ontotext

To get the most out of Ontotext Platform and its use of GraphQL, your organization should expose a single knowledge graph. This provides a unified interface for querying all of your data sources in context and allows clients to fetch data from any number of data sources simultaneously, without needing to know which data comes from which source.

> Our friends from the GraphQL Federation have pledged their support. And when Ontotext Platform's Semantic Objects are combined with yours, we shall have an army greater than any in the galaxy. The Jedi will be overwhelmed. The Republic will agree to any demands we make.
>
> Count Dooku

A single GraphQL model acts as a language and grammar that aids communication between developers, domain experts and clients.

Contents

- Knowledge Graph Training
- Web Annotation GraphQL Service
- Find Annotations with Droid tags
- Star Wars Universe GraphQL Service
- Droid Character Entity Resolution
- Federated Annotation and Star Wars Universe
- Character Similarity GraphQL Service
- Character similarity and sub class entity resolution
- GraphQL Federation over GraphDB, MongoDB and a Semantic Vector Space
- Schema Definition Language Query
- Federated Annotation and Character Similarity
- Federated Annotation, Character Similarity and Universe
- Trade Federation

Those of you who are familiar with RDFS/OWL will know that a good model acts as the

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

foundation for the design and build of all the software and APIs that make use of it. A federated, bounded context GraphQL model that simplifies and aggregates data from various source forms such as JSON-LD, RDF or perhaps a Semantic Vector Space will undoubtedly improve developer productivity and understanding.

As you model and engineer a large knowledge graph it can become very difficult to manage complexity. Building a knowledge graph, as with any complex system, requires significant time and expertise. For many, the magnitude of resources required for building and maintaining a complete knowledge graph is untenable. Parts of the model are often owned by different domain experts and/or developers, the APIs and code that interact with subgraphs become confused and ambiguity arises between entities within the graph.

It is therefore often prudent to split a knowledge graph into bounded contexts, where teams and domain experts model subgraphs and write code and separate API services in relative isolation. Providing a ubiquitous language between boundaries.

GraphQL federation supports object/entity extension within bounded context services. This allows separate services and models to evolve, using different technologies, persistence mechanisms, support different performance characteristics, all with low cohesion. Yet, at the same time, it ensures that a federated GraphQL model contributes to a wider knowledge graph through its shared objects/entities.

GraphQL federation is simple, it's just GraphQL. Existing GraphQL clients can consume federated GraphQL services without change. Services can be built using the large eco-system of tooling, quickly and efficiently and back-end data-sources that do not already have a GraphQL facade can be exposed as a GraphQL API, conforming to the declarative GraphQL federation spec with ease.

Several Graph database providers proclaim that sharding 100s of billions of triples/quads into a single (often massive!) knowledge graph database cluster is, in fact, the target architecture that everyone should strive for. After which query re-writing and/or backward chaining should be used to materialize inferred knowledge? I would disagree that this is naive at the very best and often unduly expensive. It is this kind of "magic"/"silver-bullet" that gives knowledge graph-based solutions a bad reputation.

Graphs are complex, they usually have many, many joins. Graph queries across this kind of

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

monolithic architecture, where a complex graph straddles many domain contexts, will inevitably use remote joins across many shards, leading to performance overheads and degradation. When RDF databases are used inappropriately, they exert a significant drag on application development.

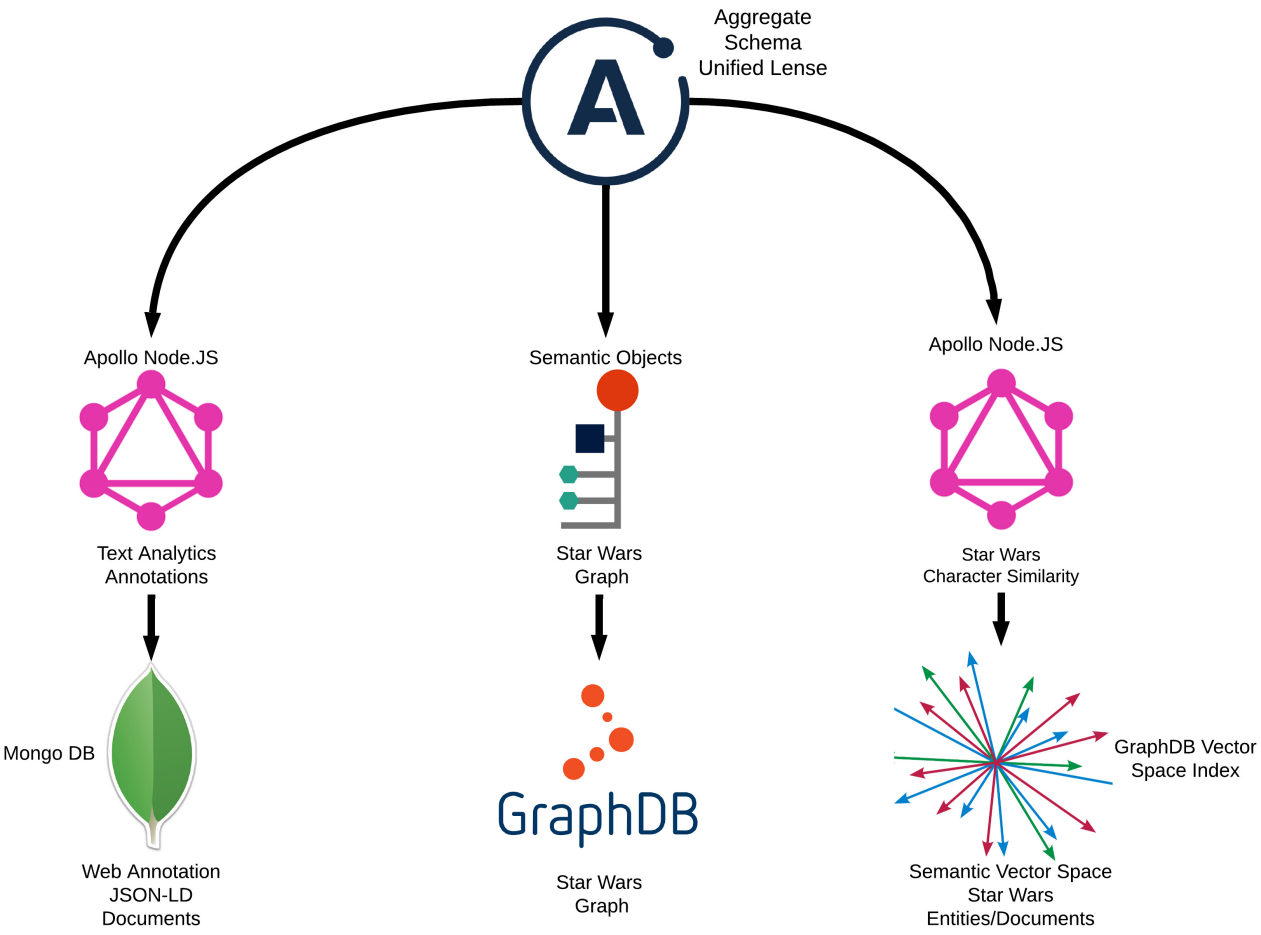
A knowledge graph that uses one data representation, technology choice, index and indeed monolithic model, has large cohesion, causes developer pain, reduces velocity and does not make the best pragmatic use of technology separation, persistence and appropriate indexing for the data or context in question.

In many scenarios, it is wise to divide and conquer. When designing a data model, always think about where and how the data will be used and what use-cases (or contexts) are appropriate for isolation and join performance.

Knowledge Graph Training

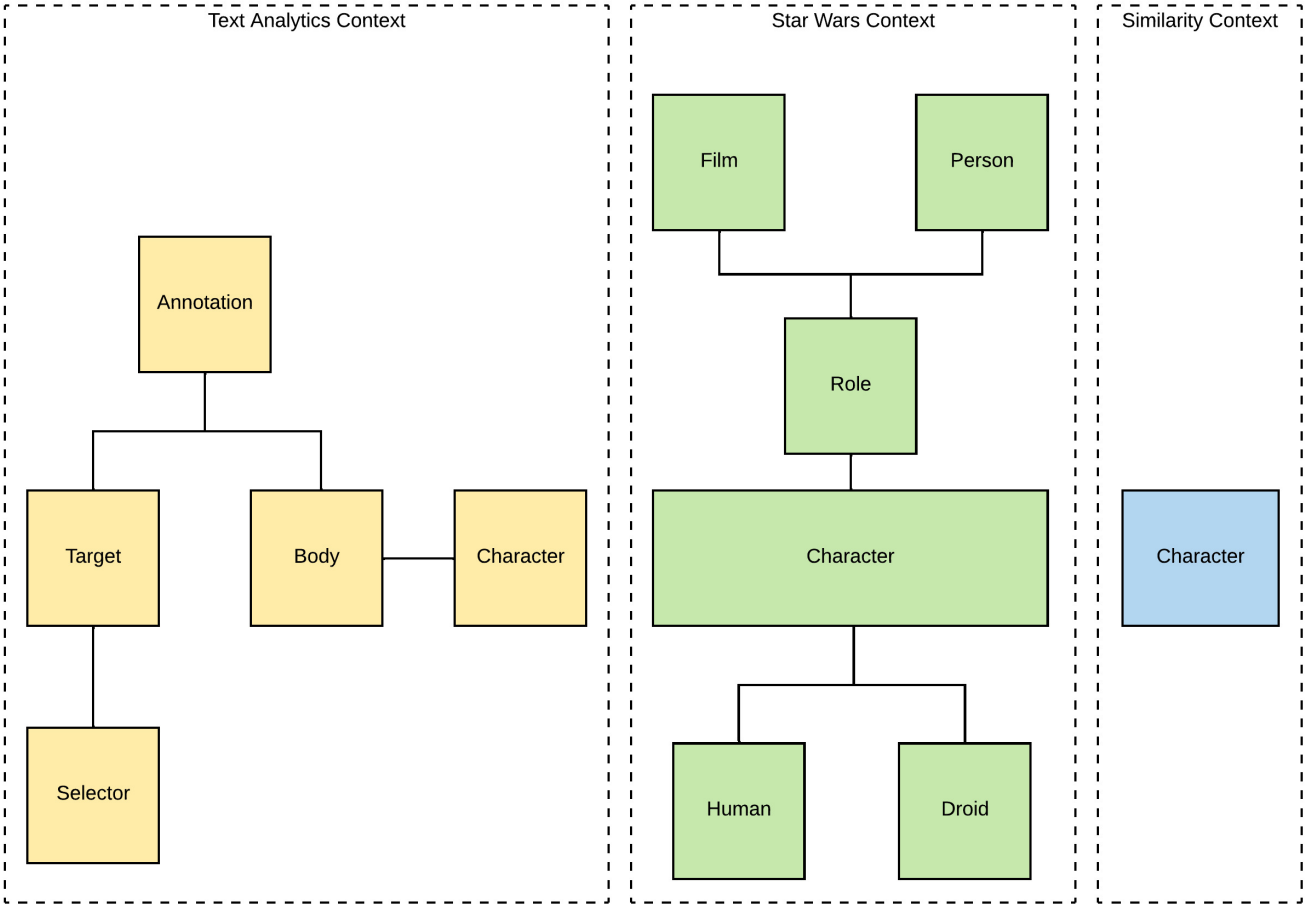
Take the following simple demo system used as part of Ontotext’s knowledge graph training:

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



This architecture diagram is supplied alongside a sample Apollo GraphQL Server code (available here: GraphQL Annotation Service and here GraphQL Similarity Service). The code bootstraps delegates learning, allowing them to build and deploy a federated knowledge graph within a couple of hours. The outcome of this bounded context data model is as follows:

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



A bounded context data model comprised of:

- text analytics: Web Annotations
- Star Wars universe: Films, Characters, Roles, Actors, Spaceships..
- Similarity: between Characters

This federated/bounded-context modelling approach, allows part of the training class to build out a text analytics GraphQL Annotation service whilst another set of delegates configure a Semantic Objects service to auto-generate a Star Wars Universe GraphQL API over RDF. With the remainder building out a GraphQL similarity service using GraphDB's semantic vector space.

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

ontotext

The teams build the code independently yet still conform and extend shared objects (in this case, Star Wars Characters and the many Species/Sub-Classes) across a single federated GraphQL graph.

So let's talk a little more about the services within the training demo.

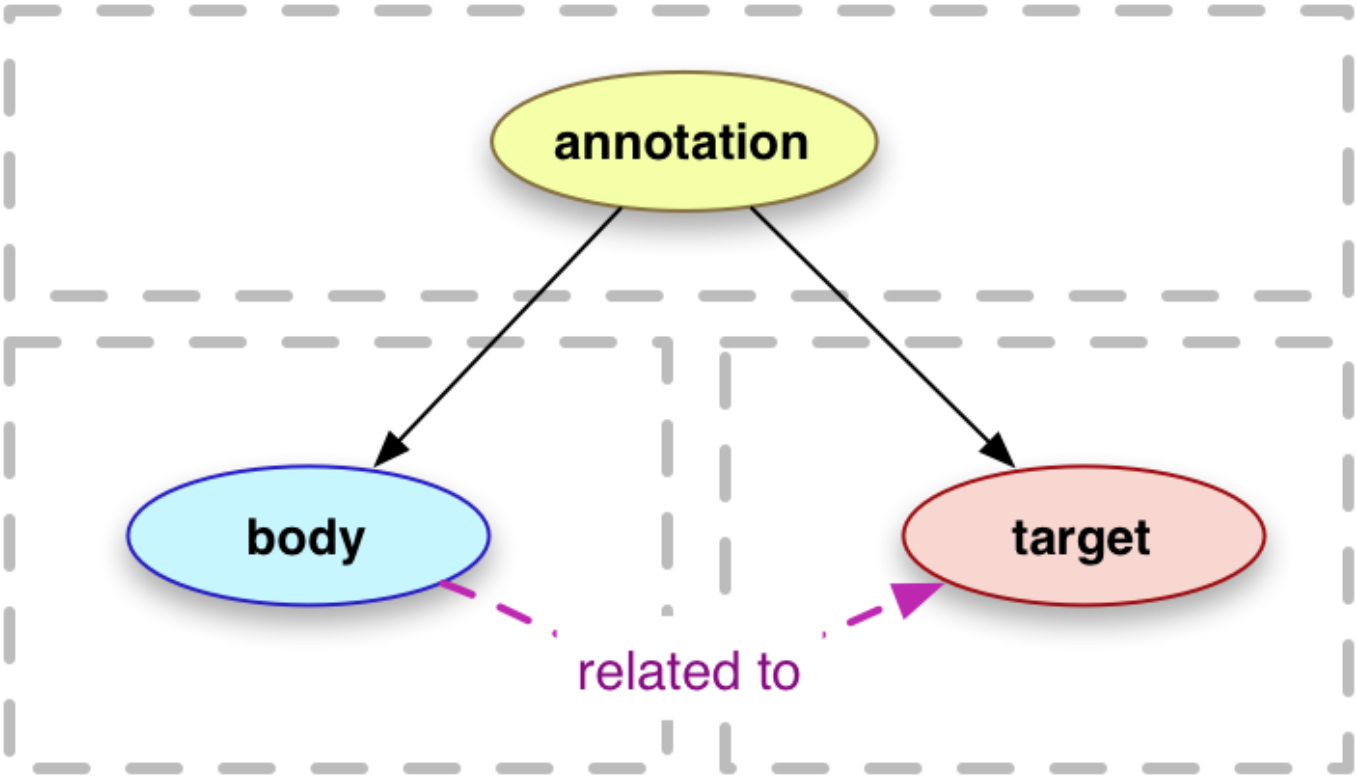
Web Annotation GraphQL Service

Ontotext Platform's text analytics service annotates un-structured/semi-structured content using the W3C's Web Annotation model.

Ontotext Platform Web Annotations use source knowledge graph identifiers <https://swapi.co/resource/human/11> to unambiguously identify subjects such as `Luke Skywalker` mentioned within unstructured/semi-structured content. Using the Web Annotation model, a body describes a target in some manner, in this case, its motivation is to tag a target, with an unambiguous knowledge graph entity.

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

ontotext

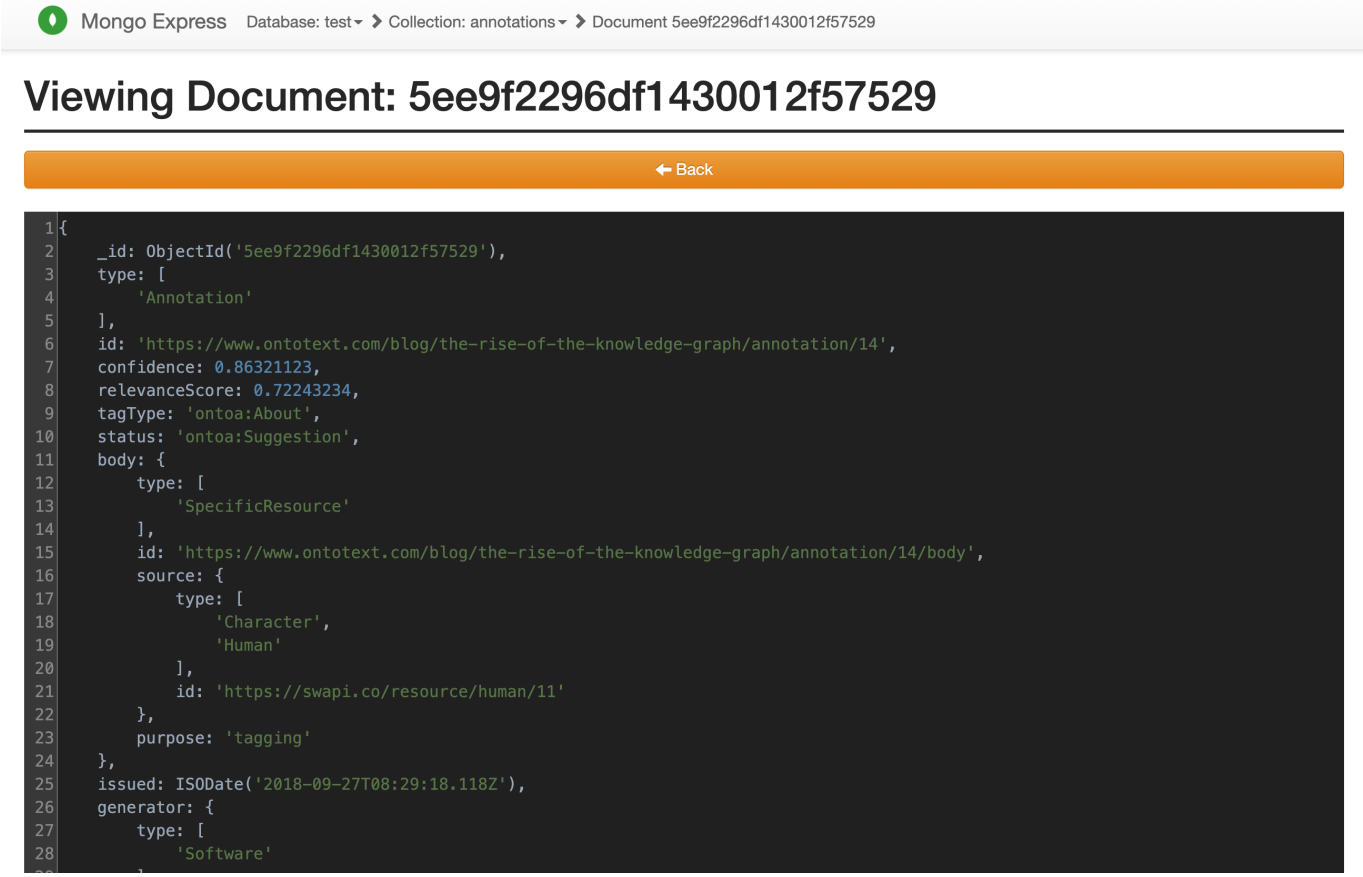


The GraphQL annotation service includes example Web Annotations that target a blog post.

You can view, examine and query the sample JSON-LD stored within MongoDB by using Mongo-Express and clicking on the following screenshot.

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

ontotext



This particular Web Annotation targets one of my previous blog posts Return of the Jedi: Ontotext Platform Metamorphosis. The annotation tags and selects the text `Luke Skywalker` by using an XPath selector:

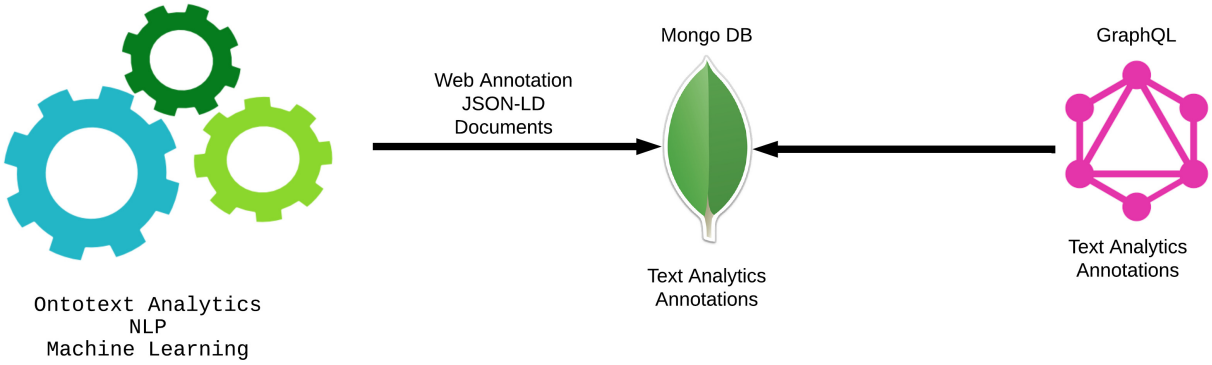
substring(/html/body/div[1]/div[2]/div/div[2]/div/div/div[1]/div/div[8]/div/div/div/div[3]/div/p[31],56,16)

The annotation also includes metrics such as confidence and relevance ascertained from the underlying machine learnt model.

Ontotext Platform manages billions of JSON-LD Web Annotation documents using the very same approach described in the training code. MongoDB stores JSON annotations in a shared fashion, and they are indexed to support isolated analytics re-processing and high-performance query execution. Allowing service infrastructure to be scaled out/down with less

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

impact on other knowledge graph contexts.



Web Annotations represented as simple JSON documents have relatively low complexity and are indeed a perfect data model for a JSON document store such as MongoDB. A database tuned to index, shard, aggregate and query JSON documents at massive scale.

I have deployed the training GraphQL Annotation API so that you can give it a try. This particular GraphQL query filters the demo set of annotations by Character species such as “Droid” or “Human”.

Web Annotation GraphQL Service: Find Annotations with Droid tags

Loading…

https://swapi-platform.ontotext.com/annotation
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TD
A0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOj
E4MzMwMzIyMTQsImlhdCI6MTY3NTM1MjIxNCwiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3RleH
QuY29tIiwic3ViIjoiMDY4MzFkZTMtM2I0Ny00ODM3LTk2MjgtYWFiN2EwNWM2YTc1IiwianRpIjoiN
TM1MTA1MzctMmI3Ny00Mjk0LWJhZWUtNGI1MDU1MDEzYWFhIiwiYXV0aGVudGljYXRpb25UeX
BlIjoiUEFTU1dPUkQiLCJlbWFpbCI6ImRyb2lkaHV0dHJlYWRAZXhhbXBsZS5jb20iLCJlbWFpbF92Z
XJpZmllZCI6dHJ1ZSwicHJlZmVycmVkX3VzZXJuYW1lIjoiZHJvaWRodXR0cmVhZCIsImFwcGxpY2F

0aW9uSWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJyb2xlcyI6WyJEcm9pZEh1dHRSZWFkIl0sImRhdGEiOnt9fQ.W0WU7-3-SPCIfV33WG1ojgIjXdyOXxiexQhl-Sb1OEg
true
query droidAnnotations ($characterType: String!) { annotation(where: {body: {source: {type: {EQ: $characterType}}}}) { id __typename confidence relevanceScore tagType status issued motivation body { id __typename source { id __typename } purpose } target { id __typename selector { id __typename value } source state { id __typename sourceDate } } generator { id __typename name } } }
{ “characterType”: “Droid” }

Most annotation queries can be self-contained within the Annotation bound context. However, you may have noticed that the sample annotations capture very limited data describing the Characters that tag the blog post. The Web Annotations only include a URI that represents a Specific Resource or Character such as `Luke Skywalker`. Although limited, these URIs are very powerful! They act as a unique identifier across the federated data set allowing possibilities for remote join and object-extension with data residing in separate/isolated data stores.

You can, for example, find similar Characters, using their unique id or perhaps films, roles or even the spaceships that the Character pilots, from data residing within the other contexts.

If you take a look at the Annotation GraphQL schema, you will observe how the GraphQL service transforms JSON-LD into a GraphQL response. It is particularly valuable to check the following section of the schema that captures the relationship between an Annotation, its Body and its source Star Wars Character.

```
type Annotation

  @key(

fields: “id”

) {

id: ID!
```

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

type:

[

String

]

confidence: Float

!

relevanceScore: Float

!

status: String

!

tagType: String

!

body: Body

!

target: Target

!

motivation: String

issued: Date

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

```
generator: Generator

  }

type Body

@key

(

fields

:

“id”

) {

id

:

ID

!

type

: [

String

]
```

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

ontotext

```
source
:
Character
!
purpose
:
String
!
  }

interface Character

@key
(fields:
“id”
) {
id
:
ID
!
  }
```

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

ontotext

Importantly, the above GraphQL schema declares the interface Character with a federation @key directive. This declaration defines the objects join point (and federation identity) so that it can be used in remote joins or fetches and extensions.

Star Wars Universe GraphQL Service

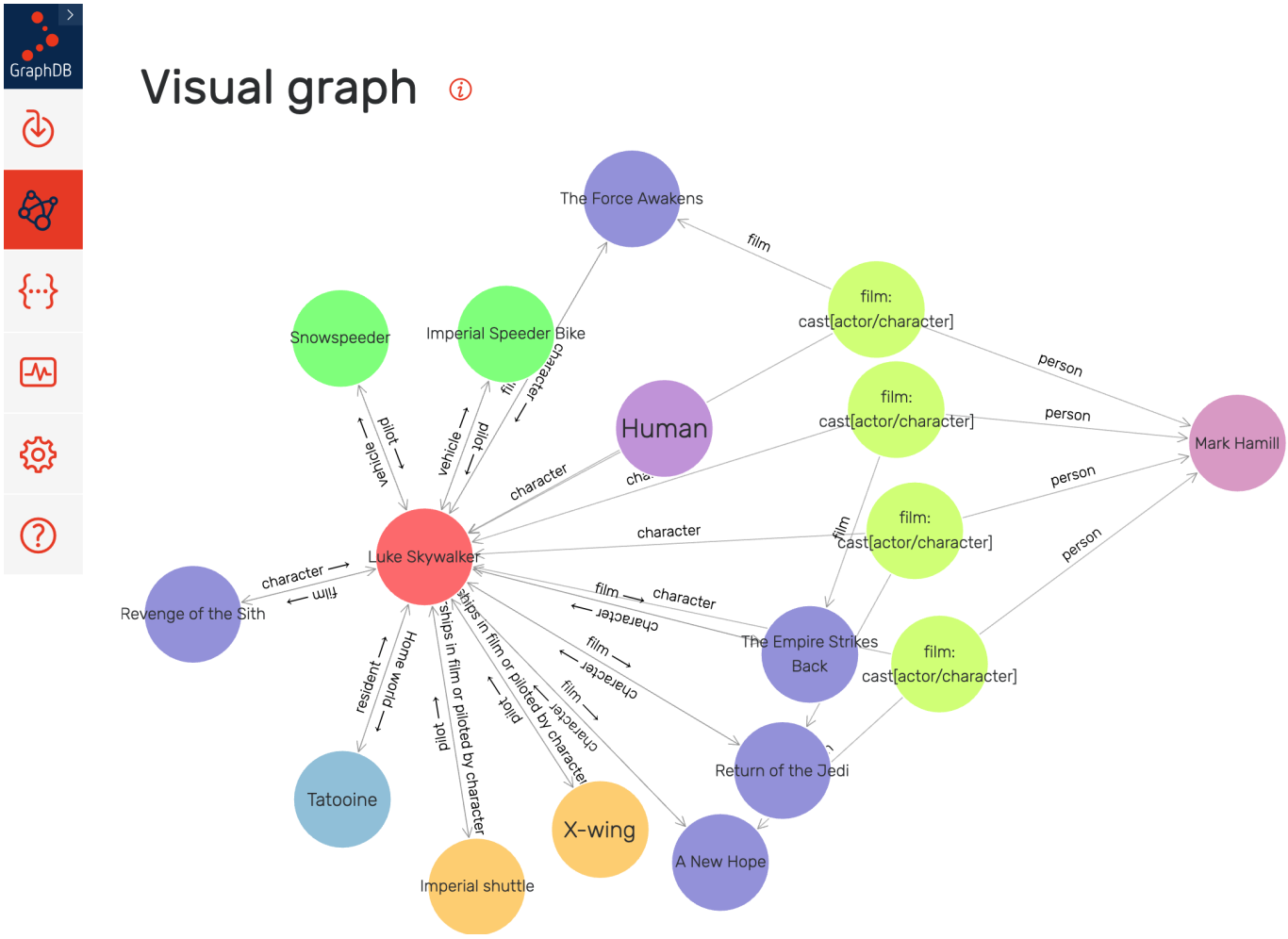
So, it's now possible to invoke GraphQL queries over MongoDB to retrieve Web Annotation JSON-LD. But now, let's join Web Annotation JSON-LD with Star Wars RDF data.

We can do this by making minor declarative adjustments to the Star Wars GraphQL service described at length in these two blog posts:

- A New Hope: The Rise of the Knowledge Graph
- Return of the Jedi: Ontotext Platform Metamorphosis

FYI: You can view, examine and query the raw Star Wars RDF data directly within GraphDB:

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



We can easily modify the Star Wars GraphQL service by changing the SOML schema to extend the Character type with properties and data that are available within GraphDB.

This is very simple and can be achieved by augmenting the Star Wars SOML Object definition with the **“extend: true“** property.

Character

:

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

ontotext

kind

:

abstract descr

:

“A Character in a star wars film”

name

:

rdfs:label

extend: true props

:

desc

: {

label

:

“Description”

}

eyeColor

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

ontotext

: {

descr

:

“Characters eye colour, including Droid eyes, such as R2-D2’s red eye!”

}

hairColor

: {

descr

:

“Characters hair colour”

}

skinColor

: {

descr

:

“Characters skin colour”

}

birthYear

: {

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

descr

:

“In BBY (Before the Battle of Yalvin) or ABY (After the battle of Yalvin”

}

film

: {

descr

:

“Star Wars films appeared in”

,

max

:

inf

,

range

:

Film

}

height

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

: {

label

:

“Height in meters”

,

range

:

decimal

}

mass

: {

label

:

“Mass in Kg”

,

range

:

decimal

}

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

homeworld

: {

label

:

“Characters homeworld(planet)”

,

range

:

Planet

}

starship

: {

label

:

“Characters starship(s)”

,

max

:

inf

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

,

range

:

Starship

}

vehicle

: {

label

:

“Characters vehicles(s)”

,

max

:

inf

,

range

:

Vehicle

}

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

species

: {

label

:

“Characters species”

,

range

:

Species

,

rdfProp

:

“rdf:type”

,

rangeCheck

:

true

}

gender

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

```
: {
label
:
“Gender”
}
        filmRole: {
inverseAlias
:
character
,
range
:
FilmRole
,
max
:
inf
}
```

After which, the Semantic Objects service will auto-generate a schema that complies with the federation specification. Extended objects such as Character, Human or Droid will be

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

ontotext

annotated with federation directives (such as @key, @external etc..) automatically:

```
interface Character

@key(fields : “id”) @extends

 {
   "IRI"

id

:

ID

!  @constraints(

minCount

  :

1

,

maxCount

  :

1

)  @external

    "type"
```

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

ontotext

type

(

orderBy

:

_OrderBy

,

limit

:

PositiveInteger

,

offset

:

PositiveInteger

,

ID

: [

ID

!],

where

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

ontotext

:

ID_Where_Multi

): [

ID

] **@constraints**

"Name"

name

:

String

! **@constraints**(

minCount

:

1

,

maxCount

:

1

)

"Description"

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

```
desc
:
String
  @constraints(
maxCount
  :
1
)
        "eyeColor"
eyeColor
:
String
  @constraints(
maxCount
  :
1
)
        "hairColor"
hairColor
```

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

ontotext

:

String

@constraints(

maxCount

:

1

)

#....
}

The Semantic Objects service will use the Star Wars Universe URIs/ids as object @keys, to enable federated joins and object extensions.

Additionally, generating an object entity resolution root query _entities, so that _Any extended type can be requested from the Star Wars universe and resolved automatically (with zero code).

_entities

(

representations

: [

_Any

!]!): [

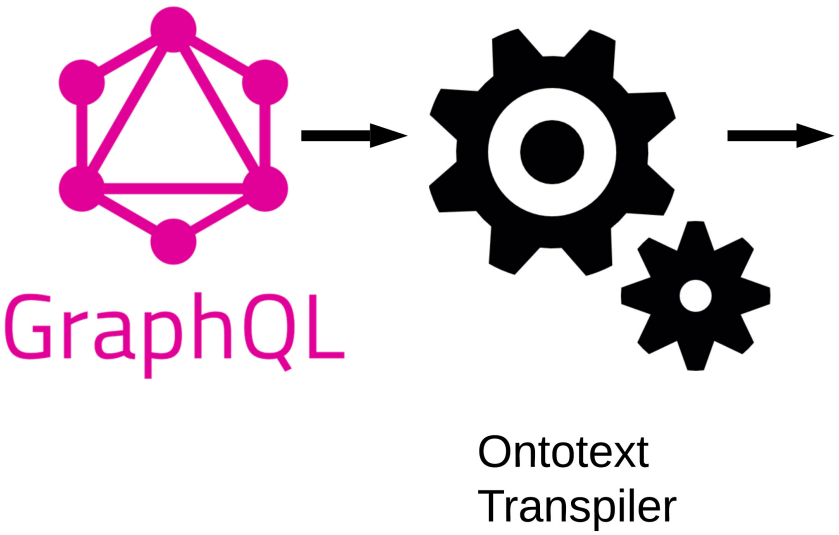
Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

_Entity

] !

Zero code Transpiler resolvers



For example, you may want to retrieve the name of each Droid tagged within this blog post as part of a federated query. The GraphQL federation service would achieve this by invoking the following _entities GraphQL query. It would bind the GraphQL variables to the Annotations' Character ids.

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

ontotext

Star Wars Universe GraphQL Service: Droid Character Entity Resolution

Loading...

https://swapi-platform.ontotext.com/graphql
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TDA0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOjE4MzMwMjU3NTMsImlhdCI6MTY3NTM0NTc1MywiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3RleHQuY29tIiwic3ViIjoiMDY4MzFkZTMtM2I0Ny00ODM3LTk2MjgtYWFiN2EwNWM2YTc1IiwianRpIjoiNzllN2FmNjQtYmQyZC00M2Y2LThlNTEtOTg5YmEwYThlZTIzIiwiYXV0aGVudGljYXRpb25UeXBlIjoiUEFTU1dPUkQiLCJlbWFpbCI6ImRyb2lkaHV0dHJlYWRAZXhhbXBsZS5jb20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwicHJlZmVycmVkX3VzZXJuYW1lIjoiZHJvaWRodXR0cmVhZCIsImFwcGxpY2F0aW9uSWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJyb2xlcyI6WyJEcm9pZEh1dHRSZWFkIl0sImRhdGEiOnt9fQ.rvXyRVaDqpq7H4uKoDpD-9gyr-VFBvCx6zxojigvQeo
true

query($representations: [_Any!]!) { _entities(representations: $representations) { ... on Aleena { name } ... on Besalisk { name } ... on Cerean { name } ... on Chagrian { name } ... on Clawdite { name } ... on Droid { name } ... on Dug { name } ... on Ewok { name } ... on Geonosian { name } ... on Gungan { name } ... on Human { name } ... on Hutt { name } ... on Iktotchi { name } ... on Kaleesh { name } ... on Kaminoan { name } ... on Keldor { name } ... on Mirialan { name } ... on Moncalamari { name } ... on Muun { name } ... on Nautolan { name } ... on Neimodian { name } ... on Pauan { name } ... on Quermian { name } ... on Rodian { name } ... on Skakoan { name } ... on Sullustan { name } ... on Tholothian { name } ... on Togruta { name } ... on Toong { name } ... on Toydarian { name } ... on Trandoshan { name } ... on Twilek { name } ... on Umbaran { name } ... on Vulptereen { name } ... on Wookiee { name } ... on Xexto { name } ... on Yodasspecies { name } ... on Zabrak { name } } }
{ “representations” : [{ “__typename” : “Droid”, “id” : “https://swapi.co/resource/droid/87” }, { “__typename” : “Droid”, “id” : “https://swapi.co/resource/droid/2” }, { “__typename” : “Droid”, “id” : “https://swapi.co/resource/droid/23” }, { “__typename” : “Droid”, “id” : “https://swapi.co/resource/droid/3” }, { “__typename” : “Droid”, “id” : “https://swapi.co/resource/droid/75” }, { “__typename” : “Droid”, “id” : “https://swapi.co/resource/droid/3” }] }

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

ontotext

The Star Wars GraphQL API can then combine Annotations within MongoDB with any Semantic Object Character property from GraphDB.

For example, the following GraphQL query retrieves an Annotation from the Annotation service (MongoDB) joined with the Character's name and film roles (person/actor and film) from the Star Wars Universe Service (GraphDB).

All this without the GraphQL client needing to know where the data actually resides.

Web Annotation and Star Wars Universe GraphQL Services : Retreive Annotation with Star Wars Universe objects and properties

Loading…

https://swapi-platform.ontotext.com/federation
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TDA0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOjE4MzMwMzE4OTQsImlhdCI6MTY3NTM1MTg5NCwiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3RleHQuY29tIiwic3ViIjoiZTlmOWQzNGQtZThmMS00ODM1LTlkMzAtOWRjNmU5YjQ4ZmMzIiwianRpIjoiMTZmZDJjOTUtNzNkZi00ZGE4LTkwMDgtMTRhYmQ1NTIyN2RhIiwiYXV0aGVudGljYXRpb25UeXBlIjoiUEFTU1dPUkQiLCJlbWFpbCI6InJlYWRvbmx5dXNlckBleGFtcGxlLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJSZWFkT25seVVzZXIiLCJhcHBsaWNhdGlvbklkIjoiNjcyY2E3YjMtYzM3Mi00ZGYzLTgyZDgtOWExYTBkN2Q2OGMxIiwicm9sZXMiOlsiUmVhZE9ubHkiXSwiZGF0YSI6e319.oGnOYmgdMPtcEK0mrEFOIiNXvrPC4W6m6PBYthP42QQ
true

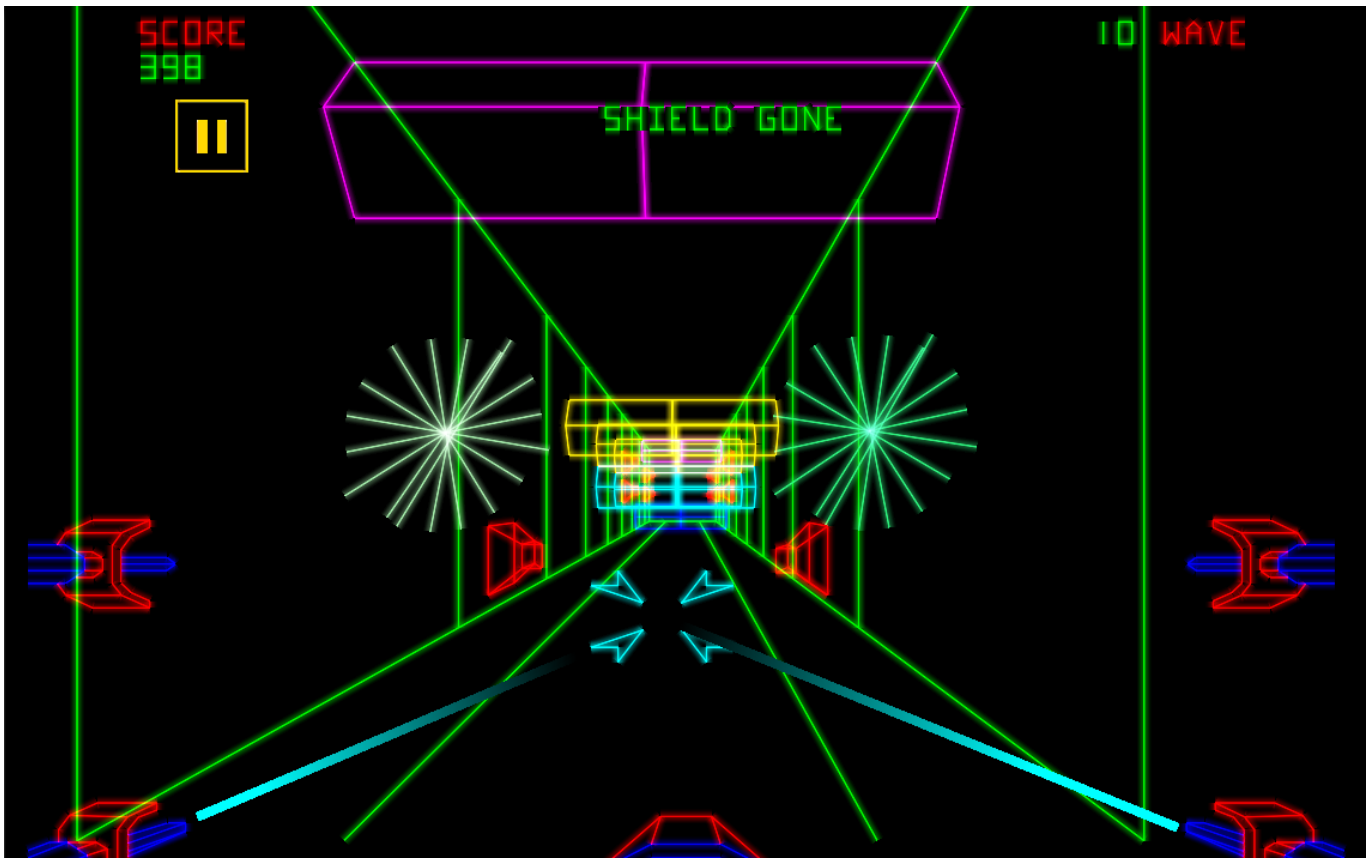
query($annotationId: ID!) { annotation(id: $annotationId) { body { source { name homeworld { name } role { person { name } film { name } } } purpose } confidence relevanceScore tagType status issued motivation target { id __typename selector { id __typename value } source state { id __typename sourceDate } } generator { id __typename name } } }
{ "annotationId": "https://www.ontotext.com/blog/the-rise-of-the-knowledge-graph/annotation/1" }

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

ontotext

Character Similarity GraphQL Service

Next, let us introduce the GraphQL Character similarity service.



Humans determine the similarity between documents based on the similarity of the words that a document is composed of and indeed their abstract meaning. Documents containing similar words are therefore semantically related, and words that frequently co-occur within the documents are also considered important.

The Star Wars Characters within the demo RDF data set have a number of literals that can be used to describe a Character. These literals include description, hairColour, eyeColour, height, etc.

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

ontotext

So, if we aggregate all the literals that describe a particular Character into a document per Character, we can use these documents to create a semantic vector space index for identifying similarity between Characters.

The GraphQL service makes use of GraphDB's embedded high-performance semantic vector space. GraphDB integrates a semantic vector library and uses the underlying Random Indexing algorithm to create it. This algorithm uses a tokenizer to translate documents into sequences of words (terms) and uses these terms to represent the documents in a vector space model. Once indexed, the vector space represents the document's abstract meaning.

A distinctive feature of the algorithm is its use of dimensionality reduction, based on Random Projection, where the initial vector state is generated randomly. When indexing each document, the term vectors are adjusted based on the word context. This approach makes the algorithm highly scalable and performant for very large document corpora.

The similarity GraphQL service builds a semantic vector space index over the Star Wars Character literals using the following SPARQL update.

```
PREFIX : <https://www.ontotext.com/graphdb/similarity/> PREFIX inst: <https://www.ontotext.com/graphdb/similarity/instance/> PREFIX pred: <https://www.ontotext.com/graphdb/similarity/psi/> INSERT { inst:embeddings :createIndex "-termweight idf" ; :analyzer "org.apache.lucene.analysis.en.EnglishAnalyzer" ; :documentID ?documentID .

?documentID :documentText ?documentText . } WHERE { SELECT ?documentID (GROUP_CONCAT(?texts) as ?documentText) { ?documentID ?p ?texts . filter(isLiteral(?texts)) } GROUP BY ?documentID }
```

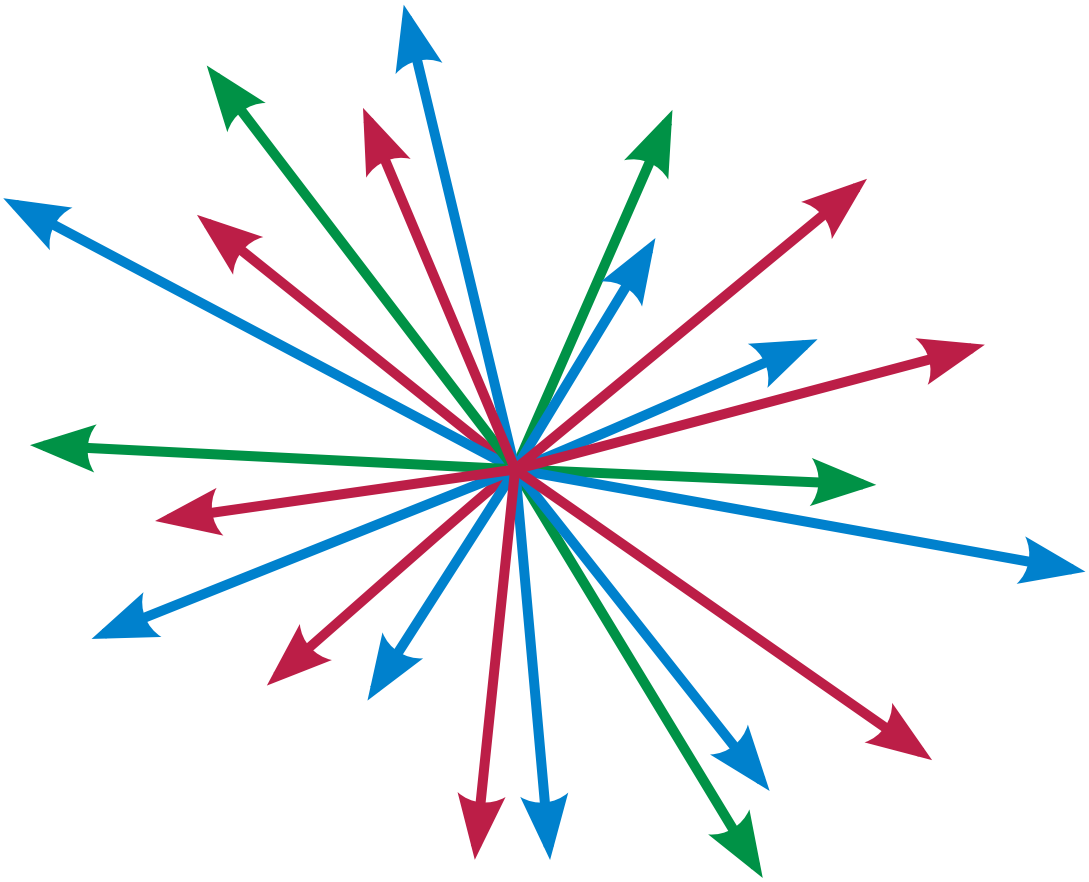
For example, this SPARQL update would first generate a document for Obi-Wan Kenobi by combining its literals. The documents terms and co-occurances are then analysed and extracted so that they can be used within its vector.

""Obi-Wan Kenobi 57BBY Obi-Wan "Ben" Kenobi is a fictional character in the Star Wars franchise. Within the original trilogy he is portrayed by Sir Alec Guinness, while in the prequel trilogy a younger version of the character is portrayed by Ewan McGregor. In the original

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

ontotext

trilogy, he is a mentor to Luke Skywalker, to whom he introduces the ways of the Jedi. In the prequel trilogy, he is a master and friend to Anakin Skywalker. He is frequently featured as a main character in various other Star Wars media., “Sir Alec Guinnesss portrayal of Obi-Wan in the original Star Wars (1977) remains the only time an actor has received an Oscar nomination (Best Supporting Actor) for acting in a Star Wars film.” blue-gray male auburn, white 182.0 77.0 fair Human””



The GraphQL similarity service then also exposes an _entities resolution query in order that it is possible to extend Characters tagged within an Annotation with those Characters that are similar (within the vector space).

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

ontotext

Star Wars Universe GraphQL Service : Character similarity and sub class entity resolution

Loading...

https://swapi-platform.ontotext.com/similarity
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TDA0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOjE4MzMwMjU3NTMsImlhdCI6MTY3NTM0NTc1MywiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3RleHQuY29tIiwic3ViIjoiMDY4MzFkZTMtM2I0Ny00ODM3LTk2MjgtYWFiN2EwNWM2YTc1IiwianRpIjoiNzllN2FmNjQtYmQyZC00M2Y2LThlNTEtOTg5YmEwYThlZTIzIiwiYXV0aGVudGljYXRpb25UeXBlIjoiUEFTU1dPUkQiLCJlbWFpbCI6ImRyb2lkaHV0dHJlYWRAZXhhbXBsZS5jb20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwicHJlZmVycmVkX3VzZXJuYW1lIjoiZHJvaWRodXR0cmVhZCIsImFwcGxpY2F0aW9uSWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJyb2xlcyI6WyJEcm9pZEh1dHRSZWFkIl0sImRhdGEiOnt9fQ.rvXyRVaDqpq7H4uKoDpD-9gyr-VFBvCx6zxojigvQeo
true

query ($representations: [_Any!]!) { _entities(representations: $representations) { ... on Human { similar { __typename id ... on Aleena { __typename id } ... on Besalisk { __typename id } ... on Cerean { __typename id } ... on Chagrian { __typename id } ... on Clawdite { __typename id } ... on Droid { __typename id } ... on Dug { __typename id } ... on Ewok { __typename id } ... on Geonosian { __typename id } ... on Gungan { __typename id } ... on Human { __typename id } ... on Hutt { __typename id } ... on Iktotchi { __typename id } ... on Kaleesh { __typename id } ... on Kaminoan { __typename id } ... on Keldor { __typename id } ... on Mirialan { __typename id } ... on Moncalamari { __typename id } ... on Muun { __typename id } ... on Nautolan { __typename id } ... on Neimodian { __typename id } ... on Pauan { __typename id } ... on Quermian { __typename id } ... on Rodian { __typename id } ... on Skakoan { __typename id } ... on Sullustan { __typename id } ... on Tholothian { __typename id } ... on Togruta { __typename id } ... on Toong { __typename id } ... on Toydarian { __typename id } ... on Trandoshan { __typename id } ... on Twilek { __typename id } ... on Umbaran { __typename id } ... on Vulptereen { __typename id } ... on Wookiee { __typename id } ... on Xexto { __typename id } ... on Yodasspecies { __typename id } ... on Zabrak { __typename id } } } } }
{ “representations”: [{ “__typename”: “Human”, “id”: “https://swapi.co/resource/human/11”

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

ontotext

}] }

The Character Similarity GraphQL service will return the ids of similar Characters. Given the Semantic Object service already extends Characters with additional properties (such as name, eyeColour, etc), they are also available at query time. The above example can be extended to include Star Wars Universe object and properties. Perhaps, returning the name, height, eyeColor etc. You can try just such a query here: Web Annotation, Star Wars Universe and Character Similarity GraphQL Services : Retrieve annotation, with similar Characters and Star Wars universe objects and properties

Providing an annotator/curator with similar/related Characters may offer more context and support the process of tag disambiguation. With this in mind, you can now find similar Characters for an annotation using a federated query as follows.

GraphQL Federation over GraphDB, MongoDB and a Semantic Vector Space

The platforms GraphQL federation service aggregates/combines each federated schema/service into a single schema. It does this by requesting the full GraphQL Schema Definition Language (SDL) schema from each service.

For example, here is the full auto-generated Semantic Objects SDL. You will notice that the SDL includes all the federation directives required to join the schemas at the federation gateway. Including @extend, @external, @requires, and the _entites root query.

Star Wars Universe GraphQL Service : Schema Definition Language Query

Loading...

https://swapi-platform.ontotext.com/graphql
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TDA0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOjE4MzMwMzE4OTQsImlhdCI6MTY3NTM1MTg5NCwiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3Rle

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

ontotext

HQuY29tIiwic3ViIjoiZTlmOWQzNGQtZThmMS00ODM1LTlkMzAtOWRjNmU5YjQ4ZmMzIiwianRpIjoiMTZmZDJjOTUtNzNkZi00ZGE4LTkwMDgtMTRhYmQ1NTIyN2RhIiwiYXV0aGVudGljYXRpb25UeXBlIjoiUEFTU1dPUkQiLCJlbWFpbCI6InJlYWRvbmx5dXNlckBleGFtcGxlLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJSZWFkT25seVVzZXIiLCJhcHBsaWNhdGlvbklkIjoiNjcyY2E3YjMtYzM3Mi00ZGYzLTgyZDgtOWExYTBkN2Q2OGMxIiwicm9sZXMiOlsiUmVhZE9ubHkiXSwiZGF0YSI6e319.oGnOYmgdMPtcEK0mrEFOIiNXvrPC4W6m6PBYthP42QQ
true

query SemanticObjectServiceDefinition { _service { sdl } }

Once the aggregated schema is in place, the following GraphQL query can be invoked to retrieve data from the following locations, without the client knowing where the data resides.

1. Web Annotation – MongoDB JSON-LD
2. Similar Characters – GraphDB Semantic Vector Space
3. Character Names – GraphDB RDF

Web Annotation, Star Wars Universe and Character Similarity GraphQL Services : Retrieve annotation, with similar Characters and Star Wars universe objects and properties

Loading...

https://swapi-platform.ontotext.com/federation
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TDA0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOjE4MzMwMzE4OTQsImlhdCI6MTY3NTM1MTg5NCwiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3RleHQuY29tIiwic3ViIjoiZTlmOWQzNGQtZThmMS00ODM1LTlkMzAtOWRjNmU5YjQ4ZmMzIiwianRpIjoiMTZmZDJjOTUtNzNkZi00ZGE4LTkwMDgtMTRhYmQ1NTIyN2RhIiwiYXV0aGVudGljYXRpb25UeXBlIjoiUEFTU1dPUkQiLCJlbWFpbCI6InJlYWRvbmx5dXNlckBleGFtcGxlLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJSZWFkT25seVVzZXIiLCJhcHBsaWNhdGlvbklkIjoiNjcyY2E3YjMtYzM3Mi00ZGYzLTgyZDgtOWExYTBkN2Q2OGMxIiwicm9sZXMiOlsiUmVhZE9ubHkiXSwiZGF0YSI6e319.oGnOYmgdMPtcEK0mrEFOIiNXvrPC4W6m6PBYthP42QQ
true

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

query { annotation(id: “https://www.ontotext.com/blog/the-rise-of-the-knowledge-graph/annotation/1”) { body { source { name similar { name height eyeColor mass hairColor } } purpose } } }

This GraphQL query can be further expanded with additional details from the Star Wars RDF graph such as Character home world, etc..etc..etc..

Web Annotation, Star Wars Universe and Character Similarity GraphQL Services : Retrieve annotation, with similar Characters, Characters Home Planet and Starships

Loading...

https://swapi-platform.ontotext.com/federation
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImJNODctcWV3SldXTWRMRjIzNUtXUmluTlp4TDA0ZngzIn0.eyJhdWQiOiI2NzJjYTdiMy1jMzcyLTRkZjMtODJkOC05YTFhMGQ3ZDY4YzEiLCJleHAiOjE4MzMwMzE4OTQsImlhdCI6MTY3NTM1MTg5NCwiaXNzIjoic3dhcGktcGxhdGZvcm0ub250b3RleHQuY29tIiwic3ViIjoiZTlmOWQzNGQtZThmMS00ODM1LTlkMzAtOWRjNmU5YjQ4ZmMzIiwianRpIjoiMTZmZDJjOTUtNzNkZi00ZGE4LTkwMDgtMTRhYmQ1NTIyN2RhIiwiYXV0aGVudGljYXRpb25UeXBlIjoiUEFTU1dPUkQiLCJlbWFpbCI6InJlYWRvbmx5dXNlckBleGFtcGxlLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJSZWFkT25seVVzZXIiLCJhcHBsaWNhdGlvbklkIjoiNjcyY2E3YjMtYzM3Mi00ZGYzLTgyZDgtOWExYTBkN2Q2OGMxIiwicm9sZXMiOlsiUmVhZE9ubHkiXSwiZGF0YSI6e319.oGnOYmgdMPtcEK0mrEFOIiNXvrPC4W6m6PBYthP42QQ
true

query { annotation(id: “https://www.ontotext.com/blog/the-rise-of-the-knowledge-graph/annotation/1”) { body { source { name similar { name homeworld { name } starship { name } } } purpose } } }

Trade Federation

Ontotext Platform includes GraphQL federation because it provides a lot of flexibility. You are able to achieve a lot:

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

ontotext

- **Avoid monolithic graph architectures** that are hard to maintain and manage. Avoiding small changes that become large challenges due to tight model and service coupling.
- **Query across multiple systems** using a single API, without needing to know the location of the data.
- **Provide a single ubiquitous API language** across bounded contexts with easily identifiable shared boundaries.
- **Flexibly develop, scale-out, deploy and manage** services in relative isolation.
- **Combine the best tools for the job;** using persistence stores that are most applicable, performant and resilient for the data in question

When you are able to make use of GraphQL federation, and bounded context architectures in combination with very large GraphDB databases you can make the right decisions as to when, where, and how you access your data.

Request License

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