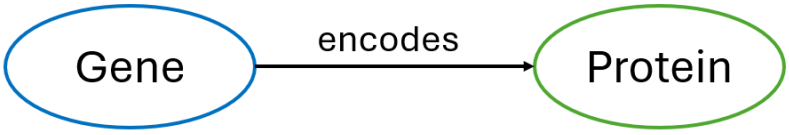
INTUITION Tutorial for BioGateway

1. Introduction

INTUITION (https://semantics.inf.um.es/intuition/) is a web application for user-friendly SPARQL query building. In this way, users can exploit RDF knowledge graphs without knowledge in SPARQL query language.

INTUITION analyses the knowledge network of an accessible endpoint, in this case, the current instance of BioGateway (http://ssb4.nt.ntnu.no:23122/sparql), and allows building biological queries graphically by defining search patterns: biological entities (nodes) that can be specified in detail through variables and are related through properties (edges).

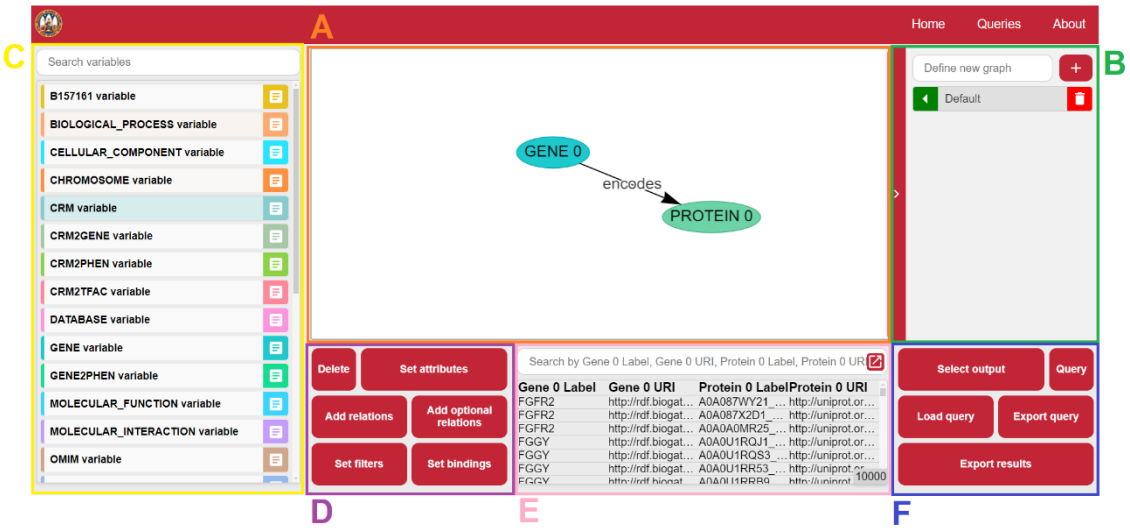
In a graph, nodes represent different types of biological entities, such as genes, proteins or CRMs, and edges (or properties) are used to specify different types of relations that exist between two nodes (for example, <Gene> <encodes> <Protein>). Some properties are also used to add attributes to entities.



2. Design

In INTUITION we distinguish different sections:

A- Query building canvas.
B- Union builder (Used for queries that use union clauses).
C- Variable browser (Main types of entities in the network).
D- Pattern designer (nodes and links):
- Set attributes: Allows a user to edit the intrinsic properties of a variable, that distinguish one entity from another. For example, the name or description of a gene. Therefore, its inclusion acts as a filter because it permits to specify the entity or entities with a certain pattern of characteristics.
- Add relations: Allows a user to include relations between entities to build queries of greater biological relevance. The inclusion of relations implies the insertion of a search pattern, so its use also selects/filters the knowledge network.
- Add optional relations: Allows a user to include optional relations between entities. Their use does not act as a filter, but adds information when the pattern is met.
- Set bindings: Enables the creation of custom variables using attributes included in the search pattern, renamed, or selected for inclusion in the output.
- Set filters. Button to specify filters on attributes used in the search pattern, renamed, or selected to be included in the output. It can also be used to apply filters on new variables (bindings).
E- Output display.
F- Query builder:
- Select output: output selector. Allow to indicate which variables are shown in the output screen.
- Query: runs the query.
- Load query: to load a previously designed query.
- Export query: export the designed query.
- Export results: exports the query results.



3. Variables and properties

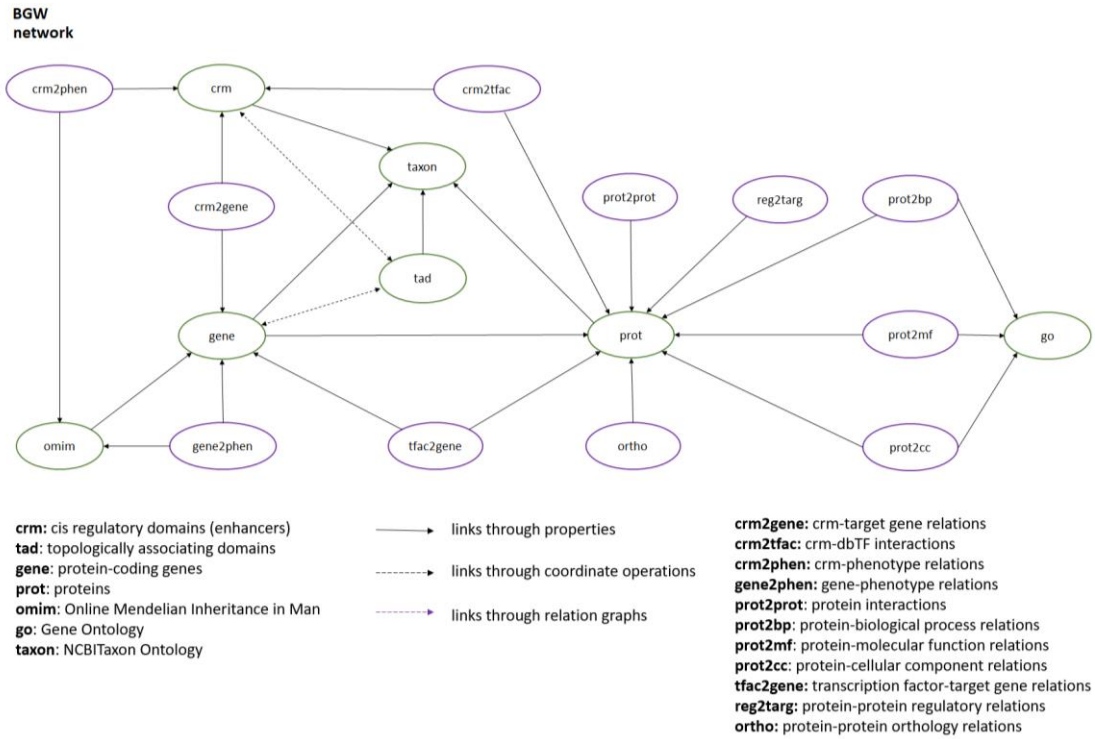
Variables correspond to biological entities that we can use to develop query patterns. We can group these variables into 3 groups:

1- Variables corresponding to biological entities modelled by BioGateway:
   - Gene variable: protein-coding genes.
   - Protein variable: proteins.
   - CRM variable: cis-regulatory module (currently only enhancers).
   - TAD variable: topologically associated domain.
   - Database variable: databases.
   - Chromosome variable: chromosomes.
   - Reference_genome variable: genome assembly.
   - Transcription factor variable: transcription factors (currently only proteins that interact with CRM).

2- Variables corresponding to biological entities imported from external ontologies:
   - OMIM variable: entities from OMIM ontology (mainly phenotypes).
   - Molecular_interaction: entities from Molecular Interactions ontology (MI).
   - Cellular_component variable: cellular components from Gene Ontology (GO).
   - Molecular_function variable: molecular functions from GO.
   - Biological_process variable: biological processes from GO.
   - Root variable: top hierarchically class of NCBITaxon Ontology.
   - Taxonomic_rank variable: top hierarchically class of NCBITaxon Ontology.

3- Variables, modelled by BioGateway, corresponding to relations between biological entities:
   - crm2gene variable: relation between CRM and gene.
   - crm2phen variable: relation between CRM and phenotype.
   - crm2tfac variable: relation between CRM and protein (transcription factor).
   - gene2phen variable: relation between gene and phenotype.
   - tfac2gene variable: relation between gene and protein.
   - prot2prot: molecular interaction relation between proteins.
   - reg2targ variable: regulatory relation between proteins.
   - Ortho variable: orthology relation between proteins.
   - prot2cc variable: relation between protein and its cellular components.
   - prot2mf: relation between protein and its molecular functions.
   - prot2bp variable: relation between protein and its biological processes.

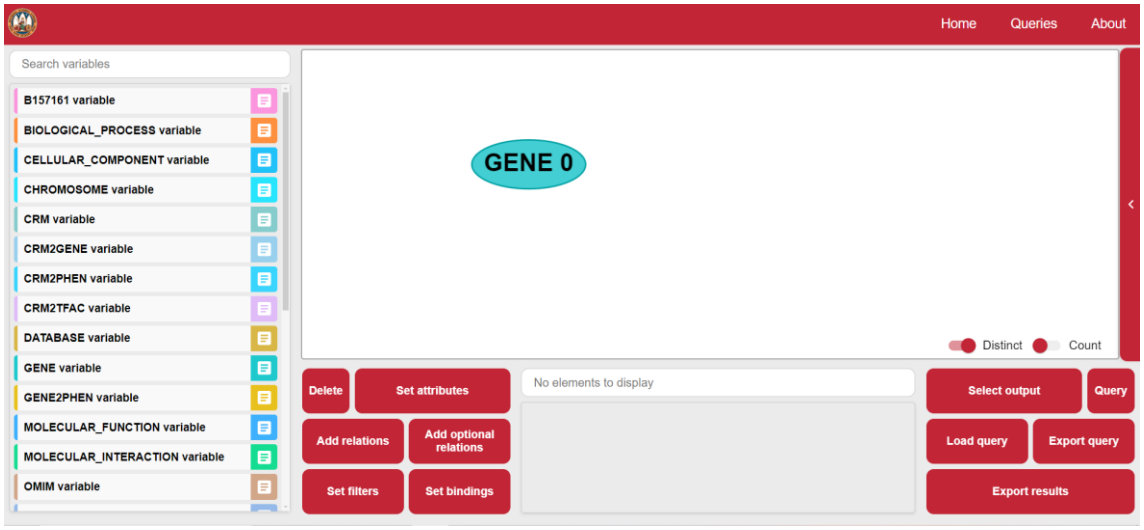
Properties are used to semantically relate different biological entities, and to provide attributes to these entities. These entities are detailed with examples and their domains here. It also includes information about the vocabularies used.



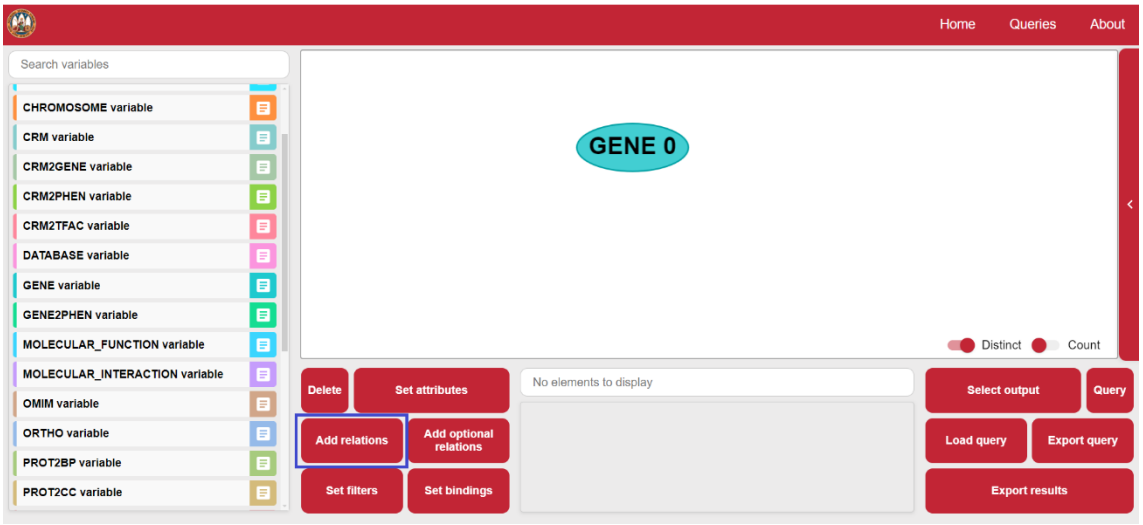
4. How to build a query in 6 steps

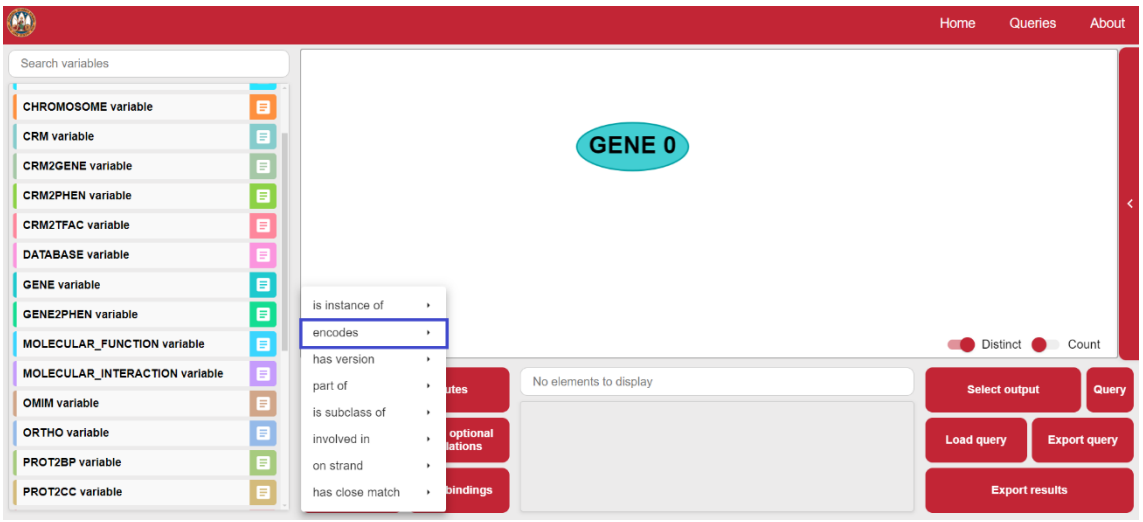
The query building process involves linking entities (nodes) with their attributes and/or other entities through properties (edges). We take as an example the previous case, the query: *Which proteins do the different genes encode?* (<Gene> <encodes> <Protein>).

1. Select the first entity (subject node), in this case, “Gene”, in the “Variable browser”.

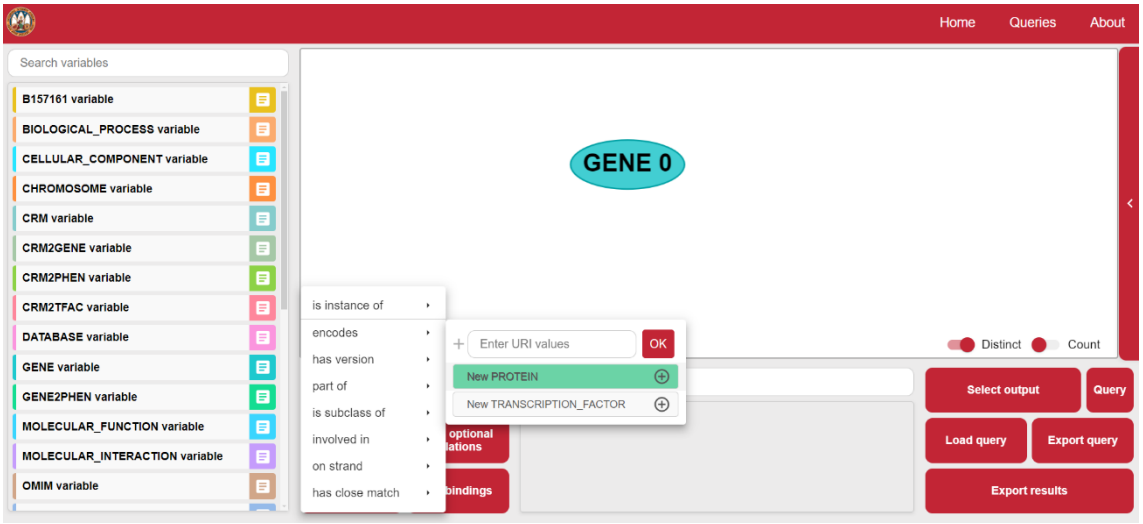


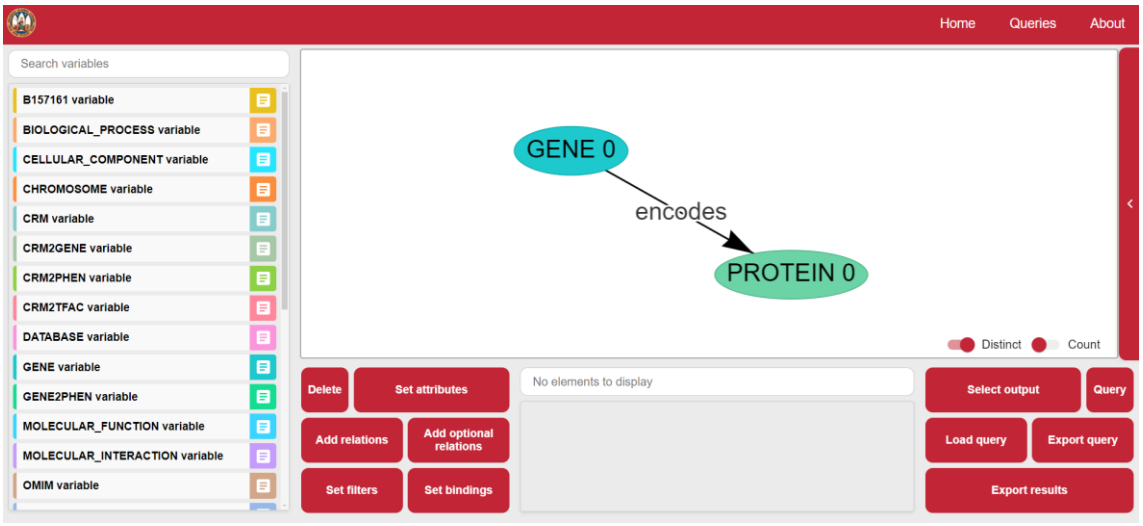
2. Select the node of interest and then, in “Add relations”, select the type of relation you want to use. In this case, “encodes”.



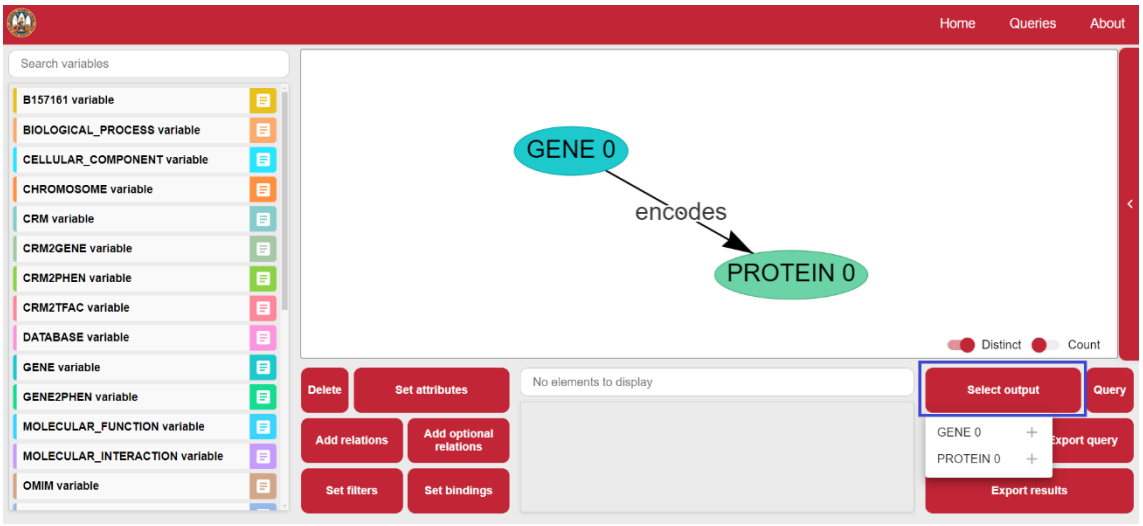


3. Select the second entity (object node), in this case, "Protein".

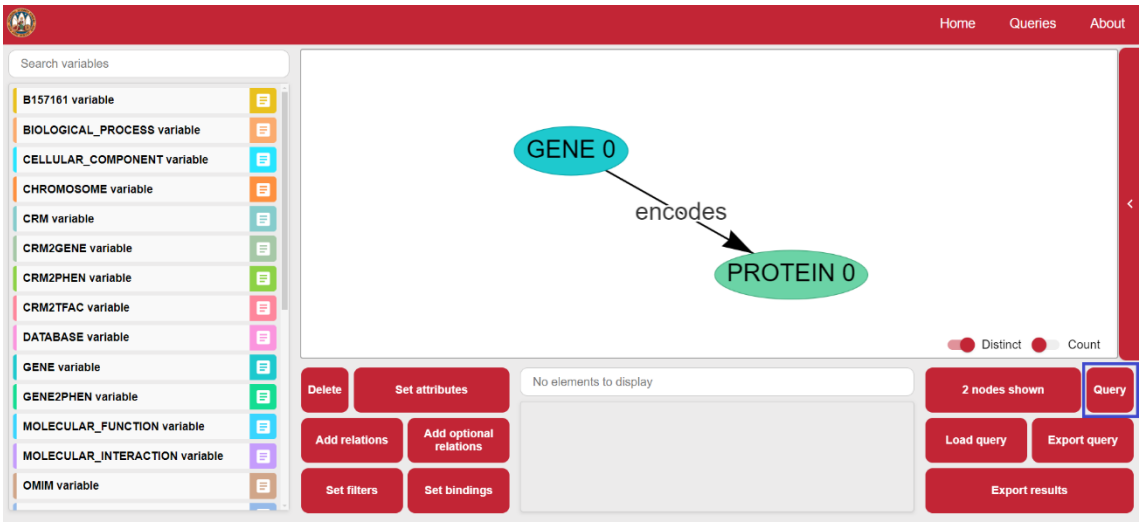




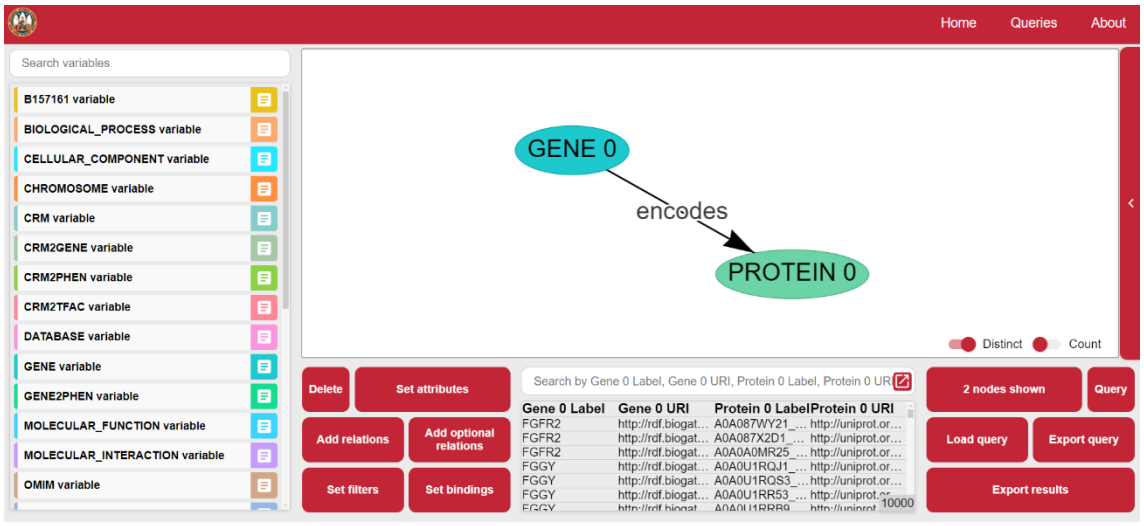
4. Select in "Select output" the data you want to show in the output (click on “+”).



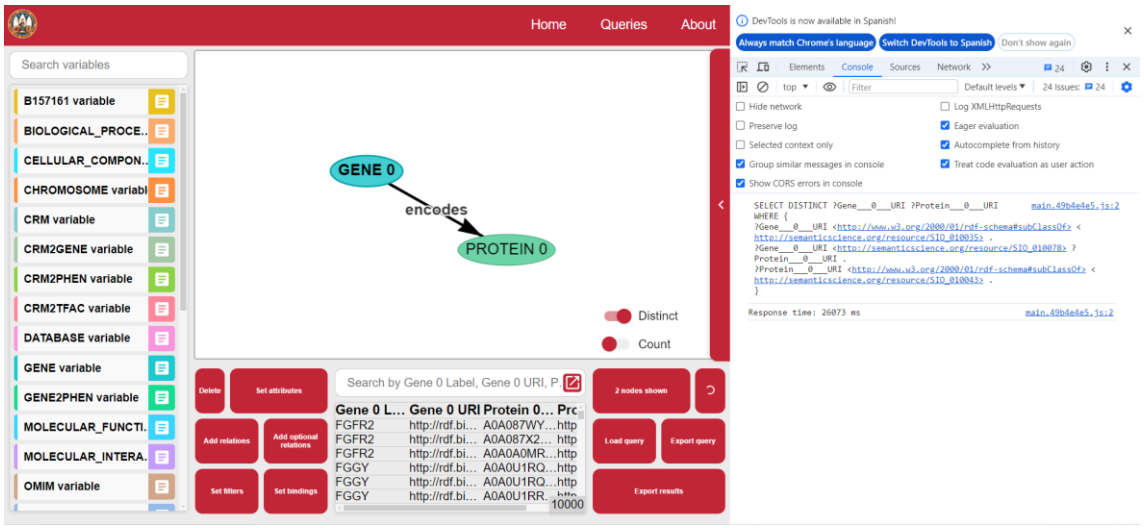
5. Click on "Query" to launch the query.



6. The results are available in the “Output display”. Click on "Export results" to download the data. Click on "Export query" to save the query (json file).

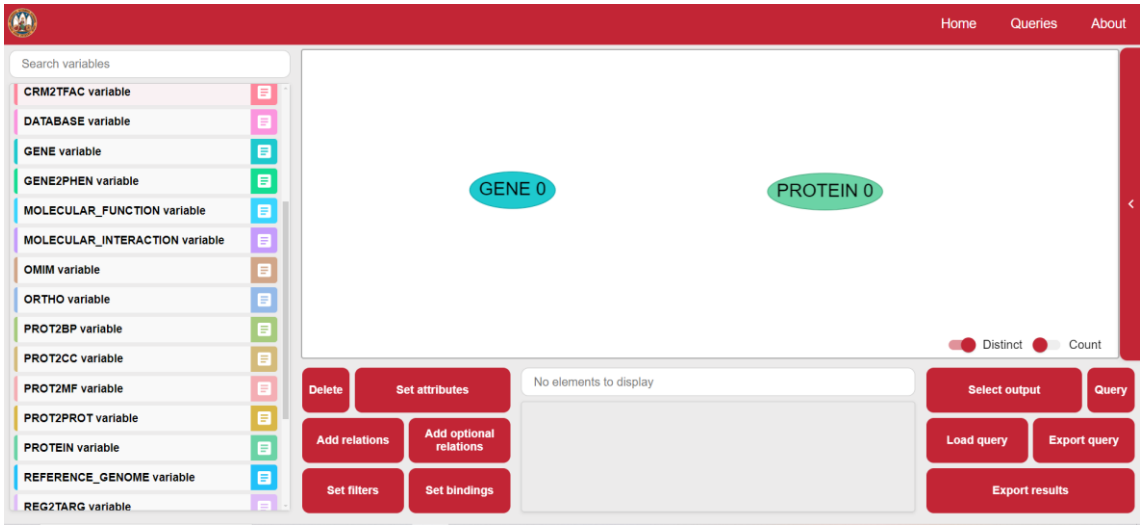


The generated SPARQL query can also be found by accessing the Console.

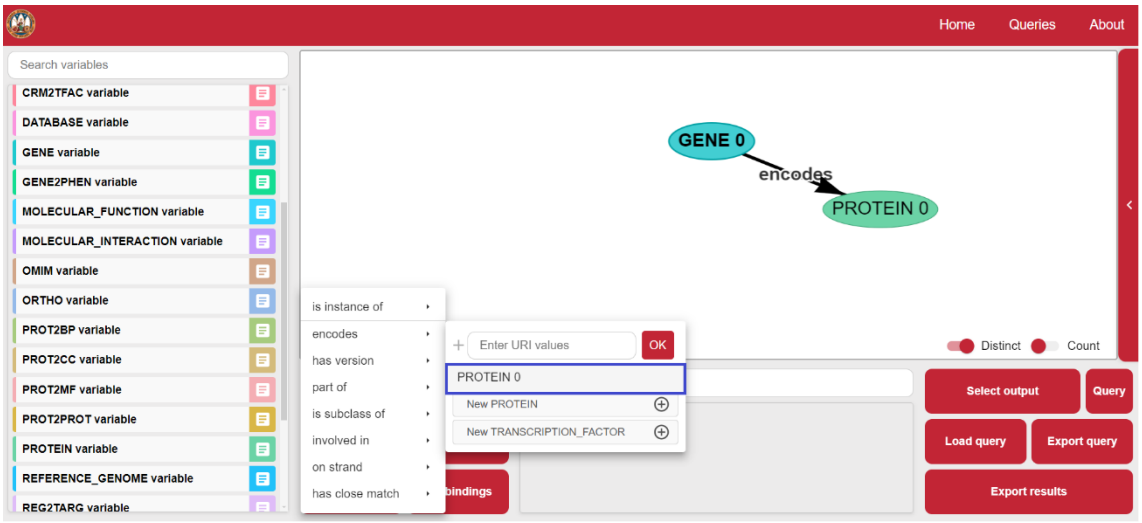


Note: Links between entities can also be established by first introducing the two nodes of interest and then the relation between them. Following the previous example:

1.



2.



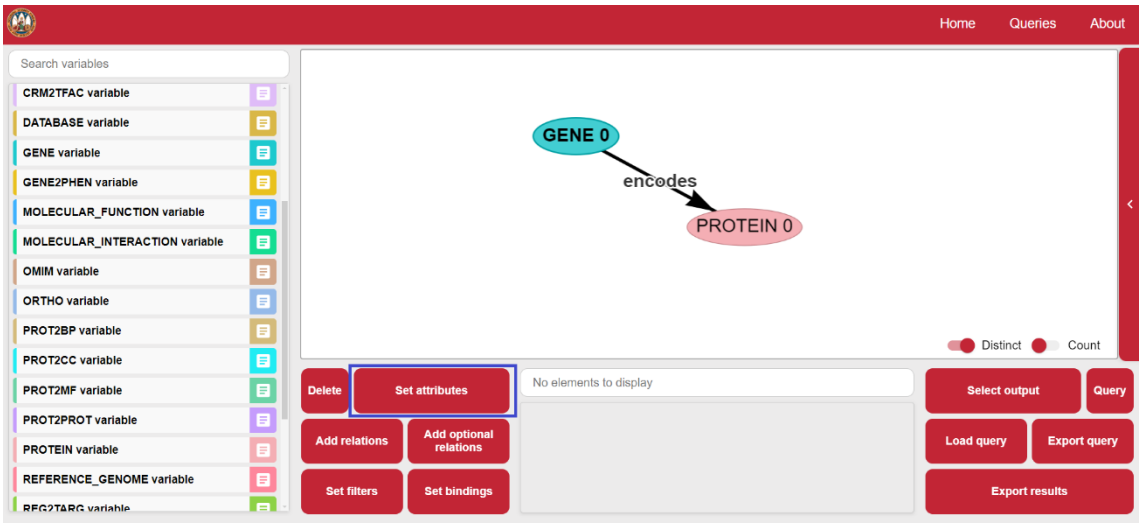
5. Data filtering and other possible operations

5.1. Filtering entities by their relations and attributes

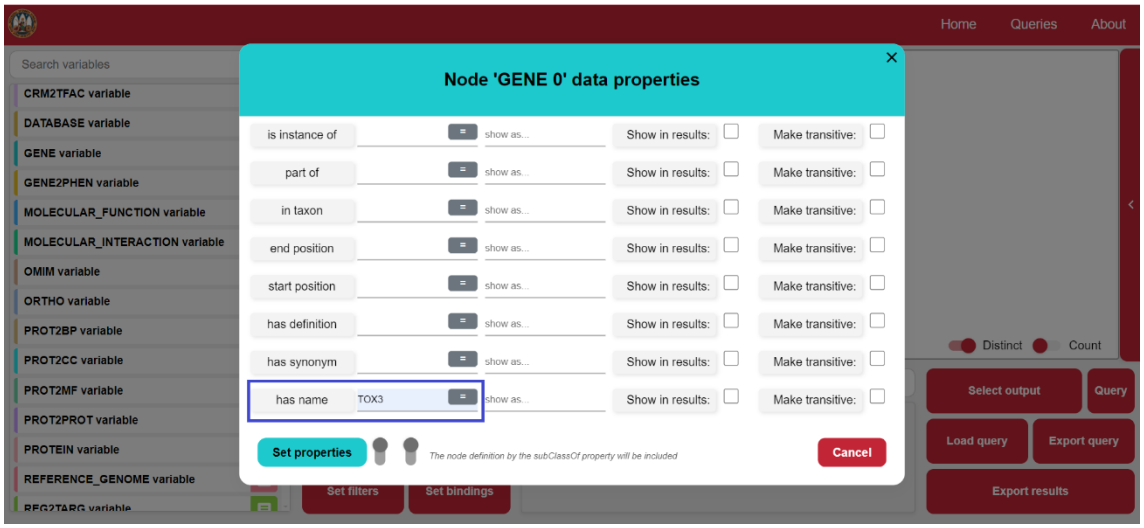
Linking two biological entities (or variables) by their relation (properties) is the simplest way to create a search pattern. A search pattern selects the desired information from the knowledge network. However, any biological entity can also be selected by its characteristics or attributes.

For example, genes can be selected by their names. Below we illustrate a use case that extends the previous query to: *Which proteins are encoded by the TOX3 gene?* To do this, we include the name of the gene in the attributes of the 'Gene' node (click on the corresponding node and then on the 'Set attributes' button).

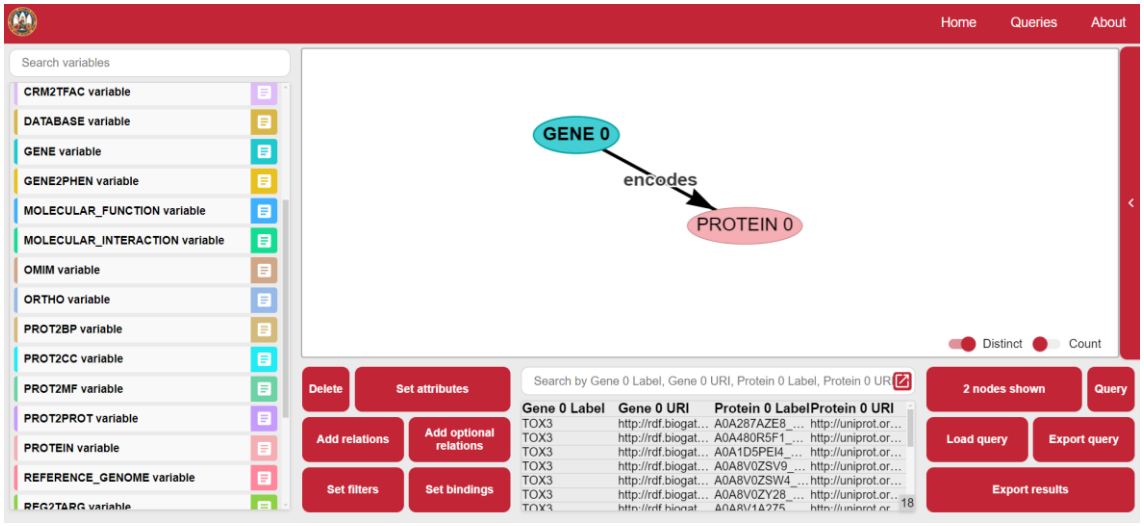
1.



2.



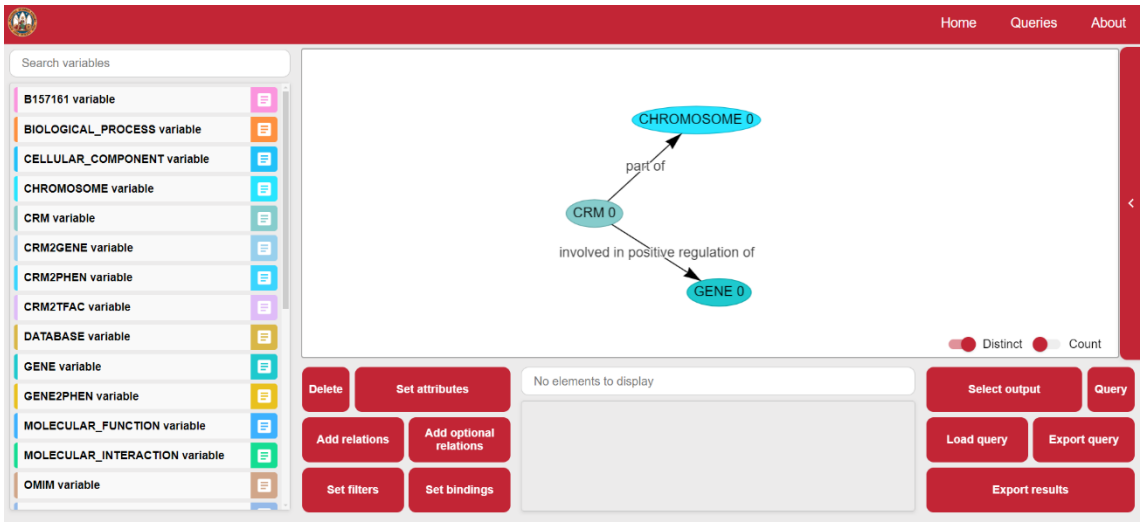
3.



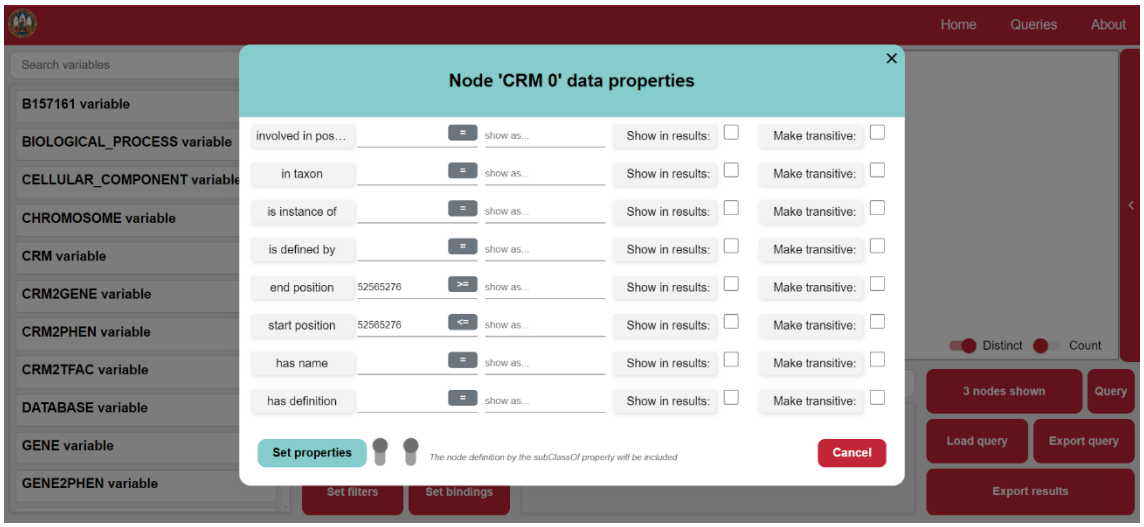
By defining the desired characteristics of biological entities (by clicking on "Set attributes") we can select entities based on these characteristics (attributes). If the character is defined as "string" composed of letters and/or numbers we can use the operator '=' to find only exact strings, or the operator '⊆' to find substrings contained in a larger string. If the character is only numeric, we can find results equal to, larger, or smaller than, using the operators '=', '>', '≥', '<', '≤'. To change the operators just click on the default operator.

For example, we can query: *Which genes are regulated by enhancers that overlap with the chr16:52565276 mutation?* i.e. CRM sequences that positively regulate gene expression. To do this, we create the relations: <CRM> <part of> <Chromosome>, and <CRM> <involved in positive regulation of> <Gene>. Then we add attributes to the Chromosome (chromosome name) and CRM (sequence coordinates) nodes. Then we select the data output (Genes) and run the query.

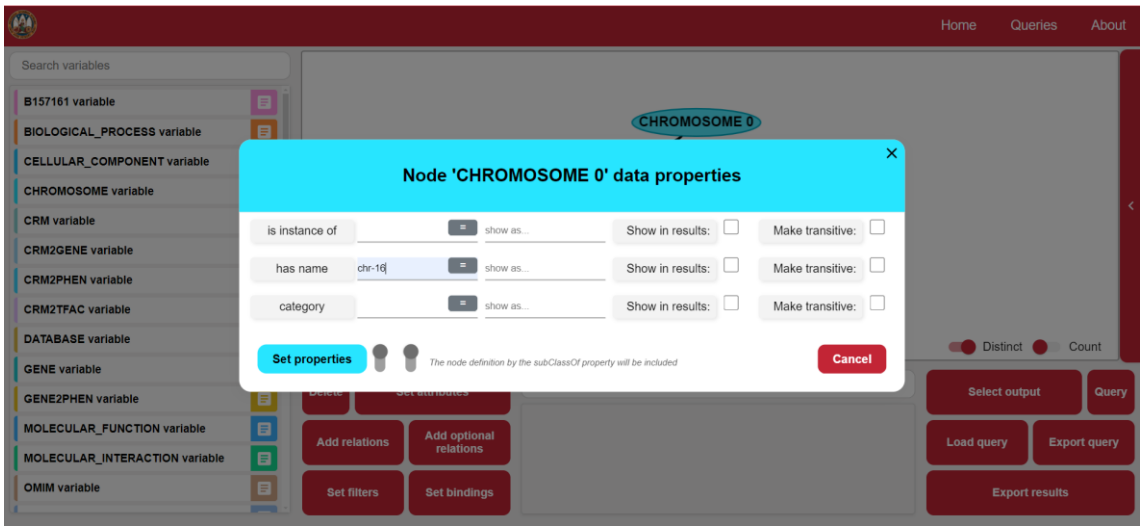
1.


Home
Queries
About
Search variables
B157161 variable
BIOLOGICAL_PROCESS variable
CELLULAR_COMPONENT variable
CHROMOSOME variable
CRM variable
CRM2GENE variable
CRM2PHEN variable
CRM2TFAC variable
DATABASE variable
GENE variable
GENE2PHEN variable
MOLECULAR_FUNCTION variable
MOLECULAR_INTERACTION variable
OMIM variable
CHROMOSOME 0
part of
CRM 0
involved in positive regulation of
GENE 0
Distinct
Count
Delete
Set attributes
No elements to display
Select output
Query
Add relations
Add optional relations
Load query
Export query
Set filters
Set bindings
Export results

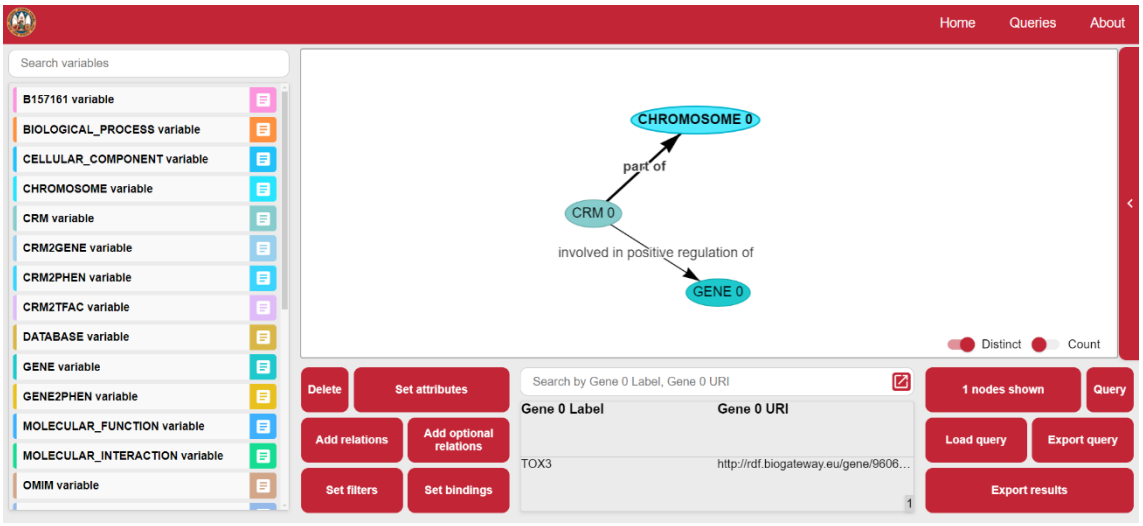
2.


Home
Queries
About
Search variables
B157161 variable
BIOLOGICAL_PROCESS variable
CELLULAR_COMPONENT variable
CHROMOSOME variable
CRM variable
CRM2GENE variable
CRM2PHEN variable
CRM2TFAC variable
DATABASE variable
GENE variable
GENE2PHEN variable
Node 'CRM 0' data properties
involved in pos…
in taxon
is instance of
is defined by
end position
52565276
start position
52565276
has name
has definition
show as...
Show in results:
Make transitive:
Set properties
The node definition by the subClassOf property will be included
Cancel
Distinct
Count
3 nodes shown
Query
Load query
Export query
Export results
Set filters
Set bindings

3.


Home
Queries
About
Search variables
B157161 variable
BIOLOGICAL_PROCESS variable
CELLULAR_COMPONENT variable
CHROMOSOME variable
CRM variable
CRM2GENE variable
CRM2PHEN variable
CRM2TFAC variable
DATABASE variable
GENE variable
GENE2PHEN variable
MOLECULAR_FUNCTION variable
MOLECULAR_INTERACTION variable
OMIM variable
CHROMOSOME 0
Node 'CHROMOSOME 0' data properties
is instance of
has name
chr-16
category
show as...
Show in results:
Make transitive:
Set properties
The node definition by the subClassOf property will be included
Cancel
Distinct
Count
Select output
Query
Load query
Export query
Add relations
Add optional relations
Export results
Set filters
Set bindings

4.



Variables can also be filtered clicking on “Set filters” button.

Some additional examples are given below:

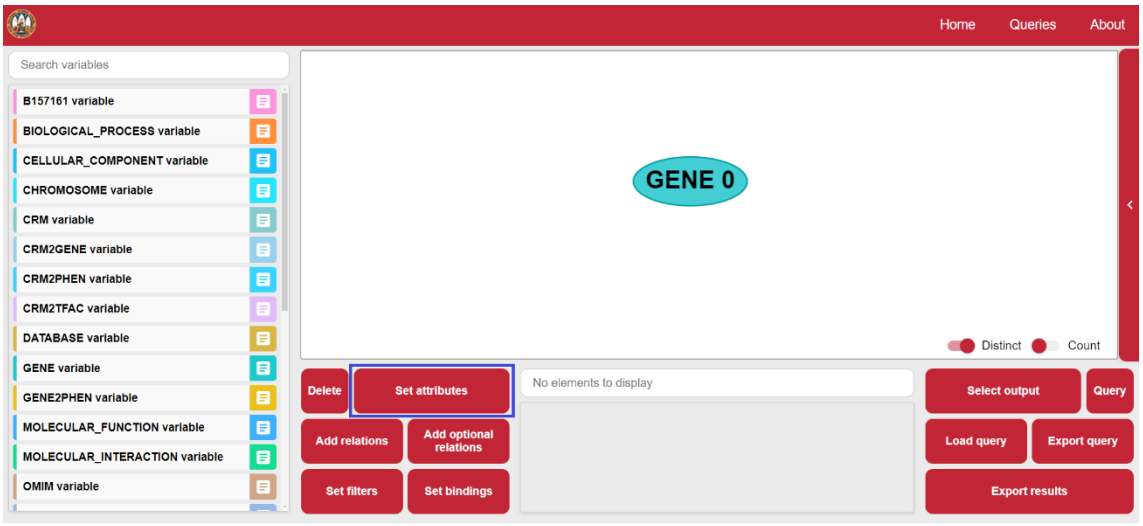
- **Example 1**: Filtering by taxon.

The resources in RDF are represented by Uniform Resource Identifiers (URIs), so these must be used when filtering a resource. This is the case for taxon (*in taxon* property, in "Set attributes"). Because URIs can be tedious to work with, e.g. “http://purl.obolibrary.org/obo/NCBITaxon_9606”, the "content in" or "⊆" operator makes it easier to work with identifiers only. Below we include a table with the taxonomic IDs of the most relevant species included in BioGateway:

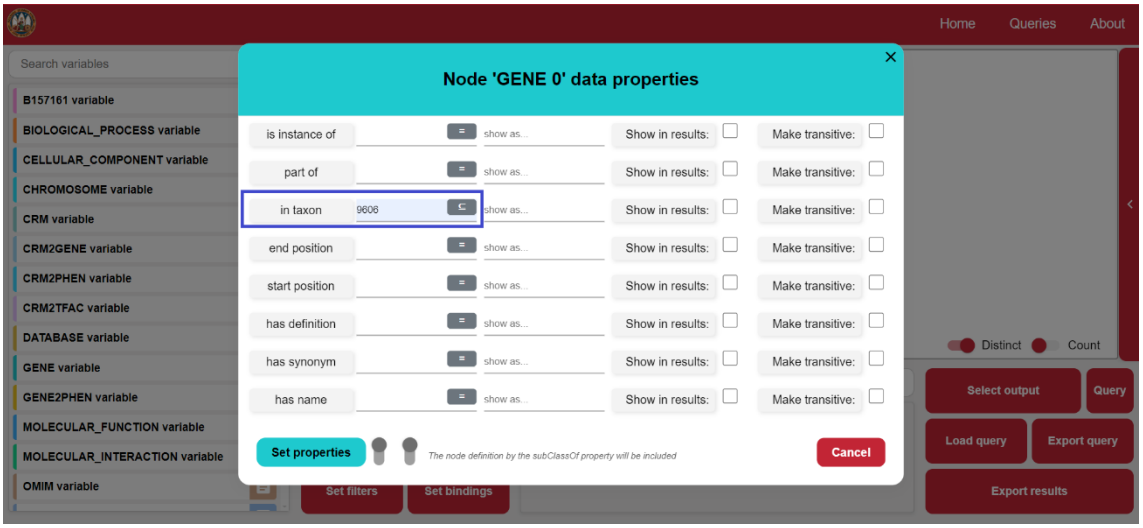
label	**Taxon ID**	**URI_taxon**
Mus musculus	10090	http://purl.obolibrary.org/obo/NCBITaxon_10090
Arabidopsis thaliana	3702	http://purl.obolibrary.org/obo/NCBITaxon_3702
Oryza sativa Japonica Group	39947	http://purl.obolibrary.org/obo/NCBITaxon_39947
Dictyostelium discoideum	44689	http://purl.obolibrary.org/obo/NCBITaxon_44689
Zea mays	4577	http://purl.obolibrary.org/obo/NCBITaxon_4577
Caenorhabditis elegans	6239	http://purl.obolibrary.org/obo/NCBITaxon_6239
Danio rerio	7955	http://purl.obolibrary.org/obo/NCBITaxon_7955
Gallus gallus	9031	http://purl.obolibrary.org/obo/NCBITaxon_9031
Sus scrofa	9823	http://purl.obolibrary.org/obo/NCBITaxon_9823
Bos taurus	9913	http://purl.obolibrary.org/obo/NCBITaxon_9913
Homo sapiens	9606	http://purl.obolibrary.org/obo/NCBITaxon_9606
Drosophila melanogaster	7227	http://purl.obolibrary.org/obo/NCBITaxon_7227
Oryctolagus cuniculus	9986	http://purl.obolibrary.org/obo/NCBITaxon_9986
Rattus norvegicus	10116	http://purl.obolibrary.org/obo/NCBITaxon_10116
Saccharomyces cerevisiae S288C	559292	http://purl.obolibrary.org/obo/NCBITaxon_559292
Schizosaccharomyces pombe 972h-	284812	http://purl.obolibrary.org/obo/NCBITaxon_284812
Chlamydomonas reinhardtii	3055	http://purl.obolibrary.org/obo/NCBITaxon_3055
Plasmodium falciparum 3D7	36329	http://purl.obolibrary.org/obo/NCBITaxon_36329
Neurospora crassa OR74A	367110	http://purl.obolibrary.org/obo/NCBITaxon_367110
Canis lupus familiaris	9615	http://purl.obolibrary.org/obo/NCBITaxon_9615

The following example illustrates the filtering of human genes using the taxon ID (9606):
What human genes does the network contain?

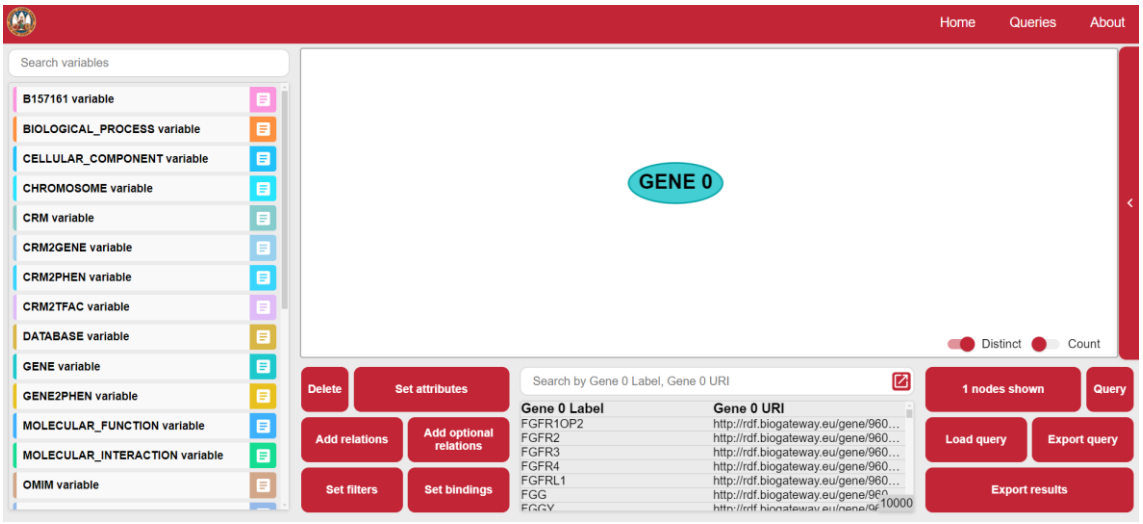
1.



2.



3.

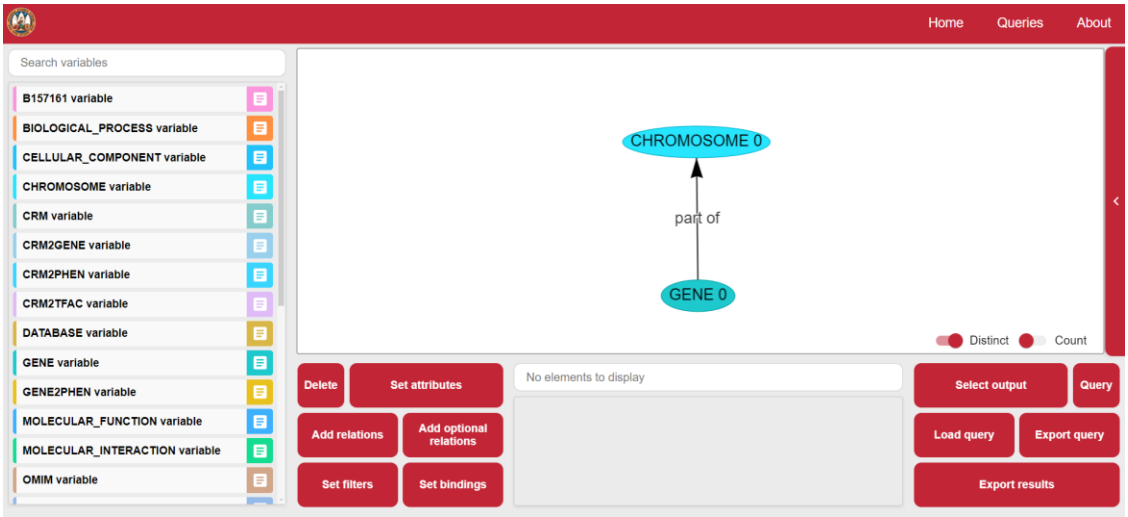


- **Example 2**: Filtering by chromosome.

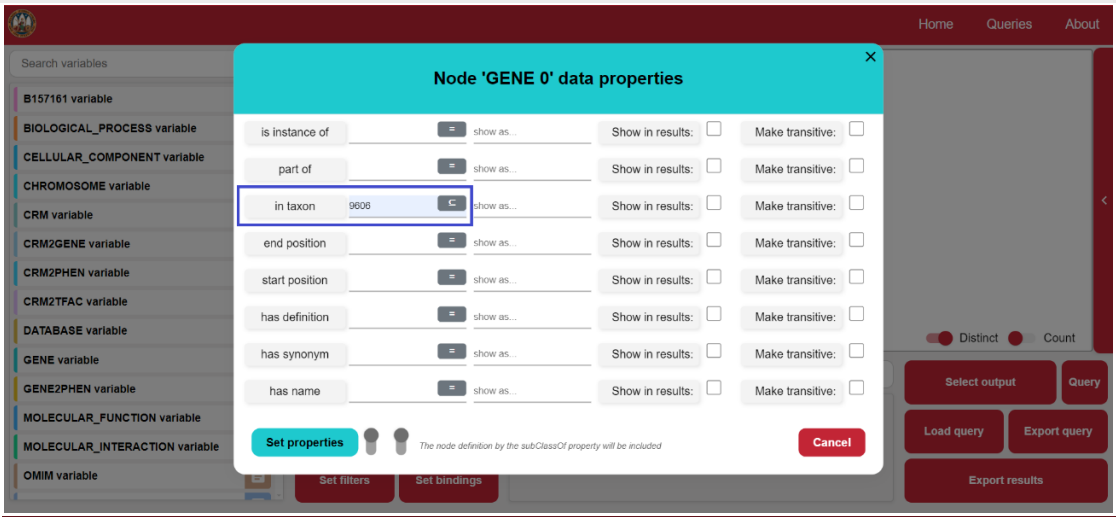
Chromosomes are entities available as variables, and are resources that have labels. Therefore, chromosomes can be filtered through their labels. Strings can be filtered in INTUITION using the "=" (exact value) or "⊆" (contained in) operators. The default configuration of string filtering is not case-sensitive.

The following example filters human genes on chromosome 1 (*What human genes are located on chr-1?*).

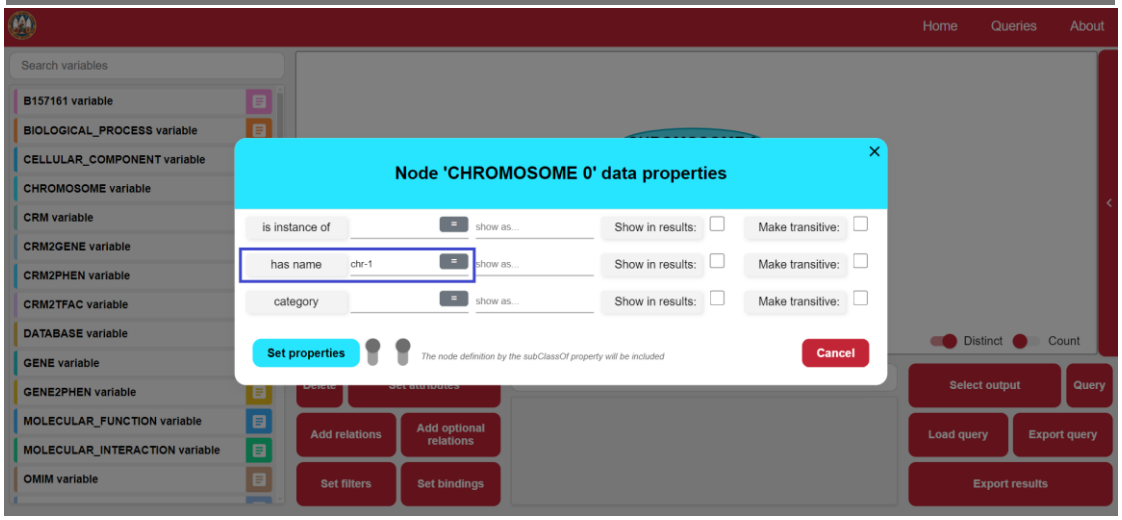
1.



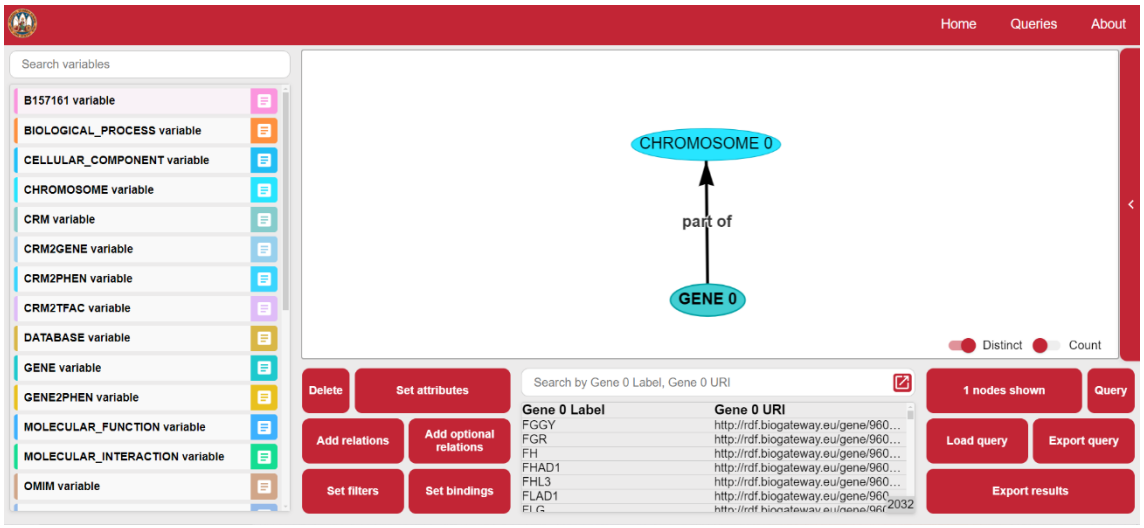
2.



3.



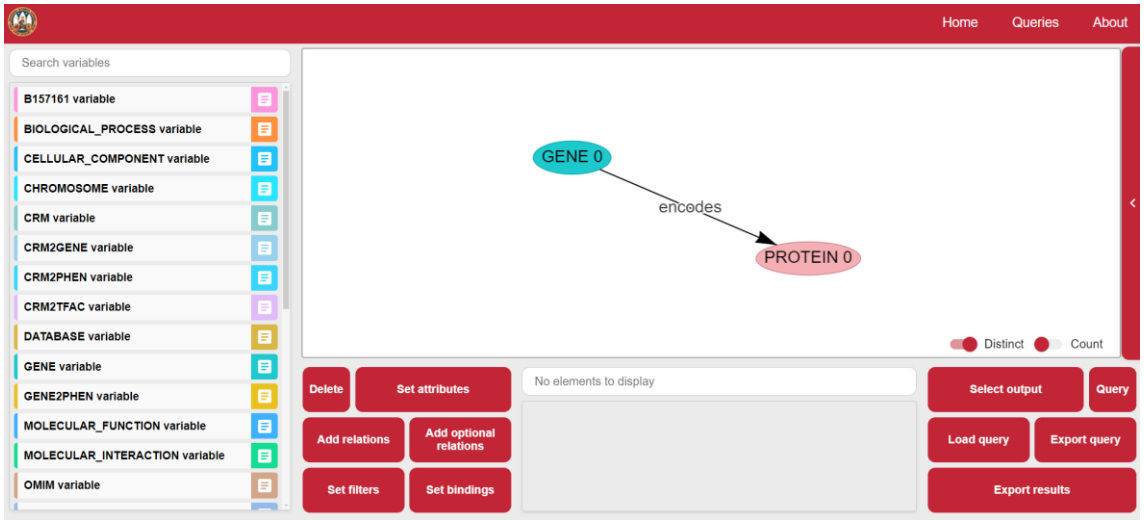
4.



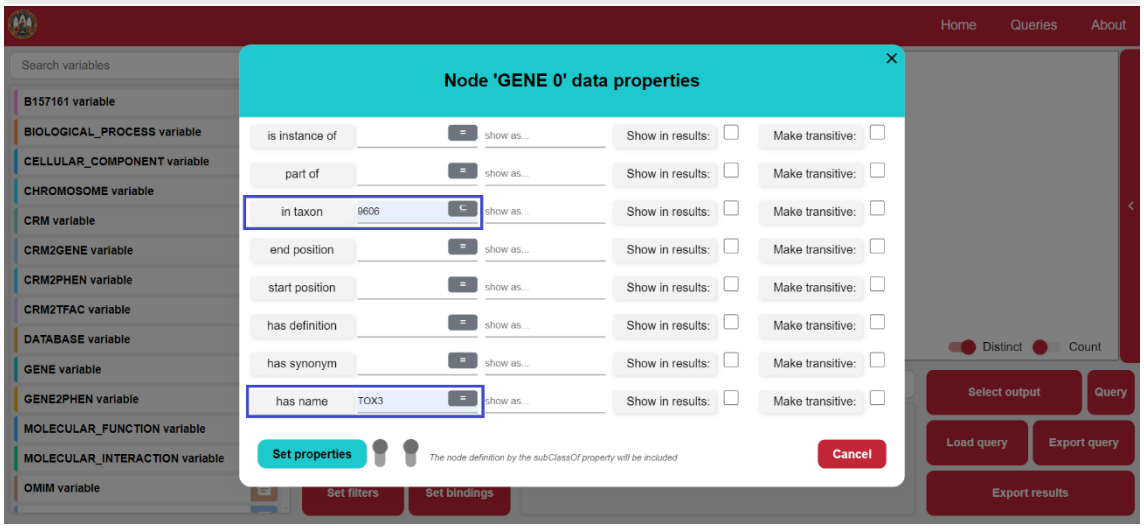
- **Example 3**: Filtering by name.

For this example we illustrate the query building to obtain the proteins encoded by the human TOX3 gene: *What proteins are encoded by the human TOX3 gene?* To do this, after including the link <Gene> <encodes> <Protein>, we modify the attributes of Gene node to indicate the name (TOX3) and the human taxon (9606).

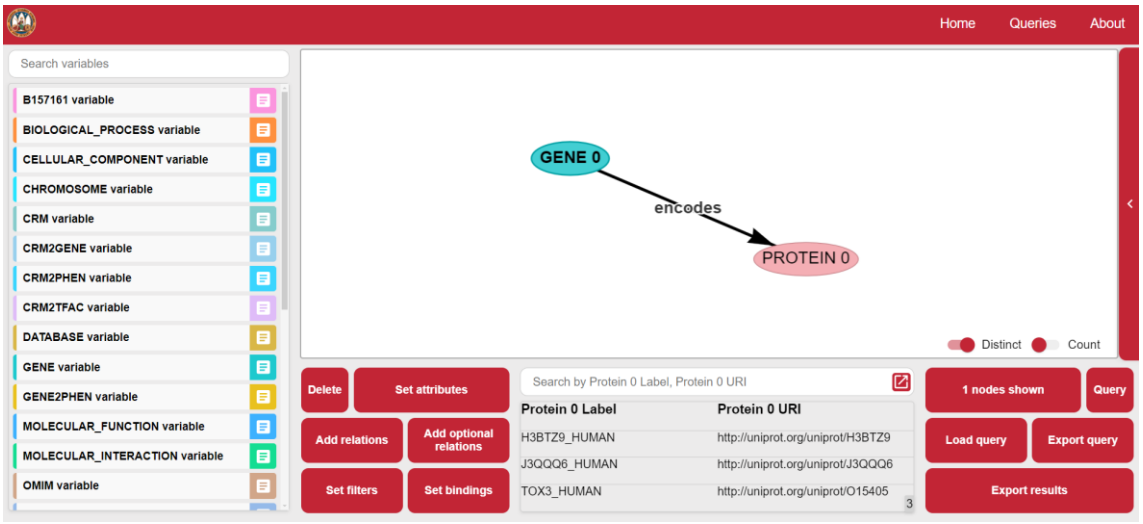
1.



2.



3.

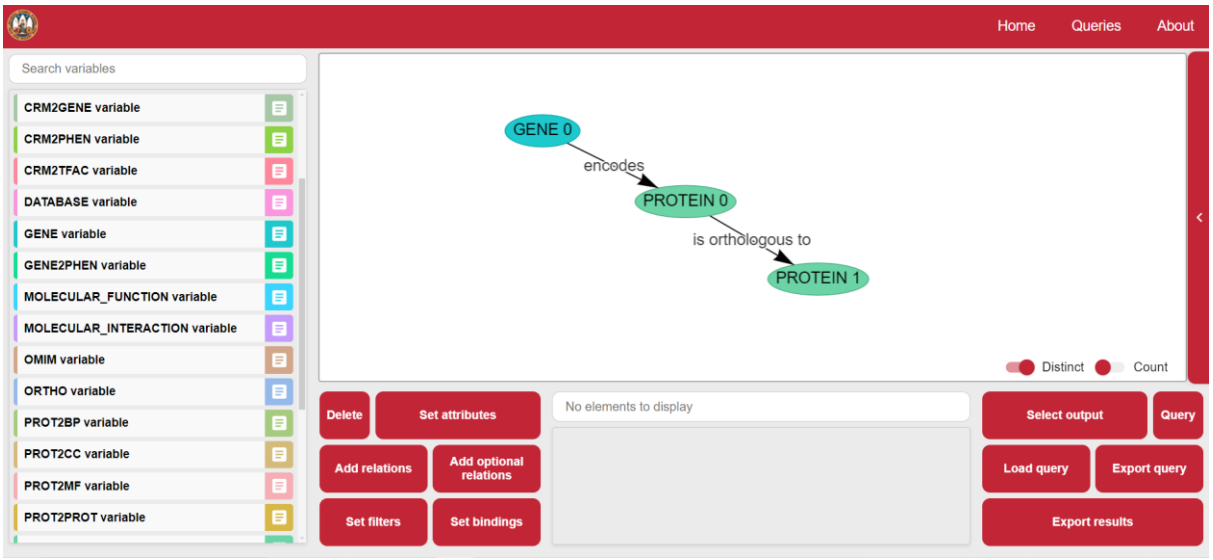


5.2. Count and display unique results

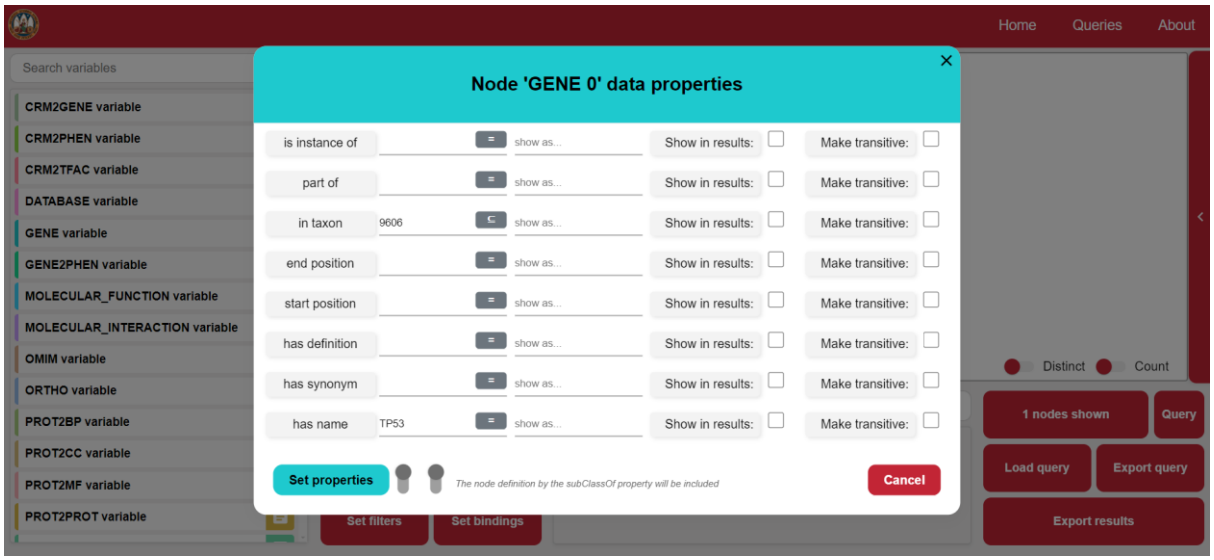
The output table shows the biological entities that meet the biological selection criteria. Since a user can design complex query patterns and has the freedom to choose which entities they want to include in the output ("Select output" button), duplicate entities might appear in the result. For this reason, the "Distinct" button is activated for automatic filtering.

For example, we can query: *What are the orthologous proteins of the human TP53 gene?* To do this, we generate the relations <Gene> <encodes> <Protein>, and <Protein> <is orthologous to> <Protein>. Then we modify Gene's attributes to indicate the name and taxon.

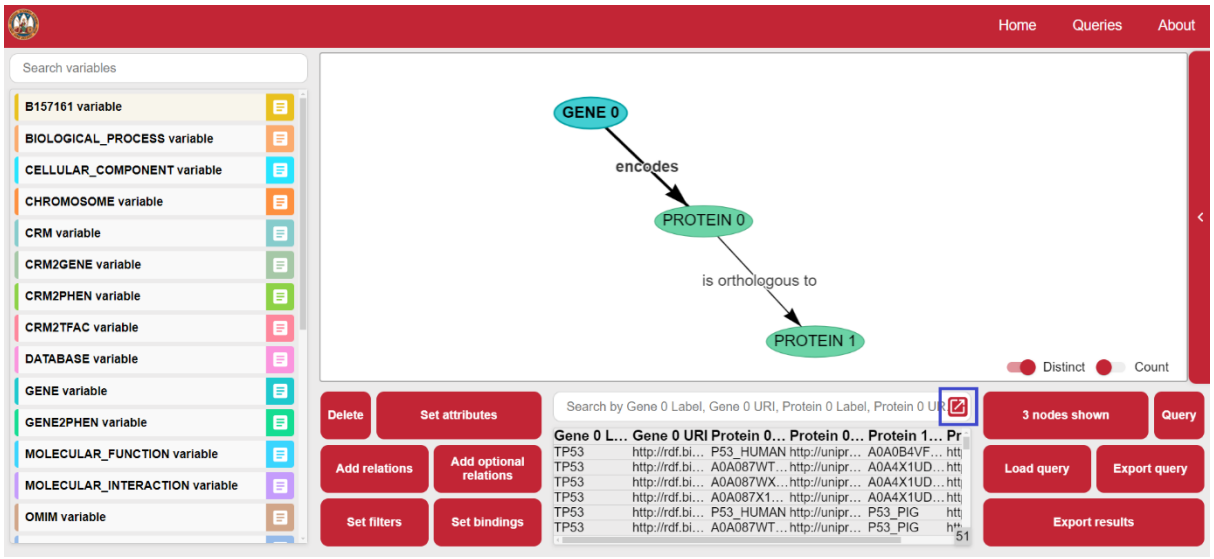
1.



2.



3.



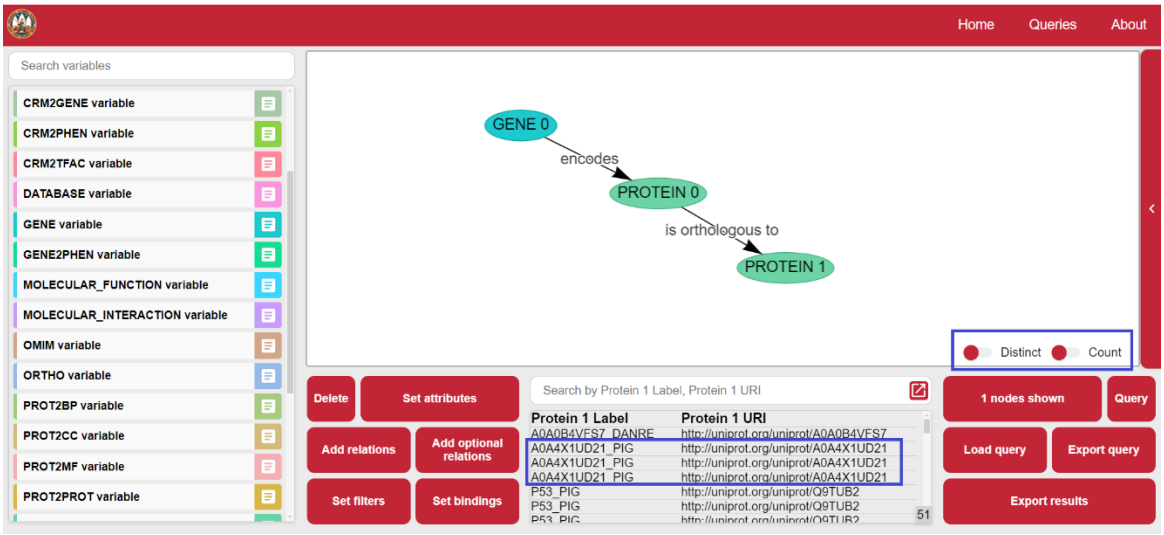
If we include all the entities in the output, and examine the extended table of results, we can see the relation between the human TP53 gene, its protein products and orthologous proteins in other organisms.

4.

Gene 0 Label	Gene 0 URI	Protein 0 Label	Protein 0 URI	Protein 1 Label	Protein 1 URI
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	A0A0B4VFS7_DANRE	http://uniprot.org/uniprot/A0A0B4VFS7
TP53	http://rdf.biogateway.eu/gene/9606/TP53	A0A087WT22_HUMAN	http://uniprot.org/uniprot/A0A087WT22	A0A4X1UD21_PIG	http://uniprot.org/uniprot/A0A4X1UD21
TP53	http://rdf.biogateway.eu/gene/9606/TP53	A0A087WXZ1_HUMAN	http://uniprot.org/uniprot/A0A087WXZ1	A0A4X1UD21_PIG	http://uniprot.org/uniprot/A0A4X1UD21
TP53	http://rdf.biogateway.eu/gene/9606/TP53	A0A087X1Q1_HUMAN	http://uniprot.org/uniprot/A0A087X1Q1	A0A4X1UD21_PIG	http://uniprot.org/uniprot/A0A4X1UD21
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	P53_PIG	http://uniprot.org/uniprot/Q9TUB2
TP53	http://rdf.biogateway.eu/gene/9606/TP53	A0A087WT22_HUMAN	http://uniprot.org/uniprot/A0A087WT22	P53_PIG	http://uniprot.org/uniprot/Q9TUB2
TP53	http://rdf.biogateway.eu/gene/9606/TP53	A0A087WXZ1_HUMAN	http://uniprot.org/uniprot/A0A087WXZ1	P53_PIG	http://uniprot.org/uniprot/Q9TUB2
TP53	http://rdf.biogateway.eu/gene/9606/TP53	A0A087X1Q1_HUMAN	http://uniprot.org/uniprot/A0A087X1Q1	P53_PIG	http://uniprot.org/uniprot/Q9TUB2
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	A0A0B4K7P1_DROME	http://uniprot.org/uniprot/A0A0B4K7P1
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	B0R0M3_DANRE	http://uniprot.org/uniprot/B0R0M3
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	B0S576_DANRE	http://uniprot.org/uniprot/B0S576
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	B0S577_DANRE	http://uniprot.org/uniprot/B0S577
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	G1K2L5_DANRE	http://uniprot.org/uniprot/G1K2L5
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	A0A167VDT2_CHICK	http://uniprot.org/uniprot/A0A167VDT2
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	F1P1U2_CHICK	http://uniprot.org/uniprot/F1P1U2
TP53	http://rdf.biogateway.eu/gene/9606/TP53	P53_HUMAN	http://uniprot.org/uniprot/P04637	D4AA88_RAT	http://uniprot.org/uniprot/D4AA88

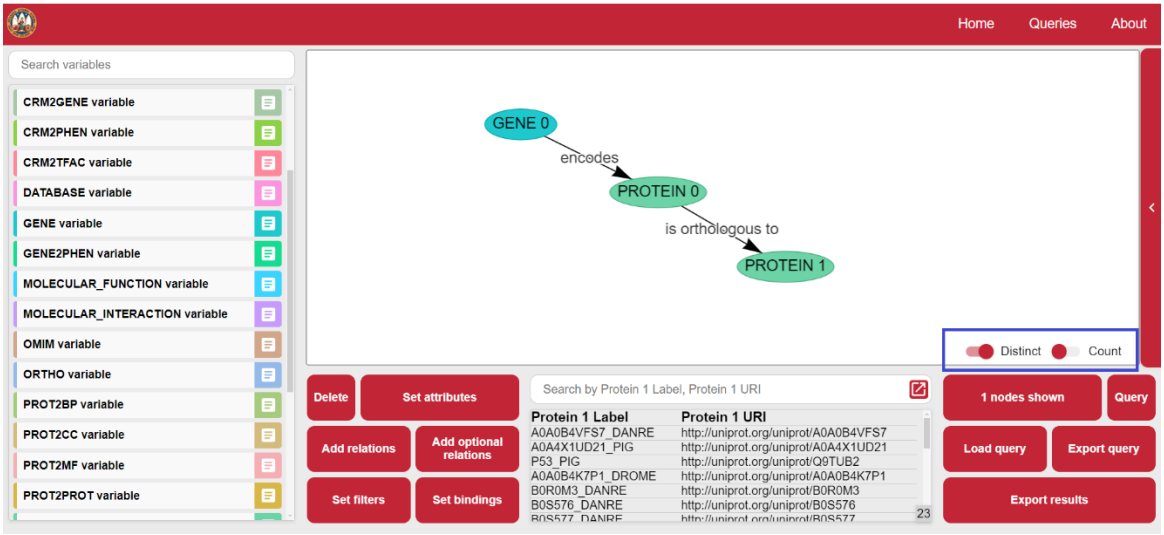
The table contains a total of 51 entries/rows, corresponding to the results that satisfy the search pattern. However, we can see that several proteins encoded by the human TP53 gene are related to the same orthologous protein. Therefore, if we now only select as output the orthologous proteins, and we deactivate the "Distinct" button, we obtain the same 51 results that satisfy the search pattern, but with repeated values, because we are only selecting the orthologous proteins.

5.

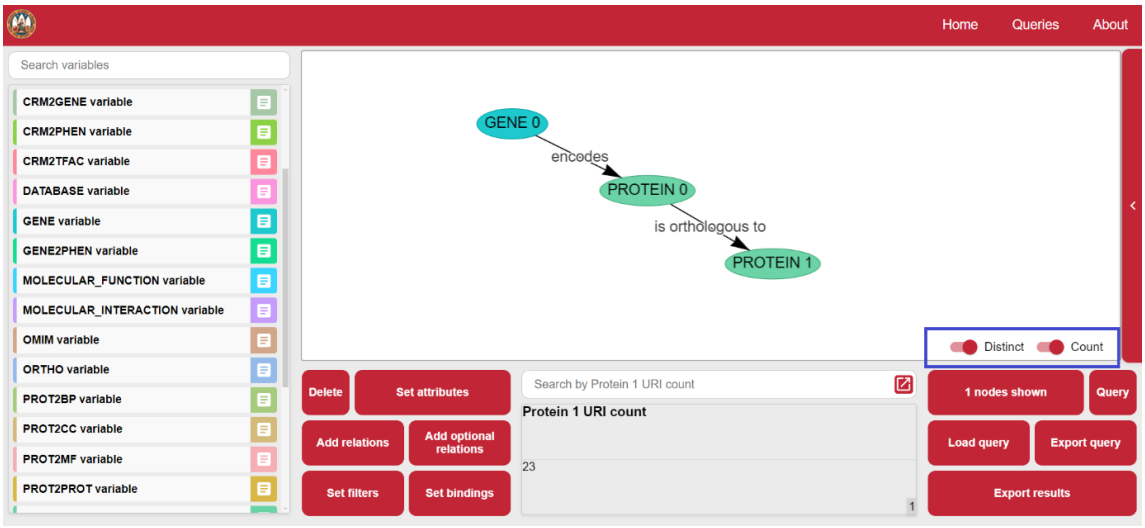


On the contrary, with the "Distinct" functionality activated (activated by default), we only obtain the unique results, which in this case are 23 (23 orthologous proteins).

6.

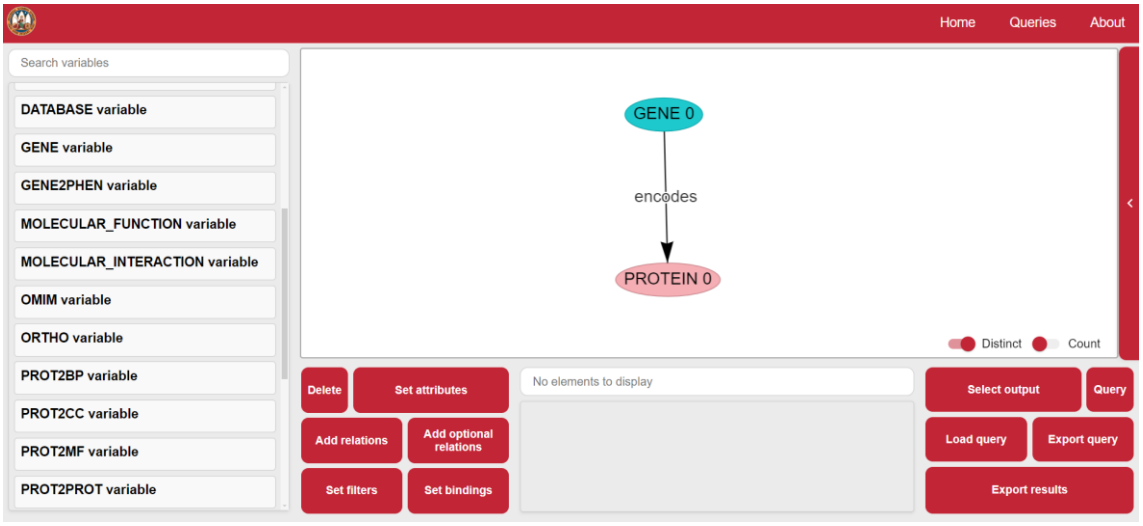


On the other hand, activating the "Count" button displays the number of entities that fit the search pattern of the query.

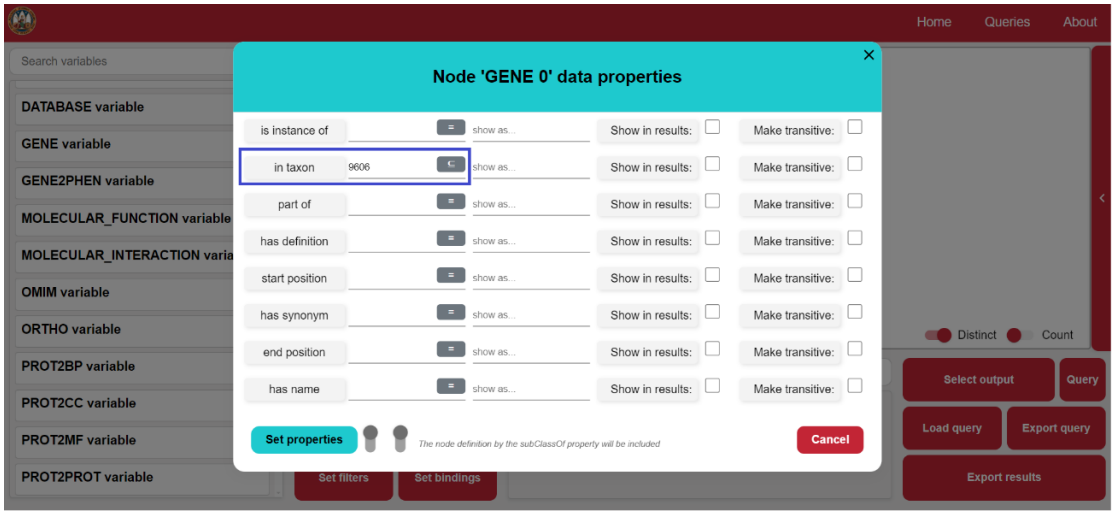


Below we include another example. We can build the query: *How many human genes encode proteins, and how many proteins are there*?

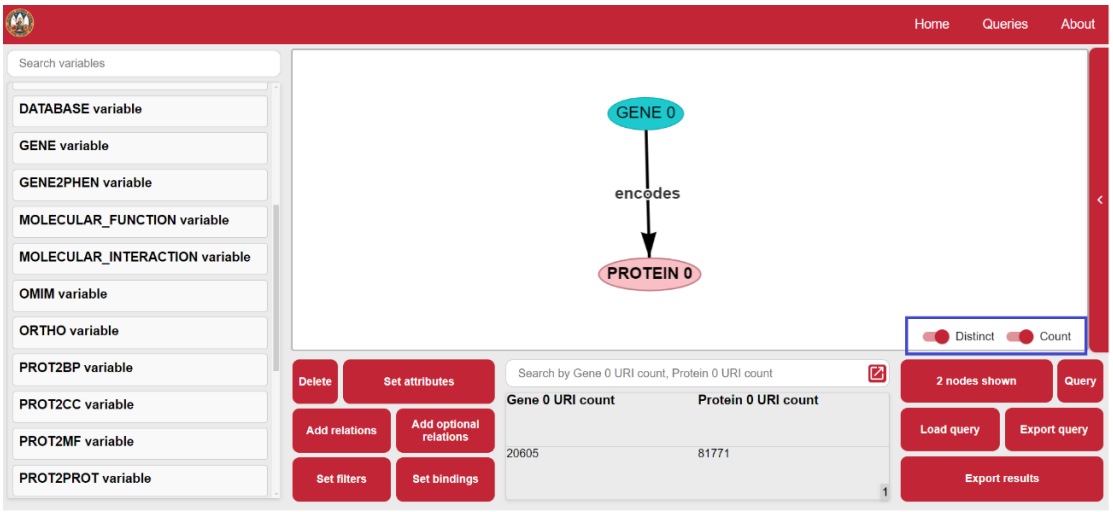
1.



2.



3.

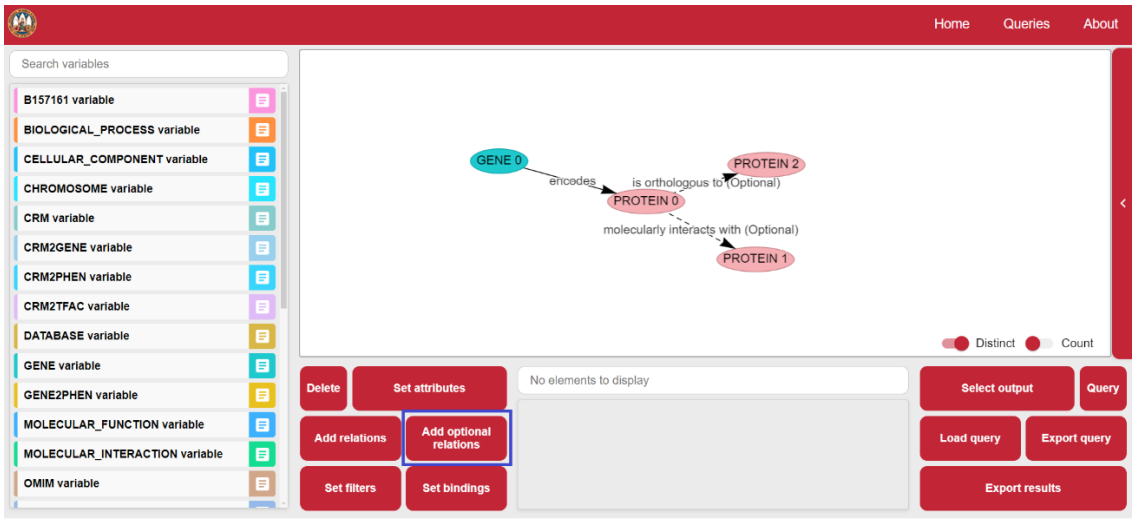


5.3. Optional relations

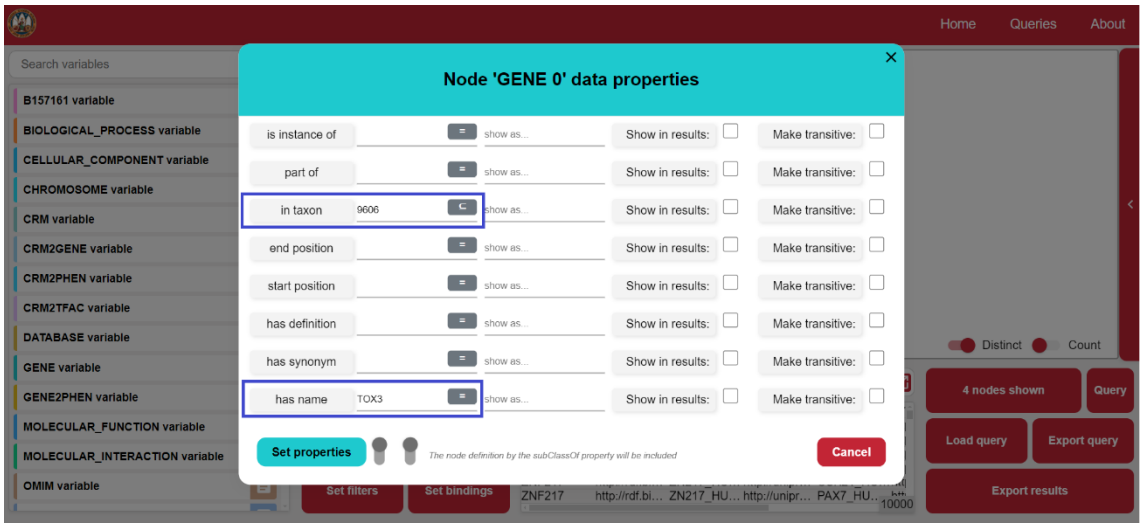
INTUITION also allows to include optional relations (“Add optional relations” button). As this is an optional pattern, the information is added if it exists, so it does not work as a filter.

In this way, INTUITION allows queries like: *What proteins are encoded by the human TOX3 gene? Do these protein products interact with any other proteins? Is there information about proteins orthologous to those encoded by the human TOX3 gene?*

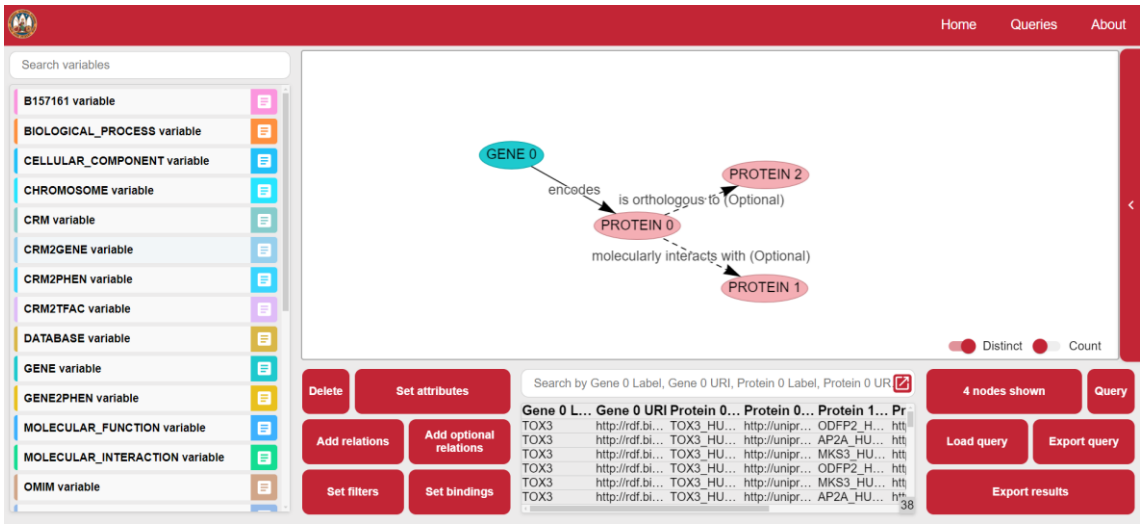
1.



2.



3.



Note: Queries can involve different variables of the same entity type. Queries may require handling the same entity as different variables, e.g. protein-protein interaction involves two different proteins, and therefore two different variables. INTUITION allows the inclusion of the same variable more than once. These nodes are numbered starting with 0. The counter is reset to zero when the application is updated, not when the node is deleted.

5.4. Multiple values

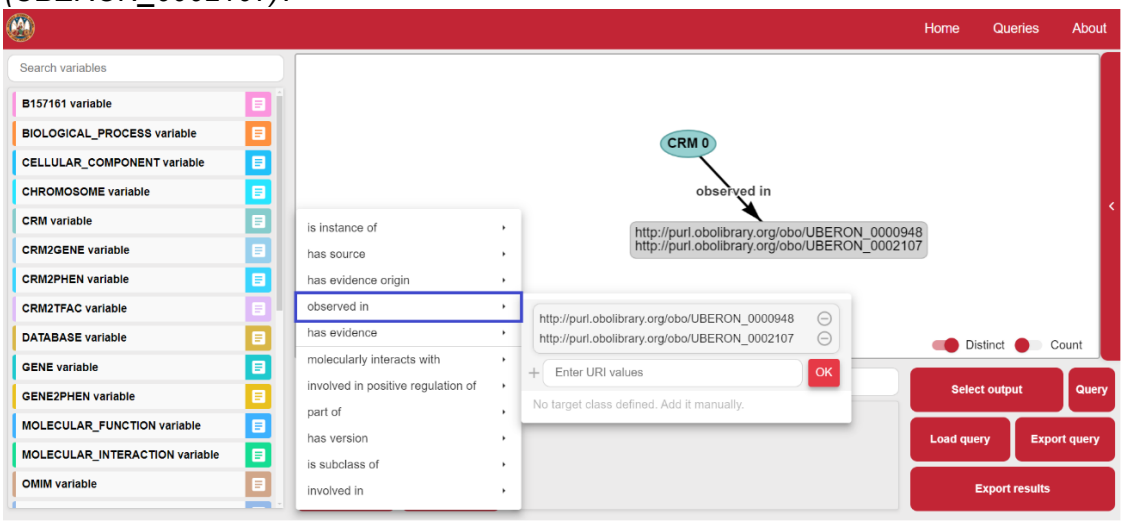
To avoid creating repetitive queries when the general structure of the query is the same, but the characteristics of the entities are different, INTUITION allows a user to assign different values to the variables.

For example, if we are interested in searching for cis-regulatory modules (CRM) identified in two or more tissues of interest, we do not need to repeat the same query for each tissue. As shown in the example (*Which CRMs have been identified in heart (UBERON_0000948) and liver (UBERON_0002107)?)*, we can specify different tissues in the "Enter URI values" cell of “observed in” property, in "Add realtions". Click on "+" to include values and click on “OK” when all values are listed. As BioGateway uses

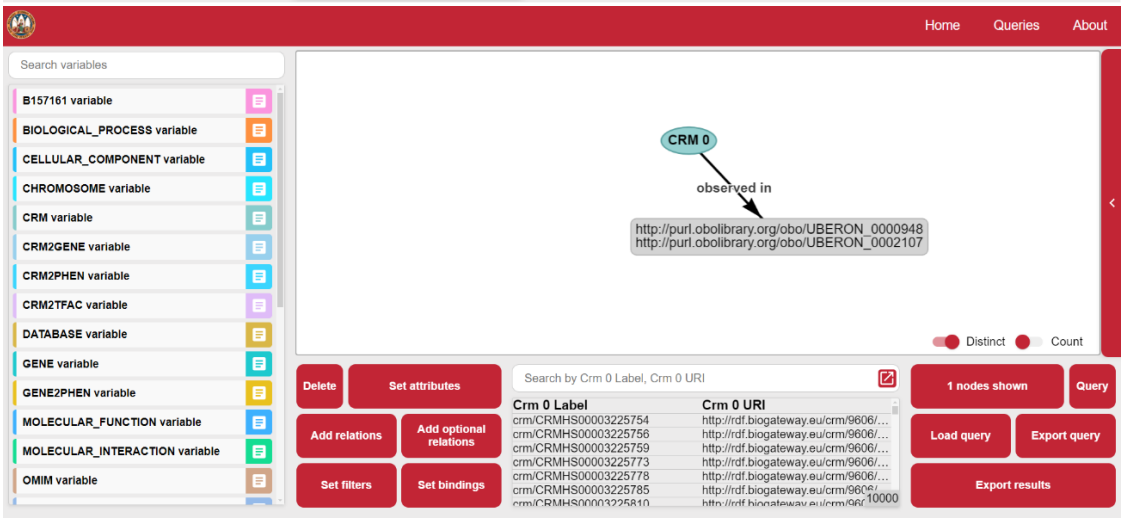
semantic resources to identify entities, the values entered must be Uniform Resource Identifiers (URIs) corresponding to these resources.

Which CRMs have been identified in heart (UBERON_0000948) and liver (UBERON_0002107)?

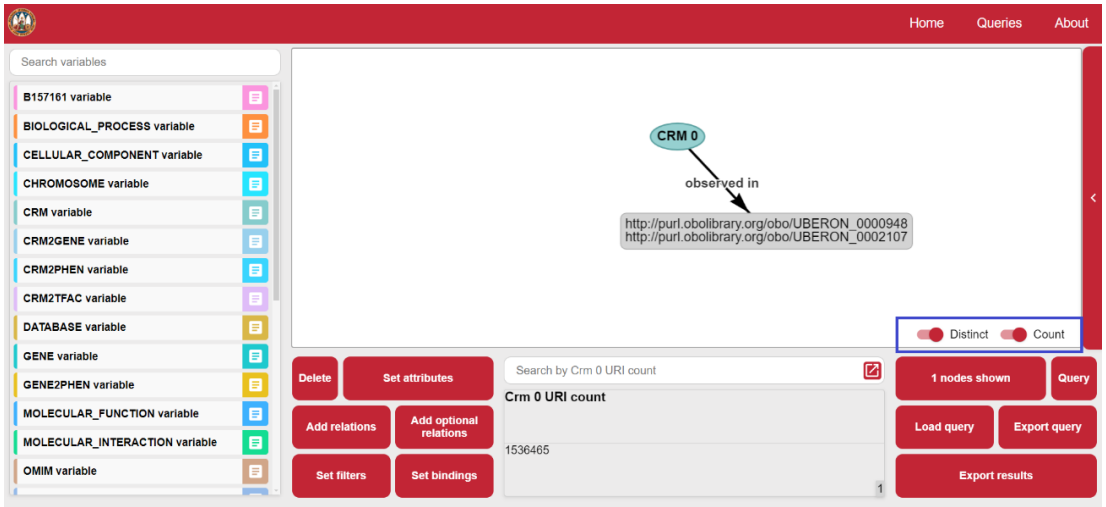
1.



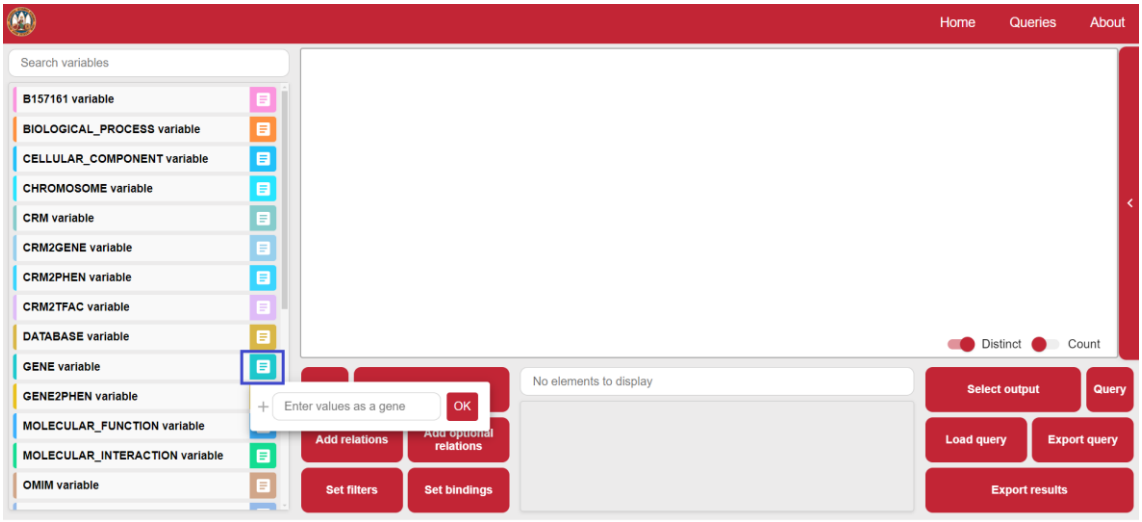
2.



If we want to count the number of CRMs, we click on the 'Count' button to activate this counting functionality: *How many CRMs have been identified in heart (UBERON_0000948) and liver (UBERON_0002107)?*

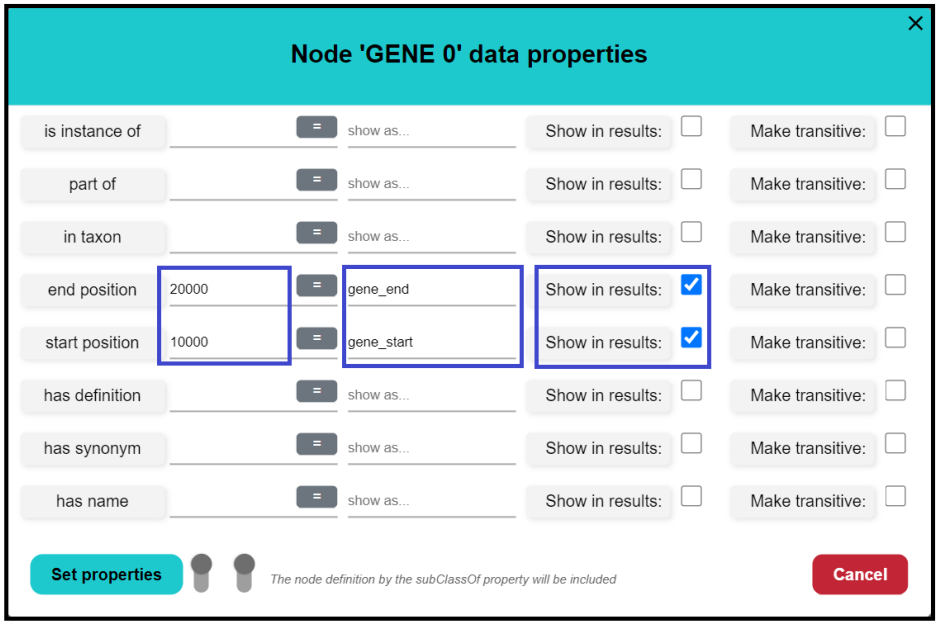


We can also add multiple values to the node that acts as the subject of the triplet:



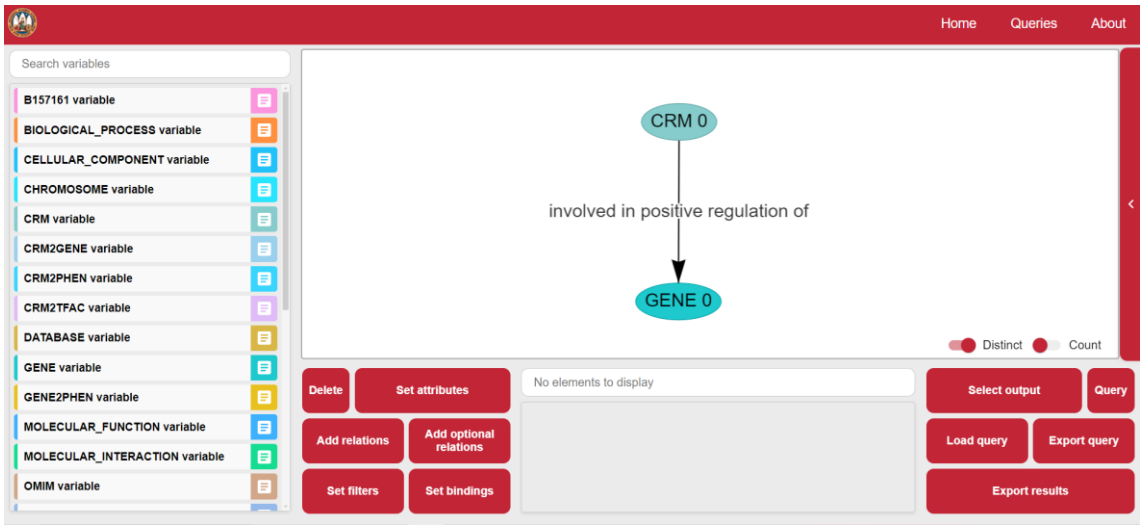
5.5. Creating and filtering variables

INTUITION allow a user to create their own selection variables. This functionality is implemented in "Set bindings" button, in the "Pattern designer", and can be applied to attributes used in the query, renamed, or selected for output. To use an attribute in the search pattern, a value must be entered to act as a filter. To rename an attribute, simply change its name in "show as". To mark it for data output, check "Shown in results".

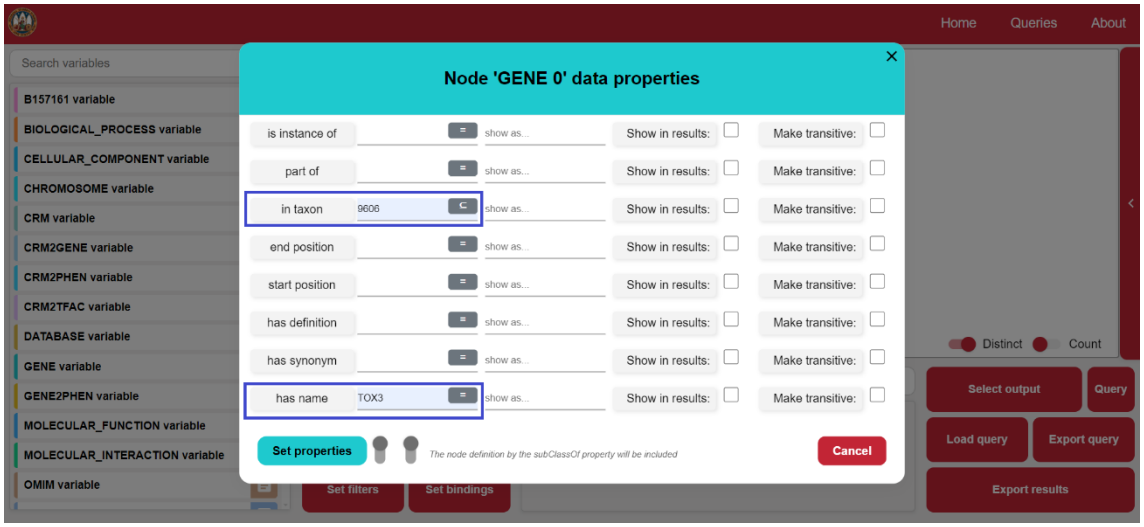


For example, by subtracting the end and start positions of the CRMs, and adding “1” to this number, we obtain the length of the sequences in a new variable. Then, we can filter this new variable in the “Set filters” button. Below we illustrate an example: *Which CRMs with a length less than or equal to 500 bp positively regulate the human TOX3 gene?* For this:

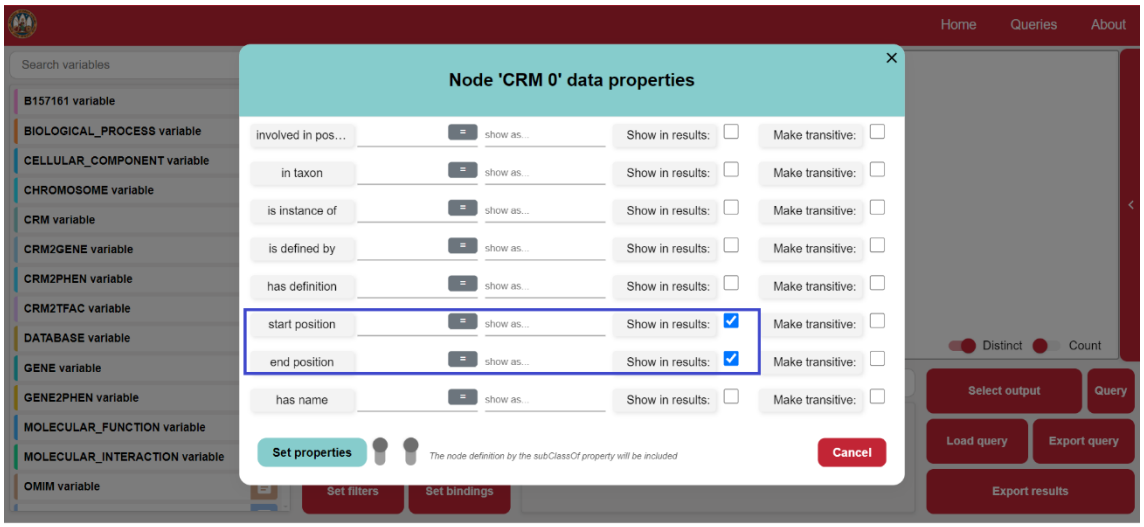
- First, we generate the relation <CRM> <involved in positive regulation of> <Gene>.



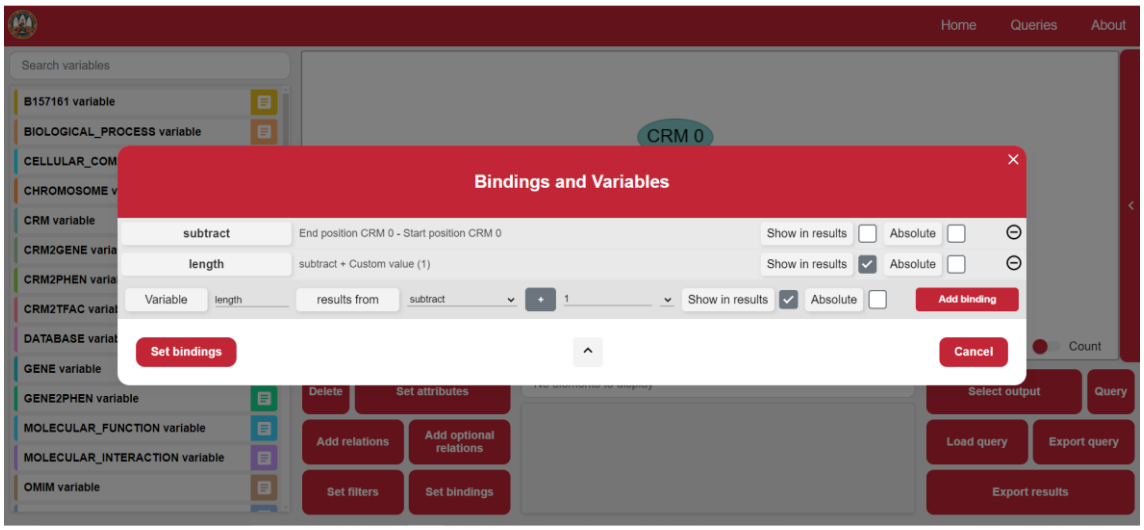
- Assign the attributes corresponding to the gene (name TOX3, and taxon).



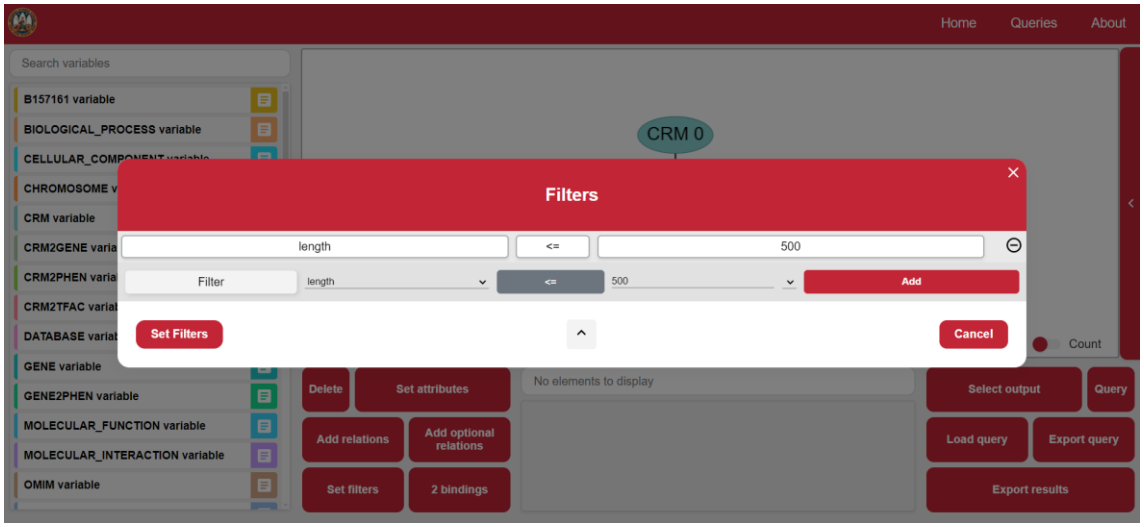
- Select the CRM attributes that we are going to use to generate the new variables.



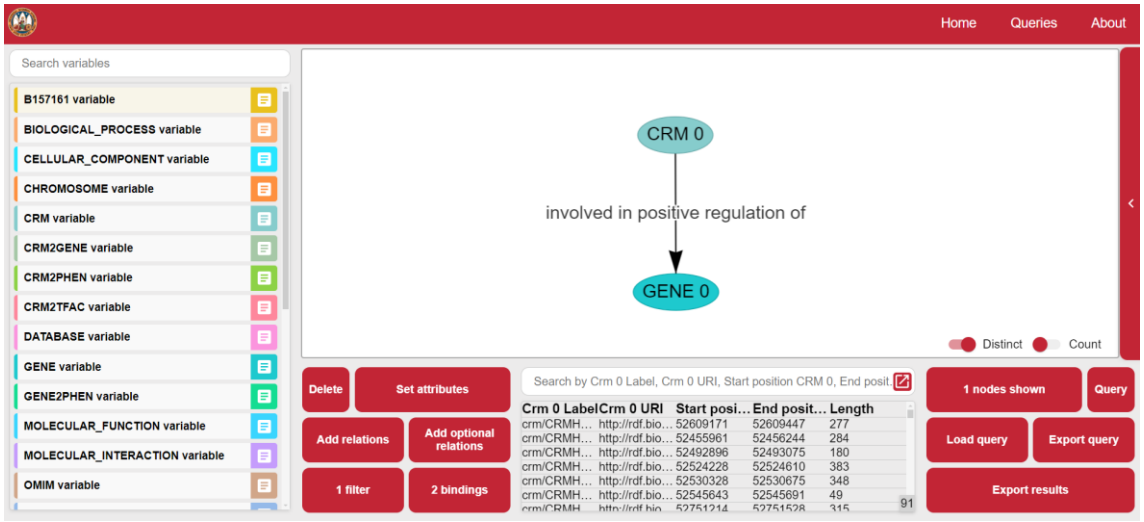
- Create the new variable in “Set bindings”.



- We filter in ‘Set filters’ the new variable.



- Select the output and run the query:



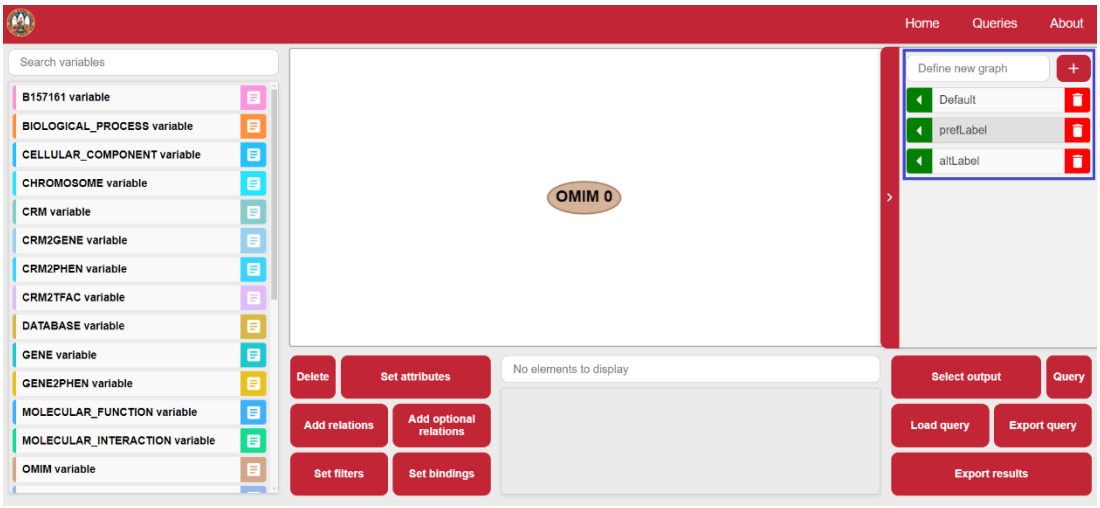
5.6. Union of queries

INTUITION also allows the use of the UNION clause of SPARQL. UNION merges subqueries through common variables in both queries. We illustrate its use through a use case.

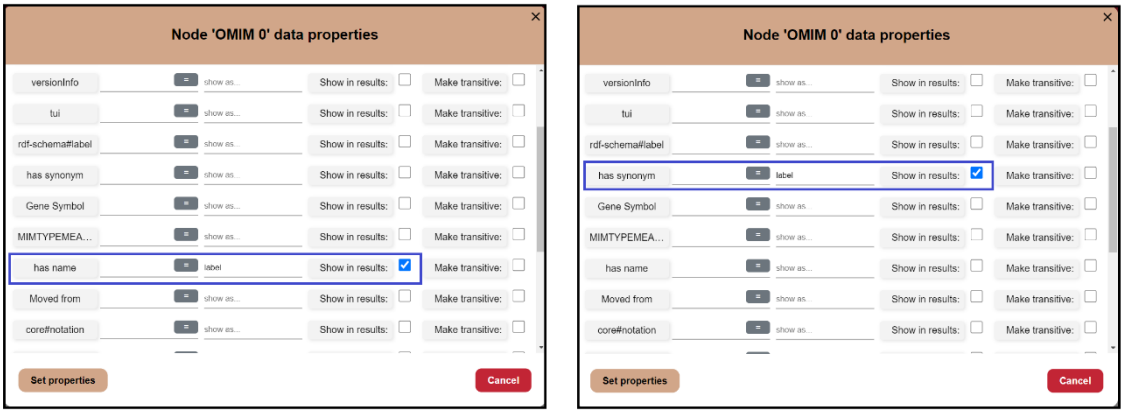
For example, we retrieve the OMIM entities that contain the string "breast cancer" as a name or synonym (*Which OMIM entities contain 'breast cancer' in their preferred label or alternative label?*), i.e. "name" and "synonym" are different attributes, but we can unite their values in a common variable using the UNION clause and a new unified rename for the attributes we want to unify ("label" in this example). To do that:

1. In the "Union builder" section, we create the graphs belonging to each of the subqueries, and we include an OMIM node in each of them. In the example, the graphs are "prefLabel" for the main label query and "altLabel" for the query of the synonym.
2. In each of the graphs we define the variable "label" according to the appropriate dataproperties ("has name" and "has synonym" properties, respectively). For this, we use the "show as" functionality, which enables to rename the variables, in this case under the common variable called "label".
3. We return to the main graph where we will join the two subqueries. For this, we include the subgraphs clicking on the green flap of each of the subgraphs.
4. Select one of the subgraphs represented as nodes and click on "Define union".
5. In "Node shown" we select the variables to be shown and in "Filters set" we filter the variable "label".
6. Run the query (Query).

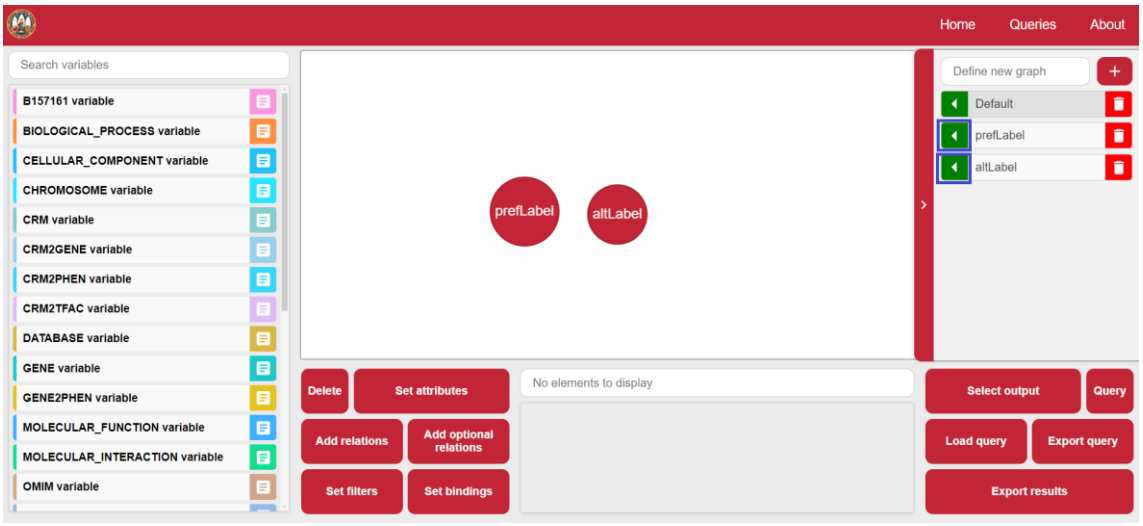
1.



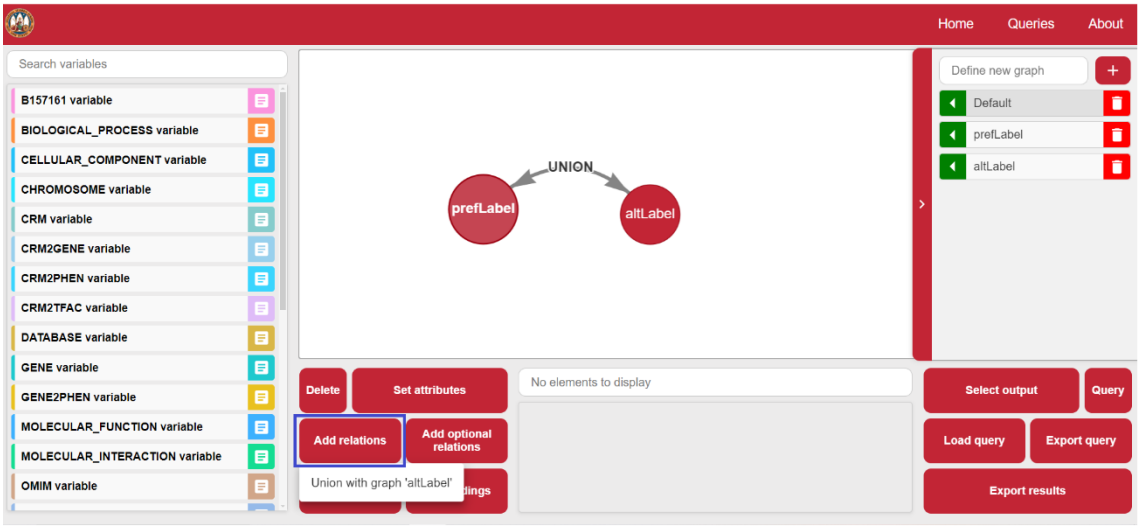
2.



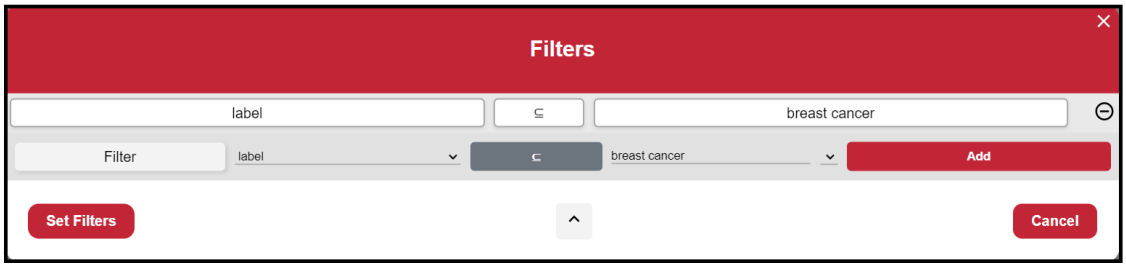
3.


Home
Queries
About
Search variables
B157161 variable
BIOLOGICAL_PROCESS variable
CELLULAR_COMPONENT variable
CHROMOSOME variable
CRM variable
CRM2GENE variable
CRM2PHEN variable
CRM2TFAC variable
DATABASE variable
GENE variable
GENE2PHEN variable
MOLECULAR_FUNCTION variable
MOLECULAR_INTERACTION variable
OMIM variable
prefLabel
altLabel
Define new graph
Default
prefLabel
altLabel
Delete
Set attributes
No elements to display
Select output
Query
Add relations
Add optional relations
Load query
Export query
Set filters
Set bindings
Export results

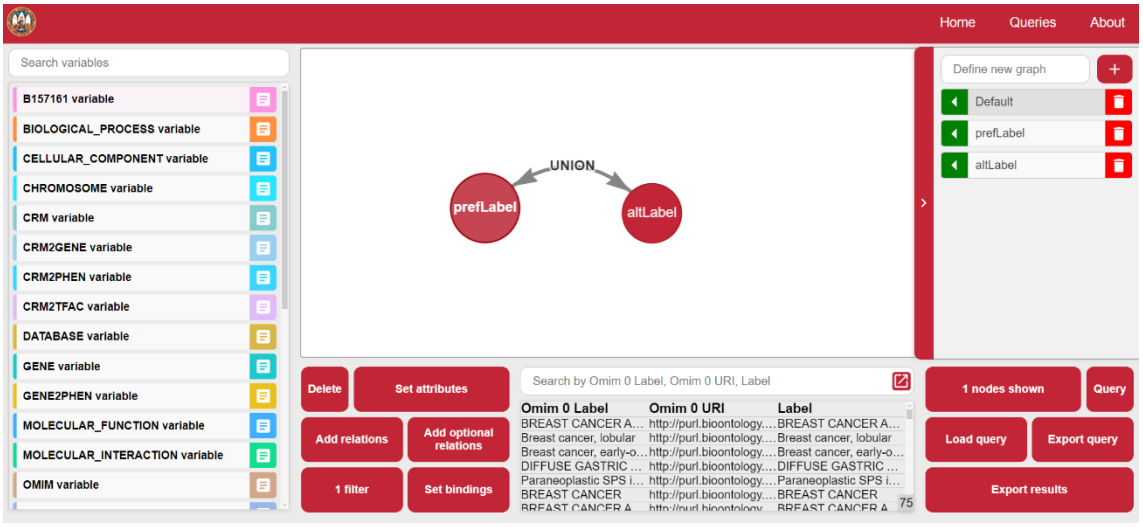
4.


Home
Queries
About
Search variables
B157161 variable
BIOLOGICAL_PROCESS variable
CELLULAR_COMPONENT variable
CHROMOSOME variable
CRM variable
CRM2GENE variable
CRM2PHEN variable
CRM2TFAC variable
DATABASE variable
GENE variable
GENE2PHEN variable
MOLECULAR_FUNCTION variable
MOLECULAR_INTERACTION variable
OMIM variable
UNION
prefLabel
altLabel
Define new graph
Default
prefLabel
altLabel
Delete
Set attributes
No elements to display
Select output
Query
Add relations
Add optional relations
Union with graph 'altLabel'
Load query
Export query
Export results

5.

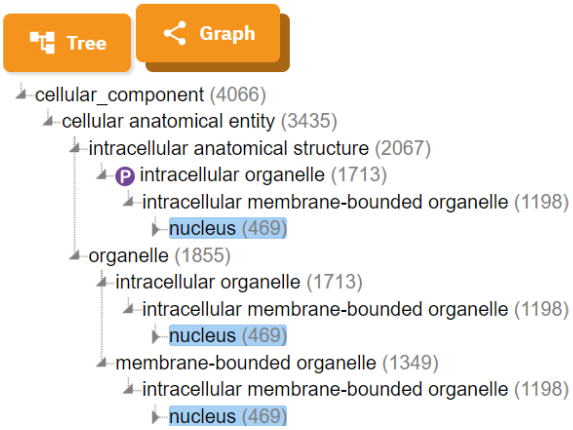

Filters
label
⊆
breast cancer
Filter
label
⊆
breast cancer
Add
Set Filters
Cancel

6.


Home
Queries
About
Search variables
B157161 variable
BIOLOGICAL_PROCESS variable
CELLULAR_COMPONENT variable
CHROMOSOME variable
CRM variable
CRM2GENE variable
CRM2PHEN variable
CRM2TFAC variable
DATABASE variable
GENE variable
GENE2PHEN variable
MOLECULAR_FUNCTION variable
MOLECULAR_INTERACTION variable
OMIM variable
UNION
prefLabel
altLabel
Define new graph
Default
prefLabel
altLabel
Delete
Set attributes
Search by Omim 0 Label, Omim 0 URI, Label
1 nodes shown
Query
Omim 0 Label
Omim 0 URI
Label
BREAST CANCER A...
http://purl.bioontology....
BREAST CANCER A...
Breast cancer, lobular
http://purl.bioontology....
Breast cancer, lobular
Breast cancer, early-o...
http://purl.bioontology....
Breast cancer, early-o...
DIFFUSE GASTRIC ...
http://purl.bioontology....
DIFFUSE GASTRIC ...
Paraneoplastic SPS i...
http://purl.bioontology....
Paraneoplastic SPS i...
BREAST CANCER
http://purl.bioontology....
BREAST CANCER
75
Add relations
Add optional relations
Load query
Export query
1 filter
Set bindings
Export results

5.7. Transitivity in INTUITION

Transitivity is a functionality that allows inferring relations between entities through existing relations between other entities, i.e. if an entity A is related to an entity B, and B is related to an entity C, it is possible to infer that A is related to C. For example, ontologies usually have a hierarchical structure. Gene Ontology (GO) is one of the ontologies imported into BioGateway and has a hierarchical structure of entities. GO has three hierarchically superior entities (biological process, cellular component and molecular function) which hold the other entities of the ontology. Thus, although the entity "nucleus" is not directly a subclass of the entity "cellular component", this relation can be inferred because "nucleus" is a subclass of an entity that inherits from "cellular component", with independence of the number of intermediate entities. Next, we illustrate this example with the hierarchical structure of this entity in GO, obtained by searching in the OLS Ontology portal:

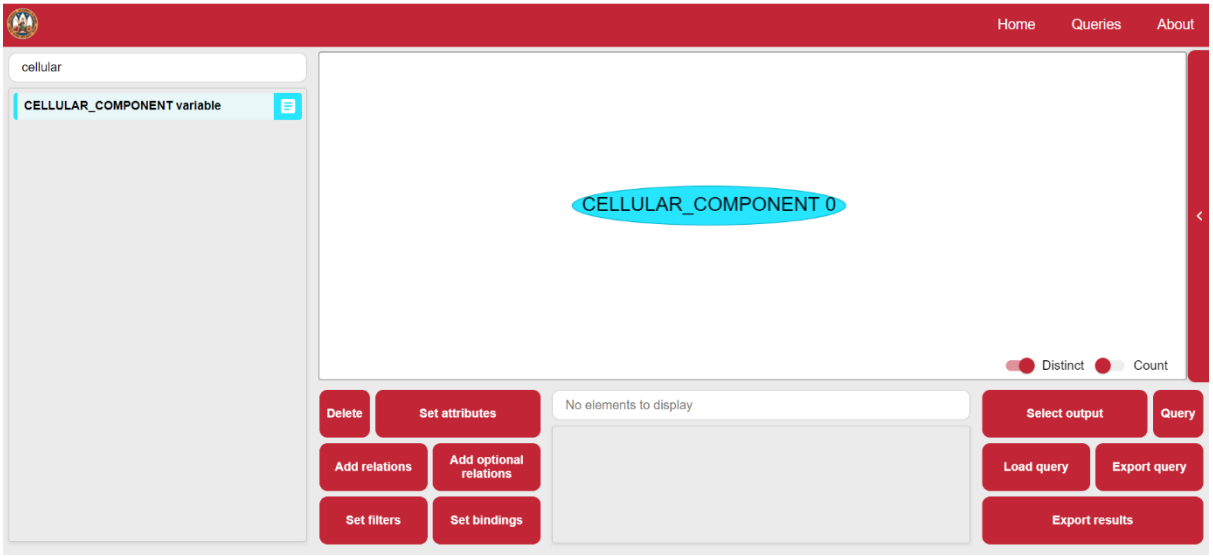


INTUITION automatically identifies hierarchically superior entities existing in the knowledge network, to present them to the user as variables available for use. The entities modelled in BioGateway (Section 9.3) are subclasses of a main entity used as a variable. This design was chosen to ease the user's exploitation of the data. In contrast, imported ontologies have a complex hierarchy. Then, we recall the variables corresponding to imported ontologies:

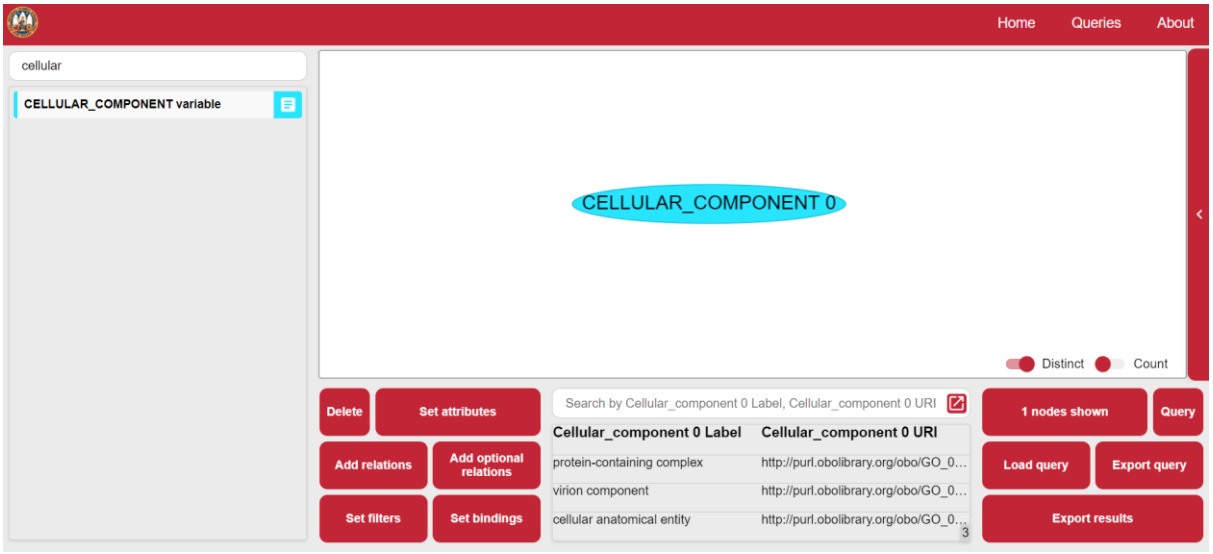
- OMIM variable: entities from OMIM ontology (mainly phenotypes).
- Molecular_interaction: entities from Molecular Interactions ontology (MI).
- Cellular_component variable: cellular components from Gene Ontology (GO).
- Molecular_function variable: molecular functions from GO.
- Biological_process variable: biological processes from GO.
- Root variable: top hierarchically class of NCBITaxon Ontology.
- Taxonomic_rank variable: top hierarchically class of NCBITaxon Ontology.

If we want to query *What cellular components does GO contain?* We can select the variable of interest (cellular component), include it for output ("Select output") and execute the query.

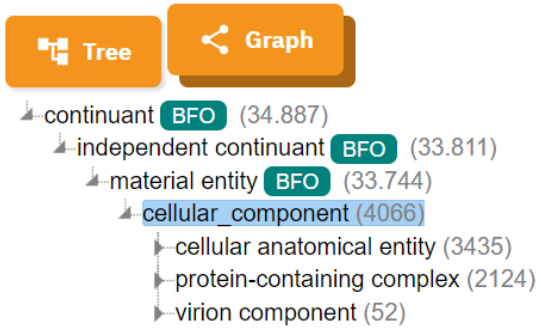
1.



2.

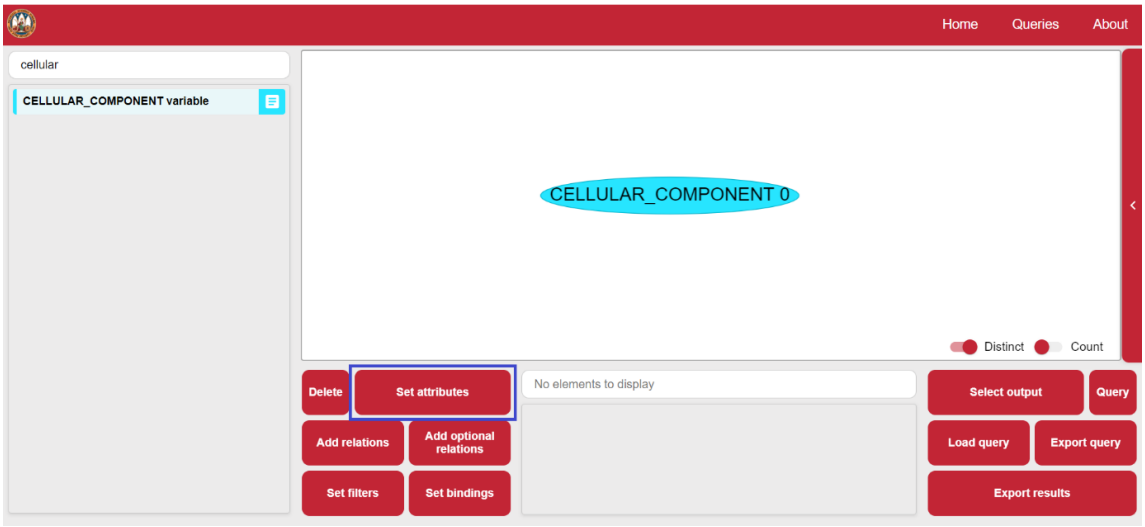


This query returns three entities (protein-containing complex, virion component, cellular anatomical entity) that correspond to the three entities that directly inherit from the GO cellular component entity:

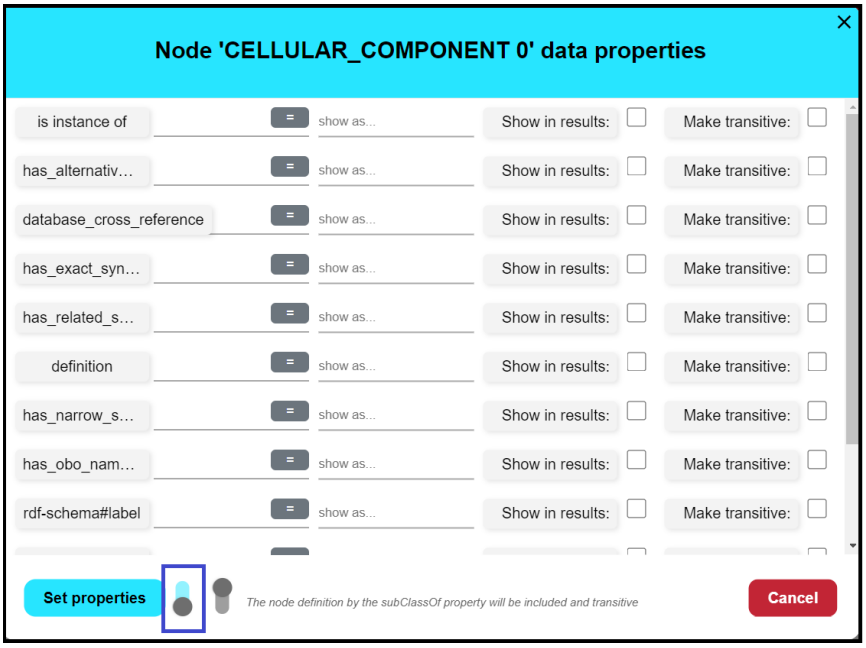


On the other hand, if we want to query which cellular components GO contains, regardless of the hierarchy level, we must run a transitive query. For this purpose, after including the variable "cellular compenent", we go to its attributes and activate the button next to "Set properties" ("The node definition by the subClassOf property will be included and transitive"). This functionality makes the property "is subclass of" transitive.

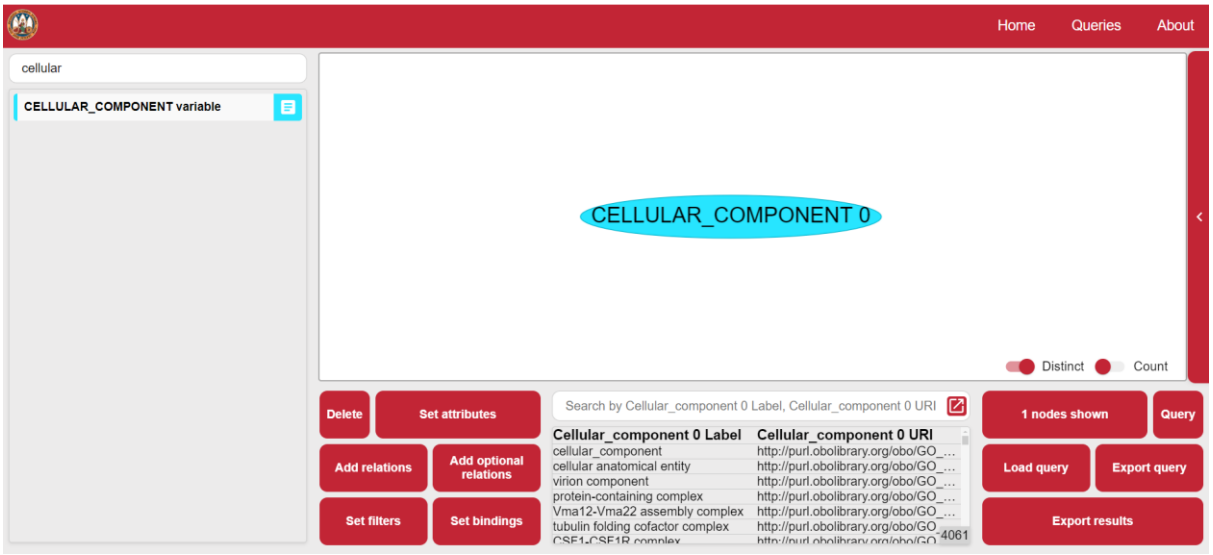
1.



2.

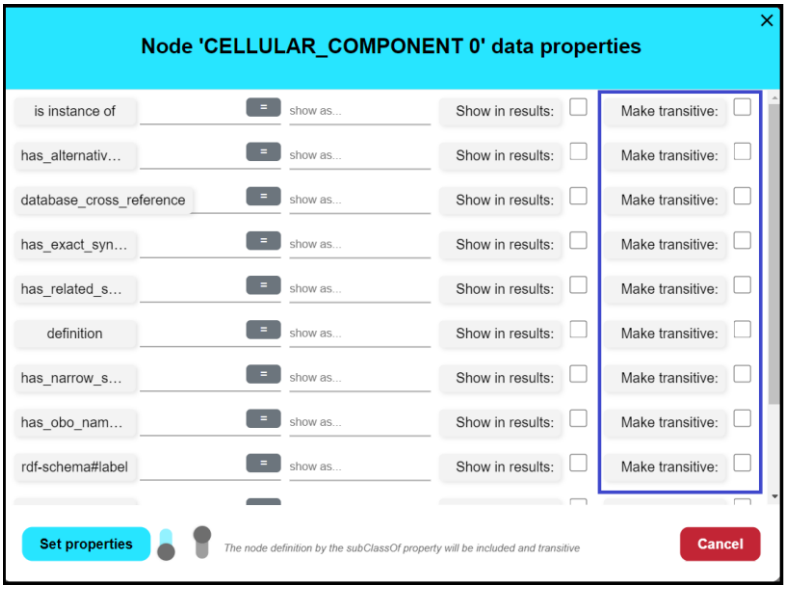


3.



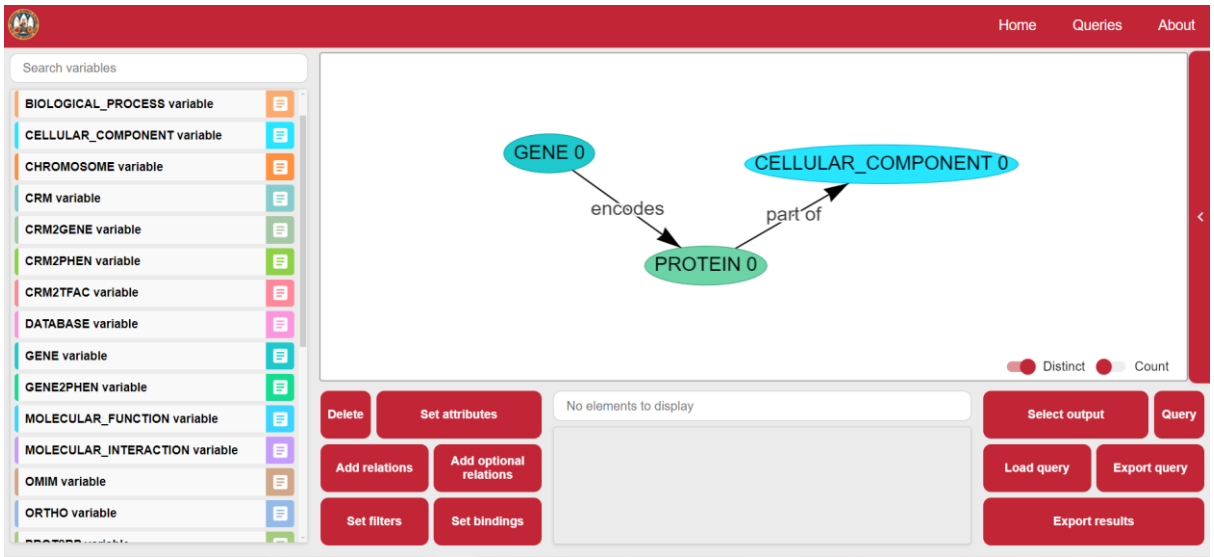
This way, instead of three results we get the 4061 cellular components, regardless of the hierarchy level.

If we want to make transitive a property of the attributes, we just have to check the related button next to the attribute.

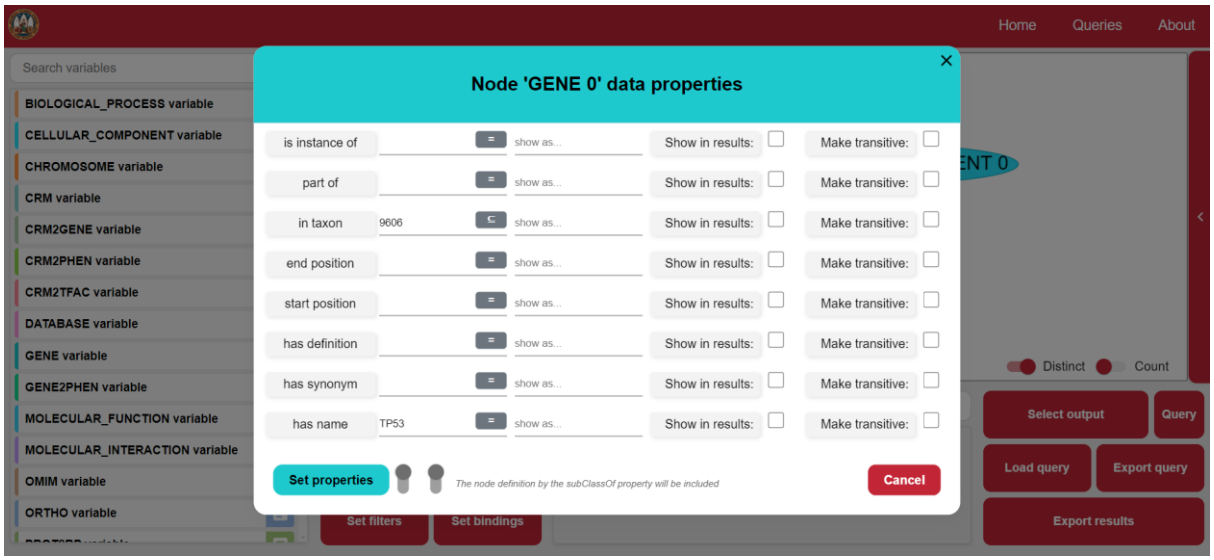


Here is another example involving transitivity: *In which cellular components is the human TP53 gene found?* To do this, we create the <Gene> <part of> <Cellular_component> relation. We then include the name and taxon of the gene in the Gene attributes. We select the data output and run the query.

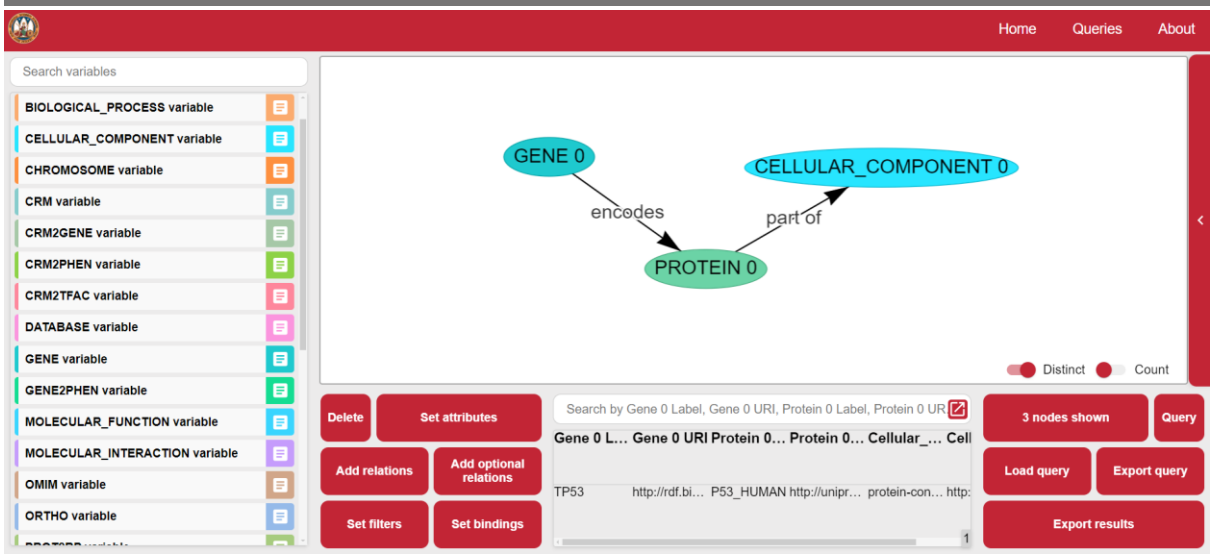
1.



2.

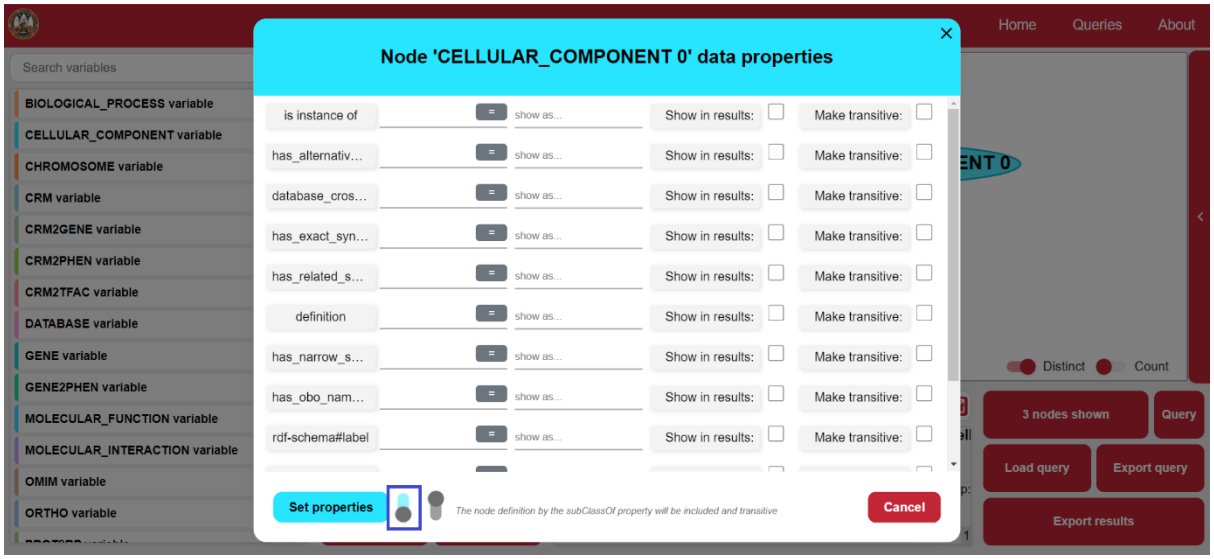


3.

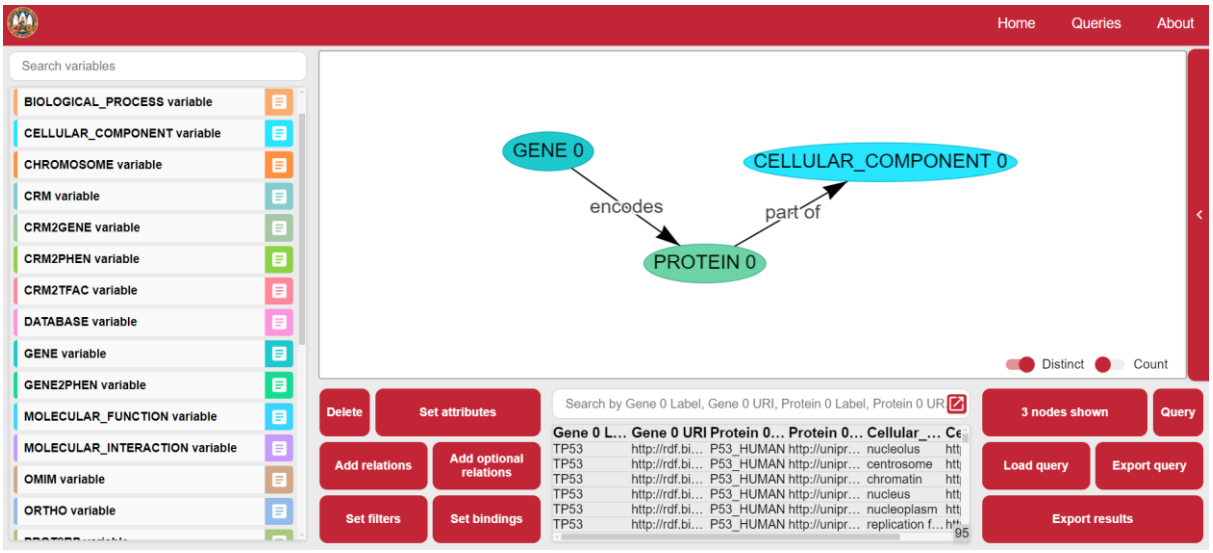


We return only one result (protein-containing complex), which is a direct subclass of "cellular component". To obtain all the results regardless of the hierarchy, we apply transitivity to the variable "cellular component":

4.

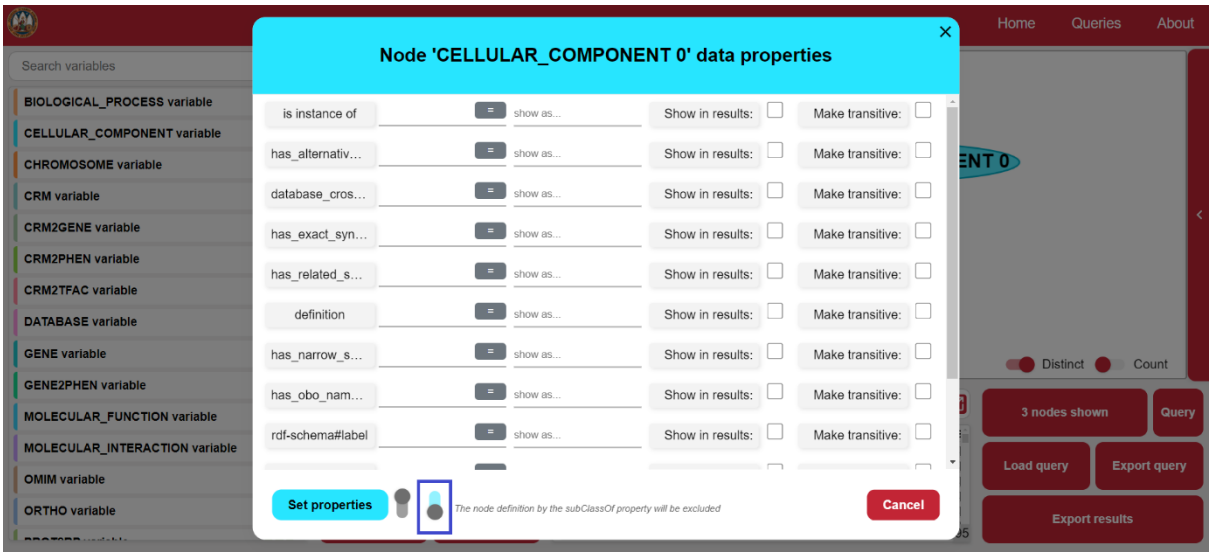


5.

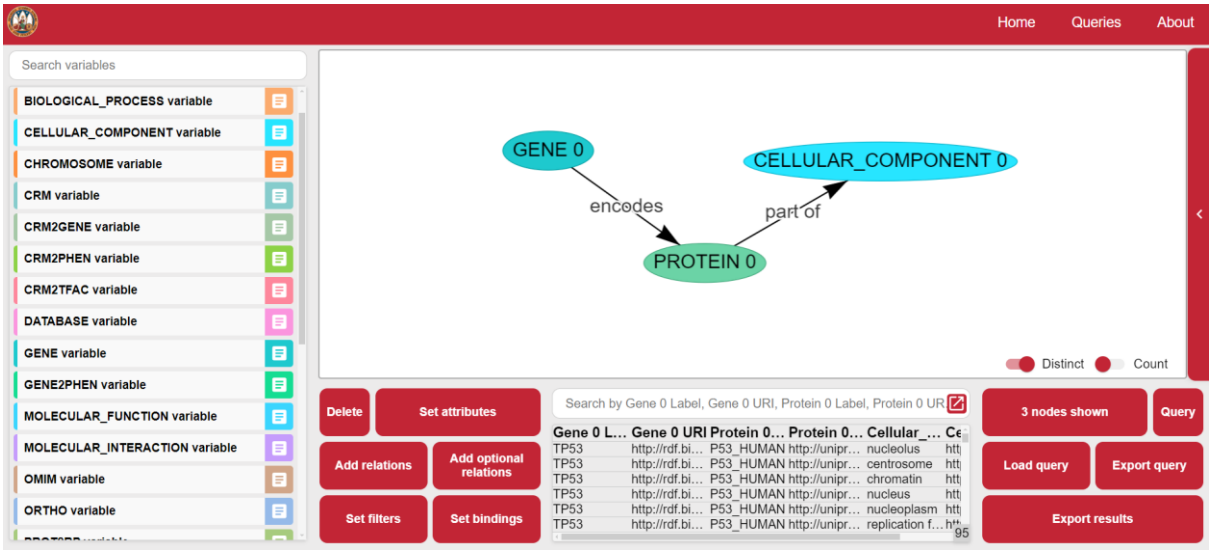


Complex queries which involve transitivity on variables that participate in relations can saturate the available memory. Therefore, the second button next to "Set properties" includes an optimised functionality for these cases (transitivity applied to variables taking part in relations).

6.



7.



6. Use Cases

The following Use Cases were developed in the paper "*Integration of chromosome locations and functional aspects of enhancers and topologically associating domains in knowledge graphs enables versatile queries about gene regulation*". The corresponding queries are attached for reproducibility and as examples of use. These use cases include complex queries that connect multiple nodes, use different filters, create variables, and join queries, so we recommend their consultation for a deeper understanding of the concepts introduced here for the graphical query building.

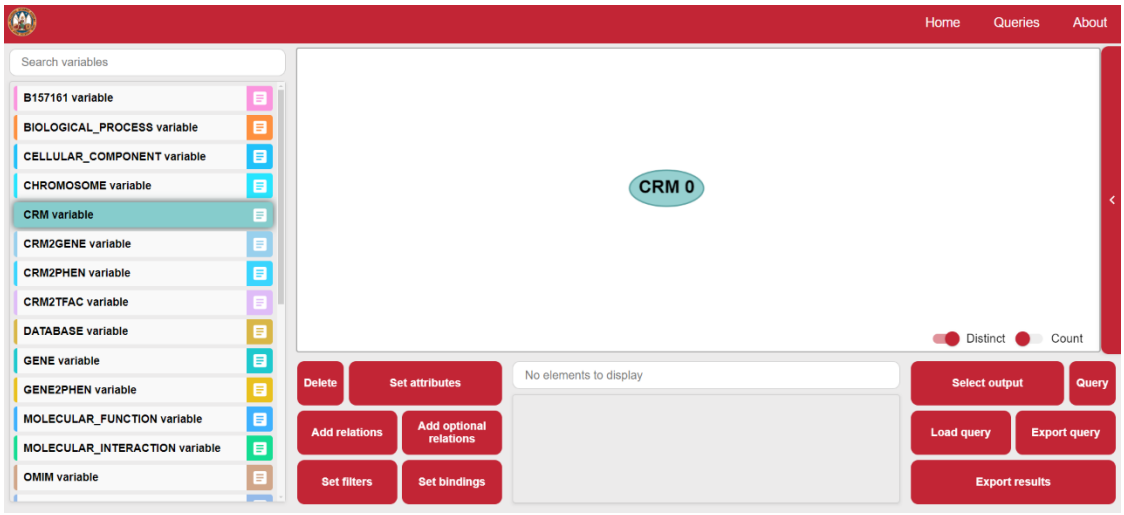
1. Use case 1: json files to load here.
2. Use case 2: json files to load here.
3. Use case 3: json files to load here.

Query execution times differ between use cases. Next, we include a table with an approximate query time.

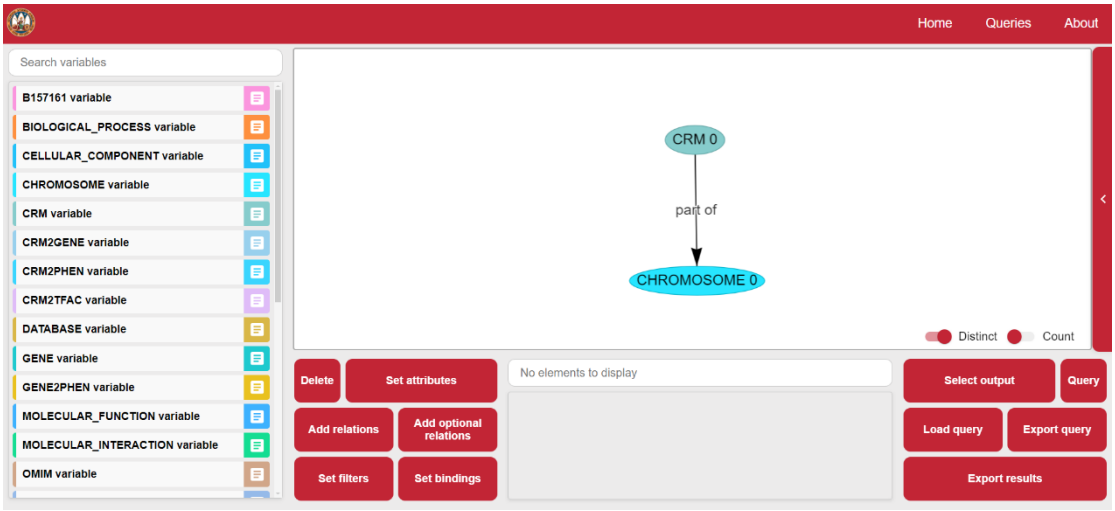
Query (Use Case – Query)	INTUITION query time (s)
UC1 - Q1	4
UC1 - Q2	26
UC1 - Q.3.1	2
UC1 - Q.3.2	3
UC1 - Q.3.3	1408
UC1 - Q.4.1	25
UC1 - Q.4.2	25
UC1 - Q.5.1	4
UC1 - Q.5.2	1
UC1 - Q.5.3	1602
UC2 - Q.1	10
UC2 - Q.2	3
UC2 - Q.3.1	8
UC2 - Q.3.2	728
UC2 - Q.4	2
UC2 - Q.5	2
UC3 – Q1	9

A guided step-by-step guide to building Use Case 1.1 is shown below: *Is the rs4784227 mutation (chr16:52565276) located in any enhancer sequence linked to target genes in the network? What databases support the sequence and what are their target genes? Is the enhancer related to any disease? Which proteins are encoded by the genes?*

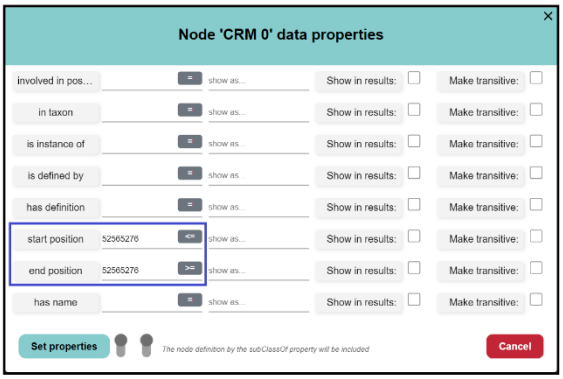
- First, we insert the CRM node by clicking on the CRM variable (“Variable browser” section).

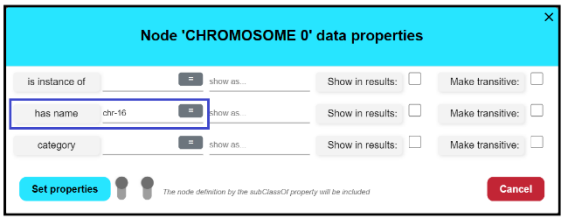


- We link the CRM to the chromosome variable (“Add relations” button).

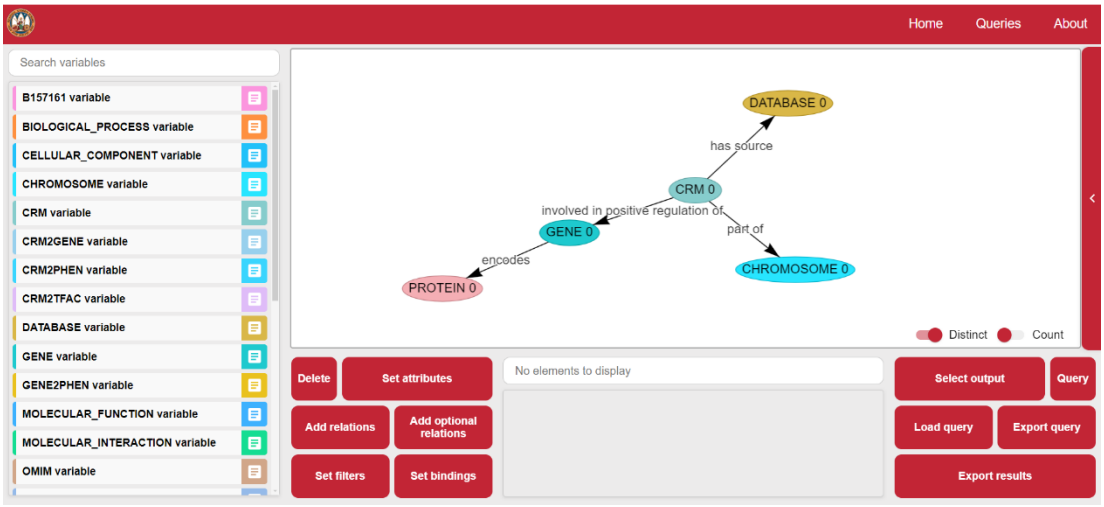


- And modify the attributes of both variables to select only those CRMs that overlap with the mutation (chr16:52565276) (“Set attributes” button).

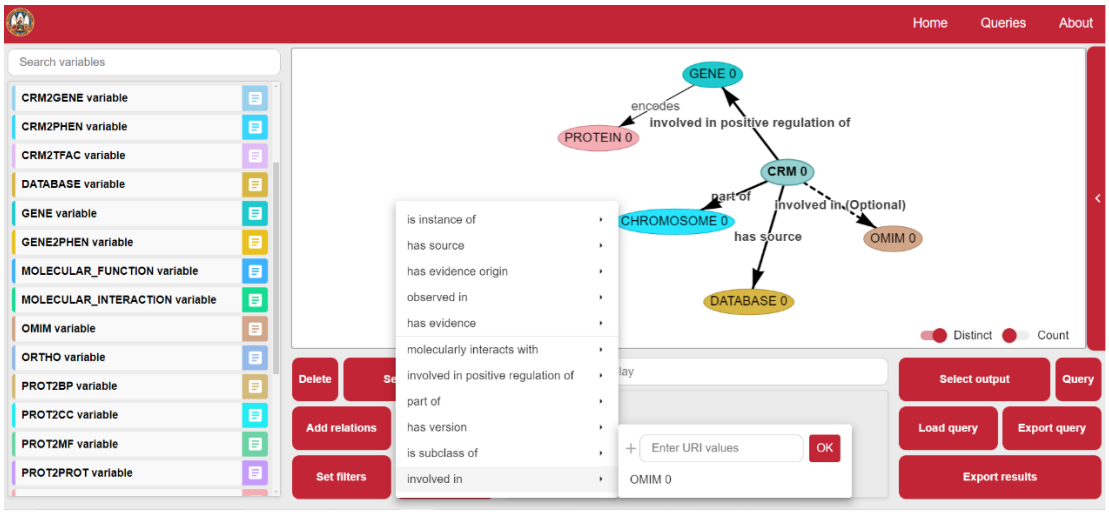




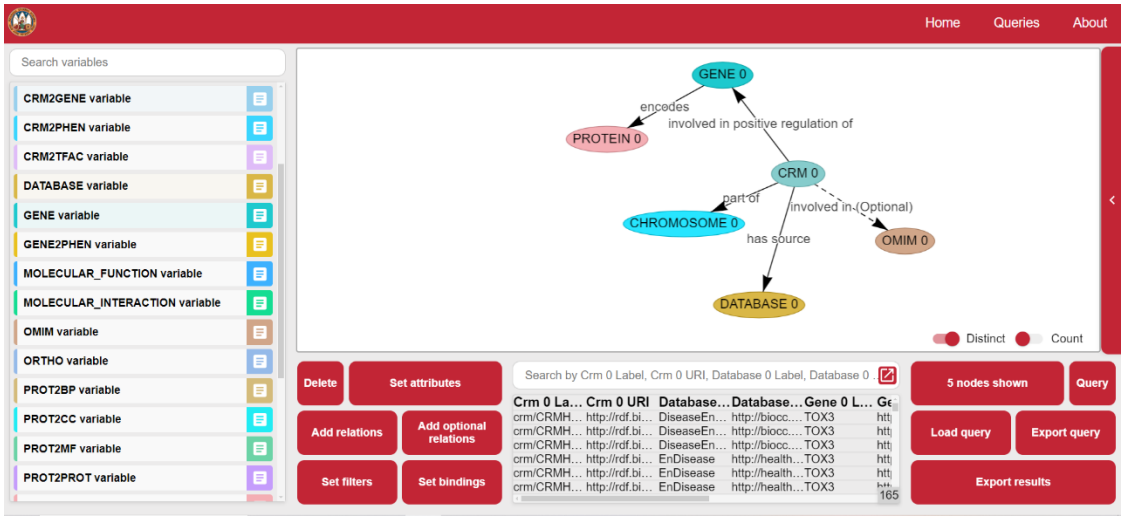
- We link the CRM entity with its database and target genes. We also link the genes to their encoded proteins (“Add relations” button).



- We include the relation between CRM and phenotype as an optional pattern (information that is included additionally and does not act as a filter) (“Add optional relations” button).



- Select the output data of interest (“Select ouput”) and run the query (“Query”).



- Finally, we can expand the results table, save the results table (“Export results”) and save the generated query (“Export query”).