SPARQL manual for BGW

1. Presentation

The aim of this document is to provide a brief guide to introduce users to SPARQL queries and facilitate the use of the BioGateway (BGW). The World Wide Web Consortium (W3C) has detailed tutorials that allow to go deeper into this standardised query language for querying RDF graphs, so here we will not cover all aspects of SPARQL and all its potential, but the most basic aspects that allow potential users to exploit the BGW model, accessible through its endpoint (http://ssb4.nt.ntnu.no:23122/sparql). Some of these more detailed and advanced tutorials are accessible on the main SPARQL Working Group (https://www.w3.org/2009/sparql/wiki/Main_Page). Many other tutorials are easily accessible through other platforms, since this is the standardised language for querying RDF knowledge graphs (RDF triple stores).

2. Introduction

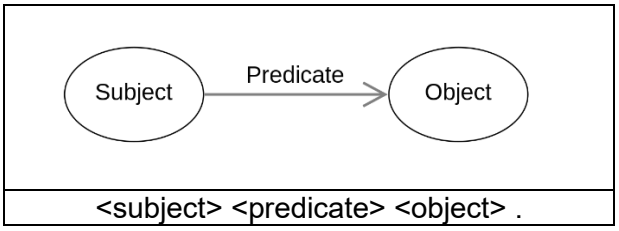
SPARQL is a query language to query and manipulate RDF graphs. It was established as a W3C recommendation in 2013, although it was introduced in 2008 and updated in 2013 as SPARQL 1.1. It is based on SQL features and does not depend on the syntax of the XML/RDF document, i.e. queries do not change if the document syntax differs.

If the data sources are interoperable and accessible through an endpoint, SPARQL allows querying multiple data sources, whether the data is RDF or accessible in the form of RDF through middleware. In addition, the output of a query can be either result sets (table) or RDF graphs.

3. Basic syntax of RDF triples queried through SPARQL

RDF syntax is based on the use of triples in which two nodes (subject and object) are related through a property (predicate).

There are different formats for serialising RDF. In BGW we use a simple syntax based on N-Triples (https://www.w3.org/TR/n-triples/), where each line corresponds to a triplet with a format: <subject> <predicate> <object> .



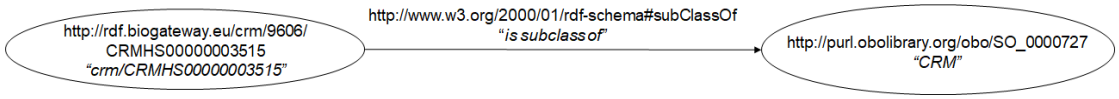
The end of each line is indicated by a dot and each entity (subject, predicate, object) is represented by an Internationalized Resource Identifier (IRI) (https://www.w3.org/TR/rdf11-concepts/). These IRIs are written in angle brackets (<IRI>). If the entity is not a resource but a literal, i.e. a string or a number, this is written

between double quotes, which can be followed by the corresponding datatype (example: "114025907"^^<http://www.w3.org/2001/XMLSchema#integer>). Example here.

In order to avoid duplication and to promote interoperability between domains, as well as other FAIR principles (10.1038/sdata.2016.18), in BGW we avoid the generation of new IRIs as much as possible and reuse existing resources in ontologies and high-level vocabularies widely used by the community. In addition, these resources provide semantics to the model.

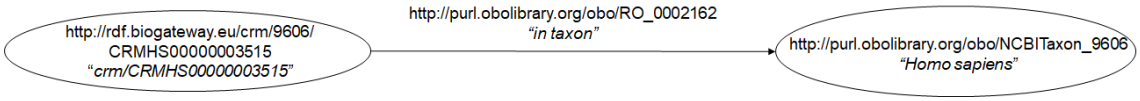
Examples:

- The enhancer with identifier CRMHS00000003515 is a subclass of cis-regulatory module (CRM):



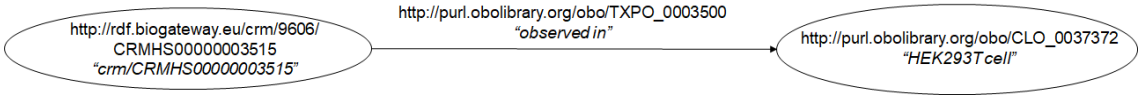
<http://rdf.biogateway.eu/crm/9606/CRMHS00000003515>
<http://www.w3.org/2000/01/rdf-schema#subClassOf>
<http://purl.obolibrary.org/obo/SO_0000727> .

- This same enhancer is characteristic of *Homo sapiens*:



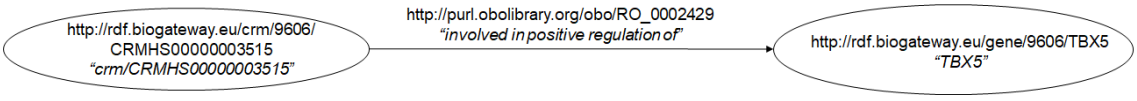
<http://rdf.biogateway.eu/crm/9606/CRMHS00000003515>
<http://purl.obolibrary.org/obo/RO_0002162>
<http://purl.obolibrary.org/obo/NCBITaxon_9606> .

- This same enhancer has been identified in the HEK293T cell line:



<http://rdf.biogateway.eu/crm/9606/CRMHS00000003515>
<http://purl.obolibrary.org/obo/TXPO_0003500>
<http://purl.obolibrary.org/obo/CLO_0037372> .

- This same enhancer regulates the target gene TBX5:



<http://rdf.biogateway.eu/crm/9606/CRMHS00000003515>
<http://purl.obolibrary.org/obo/RO_0002429> <http://rdf.biogateway.eu/gene/9606/TBX5> .

4. IRIs and variables in SPARQL

As we will discuss below, SPARQL queries are also performed following this triple format, where we can use these IRIs, but also variables. In addition, to make IRIs more readable and reduce typing, IRIs can be represented in SPARQL in a qnames format (https://www.w3.org/2001/tag/doc/qnameids.html), i.e. a prefix:ID format.

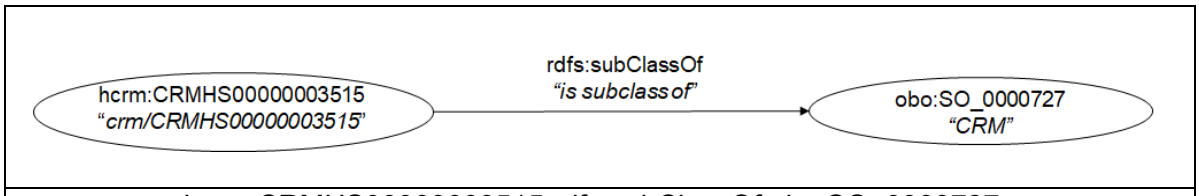
4.1. Use of qnames

Following the examples above, we can define prefixes and use them to abbreviate entities. To define prefixes SPARQL uses the PREFIX clause.

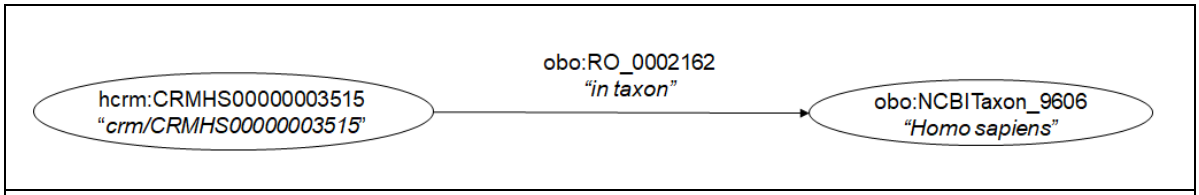
For example, if we define the following prefixes:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX hgene: <http://rdf.biogateway.eu/crm/9606/>
```

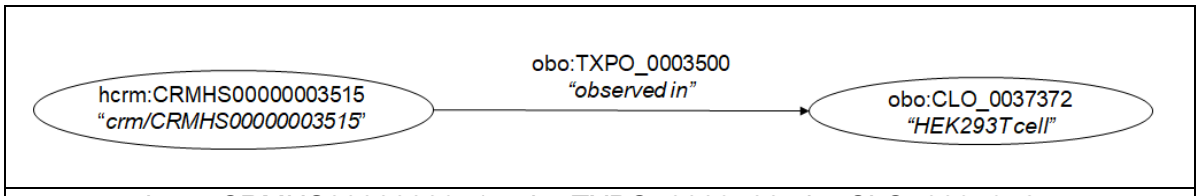
We can represent the same previous triples as:



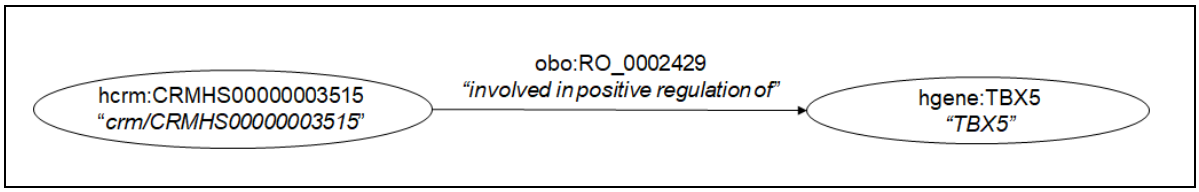
hcrm:CRMHS00000003515 rdfs:subClassOf obo:SO_0000727 .



hcrm:CRMHS00000003515 obo:RO_0002162 obo:NCBITaxon_9606 .



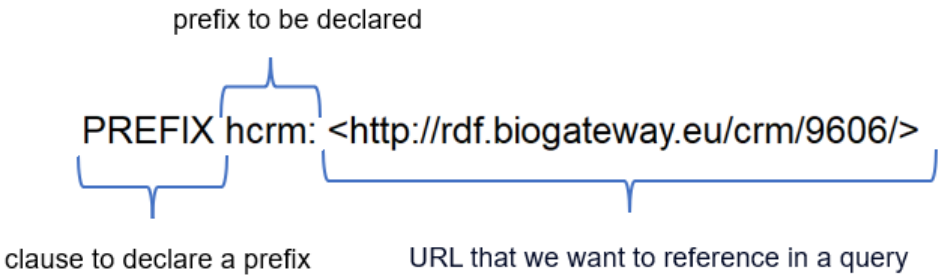
hcrm:CRMHS00000003515 obo:TXPO_0003500 obo:CLO_0037372 .



hcrm:CRMHS00000003515 obo:RO_0002429 hgene:TBX5 .

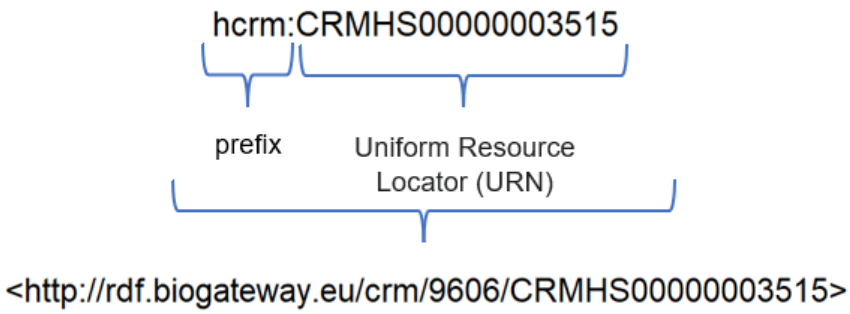
As we can see, prefixes are defined using the PREFIX clause followed by the prefix that we want to define, then by ":" and the URL of the IRI in angle brackets. This URL corresponds to the location of the uniform resource.

For example, in the first definition above:



Subsequently, we can use these prefixes to abbreviate the IRIs, and we will only have to complete them with the name of the Uniform Resource Name (URN).

Example:



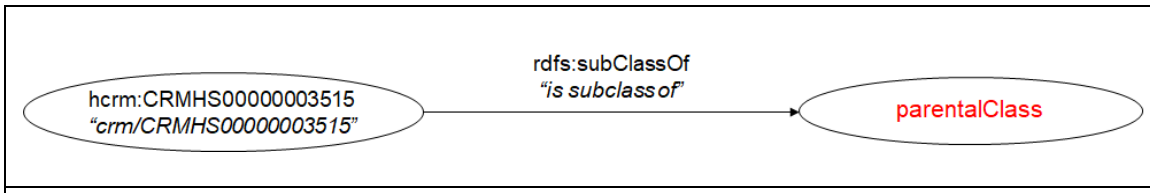
4.2. Use of variables

SPARQL triple patterns can include variables. Their use establishes a link between each variable and each data triple that fits the indicated pattern. Since these are variables, they can have the name we want, being preferable to use representative names. To include a variable in a SPARQL triple, we only need to start the variable name with a question mark, i.e. we follow the format ?variable. These variables can be assigned to any entity (subject, predicate, object).

For example, an expression of type ?subject ?predicate ?object will return all the triples in the model because no constraint is indicated.

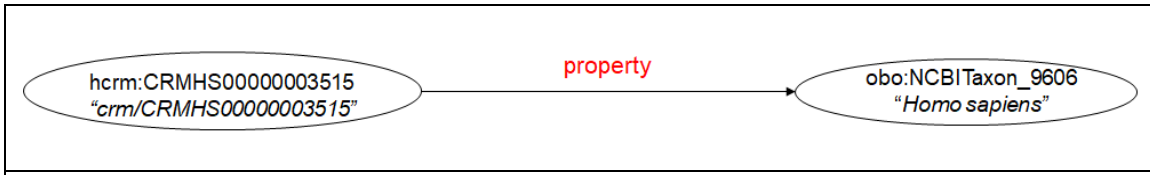
Reusing the examples above:

- An expression of the type:
  hcrm:CRMHS00000003515 rdfs:subClassOf ?parentalClass
  could be used to obtain *which class the CRMHS00000003515 enhancer belongs to*. Therefore, the variable ?parentalClass would take the value <http://purl.obolibrary.org/obo/SO_0000727>, since it is the value linked to the indicated pattern.



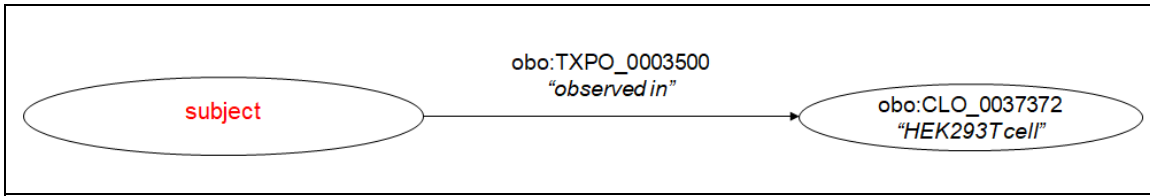
hcrm:CRMHS00000003515 rdfs:subClassOf ?parentalClass

- An expression of the type:
  hcrm:CRMHS00000003515 ?property obo:NCBITaxon_9606
  could be used to identify *which property binds both nodes*.



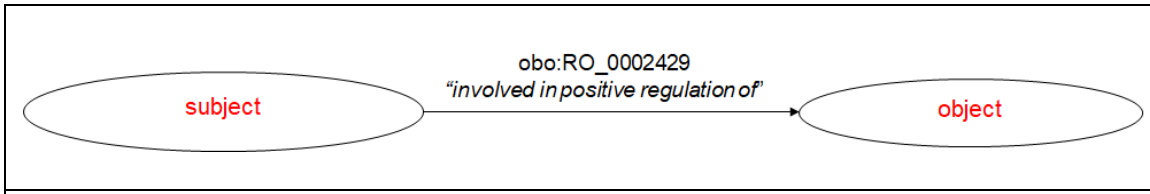
hcrm:CRMHS00000003515 ?property obo:NCBITaxon_9606

- An expression of the type:
  ?subject obo:TXPO_0003500 obo:CLO_0037372
  could be used to identify *which entities have been identified in the HEK293T cell line.*



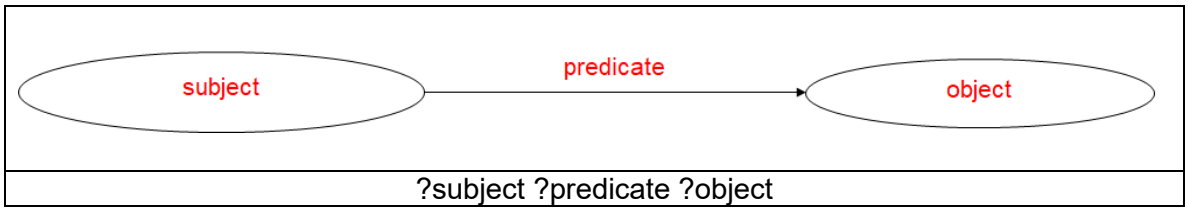
?subject obo:TXPO_0003500 obo:CLO_0037372

- An expression of the type:
  ?subject obo:RO_0002429 ?object
  could be used to obtain the *nodes that are related through the property "involved in positive regulation of"*.



?subject obo:RO_0002429 ?object

- Consequently, an expression of the type:
  ?subject ?predicate ?object
  would return *all existing triplets.*



?subject ?predicate ?object

5. Basic structure of a SPARQL query

The most basic SPARQL queries can be performed through SELECT and WHERE clauses following the structure:

```
SELECT
WHERE { }
```

5.1. SELECT clause

SELECT initiates the query and is equivalent to SELECT in SQL. It is used to indicate the data items to be returned in the query. If we want to return all items in the query, we can use *:

```
SELECT *
WHERE {?s ?p ?o}
```

Other functions can be included in SELECT, for example, to obtain only distinct results (DISTINCT) or to count results (COUNT) (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/).

```
SELECT DISTINCT *
WHERE {?s ?p ?o}
```

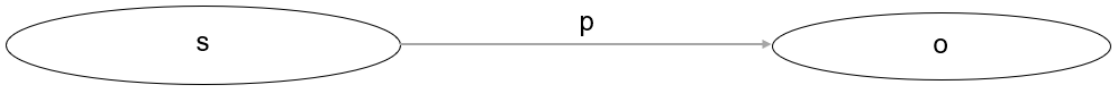
```
SELECT COUNT(*)
WHERE {?s ?p ?o}
```

More illustrative examples are given later.

5.2. WHERE clause

The WHERE clause is used to specify the pattern of triples against which the queried data will be checked, i.e. it is used to indicate the triples that we want to query.

The simplest query is to return all existing triples: ?s ?p ?o



Therefore:

```
SELECT *
WHERE {?s ?p ?o}
```

5.3. Other clauses

Other clauses can be used in queries (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/), for example:

```
BASE
PREFIX
SELECT
FROM
WHERE { }
```

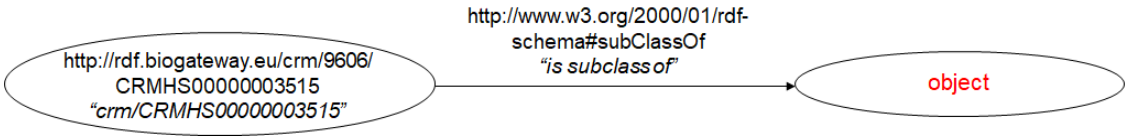
BASE is a variant to simplify the use of IRIs. PREFIX is used to declare prefixes to abbreviate IRIs. FROM identifies the data against which the query is performed (if it does not appear, the active network is queried).

5.4. Examples

To consolidate the concepts, specific examples are given below. To illustrate these examples, we will perform the queries mentioned above in section 4.2. For this, we first access to the BGW endpoint (http://ssb4.nt.ntnu.no:23122/sparql) where we can perform queries against the knowledge network. It is also possible to access this endpoint through another server. This will be discussed later in the section of federated queries (https://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/).

- *Of which entity is the CRMHS00000003515 enhancer a subclass?*
  First, we indicate in SELECT the data we want to obtain, that is, the variable that will take the value of the entity for which the enhancer is a subclass. Secondly, we indicate in WHERE the pattern of the triplet we want to query:

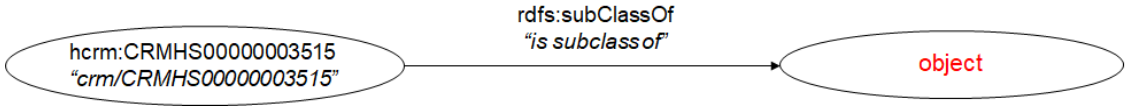
```
SELECT ?object
WHERE {
<http://rdf.biogateway.eu/crm/9606/CRMHS00000003515>
<http://www.w3.org/2000/01/rdf-schema#subClassOf> ?object .
}
```



object
http://purl.obolibrary.org/obo/SO_0000727

As we discussed earlier, we can include prefixes to abbreviate the IRIs in the queries. Therefore, this query is equivalent to:

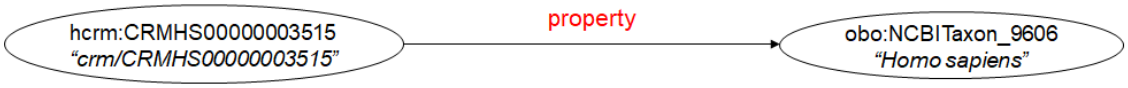
```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?object
WHERE {
  hcrm:CRMHS00000003515 rdfs:subClassOf ?object .
}
```



object
http://purl.obolibrary.org/obo/SO_0000727

- *What property relates the CRMHS00000003515 enhancer to its organism?*

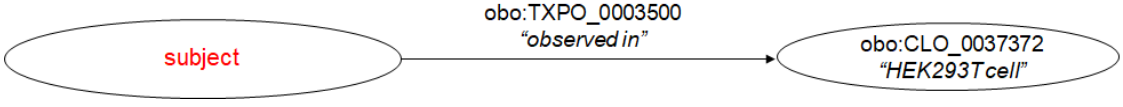
```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?property
WHERE {
  hcrm:CRMHS00000003515 ?property obo:NCBITaxon_9606
}
```



property
http://purl.obolibrary.org/obo/RO_0002162

- *What entities have been identified in the HEK293T cell line?*

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?subject
WHERE {
  ?subject obo:TXPO_0003500 obo:CLO_0037372
}
```



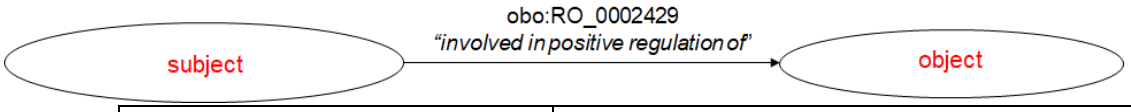
subject
http://rdf.biogateway.eu/crm/9606/CRMHS00000003515#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000003517#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000003518#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000005331#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000005332#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000005461#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000005493#endb

http://rdf.biogateway.eu/crm/9606/CRMHS00000005528#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000005551#endb
http://rdf.biogateway.eu/crm/9606/CRMHS00000005552#endb

(only the first 10 items were included)

- *Which nodes are related through the property "involved in positive regulation of"?*

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?subject ?object
WHERE {
  ?subject obo:RO_0002429 ?object
}
```



subject	object
http://rdf.biogateway.eu/crm/9606/CRMHS00028013707	http://rdf.biogateway.eu/gene/9606/FGFR1OP2
http://rdf.biogateway.eu/crm/9606/CRMHS00028013708	http://rdf.biogateway.eu/gene/9606/FGFR1OP2
http://rdf.biogateway.eu/crm/9606/CRMHS00028013710	http://rdf.biogateway.eu/gene/9606/FGFR1OP2
http://rdf.biogateway.eu/crm/9606/CRMHS00028013715	http://rdf.biogateway.eu/gene/9606/FGFR1OP2
http://rdf.biogateway.eu/crm/9606/CRMHS00028013717	http://rdf.biogateway.eu/gene/9606/FGFR1OP2
http://rdf.biogateway.eu/crm/9606/CRMHS00000005776	http://rdf.biogateway.eu/gene/9606/FGFR2
http://rdf.biogateway.eu/crm/9606/CRMHS00000005777	http://rdf.biogateway.eu/gene/9606/FGFR2
http://rdf.biogateway.eu/crm/9606/CRMHS00000005365	http://rdf.biogateway.eu/gene/9606/FGFR2
http://rdf.biogateway.eu/crm/9606/CRMHS00005046058	http://rdf.biogateway.eu/gene/9606/FGFR2
http://rdf.biogateway.eu/crm/9606/CRMHS00005046059	http://rdf.biogateway.eu/gene/9606/FGFR2

(only the first 10 items were included)

6. Structure of the results

The result of a standard SPARQL query is a table in which each row is a result, and each column corresponds to each of the variables in the SELECT clause. In addition, the name of these columns will be the name used for the variables, but they can also be renamed (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/).

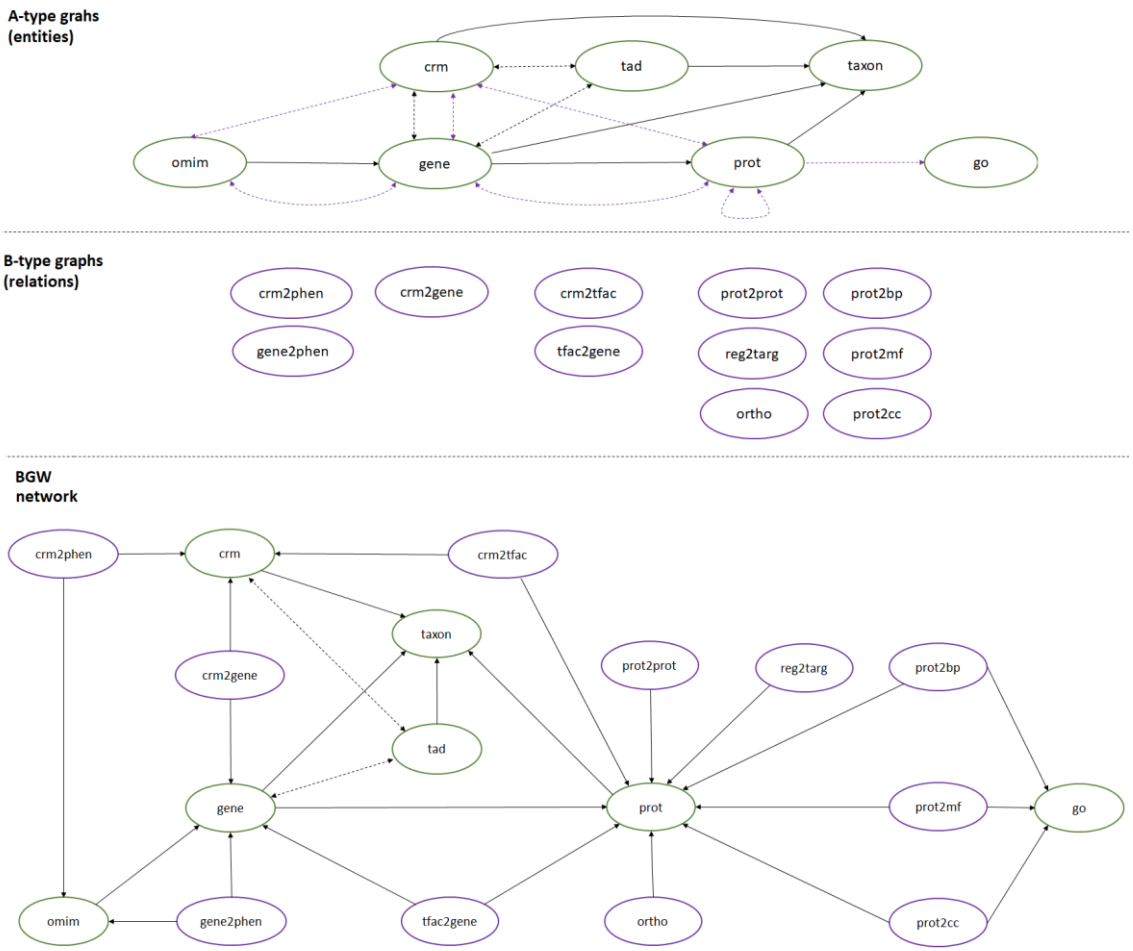
To illustrate a table of results, we use as example the last query performed (*Which nodes are related through the property "involved in positive regulation of"?*) and we show the first 10 results:

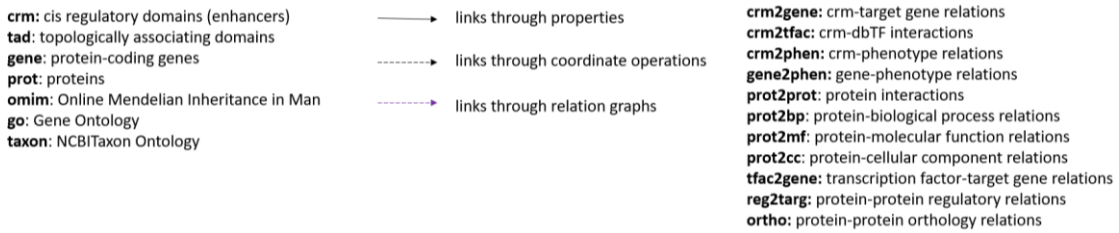
subject	object
http://uniprot.org/uniprot/Q92988	http://rdf.biogateway.eu/gene/9606/TBK1
http://uniprot.org/uniprot/P08047	http://rdf.biogateway.eu/gene/9606/TBK1

http://uniprot.org/uniprot/Q14653	http://rdf.biogateway.eu/gene/9606/TBK1
http://uniprot.org/uniprot/O75376	http://rdf.biogateway.eu/gene/9606/TBL1X
http://uniprot.org/uniprot/Q9Y618	http://rdf.biogateway.eu/gene/9606/TBL1X
http://uniprot.org/uniprot/O75376	http://rdf.biogateway.eu/gene/9606/TBL1XR1
http://uniprot.org/uniprot/Q9Y618	http://rdf.biogateway.eu/gene/9606/TBL1XR1
http://uniprot.org/uniprot/P10275	http://rdf.biogateway.eu/gene/9606/TBL1XR1
http://uniprot.org/uniprot/O96006	http://rdf.biogateway.eu/gene/9606/TBP
http://uniprot.org/uniprot/Q6SJ96	http://rdf.biogateway.eu/gene/9606/TBP

7. Queries on specific graphs

Since knowledge networks are composed of multiple graphs, a good practice is to perform the queries on the graphs we want to query. Below we include an illustration of the different graphs currently available in BGW, which constitute the different domains covered in the knowledge network. More information is available on the BGW website (https://biogateway.eu/) and in the repository of this new work (https://github.com/juan-mulero/cisreg).





In the query above (*Which nodes are related through the property "involved in positive regulation of"?)* we have obtained the nodes that are related through this property (obo:RO_0002429). This property is used in different graphs to relate different nodes. For example, it is used in the relation graph between transcriptional factors (TFs) and genes (tfac2gene) to indicate that a protein, that is a TF, is involved in the positive regulation of a gene. On the other hand, in the crm2gene graph it is used to indicate cis-regulatory sequences involved in the positive regulation of a target gene. Therefore, if we specify the graphs, the results will differ. A general query will search all the graphs in the network, while a query on a specific graph will search only on that graph.

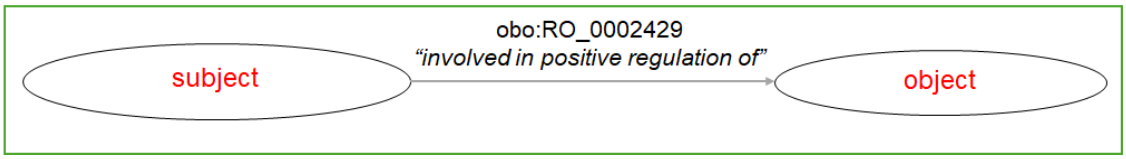
To indicate a graph, simply use the GRAPH clause within the WHERE clause (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#accessingRdfGraphs). For example, if we repeat the above query on these two different graphs, we will obtain different results. In both cases we show only the first 10 results. The limit of results can also be indicated in the query by adding the LIMIT clause at the end of the query (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#modResultLimit).

Which nodes are related through the property "involved in positive regulation of"?

- **Graph tfac2gene**: relation between transcription factors and their target genes.

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?subject ?object
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/tfac2gene> {
    ?subject obo:RO_0002429 ?object
  }
}
LIMIT 10
```

Graph tfac2gene



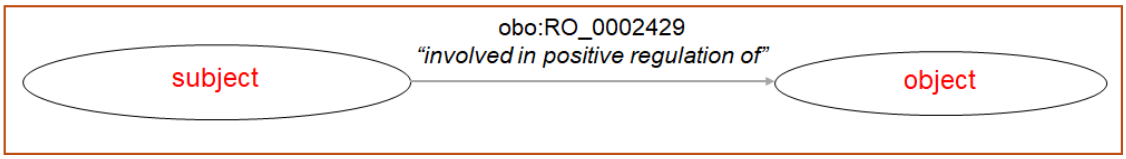
subject	object
http://uniprot.org/uniprot/Q92988	http://rdf.biogateway.eu/gene/9606/TBK1
http://uniprot.org/uniprot/P08047	http://rdf.biogateway.eu/gene/9606/TBK1
http://uniprot.org/uniprot/Q14653	http://rdf.biogateway.eu/gene/9606/TBK1
http://uniprot.org/uniprot/O75376	http://rdf.biogateway.eu/gene/9606/TBL1X
http://uniprot.org/uniprot/Q9Y618	http://rdf.biogateway.eu/gene/9606/TBL1X
http://uniprot.org/uniprot/O75376	http://rdf.biogateway.eu/gene/9606/TBL1XR1
http://uniprot.org/uniprot/Q9Y618	http://rdf.biogateway.eu/gene/9606/TBL1XR1
http://uniprot.org/uniprot/P10275	http://rdf.biogateway.eu/gene/9606/TBL1XR1
http://uniprot.org/uniprot/O96006	http://rdf.biogateway.eu/gene/9606/TBP

http://uniprot.org/uniprot/Q6SJ96	http://rdf.biogateway.eu/gene/9606/TBP

- **Graph crm2gene:** relation between cis-regulatory modules and their target genes.

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?subject ?object
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
    ?subject obo:RO_0002429 ?object
  }
}
ORDER BY ?subject
LIMIT 10
```

Graph crm2gene



subject	object
http://rdf.biogateway.eu/crm/9606/CRMHS00000000009	http://rdf.biogateway.eu/gene/9606/ARHGEF16
http://rdf.biogateway.eu/crm/9606/CRMHS00000000009	https://identifiers.org/hgnc.symbol:MIR4251
http://rdf.biogateway.eu/crm/9606/CRMHS00000000012	http://rdf.biogateway.eu/gene/9606/ARHGEF16
http://rdf.biogateway.eu/crm/9606/CRMHS00000000012	https://identifiers.org/hgnc.symbol:MIR4251
http://rdf.biogateway.eu/crm/9606/CRMHS00000000020	http://rdf.biogateway.eu/gene/9606/ERRFI1
http://rdf.biogateway.eu/crm/9606/CRMHS00000000020	http://rdf.biogateway.eu/gene/9606/SLC45A1
http://rdf.biogateway.eu/crm/9606/CRMHS00000000028	http://rdf.biogateway.eu/gene/9606/SPSB1
http://rdf.biogateway.eu/crm/9606/CRMHS00000000032	http://rdf.biogateway.eu/gene/9606/CASZ1
http://rdf.biogateway.eu/crm/9606/CRMHS00000000033	http://rdf.biogateway.eu/gene/9606/CASZ1
http://rdf.biogateway.eu/crm/9606/CRMHS00000000034	http://rdf.biogateway.eu/gene/9606/CASZ1

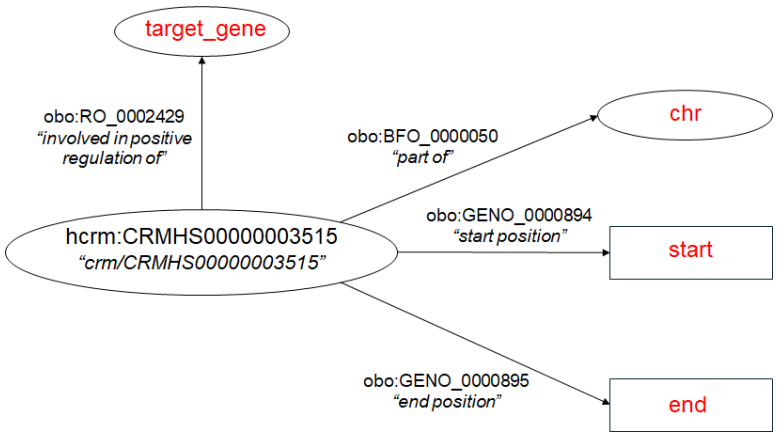
8. Structure of queries with multiple elements

In the examples so far, the queries carried out have been simple because we have only indicated a single triple as query pattern. However, there is no limit in the number of triples that can be used as query pattern. Therefore, we can generate complex patterns that include several triples, but also functions, optional clauses (OPTIONAL) or query union clauses (UNION) among others (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/).

SPARQL has multiple clauses and functions. In this tutorial we cover some of them, but we remind that this document addresses the most basic aspects to initiate beginners, so we will not cover the full potential of SPARQL. Users who want to learn more about the different features of this query language can find more information in the W3C tutorials, among others (https://www.w3.org/2009/sparql/wiki/Main_Page).

In a similar way to N-Triples files, to insert multiple clauses or triples into a query to generate a complex search pattern, we must include a full stop at the end of each triplet. Then we concatenate the new triplet. For example, we have previously run queries against the CRMHS00000003515 enhancer, but with a single triple as query pattern. We can create new queries that require more than one triplet: *What are the coordinates of this enhancer and which target genes does it regulate?*

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene
WHERE {
    hcrm:CRMHS00000003515 obo:BFO_0000050 ?chr .
    hcrm:CRMHS00000003515 obo:GENO_0000894 ?start .
    hcrm:CRMHS00000003515 obo:GENO_0000895 ?end .
    hcrm:CRMHS00000003515 obo:RO_0002429 ?target_gene .
}
```



chr	start	end	target_gene
https://www.ncbi.nlm.nih.gov/nuccore/NC_000012.12	114025907	114026275	http://rdf.biogateway.eu/gene/9606/TBX5

The queries return the results that fit the indicated patterns. Therefore, if we include more triples in the query pattern, more restrictions we are including.

If we want to create a query pattern including different triples, we have to separate each of these triples by a dot. But, if the subject of the node does not change in the next triplet, we can use ";" and omit the subject. As in the previous query the subject is the same, we exemplify this same query by using ";" instead of ";"

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene
WHERE {
    hcrm:CRMHS00000003515 obo:BFO_0000050 ?chr ;
      obo:GENO_0000894 ?start ;
      obo:GENO_0000895 ?end ;
      obo:RO_0002429 ?target_gene .
  }
```

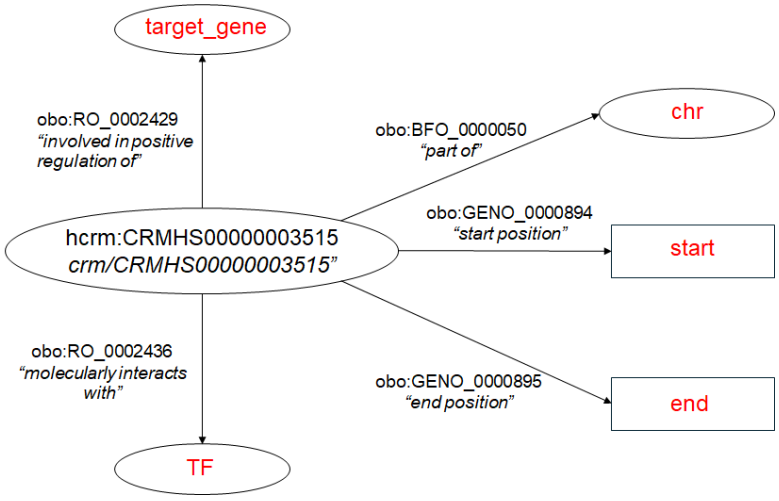
chr	start	end	target_gene
https://www.ncbi.nlm.nih.gov/nuccore/NC_000012.12	114025907	114026275	http://rdf.biogateway.eu/gene/9606/TBX5

9. Optional patterns

If we want to specify that the inclusion of a triple is optional, we must include the OPTIONAL clause before the pattern which appearance will be optional.

For example, we take the CRMHS00000000054 enhancer as reference. If we extend the previous query (*What are the coordinates of this enhancer and which target genes does it regulate?*) to also include *whether it has associated TFs*, we do not get any results because this enhancer has no TF information (*What are the coordinates of this enhancer, which target genes does it regulate and which TFs does it interact with?)*:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene ?TF
WHERE {
    hcrm:CRMHS00000000054 obo:BFO_0000050 ?chr ;
      obo:GENO_0000894 ?start ;
      obo:GENO_0000895 ?end ;
      obo:RO_0002429 ?target_gene ;
      obo:RO_0002436 ?TF
}
```

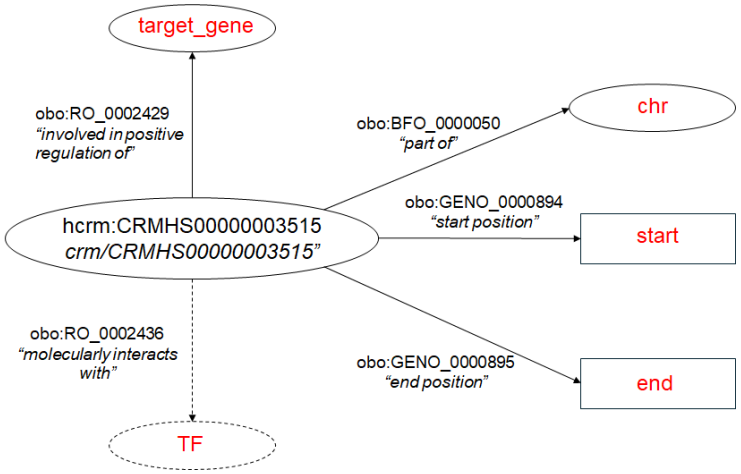


"chr"	"start"	"end"	"target_gene"	"TF"

As the search pattern does not return any results, the output table is empty.

On the contrary, if we indicate the triplet corresponding to the enhancer-TF relation as optional, the query pattern applies only to the 4 previous triplets, being the last one considered as optional. That is to say, if it has such information, it is included, but it is not applied as a pattern that restricts the search. Consequently, the query changes to: *What are the coordinates of the CRMHS00000000054 enhancer and which target genes does it regulate? Include additionally if the enhancer interacts with any TFs.*

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene ?TF
WHERE {
    hcrm:CRMHS00000000054 obo:BFO_0000050 ?chr ;
      obo:GENO_0000894 ?start ;
      obo:GENO_0000895 ?end ;
      obo:RO_0002429 ?target_gene .
    OPTIONAL { hcrm:CRMHS00000000054 obo:RO_0002436 ?TF }
}
```



chr	start	end	target_gene	TF
https://www.ncbi.nlm.nih.gov/nuccore/NC_000001.11	18632177	18633790	http://rdf.biogateway.eu/gene/9606/WNT3A	
https://www.ncbi.nlm.nih.gov/nuccore/NC_000001.11	18632177	18633790	http://rdf.biogateway.eu/gene/9606/CHRM3	
https://www.ncbi.nlm.nih.gov/nuccore/NC_000001.11	18632177	18633790	http://rdf.biogateway.eu/gene/9606/PAX7	

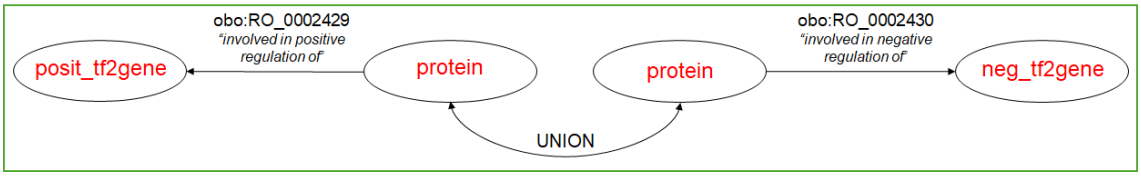
10. Union of sets in SPARQL

SPARQL also allows to combine results from different patterns when there is, at least, one common variable. To do this, we can join queries through the UNION clause.

For example, we can query on the network *to return the target genes of TFs, whether their regulation is positive or negative:*

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?protein ?posit_tf2gene  ?neg_tf2gene
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/tfac2gene> {
    {?protein obo:RO_0002429 ?posit_tf2gene}
    UNION
    {?protein obo:RO_0002430 ?neg_tf2gene}
  }
}
LIMIT 10
```

Graph tfac2gene



protein	posit_tf2gene	neg_tf2gene
http://uniprot.org/uniprot/Q92988	http://rdf.biogateway.eu/gene/9606/TBK1	
http://uniprot.org/uniprot/P08047	http://rdf.biogateway.eu/gene/9606/TBK1	
http://uniprot.org/uniprot/Q14653	http://rdf.biogateway.eu/gene/9606/TBK1	
http://uniprot.org/uniprot/O75376	http://rdf.biogateway.eu/gene/9606/TBL1X	
http://uniprot.org/uniprot/Q9Y618	http://rdf.biogateway.eu/gene/9606/TBL1X	
http://uniprot.org/uniprot/O75362		http://rdf.biogateway.eu/gene/9606/FZR1
http://uniprot.org/uniprot/O75362		http://rdf.biogateway.eu/gene/9606/CDH1
http://uniprot.org/uniprot/Q8N6T7		http://rdf.biogateway.eu/gene/9606/ISG15
http://uniprot.org/uniprot/Q8N6T7		http://rdf.biogateway.eu/gene/9606/TP53
http://uniprot.org/uniprot/A0A0B4J2D5		http://rdf.biogateway.eu/gene/9606/MAP2

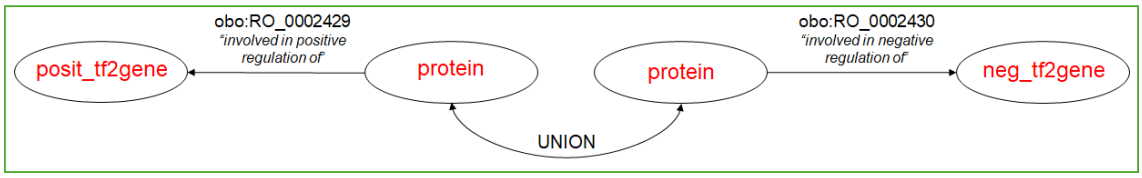
11. SPARQL operations

SPARQL also has multiple functions that allow us to perform operations and also to generate new variables through the BIND clause (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#assignment). Some of these are discussed below:

- COUNT. It counts the number of occurrences of a given expression. For example, following the example above, *how many nodes are positively regulated by TFs and how many are negatively regulated?*

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT  COUNT(?posit_tf2gene)  COUNT(?neg_tf2gene)
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/tfac2gene> {
    {?protein obo:RO_0002429 ?posit_tf2gene}
    UNION
    {?protein obo:RO_0002430 ?neg_tf2gene}
  }
}
```

Graph tfac2gene



callret-0	callret-1
129863	80457

With this query, we count the number of target genes that are positively (property RO_0002429) and negatively (property RO_0002430) regulated by TFs. However, the variable names are not intuitive. As mentioned above, we can rename variables. For this we can use AS:

How many nodes are positively regulated by TFs and how many are negatively regulated?

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT COUNT(?posit_tf2gene) AS ?count_posit_tf2gene
COUNT(?neg_tf2gene) AS ?count_neg_tf2gene
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/tfac2gene> {
    {?protein obo:RO_0002429 ?posit_tf2gene}
    UNION
    {?protein obo:RO_0002430 ?neg_tf2gene}
  }
}
```

count_posit_tf2gene	count_neg_tf2gene
129863	80457

However, it is important to note that the query performed follows the pattern of the protein-property_relation-tgene triplet. Therefore, several TFs could positively regulate the same gene, as well as several TFs could negatively regulate the same gene. In other words, we expect to quantify duplicate nodes. Against this situation, if we want to count how many genes are positively regulated by TFs (unique values) and how many genes are negatively regulated by TFs (unique values), it would be of interest to count the unique values instead of the nodes that match with the search pattern. For this we use the DISTINCT clause, discussed above:

How many different nodes are positively regulated by TFs and how many are negatively regulated?

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT COUNT(DISTINCT ?posit_tf2gene) AS ?count_posit_tf2gene
COUNT(DISTINCT ?neg_tf2gene) AS ?count_neg_tf2gene
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/tfac2gene> {
    {?protein obo:RO_0002429 ?posit_tf2gene}
    UNION
    {?protein obo:RO_0002430 ?neg_tf2gene}
  }
}
```

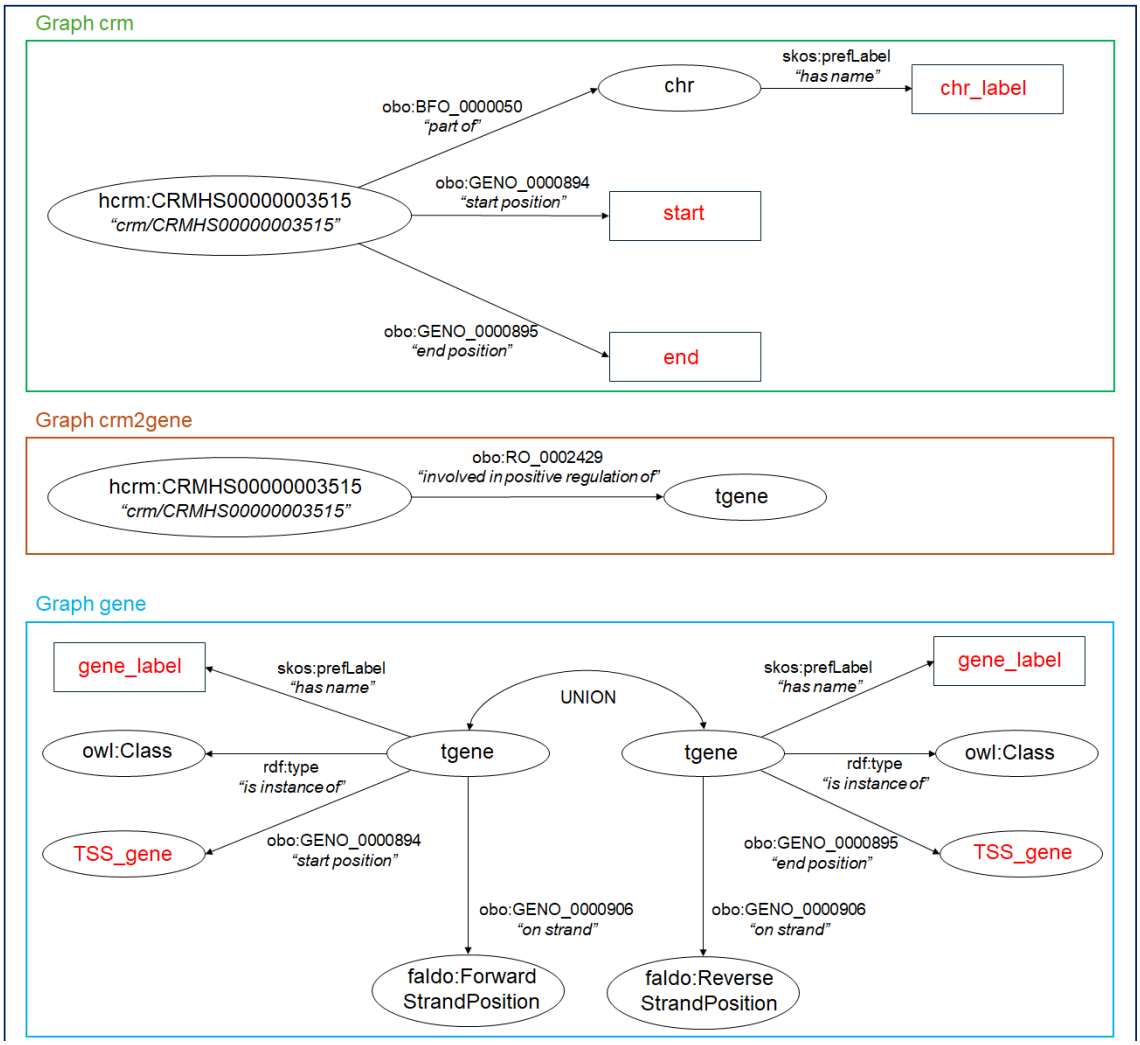
count_posit_tf2gene	count_neg_tf2gene
12713	7963

- Mathematical functions such as Sum, Avg, Min, Max, Abs or Round allow us to perform sum, average, minimum, maximum, absolute value and rounding operations, respectively. For example, we can extend the previous query with the CRMHS00000000054 enhancer (*What are the coordinates of this enhancer and which target genes does it regulate?*) to calculate the distance between the enhancer and its target genes, taking as a reference the middle position of the enhancer and the start of the gene.
  Thus, the query would be: *What are the coordinates of the CRMHS00000000054 enhancer and which genes does it regulate? Also calculate the distance that separates both sequences using as reference the middle position of the enhancer and the start of the gene.*
  We can calculate the middle point of the enhancer using its start and end positions, create a variable with this average position and use it to determine its distance to the start site of the target genes. To do this, if the target gene is on the '+' strand, we use the start position, while we use the end position if the gene is on the '-' strand. To do this we can use the UNION clause:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX faldo: <http://biohackathon.org/resource/faldo#>
SELECT DISTINCT ?chr_label ?start ?end ?crm_middle ?gene_label
?TSS_gene ?distance
WHERE {
      GRAPH <http://rdf.biogateway.eu/graph/crm> {
            hcrm:CRMHS00000000054 obo:BFO_0000050 ?chr ;
                  obo:GENO_0000894 ?start ;
                  obo:GENO_0000895 ?end .
            ?chr skos:prefLabel ?chr_label .
        }
```

```
        GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
                hcrm:CRMHS00000000054 obo:RO_0002429 ?tgene .
        }

        GRAPH <http://rdf.biogateway.eu/graph/gene> {
                ?tgene rdf:type owl:Class ;
                        skos:prefLabel ?gene_label .
                {
                ?tgene obo:GENO_0000906 faldo:ForwardStrandPosition;
                        obo:GENO_0000894 ?TSS_gene .
                } UNION {
                ?tgene obo:GENO_0000906 faldo:ReverseStrandPosition ;
                        obo:GENO_0000895 ?TSS_gene .
                }
        }

        BIND((?end - ?start) AS ?crm_range)
        BIND((ROUND(?crm_range/2) + ?start) AS ?crm_middle)
        BIND(ABS(?TSS_gene - ?crm_middle) AS ?distance)
}
```

BGW knowledge network



```
crm_range = end – start
crm_middle = (crm_range/2) + start    rounded value
distance = | TSS_gene - crm_middle |
```

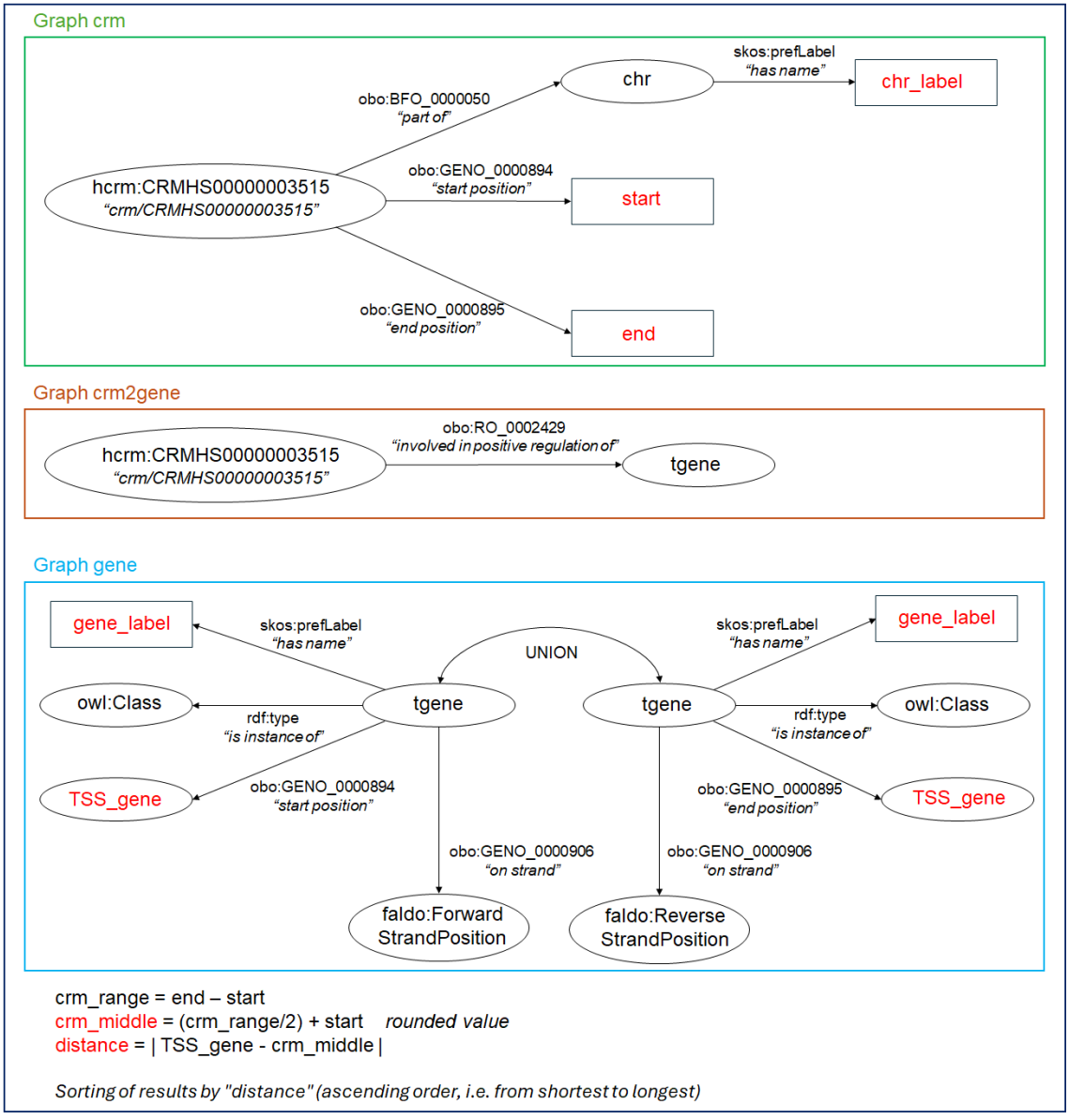
chr_label	start	end	crm_middle	gene_label	TSS_gene	distance
chr-1	18632177	18633790	18632983	CHRM3	239386565	220753582
chr-1	18632177	18633790	18632983	WNT3A	228006998	209374015
chr-1	18632177	18633790	18632983	PAX7	18630846	2137

- ORDER BY. It is a clause that allows to establish an order in the sequence of solutions. Ascending order by adding ASC() to the clause and descending order by adding DESC(). For example, we can order the above result by the variable "distance": *What are the coordinates of the CRMHS00000000054 enhancer and which genes does it regulate? Also calculate the distance that separates both sequences using as reference the middle position of the enhancer and the start of the gene. Order the results using the distance value from smallest to largest, i.e. order the enhancer-target gene relations showing first the closest neighboring sequences (theoretically more probable).*

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX faldo: <http://biohackathon.org/resource/faldo#>
SELECT DISTINCT ?chr_label ?start ?end ?crm_middle ?gene_label
?TSS_gene ?distance
WHERE {
      GRAPH <http://rdf.biogateway.eu/graph/crm> {
            hcrm:CRMHS00000000054 obo:BFO_0000050 ?chr ;
                  obo:GENO_0000894 ?start ;
                  obo:GENO_0000895 ?end .
            ?chr skos:prefLabel ?chr_label .
      }

      GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
            hcrm:CRMHS00000000054 obo:RO_0002429 ?tgene .
      }

      GRAPH <http://rdf.biogateway.eu/graph/gene> {
            ?tgene rdf:type owl:Class ;
                  skos:prefLabel ?gene_label .
            {
            ?tgene obo:GENO_0000906 faldo:ForwardStrandPosition;
                  obo:GENO_0000894 ?TSS_gene .
            } UNION {
            ?tgene obo:GENO_0000906 faldo:ReverseStrandPosition ;
                  obo:GENO_0000895 ?TSS_gene .
            }
      }
```

```
        BIND((?end - ?start) AS ?crm_range)
        BIND((ROUND(?crm_range/2) + ?start) AS ? crm_middle)
        BIND(ABS(?TSS_gene - ? crm_middle) AS ?distance)
}
ORDER BY ASC(?distance)
```

BGW knowledge network



crm_range = end – start
crm_middle = (crm_range/2) + start *rounded value*
distance = | TSS_gene - crm_middle |

Sorting of results by "distance" (ascending order, i.e. from shortest to longest)

chr_label	start	end	crm_middle	gene_label	TSS_gene	Distance
chr-1	18632177	18633790	18632983	PAX7	18630846	2137
chr-1	18632177	18633790	18632983	WNT3A	228006998	209374015
chr-1	18632177	18633790	18632983	CHRM3	239386565	220753582

12. Multiple values

Until now we have carried out queries using specific values and variables. However, when we have indicated values to a node or property, we have indicated a single value. On the other hand, we have used variables to select triples that matched a pattern. SPARQL allows to bind several values to the same variable through the VALUES clause.

Following the previous example, if we want to perform the same query to calculate the distance between enhancers and their target genes for a subset of enhancers, it is not necessary to perform the same query for each of the sequences of interest. Neither it is necessary to perform one query for all the enhancers and then filter the results of interest. We can use the VALUES clause to indicate all the enhancers we want to query. For example, we will perform the previous query for the two enhancers that we have used as examples: CRMHS00000003515 and CRMHS00000000054: *What are the coordinates of the CRMHS00000003515 and CRMHS00000000054 enhancers and which genes do they regulate? Also calculate the distance that separates both sequences (enhancers and target genes) using as reference the middle position of the enhancer and the start of the gene. Order the results using the distance value from smallest to largest, i.e. order the enhancer-target gene relations showing first the closest neighboring sequences (theoretically more probable).*

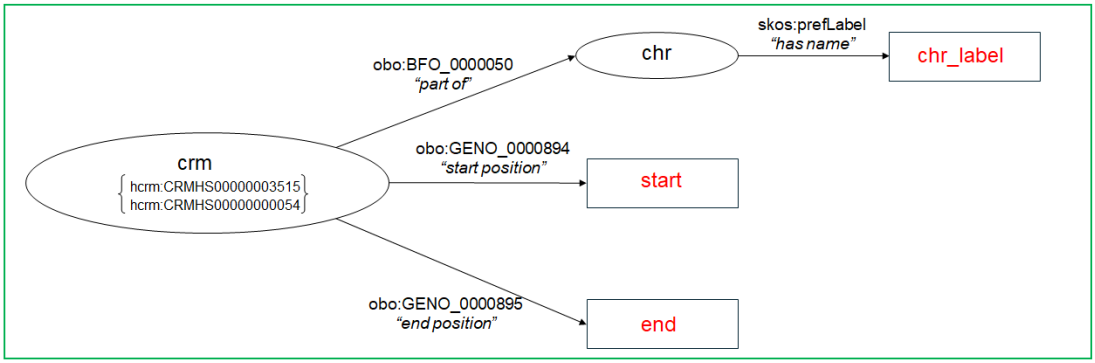
For this query, we indicate the VALUES clause followed by the variable we want to generate and, between brackets, the values we want to give to the variable:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX faldo: <http://biohackathon.org/resource/faldo#>
SELECT DISTINCT ?chr_label ?start ?end ?crm_middle ?gene_label
?TSS_gene ?distance
WHERE {
      GRAPH <http://rdf.biogateway.eu/graph/crm> {
            VALUES ?crm {
               hcrm:CRMHS00000003515
               hcrm:CRMHS00000000054
            }
            ?crm obo:BFO_0000050 ?chr ;
                  obo:GENO_0000894 ?start ;
                  obo:GENO_0000895 ?end .
            ?chr skos:prefLabel ?chr_label .
        }

        GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
                  ?crm obo:RO_0002429 ?tgene .
        }

        GRAPH <http://rdf.biogateway.eu/graph/gene> {
                  ?tgene rdf:type owl:Class ;
                           skos:prefLabel ?gene_label .
                  {
```

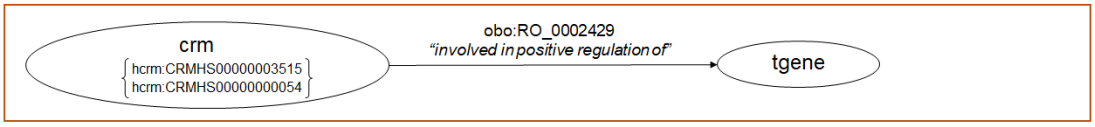
```
                ?tgene obo:GENO_0000906 faldo:ForwardStrandPosition;
                        obo:GENO_0000894 ?TSS_gene .
                } UNION {
                ?tgene obo:GENO_0000906 faldo:ReverseStrandPosition ;
                        obo:GENO_0000895 ?TSS_gene .
                }
        }
        BIND((?end - ?start) AS ?crm_range)
        BIND((ROUND(?crm_range/2) + ?start) AS ?crm_middle)
        BIND(ABS(?TSS_gene - ?crm_middle) AS ?distance)
}
ORDER BY ASC(?distance)
```

BGW knowledge network

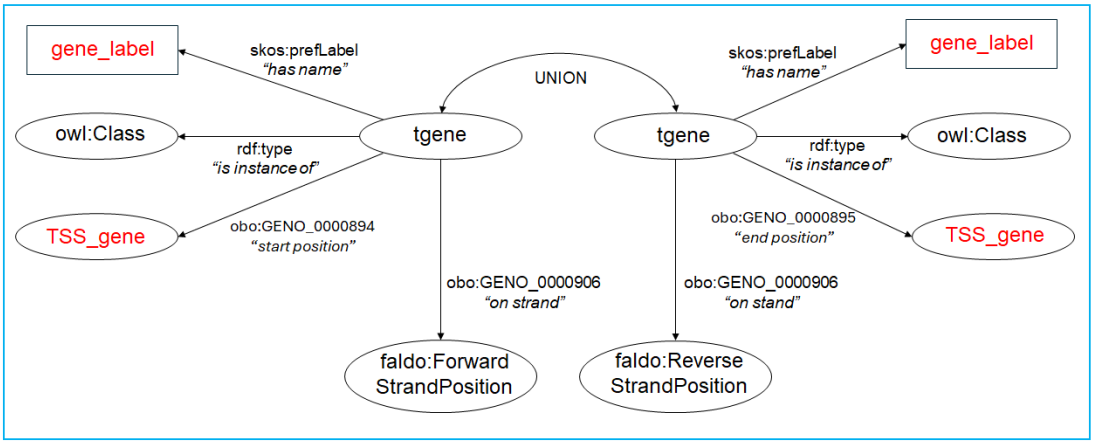
Graph crm



Graph crm2gene



Graph gene



crm_range = end – start
crm_middle = (crm_range/2) + start *rounded value*
distance = | TSS_gene - crm_middle |

Sorting of results by "distance" (ascending order, i.e. from shortest to longest)

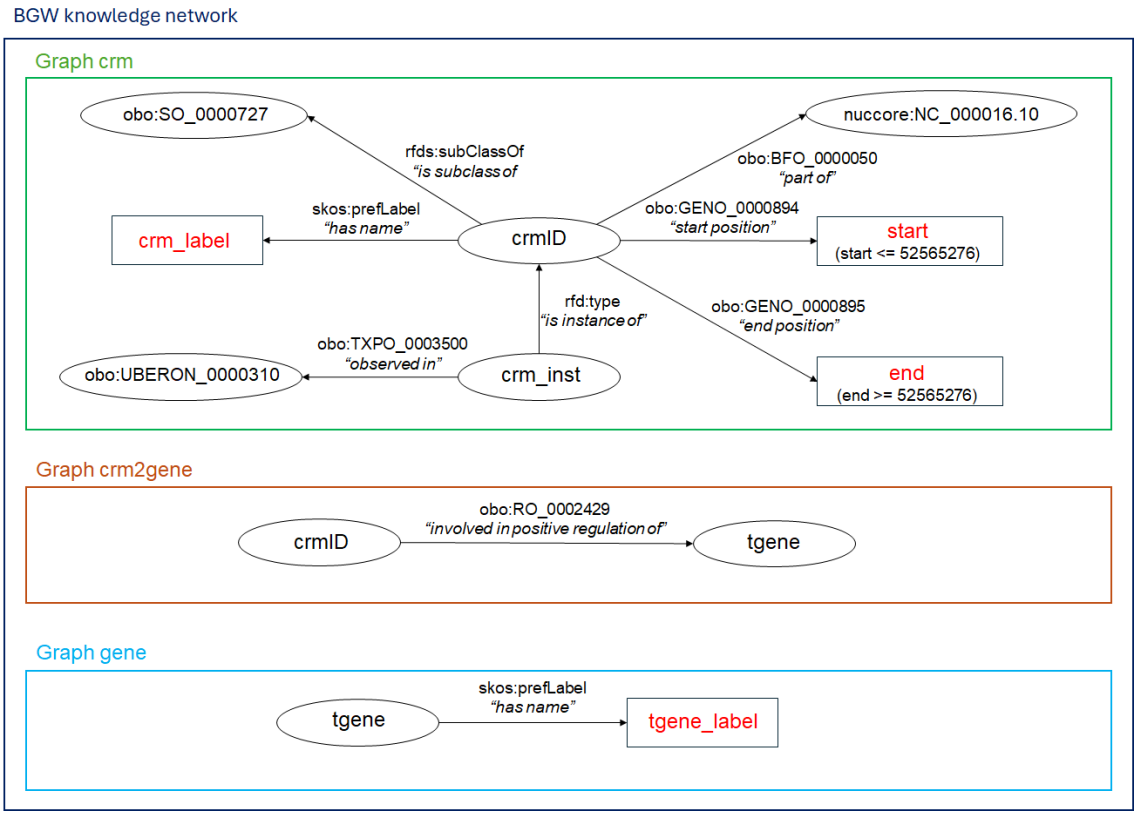
chr_label	start	end	crm_middle	gene_label	TSS_gene	distance
chr-1	18632177	18633790	18632983	PAX7	18630846	2137
chr-12	114025907	114026275	114026091	TBX5	114408708	382617
chr-1	18632177	18633790	18632983	WNT3A	228006998	209374015
chr-1	18632177	18633790	18632983	CHRM3	239386565	220753582

13. Filters

SPARQL also permits to apply filters on variables. The FILTER clause is used for this purpose. If the filter requires the use of a regular expression, this must be accompanied by the regex() function (https://www.w3.org/TR/2013/REC-sparql11-query-20130321/#func-regex).

For example, we can use filters to query which enhancer sequences with target genes overlap with a mutation of interest in a specific tissue, in order to know whether this mutation has the possibility to affect the regulation of a gene. For example, we performed this query using the rs4784227 mutation (chr16:52565276) and breast tissue: *What enhancer sequences, identified in breast tissue, and with information available about their target genes, overlap with the rs4784227 (chr16:52565276) mutation?*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX nuccore: <https://www.ncbi.nlm.nih.gov/nuccore/>
SELECT DISTINCT ?crm_label ?start ?end ?tgene_label
WHERE {
        GRAPH <http://rdf.biogateway.eu/graph/crm> {
                ?crmID rdfs:subClassOf obo:SO_0000727 ;
                        obo:BFO_0000050 nuccore:NC_000016.10 ;
                        obo:GENO_0000894 ?start ;
                        obo:GENO_0000895 ?end ;
                        skos:prefLabel ?crm_label .
                FILTER(?start <= 52565276 && ?end >= 52565276)
                ?crm_inst rdf:type ?crmID ;
                        obo:TXPO_0003500 obo:UBERON_0000310 .
        }
        GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
                ?crmID obo:RO_0002429 ?tgene .
        }

        GRAPH <http://rdf.biogateway.eu/graph/gene> {
                ?tgene skos:prefLabel ?tgene_label
        }
}
```



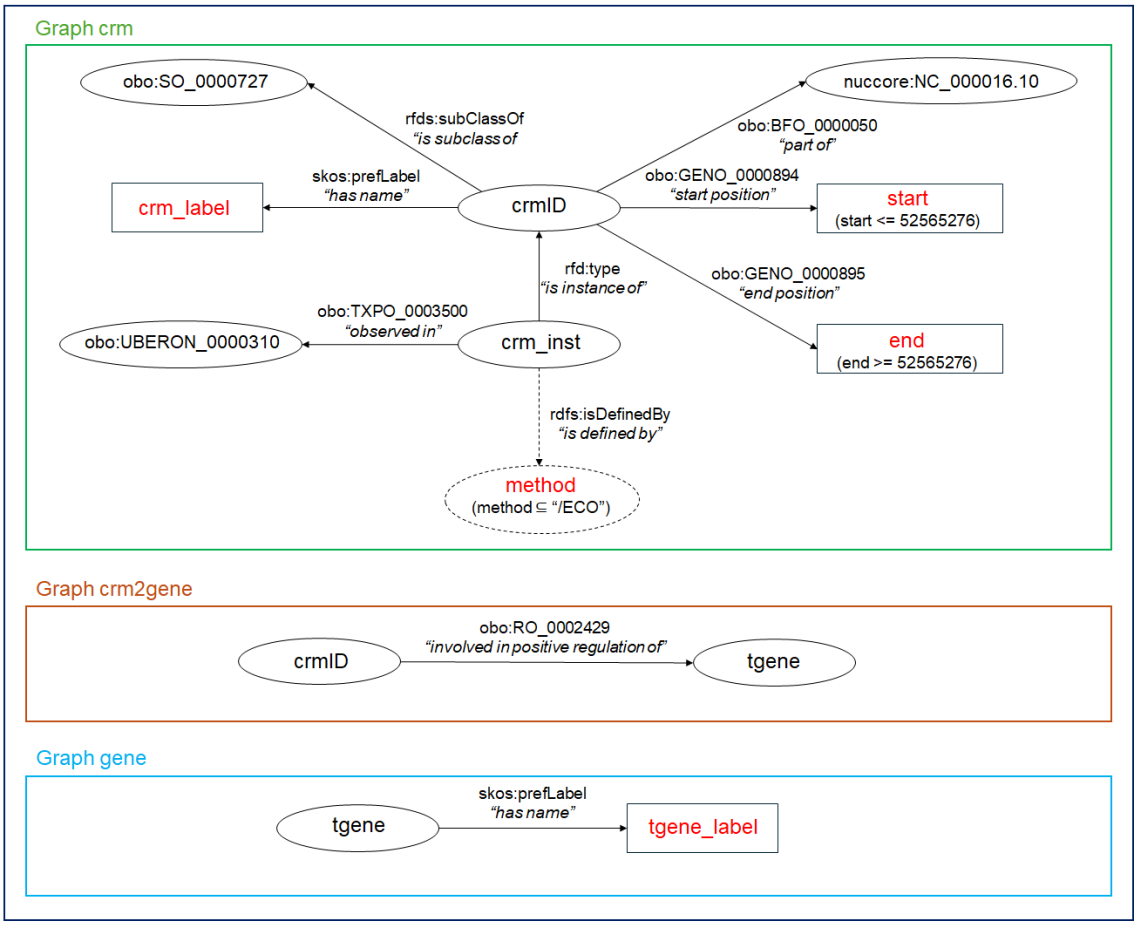
crm_label	start	end	tgene_label
crm/CRMHS00011363179	52544198	52651732	TOX3
crm/CRMHS00011895384	52544643	52578863	TOX3

To illustrate an example with the regex function, we extended this query to optionally include the ECO ontology classes about experimental methods that identified the sequences. Thus, we filter to obtain only the resources of the ECO ontology. *What enhancer sequences, identified in breast tissue, and with information available about their target genes, overlap with the rs4784227 (chr16:52565276) mutation? Include experimental evidence using the ECO ontology as reference.*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX nuccore: <https://www.ncbi.nlm.nih.gov/nuccore/>
SELECT DISTINCT ?crm_label ?start ?end ?tgene_label ?method
WHERE {
        GRAPH <http://rdf.biogateway.eu/graph/crm> {
                ?crmID rdfs:subClassOf obo:SO_0000727 ;
                        obo:BFO_0000050 nuccore:NC_000016.10 ;
                        obo:GENO_0000894 ?start ;
                        obo:GENO_0000895 ?end ;
                        skos:prefLabel ?crm_label .
                FILTER(?start <= 52565276 && ?end >= 52565276)

                ?crm_inst rdf:type ?crmID ;
                        obo:TXPO_0003500 obo:UBERON_0000310 .
```

```
        OPTIONAL {
            ?crm_inst rdfs:isDefinedBy ?method .
            FILTER (regex( ?method, "/ECO_"))
        }
    }

    GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
        ?crmID obo:RO_0002429 ?tgene .
    }

    GRAPH <http://rdf.biogateway.eu/graph/gene> {
        ?tgene skos:prefLabel ?tgene_label
    }
}
```

BGW knowledge network



crm_label	start	end	tgene_label	method
crm/CRMHS00011363179	52544198	52651732	TOX3	http://purl.obolibrary.org/obo/ECO_0000226
crm/CRMHS00011895384	52544643	52578863	TOX3	http://purl.obolibrary.org/obo/ECO_0000226

14. Federated queries

SPARQL enables users to query repositories available on the Web and to merge distributed data. For this reason, it is important to advance in the interoperability between domains and in the implementation of the FAIR principles. With this objective in mind, we have developed the BGW network, with the aim of providing a model interoperable with other data domains and allowing the integration of data from different databases that are currently not interoperable.

To access an endpoint and perform a query, SPARQL has the SERVICE clause. Next to this clause it is necessary to indicate, between angle brackets, the endpoint to be queried and to encapsulate the query between curly brackets:

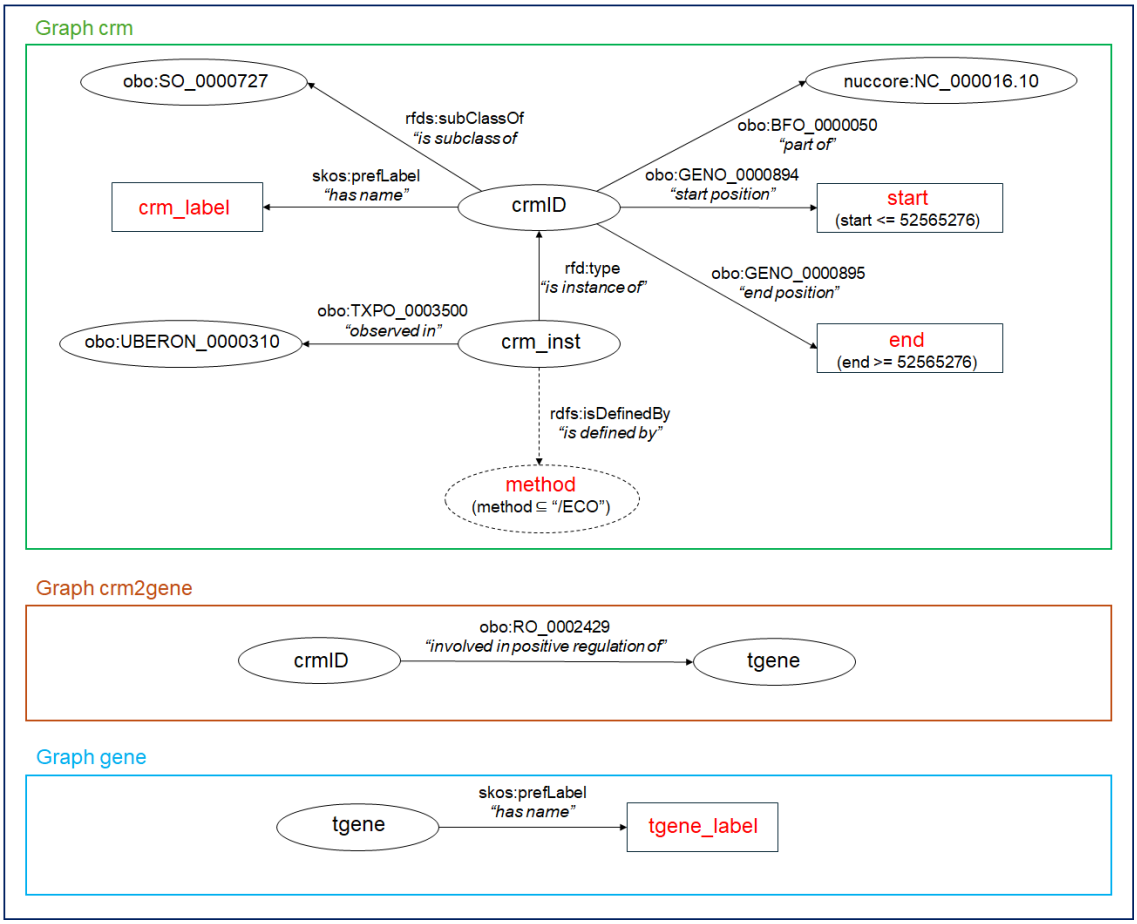
```
SELECT
WHERE {
  SERVICE <endpoint> {query}
}
```

As we noted at the beginning, the BGW network has an accessible endpoint (http://ssb4.nt.ntnu.no:23122/sparql). Therefore, in addition to querying directly through this endpoint, we can also use the SERVICE clause from a local server to perform a query against the BGW network. For example, we modified the previous query in section 13 (*What enhancer sequences, identified in breast tissue, and with information available about their target genes, overlap with the rs4784227 (chr16:52565276) mutation? Include experimental evidence using the ECO ontology as reference*) to run it from a local server, thus we need to use the SERVICE clause.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX nuccore: <https://www.ncbi.nlm.nih.gov/nuccore/>
SELECT DISTINCT ?crm_label ?start ?end ?tgene_label ?method_label
WHERE {
        SERVICE <http://ssb4.nt.ntnu.no:23122/sparql> {
                GRAPH <http://rdf.biogateway.eu/graph/crm> {
                        ?crmID rdfs:subClassOf obo:SO_0000727 ;
                                obo:BFO_0000050 nuccore:NC_000016.10 ;
                                obo:GENO_0000894 ?start ;
                                obo:GENO_0000895 ?end ;
                                skos:prefLabel ?crm_label .
                        FILTER(?start <= 52565276 && ?end >= 52565276)

                        ?crm_inst rdf:type ?crmID ;
                                obo:TXPO_0003500 obo:UBERON_0000310 .

                        OPTIONAL {
                                ?crm_inst rdfs:isDefinedBy ?method .
                                FILTER (regex( ?method, "/ECO_"))
                        }
                }
```

```
                GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
                        ?crmID obo:RO_0002429 ?tgene .
                }

                GRAPH <http://rdf.biogateway.eu/graph/gene> {
                        ?tgene skos:prefLabel ?tgene_label
                }
        }
}
```

BGW knowledge network → http://ssb4.nt.ntnu.no:23122/sparql

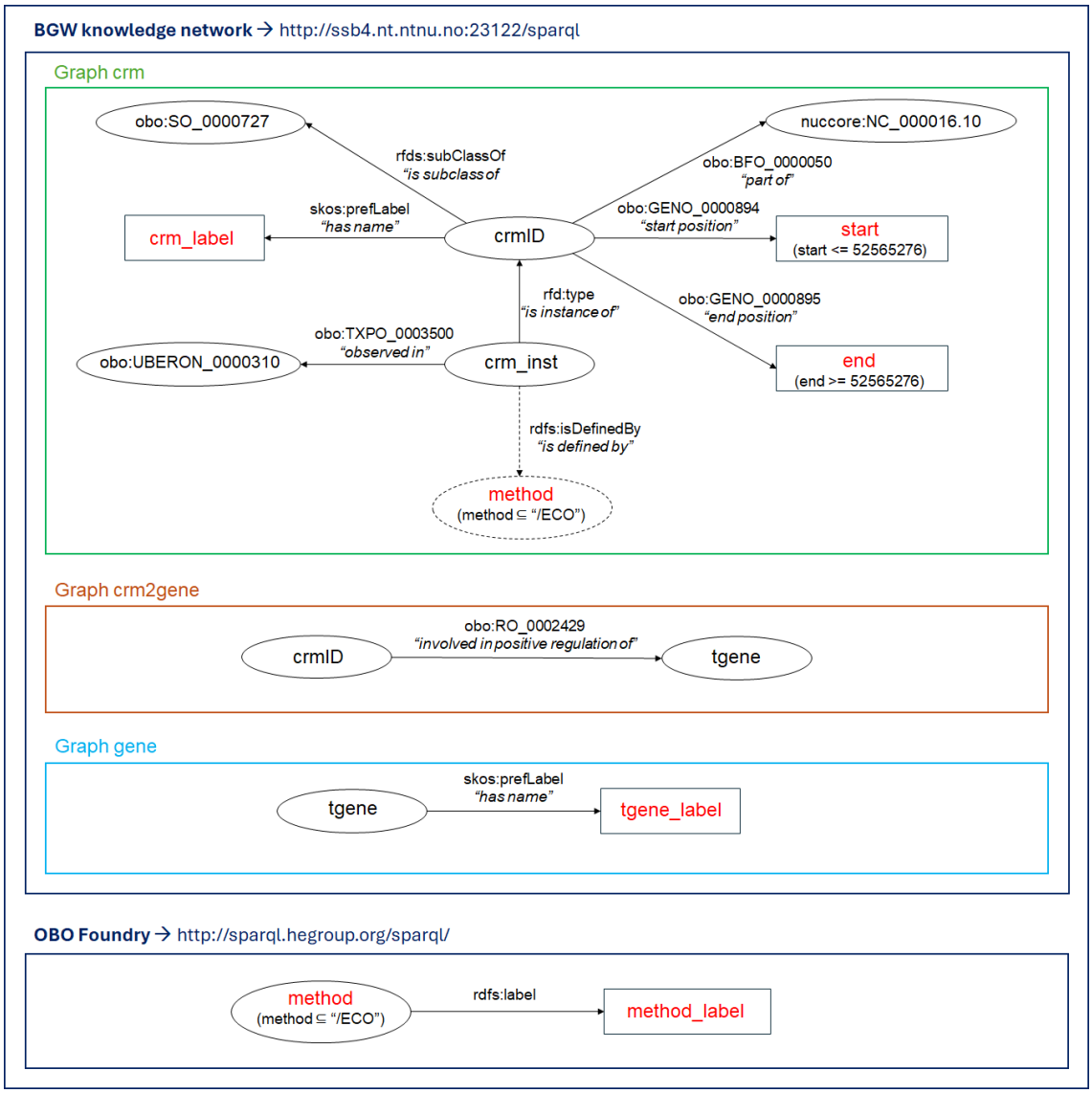


crm_label	start	end	tgene_label	method
crm/CRMHS00011363179	52544198	52651732	TOX3	http://purl.obolibrary.org/obo/ECO_0000226
crm/CRMHS00011895384	52544643	52578863	TOX3	http://purl.obolibrary.org/obo/ECO_0000226

However, the real potential of federated queries is in the combination of different data domains. A couple of examples are illustrated below:

- We use the OBO Foundry endpoint to include in the above query the labels of the obtained methods, since BGW does not import the ECO ontology.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX nuccore: <https://www.ncbi.nlm.nih.gov/nuccore/>
SELECT DISTINCT ?crm_label ?start ?end ?tgene_label ?method_label
WHERE {
        SERVICE <http://ssb4.nt.ntnu.no:23122/sparql> {
                GRAPH <http://rdf.biogateway.eu/graph/crm> {
                        ?crmID rdfs:subClassOf obo:SO_0000727 ;
                                obo:BFO_0000050 nuccore:NC_000016.10 ;
                                obo:GENO_0000894 ?start ;
                                obo:GENO_0000895 ?end ;
                                skos:prefLabel ?crm_label .
                        FILTER(?start <= 52565276 && ?end >= 52565276)

                        ?crm_inst rdf:type ?crmID ;
                                obo:TXPO_0003500 obo:UBERON_0000310 .

                        OPTIONAL {
                                ?crm_inst rdfs:isDefinedBy ?method .
                                FILTER (regex( ?method, "/ECO_"))
                        }
                }

                GRAPH <http://rdf.biogateway.eu/graph/crm2gene> {
                        ?crmID obo:RO_0002429 ?tgene .
                }

                GRAPH <http://rdf.biogateway.eu/graph/gene> {
                        ?tgene skos:prefLabel ?tgene_label
                }
        }

        SERVICE <http://sparql.hegroup.org/sparql/> {
                ?method rdfs:label ?method_label
        }
}
```

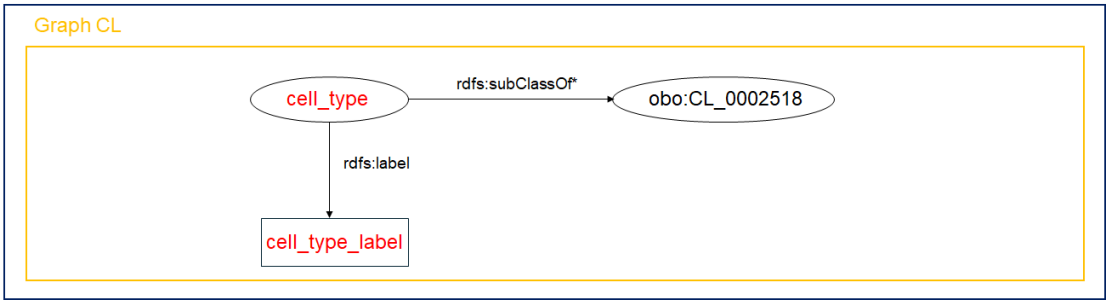


crm_label	start	end	tgene_label	method_label
crm/CRMHS00011363179	52544198	52651732	TOX3	chromatin immunoprecipitation evidence
crm/CRMHS00011895384	52544643	52578863	TOX3	chromatin immunoprecipitation evidence

- In the BGW network we use the CLO, CL and UBERON ontologies to represent three different levels of biological samples: cell lines, cell types and anatomical structures. These ontologies are available in OBO Foundry, so we do not need to load these ontologies on our local server or in the BGW network, we can exploit them directly by federated queries that include the OBO Foundry and BGW endpoints. For example, there are multiple cell types that are subclasses of "kidney epithelial cell" (CL_0002518), so we can perform a query to obtain these subclasses: *What are the different types of "kidney epithelial cell"? That is, what subclasses does this entity contain?*

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT *
WHERE {
      SERVICE <http://sparql.hegroup.org/sparql/> {
            GRAPH <http://purl.obolibrary.org/obo/merged/CL> {
                  ?cell_type rdfs:subClassOf* obo:CL_0002518 ;
                        rdfs:label ?cell_type_label .
            }
      }
}
```

OBO Foundry → http://sparql.hegroup.org/sparql/

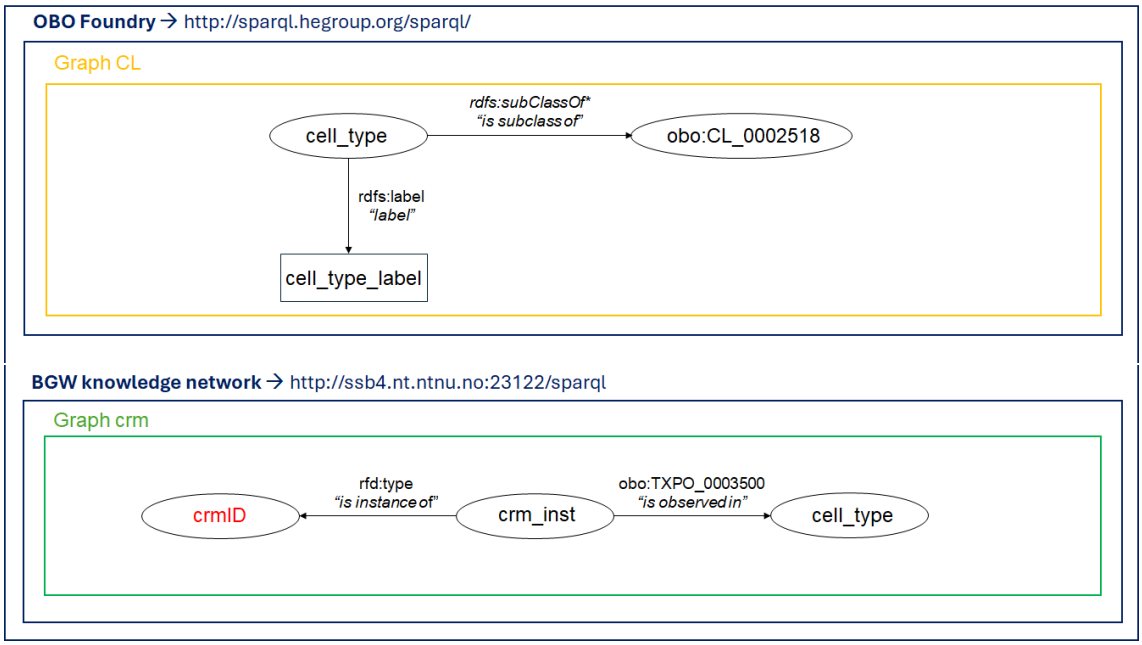


cell_type	cell_type_label
http://purl.obolibrary.org/obo/CL_4030009	epithelial cell of proximal tubule segment 1
http://purl.obolibrary.org/obo/CL_4030011	epithelial cell of proximal tubule segment 3
http://purl.obolibrary.org/obo/CL_4030012	kidney loop of Henle short descending thin limb epithelial cell
http://purl.obolibrary.org/obo/CL_4030014	kidney loop of Henle long descending thin limb inner medulla epithelial cell
http://purl.obolibrary.org/obo/CL_4030013	kidney loop of Henle long descending thin limb outer medulla epithelial cell
http://purl.obolibrary.org/obo/CL_4030008	pronephric podocyte
http://purl.obolibrary.org/obo/CL_4030019	kidney connecting tubule intercalated cell
http://purl.obolibrary.org/obo/CL_4030020	kidney connecting tubule alpha-intercalated cell
http://purl.obolibrary.org/obo/CL_4030021	kidney connecting tubule beta-intercalated cell

Therefore, we can run a query in OBO Foundry that returns all cell types that are kidney epithelial cells and use these results to query in BGW to count, for example: *How many different enhancers have been identified in any kidney epithelial cell type?*

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT COUNT(DISTINCT ?crmID) AS ?numb_crm_kidney_epith_cells
WHERE {
      SERVICE <http://sparql.hegroup.org/sparql/> {
            GRAPH <http://purl.obolibrary.org/obo/merged/CL> {
                  ?cell_type rdfs:subClassOf* obo:CL_0002518 .
            }
```

```
        }
        SERVICE <http://ssb4.nt.ntnu.no:23122/sparql> {
            GRAPH <http://rdf.biogateway.eu/graph/crm> {
                ?crm_int obo:TXPO_0003500 ?cell_type ;
                        rdf:type ?crmID .
            }
        }
    }
```

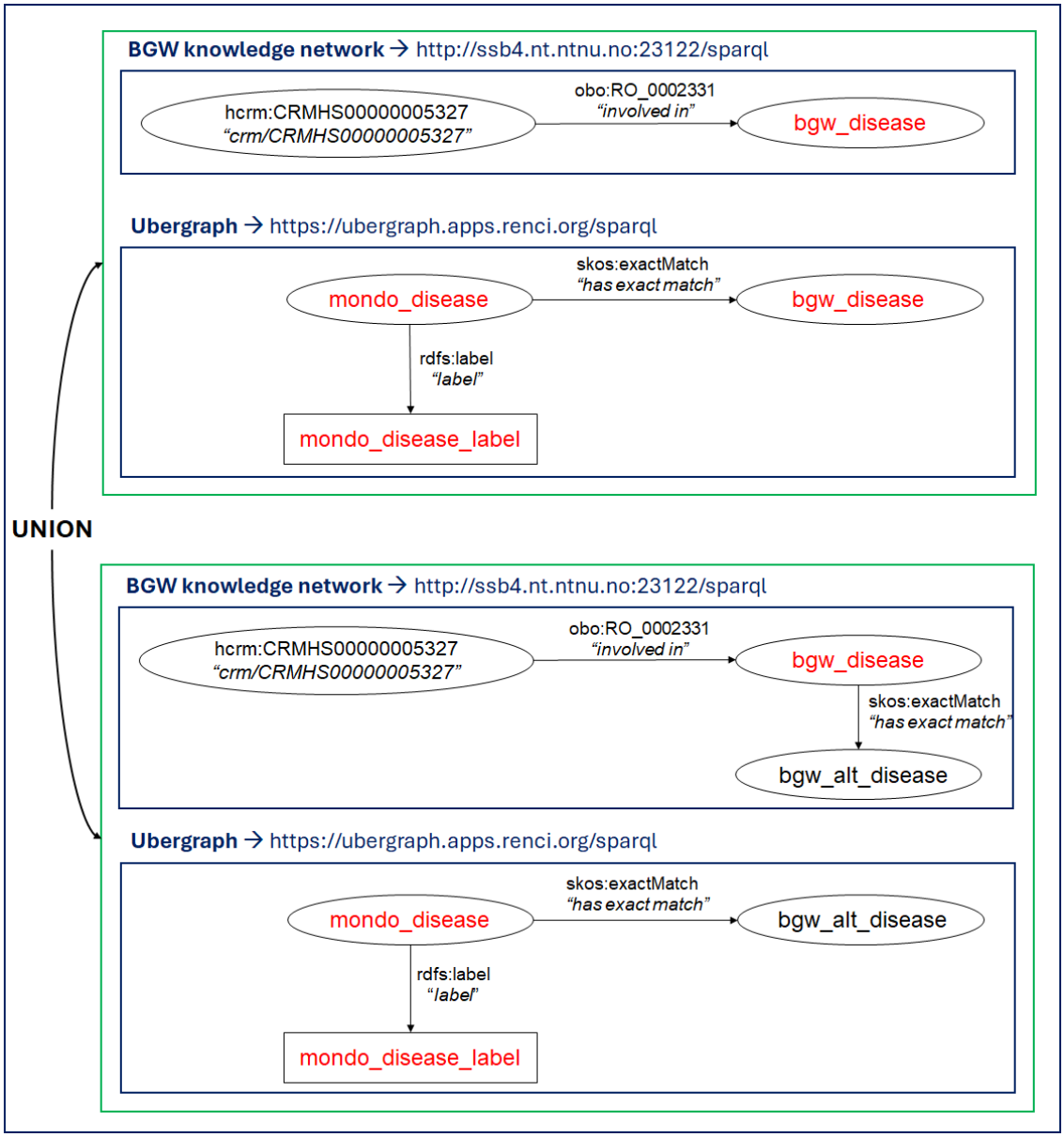


numb_crm_kidney_epith_cells
17358

- Some databases provide ontological resources to represent values. This is the case of diseases in ENdb and EnDisease databases. However, they use three different reference ontologies: Disease Ontology (DO), MeSH and OMIM. Therefore, redundant information can exist. Mondo Disease Ontology aims to merge multiple disease resources to generate a coherent merged ontology. Therefore, we can query this ontology to find if the different disease entries associated with an enhancer correspond to the same disease:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?bgw_disease ?mondo_disease ?mondo_disease_label
WHERE {
    {
    SERVICE <http://ssb4.nt.ntnu.no:23122/sparql> {
        hcrm:CRMHS00000005327 obo:RO_0002331 ?bgw_disease .
        }
```

```
        SERVICE <https://ubergraph.apps.renci.org/sparql> {
            ?mondo_disease skos:exactMatch ?bgw_disease ;
                rdfs:label ?mondo_disease_label .
            }
        } UNION {
        SERVICE <http://ssb4.nt.ntnu.no:23122/sparql> {
            hcrm:CRMHS00000005327 obo:RO_0002331 ?bgw_disease .
            ?bgw_disease skos:exactMatch ?bgw_alt_disease .
            }
        SERVICE <https://ubergraph.apps.renci.org/sparql> {
            ?mondo_disease skos:exactMatch ?bgw_alt_disease ;
                rdfs:label ?mondo_disease_label .
            }
        }
    }
```



bgw_disease	mondo_disease	mondo_disease_label
http://purl.obolibrary.org/obo/DOID_1936	http://purl.obolibrary.org/obo/MONDO_0005311	atherosclerosis
https://id.nlm.nih.gov/mesh/D050197	http://purl.obolibrary.org/obo/MONDO_0005311	atherosclerosis