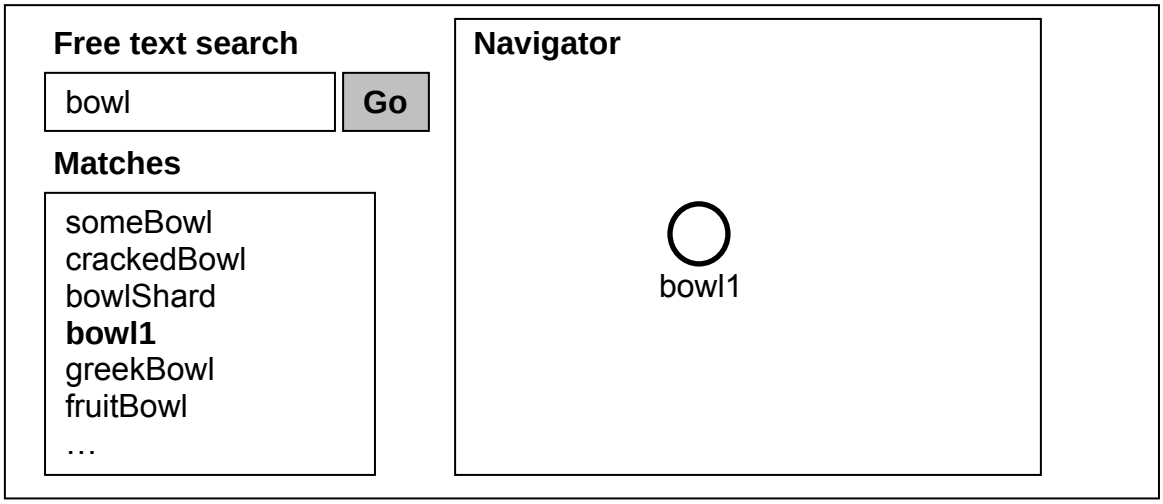
Linked data browser and annotator design

SPQR project, November, 2010.

Free text search and navigation

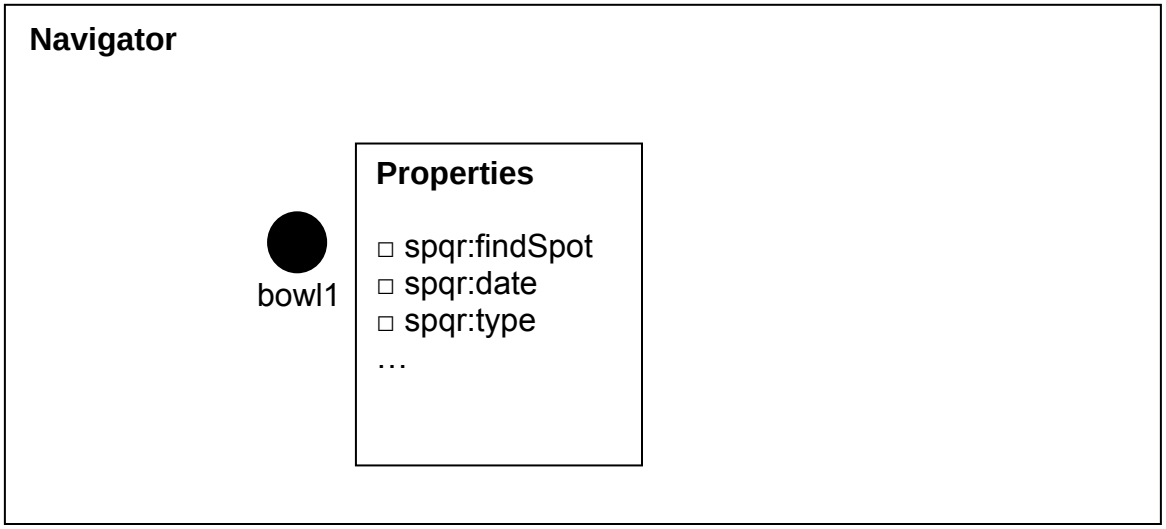
Users can run a free text search for a list of resources whose meta-data contains a specific search term:



If the user double clicks on a search result then a node for the resource is created in the Navigator. Alternatively, it can be drag-and-dropped into the Navigator area.

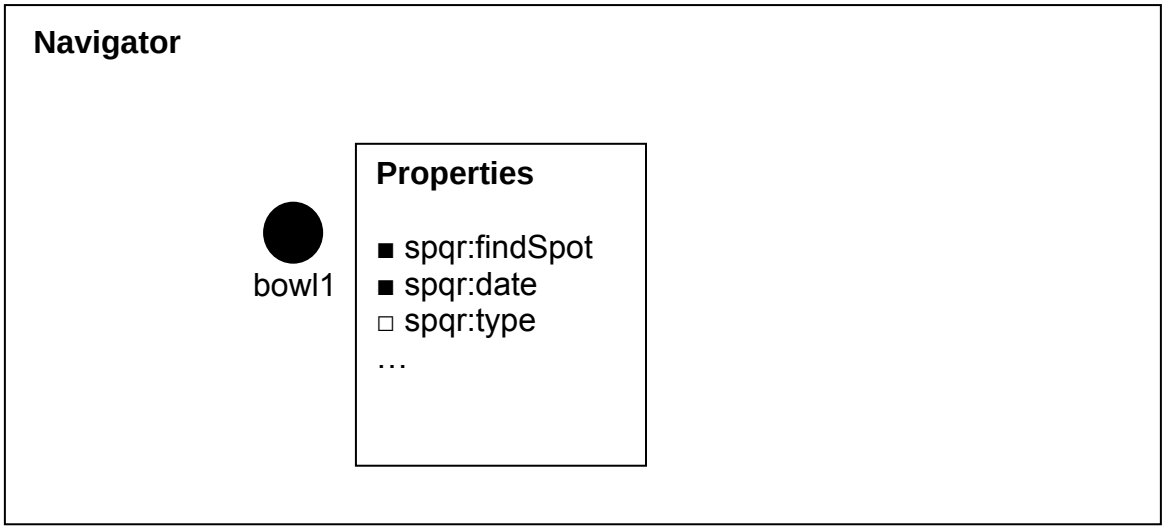
Navigation and relations / properties

Moving the mouse over a resource displays all its properties (relations) in a pop-up:



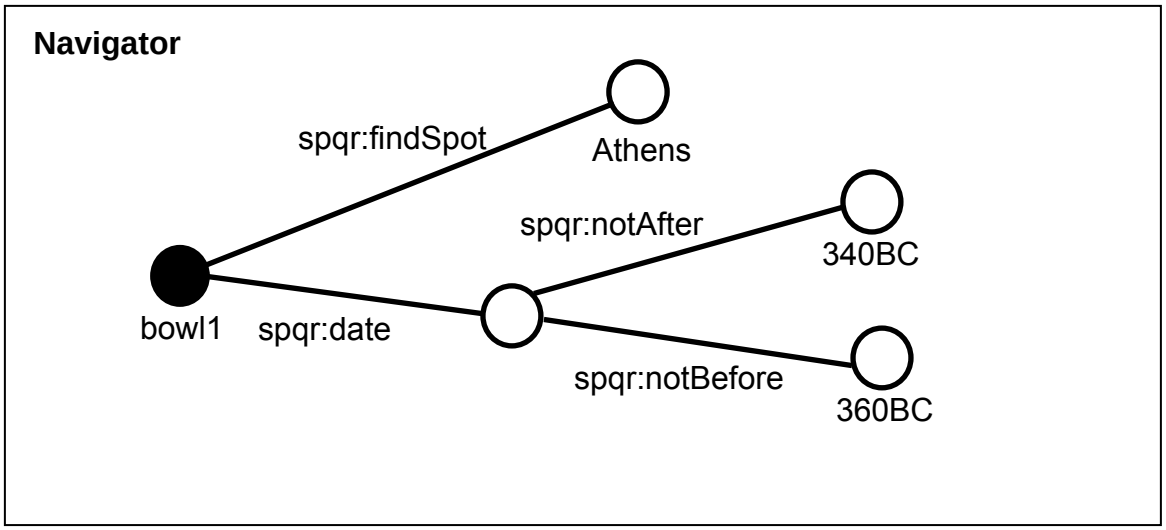
The pop-up disappears when the mouse is moved away.

A left mouse click keeps the properties pop-up open and allows selection of specific properties:

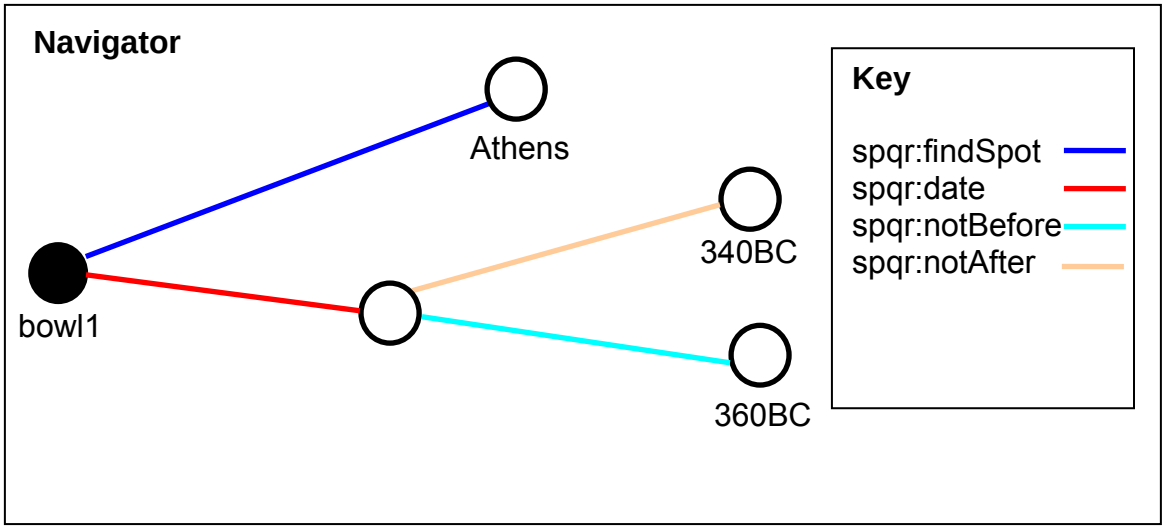


A double left click allows the user to select all the properties.

The Navigator shows the nodes at the other end of the selected properties/relations. Blank nodes (e.g. that related to spqr:notBefore) are expanded automatically:

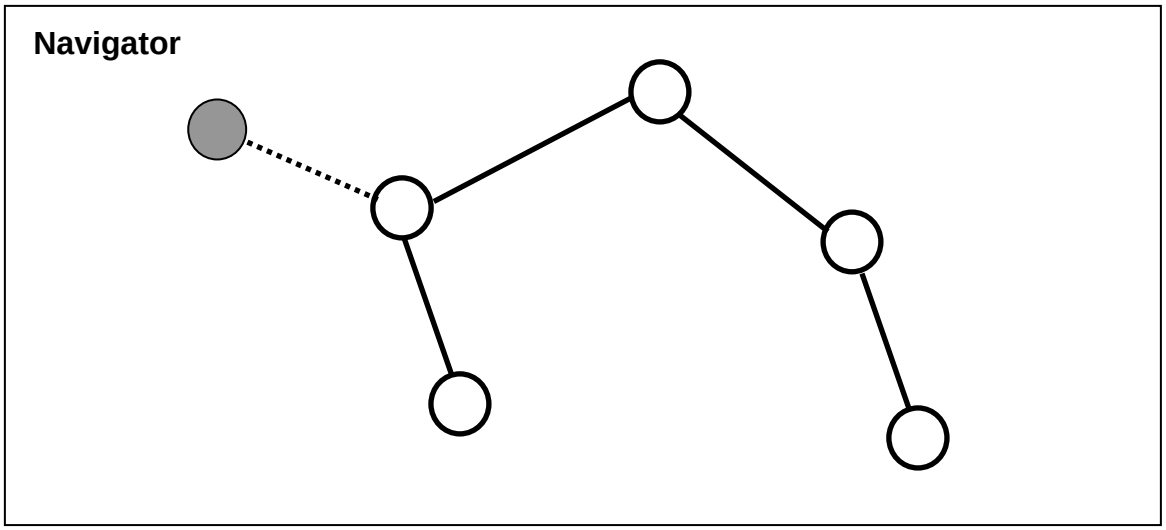


To reduce clutter, colour coding of predicates could be used instead of edge labels.



Managing what is shown – history nodes

To save screen real estate, old nodes are folded into a history node (the grey node below):

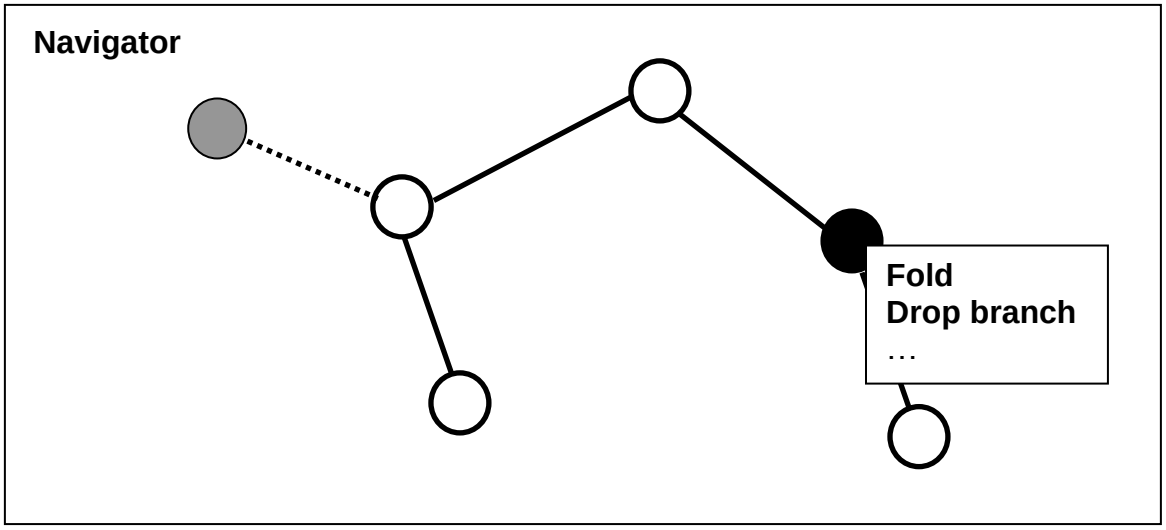


Double clicking on a history node moves back one hop.

There will be a configurable number-of-hops setting to control when folding happens. By default the graph will automatically fold nodes so the number-of-hops is conserved in each direction.

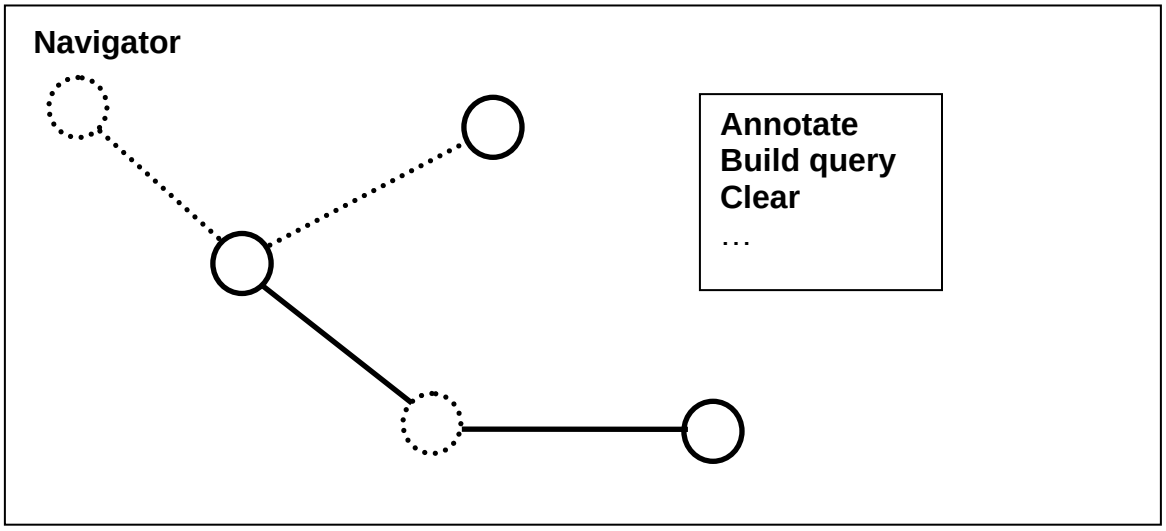
Once a history node is folded it will not be folded again unless specifically requested.

Users can request that a branch be folded or dropped:



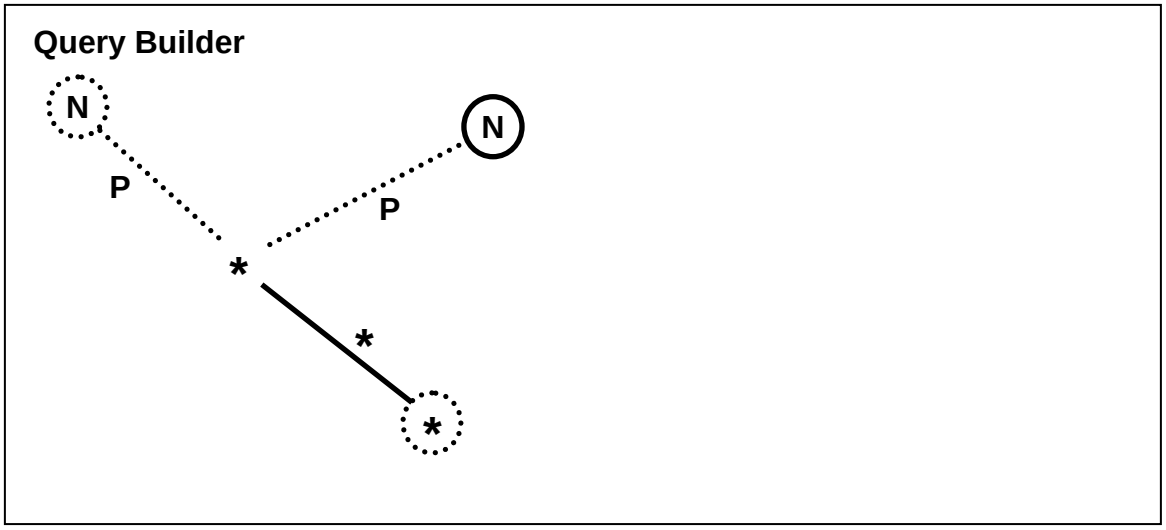
Building queries

Users can construct queries via the Navigator by selecting nodes and edges via CTRL and left clicking the mouse. Below, two nodes and two edges (with the dotted lines) have been selected:



CTRL and a click brings up a pop-up allowing the user to move the selected parts of the graph to the query builder (or the annotator described later).

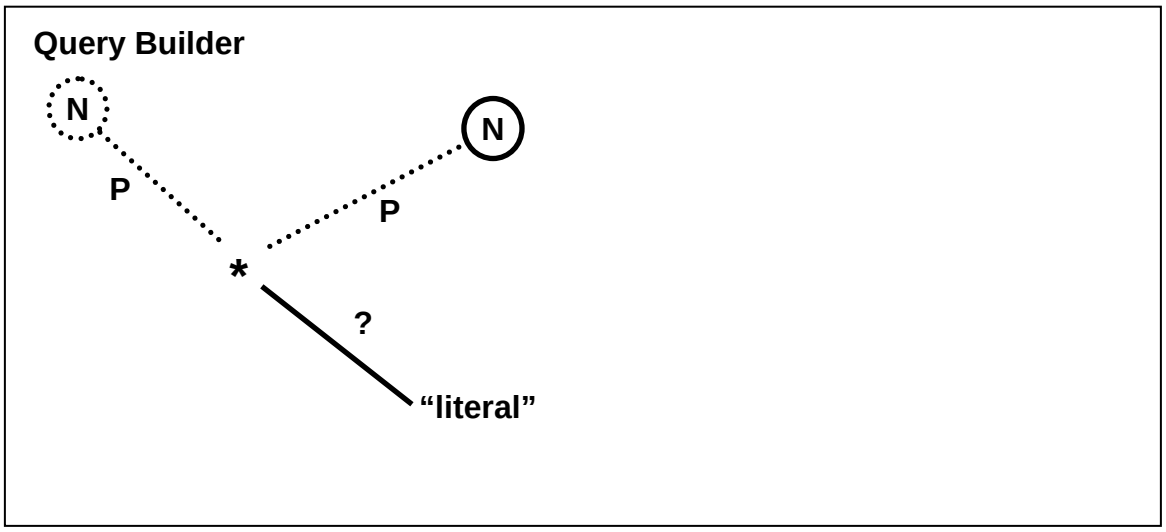
The query builder displays a template query derived from the selection and the user can then edit this:



Each node and predicate specifies a query condition e.g.:

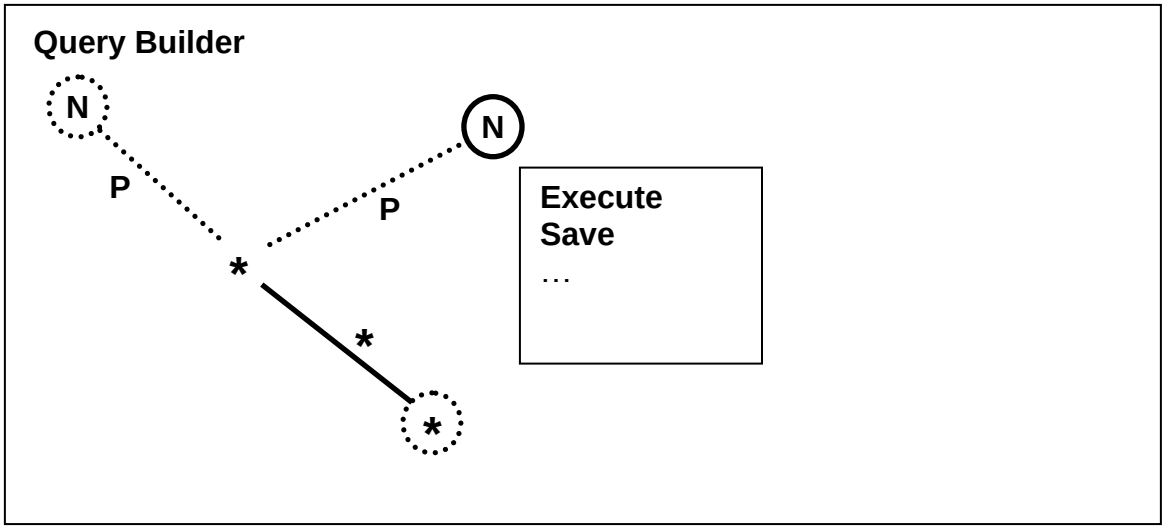
N: match a specific node URI.
P: matches a specific relation URI.
*: Match any relation; match any number of nodes.
(*): Match any single node.

This basic set can be extended to support multiple ORed predicates or nodes. For more complex operations, functions could be applied to literals e.g.



A left mouse click toggles between the options for each node or relation.

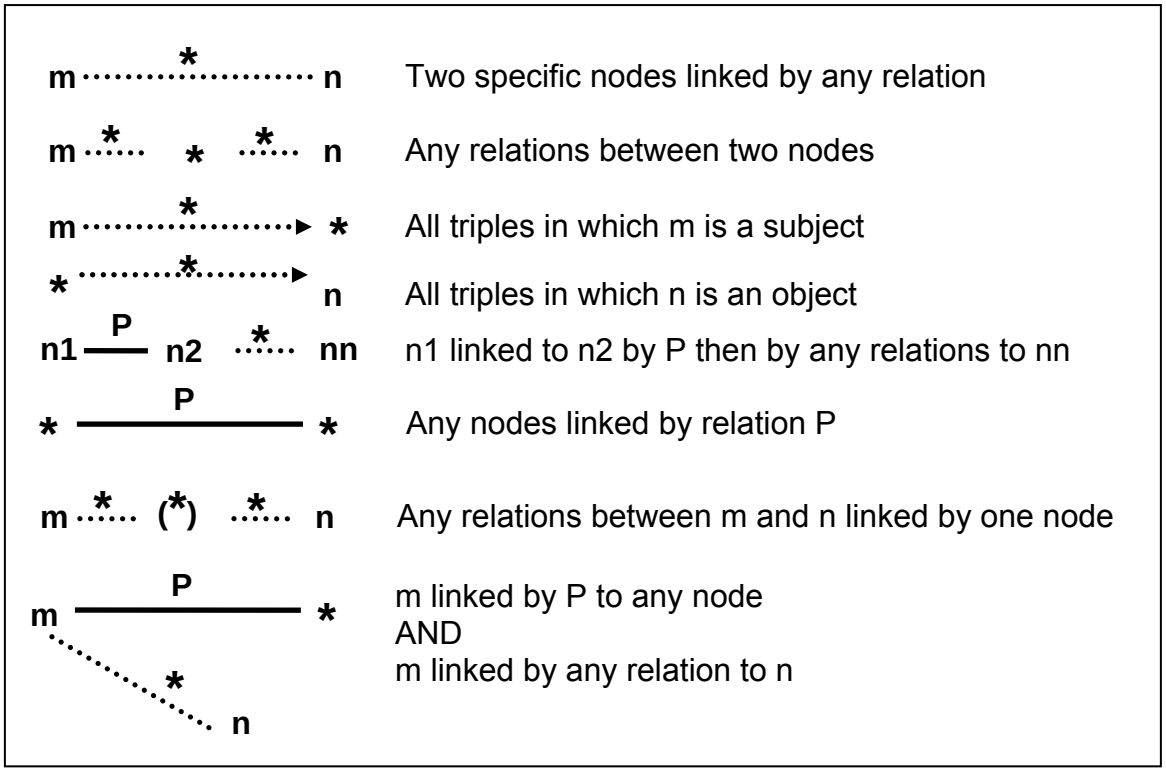
Users can opt to execute the query or save it as SPARQL:



On execution, results are displayed as a list. The navigation area can be updated to display any matching graphs.

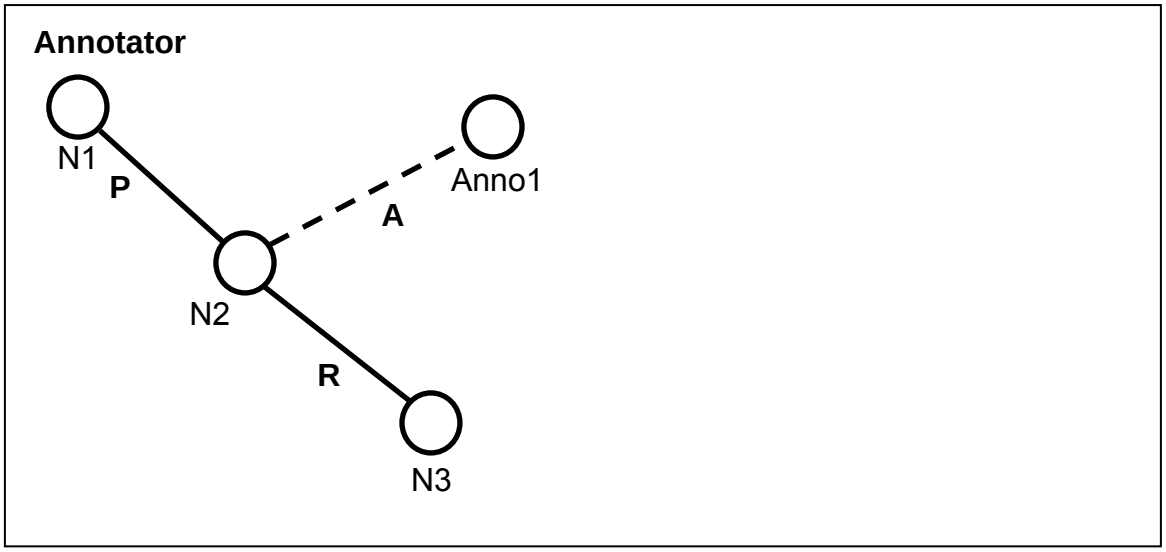
Saved queries can be edited on a SPARQL level. They will also be annotated and may link to actual nodes.

Sample graphical queries



Adding annotations

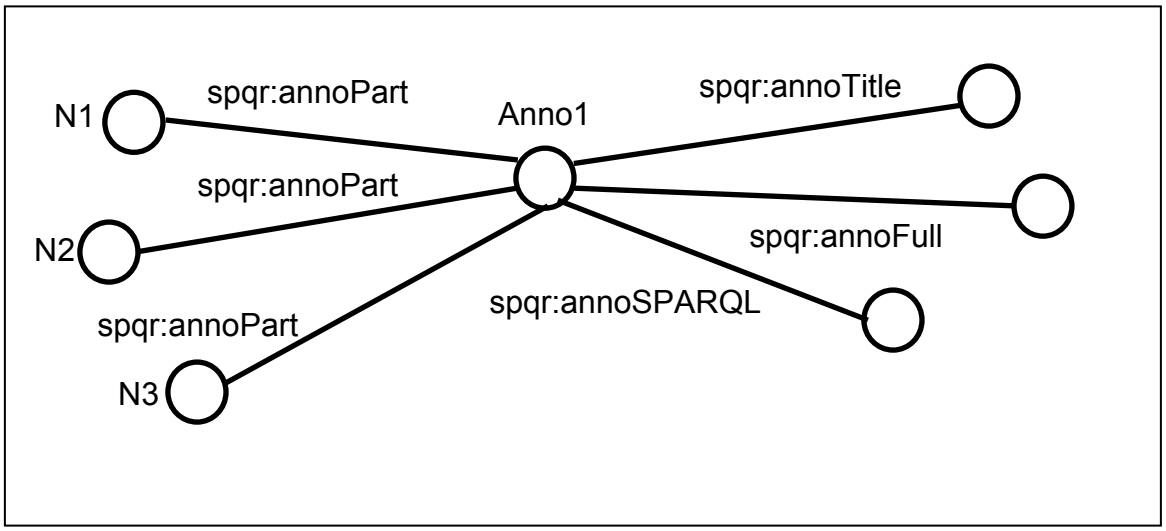
Annotations are represented as nodes and an annotation relation in the graph and can be browsed in the Navigator and created via the Query Builder.



Each annotation has the properties:

- TITLE: annotation short title
- FULL: full text of annotation
- SPARQL: exact subgraph is described by an auto-generated SPARQL query.

Each annotation becomes part of the graph with an auto-generated URI.



Annotations as aggregations gives the potential to link groups of nodes together.

Linking annotations happens in the navigation view. Need to decide whether to allow arbitrary predicates.

Annotated graphs need to include at least one node query pattens whose nodes are all wild-cards are illegal. We may be able to find a workaround.

These ideas need to be explored more.