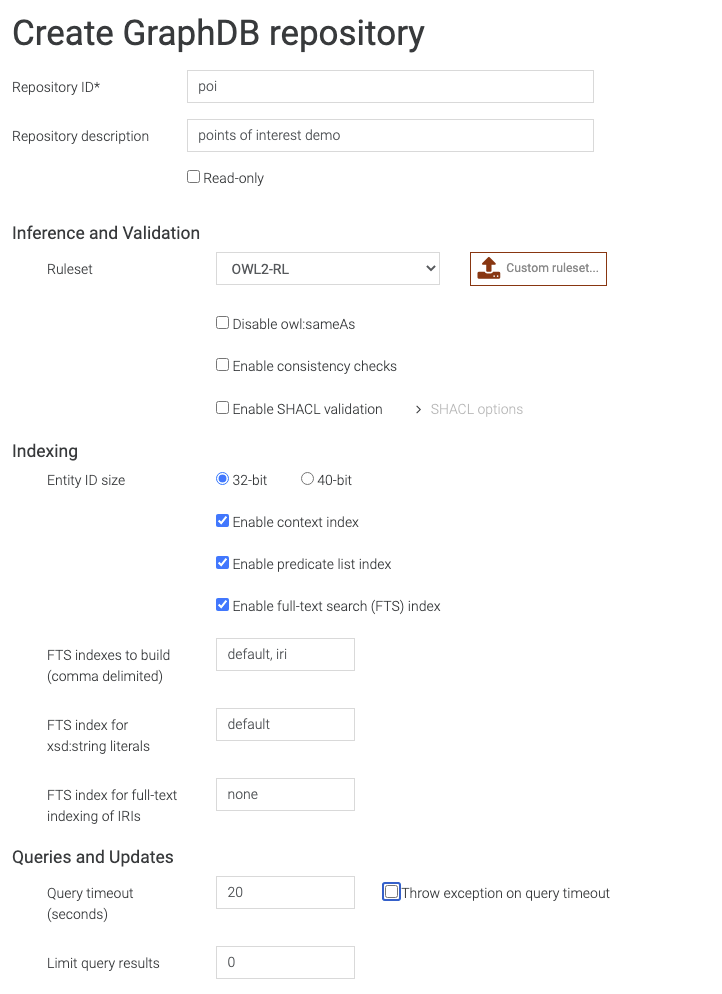
Create a repository.

On the left nav bar select “Setup”

Click “Create a new repository”

Click “GraphDB repository”

You will get a dialog as below. Fill in a repository ID. The only constraint is that it is different from other repositories. Select ruleset OWL2-RL, enable context index, enable predicate list index, enable full text search. For safety I usually set a query timeout (in seconds). Everything else can be left as default. Click create. The new repository will show up in the list of repositories.



Once the repository is created, click the link icon on the row with your repository. The hover will say “Copy repository URL to clipboard” and save the URL for when you want to programmatically query.

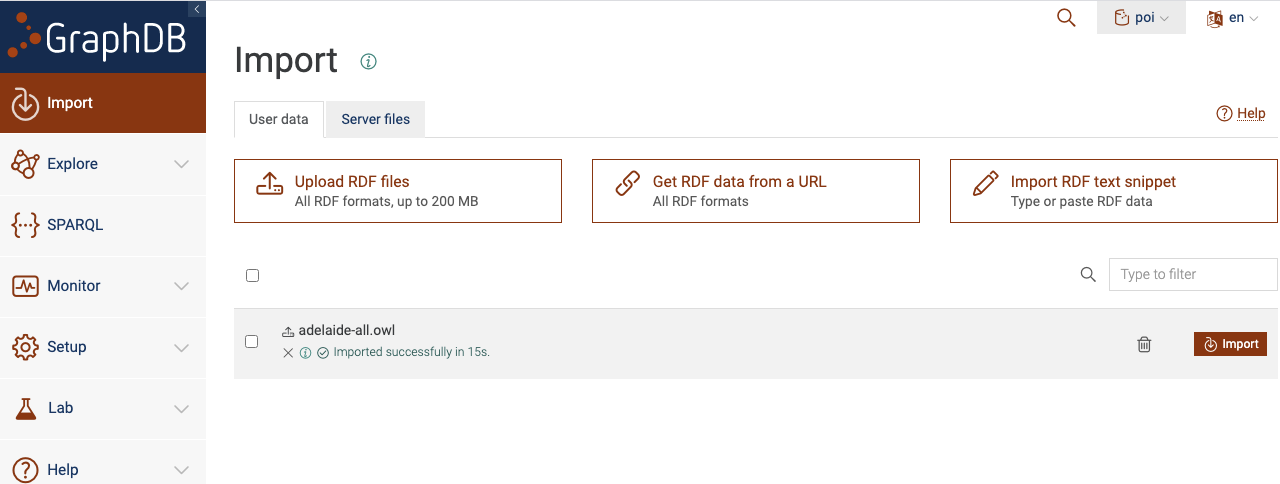
# Import adelaide-all.owl

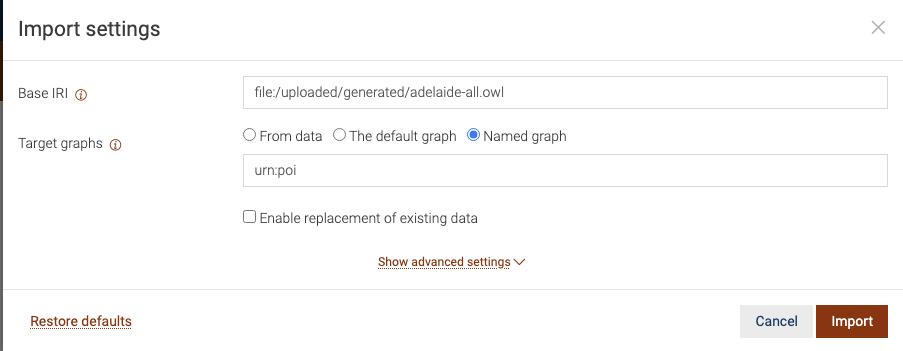
On the left navbar select Import.

Choose “Upload RDF Files”.

In the upper right there is a drop down for repository. If it isn’t already selected, select the repository you created.

You should get a file chooser. Choose adelaide-all.owl

It will take a few seconds and then you should see the below.

Click “Import”. You will get a dialog 

Base IRI is already filled out.

Choose “Named graph”

Enter an IRI. I used urn:poi

Click import and wait until done.

By putting this data in a named graph we can have other data sets in the same repository but just query from this particular set. It’s also easier to clear this out if you want to load another version of the file.

# Query for a list of things that match some string

## Using the interface

On the left navbar select SPARQL. You will do two queries to get the coordinates. The first will be a search for the places that match the string. The second will be to query that thing to get the coordinate. In the upper right, make sure your repository is selected.

Documentation for full text search is here: <https://graphdb.ontotext.com/documentation/10.1/full-text-search.html>

This query will return a list of names and IRIs where name includes “YOUR\_STRING”. Look at the doc for the various forms of query string.

PREFIX onto: <http://www.ontotext.com/>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select ?name ?poi where { graph <urn:poi> {

?poi rdfs:label ?name.

?name onto:fts "YOUR\_STRING".

}} limit 100

Adjust limit to the max number of results you want to put in your menu.

I didn’t see syntax for a leading string search – only those names that start with “YOUR\_STRING” so you can use FILTER to get just those or sort them by whether it’s leading string or internal string.

PREFIX onto: <http://www.ontotext.com/>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select ?name ?thing where { graph <urn:poi> {

?poi rdfs:label ?name.

?name onto:fts ("/YOUR\_STRING.\*/" 100).

FILTER regex(?name,"^YOUR\_STRING.\*", "i")

}} limit 100

Once you choose the IRI of the POI selected you can retrieve the coordinate with

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX measures: <http://www.ontologyrepository.com/CommonCoreOntologies/is\_a\_measurement\_of>

PREFIX lat: <http://www.ontologyrepository.com/CommonCoreOntologies/has\_latitude\_value>

PREFIX lon: <http://www.ontologyrepository.com/CommonCoreOntologies/has\_longitude\_value>

select ?lat ?lon where

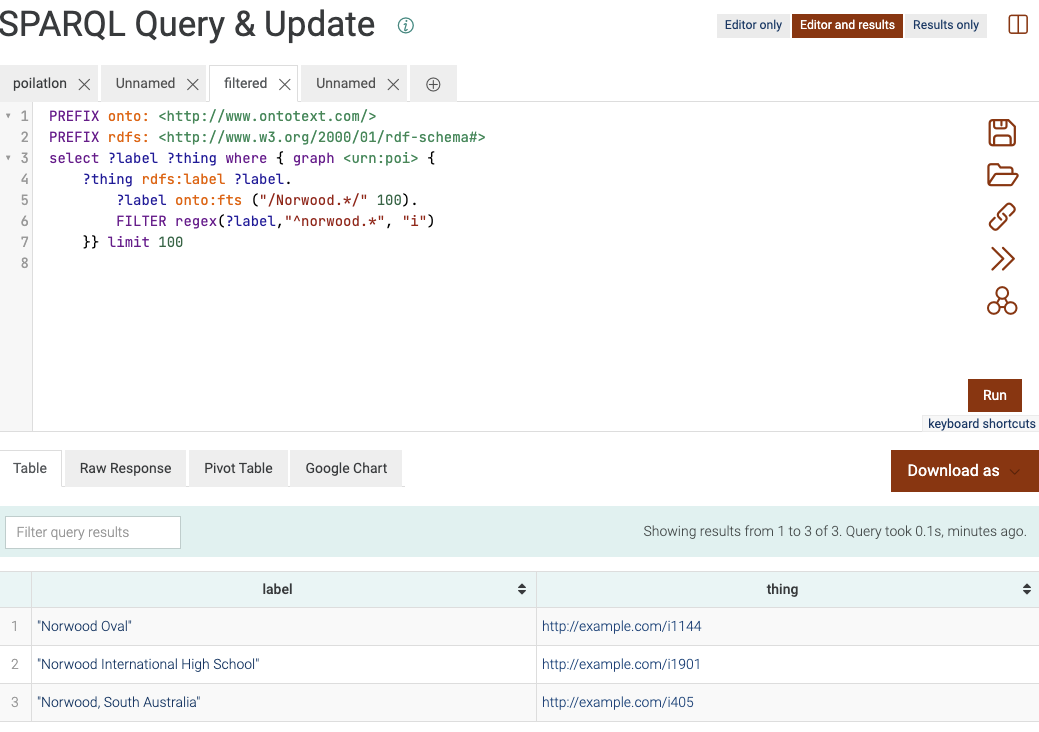
{ graph <urn:poi>

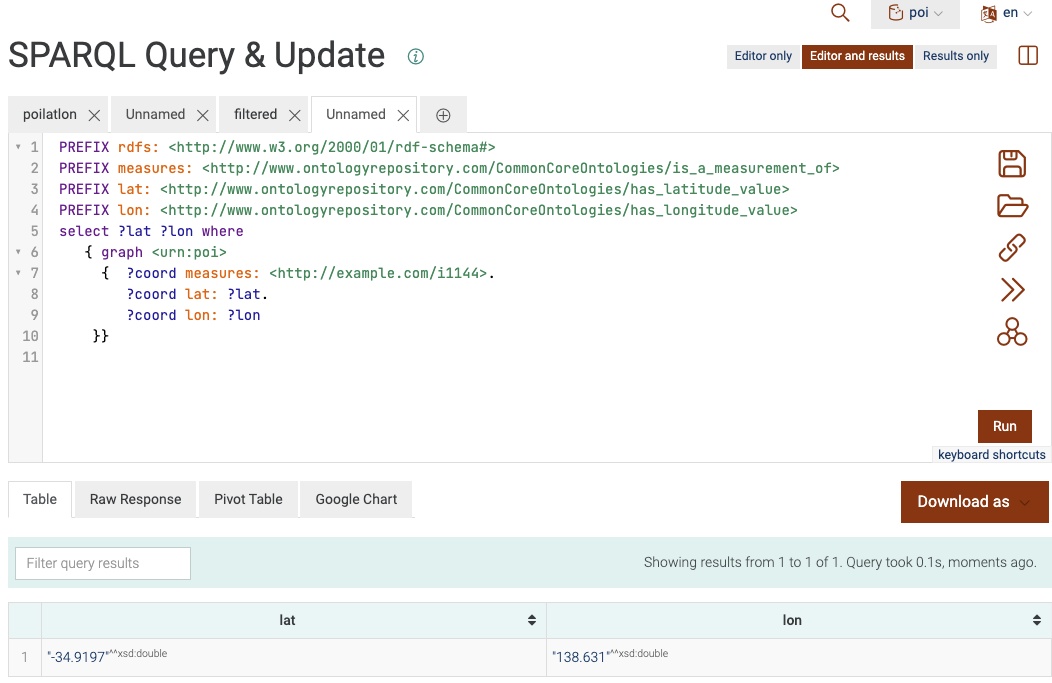
{ ?coord measures: <PLACE\_IRI>.

?coord lat: ?lat.

?coord lon: ?lon

}} limit 100





To do the last pictured query programmatically, do a POST to the repository URL with the post body url encoded property “query”. Here’s an example using curl

curl -X POST -d "query=PREFIX+rdfs%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2000%2F01%2Frdf-schema%23%3E%0APREFIX+measures%3A+%3Chttp%3A%2F%2Fwww.ontologyrepository.com%2FCommonCoreOntologies%2Fis\_a\_measurement\_of%3E%0APREFIX+lat%3A+%3Chttp%3A%2F%2Fwww.ontologyrepository.com%2FCommonCoreOntologies%2Fhas\_latitude\_value%3E%0APREFIX+lon%3A+%3Chttp%3A%2F%2Fwww.ontologyrepository.com%2FCommonCoreOntologies%2Fhas\_longitude\_value%3E%0Aselect+%3Flat+%3Flon+where+%0A+++%7B+graph+%3Curn%3Apoi%3E+%0A+++++%7B++%3Fcoord+measures%3A+%3Chttp%3A%2F%2Fexample.com%2Fi1144%3E.%0A++++++++%3Fcoord+lat%3A+%3Flat.%0A++++++++%3Fcoord+lon%3A+%3Flon%0A++++%7D%7D++" -H "Accept: **text/csv**" <http://Alans.local:7200/repositories/poi>

returns

lat,lon

-3.49197E1,1.38631E2

Mind that the CSV format will return scientific notation since lat and lon are doubles.

If you substitute application/sparql-results+json instead of text/csv you get

{

"head" : {

"vars" : [

"lat",

"lon"

]

},

"results" : {

"bindings" : [

{

"lat" : {

"datatype" : "http://www.w3.org/2001/XMLSchema#double",

"type" : "literal",

"value" : "-34.9197"

},

"lon" : {

"datatype" : "http://www.w3.org/2001/XMLSchema#double",

"type" : "literal",

"value" : "138.631"

}

}

]

}

}