

SIMPLE QUERY #1 – UNION and SHARED SUBJECTS

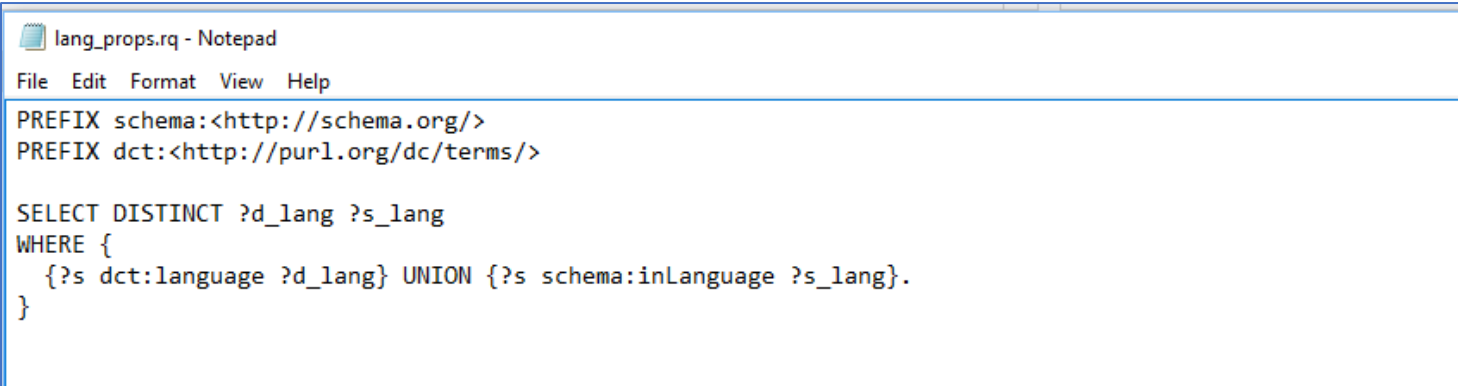
What languages are represented in this dataset?

To write this query, you need to determine one vital piece of information – what property is used when describing a resource’s language? This is a good time to refer back to the table of properties you saved to a CSV file (**SEE** Exploratory Queries section). If you skim through it, you see that there are two likely candidates: “<http://purl.org/dc/terms/language>” and “<http://schema.org/inLanguage>”.

To determine which property you should use in future queries to get the best results, you can first write a fairly simple query which will give you an idea how the dataset’s creators used these properties. After all, *aren’t you already wondering why they felt the need to use two separate properties, instead of choosing one?*

The following query allows you to compare the results of both “<http://purl.org/dc/terms/language>” and “<http://schema.org/inLanguage>” in one result set. You could have written two separate queries, but this approach saves time and gives you a chance to begin using the **UNION** keyword.

This keyword tells the query that you want the results from both specified triple patterns. You are asking for all the values from two different properties, separately, but asking for all the results to be returned at one time. Notice the syntax: you use brackets around each triple statement, put the UNION keyword between them, and nest the entire line within the **WHERE** clause (**Figure 8**).



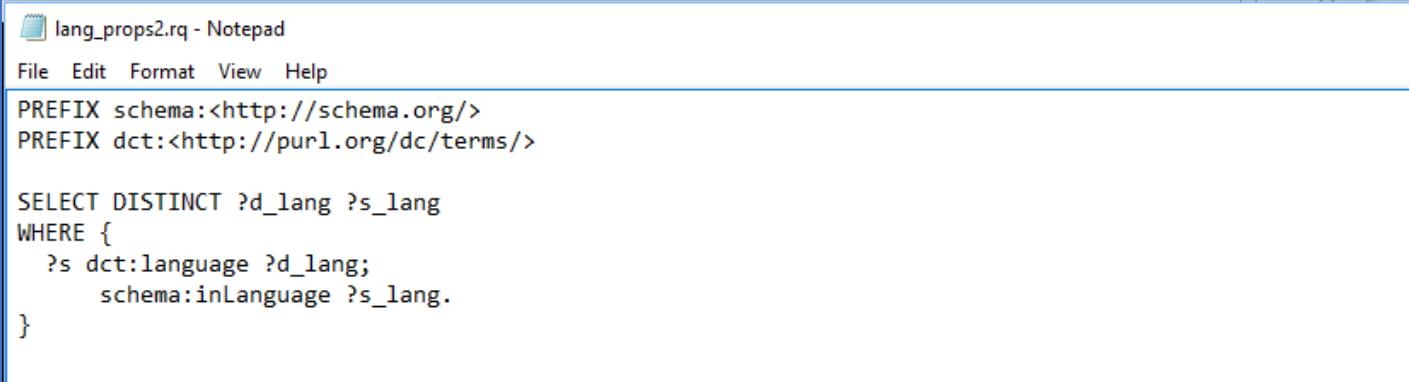
```
lang_props.rq - Notepad
File Edit Format View Help
PREFIX schema:<http://schema.org/>
PREFIX dct:<http://purl.org/dc/terms/>

SELECT DISTINCT ?d_lang ?s_lang
WHERE {
  {?s dct:language ?d_lang} UNION {?s schema:inLanguage ?s_lang}.
}
```

Figure 8: SPARQL query using UNION to check for either property

Another thing to note about this query is that it uses the keyword **PREFIX** so that you do not need to keep retyping the *namespaces* for terms from Schema.org or Dublin Core, respectively. Technically, you can use whatever abbreviations you want when you declare prefixes, but it is probably a good idea to start getting to know the more commonly used ones – mainly because it will come in handy in the future when using SPARQL endpoints (some, such as *DBpedia's SNORQL*, come with pre-declared namespaces that use standard abbreviations).

The query above carries a very different meaning than if you had written it in a subtly different way (**Figure 9**):



```
lang_props2.rq - Notepad
File Edit Format View Help
PREFIX schema:<http://schema.org/>
PREFIX dct:<http://purl.org/dc/terms/>

SELECT DISTINCT ?d_lang ?s_lang
WHERE {
  ?s dct:language ?d_lang;
     schema:inLanguage ?s_lang.
}
```

Figure 9: SPARQL query with two triple statements sharing same subject

This query is similar to the previous one in that it is asking about the same two properties. However, in this case, the triple statements are tied together because they share the *same subject*. Again, notice the syntax: rather than ending each statement with a full stop, (i.e., “.”), you end it with a semi-colon (“;”), until you reach the final statement and again use the full stop. This is a commonly used SPARQL short hand that allows you to declare the subject of the triple (“?s”) only once, in the first of the statements.

However, if you share the same subject in both triples, this means that the query *will only consider and return values* from resources that have been described in the dataset using *both the properties*. The results you get may or may not be similar to the previous query – it depends on how the creators of the dataset used these properties. If they consistently used both properties to describe most of the resources, then the

results would be very similar. However, if they alternated between one and the other, then phrasing the query this second way *may omit many relevant results*.

When you look at the result set of the **UNION** query (**Figure 10**), you see the results separated into the two variables you asked for, one for the Dublin Core property and one for the Schema property.

d_lang	s_lang
"en"	
"fr"	
"it"	
"itaspa"	
"engspa"	
"engfrespa"	

[SCROLL DOWN HERE]

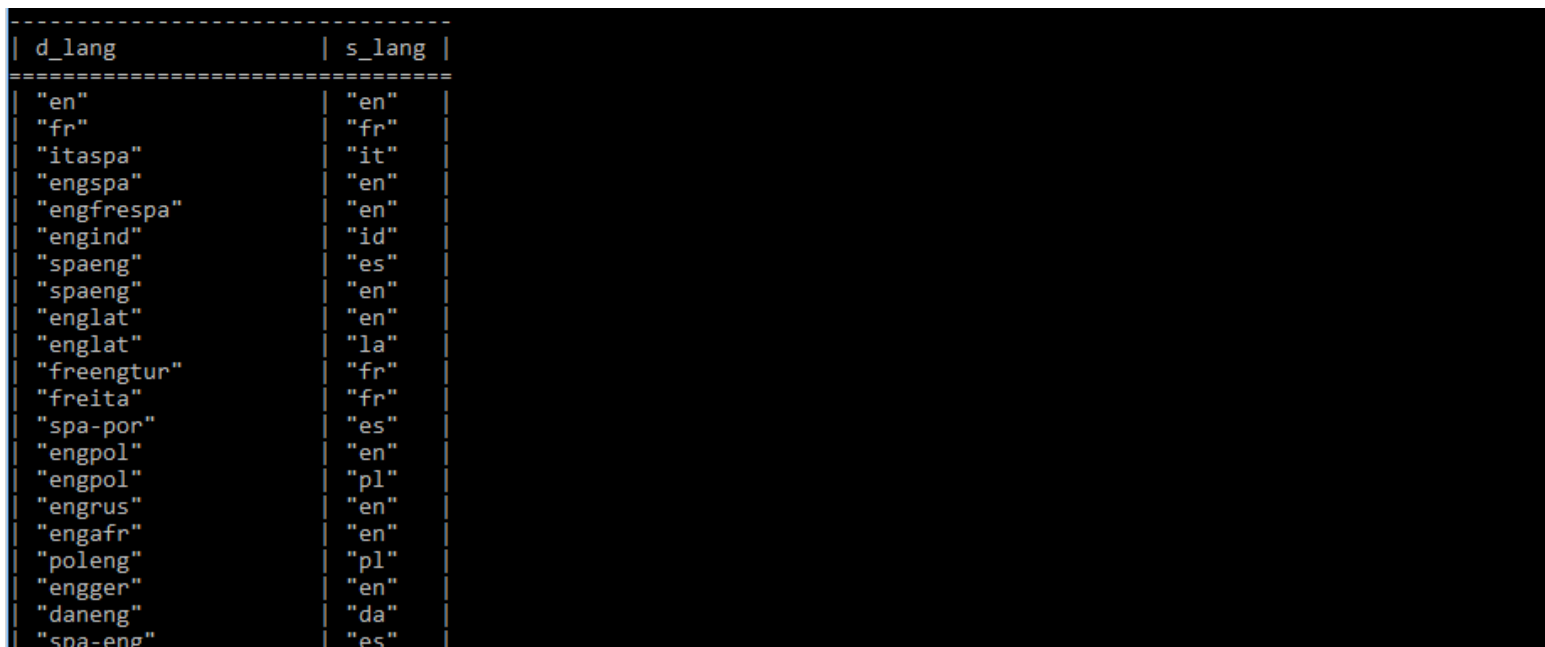
"koreng"		
"porspa"		
"polgereng"		
"polengger"		
"engmul"		
"hunger"		
"gerrus"		
"engwel"		
	"ca"	
	"cr"	
	"cy"	
	"fi"	
	"gu"	
	"is"	
	"lv"	
	"si"	
	"uz"	
	"en"	
	"fr"	
	"it"	
	"ru"	

Figure 10: Result set for UNION of two triple statements and respective variables

At first, the Schema property appears empty but, if you continue to scroll down, you will find the place where the Dublin Core properties stop and the Schema properties begin. This is because the **UNION** query looked first for resources that had a Dublin Core property and displayed the unique values for them first. Then, it displayed the unique values for all the resources it found with Schema.org properties.

Looking at these results, it appears that Dublin Core was mainly used to describe resources which use or are available in multiple languages. These multiple languages are not listed separately but run together into a single string, such as "itaspa", which you might deduce means "Italian and Spanish". On the other hand, the Schema property appears to have been used to describe single languages (maybe the resource's primary or original language) using the standard two and three letter *language codes* from the **International Standards Organization (ISO)**. So, the creators of the dataset had a definite purpose in using the two different properties.

For comparison's sake, look at the result set (**Figure 11**) when you write the query using a shared subject variable. In this case, you see values for both the Dublin Core and Schema.org property variables side-by-side, because each resource in this result set has a value for each variable. As specified by the query, those that had only one or the other were not included.



d_lang	s_lang
"en"	"en"
"fr"	"fr"
"itaspa"	"it"
"engspa"	"en"
"engfrespa"	"en"
"engind"	"id"
"spaeng"	"es"
"spaeng"	"en"
"englat"	"en"
"englat"	"la"
"freengtur"	"fr"
"freita"	"fr"
"spa-por"	"es"
"engpol"	"en"
"engpol"	"pl"
"engrus"	"en"
"engafr"	"en"
"poleng"	"pl"
"engger"	"en"
"daneng"	"da"
"spa-eng"	"es"

Figure 11: Result set for triple statements sharing subject variable