OntoTrans

Knowledge Base and Recommender System: OntoKB and OntoRec

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

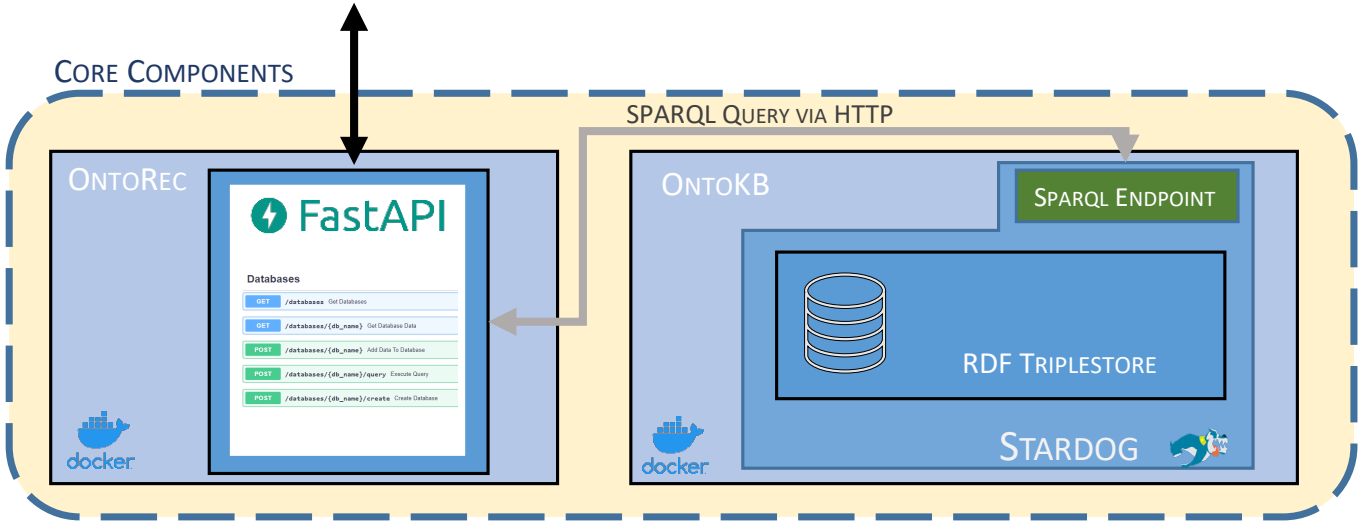
Outline

- OntoKB and OntoRec
- What is OntoKB and why do we need it in OntoTrans?
- What is inside OntoKB?
- OntoKB with Stardog as backend
- How does the contents inside OntoKB look like?
- OntoREC & its APIs
- Triplestore interface for OntoREC/OntoKB: Tripper
- How to access OntoKB

OntoKB and OntoRec

OntoKB and OntoREC are the core components in the OntoTrans EU Project, in particular

- **OntoKB** is the knowledge base for the OntoTrans system.
- **OntoREC** exposes a set of REST APIs related to the main operations that are possible to execute on a triplestore.



OntoKB and OntoRec interaction

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)

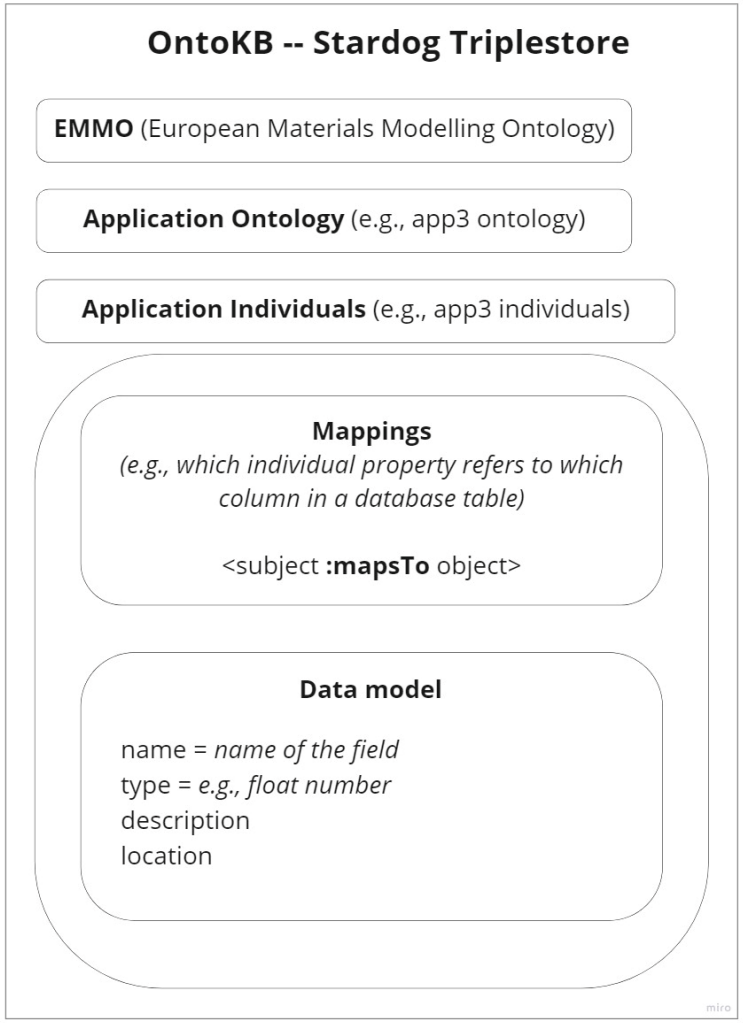
- Triplestores are frameworks that are more than just RDF databases though
- Triplestores, in fact, typically provide an Application Programming Interface (API) that allows handling and manipulating RDF data – that goes beyond simple storage and retrieval
- **You can reason on triples to make new facts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

What is inside OntoKB?

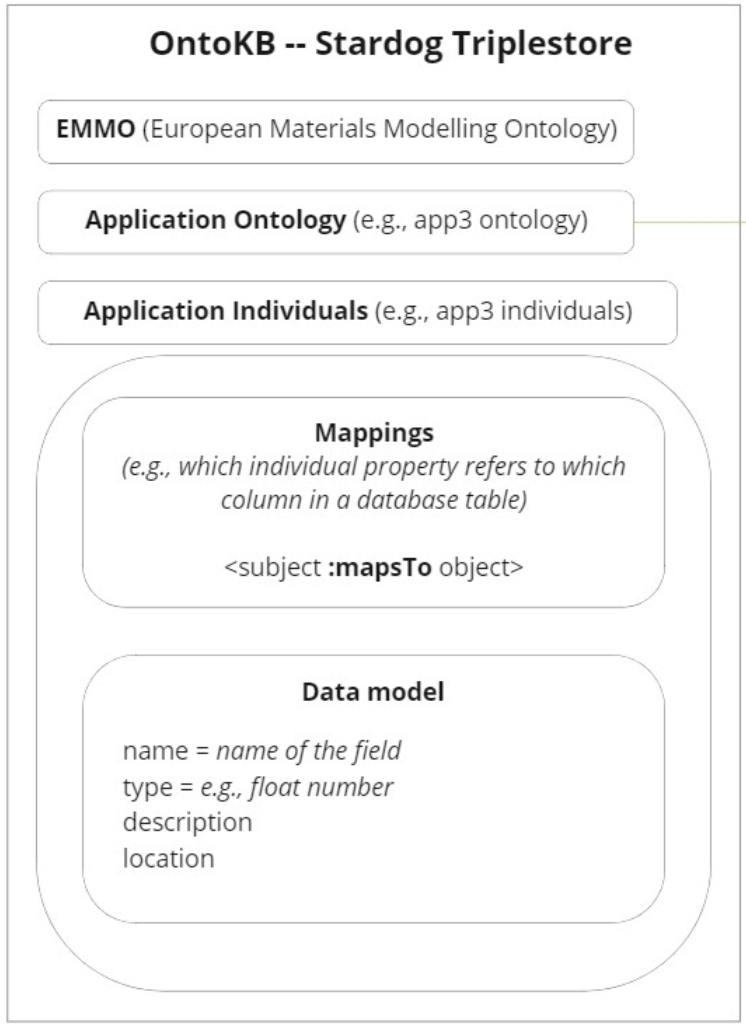
- **Raw data are not stored in OntoKB**
  - We do not want to use OntoKB as a relational database for raw data
  - We do not want OntoKB to explode with tremendous amount of raw data
  - We do not want to have proprietary raw data in OntoKB
- **OntoKB stores ontological schemas and individuals**
- Specifically, 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)

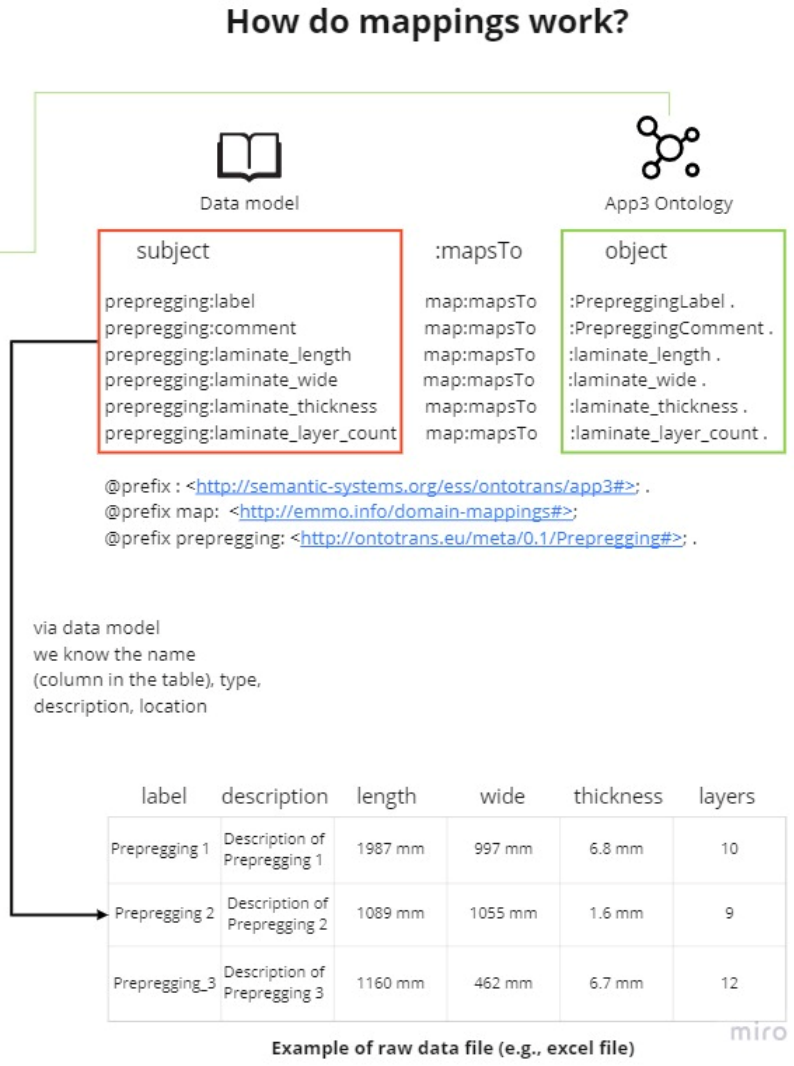
    → **OTE system can retrieve and semantically link to raw data**

What is inside OntoKB? (2/3)

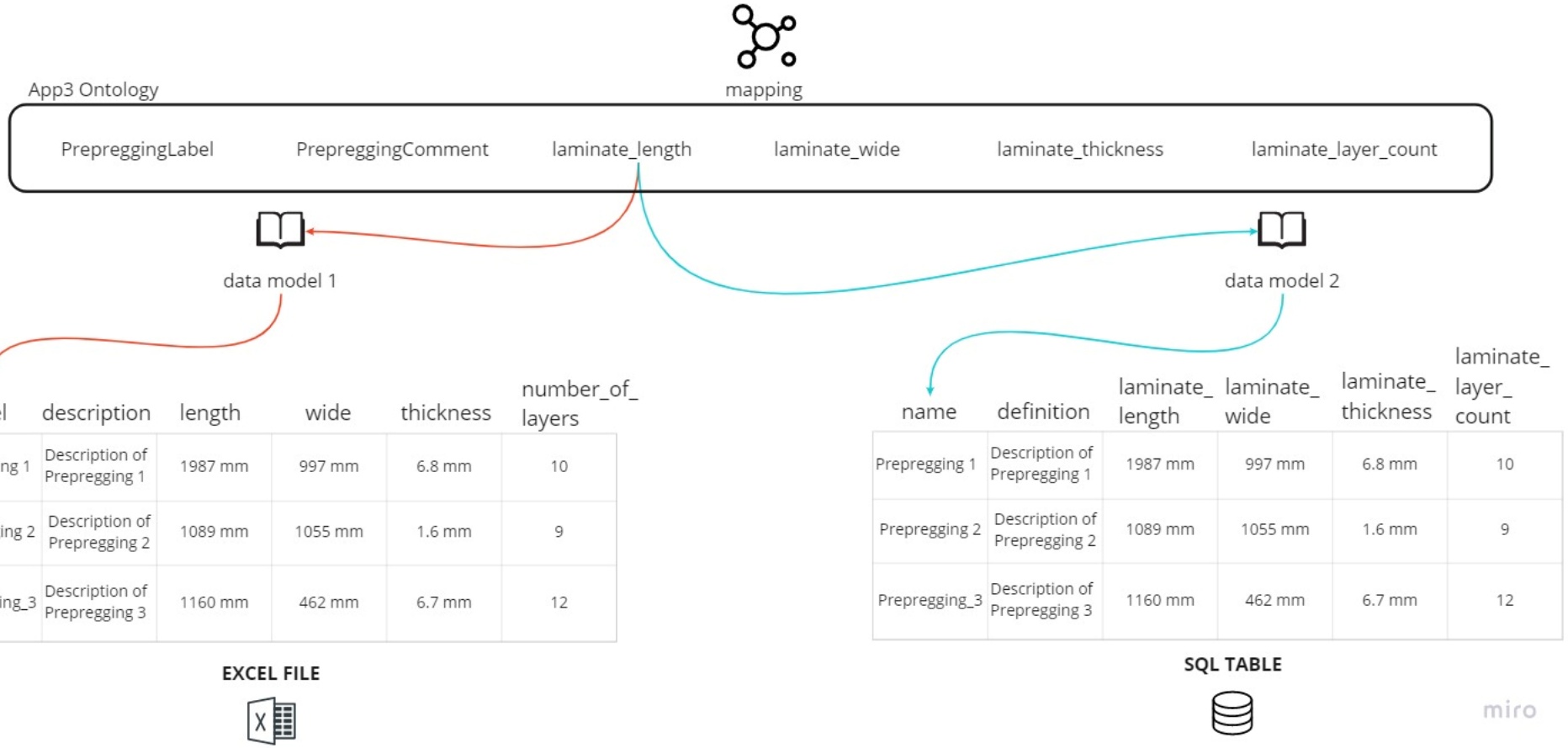


What is inside OntoKB? (3/3)





Why are mappings and data models needed?

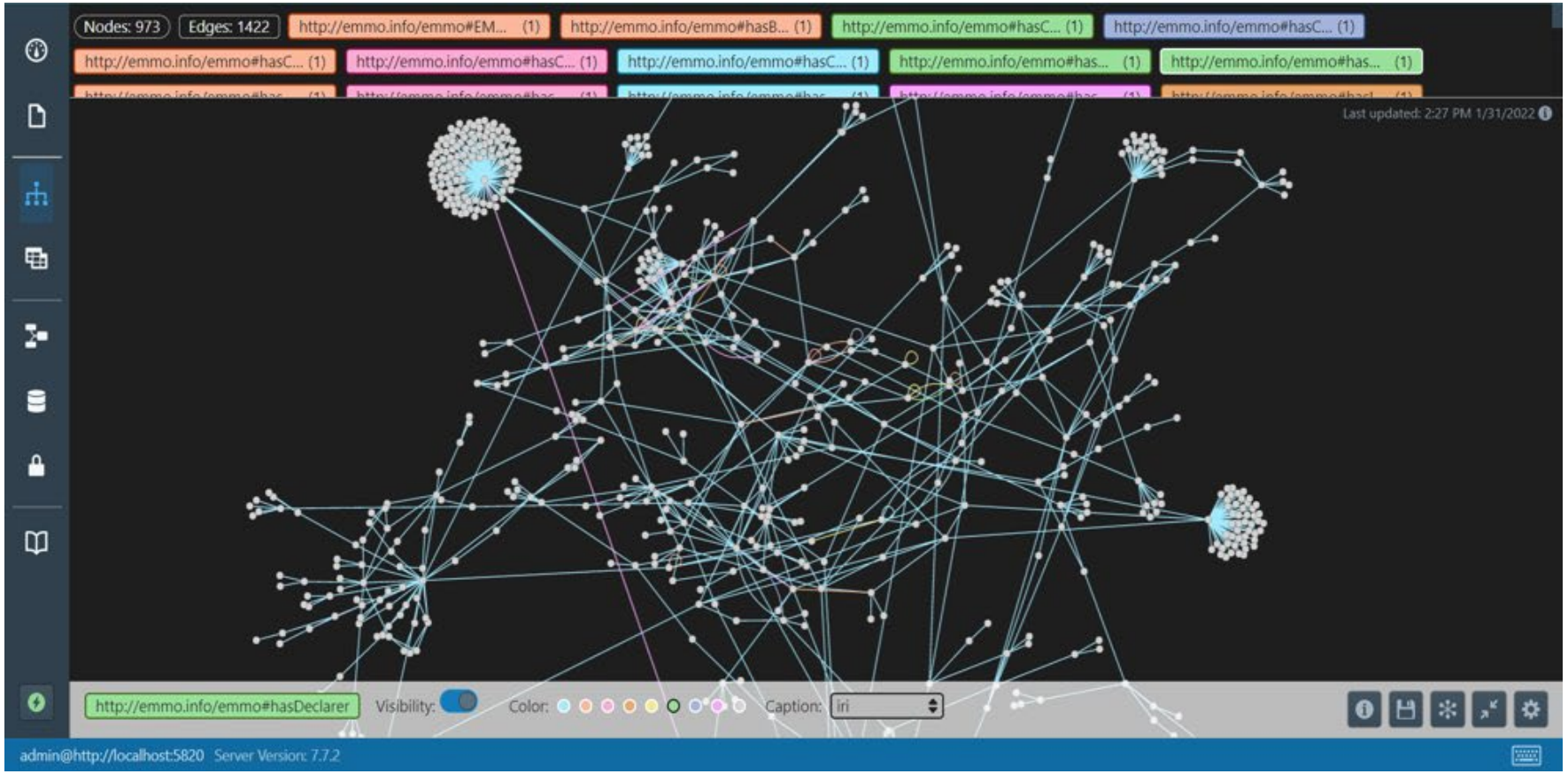


OntoKB with Stardog as backend

Current implementation of OntoKB relies on **Stardog**, a **full-fledged Triplestore** with capabilities to store ontological data, perform reasoning on it and provide a SPARQL endpoint and SPARQL querying functionalities.

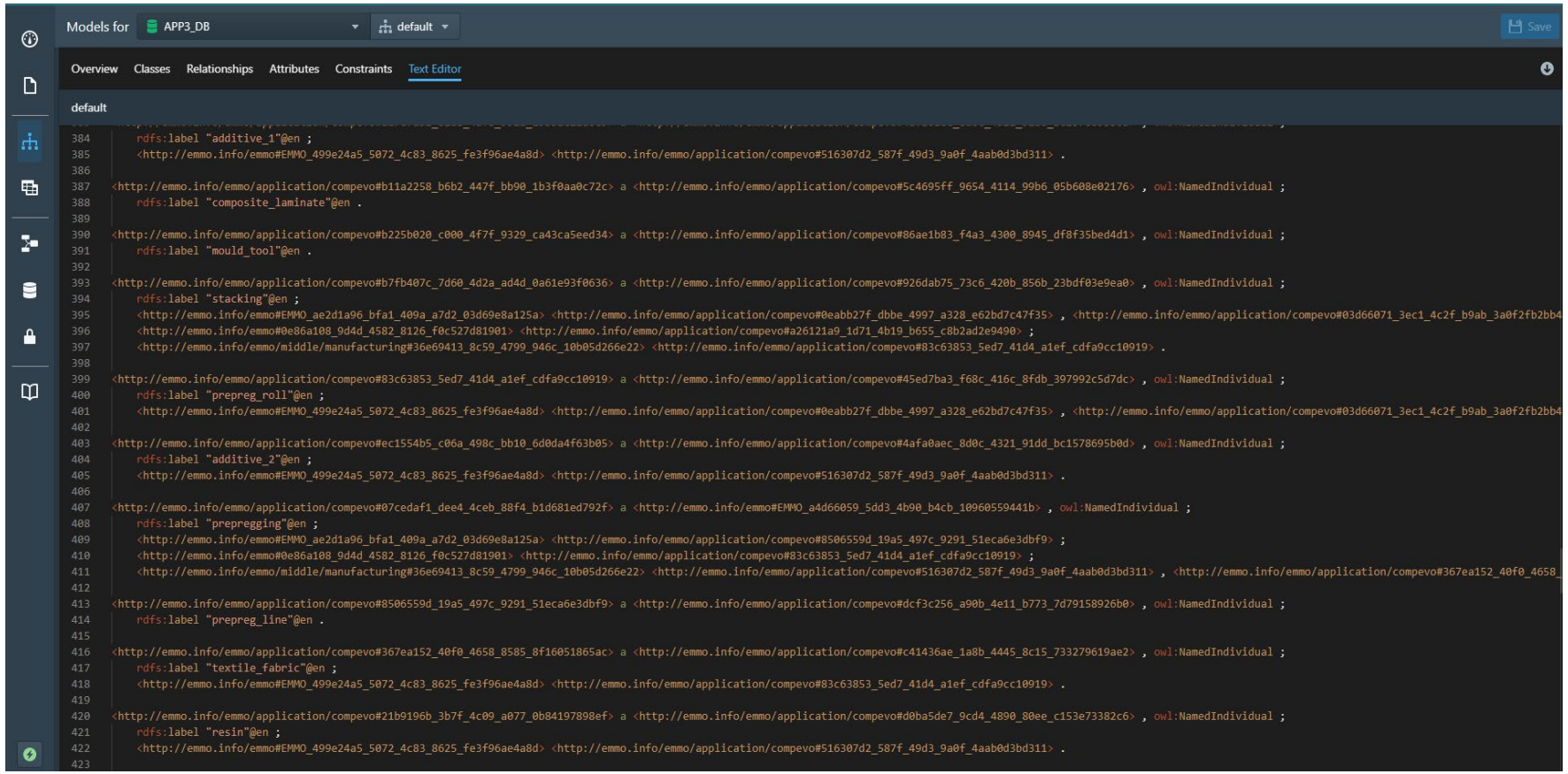
- It natively handles several ontologies serialization format (**RDF**, **Turtle**, etc…)
- It supports all the **OWL profiles** (including OWL DL)
- It provides **convenience features** for developers and general users (user-friendly web interfaces)
- It provides support for **SWRL** rules
- It comes with commercial licensing (among other trial versions) and it is currently maintained
- It comes with a free web-based interface to manually handle and manage ontologies (**Stardog-studio**)

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?



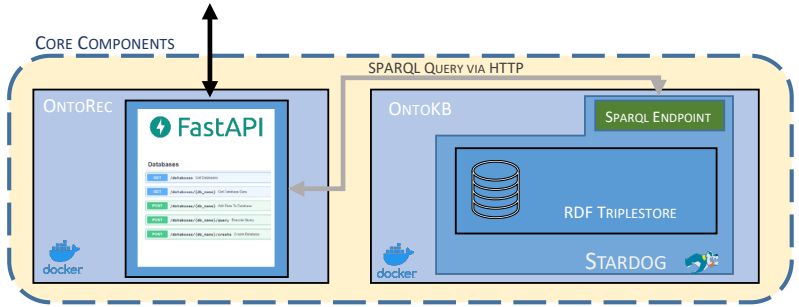
Example of text visualization of an ontology – Stardog Studio

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
- Enable triplestore independency by means of Tripper tool (see next slides)

FastAPI



OntoREC: APIs

OntoREC currently expose a set of REST APIs related to the main operation that is possible to execute on a database with respect to

- **Query submission**
  submission of (inferred) SPARQL queries
- **Database Management** as independent workspaces
- **Namespace Management**

METHOD	ENDPOINT	DESCRIPTION
GET	/databases	Get the list of all available databases
GET	/databases/{db_name}	Get all the triples contained into the database
GET	/databases/{db_name}/serialization	Get the serialization of the current database in a specified format
POST	/databases/{db_name}/query	Submit a query as a string to a specific database
POST	/databases/{db_name}/create	Create the database
POST	/databases/{db_name}	Add ontology file to existing database
POST	/databases/{db_name}/single	Add list of triples to existing database
DELETE	/databases/{db_name}	Delete an existing database
DELETE	/databases/{db_name}/single	Delete a list of triples from existing database
GET	/databases/{db_name}/namespaces	Get all the namespaces present in a database
GET	/databases/{db_name}/namespaces/base	Get the information related to the base namespace
GET	/databases/{db_name}/namespaces/{namespace_name}	Get the information related to a specific namespace
POST	/databases/{db_name}/namespaces	Add a new namespace
DELETE	/databases/{db_name}/namespaces/base	Delete the base namespace
DELETE	/databases/{db_name}/namespaces/{namespace_name}	Delete an existing namespace

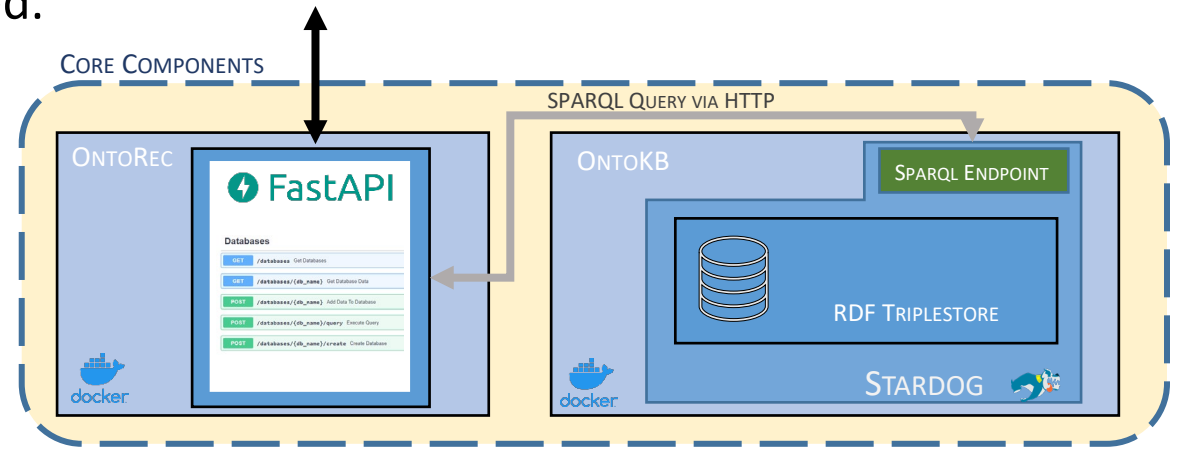
OntoKB and OntoREC as microservices

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.

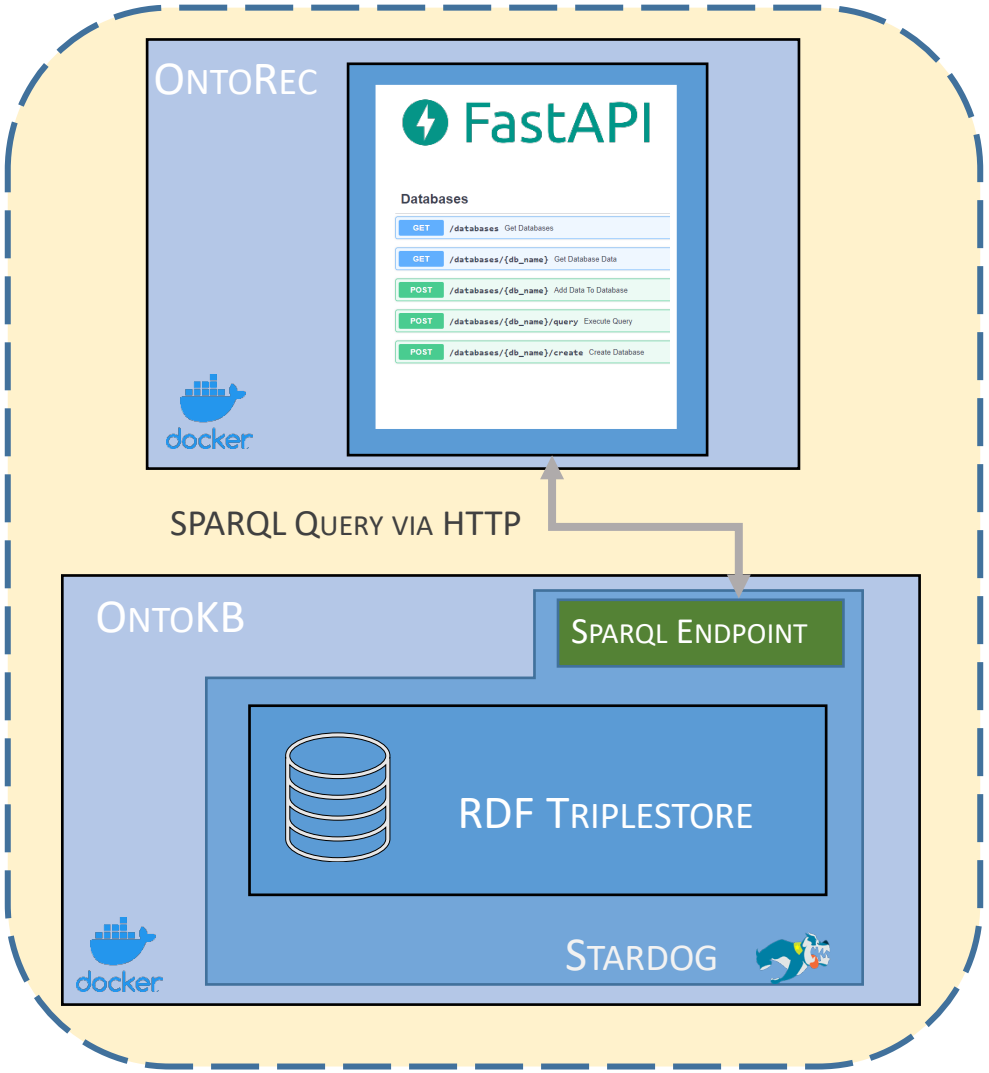
Each container has its own role, so:

- 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"
```

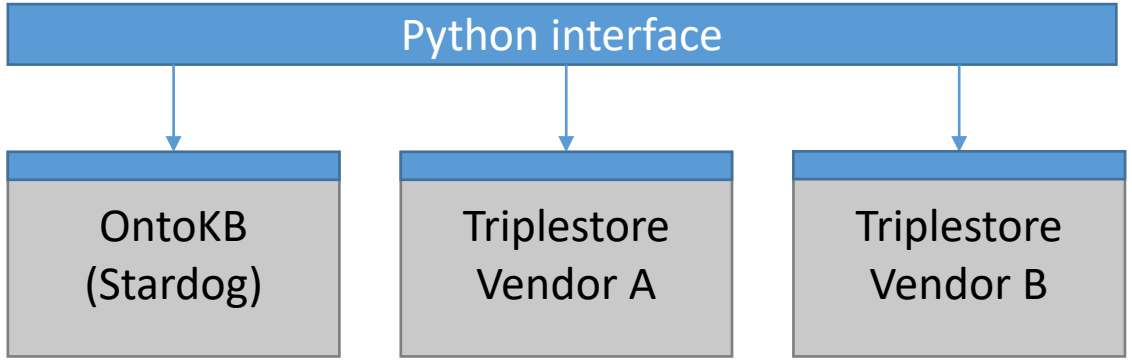
The two containers are joined to the same bridge network so that one container can address another by using its name (as the hostname)

Dockerfile

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

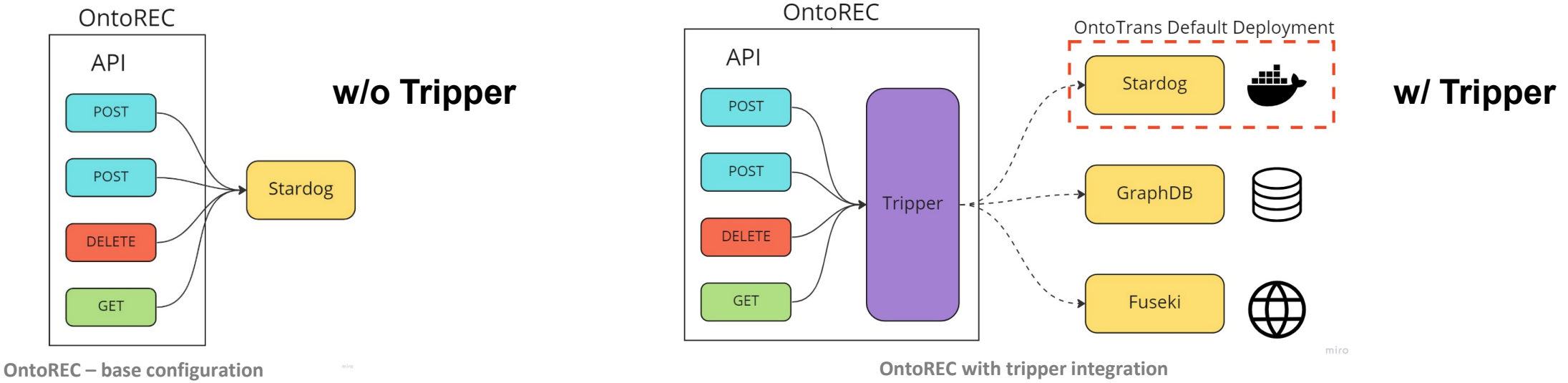
Triplestore interface for OntoREC/OntoKB: Tripper

- Triplestore-independent package for Python developed by SINTEF
- Enhancement of the tool to add OntoKB/OntoREC among the triplestores supported.
- Efforts to include new triplestores are minimized



Tripper and OntoREC integration

- Interface developed by SINTEF to support several triplestore technologies.
- It is used as an abstraction layer for the OntoREC component
  - It makes OntoREC independent from the current technology (e.g., Stardog, GraphDB, etc.)
  - Minimal effort is required to switch from one triplestore to another



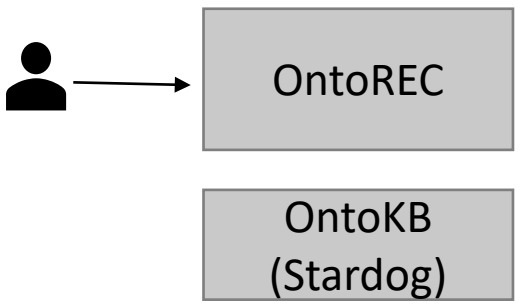
OntoREC – base configuration

OntoREC with tripper integration

How to access OntoKB

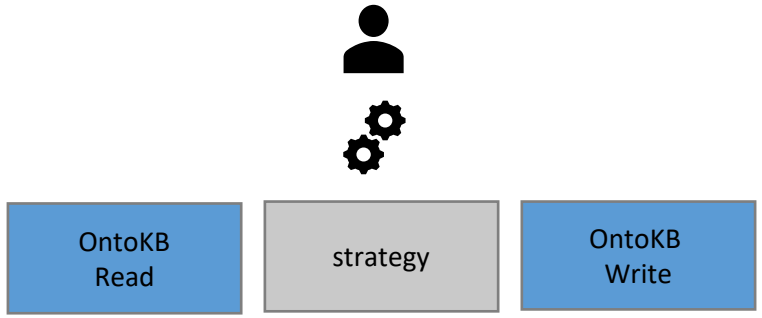
1

Direct access with OntoREC (REST APIs)



2

Access by means of the OTEAPI Framework.



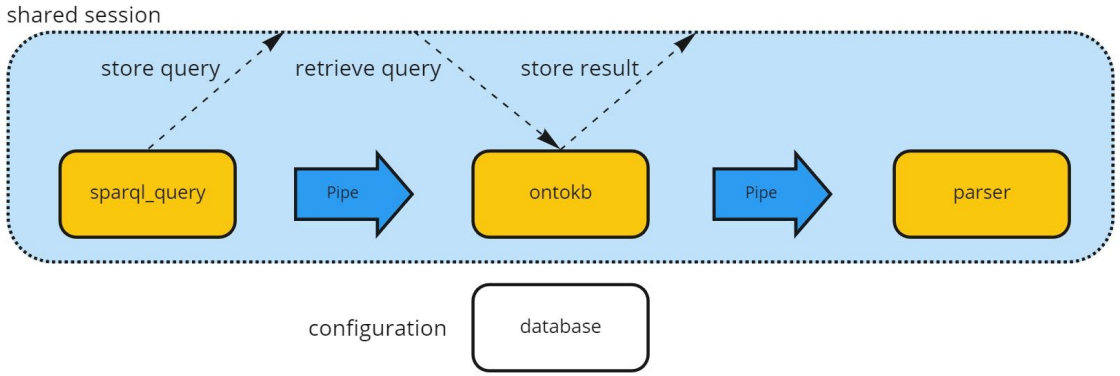
Users can choose their favourite way to access OntoKB

OntoKB Plugin

We extendend OTEAPI Framework with a set of custom plugin to enable interaction with the core components. 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

Conclusions

- 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.
- Both OntoKB and OntoREC come with dockerized deployment.
- OntoKB can be accessed via OntoREC's REST APIs or via OTEAPI plugins.
- OntoKB and OntoREC will be available as public services soon to the Consortium partners.

Thank you!

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