Linked (Open) Data for Microbial Population Biology

Carsten Fortmann-Grote[1]

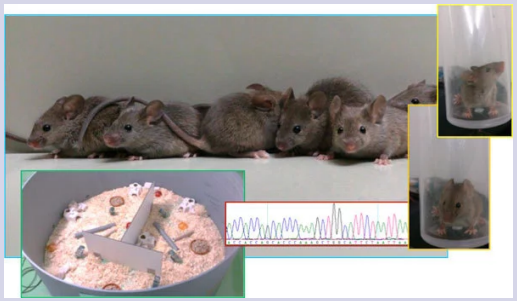
March 12 2024

[1]carsten.fortmann-grote@evolbio.mpg.de

A short tour of the institute

Evolutionary Genetics (D. Tautz (Emeritus))

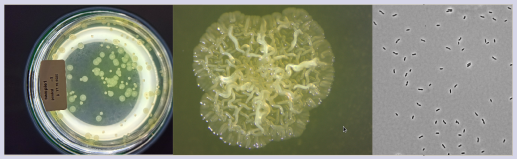
- Model organisms: Mus domesticus, mus musculus
- Behavioural genomics
- Population genetics



credits:
https://www.evolbio.mpg.de/3039130/group_evolanimalbehpers

Microbial Population Biology (P. Rainey)

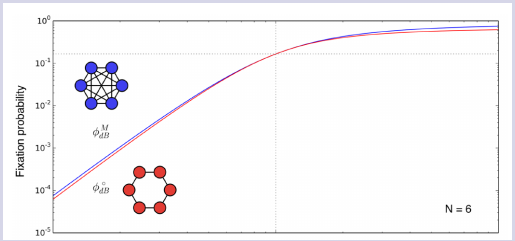
- Model organism: Pseudomonas fluorescens, Bacillus sub.
- Evolution of communities
- Host-microbe interactions
- Genetics



credits: Theodosiou (left), Schwarz (middle), Grote (right)

Theoretical Biology (A. Traulsen)

- Population Structure and Game Theory
- Metaorganisms
- Cancer Evolution



Hindersin et al, PLOS Comp. Bio (2019)
10.1371/journal.pcbi.1004437

Research topics in Microbial Population Biology

- Bacterial communities as Darwinian entities
- Antibiotica resistance
- Bacteria and phages
- Host-microbe interactions (plants, soil)
- Long term evolution experiments
- Adaptation to artificial selection pressures
- Genetic bases for evolutionary processes

A typical MPB project

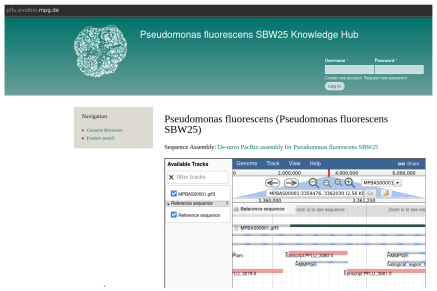
- Wildtype clone and/or genetically modified strain(s)
- Timelapse microscopy from growing microbial communities
- Time resolved whole genome / core gene NGS data
- Transcription profiles
- Optical density measurements from plaque assays
- Functional annotations

Need for integrated analysis of multiomics-multimodal data

"Find images, ELN entries for Δ mreB strains and annotations for mreB"

Integrating data the sparqling way

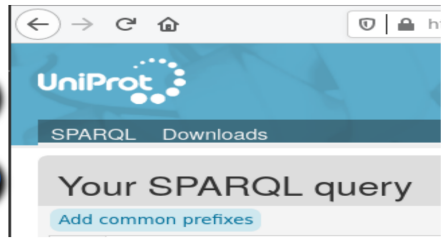
Pseudomonas fluorescens SBW25 genome database (Tripalv3)

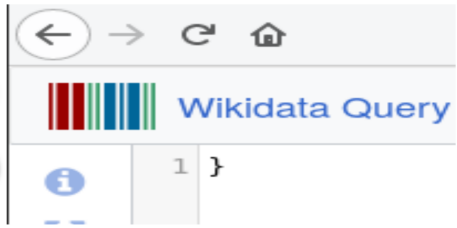


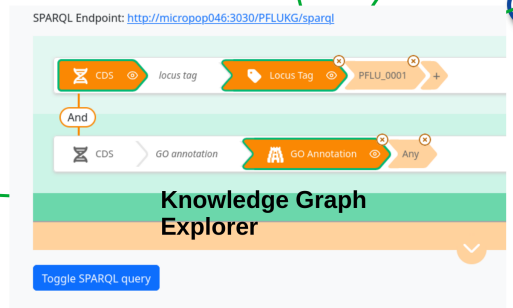
JSON API wrapper

RDF

Remote SPARQL endpoints







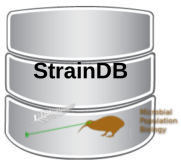
Internal DBs & web resources

API ???

RDF

omero-rdf

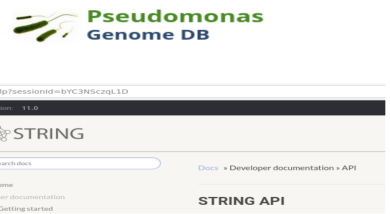
RDF



csv2rdf

RDF

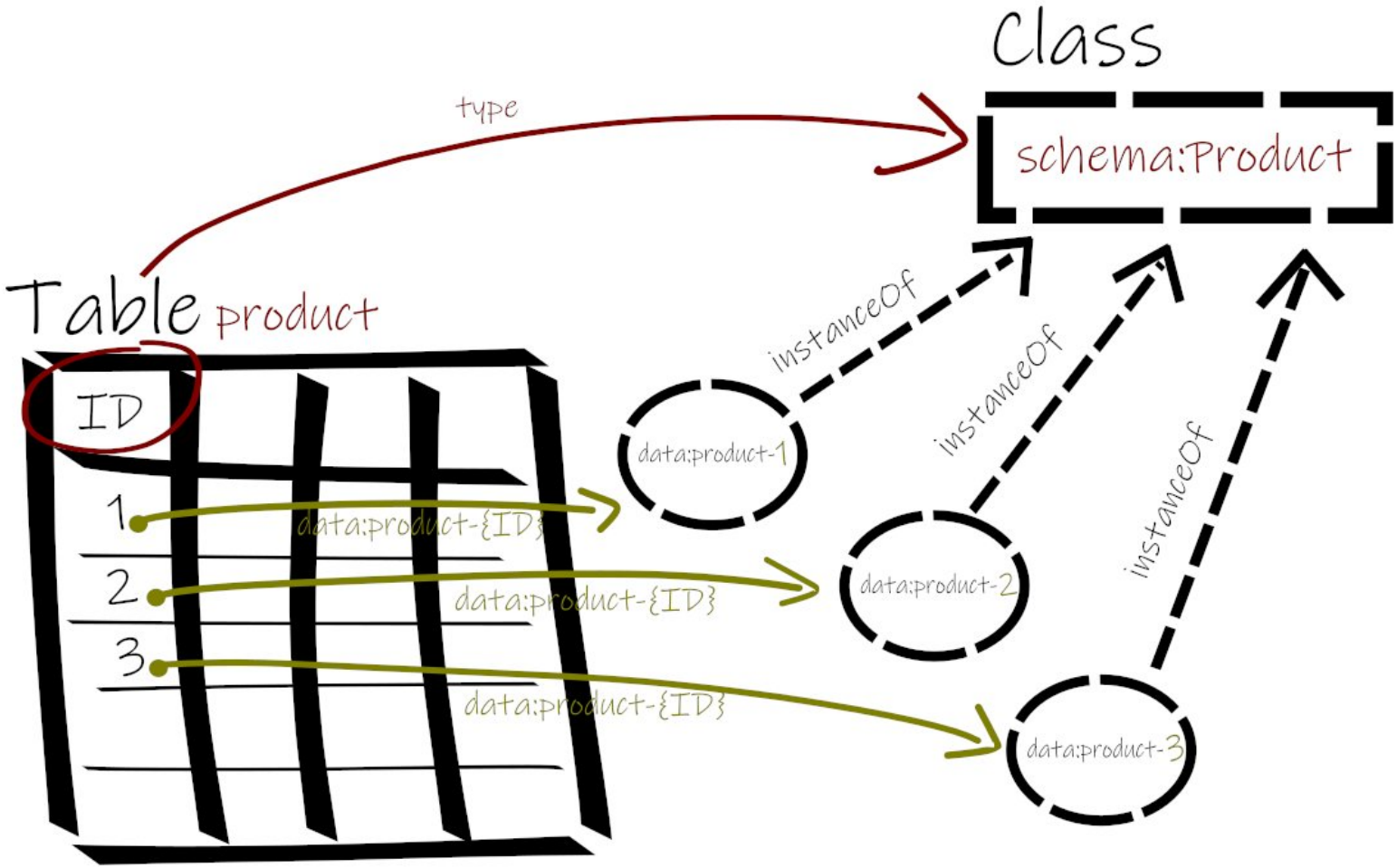
DBs with or without dedicated API



Data integration of internal sources

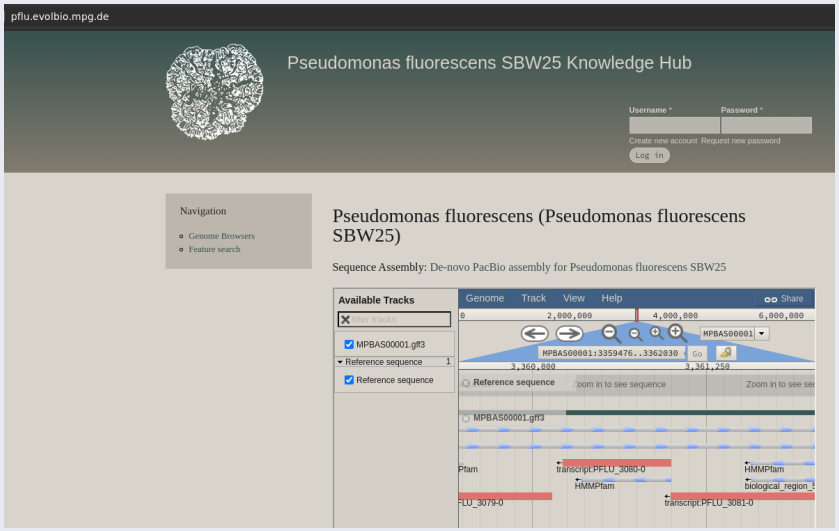
DB	API	LOD ready?	RDF conversion
Tripal Genome DB	JSON-LD	yes	SPARQL SPARQL-anything virtualization JSON to RDF serialization
OMERO	JSON-LD	yes	omero-rdf
OpenBIS	JSON	???	??? ("semantic annotation"
StrainDB	None	no	csv dump → csv2rdf

CSV2RDF


Class
type
schema:Product
Table product
ID
1
2
3
data:product-{ID}
data:product-{ID}
data:product-{ID}
instanceOf
instanceOf
instanceOf
data:product-1
data:product-2
data:product-3

Genome database

P.flu SBW25 Genome Database



Organism Database (Tripal)

- Based on Drupal CMS
- Implements Chado DB Scheme (SQL)
- Converts genome annotation into searchable, cross-linked database
- Integrates Genome Browser (JBrowse)
- Supports OBO Ontologies & Controlled Vocabularies (Sequence Ontology, Gene Ontology, . . .)
- JSON-LD API $\Rightarrow$ In principle ready for consumption by LOD

Querying a JSON-LD API

Direct query

```
SELECT distinct ?s ?p ?o where
  {
    ?s a SO:0000316 .
    ?s ?p ?o .
}
```

s	p	o
CDS:11845	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	so:0000316
CDS:11845	http://www.w3.org/2000/01/rdf-schema#label	CDS:PFLU_0001-0
CDS:11845	http://edamontology.org/data_2091	11845
CDS:11845	https://schema.org/name	CDS:PFLU_0001-0
CDS:11845	http://edamontology.org/data_0842	CDS:PFLU_0001-0
CDS:11845	http://purl.obolibrary.org/obo/OBI_0100026	Pseudomonas fluorescens
CDS:11845	http://www.w3.org/2000/01/rdf-schema#type	CDS
CDS:11845	http://edamontology.org/data_2012	CDS:11845/Sequence+coordinates
CDS:11845	http://purl.obolibrary.org/obo/OGI_0000021	CDS:11845/location+on+map
CDS:11845	http://purl.obolibrary.org/obo/SBO_0000374	CDS:11845/relationship
CDS:11845	http://purl.obolibrary.org/obo/SBO_0000554	CDS:11845/database+cross+reference
CDS:11845	http://semanticscience.org/resource/SIO_001166	CDS:11845/annotation

Object URIs are Graph URIs of linked json-ld documents. Exploring the graph across documents requires dynamic generation of graph URIs, not supported in SPARQL1.1

Querying a JSON-LD API (cont.)

SPARQL-Anything

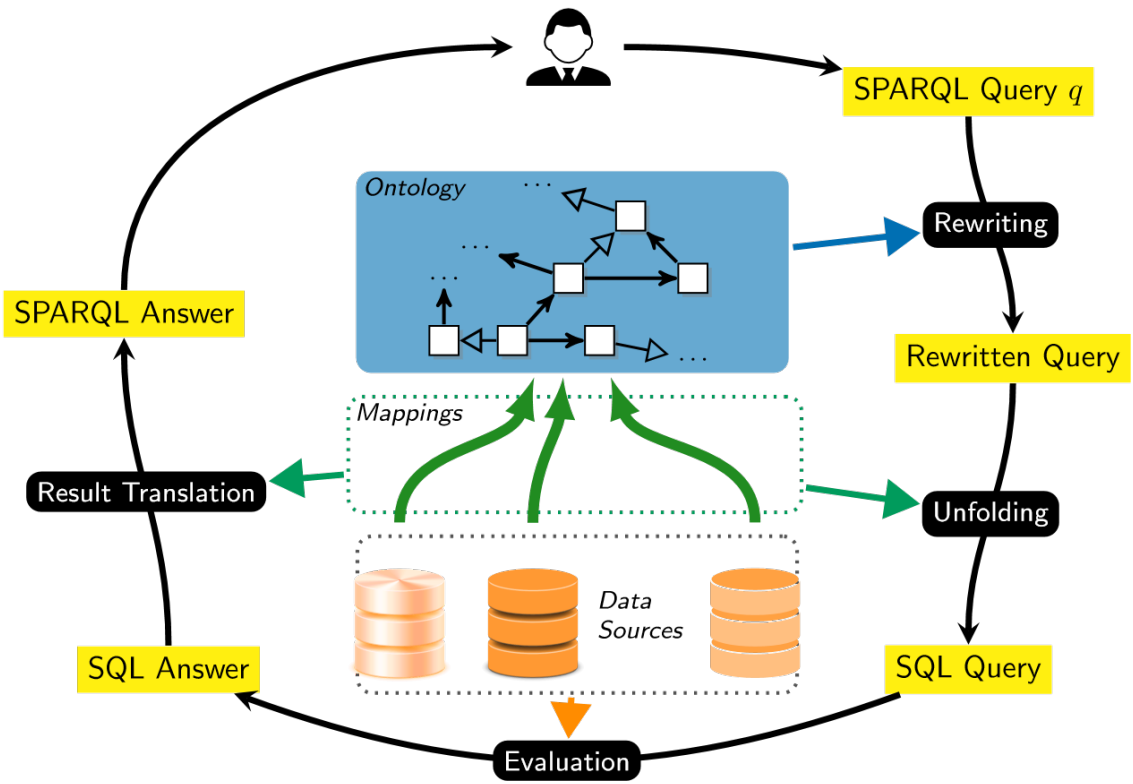
```
PREFIX fx: <http://sparql.xyz/facade-x/ns/>
PREFIX xyz: <http://sparql.xyz/facade-x/data/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT distinct ?gene_name ?go  where {
  service <x-sparql-anything:>
  {
    fx:properties fx:location "http://pflu.evolbio.mpg.de/web-services/content/v0.1/CDS/12135";
                  fx:media-type "application/ld+json" .
    ?s a <http://www.sequenceontology.org/browser/current_svn/term/SO:0000316> .
    ?gs <http://pflu.evolbio.mpg.de/cv/lookup/local/gene> ?gene_name .
    ?gs <http://semanticscience.org/resource/SIO_001166> ?annotation .
  }
optional {
  service <x-sparql-anything:>
  {
    fx:properties fx:location ?annotation ;
    fx:media-type "application/ld+json" .
    ?annotation_s <http://semanticscience.org/resource/SIO_001080> ?vocab ;
                  <http://edamontology.org/data_2091> ?go_term ;
                  <https://schema.org/name> ?go_name ;
                  <http://purl.obolibrary.org/obo/IAO_0000115> ?definition .
    bind(concat(concat(?vocab, ":"), ?go_term) as ?go)
  }
}
}
```

Querying a JSON-LD API

gene_name	go	go_name
wssH	GO:0043806	keto acid formate lyase activity
wssH	GO:0005886	plasma membrane
wssH	GO:0016021	integral component of membrane
wssH	GO:0016746	acyltransferase activity
wssH	GO:0042121	alginic acid biosynthetic process

Still, this is quite complicated and probably not well supported by SPARQL query editors

Virtualization



Xiao et al (2019)

Example: VKG for the P.flu SBW25 Genome database

Step 1: Bootstrapping with ontop-vkg

```
$> ontop bootstrap --db-url URL -p PROPS -t ONTOLOGY -m MAPPING

<http://www.semanticweb.org/grotec/ontologies/2022/5/tripalv3/gene#dbxref_id> rdf:type owl:DatatypeProperty .
: mappingId MAPPING-ID793
: target _:ontop-bnode-534{feature_id}/{dbxref_id}/{organism_id}/{name}/{uniquename}/{residues}/{seqlen}/{md5ch
g:protein_coding_gene#feature_id {feature_id}^^xsd:integer ;
g:protein_coding_gene#dbxref_id {dbxref_id}^^xsd:integer ;
g:protein_coding_gene#organism_id {organism_id}^^xsd:integer ;
g:protein_coding_gene#name {name}^^xsd:string ;
g:protein_coding_gene#uniquename {uniquename}^^xsd:string ;
g:protein_coding_gene#residues {residues}^^xsd:string ;
g:protein_coding_gene#seqlen {seqlen}^^xsd:integer ;
g:protein_coding_gene#md5checksum {md5checksum}^^xsd:string ;
g:protein_coding_gene#type_id {type_id}^^xsd:integer ;
g:protein_coding_gene#is_analysis {is_analysis}^^xsd:boolean ;
g:protein_coding_gene#is_obsolete {is_obsolete}^^xsd:boolean ;
g:protein_coding_gene#timeaccessioned {timeaccessioned}^^xsd:dateTime ;
g:protein_coding_gene#timelastmodified {timelastmodified}^^xsd:dateTime .
: source SELECT * FROM "chado"."protein_coding_gene"
```

Example: VKG for the P.flu SBW25 Genome database

Step 2: Cleanup

- Highly redundant ontology: Every column label is translated into a data property for every table it occurs in

```
<tripalv3:gene#dbxref_id> rdf:type owl:DatatypeProperty .
<tripalv3:non_protein_coding#dbxref_id> rdf:type owl:DatatypePr
```

- Define single property to be applied across all tables.
- Virtualization eventually worked but query execution time is slooooooooooooooowwwwww

Flattening a deeply nested JSON-LD graph

[12]:

Client

Client-73a06e67-d796-11ee-8cd8-e454e86b0546

Connection method: Direct

Dashboard: http://172.16.5.46:42575/status

Launch dashboard in JupyterLab

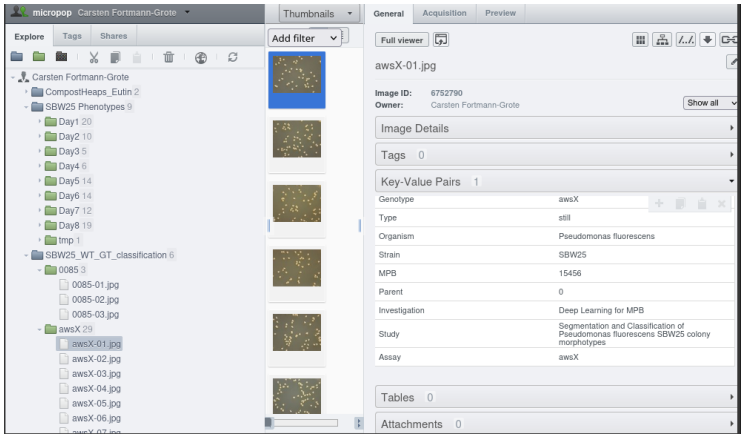
Scheduler Info

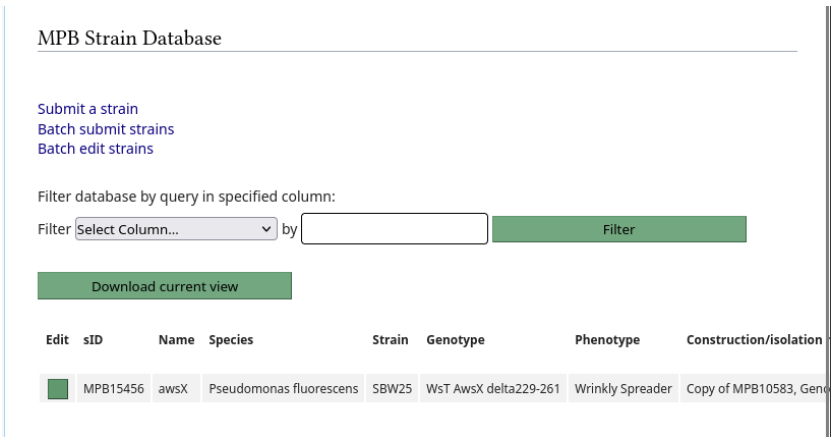
[27]:

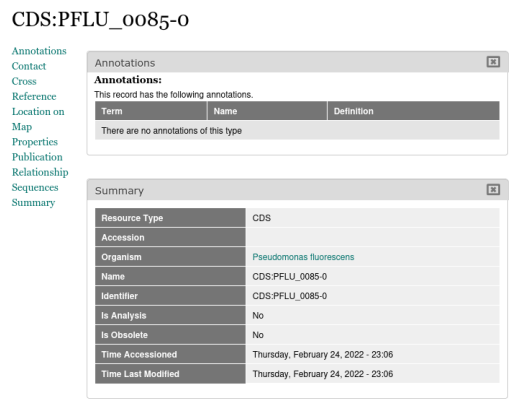
```python
def get_features(label, serialize=False, distribute=False):
    uri = URIRef('http://pflu.evolbio.mpg.de/web-services/content/v0.1/{}'.format(label))

    logger.info("Getting %s.", uri)
    graph = get_features_graph(uri)
    logger.info("Received %d terms.", len(graph))

    so_id = query_so_id(label)
    so_uri = SO.term(so_id)

    logger.info("SO term for %s is %s", label, so_id)

    subjects =   [s for s in graph.subjects(predicate=rdfs_type,
                        object=so_uri,
                        unique=True)]

    logger.info("Preparing to get %d %s features.", len(subjects), label)

    if distribute:
        logger.info("Running distributed jobs.")
        for g in client.gather([client.submit(get_feature_graph, s) for s in subjects]):
            graph += g
        logger.info("Distributed jobs ended.")
    else:
        for s in tqdm(subjects):
            graph += get_feature_graph(s)

    logger.info("Distributed job finished. Setting bindings and clean up.")
    set_bindings(graph)
    cleanup(graph)

    if serialize:
        fname = f"{label}_{today}.ttl"
        graph.serialize(fname)
        logger.info("Serialized graph to %s.", fname)

    return graph
```

- rdflib for loading json-ld into graph structure
- iteratively parses linked graph uris
- dask for distributed computing
- serialized to turtle format
- -> 2.1 million Triples in ca. 12hrs

Our P.flu SBW25 RDF playground







Image, genotype, mpb number and straindb columns

```
prefix ome_core: <http://www.openmicroscopy.org/rdf/2016-06/ome_core/>
prefix mpimap: <http://ome.evolbio.mpg.de/MapAnnotation/>
prefix mpiimg: <http://ome.evolbio.mpg.de/Image/>
prefix mpi: <http://ome.evolbio.mpg.de/>
prefix strains: <http://raineylab.evolbio.mpg.de/straindb/strains/>
prefix props: <http://raineylab.evolbio.mpg.de/straindb/properties/>
  select distinct ?img ?mpbn ?genotype ?phenotype where {
    ?img <http://www.wikidata.org/prop/direct/P180> ?ann .
    ?ann ome_core:Map ?map .
    ?map ome_core:Key ?key .
    ?map ome_core:Value ?mpb .
    filter(?key in ("", "MPB")).

    bind(concat("MPB", ?mpb) as ?mpbn ).
  service  <http://micropop046.evolbio.mpg.de:3030/MPBStrainDB/sparql> {
      ?strain props:sId ?mpbn .
      ?strain props:genotype ?genotype .
      ?strain props:phenotype ?phenotype .
  }
        }
order by ?img
limit 7
```

img	mpbn	genotype	phenotype
http://ome.evolbio.mpg.de/Image/6752778	MPB15447	PFLU0085 del 1309-1374 (del 437-458)	Wrinkly Spr
http://ome.evolbio.mpg.de/Image/6752779	MPB15447	PFLU0085 del 1309-1374 (del 437-458)	Wrinkly Spr
http://ome.evolbio.mpg.de/Image/6752780	MPB15447	PFLU0085 del 1309-1374 (del 437-458)	Wrinkly Spr
http://ome.evolbio.mpg.de/Image/6752781	MPB15456	WsT AwsX delta229-261	Wrinkly Spr
http://ome.evolbio.mpg.de/Image/6752782	MPB15456	WsT AwsX delta229-261	Wrinkly Spr
http://ome.evolbio.mpg.de/Image/6752783	MPB15456	WsT AwsX delta229-261	Wrinkly Spr
https://ome.evolbio.mpg.de/Image/6752855	MPB15455	"22A1" WspF 150bpdel (delta 677-826)	Wrinkly Spr

SparNatural for visual query generation

`file:///home/grotec/Sandbox/pflukg/index.html`

- Graphical query editor
- Supports federation and customized data sources
- No support (yet) for BIND and aggregations

On omero-rdf

```
select distinct ?key ?val where {
  # ?ann <http://www.wikidata.org/prop/direct/P180> ?img .
  ?ann ome_core:Map ?map .
  ?map ome_core:Key ?key .
  ?map ome_core:Value ?val .
  }
```

key	val
Type	still
Organism	Pseudomonas fluorescens
Strain	SBW25
MPB	15447
Parent	0
Investigation	Deep Learning for MPB
Study	Segmentation and Classif. . . 5 colony morphotype
Assay	0085
Phenotype	WS
Genotype	0085

- omero-rdf stores key, value as bnode property values.
- would need key as property, value as property value
- requires loading ontologies, controlled vocabs into omero and exposure in map annotations.

Tasks and contributions to NFDI4BI

- P.flu SBW25 Knowledge Graph as a Use Case
- Controlled vocabularies for microbial population biology and evolutionary cell biology
- Contributions to linked open data software, clients, recommendations