Grant Agreement N° 872592

PLATOON

Digital platform and analytics tools for energy

Deliverable D2.8
The PLATOON Unified Knowledge Base Creation v2

Contractual delivery date:
M27

Actual delivery date:
31/03/2022

Responsible partner:
P11: TIB, Germany

Project Title	PLATOON – Digital platform and analytic tools for energy
Deliverable number	D2.8
Deliverable title	The PLATOON Unified Knowledge Base Creation v2
Author(s):	Philipp D. Rohde and Maria-Esther Vidal (TIB)
Responsible Partner:	P11 – TIB
Date:	31.03.2022
Nature	R
Distribution level (CO, PU):	PU
Work package number	WP2 – Reference Architecture, Interoperability and Standardization

Work package leader	ENG, Italy
Abstract:	Data Integration in the context of the PLATOON project demands the development of data management techniques to efficiently overcome interoperability issues and provide a harmonized view of both data and their meaning (i.e., metadata). This deliverable reports on methodological and technological strategies developed in task T2.4; they allow for implementing data integration systems whose executions generate the PLATOON knowledge bases of pilots 1a, 2a, 3a, and 4a. These data integration systems are analyzed in terms of correspondences among the concepts of the PLATOON semantic data models and the pilots' data sources. Moreover, the characteristics of the created knowledge bases are reported, as well as queries whose execution enables the exploration of these knowledge bases.
Keyword List:	Data Integration, Unified Knowledge Base, Curation and Integration, Data Sources, Federated Query Processing

The research leading to these results has received funding from the European Community's Horizon 2020 Work Programme (H2020) under grant agreement no 872592.

Editor(s):	Philipp D. Rohde (TIB) and Maria-Esther Vidal (TIB)
Contributor(s):	Enrique Iglesias (TIB) and Sarra Ben Abbès (ENGIE)
Reviewer(s):	Erik Maqueda (TECN) Philippe Calvez (ENGIE) Martino Maggio (ENG)
Approved by:	Erik Maqueda (TECN) Philippe Calvez (ENGIE) Martino Maggio (ENG)
Recommended/mandatory readers:	WP3, WP4, WP5, and WP6

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	25/01/2022	Draft of Table of Content	Philipp D. Rohde and Maria-Esther Vidal (TIB)
v0.2	31/01/2022	Updated Table of Content	Philipp D. Rohde and Maria-Esther Vidal (TIB) and Erik Maqueda Moro (TECH)
v0.3	17/02/2022	Preliminaries, Methodology	Philipp D. Rohde (TIB)
v0.4	23/02/2022	Methodology and Pipeline Description	Enrique Iglesias and Maria-Esther Vidal (TIB)
v0.5	23/02/2022	Querying JSON-LD in MongoDB	Philipp D. Rohde (TIB)
v0.6	25/02/2022	Description of the knowledge graphs that are created in Pilots 1a (4.2-4.5), 3a (6-6.5) and 4a (7-7.5)	Sarra Ben Abbès (ENGIE)
v0.7	26/02/2022	Description of Knowledge Graph Pipeline in Pilot 2a	Maria-Esther Vidal (TIB)
v1.0	28/02/2022	First version for internal evaluation	Maria-Esther Vidal (TIB)
v1.1	28/02/2022	Continue the Description of the knowledge graphs that are created in Pilots 1a (4.2-4.5), 3a (6-6.5) and 4a (7-7.5)	Sarra Ben Abbès (ENGIE)
v1.2	01/03/2022	Continue the Description of the knowledge graphs that are created in Pilots 1a (4.2-4.5), 3a (6-6.5) and 4a (7-7.5)	Sarra Ben Abbès (ENGIE)
v1.3	05/03/2022	Empirical Evaluation of Federated Query Processing	Philipp D. Rohde and Maria-Esther Vidal (TIB)
v1.4	08/03/2022	Added missing captions and references	Philipp D. Rohde (TIB)
v1.5	09/03/2022	First version of internal Review	Philipp D. Rohde and Maria-Esther Vidal (TIB)
v1.6	21/03/2022	Comments from Internal Review	Erik Maqueda (TECN)
v2.0	28/03/2022	New Version Addressing Comments from Internal Review	Maria-Esther Vidal (TIB) Sarra Ben Abbès (ENGIE)
v3.0	28/03/2022	Formatting for Final Submission	Maria-Esther Vidal (TIB)
V3.1	30/03/2022	Final version for Submission	Maria-Esther Vidal and Ahmad Sakor (TIB)

Table of Contents

List of Figures

List of Tables

Terms and Abbreviations

API	Application Programming Interface
CSV	Comma Separated Values
DIS	Data Integration System
DoA	Description of Action
EC	European Commission
EM	Exploitation Manager
EU	European Union
FQP	Federated Query Processing
GA	Grant Agreement
GAM	General Assembly Meeting
H2020	Horizon 2020
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
MongoDB	A document-oriented database
MongoQL	Query language of MongoDB
MySQL	A SQL-compliant relational database
OJM	Object Join Map
ORM	Object Reference Map
PJTT	Predicate Join Tuple Table
PM	Project Manager
PTT	Predicate Tuple Table
PU	Public
QA	Quality Assurance
R2RML	RDB to RDF Mapping Language
RDB	Relational database
RDF	Resource Description Framework

RES	Renewable Energy Source
RFC	Request for Comments, a memorandum on Internet standards
RML	RDF Mapping Language
SCADA	Supervisory Control and Data Acquisition
SDM-RDFizer	Engine to create RDF knowledge bases from data sources whose mappings are specified in RML mapping rules
SOM	Simple Object Map
SPARQL	SPARQL Protocol and RDF Query Language
TSV	Tab Separated Values
W3C	World Wide Web Consortium
WP	Work Package
WPL	Work Package Leader
XML	Extensible Markup Language

Executive Summary

This deliverable reports on the outcomes of task T2.4 of WP2; the aim is to describe the data management techniques implemented to integrate heterogeneous data sources into the PLATOON knowledge bases. This deliverable presents a pipeline that resorts to the W3C standards (e.g., [R2]RML) to provide a declarative definition of the process of data integration. According to the data sources, mapping rules enable the definition of the classes, properties, and relationships of the PLATOON Semantic Data Models. The pipeline comprises a semantic connector and a federated query engine. The semantic connector creates the instances in a knowledge base by executing the mapping rules, while the federated query engine allows for the execution of queries against the generated knowledge bases. The results of executing the knowledge base creation pipeline are reported in the context of the pilots 1a, 2a, 3a, and 4a. They put in perspective the benefits of data management developed in WP2.

1. Introduction

Heterogeneous data sources are integrated into the PLATOON knowledge bases to offer an integrated view of data provided in different formats and data models. Diverse techniques developed in Task T2.4 allow for the transformation of this myriad of data sources into a unified knowledge base.

1.1 Purpose and Scope of the Document

This deliverable presents the main features of the developed methods and reports on pilots 1a, 2a, 3a, and 4a. In addition to the executive summary and this introduction, this document is organized as follows: Section 2 introduces the preliminary concepts required to understand the techniques applied in knowledge base creation. Section 3 describes a generic pipeline that comprises a semantic connector and a federated query processing engine. Two implementations are presented for the generic pipeline; one is based on a semantic connector built on top of SPARQL-Generate and another is based on the SDM-RDFizer. The knowledge bases of pilots 1a, 3a, and 4a have been created following the pipeline with SPARQL-Generate; Sections 4, 6, and 7 report on the results of these pilots, respectively. In pilot 2a, the knowledge base has been generated using the SDM-RDFizer; the results are presented in Section 5. The federated query processing techniques implemented in task T2.4, are empirically evaluated over pilot 2a knowledge; Section 8 reports on these results. Finally, conclusions reached in task T2.4 are outlined in Section 9.

1.2 Relationship with Other Documents

This document is related to the following deliverables of WP2: i) D2.1 [1] where the PLATOON reference architecture is defined; ii) D2.3 [2] where the PLATOON common data models for energy are defined; and iii) D2.4 [3] the previous version of this deliverable. It is also related to the deliverable D5.3 [4] of WP5 where the methods for data harmonization and knowledge extraction are described.

2. Preliminaries

2.1 Data Integration Systems

A data integration system (DIS) integrates two or more datasets. DISs provide a global schema (also known as mediated schema) to provide a reconciled view of all data available in the different data sources it integrates. Mappings between the global schema and source schema should be established to combine data residing in data sources considered in the integration process. A DIS execution results in an integrated knowledge base.

2.1.1 Virtual Data Integration Systems

In virtual data integration systems, the data to be integrated stays in its original format and under control of the data owner. Mapping rules are used to rewrite the query over the unified schema into queries over the data sources. Query planning is performed to optimize the rewritten query and to generate a query plan over the data sources. The query engine will then evaluate the query plan over the selected sources. The query answers are used to create a portion of the knowledge graph.

2.1.2 Materialized Data Integration Systems

In materialized data integration systems, the data to be integrated is transformed into a common data format and usually stored in one place. Mapping rules are executed to generate the instances of the unified schema. Controlled vocabularies are utilized for data annotation as a basis for entity alignment. Usually, this is implemented in Extract-Transform-Load (TLS) tools for data warehouses.

2.2 Mapping Languages

Mapping languages defined by the Semantic Web community can be used to transform non-RDF data sources to RDF. The rules represent mappings that define the concepts of ontology in terms of heterogeneous data sources. Such transformation can also be used to transform legacy databases, data streams, and semi-structured data sources published on the Web.

2.2.1 RDF Mapping Language (RML)

R2RML[1] is a W3C Recommendation for the transformation of relational databases to RDF. R2RML is a language for expressing customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice. An R2RML mapping is represented as a Triples Map, a rule that maps each row in the source to several RDF triples. The RDF Mapping Language ([R2]RML[2]) extends R2RML by generalizing to heterogeneous data sources. RML is a generic mapping language defined for expressing customized mappings from heterogeneous data sources, e.g., RDB, CSV, XML, JSON, to the RDF data model. These rules define correspondences between entities and properties, from data structures to RDF triples, i.e., RML Triples Maps. Each Triples Map consists of one logical source (`rr:logicalSource`), one Subject Map (`rr:subjectMap`), and zero or more Predicate-Object Maps (`rr:predicateObjectMap`). The Subject Map defines resources that

[1] https://www.w3.org/TR/r2rml/
[2] https://rml.io/specs/rml/

correspond to the instance of an RDF class. Further, the statement (`rr:predicateObjectMap`) defines the properties, and (`rr:objectMap`) expresses the object value of the property. RML enables joins between Triples Maps. A Referencing Object Map indicates the reference to another Triples Map (`rr:parentTriplesMap`). These references can be of two types: a) an object reference indicates that the object of the RDF triple that is created corresponds to the subject of another Triples Map. Both Triple Maps must be defined over the same logical data source. A join condition represents that the object of the created RDF triple corresponds to the subject of another Triples Map. The two Triples Maps can be mapped over two different logical data sources.

2.2.2 SPARQL-Generate

SPARQL-Generate [5] [6] is a graph-pattern centric mapping language. The language enables the definition of mapping rules by gathering data from various data sources and transforming the collected data into instances of a graph pattern. SPARQL-Generate can be implemented on top of existing SPARQL engines. SPARQL-Generate extends SPARQL 1.1. Using specified binding functions and iterators, it can be used to transform the data of various formats, e.g., RDB, CSV, XML, JSON, and many more. Similar to RML, SPARQL-Generate implements iterators to process the input data in different formats. These iterators are specific to one data format. In the WHERE clause of a SPARQL-Generate query, the data format specific bind functions are used to bind the values from the data entry to a variable. Similar to a CONSTRUCT query, the GENERATE clause defines the graph pattern that is being generated per data entry.

2.3 Mapping Rule Engines

Computational frameworks that enable the execution of a DIS mapping rules. Two exemplar mapping rule engines are described: a) SDM-RDFizer is an [R2]RM-compliant engine and b) SPARQL-Generate is an engine that implements the SPARQL-Generate language.

2.3.1 SDM-RDFizer

The SDM-RDFizer implements efficient data structures, data caching techniques, and query optimization strategies to scale up to a large number of mapping rules and data sets. As a result, semantic enrichment can be efficiently computed. The outcome of evaluations of the implemented techniques can be found in the reports [7], [8], and [9]. The SDM-RDFizer implements the data structures Predicate Tuple Table (PTT) and Predicate Join Tuple Table (PJTT) to execute duplicated removal and joins more efficiently. The PTT stores for each predicate P the triples that have been generated so far. The key encodes the subject and object of the triple. The PJTT stores the subjects of the triples generated by a join. PJTT is an index hash table where the key encodes each value of the attributes in the join condition. The value is a set of subject values in the second source associated with the values of the attributes in the hash key. Additionally, the physical operators Simple Object Map (SOM), Object Reference Map (ORM), and Object Join Map (OJM) have been implemented in the SDM-RDFizer. SOM generates an RDF triple from the execution of a simple Predicate Object Map. Each generated triple is checked against the associated PTT. If the triple was already generated before, it is discarded. If not, it will be added to the knowledge graph and the PTT will be updated accordingly. ORM extends SOM by using the subject of one Triples Map as the object of another Triples Map. The condition for this operator to work is that both Triples Maps must use the same data source. Afterward, the same process as with SOM is

followed, i.e., checking against the PTT to avoid duplicated triples. The OJM is an extension of ORM with the difference that the Triples Maps can be defined over different data sources and there exists a join condition between them. The corresponding PJTT is used in an index join where the outer table corresponds to the child map and the inner table to the PJTT. If an entry *e* exists with the same hash key, all the subjects in *e* are used to generate the resulting RDF triples. Finally, a similar procedure as before is followed to avoid the generation of duplicate triples. Iglesias et al. [8] [9] provide a more detailed description of the operators implemented by SDM-RDFizer.

2.3.2 SPARQL-Generate

SPARQL-Generate is implemented on top of Apache Jena and is open source[3]. It is possible to integrate it into other projects using Maven, run it as a Web API, or by executing a JAR file. Binding and iterator functions are implemented for several data formats exploiting already known techniques like XPath for XML or RFC 4180 for CSV/TSV. The implementation relies on the binding function extension mechanism of Apache Jena. SPARQL-Generate also uses this methodology for the iterator functions. Adding support for other data sources comes down to implementing the binding and iterator functions specific for the new data format. Also, additions to the existing languages can be implemented in the iterator functions. For example, SPARQL-Generate supports the iterator function `iter:JSONListKeys` to iterate over the key names of a JSON object which is not possible in JSONPath.

[3] https://github.com/sparql-generate/sparql-generate

3. Methods and Tools

This section describes the methodology followed to define the steps to integrate heterogeneous data sources into a knowledge base. Additionally, it presents the pipeline that allows for the execution of the PLATOON components for knowledge base creation and exploration.

3.1 Methodology for Defining a Data Integration System

The methodology for the definition of a data integration system requires the participation of the data providers, knowledge engineers and domain experts, and software developers. In PLATOON, the data providers are the partners from the pilots (e.g., Pilot 1a, 2a, and 3a); using the questionnaires described in deliverable D2.4. The knowledge engineers are responsible for representing the main characteristics of the energy domain by using formal data models (e.g., ontologies). The PLATOON semantic data models correspond to the unified schema of this pipeline; they have been defined by ENGIE in task T2.3 as the result of the contribution of the pilot partners. The knowledge engineers are able to describe the correspondences between the attributes of the data sources and the classes, properties, and relationships of the PLATOON semantics data models; they are expressed as mapping rules in the languages RDF Mapping Language (RML) and SPARQL-Generate. These mapping rules are validated by the data providers to certify the correctness of the definition of concepts of the PLATOON semantic data models in terms of the provided data. The partners from ENGIE defined the mapping rules of the pilots 1a, 3a, and 4a, while TIB has defined the mapping rules of 2a. The execution of the mapping rules is scheduled to generate a federation of SPARQL endpoints that logically integrates all the data transformed into the knowledge base. Additionally, the PLATOON federated query engine is configured to enable the efficient execution of queries against the federation. Software developers (e.g., from TIB, ENGIE, and UBO) generate the configurations required for the federation exploration. Figure 1 summarizes the steps of the methodology.

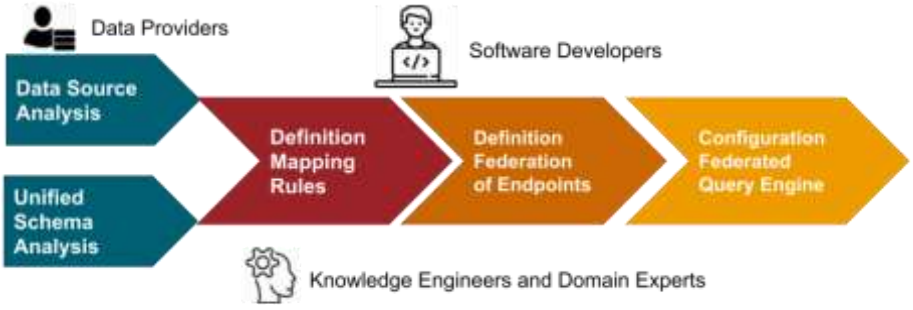

Figure 1: A Data Integration System Methodology - Methodology Steps

Following this methodology, the requirements to be satisfied by the exploration of the federation of SPARQL endpoints are also identified. The PLATOON pilots filled in questionnaires (reported in D2.4) which were used to identify the main properties of each data source. They also allow for detecting the opportunities for following the PLATOON data integration platform and the benefits that it will bring in terms of data sovereignty and secure data exchange. During the reporting period, the leader of T2.4 organized various workshops to guide the pilot owners into a more in-depth analysis of their developments. The pilot owners decided which data sources will be integrated using the PLATOON semantic data models according to their requirements. After discussions maintained with the

PLATOON partners, the coordinators, and data providers, it was decided that the knowledge bases of the pilots 1a, 2a, 3a, and 4a were created following this methodology. The results observed in these pilots are reported in the next sections.

3.2 A Generic Pipeline for Knowledge Base Creation

Following the PLATOON reference architecture presented in D2.1, a semantic adapter and a federated query engine are integrated into the pipeline depicted in Figure 2.

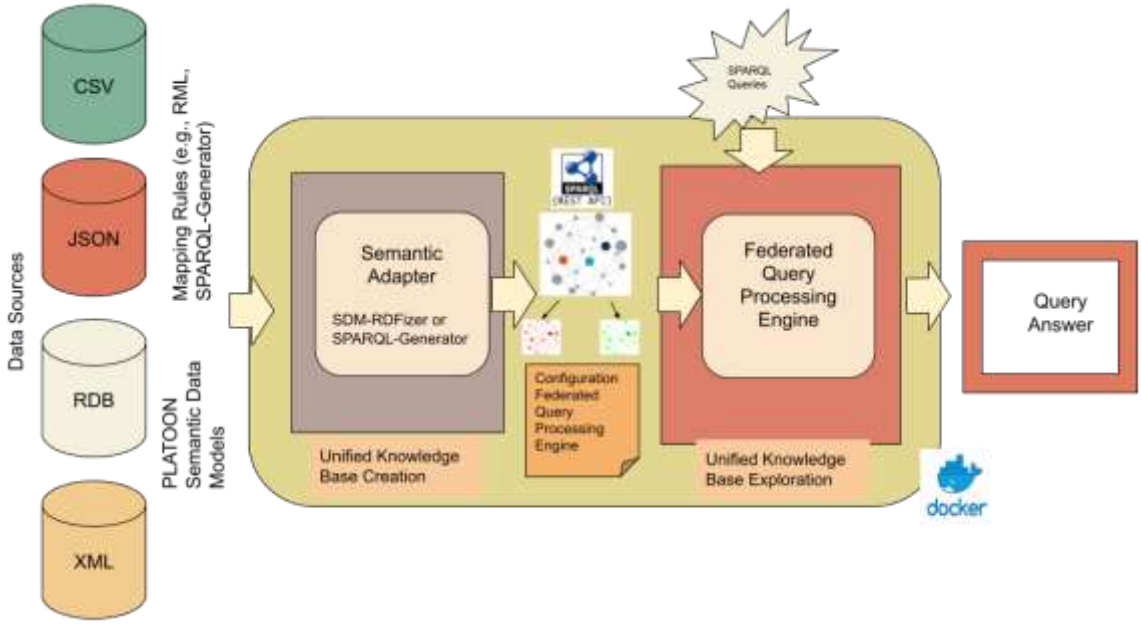

Figure 2: Knowledge Base Creation Pipeline- The Semantic Adapter receives data in different formats and transforms them into a federation of knowledge bases. RML mapping rules define concepts in the PLATOON semantic data models in terms of the data sources. The RDF data is uploaded into a federation of SPARQL endpoints which can be queried via SPARQL queries posed against a federated query engine

The PLATOON pipeline depicted in Figure 2 is generic and can be implemented by the different semantic adapter, e.g., the ones implemented with the SDM-RDFizer or SPARQL-Generate. The PLATOON knowledge base creation pipeline transforms data sources of various formats into RDF and provides the means to query the data as if they were integrated into a single SPARQL endpoint. The semantic adapter converts data sources in different formats, e.g., CSV and JSON, into RDF; this conversion is guided by mapping rules specified in SPARQL-Generate or RML. These mappings respect the PLATOON semantic data models (more on harmonization in D5.3 [4]). Additionally, the mapping rules are used to create semantic source descriptions used by the federated query engine during decomposition and source selection. Once the data is transformed to RDF it is uploaded into a federation of SPARQL endpoints (e.g., in Virtuoso or Fuseki). The federated query engine enables the user to collect results from all knowledge bases created using the pipeline with a single SPARQL query. Hence, the federated query engine provides a unified interface to the federation of SPARQL endpoints.

3.3 Pipeline based on the Semantic Connector - SDM-RDFizer

The members of TIB have developed an implementation of the pipeline in Figure 2 using SDM-RDFizer as the semantic adapter and FQP as federated query engine. The pipeline is implemented as a bash script that executes a series of docker images; each implements a different component of the pipeline. As a result, the pipeline is comprised of three docker

images: a) the SDM-RDFizer image; b) the Virtuoso images; and c) the FQP image. It has been integrated into the PLATOON framework in WP5. This pipeline is available in a GitHub repository[4] and uses dockerized components (Figure 3).

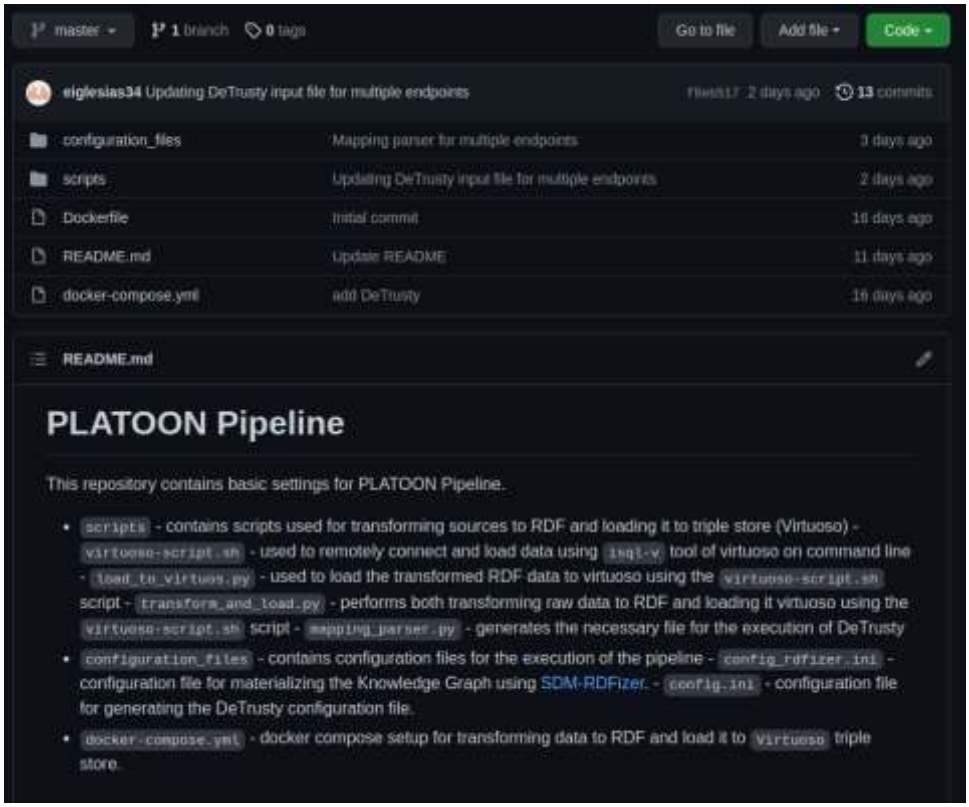

Figure 3: GitHub Repository of the PLATOON Pipeline - The GitHub repository contains all the necessary files and scripts for the execution of the PLATOON Pipeline. This includes the configuration files for the SDM-RDFizer and the mapping_parser.py script, as well as, the docker-compose.yml script that generates all required docker images.

The aforementioned GitHub repository contains required files and scripts for the execution of the PLATOON pipeline. These components are:

- ***Scripts:*** is a folder containing the scripts that transform mapping files and their corresponding data sources into RDF data, which is then loaded into the triple store (Virtuoso). These scripts are:
  - *Virtuoso-script.sh:* used to remotely connect and load data using the *isql-v* tool of Virtuoso.
  - *Load_to_virtuoso.py:* uses the *virtuoso-script.sh* to upload transformed RDF data to the Virtuoso triple store.
  - *Transform_and_load.py:* performs both the transformation of raw data into RDF data by invoking the SDM-RDFizer and uploads the data into a triple store by using *virtuoso-script.sh* (Figure 4).

[4] https://github.com/SDM-TIB/PLATOONPipeline

- *Mapping_parser.py:* generates the input file necessary for the execution of FQP by associating the classes from the mapping files with its corresponding predicates and endpoint that contains the RDF data.

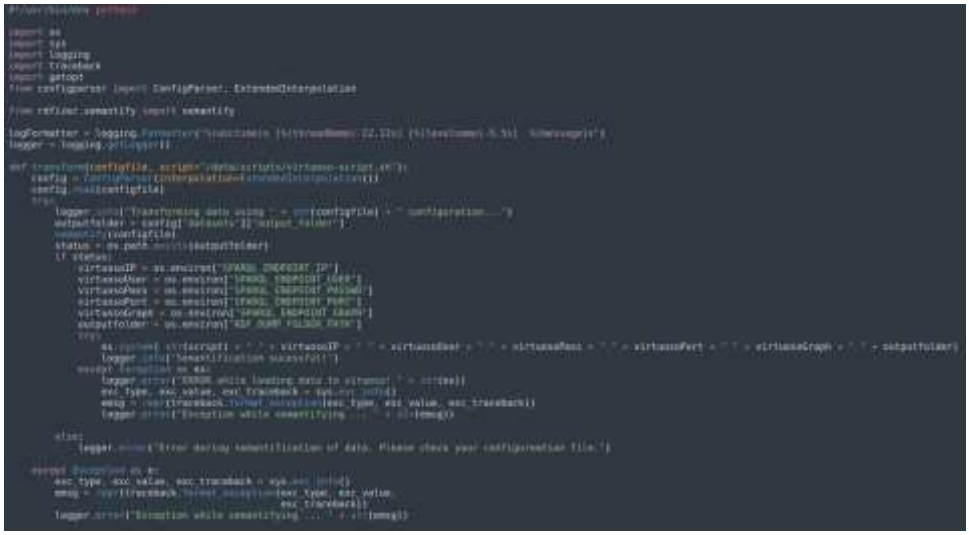

Figure 4: Portion of the transform_and_load.py script - The portion of the transform_and_load.py script illustrates the process in which the RDF data is uploaded to the triples store.

- ***Configuration_files:*** is a folder that contains the configuration files that are necessary for the execution of the SDM-RDFizer and *mapping_parser.py* script (Figure 5).

```
[default]
main_directory: /data

[datasets]
number_of_datasets: 1
output_folder: ${default:main_directory}/rdf-dump
all_in_one_file: yes
remove_duplicate: yes
name: test
enrichment: yes
dbtype: mysql
ordered: yes
large_file: false

[dataset1]
name: test
mapping: ${default:main_directory}/sam/tumor_histology_lc_clinical_notes_SLCG_transferred_mapping.ttl
```

Figure 5: Example of Configuration File - This figure illustrates an example of a configuration file for the execution of the SDM-RDFizer. The configuration file indicates the location of the mapping files, the credentials for accessing relational data bases, and the location of the output folder.

- ***Docker-compose.yml:*** docker compose set up file for the creation of the docker image of the SDM-RDFizer, DeTrusty, and Virtuoso (Figure 6).

Figure 6: Example of docker-compose.yml file - This figure illustrates the docker compose file used for the generation of the docker images that are required for the execution of the PLATOON Pipeline. Among these docker images are the images for the SDM-RDFizer, Virtuoso, and DeTrusty.

FQP[5] is a federated query engine for SPARQL endpoints. FQP decomposes the input query into star-shaped sub-queries, i.e., all triple patterns in a sub-query share the same subject. This type of decomposition was proposed by ANAPSID [10] and has been proven to be very efficient in the presence of RDF data. The source selection is guided by the aforementioned output of the mapping_parser.py. If an RDF type statement is present in the sub-query, it will be used to identify the sources that contribute to the sub-query in question. If no such statement is present, FQP selects all sources that contribute to RDF classes that contain all predicates of the sub-query. This allows FQP to minimize the number of contacted endpoints. Each sub-query is executed over the previously selected sources for the sub-query. The partial results retrieved from the endpoints are combined at the query engine level using non-blocking operators following the same idea presented already in ANAPSID [11]. FQP creates bushy plans in order to speed up the query execution. These features enable FQP to continuously generate complete and sound query results while minimizing the number of contacted endpoints and query execution time. This is in contrast to state-of-the-art federated query engines like FedX [12] which creates left-linear plans based on exclusive groups, i.e., decomposing the query into sub-queries that can be answered exclusively by

[5] https://github.com/SDM-TIB/DeTrusty

one endpoint. The current version of FQP is capable of executing SPARQL SELECT queries. The SERVICE clause from SPARQL 1.1 is also implemented. Some SPARQL 1.1 features are not yet implemented, e.g., GROUP BY and aggregate functions. FQP can be run in a Docker container. After providing FQP with the semantic source description of the set of SPARQL endpoints, it can be used via its HTTP API as if it was a regular SPARQL endpoint.

3.4 Pipeline based on the Semantic Adapter - SPARQL-Generate

The members of ENGIE and TIB have developed an implementation of the pipeline in Figure 2 using SPARQL-Generate as the semantic adapter and the query engine FQP[6] for query processing. In this implementation of the pipeline, the SPARQL-Generate tool is used to transform tabular data into the JSON-LD representation of RDF. The generated JSON-LD documents are uploaded into a MongoDB cluster. The number of documents stored in the database is constantly growing since new measurements are semantified and added to the appropriate collection in MongoDB once they arrive. FQP is a federated query engine that is capable of receiving data from SPARQL endpoints, non-RDF data sources, and RDF stores that are not accessible via SPARQL, e.g., JSON-LD stored in MongoDB. FQP receives SPARQL queries and decomposes the query into star-shaped sub-queries, i.e., all triple patterns in a sub-query share the subject. Each sub-query is executed over the appropriate sources. The sub-query results are combined to form the final query result. In the context of PLATOON, the federated aspect is not exploited but it is used in Pilots 1a and 3a to query JSON-LD documents stored in MongoDB. The FQP enables querying JSON-LD by translating the SPARQL sub-queries into their equivalent in MongoQL. FQP is an extension of Ontario [13]; a federated query engine for heterogeneous data sources, e.g., RDF via SPARQL endpoints and relational databases. The extensions include a wrapper for MongoDB as well as a streaming HTTP API. PolyWeb [14] follows a similar approach as FQP and Ontario by translating the SPARQL query into the native query language of the selected source. PolyWeb uses a decomposition type similar to exclusive groups and, like FedX, creates left-linear plans. Additionally, the operators implemented in PolyWeb are blocking, i.e., all results are generated at once. FQP is implemented in Python and supports SELECT and CONSTRUCT queries following SPARQL 1.0. It can either be used as a library in other Python applications or set up as a service via its HTTP API. Figure 7 shows how to use the FQP as a library. The data sources can be defined programmatically or as in the example read from a predefined file. FQP generates results incrementally. The example shows how to iterate over the result set so that answers can be further processed once they are received. Figure 8 **Fehler! Verweisquelle konnte nicht gefunden werden.**shows the use of the FQP as a service. In this set up, the data sources cannot be changed by the client application. In order to facilitate the use of the HTTP API, FQP streams the query answers, i.e., sending the answers once they are generated. Client applications can exploit this behavior to process the results one by one instead of waiting for all answers to be received before proceeding.

[6] https://github.com/PLATOONProject/Awudima_FQP

```python
from awudima import AwudimaFQP, Federation

fed = Federation.config("/path/to/your/config.json")
fqp = AwudimaFQP(fed)

query = "YOUR_SPARQL_QUERY_STRING"

resultSet = fqp.execute(query, keep_in_memory=True)
for row in resultSet.get():
    # process the result
    print(row)
```

Figure 7: FQP as Library

```python
import requests

URL = "http://URL_OF_AWUDIMA_INSTANCE/sparql?query="
QUERY = "YOUR_SPARQL_QUERY_STRING"

r = requests.get(URL + QUERY, stream=True)

for line in r.iter_lines():
    if line:
        # process result
        print(line)
```

Figure 8: FQP as a Service

```
SELECT DISTINCT * WHERE {
    ?tcontext <http://www.w3.org/2006/time#inXSDDateTime> ?inxsdtime .
    FILTER (?inxsdtime >= xsd:dateTime("2021-12-20T17:35:19.868869Z") &&
            ?inxsdtime <= xsd:dateTime("2021-12-30T17:40:19.868869Z") )
}
```

Figure 9: Example Query

The query in Figure 9 was executed in a test environment in pilot 1a. The pilot data was stored in a MongoDB cluster consisting of three nodes. The database consists of eight collections with an average of 240,000 JSON-LD documents per collection. Hence, there are 1.92 million documents stored in the database. The example query is executed over all collections since the query is not specific enough to rule out any of the collections available. The query returns 954,627 results in 7.5 minutes. Figure 10 shows the continuous answer generation using the FQP as a service by displaying every tenth received answer and the time passed since the query was sent to the service. We also performed a performance analysis. In this analysis we compared the execution time of the FQP with directly sending the translated query to the MongoDB cluster. We found that FQP does not add overhead to the query execution. Hence, it is entirely dependent on the performance of the cluster. This example also shows the importance of continuous answer generation. The first answers are available fast but with a blocking query engine, the user would need to wait 7.5 minutes before any result would be received.

Figure 10: FQP Output for Example Query

4. Pilot 1a: Predictive Maintenance of Wind Farms

This section aims to describe how the PLATOON semantic data models are used for the dataset of the Pilot 1a. We briefly present the data sources and the semantic data models that are used in this Pilot. Next, we explain the transformation process of data to semantic data using SPARQL-Generate tool. Finally, we provide an extract of the knowledge graph and some queries.

4.1. Pilot 1a Data Sources

Pilot 1a focuses on offshore and onshore wind turbines equipped with a doubly fed induction generator and resorts to the following primary sources of data (more details D2.4)

- **La Haute-Lys dataset** consists of data from a single, Onshore, General Electric 1.5 MW turbine (machine) placed at the La Haute-Lys wind farm in France. The dataset generated from this turbine focuses on high-frequency (500 Hz) measurements of the sensors necessary to gain insights into the electric response/behavior of the wind turbine. This data source is useful for validating the physical models or for data-driven models to capture healthy behavior.

- **ENGIE fleet dataset** consists of data from numerous turbines located in different wind farms. The focus is on Supervisory Control and Data Acquisition system data (SCADA data) sampled at 10-minute intervals. Unlike the *La Haute-Lys* dataset, *the ENGIE fleet dataset* includes turbines with more sensor types for measuring temperature signals and sensors for measuring wind speed, wind direction, generator speeds, torque, etc. Furthermore, this dataset contains fault logs with different fault scenarios, e.g., a short circuit in generator winding. One use case is identified in this pilot: LLUC 1a-01 - Failure detection using a combined data-driven and physics-based model. The ENGIE fleet dataset is an extension of the ENGIE La Haute Borne open dataset. In the DoA, this dataset was described as two separate datasets. However, given that all project partners of Pilot 1a have access to the extended dataset, we opted to use the extended dataset in all specification documents.

- **Open wind speed dataset** consists of wind measurements distributed along the Belgian North Sea. Sensor data includes wind speed and wind direction. This dataset is used in LLUC 1a-01 to assess the typical ranges of wind speeds and directions that can occur in the field. These are used as basis of understanding for defining semantic labels describing wind conditions.

- **Offshore measurement campaign data** consists of acceleration measurements collected on an offshore wind turbine drivetrain. These measurements were in the end not used in Pilot 1a, given that a new dedicated measurement campaign is conducted during the project targeting current measurements that are more appropriate for the analytics methods developed in Pilot 1a.

- **Dedicated current measurement campaign data** consists of current signals that are acquired on an onshore wind turbine. These data are similar to the La Haute Lys dataset. As such they will be merged in further discussions on data handling and analytics with the La Haute Lys data as the same processing methodology applies.

4.2 The PLATOON Semantic Data Models Defined in the Pilot 1a Data Sources

The goal of the harmonized semantic data of Pilot 1a, defined in T2.3, is to cover all needs assessed from the different data sources. The PLATOON semantic data model of Pilot 1a is composed of different modules according to these specific needs:

- **Wind turbine ontology** that includes wind turbine component such as the generator and power converter and subcomponents such as rotor and stator, etc.
- **Failures and Damages ontology** that concerns damages and failures describing each entity that performs a function with a certain efficiency.
- **Generic Property ontology** that describes properties of the wind turbine such as current, voltage, electric power, vibration, temperature which are observed by specific sensors.
- **Sensor ontology** that presents the concept Sensor which is generic and common to different domains and its observations like anemometer, weathervane, etc.
- **Status Code Alarm ontology** that details the status and alarms messages used to o distinguish normal from abnormal operation.
- **Event ontology** that presented events which are related to different features of interest such as a wind turbine. The maintenance event (e.g., Repair Maintenance, Replacement Maintenance, Device Adding Maintenance, etc.) is scheduled to take place at a certain time and date.

Table 1 shows the number of classes and relationships for each ontology used in Pilot 1a.

Table 1: Statistics of ontologies used in Pilot 1a

Ontology	#Classes	#Relationships
Wind turbine ontology	182	268
Failures and Damages ontology	71	100
Generic Property ontology	86	177
Sensor ontology	679	2930
Status Code Alarm ontology	24	37
Event ontology	54	87

4.3 The Pilot 1a PLATOON Knowledge Graph Creation

The process of semantic transformation is handled in three steps: (1) comprehension/contextualisation of the meaning of datasets with the interaction with stakeholders, (2) application of an RDF Mapping Language (SPARQL-Generate in our case) to get the data as instances of the PLATOON Semantic Data Models (SDMs), and (3) population of SDMs by processing the RDF instances from step 2. Our choice of SPARQL-Generate[7] to convert non-RDF-to-RDF is that: (i) its mapping language is based on SPARQL, (ii) the mappings can be edited in a text editor, (iii) it has an extendable/modular/flexible architecture, and (iv) it can be embedded in a more complex pipeline (e.g., Apache Beam). For all the datasets that we received, our approach is divided into static datasets (e.g., wind turbine, blade) and dynamic datasets. The URI is built only one time for the static datasets.

[7] https://ci.mines-stetienne.fr/sparql-generate/

Figure 11 shows an extract of SPARQL-Generate for generating semantic static data such as wind turbine and its properties, substation, wind farm. The *Generate clause* contains the template of the graph creation where the triples are constructed. The *iterator clause* contains the source that is used and allows to extract bits of documents (sources) and make a variable be successively bound to these extracted bits of documents.

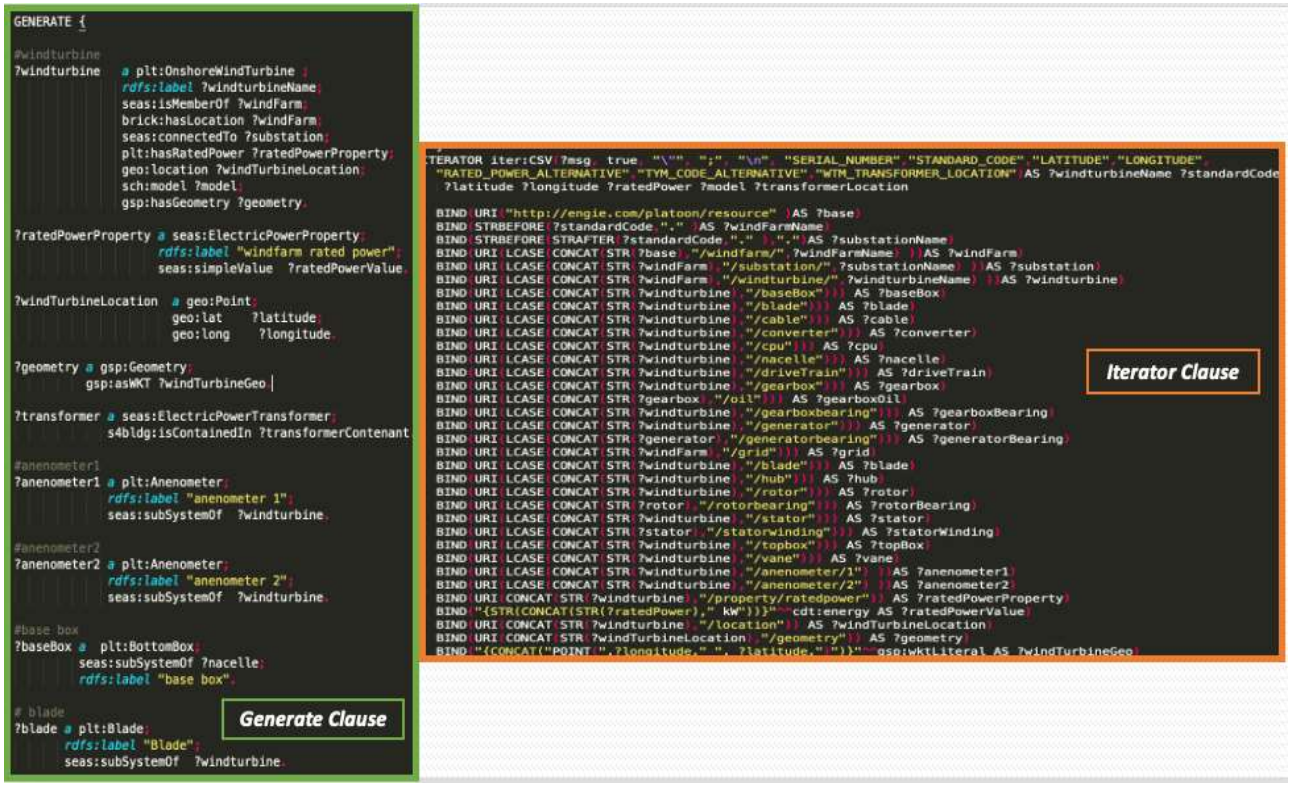

Figure 11: Example of SAPRQL query for static data of Pilot 1a

Figure 12 shows an extract of the knowledge base generated from the dataset of pilot 1a. This knowledge base contains 197,831 triples.

Some competency questions are defined in Pilot 1a, that domain experts want the ontology and the knowledge graph helps to answer:

- What are the sensors related to a wind turbine, where are they located and what do they measure?
- What are the properties of the wind turbine in each farm?

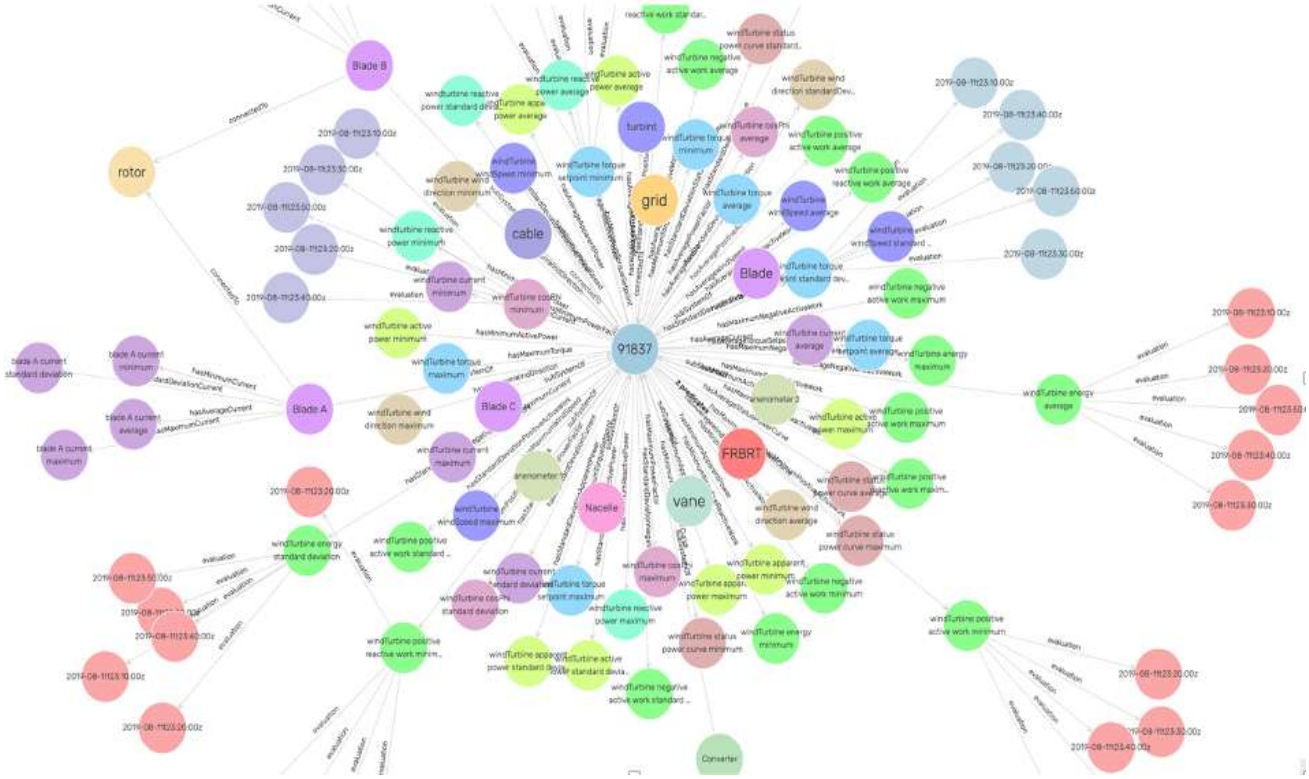

Figure 12: Extract of the Knowledge Base of Pilot 1a

Figure 13 shows an example of SPARQL query that provides an answer to the second question posed.

```
PREFIX plt: <https://w3id.org/platoon/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX seas: <https://w3id.org/seas/>
PREFIX brick: <https://brickschema.org/schema/1.1/Brick#>
select * where {
?windFarm  a plt:WindFarm ;
           rdfs:label ?windFarmName.
#windTurbine
?windTurbine  a plt:OnshoreWindTurbine ;
              rdfs:label ?windTurbineName;
              seas:isMemberOf ?windFarm;
              brick:hasLocation ?windFarm;
              plt:hasAverageWindDirection ?averageWindDirectionProperty;
              plt:hasMaximumWindDirection ?maximumWindDirectionProperty;
              plt:hasMinimumWindDirection ?minimumWindDirectionProperty;
              plt:hasStandardDeviationWindDirection ?standardDeviationWir
              plt:hasAverageActivePower ?averageActivePowerProperty; #Act
              plt:hasMaximumActivePower ?maximumActivePowerProperty;
              plt:hasMinimumActivePower ?minimumActivePowerProperty;
              plt:hasStandardDeviationActivePower ?standardDeviationActivePower
              plt:hasAverageApparentPower ?averageApparentPowerProperty; #Appar
              plt:hasMaximumApparentPower ?maximumApparentPowerProperty;
              plt:hasMinimumApparentPower ?minimumApparentPowerProperty;
              plt:hasStandardDeviationApparentPower ?standardDeviationApparentP
              plt:hasAveragePowerFactor ?averageCosPhiProperty; # windTurbine P
              plt:hasMaximumPowerFactor ?maximumCosPhiProperty;
              plt:hasMinimumPowerFactor ?minimumCosPhiProperty;
              plt:hasStandardDeviationPowerFactor ?standardDeviationCosPhiPrope
              plt:hasAverageCurrent ?averageCurrentProperty; # windTurbine Curr
              plt:hasMaximumCurrent ?maximumCurrentProperty;
              plt:hasMinimumCurrent ?minimumCurrentProperty;
              plt:hasStandardDeviationCurrent ?standardDeviationCurrentProperty;
              plt:hasAverageNegativeActiveWork ?averageNegActWorkProperty;# windTurbine Negative active work
              plt:hasMaximumNegativeActiveWork ?maximumNegActWorkProperty;
              plt:hasMinimumNegativeActiveWork ?minimumNegActWorkProperty;
              plt:hasStandardDeviationNegativeActiveWork ?standardDeviationNegActWorkProperty;
              plt:hasAverageNegativeReactiveWork ?averageNegReactWorkProperty; # windTurbine Negative reactive work
              plt:hasMaximumNegativeReactiveWork ?maximumNegReactWorkProperty;
              plt:hasMinimumNegativeReactiveWork ?minimumNegReactWorkProperty;
              plt:hasStandardDeviationNegativeReactiveWork ?standardDeviationNegReactWorkProperty;
              plt:hasAveragePositiveActiveWork ?averagePosActWorkProperty; # windTurbine positive active work
              plt:hasMaximumPositiveActiveWork ?maximumPosActWorkProperty;
              plt:hasMinimumPositiveActiveWork ?minimumPosActWorkProperty;
              plt:hasStandardDeviationPositiveActiveWork ?standardDeviationPosActWorkProperty;
              plt:hasAveragePositiveReactiveWork ?averagePosReactWorkProperty; # windTurbine positive reactive work
              plt:hasMaximumPositiveReactiveWork ?maximumPosReactWorkProperty;
              plt:hasMinimumPositiveReactiveWork ?minimumPosReactWorkProperty;
              plt:hasStandardDeviationPositiveReactiveWork ?standardDeviationPosReactWorkProperty;
              plt:hasAverageReactivePower ?averageReactivePowerProperty; # windturbine reactive power
              plt:hasMaximumReactivePower ?maximumReactivePowerProperty;
              plt:hasMinimumReactivePower ?minimumReactivePowerProperty;
              plt:hasStandardDeviationReactivePower ?standardDeviationReactivePowerProperty;
              plt:hasAverageTorque ?averageTorqueProperty;# windTurbine torque
              plt:hasMaximumTorque ?maximumTorqueProperty;
              plt:hasMinimumTorque ?minimumTorqueProperty;
              plt:hasStandardDeviationTorque ?standardDeviationTorqueProperty;
              plt:hasAverageTorqueSetpoint ?averageSetpointTorqueProperty;# windTurbine torque setpoint
              plt:hasMaximumTorqueSetpoint ?maximumSetpointTorqueProperty;
              plt:hasMinimumTorqueSetpoint ?minimumSetpointTorqueProperty;
              plt:hasStandardDeviationTorqueSetpoint ?standardDeviationSetpointTorqueProperty;
              plt:hasAverageStatusPowerCurve ?averageStatusPowerCurveProperty;# windTurbine status power curve
              plt:hasMaximumStatusPowerCurve ?maximumStatusPowerCurveProperty;
              plt:hasMinimumStatusPowerCurve ?minimumStatusPowerCurveProperty;
              plt:hasStandardDeviationStatusPowerCurve ?standardDeviationStatusPowerCurveProperty;
              plt:hasAverageWindSpeed ?averageWindSpeedProperty;  # windTurbine wind Speed
              plt:hasMaximumWindSpeed ?maximumWindSpeedProperty;
              plt:hasMinimumWindSpeed ?minimumWindSpeedProperty;
              plt:hasStandardDeviationWindSpeed ?standardDeviationWindSpeedProperty;
              plt:hasWindSpeedTurbInt  ?windSpeed_turbIntProperty. # windTurbine
```

Figure 13: SPARQL query corresp. to the second natural language question of Pilot 1a

Figure 14 shows an extract of the answer of the above query. In each wind farm (e.g., FRBRT represented by <http://engie.com/platoon/resource/windfarm/frbrt>), we have several wind turbines (e.g. 91848 represented by

<http://engie.com/platoon/resource/windfarm/frbrt/windturbine/91848>). Different properties are available for each wind turbine such as wind direction, active power, apparent power, current, reactive power, torque setpoint, etc.

	windFarmURI	windFarmName	windTurbineName	dateTime	averageWindDirectionValue	maximumWindDirectionValue	minimumWindDirectionValue	standardDeviationWindDirectionValue	averageActivePowerValue
1	http://engie.com/platoon/resource/windfarm/frbrt	FRBRT	91848	"2019-08-18T06:00:00Z"	"195.69 deg"	"223.94 deg"	"171.7 deg"	"8.46 deg"	"426.71 kW"
2	http://engie.com/platoon/resource/windfarm/frbrt	FRBRT	91848	"2019-08-18T13:50:00Z"	"222.47 deg"	"274.31 deg"	"132.0 deg"	"117.53 deg"	"871.57 kW"
3	http://engie.com/platoon/resource/windfarm/frbrt	FRBRT	91848	"2019-08-18T19:30:00Z"	"281.09 deg"	"319.01 deg"	"258.19 deg"	"9.78 deg"	"157.3 kW"

Figure 14 - Answer extract of the above SPARQL query

5. Pilot 2a: Electricity Balance and Predictive Maintenance

This section explains the process performed to create the Pilot 2a knowledge base. First, the data sources are briefly presented and defined in terms of the classes of the PLATOON semantic data models. Next, the mapping rules that establish the correspondences among the attributes of these data sources and the classes and properties are described. Finally, the main characteristics of the Pilot 2a knowledge base are reported.

5.1. Pilot 2a Data Sources

Pilot 2a focuses on integrating and deploying different PLATOON analytical services with the Institute Mihajlo Pupin (IMP) proprietary VIEW4 Supervisory control and data acquisition (SCADA) system deploys the energy value chain in Serbia. Energy resources related to Renewable Energy Sources (RES) in this pilot include: wind power plants and PV power Plants. Electricity production from solar and wind plants is subject to forecast errors that drive demand for balancing. These data sources are described as follows:

- **PUPIN-RES-PROD:** Historical Wind Power Production Measurements; it contains measurements of the production from the wind **power plant,** and **topology data**.
- **PUPIN-RES-PV (Predictive Maintenance)**: Data is collected by the Phasor Measurement Unit installed at IMP side.
- **PUPIN-WeatherBit**: Meteorological Data for RES Production (Generation) Forecasting Modelling Data. Meteorological dataset is utilized for RES production forecasting models training process as input data. Data is historical observational data.
- **PUPIN-RES-Effects**: Effects of Renewable Energy Sources on the Power System calculated based on the input by Phasor Measurement Unit installed at PUPIN.
- **PUPIN-ENTSO-E:** Transparency Platform-Energy Identification Codes (EICs); it maintains data about 39 electricity transmission system operators (TSOs) from 35 countries across Europe.

5.2 The PLATOON Semantic Data Models Defined in the Pilot 2a Data Sources

The following table summarizes the number of classes in the PLATOON semantic data models that are defined with attributes from the Pilot 2a data sources. A detailed description of the classes per data source is presented in Appendix A. The Pilot 2a data sources populate 158 different classes out of 616 classes in the PLATOON semantic data models, i.e., 25.65% of classes are defined; they also define 107 predicates of these classes.

Table 2: Statistics of Data Sources used in Pilot 2a

Pilot 2a Data Source	Number of Classes Populated with the Data Source	Percentage of Defined Classes	Number of Predicates Defined	Percentage of Defined Predicate
PUPIN-WeatherBit	87	14.13%	75	17.18%
PUPIN-RES-PROD	34	5.52%	24	5.69%
PUPIN-RES-PV	26	4.22%	23	5.45%
PUPIN-ENTSO-E	22	3.58%	85	20.14%
Total	**158 Different Classes**	**25.65%**	**107 Different Predicates**	**25.36%**

5.3 Mapping Rules in Pilot 2a

The correspondences between the PLATOON semantic data models and the Pilot 2a data sources are defined in terms of 2,093 RML mapping rules. They transform data stored in a relational database into instances of RDF. Figure 15 summarizes the number of mapping rules per class in the PLATOON semantic data models and data sources. On average each class is defined by 12.57 mapping rules, and the number of mapping rules ranges from 3 to 272. The class https://w3id.org/seas/Forecast is defined by 272 mapping rules with data collected from the data source WeatherBit. On the other hand, https://w3id.org/seas/FeatureOfInterest is defined by 228 rules; attributes from the data source PUPIN-ENTSO-E are defined using 101, and 77, 27, and 23 mapping rules define this class with attributes from PUPIN-RES-PROD, PUPIN-RES-PV, and WeatherBit. The class https://w3id.org/platoon/WindFarm is populated with data from PUPIN-RES-PROD; 27 mapping rules define this process.

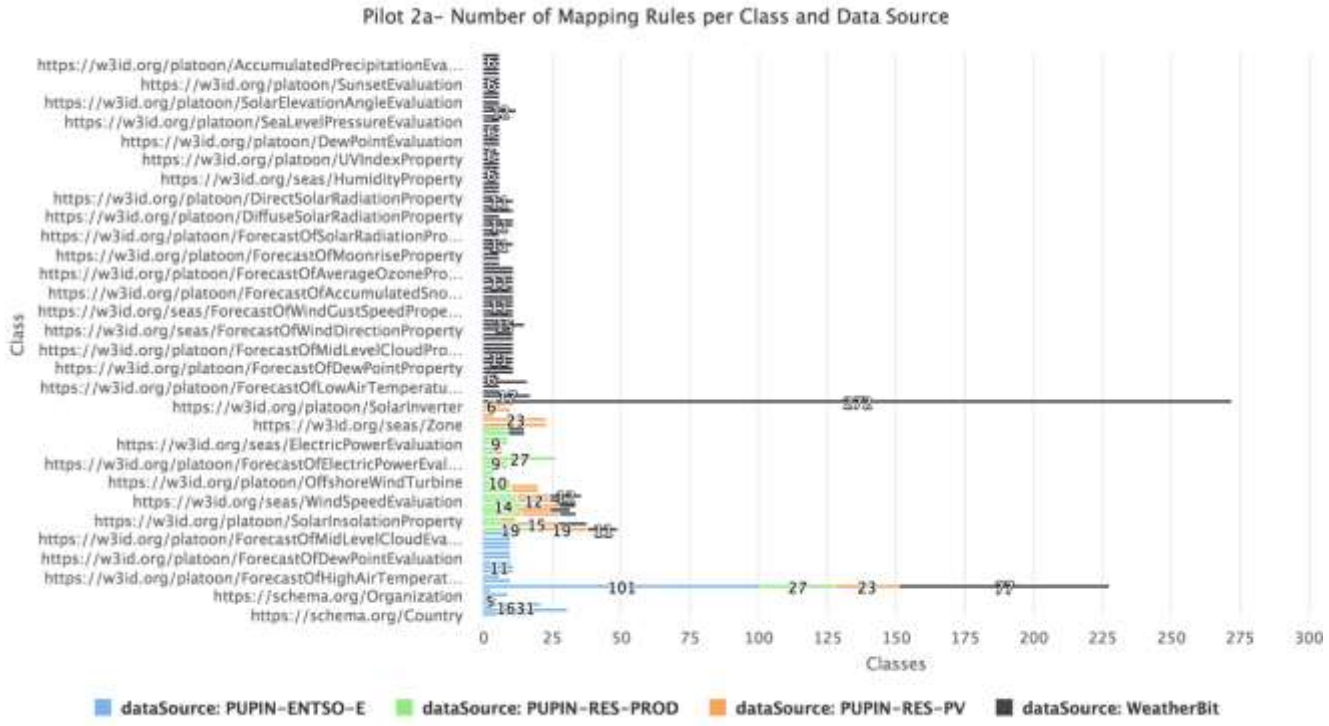

Figure 15: Number of Mapping Rules per Class and Data Source in Pilot 2a

The following screenshot (Figure 16) illustrates the definition of the class https://w3id.org/platoon/WindFarm using RML. The entities that populate this class are characterized by OWL object properties `sch:location`, `plt:datasource`, `seas:connectedTo` (i.e., control area to which the wind farm is connected), `seas:isMemberOf` (i.e., the asset name), `rdfs:label`, and `plt:country` and `sch:city` (i.e., country code and the city where the wind farm is located).

Figure 16: RML Mapping for Class https://w3id.org/platoon/WindFarm

5.4 The PLATOON Knowledge Base Generation in Pilot 2a

The RML mapping rules and the Pilot 2a datasets are processed using the semantic connector explained in 3.3. This connector implemented by the SDM-RDFizer plans the execution of the mapping rules and loading of the dataset to speed up the process of knowledge base creation. These transformations are supported by well-known properties of relational algebra, e.g., the pushing down of projections and selections into the data sets. They enable the reduction of the size of data sources and the elimination of duplicates and physical data structures that reduce the pipeline's execution time [8] [9] [15]. Given the complexity of the Pilot 2a mapping rules and the size of the datasets, the features offered by. SDM-RDFizer has been crucial for enabling the generation of the knowledge base. Following the pipeline presented in 3.2, SPARQL endpoints according to the configuration given as input once the Pilot 2a knowledge base is generated. Additionally, the metadata that defines the knowledge bases stored in the created endpoints is created. This metadata describes the federation of knowledge bases and provides all the information required for the federated query engine to work. Figure 17 illustrates a portion of the configuration file that defines the pipeline. It is composed of 21 files of mapping rules which compose 272 mapping rules that define the whole process. Mapping rules are executed against a MySQL database that maintains the data collected from Pilot 2a data sources.

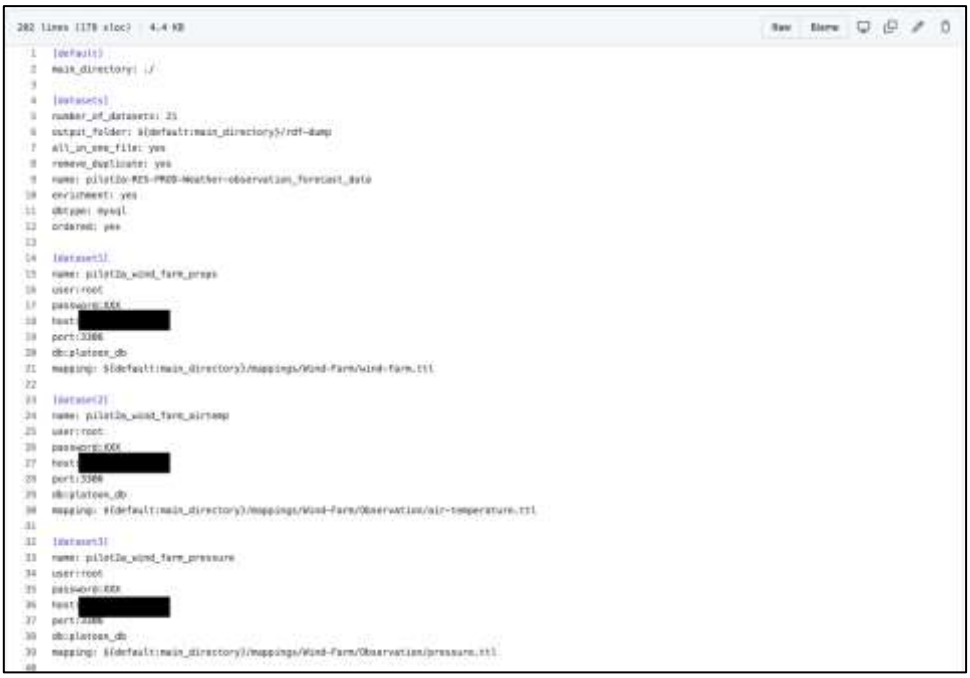

Figure 17: Portion of Pipeline Configuration for Pilot 2a

The pipeline also creates metadata required by the federated query engine; it includes per class in the knowledge base the predicates where this class participates as domain, the direction of the endpoint, and the links between classes. Figure 18 illustrates a fragment of the generated metadata. The SPARQL endpoints are deleted from the image.

Figure 18: Fragment of generated Metadata

The pipeline for Pilot 2a is shared as a docker container and available in the GitHub repository of PLATOON (Figure 19). This instance of the pipeline is implemented in 13.8k lines of code.

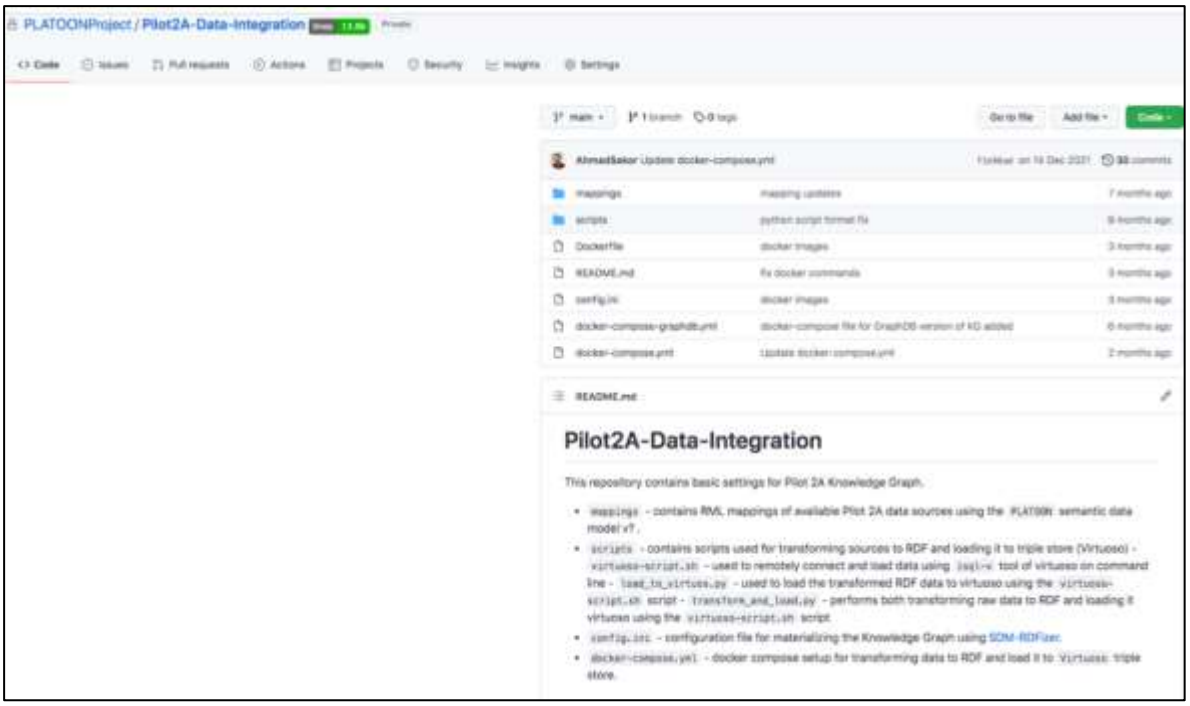

Figure 19: GitHub Page for the Pilot 2a specific Pipeline Implementation

The next instructions describe the steps required to execute the pipeline. They were executed in the computer servers of the Pilot 2a partners (IMP) to create locally their knowledge base (Figure 20).

Figure 20: Instructions for running the Pipeline for Pilot 2a

5.5 The Pilot 2a PLATOON Knowledge Base

The current version of the Pilot 2a knowledge base (by March 2022) comprises 80,762,377 resources described in terms of 220,204,301 RDF triples. The resources are part of 162 classes. Figure 21 depicts in logarithm scale, the cardinality of the classes in the pilot 2a knowledge base. The class `plt:AirTemperatureEvaluate` 25,950,820 instances and corresponds to the class with the large number of resources in the Pilot 2a knowledge base.

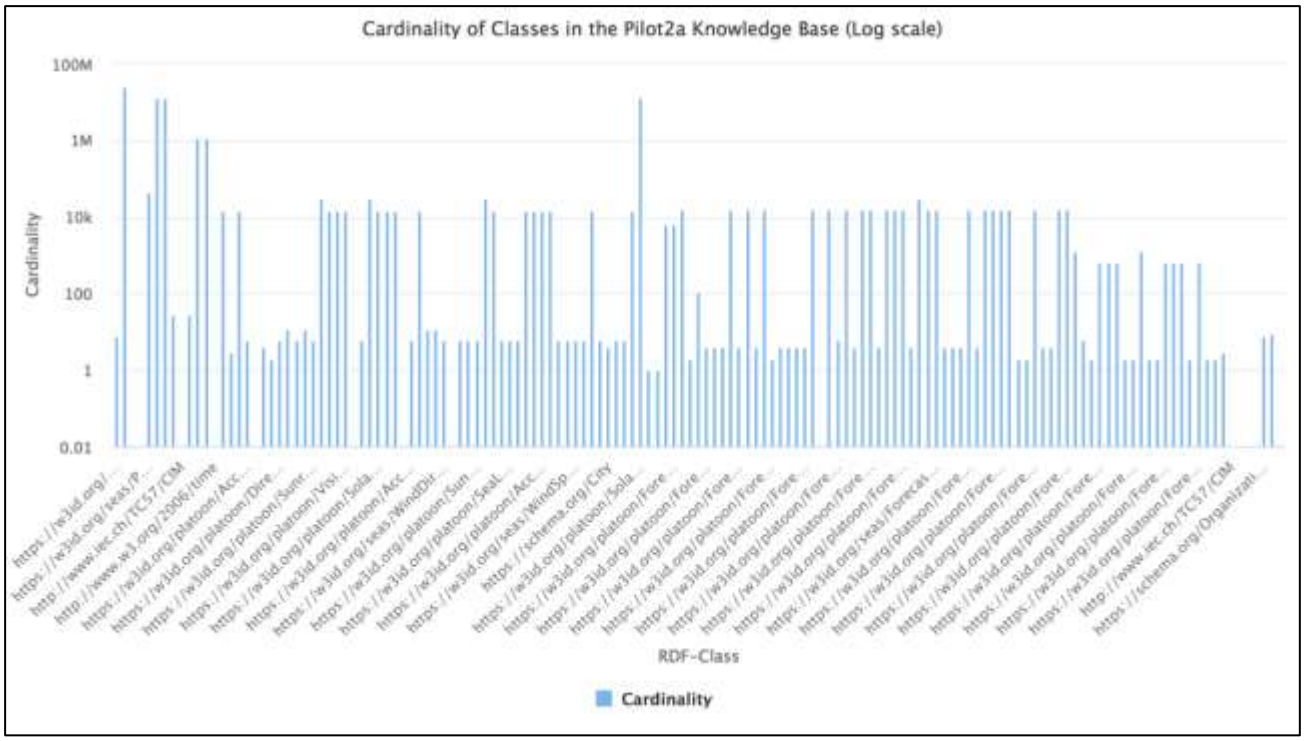

Figure 21: Cardinality of Classes in the Pilot 2a Knowledge Base

A network analysis is conducted to study the basic characteristics of the Pilot 2a knowledge base. A directed graph G=(V,E) is created from the Pilot 2a knowledge base to perform traditional network analysis methods. Vertices in V correspond to the classes in the pilot 2a knowledge base with at least one resource or instance. A labeled directed edge e=(q,p,k) belongs to E if there are classes Q and K in V, and q and k are instances of Q and K and the RDF triple (q p k) belong to the knowledge base. G provides an aggregated representation of the data represented in the knowledge base and illustrates the level of connectivity that exists between the classes in the current version of the knowledge base. Figure 22 depicts the directed graph G=(V,E); it is composed of 143 vertices and 785 directed edges.

Using Cytoscape[8], the main properties of the knowledge base are analyzed in terms of graph measures. Table 3 reports on the results of these measures computed by the network analysis tool of Cytoscape. An average number of neighbors indicates the average connectivity of a vertex or node in a graph. Network diameter measures the shortest path that connects the two most distant nodes in a graph. The clustering coefficient measures the tendency of nodes that share the same connections in a graph to become connected. If a neighborhood is fully connected, the clustering coefficient is 1.0, while a value close to 0.0 means no link in the neighborhood. Network density measures the portion of potential edges in a graph that are actually edges. A value close to 1.0 indicates that the graph is fully connected, while in disconnected graphs, density is close to 0.0.

[8] https://cytoscape.org/

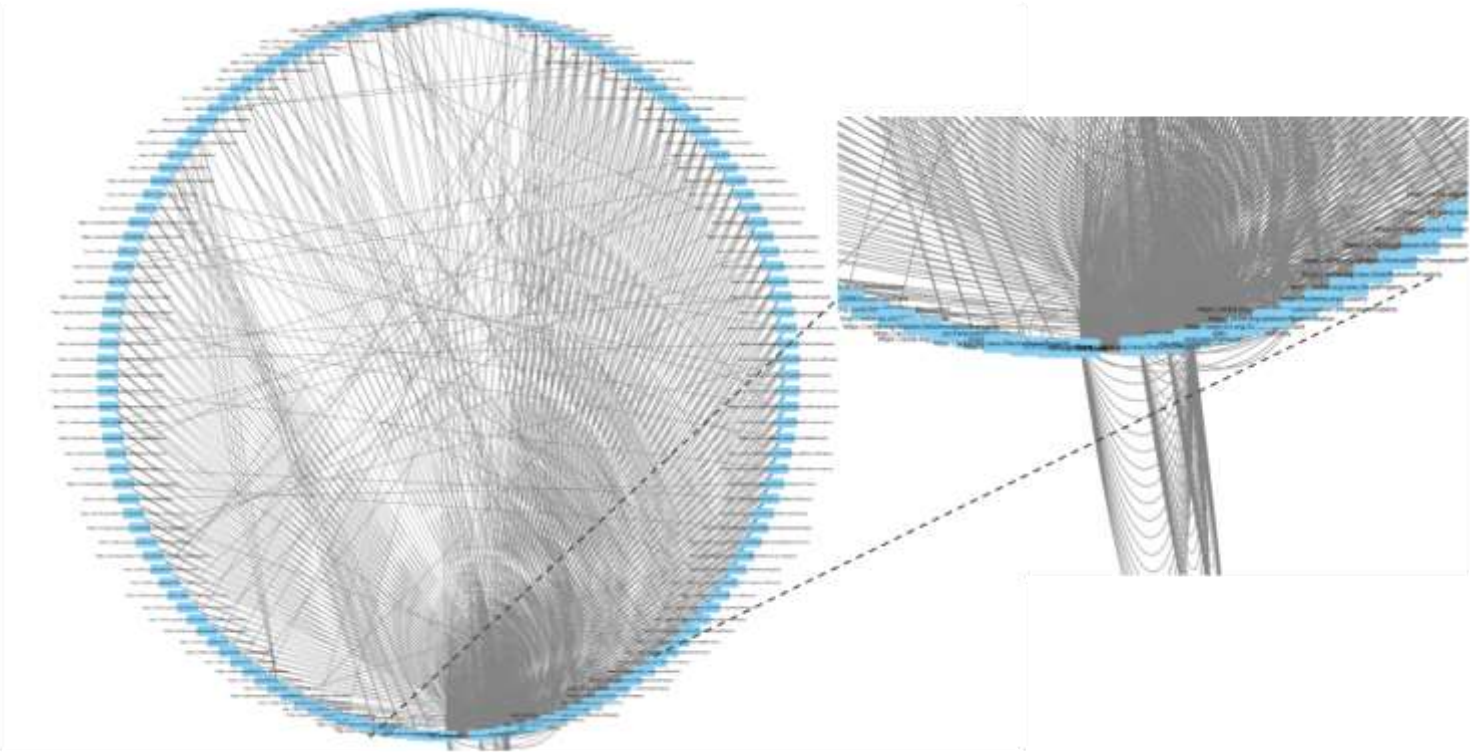

Figure 22: Connectivity of the Pilot 2a Knowledge Base

Lastly, the number of connected components shows the number of subgraphs composed of vertices connected by at least one path. Several connected components greater than one indicate that portions in a graph are disconnected. As reported in Table 3, the current knowledge base is connected (i.e., it only has one connected component). The average number of neighbors suggests that the knowledge base comprises more connected classes. At the same time, the clustering coefficient provides evidence that the conducted data integration techniques increase the connectivity in the neighborhoods.

Table 3: Network Analysis of Pilot 2a Knowledge Base

Metric	Value in Current Version of Pilot 2a KB
Number Nodes	143
Number Edges	785
Avg. Number of Neighbors	6.86
Network diameter	5
Clustering coefficient	0.098
Network density	0.032
Number of Connected Components	1

Cytoscape also plots the closeness centrality distribution of graph G. Closeness centrality indicates how close a vertex is to the other reachable vertices in the graph. It is a higher-is-better metric and is computed as the average of the shortest distances to all other nodes in the graph. Figure 23 depicts the values of closeness centrality versus the degree of a vertex. In general, the values are high. Only five classes have a value of closeness centrality equal to 0.0; these classes are `time:Interval`, `time:TimeZone`, `schema:Country`, `platoon:EICFunction`, `seas:ForecastOfWindDirectionEvaluation`. The rest of the

classes have values higher of 0.40. Specifically, the top-6 classes with the highest values of closeness centrality are `time:DateTimeDescription`, `time:Instant`, `time:TemporalEntity`, `schema:Location`, `seas:FeatureOfInterest`, and `geo:Feature`. The high values observed in these six classes indicate that most of the entities in the Pilot 2a are instances of these classes or have properties whose values are instances of them. The knowledge base is populated with observational energy data, thus, this statement is consistent with the data types included in the knowledge base.

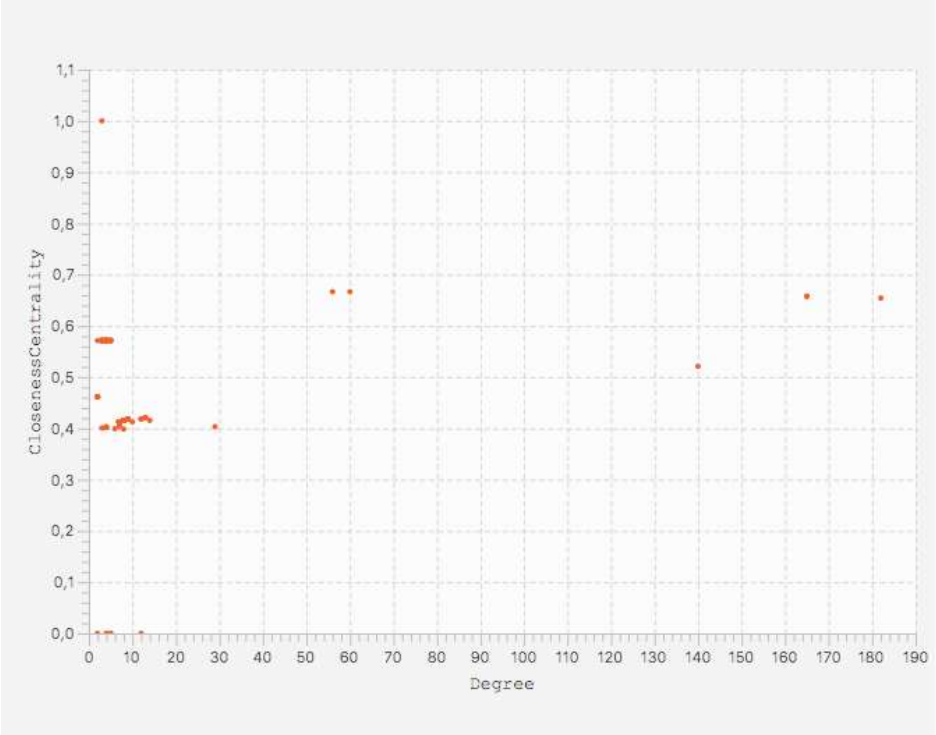

Figure 23: Closeness Centrality Distribution

A similar analysis is conducted with the metric betweenness centrality; Figure 24 presents the distribution. Betweenness centrality indicates the control of a vertex over the rest of the vertices in a graph; it measures if a vertex is part of the paths between other vertices. In the studied directed graph, vertices with high betweenness centrality represent classes between several classes that are not directly connected. The values of betweenness centrality are higher for the classes: `geo:Feature`, `schema:Location`, `seas:FeatureOfInterest`, and seas:Forescast, while others are relatively low. These results indicate that these classes are part of the paths connecting many class pairs. These results corroborate the observation that classes representing types of features and the locations from which they are collected, characterized many of the entities in the Pilot 2a knowledge graph.

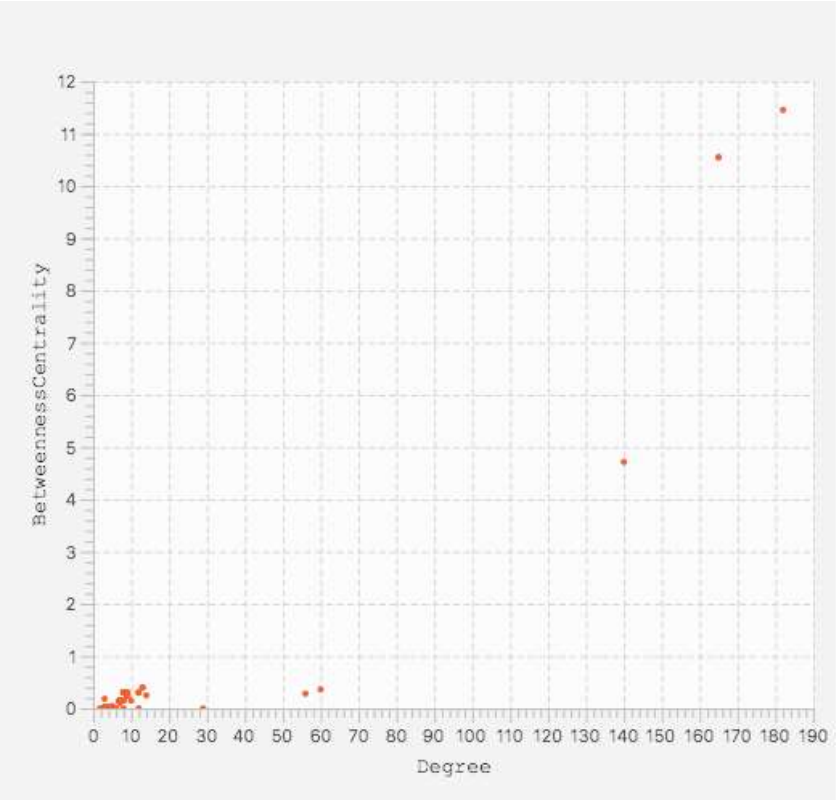

Figure 24: Betweenness Centrality Distribution

6. Pilot 3a: Office building - Operation performance thanks to physical models and IA algorithms

The aim of this section is to describe how the PLATOON semantic data models are used for the dataset of the Pilot 3a. We briefly present the data sources and the semantic data models that are used in this Pilot. We provide an extract of the knowledge graph created by using SPARQL-Generate (detailed in section of Pilot 1a) and some queries.

6.1. Pilot 3a Data Sources

Pilot 3a concerns an office building with a focus on optimizing the HVAC system performance and provides new kind of services (supporting grid management).

- **LAN & WIFI dataset** contains zone information and number of connections every 5 minutes; WIFI connections with location in the building and LAN connections (type of device, mobile phone or other) with location in the building. It will enable to map the building occupancy in real time.
- **Weather dataset** consists of a collection of: (i) real time weather data: air temperature and solar irradiation, and (ii) weather forecast: air temperature and solar irradiation
- **BMS (Building Management System) dataset** contains temperature measurements related to the zone (internal and setpoints), the percentage of the opening valve for heating and cooling, and the gas and electricity consumption for heating and cooling.

6.2 The PLATOON Semantic Data Models Defined in the Pilot 3a Data Sources

The harmonized semantic data of Pilot 3a, defined in T2.3, covered all requirements of the different datasets. The PLATOON semantic data model of Pilot 3a is composed of different modules according to these specific needs:

- **Building ontology** that includes the building**,** its different types and its relationships with the zone and building space.
- **HVAC ontology** that concerns the concepts that are related to the heating, ventilation and air conditioning systems and devices.
- **Sensor ontology** that presents the concept Sensor which is generic and common to different domains and its observations such as Temperature and Airflow sensors.
- **Generic Property ontology** that describes properties of the building and HVAC such as occupancy, area, volume, etc.
- **Electric Power System ontology** that details electric power systems that consume, produce or store electricity.
- **Energy Measure ontology** that concerns the energy (electricity, gas, thermal) consumption and production and its forecasts.
- **Forecasting ontology** that extends the Procedure Execution ontology (pep) and seas forecasting ontology.
- **Weather ontology** that describes different notions related to the weather like humidity, solar insolation, wind speed, wind direction, etc.

Table 4 shows the number of classes and relationships for each ontology used in Pilot 3a

Table 4: Statistics of ontologies used in Pilot 3a

Ontology	#Classes	#Relationships
Building ontology	125	167
HVAC ontology	70	105
Generic Property ontology	86	177
Sensor ontology	679	2,930
Electric Power System ontology	340	753
Energy Measure ontology	189	562
Forecasting ontology	17	17
Weather ontology	703	3,029

6.3 The Pilot 3a PLATOON Knowledge Graph Creation

Figure 25 shows an extract of the knowledge base generated from the dataset of pilot 3a. This knowledge base contains 71,865 triples. The building of CRIGEN-Stains contains three storeys (e.g., we see 2 zones represented by red nodes).

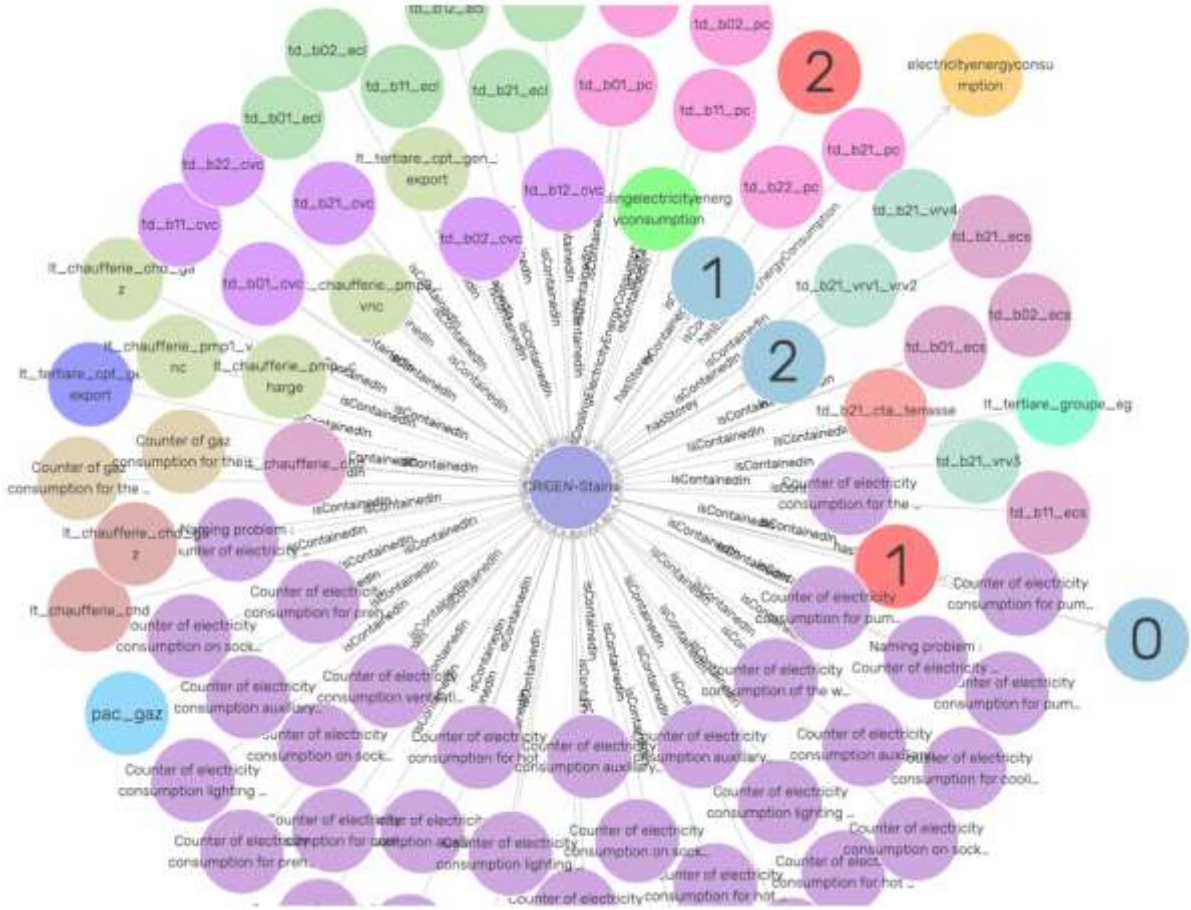

Figure 25: Extract of the Knowledge Base of Pilot 3a

Some competency questions are defined in Pilot 3a, that domain experts want the ontology and the knowledge graph helps to answer:

- What are the sensors installed in the building of CRIGEN-Stains, and what are their types?
- What is the energy consumption of each system contained in a building of CRIGEN-Stains?
- What is the occupancy of a zone and what is the forecast of the occupancy?

Figure 26 shows a SPARQL query that provides an answer to the first question posed.

```
# What are the sensors installed in the building of CRIGEN-Stains,
# and what are their types?
PREFIX bot: <https://w3id.org/bot#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX s4bldg: <https://w3id.org/saref4bldg#>
PREFIX ssn: <http://www.w3.org/ns/ssn/>
PREFIX saref: <https://w3id.org/saref#>
PREFIX seas: <https://w3id.org/seas/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT  ?sensorURI ?propertyType where {
?sensorURI a saref:Sensor;
           rdfs:label ?sensorLabel;
           a ?propertyType.
 ?propertyType rdfs:subClassOf saref:Sensor.
 FILTER not exists {saref:Sensor rdfs:subClassOf+ ?propertyType}
 FILTER (!isBlank(?propertyType))
} ORDER BY ?sensor
```

Figure 26: SPARQL query corresponding to the First natural language question of Pilot 3a

Figure 27 shows an extract of the answer of the above query. Two main sensors are available in the building of CRIGEN-Stains; (i) humidity sensor that measures the humidity in its environment and converts its findings into a corresponding electrical signal, and (ii) temperature sensor which is a device used to measure temperature. Each zone is occupied by these 2 kinds of sensors.

	sensorURI	sensorLabel	typeSensor
1	http://engie.com/data/platoon/building/crigen-stains/zone/1/sensor/humidity	humidity sensor zone1	dogont:HumiditySensor
2	http://engie.com/data/platoon/building/crigen-stains/zone/1/sensor/temperature	temperature sensor zone1	dogont:TemperatureSensor
3	http://engie.com/data/platoon/building/crigen-stains/zone/10/sensor/humidity	humidity sensor zone10	dogont:HumiditySensor
4	http://engie.com/data/platoon/building/crigen-stains/zone/10/sensor/temperature	temperature sensor zone10	dogont:TemperatureSensor
5	http://engie.com/data/platoon/building/crigen-stains/zone/2/sensor/humidity	humidity sensor zone2	dogont:HumiditySensor
6	http://engie.com/data/platoon/building/crigen-stains/zone/2/sensor/temperature	temperature sensor zone2	dogont:TemperatureSensor
7	http://engie.com/data/platoon/building/crigen-stains/zone/3/sensor/humidity	humidity sensor zone3	dogont:HumiditySensor
8	http://engie.com/data/platoon/building/crigen-stains/zone/3/sensor/temperature	temperature sensor zone3	dogont:TemperatureSensor
9	http://engie.com/data/platoon/building/crigen-stains/zone/4/sensor/humidity	humidity sensor zone4	dogont:HumiditySensor
10	http://engie.com/data/platoon/building/crigen-stains/zone/4/sensor/temperature	temperature sensor zone4	dogont:TemperatureSensor
11	http://engie.com/data/platoon/building/crigen-stains/zone/5/sensor/humidity	humidity sensor zone5	dogont:HumiditySensor
12	http://engie.com/data/platoon/building/crigen-stains/zone/5/sensor/temperature	temperature sensor zone5	dogont:TemperatureSensor
13	http://engie.com/data/platoon/building/crigen-stains/zone/6/sensor/humidity	humidity sensor zone6	dogont:HumiditySensor
14	http://engie.com/data/platoon/building/crigen-stains/zone/6/sensor/temperature	temperature sensor zone6	dogont:TemperatureSensor
15	http://engie.com/data/platoon/building/crigen-stains/zone/7/sensor/humidity	humidity sensor zone7	dogont:HumiditySensor
16	http://engie.com/data/platoon/building/crigen-stains/zone/7/sensor/temperature	temperature sensor zone7	dogont:TemperatureSensor
17	http://engie.com/data/platoon/building/crigen-stains/zone/8/sensor/humidity	humidity sensor zone8	dogont:HumiditySensor
18	http://engie.com/data/platoon/building/crigen-stains/zone/8/sensor/temperature	temperature sensor zone8	dogont:TemperatureSensor
19	http://engie.com/data/platoon/building/crigen-stains/zone/9/sensor/humidity	humidity sensor zone9	dogont:HumiditySensor
20	http://engie.com/data/platoon/building/crigen-stains/zone/9/sensor/temperature	temperature sensor zone9	dogont:TemperatureSensor

Figure 27: Answer extract of the above SPARQL query

7. Pilot 4a: Energy Management in microgrids

The goal of this section is to describe how the PLATOON semantic data models are used for the dataset of the Pilot 4a. We briefly present the data sources and the semantic data models that are used in this Pilot. We provide an extract of the knowledge graph created by using SPARQL-Generate (detailed in section of Pilot 1a) and some queries.

7.1. Pilot 4a Data Sources

● **Microgrid PV power production and forecast (MicroGridPVPilot4a)**: consists of forecasting and modeling of Photovoltaic (PV) power. The dataset is expected to grow with more than 30K records per day, and the updates are per minute.
●**Microgrid battery (MicroGridBatteryPilot4a)**: comprises observations of batteries described in terms of State of Charge (SOC), State of Health (SOH), Direct Current (DC), and Alternate Current (AC). Current and voltage are registered, as well as average cell temperature and average ambient temperature. This dataset grows in 86K records per day, and new observations arrive per 1 sec.
● **Microgrid potable water production (MPWPPilot4a)**: contains relevant measurements of a plant for potable water production. The dataset collects active and reactive power values, frequency of pump rotation, feed and permeate water conductivity, concentrate and permeate water flow rate, and temperature and pressure in the hydraulic circuit. It has a growth trend of 1,440 records per day, and updates are per minute.
●**Microgrid weather parameters (MicroGridWeatherStationPilot4a):** consist of observations sensed by a weather station. It reports ambient temperature, wind speed, wind direction, relative humidity, rain, and irradiance. The growth trend is 65K records per day, and observations are registered every 10 seconds.
● **Microgrid full sky imaging (MicroGridFSIPilot4a):** comprises full-sky images in JPEG format. It grows in more than 250 records per day every 5 minutes.

7.2 The PLATOON Semantic Data Models Defined in the Pilot 4a Data Sources

- **Grid ontology** that describes the types of grids (e.g., electrical grid) and their properties.
- **Storage System ontology** that presents different types of storages such as *plt:HydrogenPowerToPowerSystem, plt:OxygenStorageSystem, plt:HydrogenStorageSystem, plt:ThermalStorageSystem* and a *seas:Battery.*
- **Electric Power Transformer ontology** that contains different notions related to the transformer *(s4bldg:Transformer),* its connections with for example other concepts *plt:SecondaryWinding*, *plt:PrimaryWinding*, *plt:Insultation* and *plt:Casing*, etc, and its properties such as active power, reactive power, voltage, etc.
- **Forecasting ontology** that extends the Procedure Execution ontology (pep) and seas forecasting ontology.
- **Weather ontology** that describes different notions related to the weather like humidity, solar insolation, wind speed, wind direction, etc.
- **Sensor ontology** that describes the concept Sensor which is generic and common to different domains and its observations such as Temperature and Humidity sensors.
- **Generic Property ontology** that describes properties of the grid and the storage systems such as percentage of charge, electric power generation capacity, etc.

Table 5 shows the number of classes and relationships for each ontology used in Pilot 4a

Table 5: Statistics of ontologies used in Pilot 4a

Ontology	#Classes	#Relationships
Grid ontology	30	44
Storage System ontology	315	852
Electric Power Transformer ontology	24	31
Generic Property ontology	86	177
Sensor ontology	679	2930
Forecasting ontology	17	17
Weather ontology	703	3029

7.3 The Pilot 4a PLATOON Knowledge Graph Creation

Figure 28 shows an extract of the knowledge base generated from the dataset of pilot **4**a. This knowledge base contains 2,580 triples. The Polytechnic University of Milan has 3 main buildings BL25A, BL27 and B37.

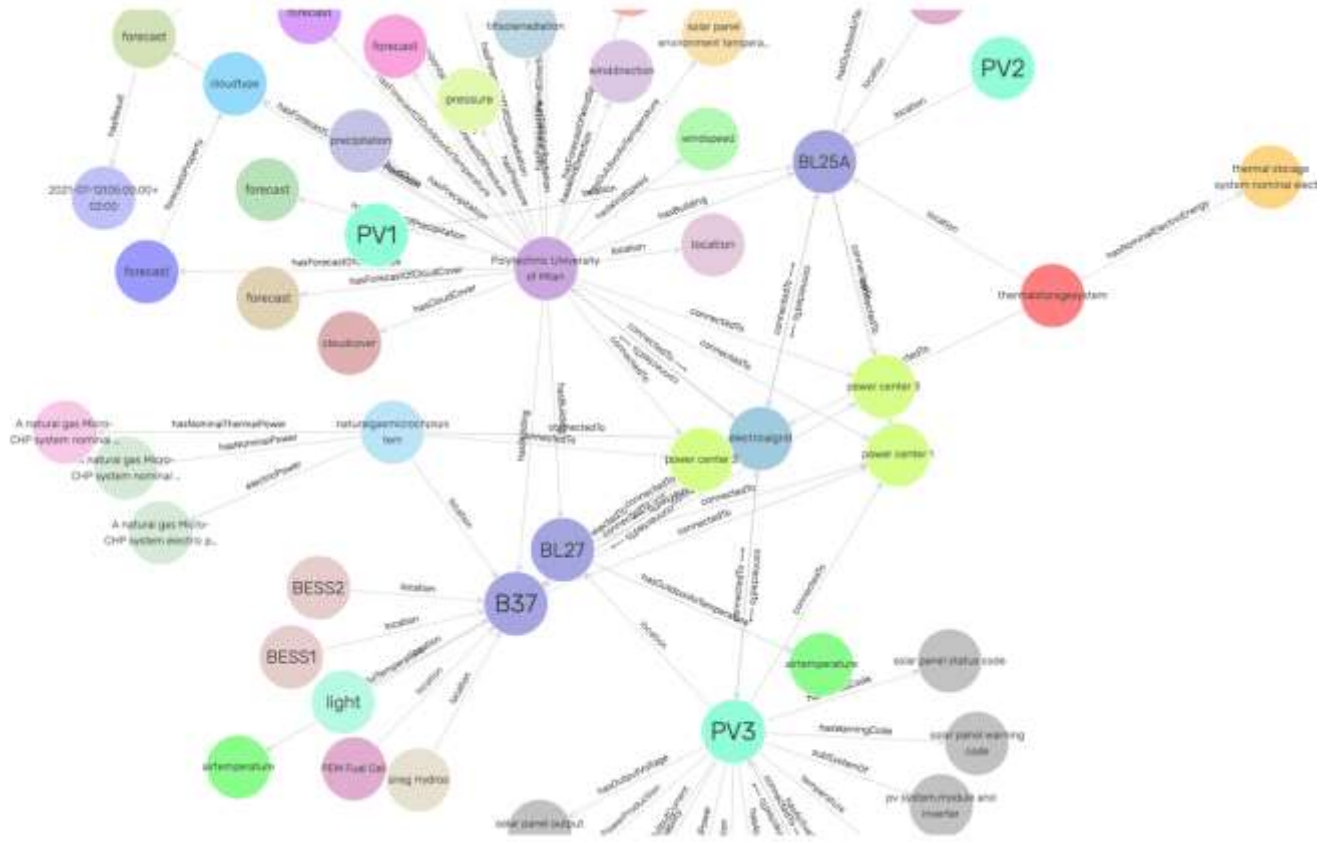

Figure 28: Extract of the Knowledge Base of Pilot 4a

Some competency questions are defined in Pilot 4a, that domain experts want the ontology, and the knowledge graph helps to answer:

- What are the systems that interconnect with microgrid?
- What are the different systems of storage?
- What are all devices and systems/and their measurements related to weather conditions?

- What are all measurements of energy consumption by loads?

Figure 29 shows a SPARQL query that provides an answer to the first question posed.

```
#What are the systems that interconnects with microgrid?

PREFIX plt: <https://w3id.org/platoon/>
PREFIX seas: <https://w3id.org/seas/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select * where {
    ?gridURI a  plt:SmartMicroGrid ;
             seas:connectedTo ?systemURI.
    ?systemURI a ?systemType.
    ?systemType rdfs:subClassOf seas:System.
}
```

Figure 29: SPARQL query corresponding to the First natural language question of Pilot 4a

Figure 30 shows an extract of the answer of the above query. The smart electrical Grid electric power <http://engie.com/platoon/resource/site/polytechnicuniversityofmilan/electricalgrid> that is connected to *Polytechnic University of Milan,* is also connected to a Thermal Storage System (*plt:ThermalStorageSystem*) which is a Storage System (*plt:StorageSystem*).

	gridURI	systemURI	systemType
1	http://engie.com/platoon/resource /site/polytechnicuniversityofmilan/electricalgrid	http://engie.com/platoon/resource /site/polytechnicuniversityofmilan/electricalgrid /thermalstoragesystem	plt:ThermalStorageSystem
2	http://engie.com/platoon/resource /site/polytechnicuniversityofmilan/electricalgrid	http://engie.com/platoon/resource /site/polytechnicuniversityofmilan/electricalgrid /thermalstoragesystem	plt:StorageSystem

Figure 30: Answer Extract of the above SPARQL query

8. Empirical Evaluation of Federated Query Processing

The described federated query engines implement strategies of query processing that facilitate query execution in complex scenarios, e.g., non-selective queries or large knowledge bases. This section reports on an empirical study where these strategies are empirically analyzed; the aim of this study is to answer the following research questions:

- RQ1: What is the impact of query decomposition in federated query processing?
- RQ2: What is the effect of query planning in continuous query answering?

The following experimental study was configured to assess the research questions.

- **Benchmark**: five queries, defined over the knowledge base of Pilot 2a, were defined. Table 6 summarizes the properties of the queries presented appendix B. The queries are of different complexity (e.g., with five or nine triple patterns and up to eight

joins); they retrieve data values of different types (e.g., pressure or humidity) and the answer cardinalities ranges from 14,527 to 1,194,991.

Table 6: Description of Five SPARQL queries included in the empirical study. Q3 and Q4 are non-selective queries, while the other queries are considered selective.

Query	Number of Triple Patterns	Number of Results
Q1	Nine triple patterns to retrieve values of Pressure and the assets that generated the measurements	14,527
Q2	Nine triple patterns to retrieve values of Humidity, the assets that generated the measurements, and their locations	43,680
Q3	Five triple patterns to retrieve values of Active Power Evaluation	1,194,991
Q4	Five triple patterns to retrieve values of Electric Power Evaluation	1,194,991
Q5	Five triple patterns to retrieve values of Humidity measurements and the property holding this information	14,527

- **Federated Query Processing Methods**: three different strategies are compared:
  - **Baseline:** a query is executed over a knowledge base through a SPARQL endpoint. It recreates a query evaluation without the use of a query engine.
  - **Exclusive Groups (EG-FQP)**: the execution of a query is posed over a knowledge base via a SPARQL endpoint. The query planner schedules the execution of the query in blocks according to an estimated cardinality of the answer. This execution resembles the federated query processing strategies implemented in the state-of-the-art SPARQL federated query engine FedEx [12].
  - **Federated Query Processing based on Star-Shaped Groups (FQP)**: a SPARQL query is decomposed into star-shaped subqueries over subjects of the same type. The execution of the subqueries is scheduled in a bushy tree where the leaves of the tree correspond to the subqueries and internal nodes represent physical operators that merge the results produced by the subtrees that correspond to the node children. The star-shaped subqueries are executed in blocks; this decision is taken by the planner based on the selectivity of each subquery. These strategies are part of the PLATOON FQP and also in ANAPSID [11], MULDER [16], and Ontario [13].
- **Metrics**: The three query processing techniques are compared in terms of the following metrics: a) Execution Time: Elapsed time between the submission of a query to a query engine and the generation of the answers. Time corresponds to absolute wall-clock system time as reported by the Python time.time() function. b) Completeness: Query result percentage with respect to the query answer cardinality. c) Diefficiency at time t, dief@t [17], measures the continuous efficiency of an engine in the first t time units of query execution. The Diefficiency metric is described in terms of: Time for the first tuple (TFFT), Total execution time (ET), Number of answers produced (comp), and Throughput (T). dief@t computes AUC (area-under-the-curve) of the answer distribution until time t; a higher value means the query engine has a steadier answer production.

- **Computational Environment**: The pipeline described in Section 5 is executed to create the Pilot 2a knowledge base. All containers run at the same server and network cost is neglected. The knowledge base is accessible via a SPARQL endpoint implemented in Virtuoso 6.01.3127 configured to use up to 16 GiB. The experiments are executed on an Ubuntu 18.04.4 LTS 64 bit machine with an Intel® Xeon® E5-1630v4 CPU (four physical cores, eight threads), and 64 GiB DDR4 RAM.

8.1. Efficiency of the PLATOON Federated Query Processing Strategies

The efficiency of the developed FQP is measured in terms of the time required to produce all the answers to a query (i.e., elapsed time). The ten queries were run using the three federated query engines previously described with a timeout of 10 minutes; all the caches are flushed between the execution of two queries to ensure reproducibility. The block or "paginating" is configured to 10,000 answers. Table 7 reports the elapsed time (secs) to produce the first result of three query execution strategies. The percentage of improvement concerning the baseline approach is also reported. As observed, the speed up to the first result is especially high for executing non-selective queries (i.e., Q3 and Q4) with EG_FQP and the PLATOON federated query engine FQP. The studied federated query processing strategies also speed up the baseline approach for the other queries.

Table 7: Elapsed time (in secs.) to the first answer of five queries executed over Pilot 2a knowledge base. Best results are highlighted in bold. FQP is producing the first answer first for the queries Q1, Q2, Q4, and Q5. EG_FQP produces the first answer for query Q4 slightly earlier than FQP. Both approaches manage to produce results earlier than the baseline.

Query	Baseline (secs.)	EG_FQP		FQP	
		EG-FPR (secs.)	%Speed Up	FQP (secs.)	%Speed Up
Q1	6.12	4.49	26.65%	**4.27**	**30.26%**
Q2	11.85	2.43	79.48%	**0.75**	**93.67%**
Q3	80.08	**0.63**	**99.21%**	0.70	99.11%
Q4	104.69	0.88	99.16%	**0.64**	**99.39%**
Q5	1.33	0.95	28.24%	**0.73**	**45.32%**

8.2. Continuous Behavior of the PLATOON Federated Query Processing Strategies

The baseline produces all the results simultaneously, while EG_FPQ and FQP output answers incrementally. The assessment outcomes of these engines' behavior are visualized in radar plots. Figure 31 explains the meaning of each of the metrics presented in the radar in terms of the trace of the execution of a query (Figure 31a). As observed, the time to generate the first answer is measured (TFFT), and the total execution time is reported in ET; in the radar plot, the inverse values of both metrics are presented. Comp and T represent the percentage of completeness of the produced query answer and throughput. Lastly, dief@t measures the steady generation of the query answer (Figure 31b).

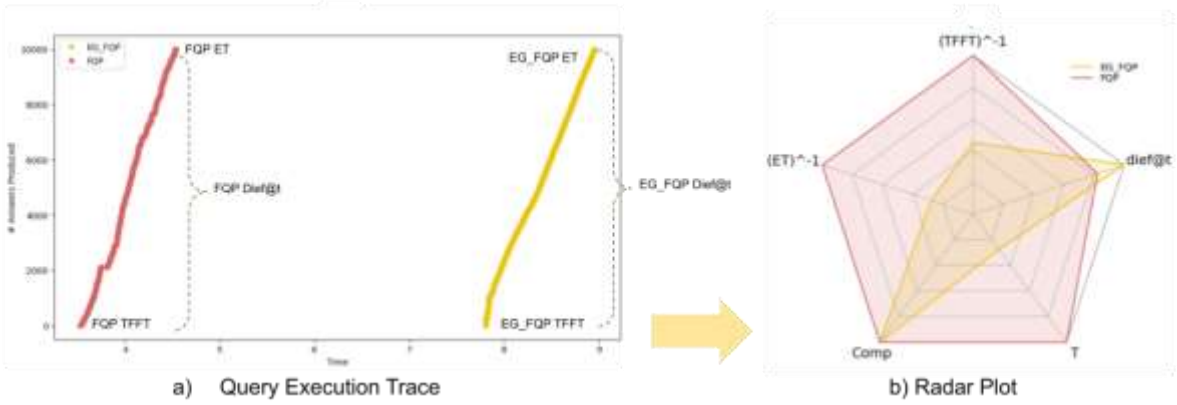

Figure 31: Overview of Result Plots. a) Traces showing the incremental generation of the Q1 answers; b) Diefficiency at time t (dief@t); TFFT: time for the first results, ET: execution time; Total execution time (ET), Number of answers produced (Comp), and Throughput (T).

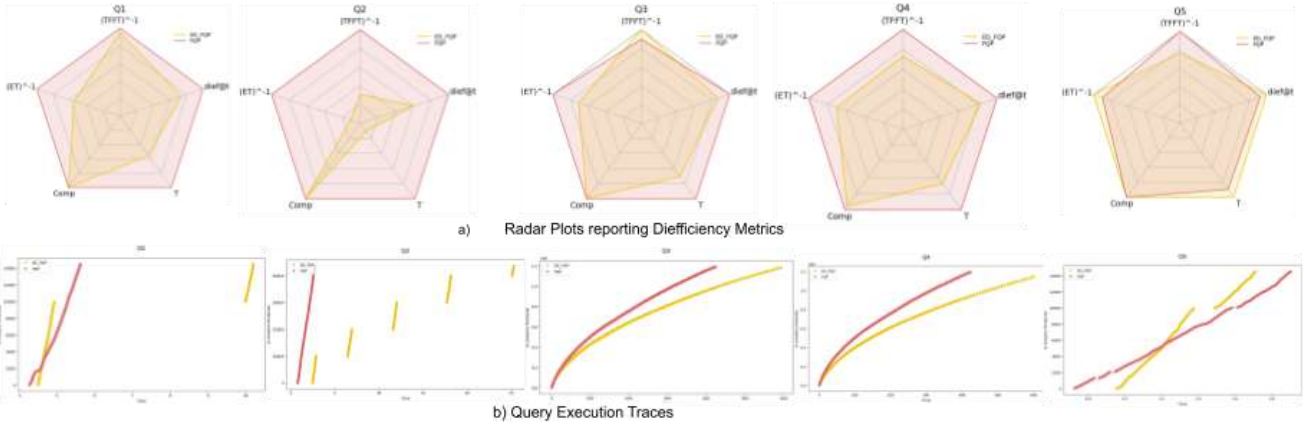

Figure 32: Continuous behavior of EG_FQP and FQP. In all the queries, FQP generates the first answer ahead of EG_FQP and finishes faster. FQP also exhibits a steady answer production even in non-selective queries (i.e., Q3 and Q4).

As can be seen in Figure 32, FQP outperforms EG_FQP in terms of continuous behavior for the queries Q1, Q2, Q3, and Q4. EG_FQP performs slightly better for query Q5. Q5 is a very simple query, hence, splitting the query into sub-queries adds some overhead. The traces for Q1 and Q2 show the continuous answer generation of FQP while EG_FQP generates the answers in portions. For Q3 and Q4 both approaches show a steady generation of the query results even though the answer generation rate drops a little after the first 200,000 results. However, FQP maintains a higher rate than EG_FQP.

9. Conclusion and Future Work

This document reports on the outcomes of performing task T2.4 – Data Integration of WP2, conducted during M25 and M27 of the PLATOON project. The reported results have allowed for the understanding of the role played by the data management techniques implemented in the PLATOON framework. These techniques enable the integration of data sources in diverse formats (e.g., CSV, JSON, RDB), and managed by various database systems (e.g., MySQL or MongoDB). Additionally, the reported statistics facilitate the understanding of the amount of represented knowledge and the opportunities that it offers for knowledge exploration and discovery. These statistics also describe the relations that have been established in the PLATOON knowledge bases as a result of data integration.

The pipelines described in this document, have been integrated into the PLATOON framework in the context of WP5. This integration enables the connection of the PLATOON knowledge bases with analytical methods implemented in the SANSA-Stack's. SANSA[9] is a processing data flow engine that provides data distribution, and fault tolerance for distributed computations over large-scale RDF knowledge bases. The knowledge bases created in the context of T2.4 are being utilized in the work packages WP4 and WP6 to develop analytical toolboxes which will enable the analytical requirements of Pilots 1a, 2a, 3a, and 4a.

[9] https://sda.tech/projects/sansa-stack/

Appendix A - Classes of the PLATOON Semantic Data Models

Pilot 1a

Table 8: PLATOON Semantic Data Models in Pilot 1a

Data Source	Column/attribute name	Semantic entity: subject	Semantic entity: predicate	Semantic entity: object
PLATOON_Shared SCADA data E3F 2013	ACTIVE POWER	plt:WindTurbine	plt:hasActivePower	seas:ElectricPowerProperty
PLATOON_Shared SCADA data E3F 2013	ACTIVE POWER REFERENCE	plt:WindTurbine	plt:hasActivePowerReference	seas:ElectricPowerProperty
PLATOON_Shared SCADA data E3F 2013	POWER FACTOR	plt:WindTurbine	plt:hasPowerFactor	plt:PowerFactor
PLATOON_Shared SCADA data E3F 2013	REACTIVE POWER	plt:WindTurbine	plt:hasReactivePower	seas:ElectricPowerProperty
PLATOON_Shared SCADA data E3F 2013	PHASE VOLTAGE A	plt:Generator	seas:RNVoltage	seas:VoltageProperty
PLATOON_Shared SCADA data E3F 2013	PHASE VOLTAGE B	plt:Generator	seas:SNVoltage	seas:VoltageProperty
PLATOON_Shared SCADA data E3F 2013	PHASE VOLTAGE C	plt:Generator	seas:TNVoltage	seas:VoltageProperty
PLATOON_Shared SCADA data E3F 2013	PHASE CURRENT A	plt:Generator	seas:rCurrent	seas:CurrentProperty
PLATOON_Shared SCADA data E3F 2013	PHASE CURRENT B	plt:Generator	seas:sCurrent	seas:CurrentProperty
PLATOON_Shared SCADA data E3F 2013	PHASE CURRENT C	plt:Generator	seas:tCurrent	seas:CurrentProperty
PLATOON_Shared SCADA data E3F 2013	GENERATOR SPEED	plt:Generator	plt:hasRotationalSpeed	plt:RotationalSpeedProperty
PLATOON_Shared SCADA data E3F 2013	HUB SPEED – to check if Hub Speed or Rotor Speed	plt:Rotor	plt:hasRotationalSpeed	plt:RotationalSpeedProperty
PLATOON_Shared SCADA data E3F 2013	BLADE A SET VALUE	plt:Blade	plt:hasPosition	plt:PositionProperty
PLATOON_Shared SCADA data E3F 2013	WIND SPEED 1	plt:Anenometer	seas:observesProperty	seas:WindSpeedProperty
PLATOON_Shared SCADA data E3F 2013	NACELLE POSITION	plt:Nacelle	plt:hasPosition	plt:PositionProperty
PLATOON_Shared SCADA data E3F 2013	GENERATOR SPEED 2	plt:Generator	plt:hasRotationalSpeed	plt:RotationalSpeedProperty
PLATOON_Shared SCADA data E3F 2013	GENERATOR WINDING TEMPERATURE A	plt:GeneratorWinding	saref:temperature	saref:Temperature
PLATOON_Shared SCADA data E3F 2013	GENERATOR DE BEARING TEMPERATURE	plt:GeneratorBearing	saref:temperature	saref:Temperature
PLATOON_Shared SCADA data E3F	NACELLE TEMPERATUR	plt:Nacelle	saref:temperature	saref:Temperature

2013	E 1			
PLATOON_Shared SCADA data E3F 2013	TOWER OSCILLATION	plt:Tower	plt:hasVibration	plt:VibrationProperty
PLATOON_Shared SCADA data E3F 2013	TRANSFORMER TEMPERATURE 1	plt:Transformer	saref:temperature	saref:Temperature
PLATOON_Shared SCADA data E3F 2013	GEARBOX BEARING 1 GLOBAL VIBRATION	plt:Gearbox	plt:hasVibration	plt:VibrationProperty
80743_08-alarms-events	parc_code	plt:WindFarm	rdfs:label	xsd:String
80743_08-alarms-events	mac_code	plt:OnshoreWindTurbine	rdfs:label	xsd:String
80743_08-alarms-events		plt:OnshoreWindTurbine	seas:isMemberOf	plt:WindFarm
80743_08-alarms-events		plt:OnshoreWindTurbine	plt:hasStatusCode	plt:StatusCodeProperty
80743_08-alarms-events		plt:StatusCodeProperty	seas:evaluation	plt:StatusCodeEvaluation
80743_08-alarms-events	date	plt:StatusCodeEvaluation	seas:hasTemporalContext	time:Instant
80743_08-alarms-events	statusCode.code	plt:StatusCodeEvaluation	plt:hasRenewableEnergyProductionStatus	RenewableEnergyProductionStatus
80743_08-alarms-events	statusCode.type	plt:StatusCodeEvaluation	plt:hasStatusCodetype	xsd:interger
80743_08-alarms-events	statusCode.phase	plt:StatusCodeEvaluation	plt:hasStatusCodePhase	xsd:string
static-data (wind farm)		plt:WindFarm	rdfs:label	xsd:string
static-data		plt:WindFarm	plt:isConnected	xsd:boolean
static-data		plt:WindFarm	plt:hasRatedPower	seas:ElectricPowerProperty
static-data		plt:WindFarm	geo:location	geo:Point
static-data		plt:WindFarm	gsp:hasGeometry	gsp:Geometry
static-data		seas:ElectricPowerProperty	seas:simpleValue	xsd:float
static-data		geo:Point	geo:lat	xsd:float
static-data		geo:Point	geo:long	xsd:float
static-data		gsp:Geometry	gsp:asWKT	gsp:wktLiteral
static-data (wind turbine)		plt:OnshoreWindTurbine	rdfs:label	xsd:string
static-data		plt:OnshoreWindTurbine	seas:isMemberOf	plt:WindFarm
static-data		plt:OnshoreWindTurbine	brick:hasLocation	plt:WindFarm
static-data		plt:OnshoreWindTurbine	seas:connectedTo	plt:Substation
static-data		plt:OnshoreWindTurbine	plt:hasRatedPower	seas:ElectricPowerProperty
static-data		plt:OnshoreWindTurbine	geo:location	geo:Point
static-data		plt:OnshoreWindTurbine	sch:model	xsd:string
static-data		plt:OnshoreWindTurbine	gsp:hasGeometry	gsp:Geometry
static-data (substation)		plt:Substation	rdfs:label	xsd:string
static-data		plt:Substation	brick:hasLocation	plt:WindFarm
static-data		plt:Substation	saref:hasState	saref:State
static-data		plt:Substation	plt:hasRatedPower	seas:ElectricPowerProperty
static-data		plt:Substation	geo:location	geo:Point
static-data		plt:Substation	plt:hasContractOperationDate	xsd:datetime
static-data		plt:Substation	gsp:hasGeometry	gsp:Geometry

static-data		geo:Point	sch:addressCountry	sch:Country
static-data		sch:Country	rdfs:label	xsd:string
static-data – (wind turbine components from wind turbine data)		plt:Anenometer	rdfs:label	xsd:string
static-data		plt:Anenometer	seas:subSystemOf	plt:OnshoreWindTurbine
static-data		plt:Nacelle	seas:subSystemOf	plt:OnshoreWindTurbine
static-data		plt:BottomBox	seas:subSystemOf	plt:Nacelle
static-data		plt:Blade	seas:subSystemOf	plt:OnshoreWindTurbine
static-data		plt:Hub	seas:connectedTo	plt:Blade
static-data		plt:Hub	seas:connectedTo	plt:PitchSystem
static-data		plt:PitchSystem	s4bldg:isContainedIn	plt:Hub
static-data		plt:PitchSystem	seas:hasSubSystem	plt:Motor
static-data		plt:CableUtil	seas:connectedTo	plt:OnshoreWindTurbine
static-data		plt:Converter	seas:subSystemOf	plt:OnshoreWindTurbine
static-data		plt:CPU_And_DisplayUnit	s4bldg:isContainedIn	plt:Nacelle
static-data		plt:DriveTrain	seas:subSystemOf	plt:Nacelle
static-data		plt:Gearbox	s4bldg:isContainedIn	plt:Nacelle
static-data		brick:Oil	s4bldg:isContainedIn	plt:Gearbox
static-data		plt:GearboxBearing	seas:subSystemOf	plt:Gearbox
static-data		plt:ElectricalGrid	seas:connectedTo	plt:OnshoreWindTurbine
static-data		plt:Rotor	seas:subSystemOf	plt:Generator
static-data		plt:RotorBearing	seas:subSystemOf	plt:Rotor
static-data		plt:Stator	seas:subSystemOf	plt:Generator
static-data		plt:Stator	seas:connectedTo	plt:Rotor
static-data		plt:StatorWinding	seas:subSystemOf	plt:Stator
static-data		plt:TopBox	seas:subSystemOf	plt:Nacelle
static-data		plt:Vane	seas:subSystemOf	plt:OnshoreWindTurbine
91837_11-scada-data	parc_code	plt:WindFarm	rdfs:label	xsd:String
91837_11-scada-data	mac_code	plt:OnshoreWindTurbine	rdfs:label	xsd:String
91837_11-scada-data		plt:OnshoreWindTurbine	brick:hasLocation	plt:WindFarm
Maintenance_extract.csv		time:Interval	time:hasBeginning	time:Instant
Maintenance_extract.csv		time:Interval	time:hasEnd	time:Instant
Maintenance_extract.csv		time:Instant	time:date	xsd:date
Maintenance_extract.csv		plt:SlipRing	seas:subSystemOf	plt:Rotor
Maintenance_extract.csv		plt:GeneratorFan	seas:subSystemOf	plt:Generator
Maintenance_extract.csv		saref:TemperatureSensor	s4bldg:isContainedIn	plt:Rotor
Maintenance_extract.csv		plt:InsulatedGateBipolarTransistor	seas:subSystemOf	plt:Converter
Maintenance_extract.csv		plt:InsulatedGateBipolarTransistorRack	seas:subSystemOf	plt:InsulatedGateBipolarTransistor

Pilot 2a

Table 9: PLATOON Semantic Data Models in Pilot 2a

Class PLATOON Semantic Data Models	Pilot 2a Data Source
schema:Organization	platoon:PUPIN-ENTSO-E
sopropi:UVIndex	platoon:PUPIN-WeatherBit
cim:ActivePower	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
cim:Asset	platoon:PUPIN-ENTSO-E
cim:ControlArea	platoon:PUPIN-ENTSO-E
cim:ControlAreaOperator	platoon:PUPIN-ENTSO-E
cim:GeneratingUnit	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
cim:MktParticipant	platoon:PUPIN-ENTSO-E
cim:Organization	platoon:PUPIN-ENTSO-E
cim:Plant	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
cim:SolarGeneratingUnit	platoon:PUPIN-RES-PV
cim:WindGeneratingUnit	platoon:PUPIN-RES-PROD
cim:WindPlantIEC	platoon:PUPIN-RES-PROD
ws:Pressure	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
ws:WindDirection	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
ws:WindTurbine	platoon:PUPIN-RES-PROD
time:Instant	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
	platoon:PUPIN-ENTSO-E
time:Interval	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
time:TemporalEntity	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
	platoon:PUPIN-ENTSO-E
saref:Power	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
schema:Country	platoon:PUPIN-ENTSO-E

platoon:AccumulatedPrecipitationEvaluation	platoon:PUPIN-WeatherBit
platoon:AccumulatedPrecipitationProperty	platoon:PUPIN-WeatherBit
platoon:AccumulatedSnowFallEvaluation	platoon:PUPIN-WeatherBit
platoon:AccumulatedSnowFallProperty	platoon:PUPIN-WeatherBit
platoon:ActivePowerEvaluation	platoon:PUPIN-RES-PROD
platoon:AirTemperatureEvaluation	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
platoon:AirTemperatureProperty	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
platoon:CloudCoverageEvaluation	platoon:PUPIN-WeatherBit
platoon:CloudCoverageProperty	platoon:PUPIN-WeatherBit
platoon:DewPointEvaluation	platoon:PUPIN-WeatherBit
platoon:DewPointProperty	platoon:PUPIN-WeatherBit
platoon:DiffuseSolarRadiationEvaluation	platoon:PUPIN-WeatherBit
platoon:DiffuseSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:DirectSolarRadiationEvaluation	platoon:PUPIN-WeatherBit
platoon:DirectSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfAccumulatedPrecipitationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfAccumulatedSnowFallEvaluation	platoon:PUPIN-WeatherBit
platoon:ForecastOfAccumulatedSnowFallProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfActivePower	platoon:PUPIN-RES-PROD
platoon:ForecastOfActivePowerEvaluation	platoon:PUPIN-RES-PROD
platoon:ForecastOfAirTemperatureEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfAirTemperatureProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfAverageOzoneProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfCloudCoverageProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfCloudEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfDewPointEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfDewPointProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfDiffuseSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfDirectSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfElectricPowerEvaluation	platoon:PUPIN-RES-PROD
platoon:ForecastOfElectricProductionProperty	platoon:PUPIN-RES-PROD
platoon:ForecastOfGlobalSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfHighAirTemperatureEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfHighAirTemperatureProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfHighLevelCloudEvaluation	platoon:PUPIN-ENTSO-E

platoon:ForecastOfHighLevelCloudProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfLowAirTemperatureProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfLowLevelCloudEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfLowLevelCloudProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfMaxAirTemperatureEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfMaxAirTemperatureProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfMaxDiffuseSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfMidLevelCloudEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfMidLevelCloudProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfMinAirTemperatureEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfMinAirTemperatureProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfMoonriseProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfMoonsetProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfProbabilityOfPrecipitationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfRelativeHumidityProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfSeaLevelPressureEvaluation	platoon:PUPIN-ENTSO-E
platoon:ForecastOfSeaLevelPressureProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfSnowDepthProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfSolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfSunriseProperty	platoon:PUPIN-WeatherBit
	platoon:PUPIN-WeatherBit
platoon:ForecastOfUVIndexProperty	platoon:PUPIN-WeatherBit
platoon:ForecastOfVisibilityProperty	platoon:PUPIN-WeatherBit
platoon:OffshoreWindTurbine	platoon:PUPIN-RES-PROD
platoon:RelativeHumidityEvaluation	platoon:PUPIN-WeatherBit
platoon:RelativeHumidityProperty	platoon:PUPIN-WeatherBit
platoon:SeaLevelPressureEvaluation	platoon:PUPIN-WeatherBit
platoon:SeaLevelPressureProperty	platoon:PUPIN-WeatherBit
platoon:SolarElevationAngleEvaluation	platoon:PUPIN-WeatherBit
platoon:SolarElevationAngleProperty	platoon:PUPIN-WeatherBit
platoon:SolarHourAngleProperty	platoon:PUPIN-WeatherBit
platoon:SolarInsolationEvaluation	platoon:PUPIN-RES-PV
platoon:SolarInsolationProperty	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
platoon:SolarInverter	platoon:PUPIN-RES-PV
platoon:SolarRadiationEvaluation	platoon:PUPIN-WeatherBit
platoon:SolarRadiationProperty	platoon:PUPIN-WeatherBit
platoon:SunriseProperty	platoon:PUPIN-WeatherBit
platoon:SunsetEvaluation	platoon:PUPIN-WeatherBit

platoon:SunsetProperty	platoon:PUPIN-WeatherBit
platoon:UVIndexEvaluation	platoon:PUPIN-WeatherBit
platoon:UVIndexProperty	platoon:PUPIN-WeatherBit
platoon:VisibilityEvaluation	platoon:PUPIN-WeatherBit
platoon:VisibilityProperty	platoon:PUPIN-WeatherBit
platoon:WeatherStation	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
platoon:WindFarm	platoon:PUPIN-RES-PROD
seas:AngleEvaluation	platoon:PUPIN-WeatherBit
seas:AngleProperty	platoon:PUPIN-WeatherBit
seas:ElectricPowerEvaluation	platoon:PUPIN-RES-PROD
seas:ElectricPowerProducer	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
seas:ElectricPowerProperty	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
seas:ElectricPowerSystem	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
seas:FeatureOfInterest	platoon:PUPIN-RES-PV
seas:FeatureOfInterest	platoon:PUPIN-RES-PROD
seas:Forecast	platoon:PUPIN-WeatherBit
seas:ForecastOfPressureEvaluation	platoon:PUPIN-ENTSO-E
seas:ForecastOfPressureProperty	platoon:PUPIN-WeatherBit
seas:ForecastOfWindDirectionEvaluation	platoon:PUPIN-WeatherBit
seas:ForecastOfWindDirectionProperty	platoon:PUPIN-WeatherBit
seas:ForecastOfWindGustSpeedEvaluation	platoon:PUPIN-WeatherBit
seas:ForecastOfWindGustSpeedProperty	platoon:PUPIN-WeatherBit
seas:ForecastOfWindSpeedEvaluation	platoon:PUPIN-WeatherBit
seas:ForecastOfWindSpeedProperty	platoon:PUPIN-WeatherBit
seas:HumidityEvaluation	platoon:PUPIN-WeatherBit
seas:HumidityProperty	platoon:PUPIN-WeatherBit
seas:PressureEvaluation	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
seas:PressureProperty	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
seas:SolarArray	platoon:PUPIN-RES-PV
seas:SolarPanel	platoon:PUPIN-RES-PV
seas:SolarRadiationEvaluation	platoon:PUPIN-WeatherBit
seas:SolarRadiationProperty	platoon:PUPIN-WeatherBit

seas:TemperatureEvaluation	platoon:PUPIN-WeatherBit
seas:TemperatureProperty	platoon:PUPIN-WeatherBit
seas:WindDirectionEvaluation	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
seas:WindDirectionProperty	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
seas:WindSpeedEvaluation	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
seas:WindSpeedProperty	platoon:PUPIN-RES-PV
	platoon:PUPIN-RES-PROD
	platoon:PUPIN-WeatherBit
seas:Zone	platoon:PUPIN-RES-PV
geo:Feature	platoon:PUPIN-ENTSO-E

Pilot 3a

Table 10: PLATOON Semantic Data Models in Pilot 3a

Data Source	Column/attribute name	Semantic Entity: subject	Semantic entity: predicate	Semantic entity: Object
lan-data-202105_extract.json	CRIGEN-STAINS	bot:Building	bot:containsZone	bot:Zone
lan-data-202105_extract.json	PORT	bot:Zone	plt:hasPort	Value (String)
lan-data-202105_extract.json	CONNECTIONS	bot:Zone	plt:hasOccupancy	saref:Occupancy
lan-data-202105_extract.json	CONNECTIONS	saref:Occupancy	seas:isPropertyOf	bot:Zone
lan-data-202105_extract.json	CONNECTIONS	saref:Occupancy	seas:evaluation	plt:OccupiedNumberEvaluation
lan-data-202105_extract.json	CONNECTIONS	plt:OccupiedNumberEvaluation	seas:hasTemporalContext	time:Instant
lan-data-202105_extract.json	CONNECTIONS	plt:OccupiedNumberEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-estimated-actuals_.json	cloud_opacity	bot:Building	plt:hasCloudOpacity	plt:CloudOpacityProperty
weather-estimated-actuals_.json	ghi	bot:Building	plt:hasGlobalHorizontalIrradiance	plt:SolarRadiationProperty
weather-estimated-actuals_.json	ebh	bot:Building	plt:hasDirectHorizontalIrradiance	plt:DirectSolarRadiationProperty

weather-estimated-actuals_.json	dhi	bot:Building	plt:hasDiffuseHorizontalIrradiance	plt:DiffuseSolarRadiationProperty
weather-estimated-actuals_.json	dni	bot:Building	plt:hasDirectNormalIrradiance	plt:DirectSolarRadiationProperty
weather-estimated-actuals_.json	cloud_opacity	plt:CloudOpacityProperty	seas:evaluation	plt:CloudOpacityEvaluation
weather-estimated-actuals_.json	period_end	plt:CloudOpacityEvaluation	seas:hasTemporalContext	time:Instant
weather-estimated-actuals_.json	cloud_opacity	plt:CloudOpacityEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-estimated-actuals_.json	period	plt:CloudOpacityEvaluation	time:duration	xsd:dateTime
weather-estimated-actuals_.json	ghi	plt:SolarRadiationProperty	seas:evaluation	plt:SolarRadiationEvaluation
weather-estimated-actuals_.json	period_end	plt:SolarRadiationEvaluation	seas:hasTemporalContext	time:Instant
weather-estimated-actuals_.json	ghi	plt:SolarRadiationEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-estimated-actuals_.json	period	plt:SolarRadiationEvaluation	time:duration	xsd:dateTime
weather-estimated-actuals_.json	ebh	plt:DirectSolarRadiationProperty	seas:evaluation	plt:DirectSolarRadiationEvaluation
weather-estimated-actuals_.json	period_end	plt:DirectSolarRadiationEvaluation	seas:hasTemporalContext	time:Instant
weather-estimated-actuals_.json	ebh	plt:DirectSolarRadiationEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-estimated-actuals_.json	period	plt:DirectSolarRadiationEvaluation	time:duration	xsd:dateTime
weather-estimated-actuals_.json	dhi	plt:DiffuseSolarRadiationProperty	seas:evaluation	plt:DiffuseSolarRadiationEvaluation
weather-estimated-actuals_.json	period_end	plt:DiffuseSolarRadiationEvaluation	seas:hasTemporalContext	time:Instant
weather-estimated-actuals_.json	dhi	plt:DiffuseSolarRadiationEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-estimated-actuals_.json	period	plt:DiffuseSolarRadiationEvaluation	time:duration	xsd:dateTime
weather-estimated-actuals_.json	dni	plt:DirectSolarRadiationProperty	seas:evaluation	plt:DirectSolarRadiationEvaluation
weather-estimated-actuals_.json	period_end	plt:DirectSolarRadiationEvaluation	seas:hasTemporalContext	time:Instant
weather-estimated-actuals_.json	dni	plt:DirectSolarRadiationEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-estimated-actuals_.json	period	plt:DirectSolarRadiationEvaluation	time:duration	xsd:dateTime
weather-forecast_extract.json	air_temp	bot:Building	plt:hasForecastOfOutdoorAirTemperature	plt:ForecastOfAirTemperatureProperty

weather-forecast_extract.json	cloud_opacity	bot:Building	plt:hasForecastOfCloudOpacity	plt:ForecastOfCloudOpacityProperty
weather-forecast_extract.json	air_temp	plt:ForecastOfAirTemperatureProperty	seas:forecastsProperty	plt:AirTemperatureProperty
weather-forecast_extract.json	air_temp	plt:ForecastOfAirTemperatureProperty	pep:hasResult	plt:AirTemperatureEvaluation, seas:Forecast
weather-forecast_extract.json	period_end	plt:AirTemperatureEvaluation	seas:hasTemporalContext	time:Instant
weather-forecast_extract.json	air_temp	plt:AirTemperatureEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-forecast_extract.json	period	plt:AirTemperatureEvaluation	time:duration	period (date time)
weather-forecast_extract.json	cloud_opacity	plt:ForecastOfCloudOpacityProperty	seas:forecastsProperty	plt:CloudOpacityProperty
weather-forecast_extract.json	cloud_opacity	plt:ForecastOfCloudOpacityProperty	pep:hasResult	plt:CloudOpacityEvaluation, seas:Forecast
weather-forecast_extract.json	period_end	plt:CloudOpacityEvaluation	seas:hasTemporalContext	time:Instant
weather-forecast_extract.json	cloud_opacity	plt:CloudOpacityEvaluation	seas:evaluatedSimpleValue	xsd:float
weather-forecast_extract.json	period	plt:CloudOpacityEvaluation	time:duration	xsd:dateTime
wifi-data_extract.json		bot:Zone	plt:hasPort	xsd:String
wifi-data_extract.json		bot:Zone	plt:hasOccupancy	saref:Occupancy
wifi-data_extract.json		saref:Occupancy	seas:isPropertyOf	bot:Zone
wifi-data_extract.json		saref:Occupancy	seas:evaluation	plt:OccupiedNumberEvaluation
wifi-data_extract.json		plt:OccupiedNumberEvaluation	prov:wasGeneratedBy	prov:Activity, pep:ProcedureExecution
wifi-data_extract.json		prov:Activity, pep:ProcedureExecution	rdfs:label	xsd:String
bms_part1.json		plt:HVACValveController	s4bldg:isContainedIn	bot:Zone
bms_part1.json		plt:HVACValveController	seas:connectedTo	saref:TemperatureSensor
bms_part1.json		plt:HVACValveController	brick:controls	plt:HeatingValve
bms_part1.json		plt:HVACValveController	brick:controls	plt:CoolingValve
bms_part2.json		bot:Building	bot:hasStorey	bot:Floor
bms_part2.json		bot:Floor	seas:subZoneOf	bot:Building
bms_part2.json		seas:Zone	seas:subZoneOf	bot:Floor
bms_part2.json		seas:ElectricityMeter OR	s4bldg:isContainedIn	bot:Floor, bot:Zone

		seas:GasMeter		
bms_part2.json		seas:ElectricityMeter OR seas:GasMeter	seas:measuresProperty	plt:HeatingGasEnergyConsumptionProperty OR plt:ElectricityEnergyConsumptionProperty
bms_part2.json		seas:ElectricityMeter OR seas:GasMeter	rdfs:label	xsd:String
bms_part2.json		plt:HeatingGasEnergyConsumptionEvaluation OR plt:ElectricityEnergyConsumptionEvaluation	seas:evaluationOf	plt:HeatingGasEnergyConsumptionProperty OR plt:ElectricityEnergyConsumptionProperty
bms_part2.json		plt:HeatingGasEnergyConsumptionEvaluation OR plt:ElectricityEnergyConsumptionEvaluation	seas:hasTemporalContext	time:Instant
bms_part2.json		plt:HeatingGasEnergyConsumptionEvaluation OR plt:ElectricityEnergyConsumptionEvaluation	seas:evaluatedSimpleValue	xsd:float

Pilot 4a

Table 11: PLATOON Semantic Data Models in Pilot 4a

Data Source	Semantic Entity: subject	Semantic entity: predicate	Semantic entity: Object
weather-data-sample.json	seas:EducationalBuilding	geo:location	geo:Point
weather-data-sample.json	seas:EducationalBuilding	plt:hasAirTemperature	plt:AirTemperatureProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfOutdoorAirTemperature	plt:ForecastOfAirTemperatureProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasHorizontalSolarRadiation	plt:SolarRadiationProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfHorizontalSolarRadiation	plt:ForecastOfSolarRadiationProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasTiltSolarRadiation	plt:SolarRadiationProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfTiltSolarRadiation	plt:ForecastOfSolarRadiationProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasWindSpeed	seas:WindSpeedProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfWindSpeed	plt:ForecastOfWindSpeedProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasPressure	seas:PressureProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfPressure	plt:ForecastOfPressureProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasWindDirection	seas:WindDirectionProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfWindDirection	plt:ForecastOfWindDirectionProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasPrecipitation	plt:PrecipitationProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfPrecipitation	plt:ForecastOfPrecipitationPropert
weather-data-sample.json	seas:EducationalBuilding	plt:hasCloudCover	plt:CloudCoverProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfCloudCover	plt:ForecastOfCloudCoverProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasCloudType	plt:CloudTypeProperty
weather-data-sample.json	seas:EducationalBuilding	plt:hasForecastOfCloudType	plt:ForecastOfCloudTypeProperty
weather-data-sample.json	seas:EducationalBuilding	gsp:hasGeometry	gsp:Geometry

weather-data-sample.json	plt:ForecastOfAir TemperatureProperty	seas:forecastsProperty	plt:AirTemperature Property
weather-data-sample.json	plt:ForecastOfAir TemperatureProperty	pep:hasResult	plt:AirTemperature Evaluation, seas:Forecast
weather-data-sample.json	plt:AirTemperatureEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	plt:AirTemperatureEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOfSolar RadiationProperty	seas:forecastsProperty	plt:SolarRadiationProperty
weather-data-sample.json	plt:ForecastOfSolar RadiationProperty	pep:hasResult	plt:SolarRadiation Evaluation, s eas:Forecast
weather-data-sample.json	plt:SolarRadiationEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	plt:SolarRadiationEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOfWind SpeedProperty	seas:forecastsProperty	seas:WindSpeedProperty
weather-data-sample.json	plt:ForecastOfWind SpeedProperty	pep:hasResult	seas:WindSpeed Evaluation, seas:Forecast
weather-data-sample.json	seas:WindSpeedEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	seas:WindSpeedEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOf PressureProperty	seas:forecastsProperty	seas:PressureProperty
weather-data-sample.json	plt:ForecastOf PressureProperty	pep:hasResult	seas:Pressure Evaluation, seas:Forecast
weather-data-sample.json	seas:PressureEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	seas:PressureEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOfWind DirectionProperty	seas:forecastsProperty	seas:WindDirection Property
weather-data-sample.json	plt:ForecastOfWind DirectionProperty	pep:hasResult	seas:Wind DirectionEvaluation, seas:Forecast
weather-data-sample.json	seas:WindDirectionEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	seas:WindDirectionEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOf PrecipitationProperty	seas:forecastsProperty	plt:Precipitation Property
weather-data-sample.json	plt:ForecastOf PrecipitationProperty	pep:hasResult	plt:Precipitation Evaluation, seas:Forecast
weather-data-sample.json	plt:PrecipitationEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	plt:PrecipitationEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOf CloudCoverProperty	seas:forecastsProperty	plt:CloudCover Property
weather-data-sample.json	plt:ForecastOf CloudCoverProperty	pep:hasResult	plt:CloudCover Evaluation, seas:Forecast
weather-data-sample.json	plt:CloudCoverEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	plt:CloudCoverEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float
weather-data-sample.json	plt:ForecastOf CloudTypeProperty	seas:forecastsProperty	plt:Cloud TypeProperty
weather-data-sample.json	plt:ForecastOf CloudTypeProperty	pep:hasResult	plt:CloudT ypeEvaluation,

			seas:Forecast
weather-data-sample.json	plt:CloudTypeEvaluation, seas:Forecast	seas:hasTemporalContext	time:Instant
weather-data-sample.json	plt:CloudTypeEvaluation, seas:Forecast	seas:evaluatedSimpleValue	xsd:float

Appendix B – Queries over the Knowledge Base of Pilot 2a

Query 1

```
SELECT DISTINCT ?pressure ?pressureValue ?feature
WHERE {
  ?pressure a <https://w3id.org/seas/PressureEvaluation> .
  ?pressure <https://w3id.org/seas/evaluatedSimpleValue> ?pressureValue .
  ?pressure <https://w3id.org/seas/hasTemporalContext> ?tempContext .

  ?pressureProperty a <https://w3id.org/seas/PressureProperty> .
  ?pressureProperty <https://w3id.org/seas/evaluation> ?pressure .

  ?feature <https://w3id.org/platoon/hasPressure> ?pressureProperty .
  ?feature a <https://w3id.org/seas/FeatureOfInterest> .
  ?generator <https://schema.org/location> ?feature .
  ?s3 <http://www.w3.org/ns/prov#wasGeneratedBy> ?generator .
}
```

Query 2

```
SELECT DISTINCT ?humidity ?humidityValue ?humidityProperty ?feature ?generator
WHERE {
  ?humidity a <https://w3id.org/platoon/RelativeHumidityEvaluation> .
  ?humidity <https://w3id.org/seas/evaluatedSimpleValue> ?humidityValue .
  ?humidity <https://w3id.org/seas/hasTemporalContext> ?tempContext .

  ?humidityProperty a <https://w3id.org/seas/HumidityProperty> .
  ?humidityProperty <https://w3id.org/seas/evaluation> ?humidity .


  ?feature <https://w3id.org/platoon/hasRelativeHumidity> ?humidityProperty .
  ?feature a <https://w3id.org/seas/FeatureOfInterest> .
  ?generator <https://schema.org/location> ?feature .
  ?s3 <http://www.w3.org/ns/prov#wasGeneratedBy> ?generator .
}
```

Query 3

```
SELECT DISTINCT ?powerEval ?powerEvalValue
WHERE {
  ?powerEval a <https://w3id.org/platoon/ActivePowerEvaluation> .
  ?powerEval <https://w3id.org/seas/evaluatedSimpleValue> ?powerEvalValue .
  ?powerEval <https://w3id.org/seas/hasTemporalContext> ?tempContext .

  ?powerEvalProp a <https://w3id.org/seas/ElectricPowerProperty> .
  ?powerEvalProp <https://w3id.org/seas/evaluation> ?powerEval .
}
```

Query 4

```
SELECT DISTINCT ?electrical ?electricalValue ?electricalProperty
WHERE {
  ?electrical a <https://w3id.org/seas/ElectricPowerEvaluation> .
  ?electrical <https://w3id.org/seas/evaluatedSimpleValue> ?electricalValue .
  ?electrical <https://w3id.org/seas/hasTemporalContext> ?tempContext .

  ?electricalProperty a <https://w3id.org/seas/ElectricPowerProperty> .
  ?electricalProperty <https://w3id.org/seas/evaluation>  ?electrical .
}
```

Query 5

```
SELECT DISTINCT ?humidity ?humidityValue ?humidityProperty
WHERE {
  ?humidity a <https://w3id.org/platoon/RelativeHumidityEvaluation> .
  ?humidity <https://w3id.org/seas/evaluatedSimpleValue> ?humidityValue .
  ?humidity <https://w3id.org/seas/hasTemporalContext> ?tempContext .

  ?humidityProperty a <https://w3id.org/seas/HumidityProperty> .
  ?humidityProperty <https://w3id.org/seas/evaluation> ?humidity .
}
```

References

[1] "PLATOON D2.1: PLATOON Reference Architecture," 2020.

[2] "PLATOON D2.3: PLATOON Common Data Models for Energy," 2020.

[3] "PLATOON D2.4: The PLATOON Unified Knowledge Base Creation," 2020.

[4] "PLATOON D5.3: Harmonization and Knowledge Extraction Services," 2022.

[5] M. Lefrançois, A. Zimmermann and N. Bakerally, "Flexible RDF generation from RDF and heterogeneous data sources with SPARQL-Generate," in *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16)*, 2016.

[6] M. Lefrançois, A. Zimmermann and N. Bakerally, "A SPARQL extension for generating RDF from heteogeneous formats," in *Proc. Extended Semantic Web Conference (ESWC '17)*, 2017.

[7] S. Jozashoori and M.-E. Vidal, "MapSDI: A Scaled-Up Semantic Data Integration Framework for Knowledge Graph Creation," CooPIS, 2019.

[8] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana and M.-E. Vidal, "SDM-RDFizer: An RML interpreter for the efficient creation to RDF knowledge graphs," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 3039--3046.

[9] S. Jozashoori, D. Chaves-Fraga, E. Iglesias, M.-E. Vidal and O. Corcho, "FunMap: Efficient Execution of Functional Mappings for Knowledge Graph Creation," ISWC, 2020.

[10] M.-E. Vidal, E. Ruckhaus, T. Lampo, A. Martínez, J. Sierra and A. Polleres, "Efficiently Joining Group Patterns in SPARQL Queries," ESWC, 2010.

[11] M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo and E. Ruckhaus, "ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints," IWSC, 2011.

[12] A. Schwarte, P. Haase, K. Hose, R. Schenkel and M. Schmidt, "FedX: Optimization Techniques for Federated Query Processing on Linked Data," ISWC, 2011.

[13] K. M. Endris, P. D. Rohde, M.-E. Vidal and S. Auer, "Ontario: Federated Query Processing against a Semantic Data Lake," DEXA, 2019.

[14] Y. Khan, A. Zimmermann, A. Jha, V. Gadepally, M. D'Aquin and R. Sahay, "One Size Does Not Fit All: Querying Web Polystores," *IEEE Access*, *Volume 7*, 2019.

[15] E. Iglesias, S. Jozashoori and M.-E. Vidal, "Scaling Up Knowledge Graph Creation to Large and Heterogeneous Data Sources," https://arxiv.org/abs/2201.09694, 2022.

[16] K. M. Endris, M. Galkin, I. Lytra, M. N. Mami, M.-E. Vidal and S. Auer, "Querying Interlinked Data by Bridging RDF Molecule Templates," *Transactions on Large-Scale Data and Knowledge-Centered Systems,* 2018.

[17] M. Acosta, M.-E. Vidal and Y. Sure-Vetter, "Diefficiency Metrics: Measuring the Continuous Efficiency of Query Processing Approaches," ISWC, 2017.