GraphDB's MongoDB connector unifies the Ontotext Platform's knowledge graph and annotation RDF stores. This blog post describes how JSON aggregate expressions combined with expressive SPARQL can support a global view across billions of knowledge statements and billions of annotation documents.

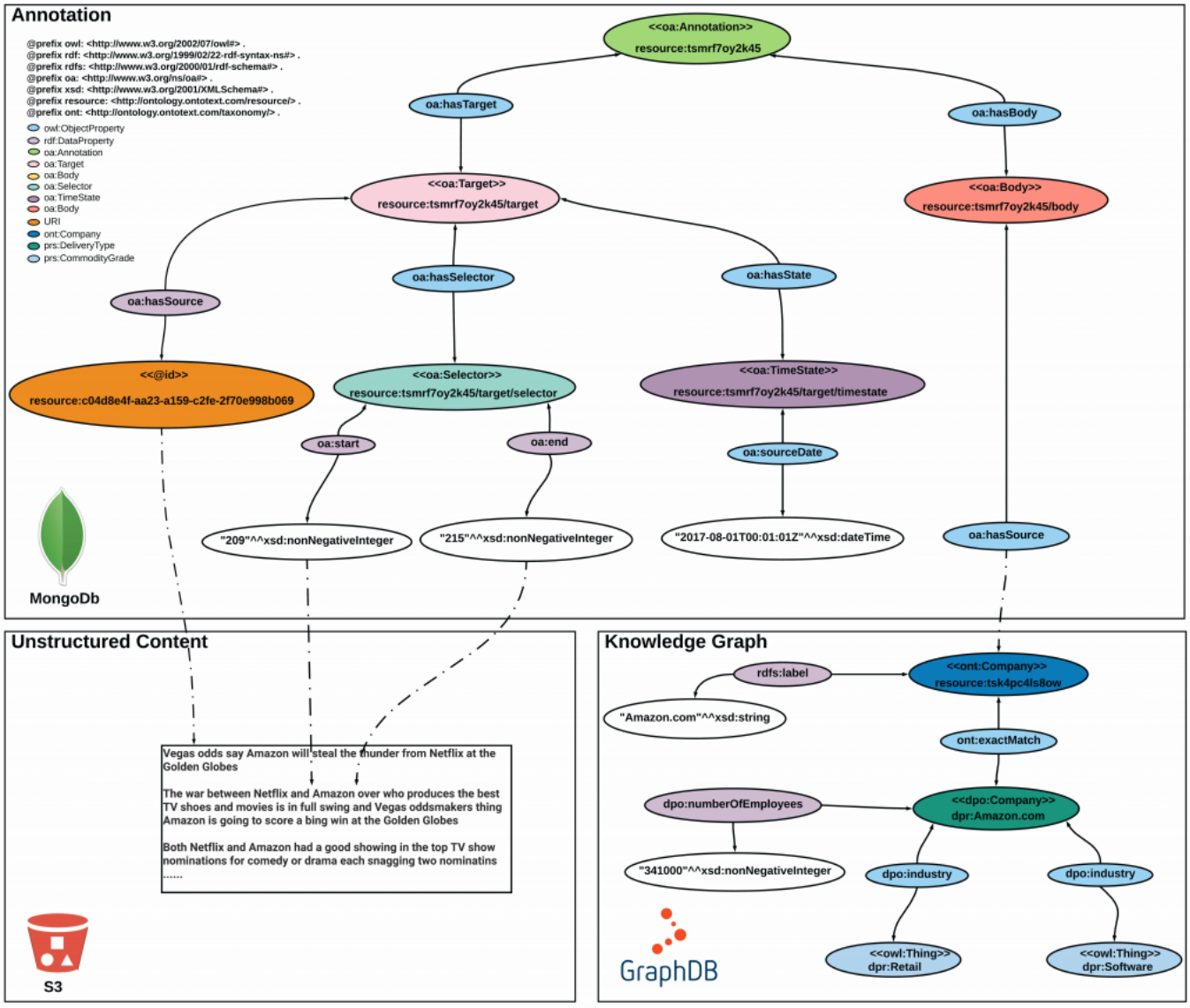
As discussed in my previous post, the Ontotext Platform is often required to process and reprocess millions of unstructured content items using the platform's text analytics components.

An unstructured content archive may need to be processed or re-processed to discover and add additional knowledge or train a machine learning model. Ontotext's text analytics components in these scenarios may well create 10's of billions of annotations that need to be processed, re-processed and stored quickly with little or indeed no impact to a live running knowledge graph.

The platform annotates unstructured content using JSON-LD conforming to the W3C Web Annotation Model [WA]. The JSON-LD documents convey information about target content items by using URIs that reference domain entities within a GraphDB knowledge graph.

The following diagram describes how the data points are interlinked and indeed where they are stored and managed within the platform. The selection of text "Amazon" contained within a plain text document is annotated by the Amazon.com entity.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com



- **Knowledge graph**: GraphDB is utilized to manage the FactForge knowledge graph.
- **Annotation**: MongoDB manages the Web Annotations.
- **Unstructured content**: The example depicts AWS S3 managing the unstructured content. However, unstructured content management is dependent on the content type.

The FactForge knowledge graph contains billions of entities and the diagram (above) only

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

includes a small selective set of instances and properties to indicate how the annotation makes reference to the knowledge contained within the graph. If you want to traverse and query the FactForge knowledge graph, you can follow this entry point: DBpedia Amazon.com entity.

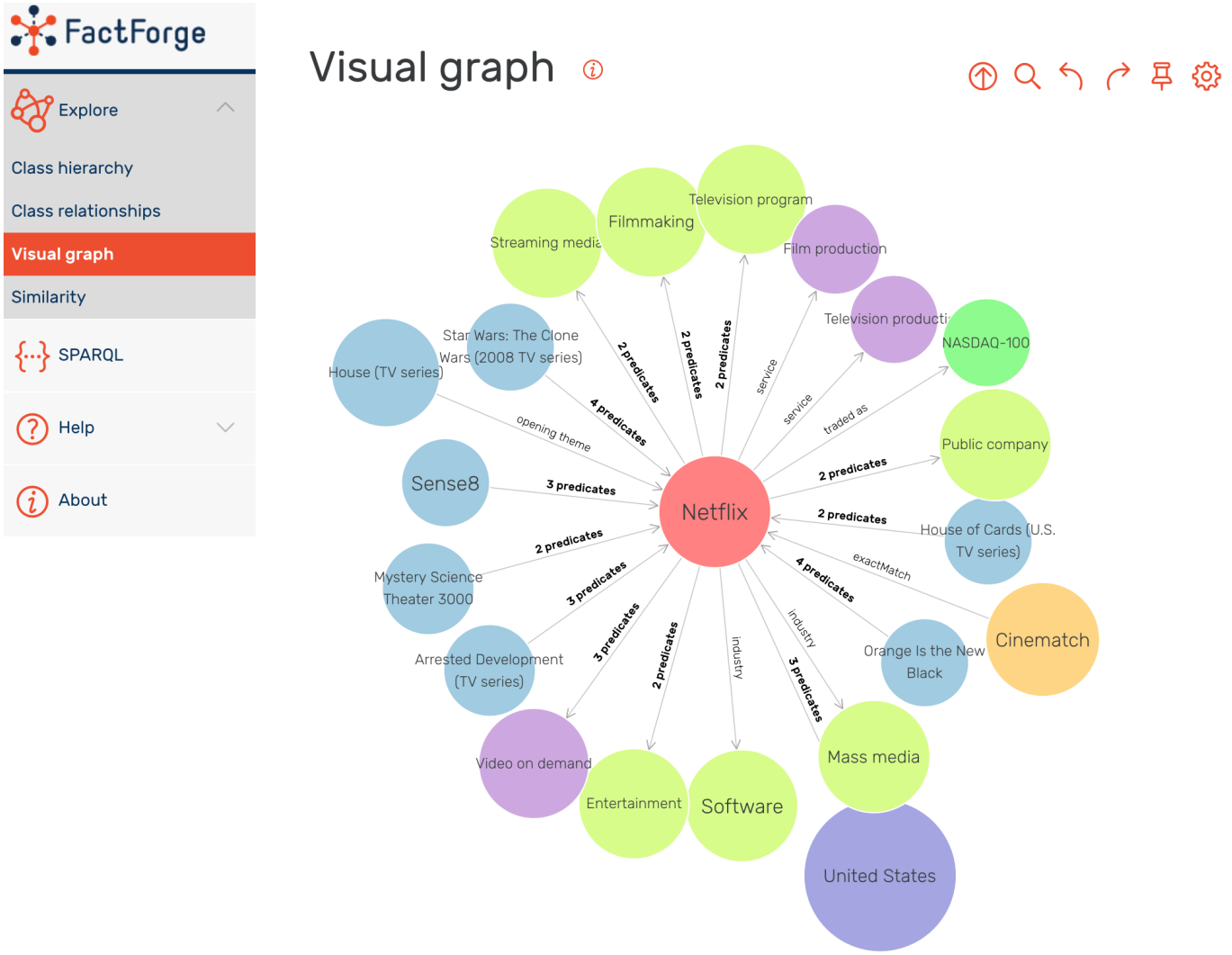
The following JSON-LD playground links are included to provide examples of the Annotation JSON-LD / RDF, which is stored in MongoDB.

- Amazon Annotation
- Netflix Annotation

The knowledge graph RDF can be examined by using Ontotext's Fact Forge GraphDB instance.

- Amazon.com: entity within FactForge
- Netflix: entity within FactForge

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com



RDF enables the data to be managed and persisted in isolation, yet re-joined pragmatically when required. The GraphDB knowledge graph can be queried in isolation using SPARQL and indeed the annotations within MongoDB can be queried using JSON queries.

Querying Web Annotations Directly

The platform is decomposed into cohesive bounded context chunks. These are aligned to problem spaces such as knowledge graphs and annotation.

Most platform annotation service calls are dealt with by directly querying the RDF (JSON-LD)

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

within MongoDB.

For example, the following MongoDB shell query will:

"Find Annotations, where the Resource (Unstructured Content) is annotated with "Amazon" or "Netflix", with relevance scores greater than .65 ordered by the sourceDate (publication date) of the target Resource"

“Find Annotations, where the Resource (Unstructured Content) is annotated with “Amazon” or “Netflix”, with relevance scores greater than .65 ordered by the sourceDate (publication date) of the target Resource ”

```
db.annotations.aggregate([
        {
                "$match": {
                        "$and": [
                                {
                                        "$or": [
                                                {
                                                        "body.source":
""
                                                },
                                                {
                                                        "body.source":
"resource:tsmrf7oy2j28"
                                                }
                                        ]
                                },
                                {
                                        "body.relevanceScore": {
                                                "$gte": "0.65"
                                        }
                                }
                        ]
```

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

```
                }
        },
        {
        "$sort": {
                "target.state.sourceDate.@date": -1
        }
}
])
```

GraphDB MongoDB Connector

In some cases, it is useful to join the annotation model with the knowledge contained within the knowledge graph. These types of use cases normally require graph traversal to provide more context to the results.

Ontotext has developed a MongoDB connector for GraphDB. It supports querying RDF stored within both data stores using a single combined GraphDB SPARQL+JSON query. Thus providing a pragmatic virtualized joint between GraphDB and MongoDB.

The integration between GraphDB and MongoDB is achieved by a GraphDB plugin that sends a request to MongoDB and then transforms the result into an RDF model.

It is assumed that the documents within MongoDB are valid JSON-LD. JSON returned by the MongoDB query that is not valid JSON-LD will be ignored and not included in the virtualized RDF graph.

Each MongoDB document should have its own context in order that CURIEs can be expanded into fully formed URIs.

Creating a Connector

The following SPARQL query creates a virtualized connection between GraphDB and a MongoDB collection. This allows combined SPARQL+JSON queries to be invoked to join the knowledge graph with the Web Annotations:

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

```
## Create MongoDb Connector
SPARQL Query:

PREFIX: <https://www.ontotext.com/connectors/mongodb#>
PREFIX inst: <https://www.ontotext.com/connectors/mongodb/instance#>
insert data {
        inst: blog - post: service "mongodb://localhost:27017";
                : database "blog-post";
                : collection "annotations".
}
```

The creation of a MongoDB connector is self-explanatory. The following predicates are supported and link directly to the MongoDB configuration.

- :service; Mongo connection string
- :database; database
- :collection; collection
- :user; (optional) user for the connection
- :password; (optional) password
- :authDb; (optional) database against which the user is authenticated

Querying Annotations with Knowledge

The following sample SPARQL query will join the annotations in MongoDB to the knowledge graph entities within GraphDB.

"Discover documents (resources) that are annotated with

Amazon.com

or

Netflix

, include the DBpedia industry and size of company (employee count)"

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

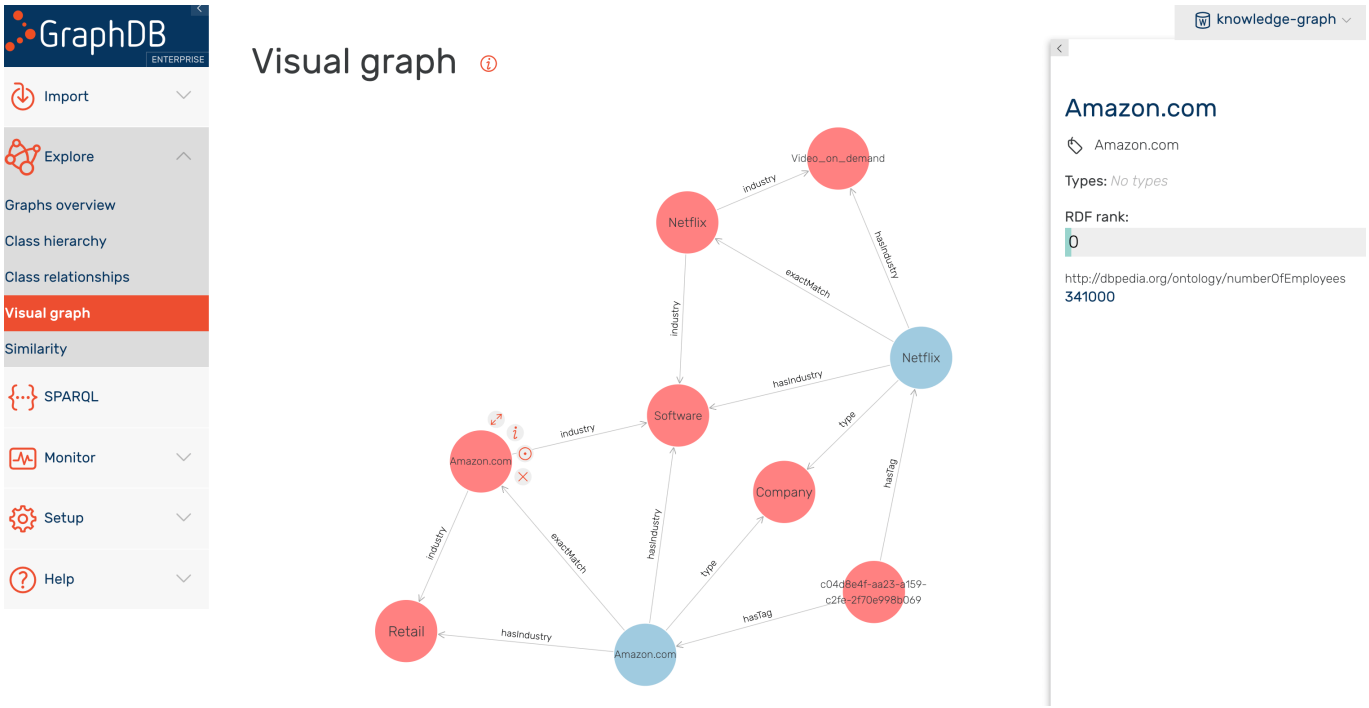
```
PREFIX inst: <https://www.ontotext.com/connectors/mongodb/instance#>
PREFIX: <https://www.ontotext.com/connectors/mongodb#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX tax: <http://ontology.ontotext.com/taxonomy/>
PREFIX dbpr: <http://dbpedia.org/resource/>
PREFIX dbpo: <http://dbpedia.org/ontology/>

select
        ?resource ?tag ?label ?numberOfEmployees
where {
        ?search a inst:blog-post ;
                :aggregate '''[
        {
                "$match": {
                        "$and": [
                                {
                                        "$or": [
                                                {"body.source":
"ontop:organization/Amazon"},
                                                {"body.source":
"resource:tsmrf7oy2j28"}
                                        ]
                                },
                                { "body.relevanceScore": { "$gte":
"0.65" } }
                        ]
                }
        },
        {
                "$sort": {"target.state.sourceDate.@date": -1}
        }
]''' ;
        :entity ?entity .
        graph inst:blog-post {
```

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

```
            ?annotation oa:hasTarget ?target ;
                                        oa:hasBody ?body .
            ?target oa:hasSource ?resource .
            ?body oa:hasSource ?tag .
        }
        ?tag rdfs:label ?label ;
                tax:exactMatch ?dbpediaResource .
        ?dbpediaResource dbpo:industry dbpr:Software ;
                                        dbpo:numberOfEmployees
?numberOfEmployees .
}
```

The result of that could be visualized by using GraphDB's SPARQL visualizer:



The MongoDB connector supports the following predicates, linked directly to MongoDB operations:

- **:find**; accepts single BSON and sets a query string. The value is used to call db.{collection}.find()

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

- **:project**; accepts single BSON. The value is used to select the projection for the results returned by :find.
- **:aggregate**; accepts an array of BSONs. Calls db.collection.aggregate(). This is the most flexible way to make a mongo query (the find() method is just a single phase of the aggregation pipeline). Will take precedence over :find and :project.
- **:graph**; accepts an IRI. It is used to set custom value for the named graph. If not set default to the IRI of the index.
- **:entity**; (REQUIRED) returns the IRI of the Mongo doc. So the value of @id or @graph.@id. If there are multiple values, the first one will be chosen and you will get a warning in the log. If the JSON-LD has no name graph then the value of @id node will be used.
- **:hint**; used to specify the index which should be used when executing the query (calls cursor.hint()

Closing

GraphDB's MongoDB connector unifies the Ontotext Platform's knowledge graph and annotation RDF stores. It combines JSON aggregate expressions with expressive SPARQL into unified queries. Supporting a global view across billions of Knowledge statements and billions of annotation documents.

GraphDbs MongoDB integration was released as part of GraphDb 8.8.0. For more information, please refer to Integrating GraphDb with MongoDB.

RDF is the core enabler that allows data to be managed and persisted in isolation, yet re-joined pragmatically when required.

Request Ontotext Platform

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com