OntoKB: Translation Knowledge Base

Luca Foschini, UNIBO
Alessio Mora, UNIBO
Alessandro Calvio, UNIBO

OntoKB

What is OntoKB and why do we need it in OntoTrans?

OntoKB is the knowledge base for the OntoTrans system. It is an instance of a Triplestore containing the semantic knowledge necessary for the Open Translation Environment (OTE) system

But what is a Triplestore?

The term Triplestore typically refers to any system that has facilities for persistent storage of **Resource Description Framework (RDF) data triples** (i.e., subject-predicate-object)

- Triplestores are frameworks that are more than just RDF databases though
- Triplestores, in fact, typically provide an Application Programming Interface (API) that allows handling and manipulating RDF data – that goes beyond simple storage and retrieval
- **You can reason on triples to make new facts emerge**, something that has not been explicitly written down in the stored triples!
- **A Triplestore will provide built-in reasoner(s)**

Suitable Triplestore for OntoKB

OntoKB needs a Triplestore framework that provides, at least, the following features:

- Storing and handling RDF-like triples (ontologies, individuals)
- Supporting SPARQL 1.1 HTTP Protocol (i.e., SPARQL queries over HTTP)
- Enabling built-in reasoning (up to a certain level/profile of expressivity)

We considered different alternatives to be used as Triplestore

GraphDB

rdf4j

AllegroGraph

STARDOG

OPENLINK VIRTUOSO

MarkLogic

Suitable Triplestore for OntoKB

OntoKB needs a Triplestore framework that provides, at least, the following features:

- Storing and handling RDF-like triples (ontologies, individuals)
- Supporting SPARQL 1.1 HTTP Protocol (i.e., SPARQL queries over HTTP)
- Enabling built-in reasoning (up to a certain level/profile of expressivity)

We considered different alternatives to be used as Triplestore:

GraphDB

rdf4j

AllegroGraph

STARDOG

OPENLINK VIRTUOSO

MarkLogic

OntoKB (currently) is using Stardog as Triplestore

- Stardog is a commercial solution (License required, but constantly maintained and up-to-date)

Stardog as OntoKB Triplestore

STARDOG TRIPLESTORE

Features:

- HTTP SPARQL Endpoint
- Different built-in reasoning levels (up to OWL2 DL)
- Convenient Administration Panel (Stardog Studio)
- Comes in dockerized flavor
- Provides APIs to connect and query w/ or w/o reasoning (e.g., *Phyton rdflib*)
- Developer-friendly documentation

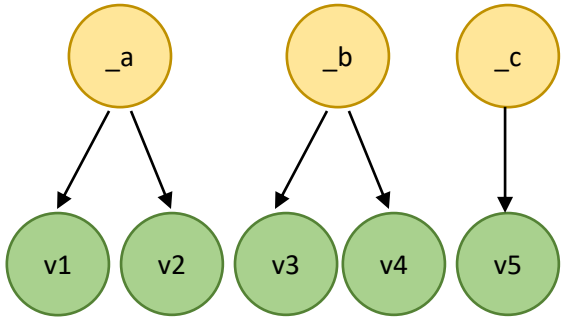
https://www.stardog.com/

How does SPARQL queries look like?

SPARQL: **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage

SPARQL query contains a set of triple patterns called a ***basic graph pattern***. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable
A basic graph pattern ***matches*** **a subgraph of the RDF data** when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Johnny Lee Outlaw" .    (v1)
_:a foaf:mbox <mailto:jlow@example.com> .    (v2)
_:b foaf:name "Peter Goodguy" .    (v3)
_:b foaf:mbox <mailto:peter@example.org> .    (v4)
_:c foaf:mbox <mailto:carol@example.org> .    (v5)
```

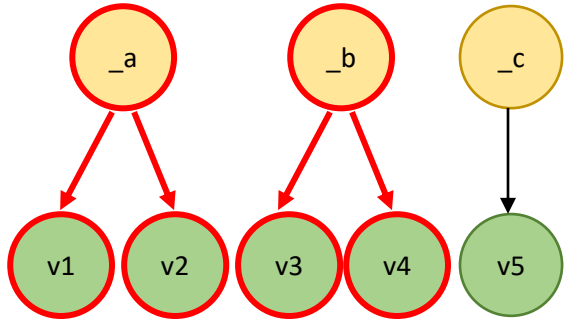


SPARQL Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?mbox
WHERE {
   ?x foaf:name ?name .
   ?x foaf:mbox ?mbox
}
```

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

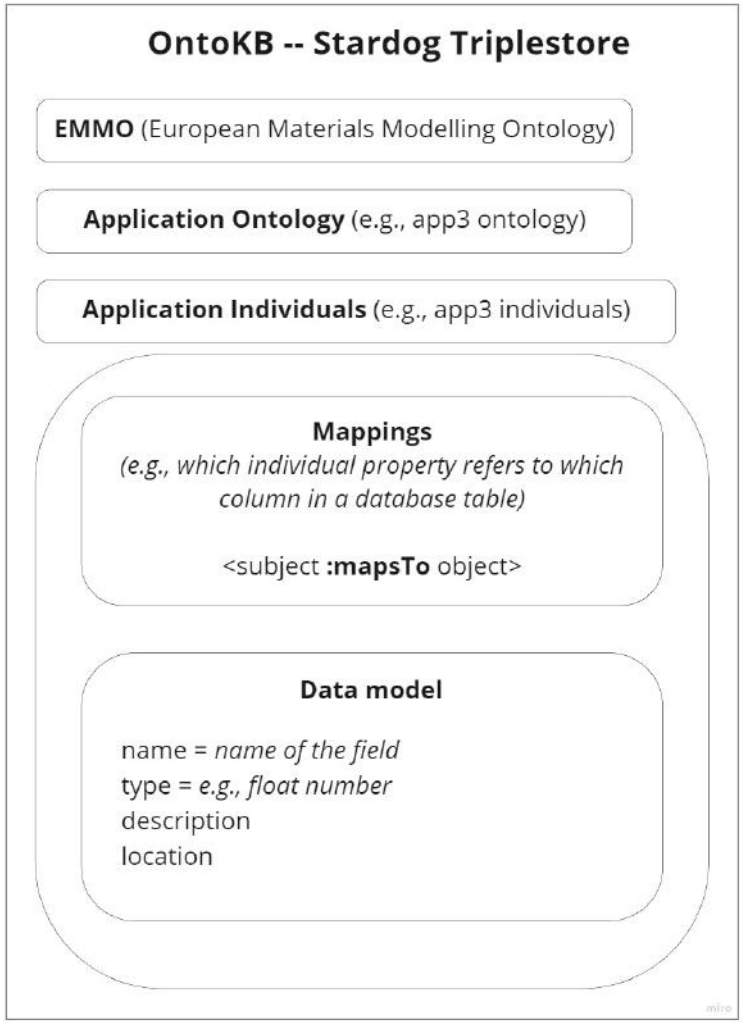


Back to OTE: What is inside OntoKB?

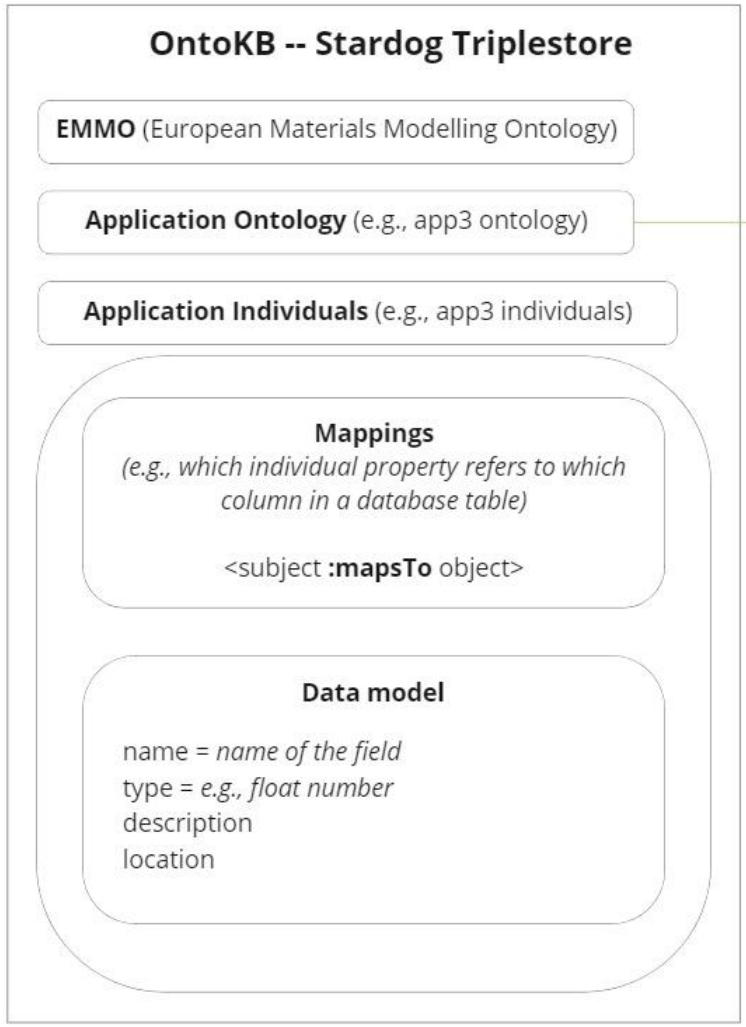
- **Raw data are not stored in OntoKB**
  - We do not want to use OntoKB as a relational database for raw data
  - We do not want OntoKB to explode with tremendous amount of raw data
  - We do not want to have proprietary raw data in OntoKB
- **OntoKB stores ontological schemas and individuals**
- Specifically, OntoKB stores
    - The mapping between ontological concepts and data models
    - Data models represent the structure of raw data (e.g., which ontological property refers to which column in a table)
    - The location of raw data (e.g., address of a specific DB, local path of an Excel file)

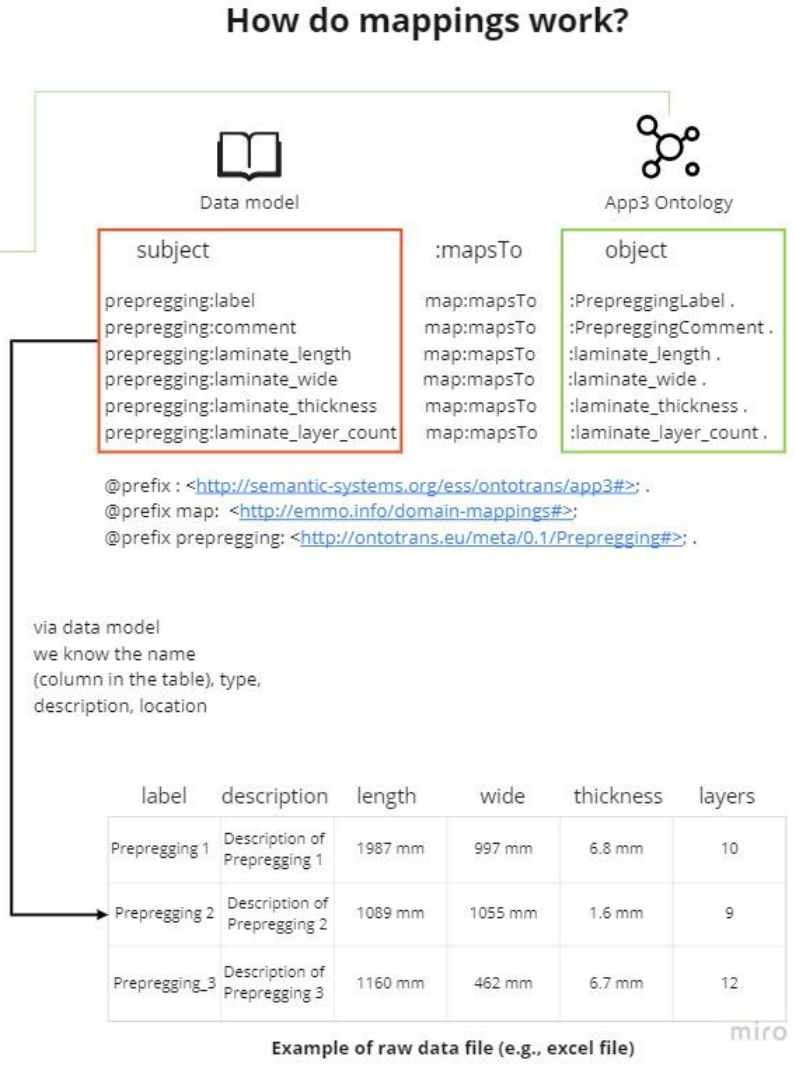
    → **OTE system can retrieve and semantically link to raw data**

What is inside OntoKB?

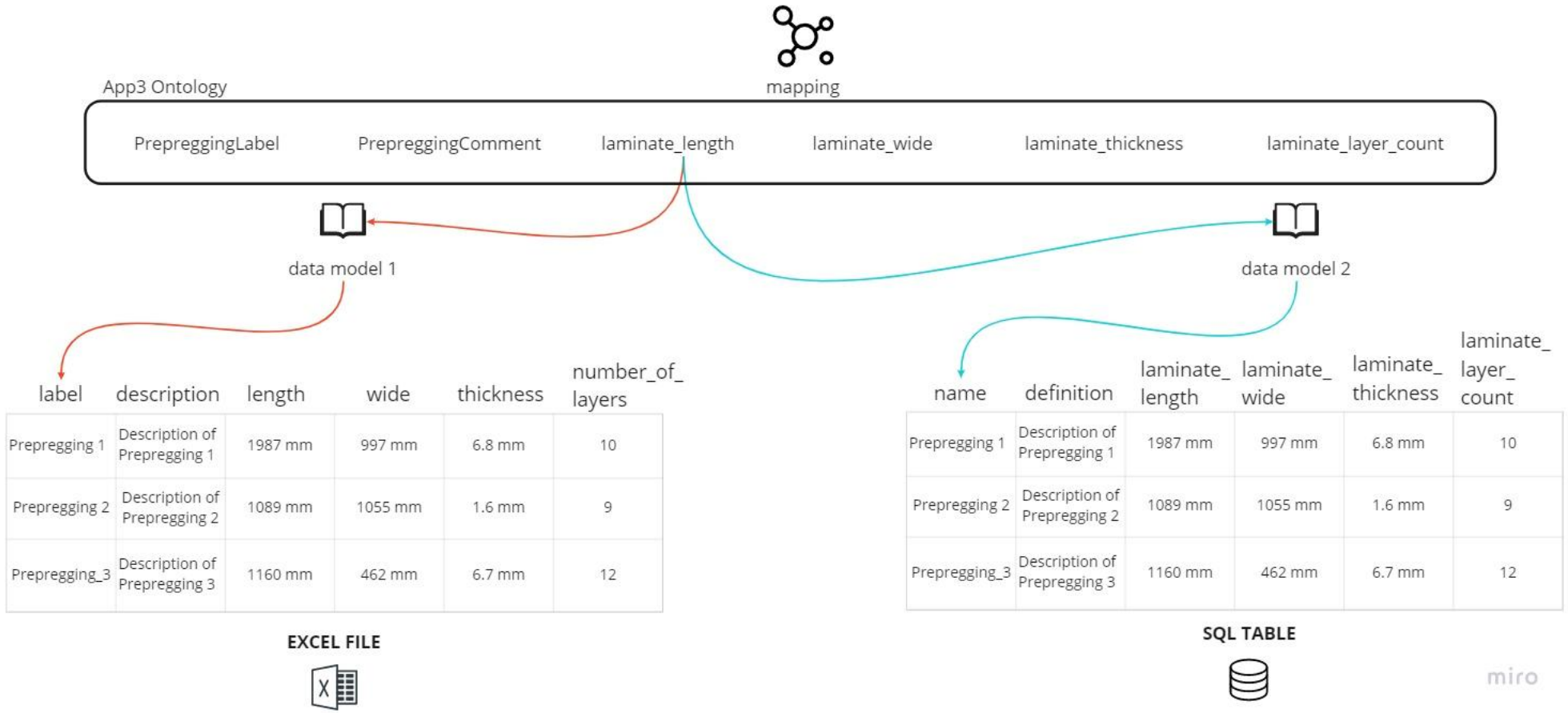


What is inside OntoKB?

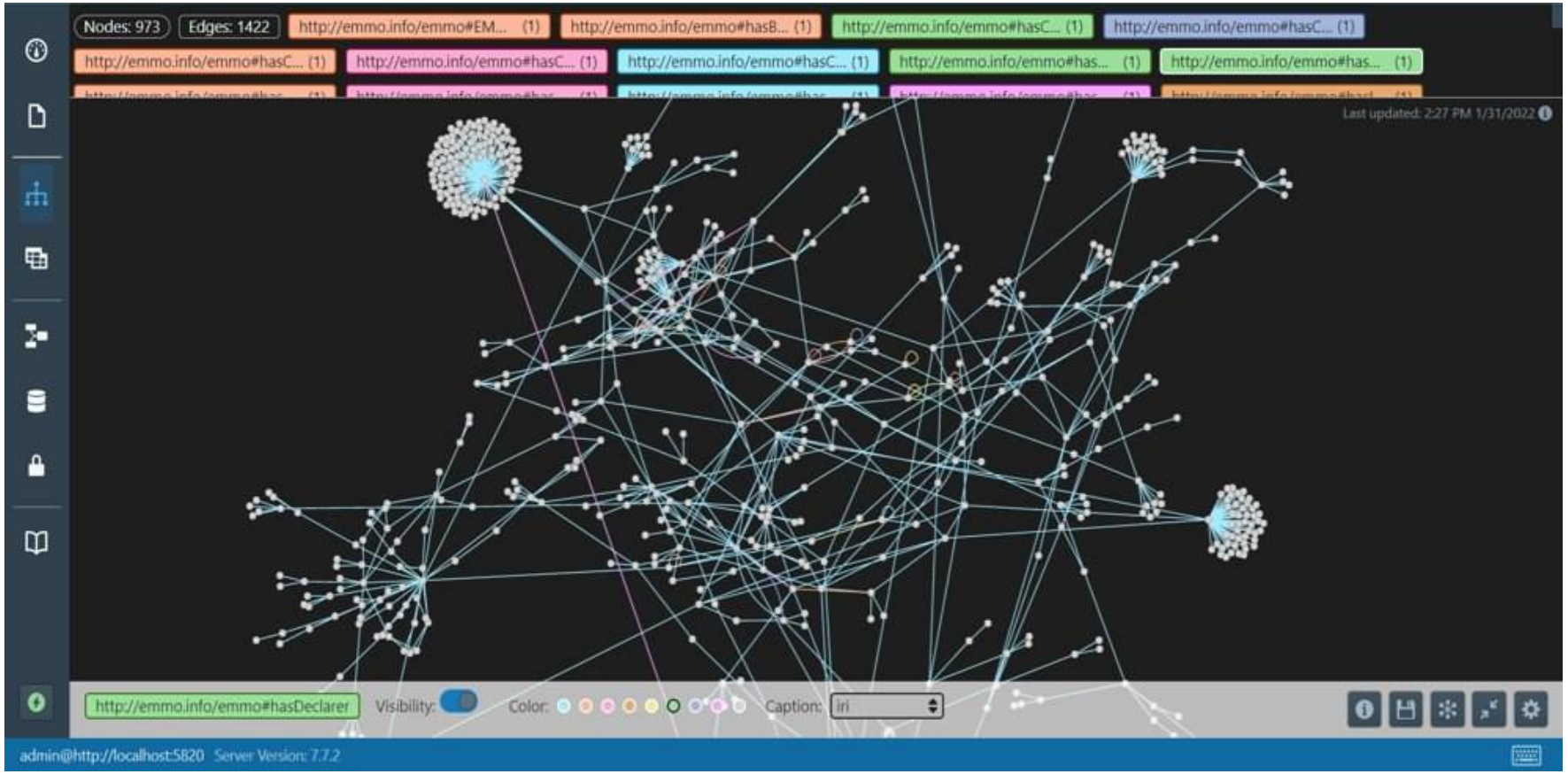




Why are mappings and data models needed?

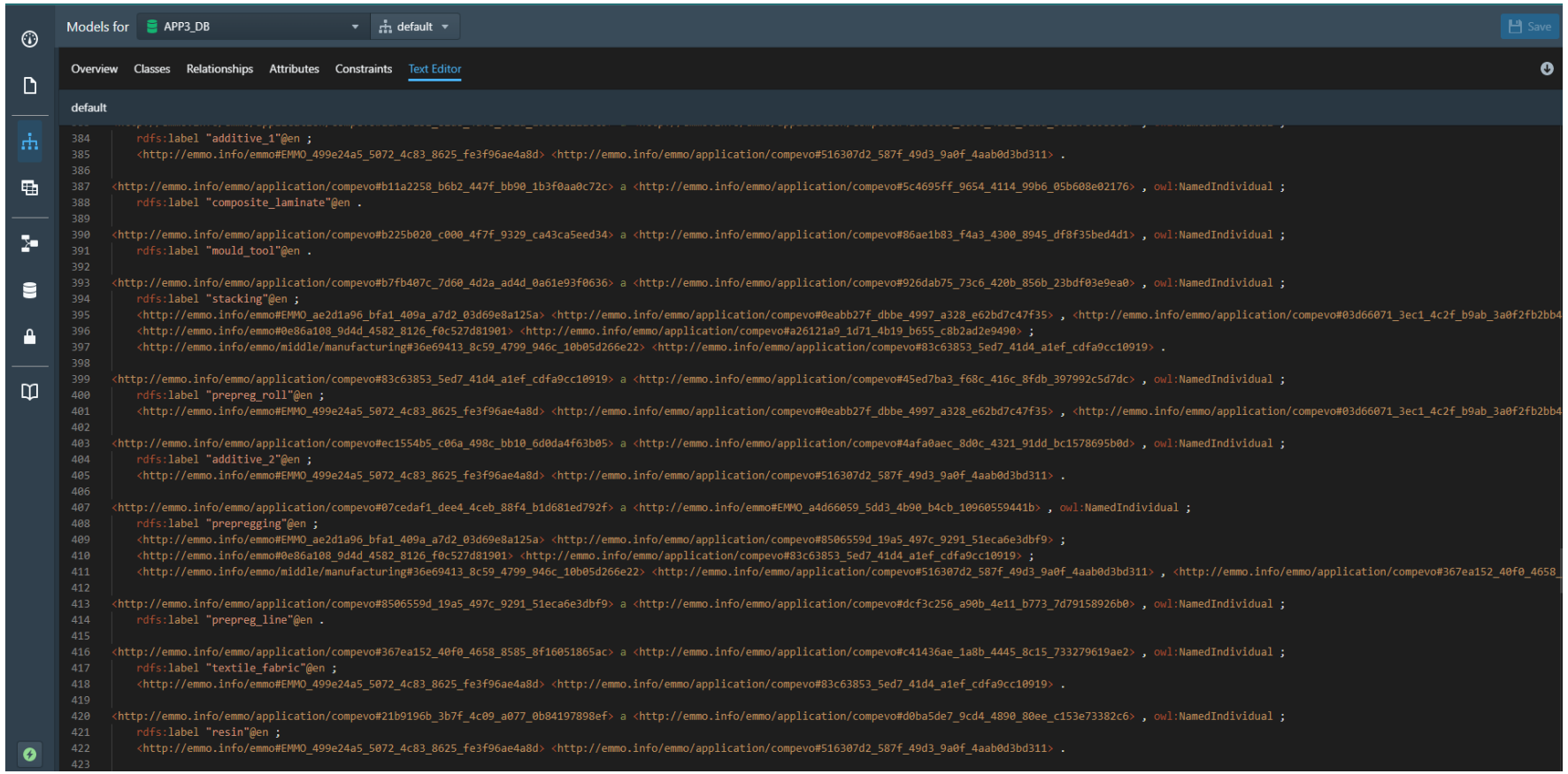


How does the contents inside OntoKB look like?



Example of graph visualization of an ontology – Stardog Studio

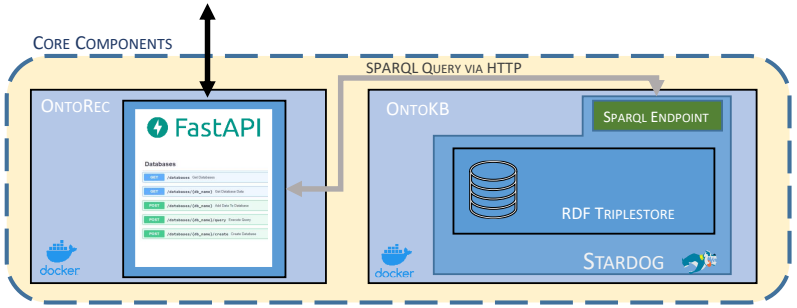
How does the contents inside OntoKB look like?




```
384     rdfs:label "additive_1"@en ;
385     <http://emmo.info/emmo#EMMO_499e24a5_5072_4c83_8625_fe3f96ae4a8d> <http://emmo.info/emmo/application/compevo#516307d2_587f_49d3_9a0f_4aab0d3bd311> .
386
387 <http://emmo.info/emmo/application/compevo#b11a2258_b6b2_447f_bb90_1b3f0aa0c72c> a <http://emmo.info/emmo/application/compevo#5c4695ff_9654_4114_99b6_05b608e02176> , owl:NamedIndividual ;
388     rdfs:label "composite_laminate"@en .
389
390 <http://emmo.info/emmo/application/compevo#b225b020_c000_4f7f_9329_ca43ca5eed34> a <http://emmo.info/emmo/application/compevo#86ae1b83_f4a3_4300_8945_df8f35bed4d1> , owl:NamedIndividual ;
391     rdfs:label "mould_tool"@en .
392
393 <http://emmo.info/emmo/application/compevo#b7fb407c_7d60_4d2a_ad4d_0a61e93f0636> a <http://emmo.info/emmo/application/compevo#926dab75_73c6_420b_856b_23bdf03e9ea0> , owl:NamedIndividual ;
394     rdfs:label "stacking"@en ;
395     <http://emmo.info/emmo#EMMO_ae2d1a96_bfa1_409a_a7d2_03d69e8a125a> <http://emmo.info/emmo/application/compevo#0eabb27f_dbbe_4997_a328_e62bd7c47f35> , <http://emmo.info/emmo/application/compevo#03d66071_3ec1_4c2f_b9ab_3a0f2fb2bb4
396     <http://emmo.info/emmo#0e86a108_9d4d_4582_8126_f0c527d81901> <http://emmo.info/emmo/application/compevo#a26121a9_1d71_4b19_b655_c8b2ad2e9490> ;
397     <http://emmo.info/emmo/middle/manufacturing#36e69413_8c59_4799_946c_10b05d266e22> <http://emmo.info/emmo/application/compevo#83c63853_5ed7_41d4_a1ef_cdfa9cc10919> .
398
399 <http://emmo.info/emmo/application/compevo#83c63853_5ed7_41d4_a1ef_cdfa9cc10919> a <http://emmo.info/emmo/application/compevo#45ed7ba3_f68c_416c_8fdb_397992c5d7dc> , owl:NamedIndividual ;
400     rdfs:label "prepreg_roll"@en ;
401     <http://emmo.info/emmo#EMMO_499e24a5_5072_4c83_8625_fe3f96ae4a8d> <http://emmo.info/emmo/application/compevo#0eabb27f_dbbe_4997_a328_e62bd7c47f35> , <http://emmo.info/emmo/application/compevo#03d66071_3ec1_4c2f_b9ab_3a0f2fb2bb4
402
403 <http://emmo.info/emmo/application/compevo#ec1554b5_c06a_498c_bb10_6d0da4f63b05> a <http://emmo.info/emmo/application/compevo#4afa0aec_8d0c_4321_91dd_bc1578695b0d> , owl:NamedIndividual ;
404     rdfs:label "additive_2"@en ;
405     <http://emmo.info/emmo#EMMO_499e24a5_5072_4c83_8625_fe3f96ae4a8d> <http://emmo.info/emmo/application/compevo#516307d2_587f_49d3_9a0f_4aab0d3bd311> .
406
407 <http://emmo.info/emmo/application/compevo#07cedaf1_dee4_4ceb_88f4_b1d681ed792f> a <http://emmo.info/emmo#EMMO_a4d66059_5dd3_4b90_b4cb_10960559441b> , owl:NamedIndividual ;
408     rdfs:label "prepregging"@en ;
409     <http://emmo.info/emmo#EMMO_ae2d1a96_bfa1_409a_a7d2_03d69e8a125a> <http://emmo.info/emmo/application/compevo#8506559d_19a5_497c_9291_51eca6e3dbf9> ;
410     <http://emmo.info/emmo#0e86a108_9d4d_4582_8126_f0c527d81901> <http://emmo.info/emmo/application/compevo#83c63853_5ed7_41d4_a1ef_cdfa9cc10919> ;
411     <http://emmo.info/emmo/middle/manufacturing#36e69413_8c59_4799_946c_10b05d266e22> <http://emmo.info/emmo/application/compevo#516307d2_587f_49d3_9a0f_4aab0d3bd311> , <http://emmo.info/emmo/application/compevo#367ea152_40f0_4658_
412
413 <http://emmo.info/emmo/application/compevo#8506559d_19a5_497c_9291_51eca6e3dbf9> a <http://emmo.info/emmo/application/compevo#dcf3c256_a90b_4e11_b773_7d79158926b0> , owl:NamedIndividual ;
414     rdfs:label "prepreg_line"@en .
415
416 <http://emmo.info/emmo/application/compevo#367ea152_40f0_4658_8585_8f16051865ac> a <http://emmo.info/emmo/application/compevo#c41436ae_1a8b_4445_8c15_733279619ae2> , owl:NamedIndividual ;
417     rdfs:label "textile_fabric"@en ;
418     <http://emmo.info/emmo#EMMO_499e24a5_5072_4c83_8625_fe3f96ae4a8d> <http://emmo.info/emmo/application/compevo#83c63853_5ed7_41d4_a1ef_cdfa9cc10919> .
419
420 <http://emmo.info/emmo/application/compevo#21b9196b_3b7f_4c09_a077_0b84197898ef> a <http://emmo.info/emmo/application/compevo#d0ba5de7_9cd4_4890_80ee_c153e73382c6> , owl:NamedIndividual ;
421     rdfs:label "resin"@en ;
422     <http://emmo.info/emmo#EMMO_499e24a5_5072_4c83_8625_fe3f96ae4a8d> <http://emmo.info/emmo/application/compevo#516307d2_587f_49d3_9a0f_4aab0d3bd311> .
423
```

Example of text visualization of an ontology – Stardog Studio

OntoREC



OntoREC offers a controlled access point to OntoKB functionalities by means of a high-level REST API

- Separated component with respect to OntoKB
- Built on:
  - FastAPI web framework for API
  - Pydantic data models to offer updated documentation and validation features
- Exploits RDF standard APIs (RDFlib) to connect to and use the native SPARQL endpoint of Stardog through HTTP protocol

Since it has been designed to be put in front of OntoKB triplestore it could allow several further optimizations

OntoREC: API

We currently have 5 different APIs regarding the main operation that can be executed on a Stardog database. All responses are in JSON format.
The path structure tries to reflect the semantic meaning of the specific endpoint:

/entity/name/operation

GET	/databases	Return the list of available databases in the triplestore
GET	/databases/{db_name}	Return all the triples contained into the specified DB
POST	/databases/{db_name}/query	Execute a specific query (with reasoning capabilities) passed as a string
POST	/databases/{db_name}/create	Create a database with the specified name
POST	/databases/{db_name}	Add an RDF or TTL ontology to the specified database.

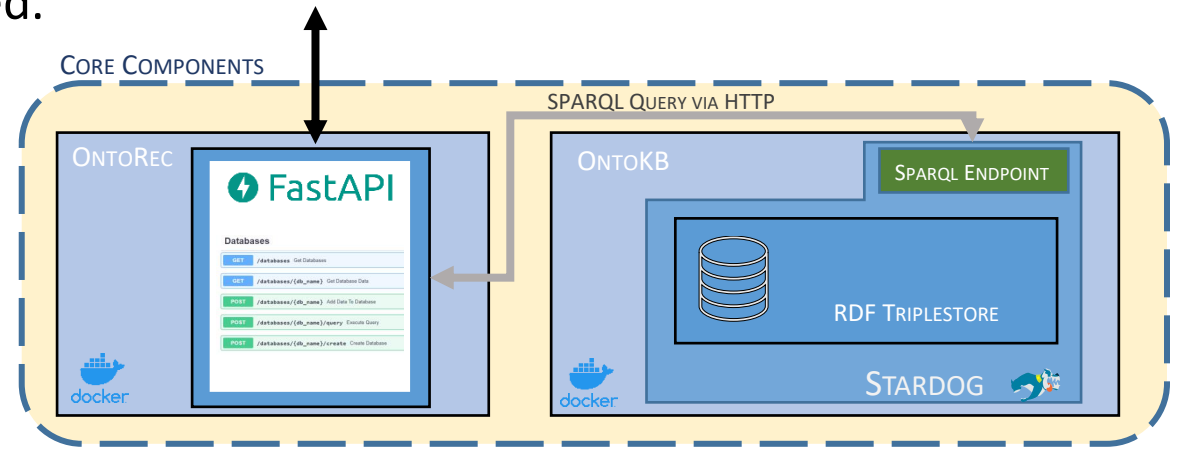
OntoKB and OntoRec as microservices

OntoKB and OntoRec, are thought to be deployed together as two separated and containerized services. In this way you can deploy and configure them according to your needs.

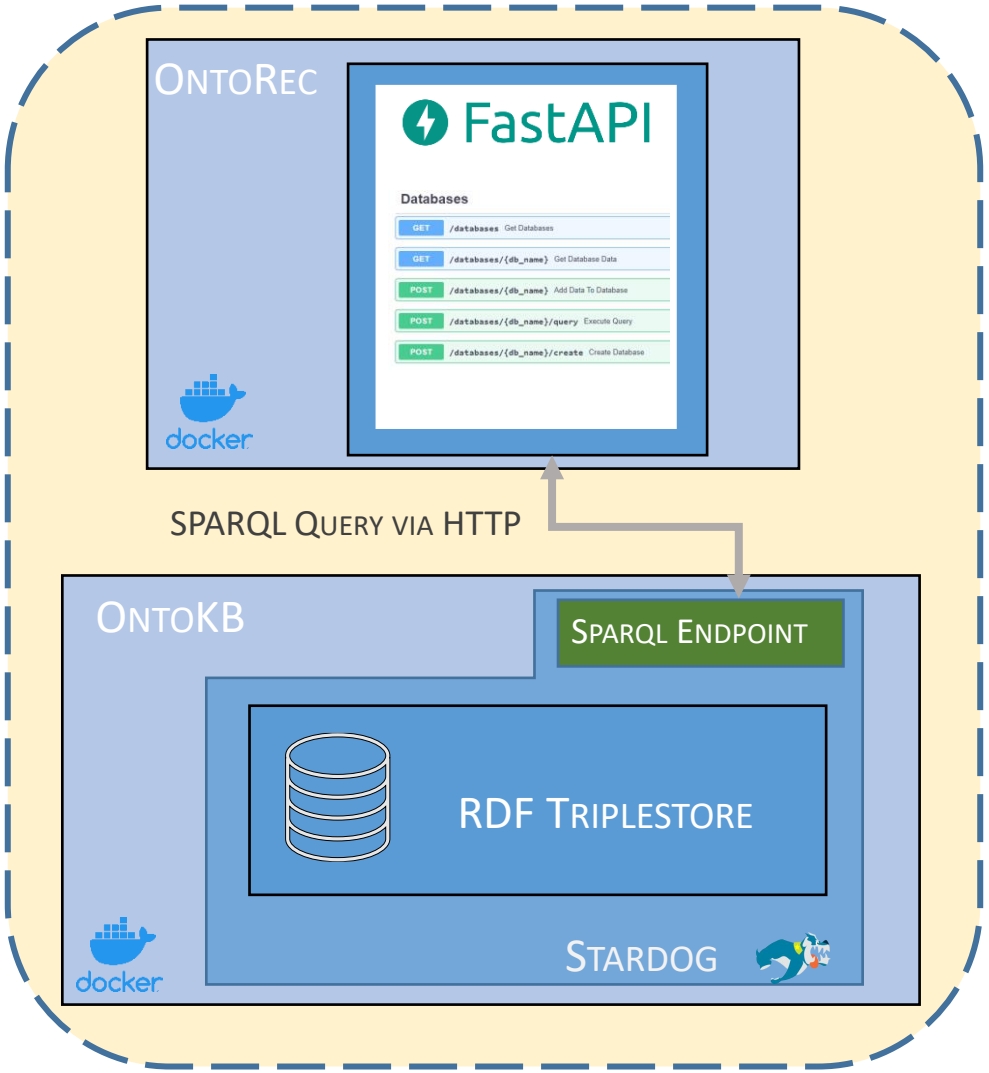
Each container has its own role, so:

- OntoKB stores the knowledge base (**application ontologies** and **individuals**, **meta-data**, i.e., mappings for the actual raw data and location of raw data).
- OntoRec provides tailored REST APIs to access OntoKB (e.g., uploading and fetching ontologies) encapsulating the actual SPARQL queries.

OntoKB can still be directly accessed through its HTTP endpoint bypassing OntoRec when low-level access is needed.



OntoKB and OntoRec deployment



Dockerfile

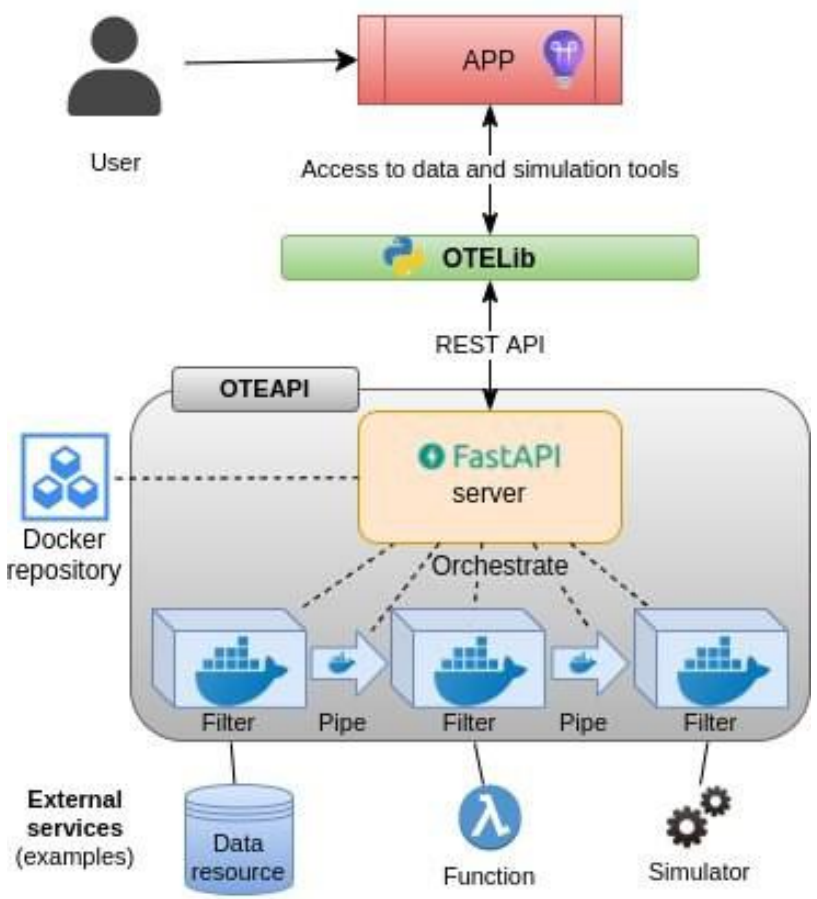
```
ontotranscore-api:
    image: <GITLAB REGISTRY IMAGE>
    depends_on:
        - stardog
    ports:
        - "80:80"
```

The two containers are joined to the same bridge network so that one container can address another by using its name (as the hostname)

Dockerfile

```
stardog:
    image: "stardog/stardog"
    ports:
        - "5820:5820"
    volumes:
        - <PATH TO LOCAL STARDOG HOME>:/var/opt/stardog
```

OTEAPI framework



The OTEAPI provides a **REST API for connecting a sequence of microservices** into a **pipeline** for connecting simulation tools to data sources in a very flexible and reusable way

The purpose of the framework is **to separate the process of connecting to external data resources, mapping it to a common ontology** (for interoperability), data processing, and running simulations into a set of reusable and independent components

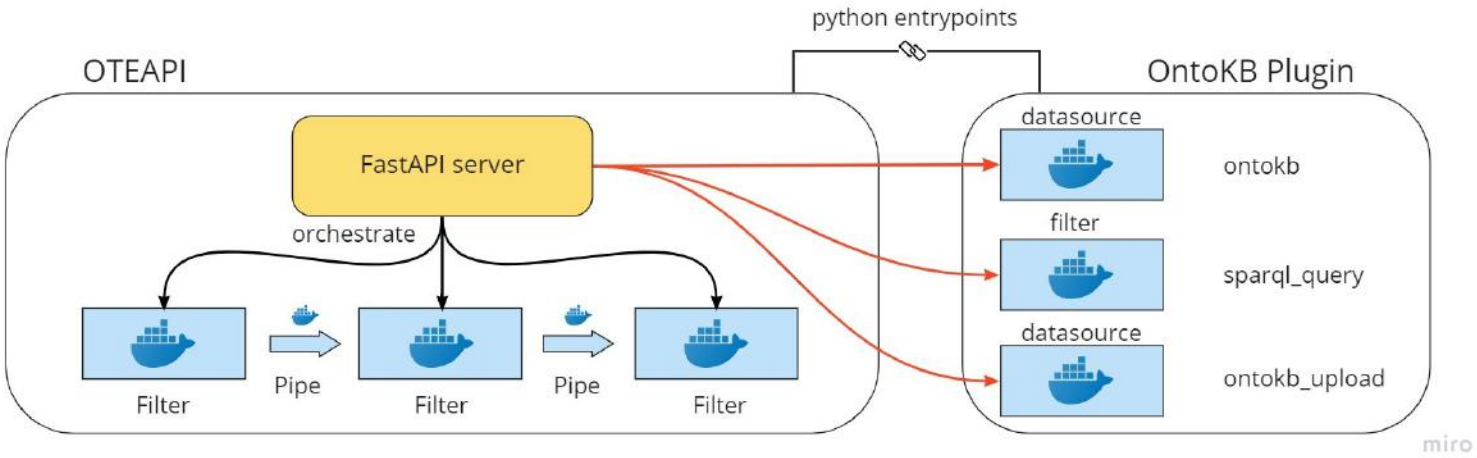
OTElib is a **Python library that wants to help developers by facilitating the creation of a pipeline** and abstracting from too technical details

OTEAPI Integration - plugins

The OTEAPI framework takes advantage of the pipe and filter design pattern to create standalone modules (**strategy**) that can be combined and developed in a loosely coupled way

This means that even if the framework already provides **some strategy the core functionalities** can be expanded by developing **third-party's plugins** that can provide new implementations of a strategy or features

The expansion is possible by exploiting the python entrypoint system

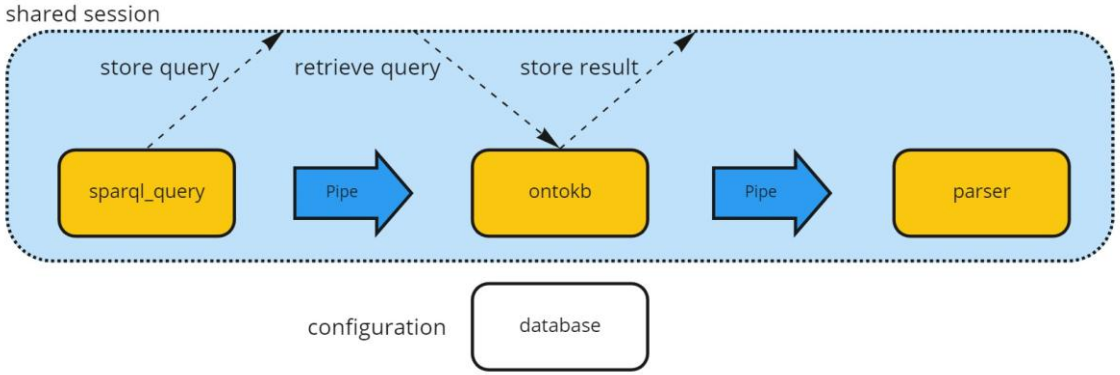


OntoKB Plugin

The OntoKB plugin implements three new strategies to allow OTEAPI users to perform some main functionality on existing databases on the triplestore

- dataresource strategy **ontokb** to read data on a specific database
- filter strategy **sparql_query** to specify a query to execute on a database
- dataresource strategy **ontokb_upload** to upload data on existing databases

The plugin uses the session to share information between strategies and make them interact properly



ontokb executes the actual query if exists in session otherwise returns all the data in the specified database

What we are going to show during the demo

OntoKB and OntoREC

- Stardog Studio
  - Instance connection
  - Graph visualization
  - Text visualization and editing
- Some OntoREC example with Postman
  - Creation of a (*Stardog*) database
  - Add data to a (*Stardog*) database
  - Query of a (*Stardog*) database

OTEAPI Interaction

- Build a pipeline with our strategies
- Show a plugin example w and w/o SPARQL query

Conclusion

- OntoKB is an instance of a Triplestore (Stardog) and **stores the knowledge base necessary for the OTE system**
- Triplestores, such as Stardog, **provide reasoning capabilities** besides the regular SPARQL queries
- OntoKB is implemented as a Dockerized Stardog (with commercial license)
- OntoRec is a component that **facilitates the interaction with Stardog** and provides more abstract (REST-based) APIs

- OntoKB does not contain any raw data. It provides all the information (*application ontologies, mappings, datamodels*) to retrieve them and to link to semantic application ontologies (e.g., needed for components like Exploratory Search Systems)
- OntoKB and OntoREC are integrated within the OTEAPI framework and can be used in *pipelines*

Thank you!

The OntoTrans project has received funding from the European Union‘s Horizon 2020 research and innovation programme under Grant Agreement No 862136.