis.blend prepared.hdf5
input_parameters	parameters.yaml
script	script.py
hashsums	sha256sums.txt



LinkAhead (formerly: CaosDB)

 **LinkAhead**
~~CaosDB~~

FIND DataAnalysis with date in 2023 and identifier like *Blender*

R

DataAnalysis

References

date	2023-02-17
identifier	BlenderExample
project	190
runs	194 193 192
environment	blender.sif blender_vis.blend
input_data	prepared.hdf5

R

Run

References

Project	190
date	2023-02-17T13:16:19+0000
commandline	commandline.sh
output	slurm-14873163.out wave_video_singularity0001-0512.avi
environment	blender.sif
input_data	blender_vis.blend prepared.hdf5
input_parameters	parameters.yaml
script	script.py
hashsums	sha256sums.txt

LinkAhead (formerly: CaosDB)

 ~~CaosDB~~ **LinkAhead**

FIND DataAnalysis with date in 2023 and identifier like *Blender*

DataAnalysis **References**   

☒ date	2023-02-17
☒ identifier	BlenderExample
☒ project	 190
☒ runs	 194 193 192
☒ environment	 blender.sif blender_vis.blend
☒ input_data	 prepared.hdf5

Run **References**   

☒ Project	 190
☒ date	2023-02-17T13:16:19+0000
☒ commandline	 commandline.sh
☒ output	 slurm-14873163.out wave_video_singularity0001-0512.avi
☒ environment	 blender.sif
☒ input_data	 blender_vis.blend prepared.hdf5
☒ input_parameters	 parameters.yaml
☒ script	 script.py
☒ hashsums	 sha256sums.txt

Navigation icons:       

LinkAhead (formerly: CaosDB)

LinkAhead
~~CaosDB~~

FIND Run which is referenced by
DataAnalysis with date in 2023 and identifier like *Blender*

DataAnalysis

date	2023-02-17
identifier	BlenderExample
project	190
runs	194 193 192
environment	blender.sif blender_vis.blend
input_data	prepared.hdf5

Run

References

Project	190
date	2023-02-17T13:16:19+0000
commandline	commandline.sh
output	slurm-14873163.out wave_video_singularity0001-0512.avi
environment	blender.sif
input_data	blender_vis.blend prepared.hdf5
input_parameters	parameters.yaml
script	script.py
hashsums	sha256sums.txt

LinkAhead (formerly: ChaosDB)

- ▶ 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 GmbH¹ (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

¹A.S. is a co-founder of IndiScale.

Data Integration

```
example/  
├ runs/  
│   └ 2023-02-17T13_16/  
│       ├── prepared.hdf5  
│       ├── sha256sums.txt  
│       ├── parameters.yaml  
│       ├── cmdline.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
```

The screenshot displays the CaosDB web interface. At the top, the 'CaosDB' logo is visible. Below it, there are two main panels. The first panel, titled 'DataAnalysis', shows metadata for a task. The second panel, titled 'Run', shows details for a specific execution of the task.

DataAnalysis Panel:

- date:** 2023-02-17
- identifier:** BlenderExample
- project:** 190
- runs:** 194, 193, 192
- environment:** blender.sif, blender_viz.blend
- input_data:** prepared.hdf5

Run Panel:

- Project:** 190
- date:** 2023-02-17T13:16:19+0000
- commandline:** commandline.sh
- output:** slurm-14873163.out, wave_video_singularity0001-0512.avi
- environment:** blender.sif
- input_data:** blender_viz.blend, prepared.hdf5
- input_parameters:** parameters.yaml
- script:** script.py
- hashsums:** sha256sums.txt

Data Integration

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

The screenshot displays the CaosDB web interface. A blue arrow points from the file tree on the left to the 'DataAnalysis' task entry. The interface shows two panels: 'DataAnalysis' and 'Run'.

DataAnalysis Panel:

- date: 2023-02-17
- identifier: BlenderExample
- project: 190
- runs: 194, 193, 192
- environment: blender:sl | blender_viz.blend
- input_data: prepared.hdf5

Run Panel:

- Project: 190
- date: 2023-02-17T13:16:19+0000
- commandline: commandline.sh
- output: slurm-14873163.out | wave_video_singularity0001-0512.avi
- environment: blender:sl
- input_data: blender_viz.blend | prepared.hdf5
- input_parameters: parameters.yaml
- script: script.py
- hashsums: sha256sums.txt

Data Integration

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

 Run	
Project	190
date	2023-02-17T13:16:19+0000
commandline	commandline.sh
output	slurm-14873163.out wave_video_
environment	blender.sif
input_data	blender_vis.blend prepared.hdf5
input_parameters	parameters.yaml
script	script.py
hashsums	sha256sums.txt

Data Integration

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

Run

Project	2023-02-17T13:16:19+0000
date	2023-02-17T13:16:19+0000
commandline	commandline.sh
output	slurm-14873163.out wave_video_
environment	blender.sif
input_data	blender_vis.blend prepared.hdf5
input_parameters	parameters.yaml
script	script.py
hashsums	sha256sums.txt

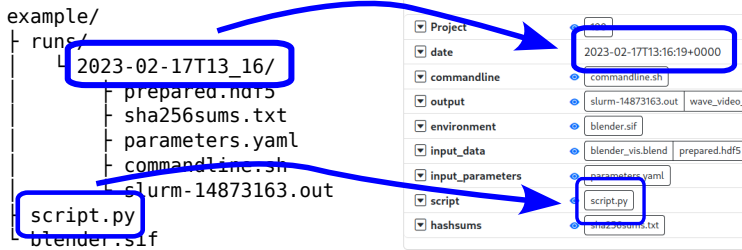
Data Integration

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

The screenshot shows a workflow configuration interface with a 'Run' button at the top. Below it, several fields are listed with expandable dropdown menus. Blue boxes and arrows highlight specific values: the 'Project' field is set to 'xop'; the 'date' field is set to '2023-02-17T13:16:19+0000'; the 'script' field is set to 'script.py'; and the 'input_parameters' field is set to 'parameters.yaml'. Other fields include 'commandline' (set to 'commandline.sh'), 'output' (set to 'slurm-14873163.out' and 'wave_video'), 'environment' (set to 'blender.sif'), 'input_data' (set to 'blender_vis.blend' and 'prepared.hdf5'), and 'hashsums' (set to 'sha256sums.txt').

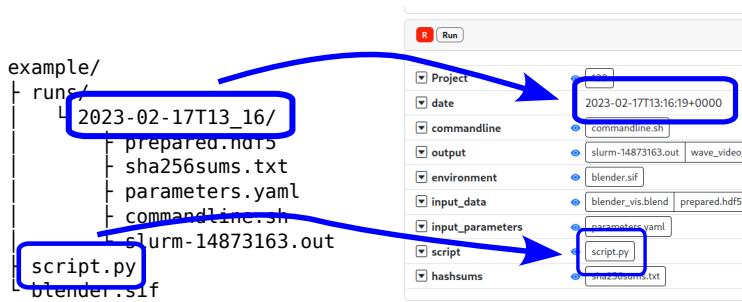
Field	Value
Project	xop
date	2023-02-17T13:16:19+0000
commandline	commandline.sh
output	slurm-14873163.out, wave_video
environment	blender.sif
input_data	blender_vis.blend, prepared.hdf5
input_parameters	parameters.yaml
script	script.py
hashsums	sha256sums.txt

Data Integration



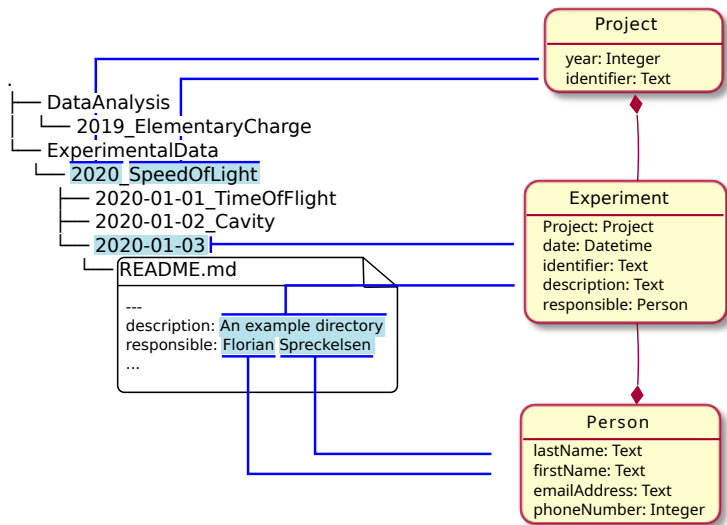
- ▶ Extract (meta) data from file names, directory names, and file contents

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

```
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

```
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

```
ExperimentalData_Dir:  
  type: Directory  
  match: "ExperimentalData"  
  subtree:
```

```
    Project_Dir:  
      type: Directory  
      match: (?P<year>[0-9]{4,4})_(?P<name>.*)  
      records:
```

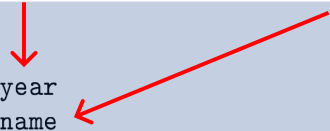
```
        Project:  
          year: $year  
          name: $name
```

```
# (...)
```

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

YAML specification

```
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
# (...)
```



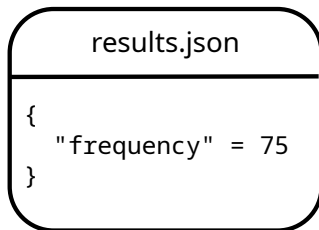
For each subfolder, a record of type **Project** is created, setting 2 properties **year** and **name** from the regexp.

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
# (...)
```

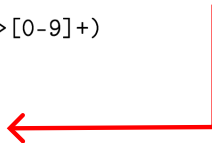
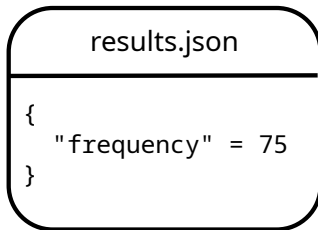
Extract contents from files

```
# (...)  
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