Semantic Data Lake

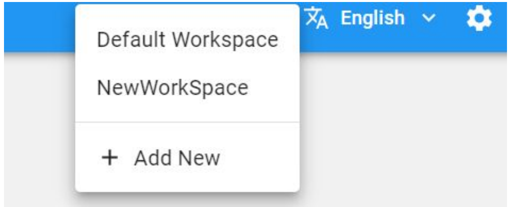
- Members:
  1. Sayed Hoseini
  2. Tobias Claas
  3. Maher Fallouh
  4. Abdullah Zaid
  5. Muhammad Noman

Semantic Data Integration - Summer

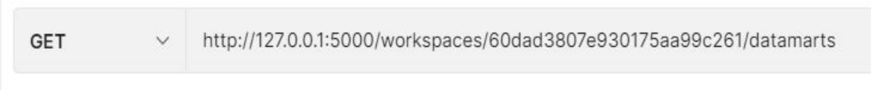
Main Contributions

- Logical Separation of Data by Workspace
- Property or Attribute is now standard Ontology
- Annotation API
  - Backend done
- Auto Completion
  - Backend done
- Ontology Management with Fuseki
- WorkFlow
- New FrontEnd written in TS/React

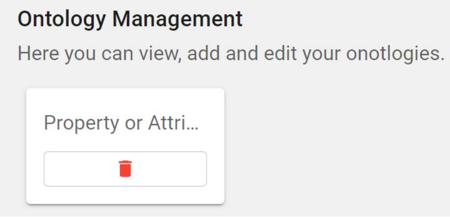
Workspace



Datamarts are separated by WorkSpaceID:

- One Fuseki, Hadoop, MongoDB, Postgres database for every workspace
- API URL: 

  GET http://127.0.0.1:5000/workspaces/60dad3807e930175aa99c261/datamarts
- Each Workspace has the NCIT-Property-or-Attribute-Ontology by default

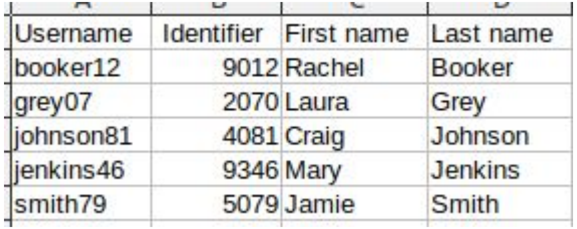


Ontology Management

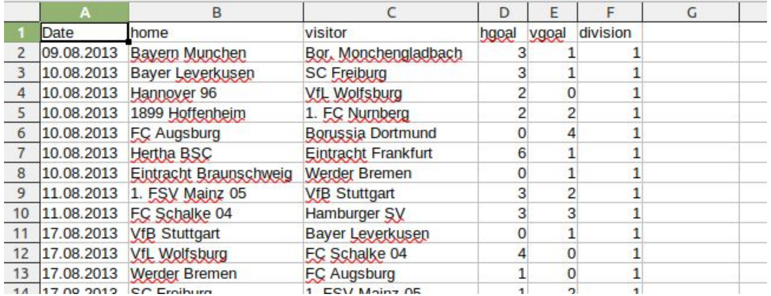
Here you can view, add and edit your onotlogies.

Annotations in MongoDB

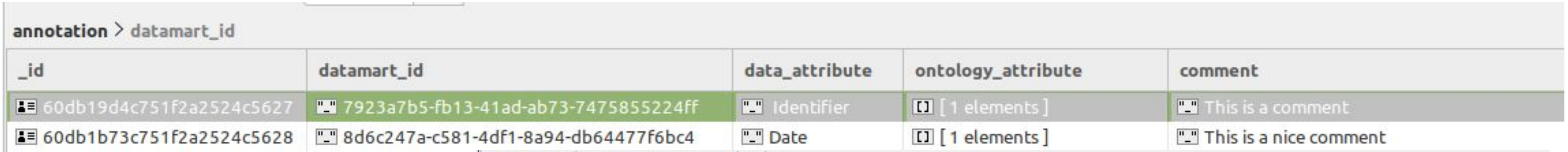
- Annotation uniquely identified by: datamart_id, data_attribute
  - Ontology_attribute is a list of tuples [(descr.,ontology_attribute),...]
  - Integrated feedback of last time
- Integrity checks on datamart_id, datamart-columns and ontology-attribute
- Example:



Username	Identifier	First name	Last name
booker12	9012	Rachel	Booker
grey07	2070	Laura	Grey
johnson81	4081	Craig	Johnson
jenkins46	9346	Mary	Jenkins
smith79	5079	Jamie	Smith

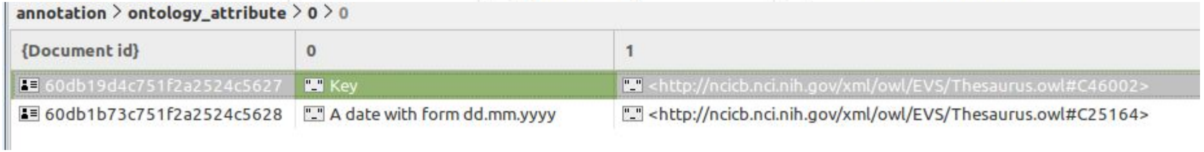


	A	B	C	D	E	F	G
1	Date	home	visitor	hgoal	vgoal	division	
2	09.08.2013	Bayern Munchen	Bor. Monchengladbach	3	1	1	
3	10.08.2013	Bayer Leverkusen	SC Freiburg	3	1	1	
4	10.08.2013	Hannover 96	VfL Wolfsburg	2	0	1	
5	10.08.2013	1899 Hoffenheim	1. FC Nurnberg	2	2	1	
6	10.08.2013	FC Augsburg	Borussia Dortmund	0	4	1	
7	10.08.2013	Hertha BSC	Eintracht Frankfurt	6	1	1	
8	10.08.2013	Eintracht Braunschweig	Werder Bremen	0	1	1	
9	11.08.2013	1. FSV Mainz 05	VfB Stuttgart	3	2	1	
10	11.08.2013	FC Schalke 04	Hamburger SV	3	3	1	
11	17.08.2013	VfB Stuttgart	Bayer Leverkusen	0	1	1	
12	17.08.2013	VfL Wolfsburg	FC Schalke 04	4	0	1	
13	17.08.2013	Werder Bremen	FC Augsburg	1	0	1	
14	17.08.2013	SC Freiburg	1. FSV Mainz 05	1	2	1	



annotation > datamart_id

_id	datamart_id	data_attribute	ontology_attribute	comment
60db19d4c751f2a2524c5627	7923a7b5-fb13-41ad-ab73-7475855224ff	Identifier	[1 elements]	This is a comment
60db1b73c751f2a2524c5628	8d6c247a-c581-4df1-8a94-db64477f6bc4	Date	[1 elements]	This is a nice comment

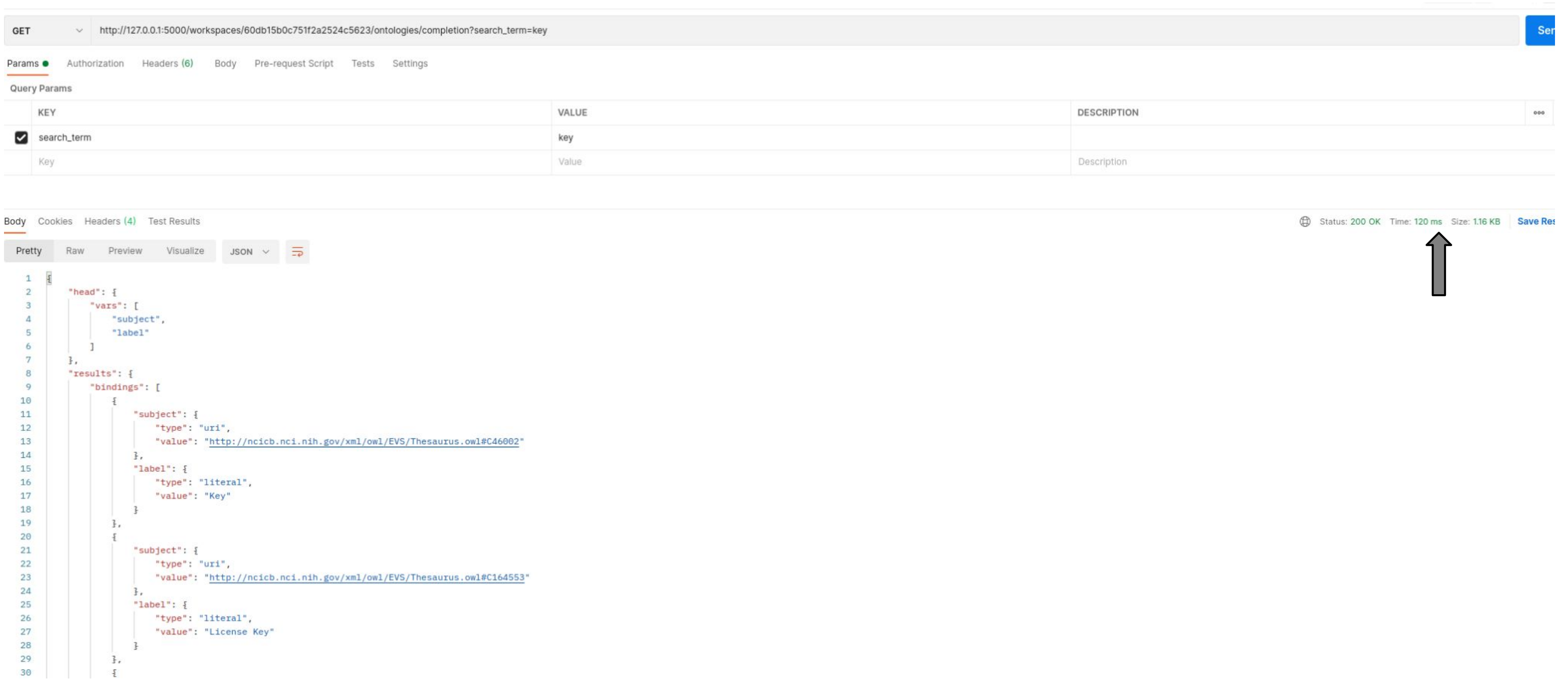


annotation > ontology_attribute > 0 > 0

{Document id}	0	1
60db19d4c751f2a2524c5627	Key	<http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C46002>
60db1b73c751f2a2524c5628	A date with form dd.mm.yyyy	<http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C25164>

Auto Completion

- Search for ontology-attributes
  - Search for Class in Ontology or it’s Label



GET http://127.0.0.1:5000/workspaces/60db15b0c751f2a2524c5623/ontologies/completion?search_term=key

Query Params

	KEY	VALUE	DESCRIPTION
☑	search_term	key	
	Key	Value	Description

Status: 200 OK Time: 120 ms Size: 1.16 KB

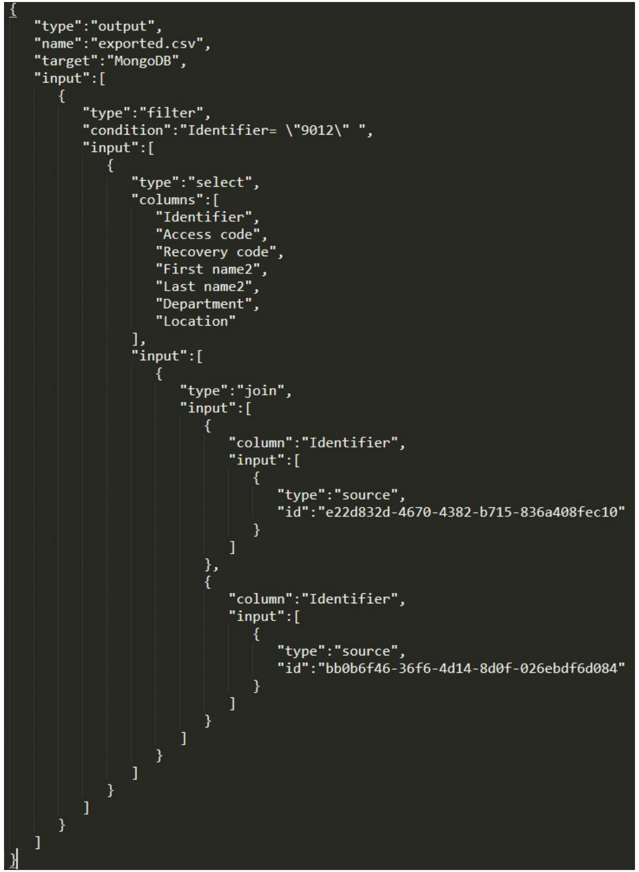
```
{
    "head": {
        "vars": [
            "subject",
            "label"
        ]
    },
    "results": {
        "bindings": [
            {
                "subject": {
                    "type": "uri",
                    "value": "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C46002"
                },
                "label": {
                    "type": "literal",
                    "value": "Key"
                }
            },
            {
                "subject": {
                    "type": "uri",
                    "value": "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C164553"
                },
                "label": {
                    "type": "literal",
                    "value": "License Key"
                }
            },
            {
```

Semantic Data Integration - Summer

WorkFlow Diagram

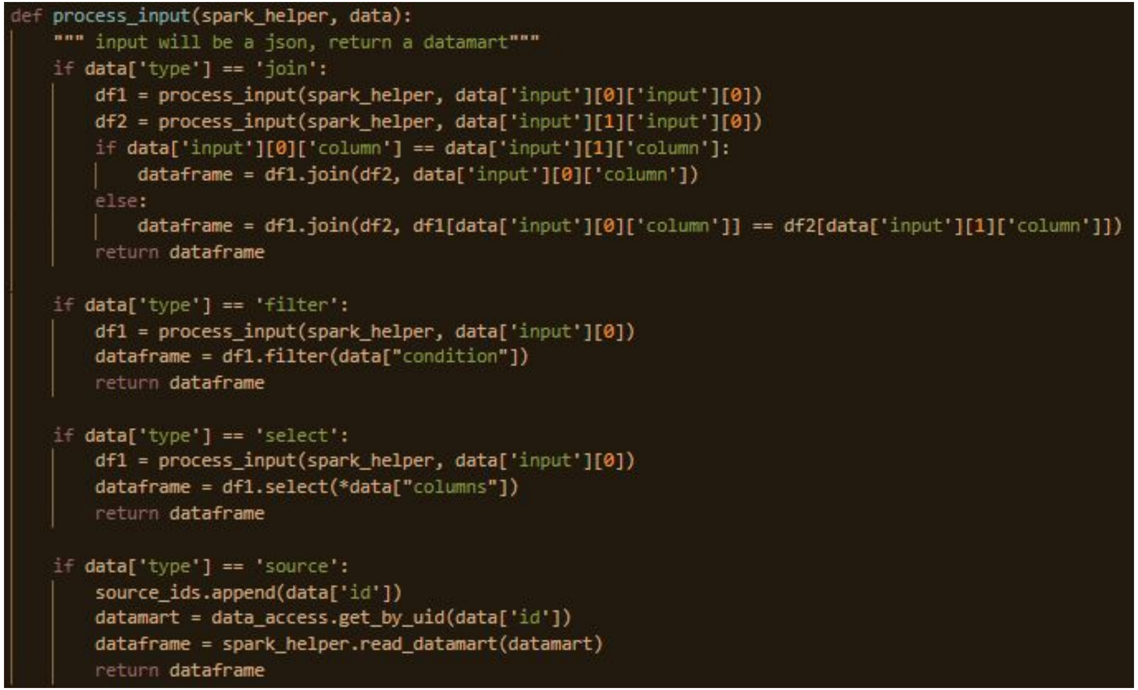
- Agreed on a Nested JSON as representation of the Workflow Diagram with ReactFlow, set up an API for the incoming JSON
- Written a recursive function to process incoming JSON
- WorkFlow is executed in a single Spark Session
- Currently supported operations are ‘Select Data Source’, ‘Filter’, ‘Join’, ‘Select Columns’, ‘Export’
- Chained Transformations possible
- Extendable to more operations

Nested JSON representation of the chained Data Transformation defined in Workflow



```
{
    "type":"output",
    "name":"exported.csv",
    "target":"MongoDB",
    "input":[
        {
            "type":"filter",
            "condition":"Identifier= \"9012\" ",
            "input":[
                {
                    "type":"select",
                    "columns":[
                        "Identifier",
                        "Access code",
                        "Recovery code",
                        "First name2",
                        "Last name2",
                        "Department",
                        "Location"
                    ],
                    "input":[
                        {
                            "type":"join",
                            "input":[
                                {
                                    "column":"Identifier",
                                    "input":[
                                        {
                                            "type":"source",
                                            "id":"e22d832d-4670-4382-b715-836a408fec10"
                                        }
                                    ]
                                },
                                {
                                    "column":"Identifier",
                                    "input":[
                                        {
                                            "type":"source",
                                            "id":"bb0b6f46-36f6-4d14-8d0f-026ebdf6d084"
                                        }
                                    ]
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

Recursive function in backend to process Spark dataframes

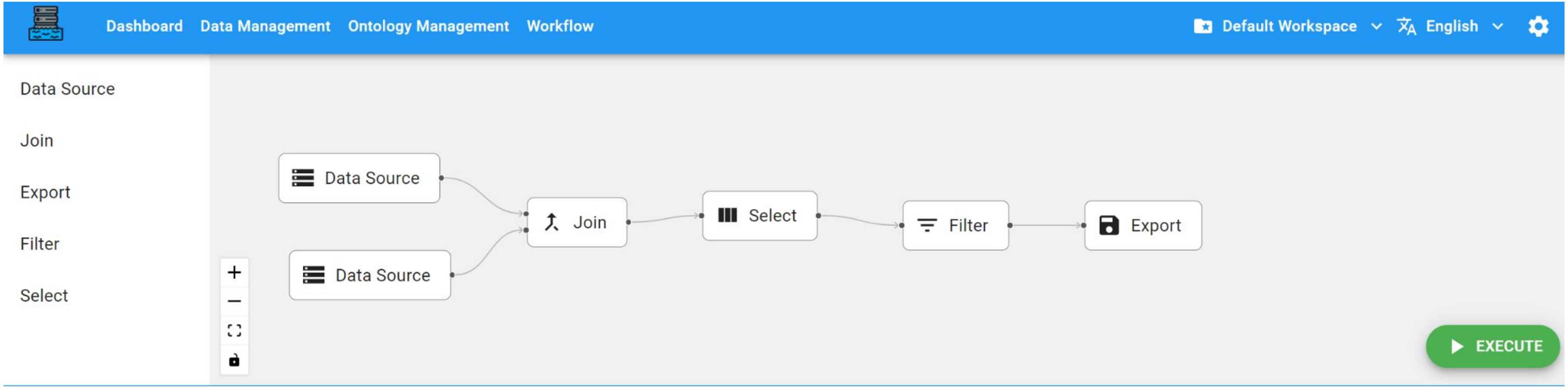


```python
def process_input(spark_helper, data):
    """ input will be a json, return a datamart"""
    if data['type'] == 'join':
        df1 = process_input(spark_helper, data['input'][0]['input'][0])
        df2 = process_input(spark_helper, data['input'][1]['input'][0])
        if data['input'][0]['column'] == data['input'][1]['column']:
            dataframe = df1.join(df2, data['input'][0]['column'])
        else:
            dataframe = df1.join(df2, df1[data['input'][0]['column']] == df2[data['input'][1]['column']])
        return dataframe

    if data['type'] == 'filter':
        df1 = process_input(spark_helper, data['input'][0])
        dataframe = df1.filter(data["condition"])
        return dataframe

    if data['type'] == 'select':
        df1 = process_input(spark_helper, data['input'][0])
        dataframe = df1.select(*data["columns"])
        return dataframe

    if data['type'] == 'source':
        source_ids.append(data['id'])
        datamart = data_access.get_by_uid(data['id'])
        dataframe = spark_helper.read_datamart(datamart)
        return dataframe
```

Extendable to more operations

FrontEnd written in TS/React

- Currently 5 operations are implemented
- Schema integration with UI



Semantic Data Integration - Summer

Thank you for your Attention