The Ontotext Platform is cloud native, it is based on the premise that infrastructure is dispersed. The micro-service architecture tends to use a shared-nothing approach where each bounded context has its own code base, data store and team.

Platform micro-services utilize polyglot persistence to ensure that data is stored in an optimal manner. In addition, the platform attempts to isolate processing and storage concerns for different bounded contexts to ensure the platform components can scale independently.

As an example, 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 plain text

```
A document about Ontotext the organization....based in
Sofia...GraphDB....Text Analytics
```

could, for instance, be annotated with an example (cut-down) extended W3C Web Annotation JSON-LD document.

```
{
    "id": "resource:tsltbki6oj66/annotation/57283",
    "type": ["Annotation", "CurationAnnotation"],
    "body": {
        "id": "resource:tsltbki6oj66/annotation/57283/body",
        "type": ["Concept", "SpecificResource"],
        "class": "http://ontology.ontotext.com/Organisation",
        "confidence": "0.86321123",
        "preferredLabel": "Ontotext",
        "purpose": "tagging",
        "source": "resource:tsltbki6oi68",
        "status": "suggested",
        "tagType": "ann:about"
    },
    "dcterms:issued": {
```

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

```
        "type": "xsd:dateTime",
        "@value": "2018-09-27T08:29:18.118Z"
    },
    "generator": {
        "id": "ontop:tsltbki6ox34",
        "type": "Software",
        "name": "Ontotext Text Analytics"
    },
    "motivation": "tagging",
    "target": {
        "id": "resource:tsltbki6oj66/annotation/57283/target",
        "type": "Text",
        "selector": {
            "id":
"resource:tsltbki6oj66/annotation/57283/target/selector",
            "type": "TextPositionSelector",
            "end": "26",
            "start": "17"
        },
        "source": "resource:tsltbki6oj66",
        "state": {
            "id":
"resource:tsltbki6oj66/annotation/57283/target/state",
            "type": "TimeState",
            "sourceDate": "2017-08-01T00:01:01Z"
        }
    },
    "@context": ["http://www.w3.org/ns/anno.jsonld", {
        "onto": "http://ontology.ontotext.com/taxonomy/",
        "ontoa": "http://ontology.ontotext.com/annotation#",
        "nif":
"http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nifcore#",
        "xsd": "http://www.w3.org/2001/XMLSchema#",
        "ann": "http://data.ontotext.com/annotation/",
        "ontocontent": "http://ontology.ontotext.com/content#",
```


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

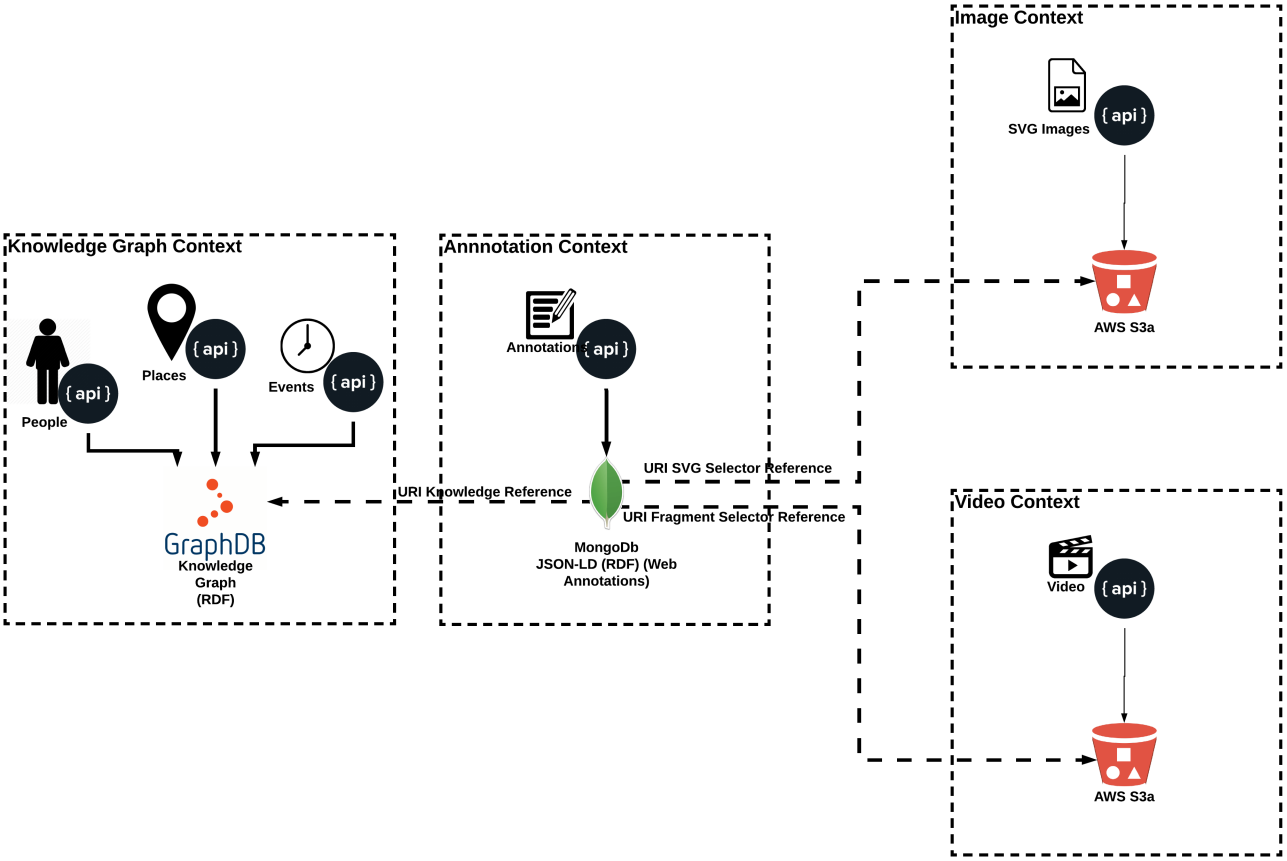
```
        "ontop": "http://data.ontotext.com/publishing/",
        "resource": "http://ontology.ontotext.com/resource/",
        "Concept": "onto:Concept",
        "CurationAnnotation": "ontoa:CurationAnnotation",
        "class": "ontoa:class",
        "confidence": {
            "@id": "nif:confidence",
            "@type": "xsd:double"
        },
        "relevanceScore": {
            "@id": "ontoa:relevanceScore",
            "@type": "xsd:double"
        },
        "status": "ontoa:status",
        "tagType": {
            "@id": "ontoa:tagType",
            "@type": "@id"
        }
    }]
}
```

People, Place and Event domain entities and relationships would be managed and stored within GraphDB, Web Annotations within MongoDB and perhaps plain text on AWS S3. Thus, the platform separates unstructured content, annotation and knowledge graph models into distinct bounded contexts.

The following 30,000ft diagram describes the Ontotext Platform bounded context for components that manage analyzed SVG's and Videos. The annotation semantic fingerprinting references People, Locations and Events within the knowledge graph.

Please, refer to a more detailed description of semantic annotation if you would like to understand more.

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



The platform design takes heed of typical domain driven design approaches:

> Explicitly define the context within which a model applies. Explicitly set boundaries in terms of team organization, usage within specific parts of the application, and physical manifestations such as code bases and database schema's.
>
> Eric Evans, Author of Domain-Driven Design

With careful consideration, the platform attempts to manage data using the most optimal representation and persistence mechanism.

GraphDB manages the data that represents a business domain. A business domain data

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

model requires interconnections, classification, inference and reasoning to represent a domain correctly. The RDF graph model is ideally suited and allows developers to solve the graph problem efficiently and with elegance. GraphDB represents information in a manner that is similar to how a human understands information and, in turn, provides a very suitable data representation for knowledge graphs.

We tend to recommend that a single team and bounded context are dedicated to the management of domain knowledge within a governed GraphDB knowledge graph. A knowledge graph bounded context allows multiple domain services (APIs) to use a shared GraphDB instance. Sharing GraphDB within this context provides cohesion, it also ensures that the team has deep domain knowledge and reduces the opportunity cost associated with domain separation.

Beyond the knowledge graph, the platform utilizes MongoDB to manage content annotations. An annotation may represent a content tag, classification, relationship or perhaps sentiment. The annotations are able to target sub-elements of the target content using Web Annotation selectors such as XPath or plain text position/offsets, etc. Web Annotation selectors allow the platform to support annotations that reference plain text, mark-up (XML, XHTML), PDFs, binaries, etc. The separation of annotations into its own bounded context ensures that annotation data does not pollute the knowledge model. Separation also supports independent processing and storage that allows the different context components to scale with different profiles.

A common platform use case requires processing and reprocessing of millions of unstructured text items using text analytics [TA] components. An archive may need to be processed/re-processed to 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.

Ontotext's text analytics components discover named entities, novel entities, relationships, classifications, sentiment, etc. within the unstructured content. The TA services can represent suggestions using Ontotext's extended version of WA JSON-LD. Annotations capture the semantic fingerprint of unstructured content; the structured knowledge contained within the unstructured content using URI references to the knowledge graph.

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

Annotations also include quantitative attribution such as confidence or relevance. Annotations are published as events to an event queue to allow processing to be performed in an asynchronous fashion. The events are consumed and, in some cases, the annotations are moderated by a team of annotators using the platform's Inter Annotation Agreement "Curation" tooling (the follow up blog post – *Ontotext Platform: Semantic Annotation Quality Assurance & Inter-Annotator Agreement* describes the Inter Annotation Curation tooling and its effects on data quality). After processing, the annotations are made persistent within MongoDB.

Most platform annotation service calls such as "*Find me all the content which mentions entity X and Y*" are dealt with by directly querying the Web Annotation RDF (JSON-LD) within MongoDB. However, 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. Typically, use cases of this type can be dealt with at a service layer by combining multiple query results (SPARQL [GraphDB] and JSON [MongoDB]).

For a simpler, streamlined developer experience, we have 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 join between GraphDB and MongoDB.

GraphDB's MongoDB integration was released as part of GraphDB 8.8.0. For more information, please refer to Integrating GraphDB with MongoDB.

Please also take a look at another follow-up blog post – *Ontotext Platform: A Global View Across Knowledge Graphs and Content Annotations* to get more details and insight as to how the Ontotext Platform manages Web Annotations.

Closing

Ontotext's micro-service, bounded context platform architecture follows a pattern that has worked well for a number of integrations. We always sensibly moderate and govern the intake of quick fixes and exploit shared RDF stores only when there is maximum effect and return. We decompose our platform into cohesive chunks aligned to problem spaces such as

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

knowledge graphs and annotation. We support integration using event sourcing, messaging and, in some cases, we support integration via shared databases and connectors, to provide cost-effective, performant virtualization.

The Ontotext Platform and, indeed, the solutions that Ontotext builds for its clients are based on well defined bounded context architectures. The best data integration approach, with the right level of cohesion, is chosen to provide the most effective and pragmatic win for the business problem at hand.

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

Ontotext have developed MongoDB integration with GraphDB to join and query RDF stored within GraphDB and MongoDB. This supports bounded context services, which need to join and integrate data across multiple shared stores using an elegant, low effort connector.

Request Ontotext Platform

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