SPARQL manual for hBGW

1. **Presentation** 2
2. **Introduction** 2
3. **Basic syntax of RDF triples queried through SPARQL** 2
4. **IRIs and variables in SPARQL** 3
   - 4.1. **Use of qnames** 3
   - 4.2. **Use of variables** 4
5. **Basic structure of a SPARQL query** 5
6. **Structure of the results** 7
7. **Queries on specific graphs** 7
8. **Structure of queries with multiple elements** 9
9. **Optional patterns** 10
10. **Union of sets in SPARQL** 11
11. **SPARQL operations** 12
12. **Multiple values** 15
13. **Filters** 16
14. **Federated queries** 18

1. Presentation

The aim of this document is to provide a brief guide to introduce users to SPARQL queries. 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 hBGW model, accessible through its endpoint (http://ssb4.nt.ntnu.no:10022/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 hBGW 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 (https://www.w3.org/TR/rdf11-concepts/).

Example: "114025907"^^<http://www.w3.org/2001/XMLSchema#integer>

In order to avoid duplication and to promote interoperability between domains, as well as other FAIR principles (wilkinson2016fair), in hBGW 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_0000052>
  <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/CLO_0037210>
  <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_0002428>
  <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:

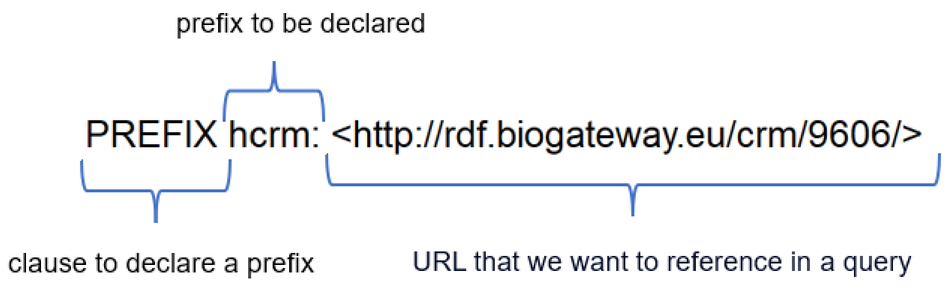
hcrm:CRMHS00000003515 rdfs:subClassOf obo:SO_0000727

hcrm:CRMHS00000003515 obo:RO_0000052 obo:NCBITaxon_9606 hcrm:CRMHS00000003515 obo:CLO_0037210 obo:CLO_0037372

hcrm:CRMHS00000003515 obo:RO_0002428 hgene:TBX5

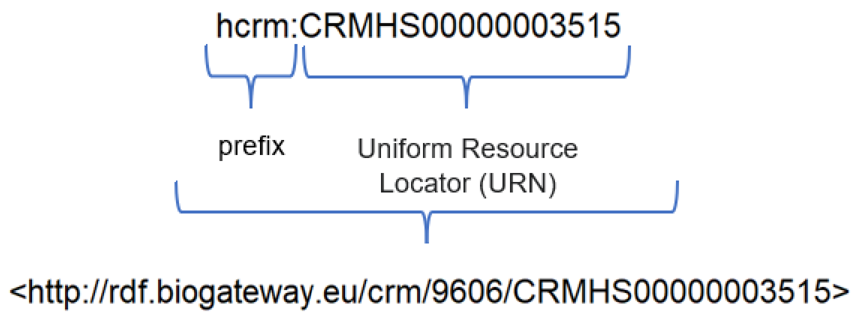
As we can see, prefixes are defined using the PREFIX clause followed by the prefix that we want to define, then by ":" and the URL of the IRI in angle brackets. This URL corresponds to the location of the uniform resource.

For example, in the first definition above:


prefix to be declared
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
clause to declare a prefix
URL that we want to reference in a query

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:


hcrm:CRMHS00000003515
prefix
Uniform Resource Locator (URN)
<http://rdf.biogateway.eu/crm/9606/CRMHS00000003515>

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.

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

- An expression of the type:
  ?subject obo:CLO_0037210 obo:CLO_0037372
  could be used to identify which entities have been identified in the HEK293T cell line.

- An expression of the type:
  ?subject obo:RO_0002428 ?object
  could be used to obtain the nodes that are related through the property "involved in regulation of".

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 { }
```

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 { }
```

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/).

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.

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).

To illustrate examples, we will perform the queries mentioned above in section 4.2. For this, we first access to the hBGW endpoint (http://ssb4.nt.ntnu.no:10022/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 .
  }
  ```

  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 .
  }
  ```

- 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
  }
  ```

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

  ```
  PREFIX obo: <http://purl.obolibrary.org/obo/>
  SELECT ?subject
  WHERE {
    ?subject obo:CLO_0037210 obo:CLO_0037372
  }
  ```

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

  ```
  PREFIX obo: <http://purl.obolibrary.org/obo/>
  ```

```
SELECT ?subject ?object
WHERE {
  ?subject obo:RO_0002428 ?object
}
```

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 regulation of"?) and we show the first 10 results:

"subject"	"object"
"http://rdf.biogateway.eu/prot/9606/A1YPR0"	"http://rdf.biogateway.eu/gene/9606/CDKN1A"
"http://rdf.biogateway.eu/prot/9606/A1YPR0"	"http://rdf.biogateway.eu/gene/9606/FASN"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/TRIM28"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/PDLIM7"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/RGS2"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/S100A9"
"http://rdf.biogateway.eu/prot/9606/A6NJT0"	"http://rdf.biogateway.eu/gene/9606/UNCX"
"http://rdf.biogateway.eu/prot/9606/A6NJT0"	"http://rdf.biogateway.eu/gene/9606/EGLN1"
"http://rdf.biogateway.eu/prot/9606/A6NJT0"	"http://rdf.biogateway.eu/gene/9606/EGLN3"
"http://rdf.biogateway.eu/prot/9606/O00267"	"http://rdf.biogateway.eu/gene/9606/ANKRD37"

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. In the query above we have obtained the nodes that are related through the property "involved in regulation of". 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 to indicate that a protein, that is a TF, is involved in the regulation of a gene. On the other hand, in the CRM graph it is used to indicate cis-regulatory sequences involved in the 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).

- Graph tfac2gene

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?subject ?object
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/tfac2gene> {
    ?subject obo:RO_0002428 ?object
  }
}
LIMIT 10
```

"subject"	"object"
"http://rdf.biogateway.eu/prot/9606/A1YPR0"	"http://rdf.biogateway.eu/gene/9606/CDKN1A"
"http://rdf.biogateway.eu/prot/9606/A1YPR0"	"http://rdf.biogateway.eu/gene/9606/FASN"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/TRIM28"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/PDLIM7"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/RGS2"
"http://rdf.biogateway.eu/prot/9606/A6NFI3"	"http://rdf.biogateway.eu/gene/9606/S100A9"
"http://rdf.biogateway.eu/prot/9606/A6NJT0"	"http://rdf.biogateway.eu/gene/9606/UNCX"
"http://rdf.biogateway.eu/prot/9606/A6NJT0"	"http://rdf.biogateway.eu/gene/9606/EGLN1"
"http://rdf.biogateway.eu/prot/9606/A6NJT0"	"http://rdf.biogateway.eu/gene/9606/EGLN3"
"http://rdf.biogateway.eu/prot/9606/O00267"	"http://rdf.biogateway.eu/gene/9606/ANKRD37"

- Graph crm

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?subject ?object
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/crm> {
    ?subject obo:RO_0002428 ?object
  }
}
LIMIT 10
```

"subject"	"object"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000096925#fantom5"	"http://rdf.biogateway.eu/gene/9606/SPNS1"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097191#fantom5"	"http://rdf.biogateway.eu/gene/9606/ADCY7"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097191#fantom5"	"http://rdf.biogateway.eu/gene/9606/NOD2"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097191#fantom5"	"http://rdf.biogateway.eu/gene/9606/CYLD"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097191#fantom5"	"http://rdf.biogateway.eu/gene/9606/SNX20"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097338#fantom5"	"http://rdf.biogateway.eu/gene/9606/CCL22"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097532#fantom5"	"http://rdf.biogateway.eu/gene/9606/TPPP3"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097532#fantom5"	"http://rdf.biogateway.eu/gene/9606/CENPT"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097626#fantom5"	"http://rdf.biogateway.eu/gene/9606/EXOSC6"
"http://rdf.biogateway.eu/crm/9606/CRMHS00000097776#fantom5"	"http://rdf.biogateway.eu/gene/9606/TERF2IP"

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. This document covers the basics, 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, we must include a full stop at the end of each triplet if we want to include more than one triplet as a search pattern. If the subject of the node does not change in the next triplet, we can use ";" and omit the subject.

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 coordinates does this enhancer have, and which target genes does it regulate? We first exemplify this query by using "." at the end of each triplet and then using ";".

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/crm> {
    hcrm:CRMHS00000003515 obo:RO_0001025 ?chr .
    hcrm:CRMHS00000003515 obo:OGI_1000004 ?start .
    hcrm:CRMHS00000003515 obo:OGI_1000003 ?end .
    hcrm:CRMHS00000003515 obo:RO_0001025 ?chr .
    hcrm:CRMHS00000003515 obo:RO_0002428 ?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"

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/crm> {
    hcrm:CRMHS00000003515 obo:RO_0001025 ?chr ;
      obo:OGI_1000004 ?start ;
      obo:OGI_1000003 ?end ;
      obo:RO_0001025 ?chr ;
      obo:RO_0002428 ?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

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 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 coordinates does a specific enhancer have and what 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:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene ?TF
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/crm> {
    hcrm:CRMHS00000000054 obo:RO_0001025 ?chr ;
      obo:OGI_1000004 ?start ;
      obo:OGI_1000003 ?end ;
      obo:RO_0001025 ?chr ;
      obo:RO_0002428 ?target_gene ;
      obo:RO_0002436 ?TF
  }
}
```

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

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:

```
PREFIX hcrm: <http://rdf.biogateway.eu/crm/9606/>
PREFIX obo: <http://purl.obolibrary.org/obo/>
SELECT ?chr ?start ?end ?target_gene ?TF
WHERE {
  GRAPH <http://rdf.biogateway.eu/graph/crm> {
    hcrm:CRMHS00000000054 obo:RO_0001025 ?chr ;
      obo:OGI_1000004 ?start ;
      obo:OGI_1000003 ?end ;
      obo:RO_0001025 ?chr ;
      obo:RO_0002428 ?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

In addition, SPARQL 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
```

"protein"	"posit_tf2gene"	"neg_tf2gene"
"http://rdf.biogateway.eu/prot/9606/A1YPR0"	"http://rdf.biogateway.eu/gene/9606/FASN"	
"http://rdf.biogateway.eu/prot/9606/O00570"	"http://rdf.biogateway.eu/gene/9606/NEUROG1"	
"http://rdf.biogateway.eu/prot/9606/O14948"	"http://rdf.biogateway.eu/gene/9606/MYH9"	
"http://rdf.biogateway.eu/prot/9606/O15119"	"http://rdf.biogateway.eu/gene/9606/JDP2"	
"http://rdf.biogateway.eu/prot/9606/O15156"	"http://rdf.biogateway.eu/gene/9606/CD4"	
"http://rdf.biogateway.eu/prot/9606/O15156"	"http://rdf.biogateway.eu/gene/9606/COL2A1"	
"http://rdf.biogateway.eu/prot/9606/O15178"	"http://rdf.biogateway.eu/gene/9606/OTP"	
"http://rdf.biogateway.eu/prot/9606/O15178"	"http://rdf.biogateway.eu/gene/9606/IFNG"	
"http://rdf.biogateway.eu/prot/9606/O15350"	"http://rdf.biogateway.eu/gene/9606/TP53I3"	
"http://rdf.biogateway.eu/prot/9606/O15350"	"http://rdf.biogateway.eu/gene/9606/VDR"	

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}
  }
}
```

"callret-0"	"callret-1"
14552	6407

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:

```
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"
14552	6407

However, it is important to note that the query performed follows the pattern of the protein-property_relation-targetgene 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:

```
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"
3304	1885

- 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, if we take the CRMHS00000000054 enhancer from the last example with enhancers, we can perform a query that calculates its middle position using its initial and final coordinates, generate a variable with this average position and use it to determine the distance that separates this enhancer from the start site of its target genes. For this purpose, if the target gene is located on the "+" strand, we use the start position of the gene as reference, while we use the end position if the gene is located on the "-" strand:

```
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#>
SELECT DISTINCT ?chr_label ?start ?end ?enh_middle ?gene_label
?TSS_gene ?distance
WHERE {
     GRAPH <http://rdf.biogateway.eu/graph/crm> {
          hcrm:CRMHS00000000054 rdf:type owl:Class ;
               obo:OGI_1000004 ?start;
               obo:OGI_1000003 ?end ;
               obo:RO_0001025 ?chr ;
               obo:RO_0002428 ?tgene .
          ?chr skos:prefLabel ?chr_label .
       }

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

```
                    obo:GENO_0000906 "+" ;
                    obo:OGI_1000004 ?TSS_gene .
            } UNION {
            ?tgene rdf:type owl:Class ;
                    skos:prefLabel ?gene_label ;
                    obo:GENO_0000906 "-" ;
                    obo:OGI_1000003 ?TSS_gene .
            }
        }
        BIND((?end - ?start) AS ?enh_range)
        BIND((ROUND(?enh_range/2) + ?start) AS ?enh_middle)
        BIND(ABS(?TSS_gene - ?enh_middle) AS ?distance)
}
```

"chr_label"	"start"	"end"	"enh_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":

```
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#>
SELECT DISTINCT ?chr_label ?start ?end ?enh_middle ?gene_label
?TSS_gene ?distance
WHERE {
    GRAPH <http://rdf.biogateway.eu/graph/crm> {
        hcrm:CRMHS00000000054 rdf:type owl:Class ;
            obo:OGI_1000004 ?start;
            obo:OGI_1000003 ?end ;
            obo:RO_0001025 ?chr ;
            obo:RO_0002428 ?tgene .
        ?chr skos:prefLabel ?chr_label .
    }

    GRAPH <http://rdf.biogateway.eu/graph/gene> {
            {
            ?tgene rdf:type owl:Class ;
                    skos:prefLabel ?gene_label ;
                    obo:GENO_0000906 "+" ;
                    obo:OGI_1000004 ?TSS_gene .
```

```
                } UNION {
                ?tgene rdf:type owl:Class ;
                        skos:prefLabel ?gene_label ;
                        obo:GENO_0000906 "-" ;
                        obo:OGI_1000003 ?TSS_gene .
                }
        }

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

"chr_label"	"start"	"end"	"enh_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. For this, 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#>
SELECT DISTINCT ?chr_label ?start ?end ?enh_middle ?gene_label
?TSS_gene ?distance
WHERE {
    GRAPH <http://rdf.biogateway.eu/graph/crm> {
        VALUES ?enh {
          hcrm:CRMHS00000003515
          hcrm:CRMHS00000000054
```

```
            }
            ?enh rdf:type owl:Class ;
                        obo:OGI_1000004 ?start;
                 obo:OGI_1000003 ?end ;
                 obo:RO_0001025 ?chr ;
                 obo:RO_0002428 ?tgene .
            ?chr skos:prefLabel ?chr_label .
        }

        GRAPH <http://rdf.biogateway.eu/graph/gene> {
                {
                ?tgene rdf:type owl:Class ;
                        skos:prefLabel ?gene_label ;
                        obo:GENO_0000906 "+" ;
                        obo:OGI_1000004 ?TSS_gene .
                } UNION {
                ?tgene rdf:type owl:Class ;
                        skos:prefLabel ?gene_label ;
                        obo:GENO_0000906 "-" ;
                        obo:OGI_1000003 ?TSS_gene .
                }
        }

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

"chr_label"	"start"	"end"	"enh_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 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):

```
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#>
SELECT DISTINCT ?enh_label ?chr_label ?start ?end ?tgene_label
WHERE {
      GRAPH <http://rdf.biogateway.eu/graph/crm> {
            ?enhID rdf:type owl:Class ;
                  obo:OGI_1000004 ?start;
                  obo:OGI_1000003 ?end ;
                  obo:RO_0001025 ?chr ;
                  skos:prefLabel ?enh_label ;
                  obo:RO_0002428 ?tgene .
            ?chr skos:prefLabel ?chr_label .
            FILTER (?chr_label = "chr-16" && ?start <= 52565276 && ?end >= 52565276)
        }

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

"enh_label"	"chr_label"	"start"	"end"	"tgene_label"
"crm/CRMHS00000032653"	"chr-16"	52528332	52580463	"TOX3"
"crm/CRMHS00000032654"	"chr-16"	52544365	52579842	"TOX3"
"crm/CRMHS00000005858"	"chr-16"	52556490	52566288	"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.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
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#>
SELECT DISTINCT ?enh_label ?chr_label ?start ?end ?tgene_label ?method
WHERE {
      GRAPH <http://rdf.biogateway.eu/graph/crm> {
            ?enhID rdf:type owl:Class ;
                  obo:OGI_1000004 ?start;
                  obo:OGI_1000003 ?end ;
                  obo:RO_0001025 ?chr ;
                  skos:prefLabel ?enh_label ;
                  obo:RO_0002428 ?tgene .
            OPTIONAL {
              ?enhID rdfs:isDefinedBy ?method .
               FILTER (regex( ?method, "/ECO_"))
             }
            ?chr skos:prefLabel ?chr_label .
            FILTER (?chr_label = "chr-16" && ?start <= 52565276 && ?end >= 52565276)
        }
      GRAPH <http://rdf.biogateway.eu/graph/gene> {
            ?tgene skos:prefLabel ?tgene_label
      }
}
```

enh_label	chr_label	start	end	tgene_label	method
"crm/CRMHS00000032653"	chr-16	52528332	52580463	"TOX3"	http://purl.obolibrary.org/obo/ECO_0000226
"crm/CRMHS00000032654"	chr-16	52544365	52579842	"TOX3"	http://purl.obolibrary.org/obo/ECO_0000226
"crm/CRMHS00000005858"	chr-16	52556490	52566288	"TOX3"	

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 hBGW 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 hBGW network has an accessible endpoint (http://ssb4.nt.ntnu.no:10022/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 hBGW network. For example, we modified the previous query in section 13 (filters) to run it from a local server, thus we need to use the SERVICE clause.

```
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#>
SELECT DISTINCT ?enh_label ?chr_label ?start ?end ?tgene_label
WHERE {
      SERVICE <http://ssb4.nt.ntnu.no:10022/sparql> {
            GRAPH <http://rdf.biogateway.eu/graph/crm> {
                  ?enhID rdf:type owl:Class ;
```

```
                    obo:OGI_1000004 ?start;
                    obo:OGI_1000003 ?end ;
                    obo:RO_0001025 ?chr ;
                    skos:prefLabel ?enh_label ;
                    obo:RO_0002428 ?tgene .
                ?chr skos:prefLabel ?chr_label .
                FILTER (?chr_label = "chr-16" && ?start <= 52565276 && ?end >=
52565276)
            }

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

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

- In the hBGW 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 hBGW network, we can exploit them directly by federated queries that include the OBO Foundry and hGBW endpoints. For example, there are multiple cell types that are subclasses of "kidney epithelial cell" (CL_0002518):

  ```
  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 .
          }
      }
  }
  ```

  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 hBGW to count, for example, how many different enhancers have been identified in any of these cell types:

  ```
  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 ?enhID) AS ?numb_enh_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:10022/sparql> {
            GRAPH <http://rdf.biogateway.eu/graph/crm> {
                ?enhID obo:CLO_0037209 ?cell_type ;
                    rdf:type owl:Class .
            }
        }
    }
```

numb_enh_kidney_epith_cells
56

- 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:10022/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:10022/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