KDI : Knowledge and Data Integration

GraphDB

W10.L19.M6.T19.2.2

Contents

Graph Databases

- In computing, a graph database (GDB) is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data.
- A key concept of the system is the graph (or edge or relationship). The graph relates the data items in the store to a collection of nodes and edges, the edges representing the relationships between the nodes.
- The relationships allow data in the store to be linked together directly and, in many cases, retrieved with one operation.
- Querying relationships is fast because they are perpetually stored in the database. Relationships can be intuitively visualized using graph databases, making them useful for heavily inter-connected data.

Graph Databases [Contd.]

- Graph databases are a type of NoSQL database, created to address the limitations of relational databases.
- While the graph model explicitly lays out the dependencies between nodes of data, the relational model and other NoSQL database models link the data by implicit connections.
- In other words, relationships are a first-class citizen in a graph database and can be labelled, directed, and given properties. This is compared to relational approaches where these relationships are implied and must be reified at run-time.
- Retrieving data from a graph database requires a (graph pattern-matching based) query language other than SQL (which was designed for the manipulation of data in a relational system). There are many such graph query languages.

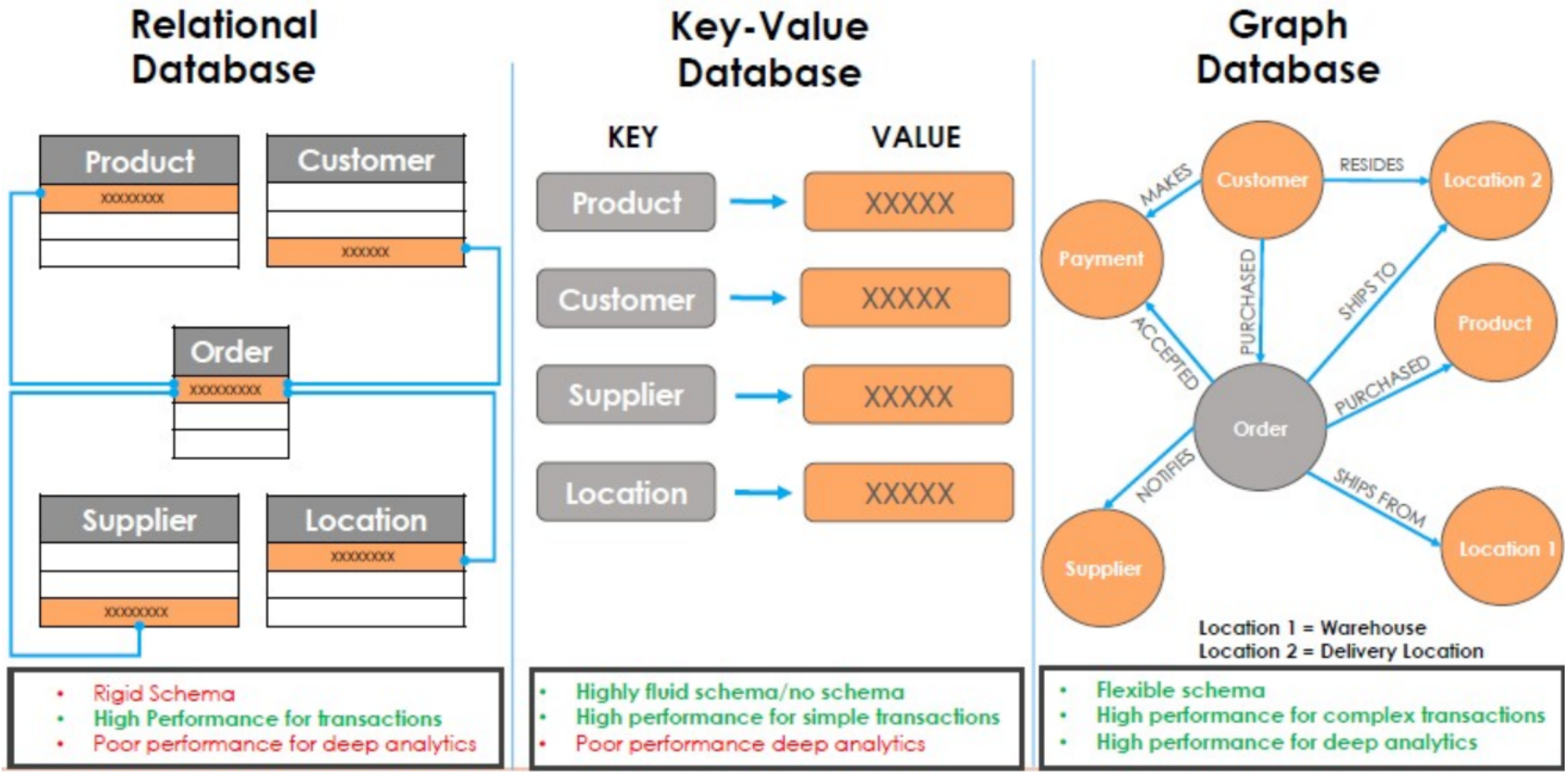
Graph Databases [Contd.]

A (not exhaustive) list of graph databases (many have a free/community edition):

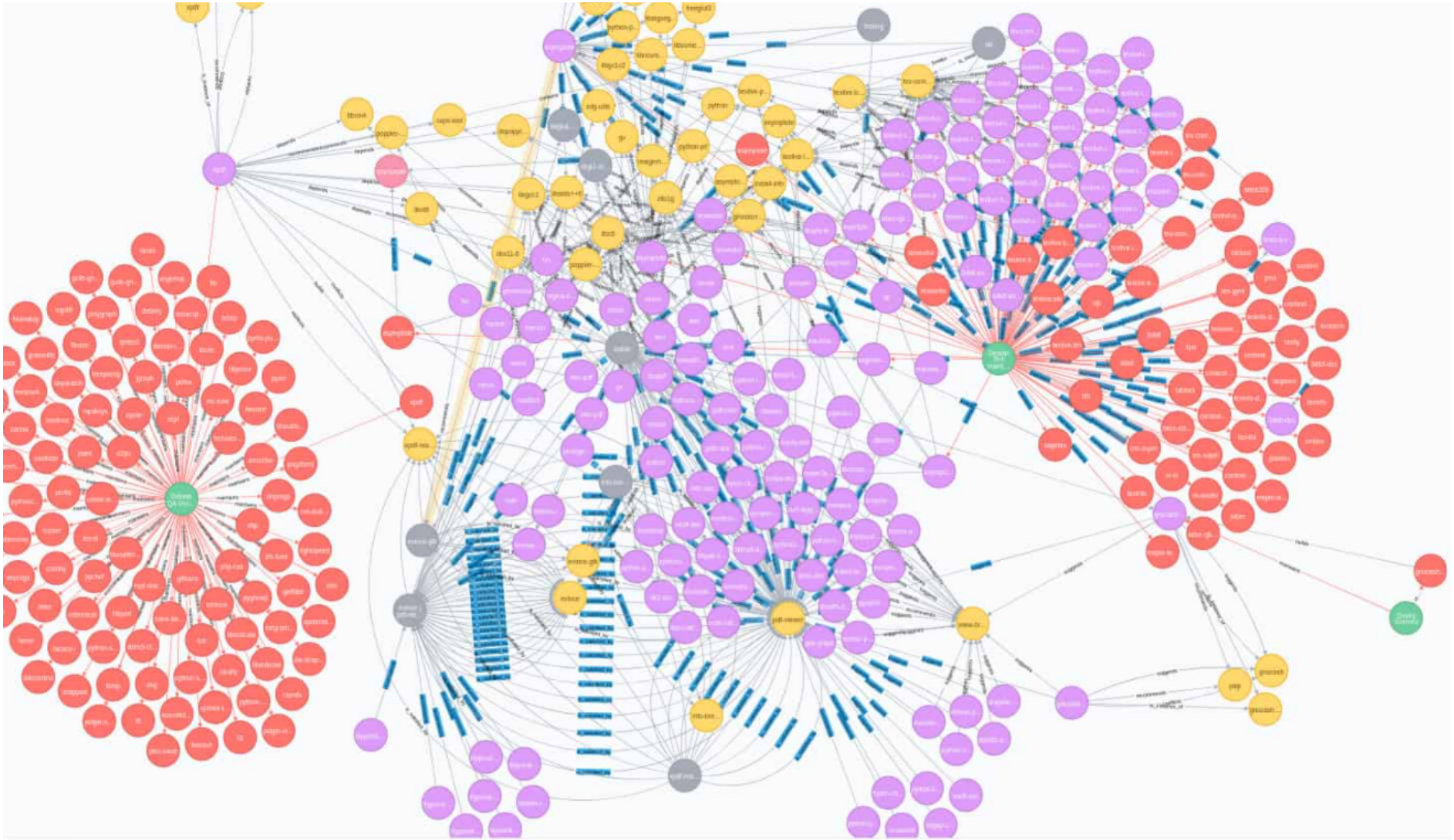
- **GraphDB**
- Neo4j
- Amazon Neptune
- AnzoGraph DB
- AllegroGraph (etc.)

They often differ in their underlying graph data models, and accordingly their graph query languages also differ.

Comparison of Databases



Visualization in Graph Databases



Graph Database: Usage in iTelos Methodology

- By now, we know that RDF is the underlying graph data model in our iTelos KG development methodology
- In the Data Integration phase, the output of KarmaLinker is an RDF file defining the Data Knowledge Graph (DKG) which can be represented, visualized and queried upon using graph databases.
- Our choice of Graph Database for this phase is **GraphDB Free**.
- GraphDB Free is a highly efficient, robust, and scalable Semantic Graph Database (native RDF), providing the core infrastructure for solutions where modelling agility, data integration and relationship exploration are important.

GraphDB Free: Features

GraphDB Free is one of the few triplestores that can perform semantic inferencing at scale, allowing users to derive new semantic facts from existing facts. Some of its important features are as follows:-

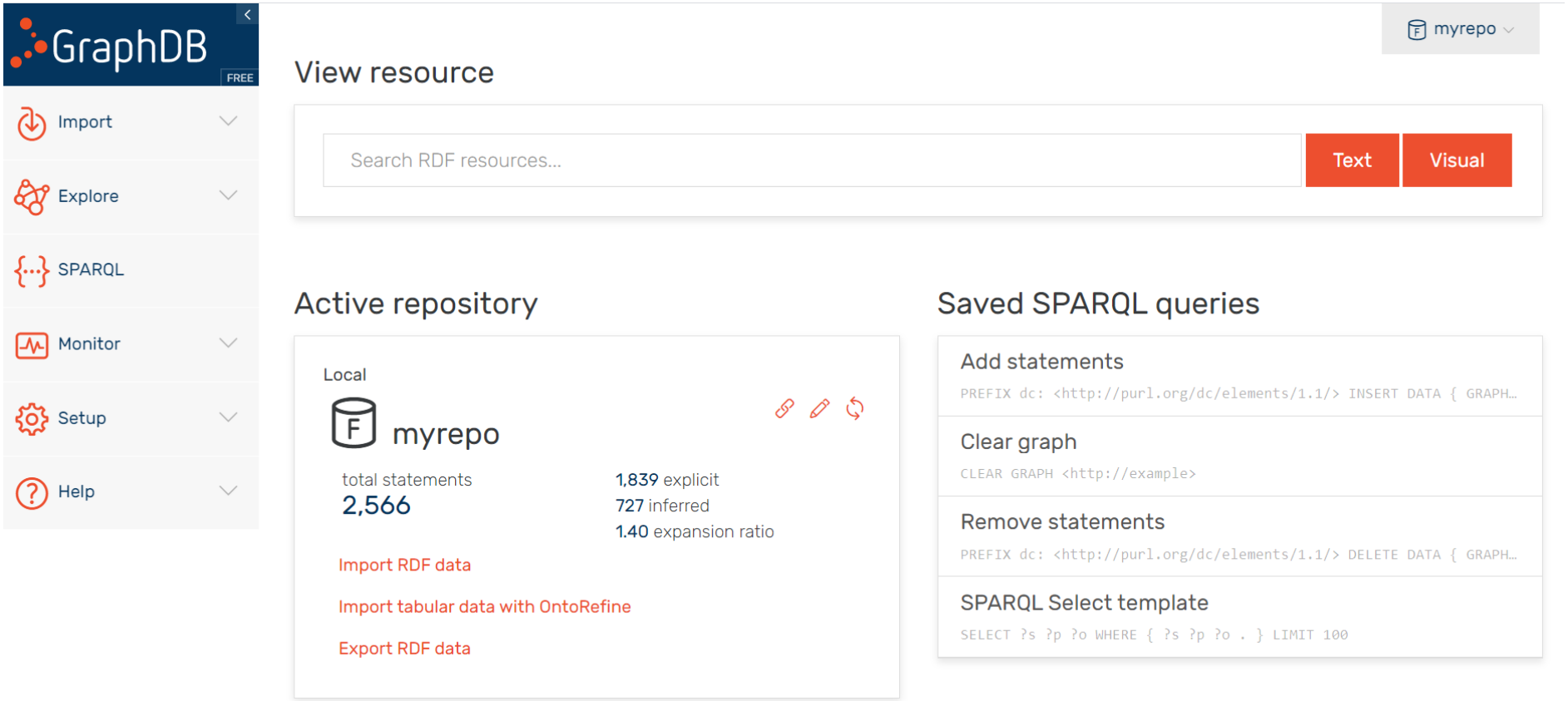
- Free to use
- Manages tens of billions of RDF statements on a single server.
- Performs query and reasoning operations.
- Scalability both in terms of data volume and loading and inferencing speed
- Does not require a license file. It is, however, not open source
- Fully W3C standard-compliant [RDF, RDFS, OWL, SPARQL].

Contents

Installation

- The easiest way to set up and run GraphDB is to use the native installations provided for the GraphDB Free edition.
- This kind of installation is the best option for your laptop/desktop computer, and does not require the use of a console, as it works in a graphic user interface (GUI).
- For this distribution, you do not need to download Java, as it comes pre-configured.
- Go to GraphDB Free and request your copy.
- You will receive an email with the download link and instructions (as per OS).

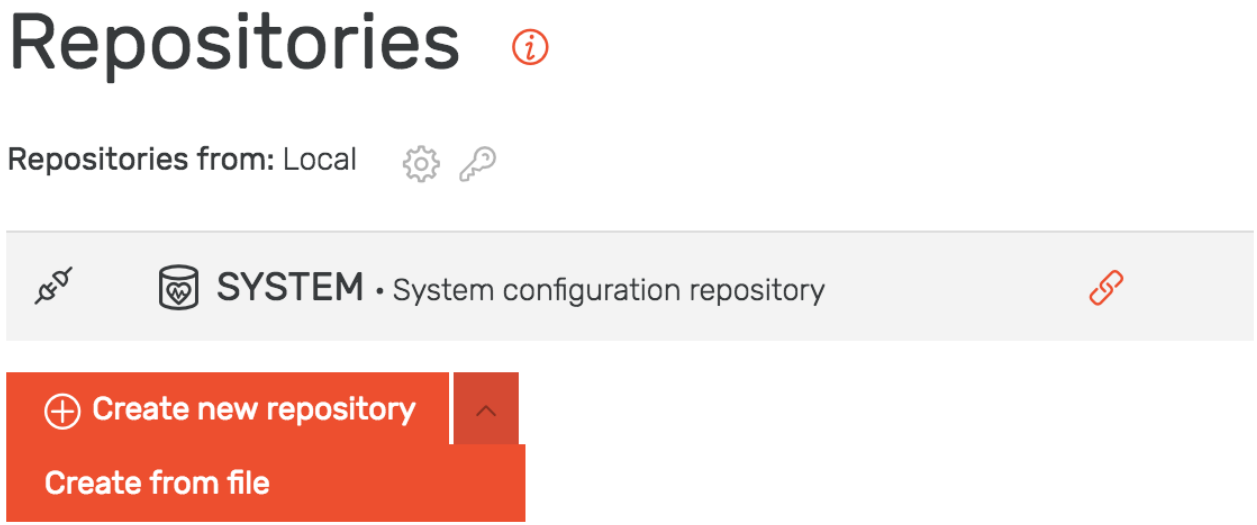
GraphDB Free GUI



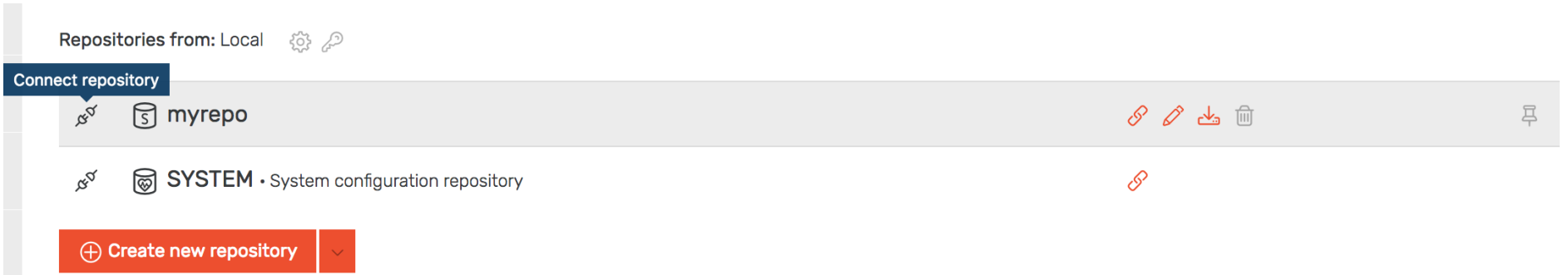
Creating a Repository

- The first step is to create a repository.
- Go to *Setup -> Repositories*.
- Click *Create new repository*.
- Enter a name (as per choice) as a *Repository ID* and leave all other optional configuration settings at their default values.
- Click the *Connect* button to set the newly created repository as the repository for this location.
- Use the pin to select it as the default repository.

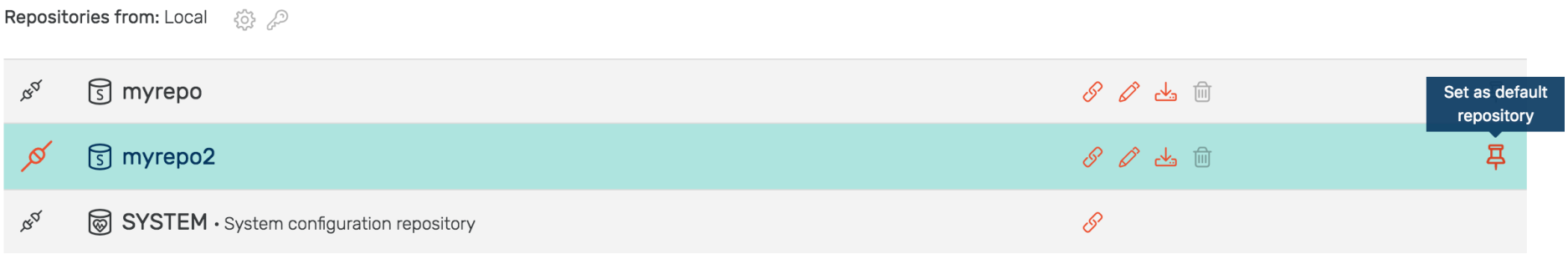
Creating a Repository: Illustration



Creating a Repository: Illustration [Contd.]



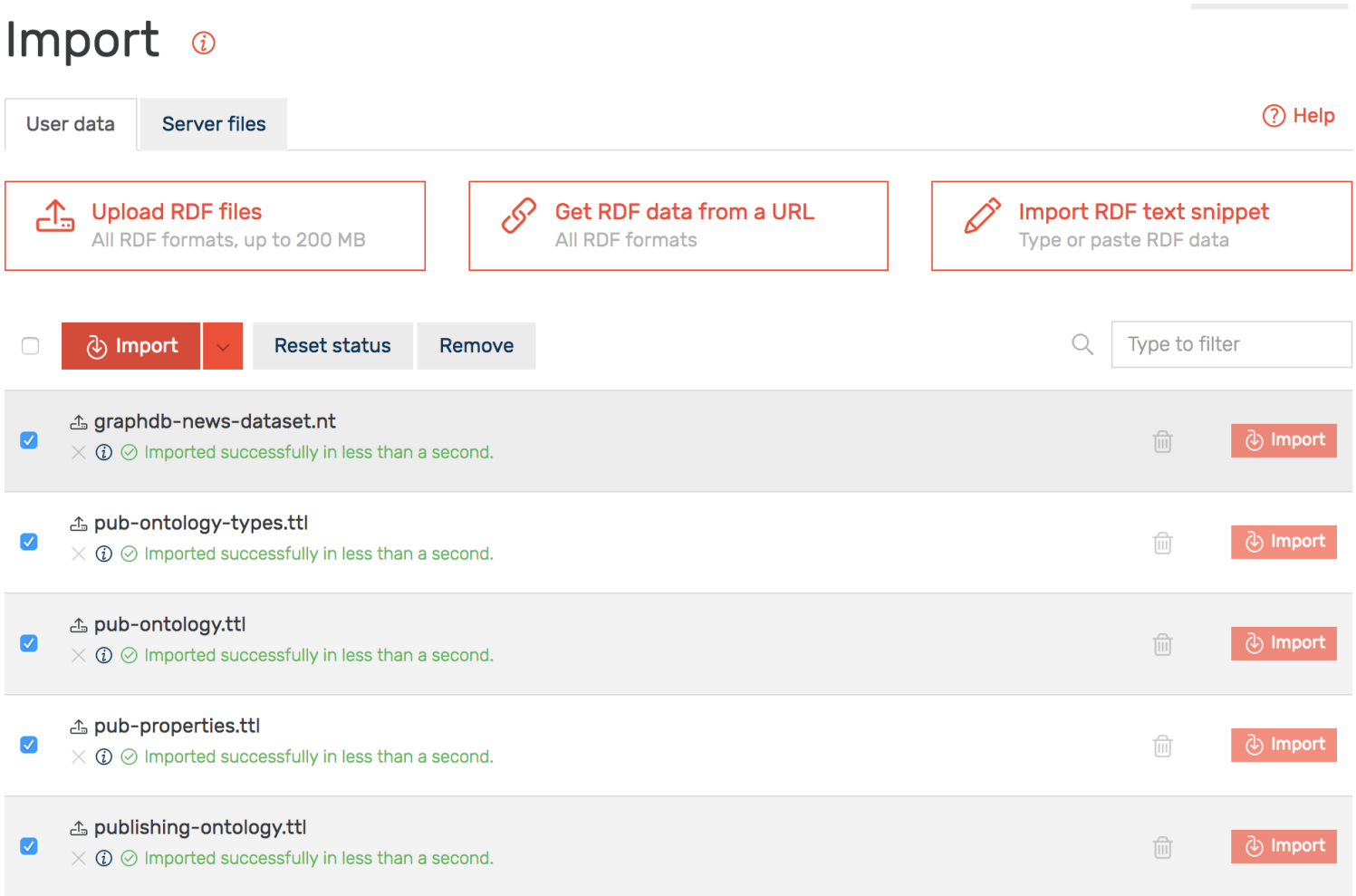
Creating a Repository: Illustration [Contd.]



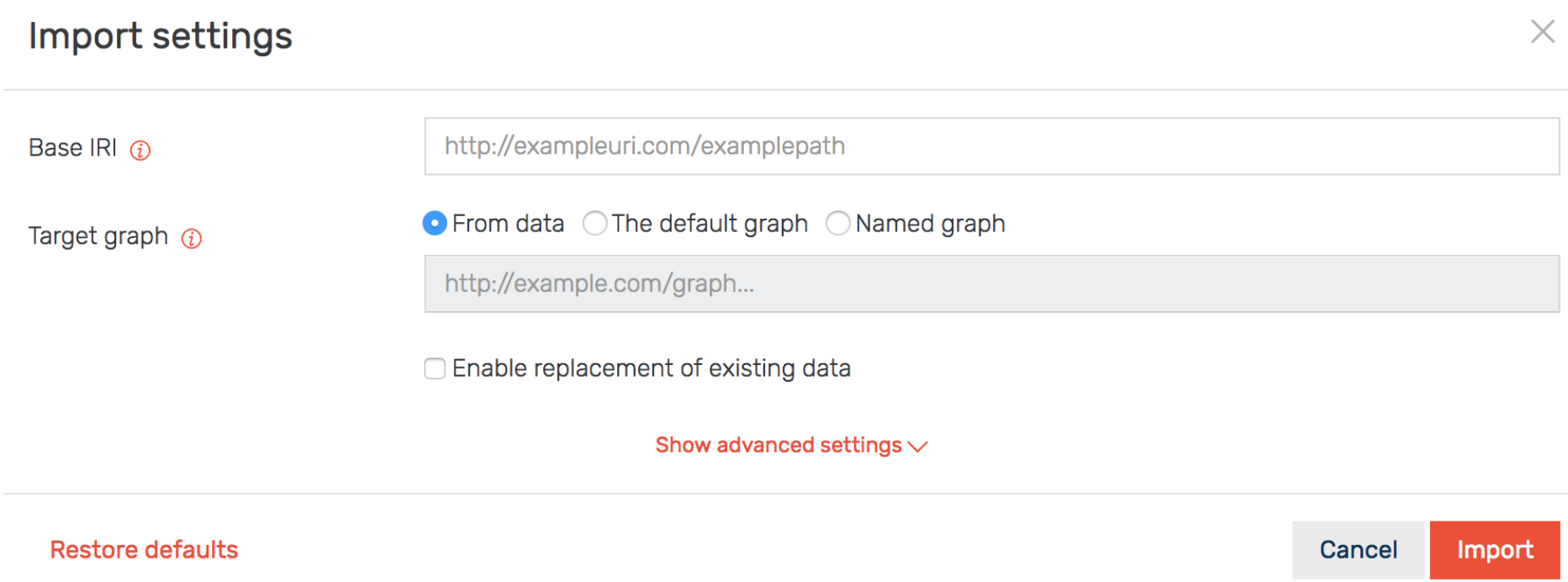
Loading a DKG through the GraphDB Workbench

- Go to *Import -> RDF*.
- Open the *User data* tab and click the *Upload RDF files* to upload the file
- Click the *Import* button.
- Enter the *Import* settings in the pop-up window.
- Start importing by clicking the *Import* button.

Loading a DKG through the GraphDB Workbench: Illustration



Loading a DKG through the GraphDB Workbench: Illustration [Contd.]



Exploring Instances

To explore instances and their relationships, navigate to *Explore -> Visual graph*, and find an instance of interest through the *Easy graph* search box.

Exploring Instances [Contd.]

Hover over a node to see a menu for the following actions:

- Expand a node to show its relationships or collapse to hide them if already expanded. You can also expand the node by double-clicking on it.
- Copy a node's IRI to the clipboard.
- Focus on a node to restart the graph with this instance as the central one. Note that you will lose the current state of your graph.
- Delete a node to hide its relationships and hide it from the graph.
- Click on a node to see more info about it: a side panel opens on the right, including a short description (rdfs:comment), labels (rdfs:label), image (foaf:depiction) if present, and all DataType properties.

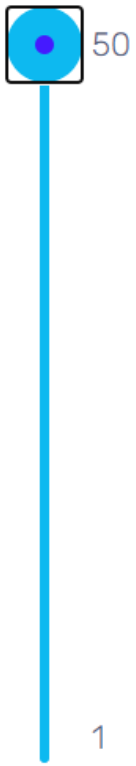
Class Hierarchy

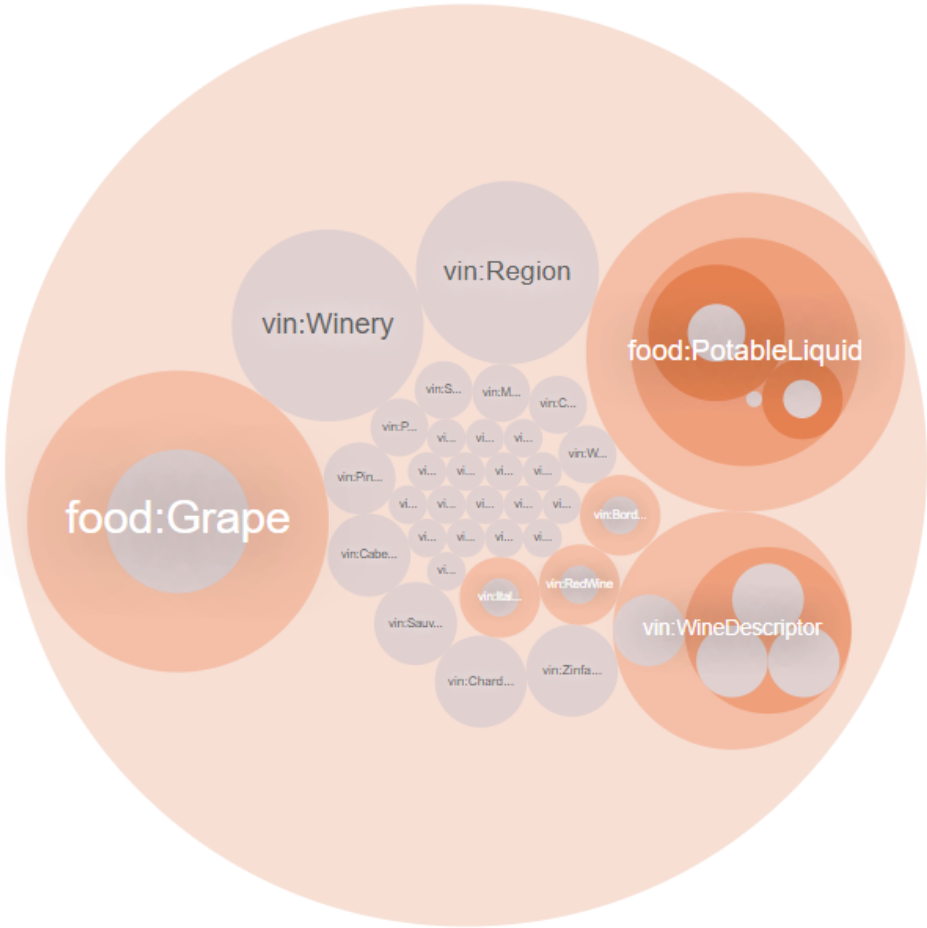
- To explore your data, navigate to *Explore -> Class hierarchy*.
- You can see a diagram depicting the hierarchy of the imported RDF classes by number of instances.
- The biggest circles are the parent classes and the nested ones are their children.

Class Hierarchy: Illustration

Class hierarchy

Class Count





Class Hierarchy: Illustration [Contd.]

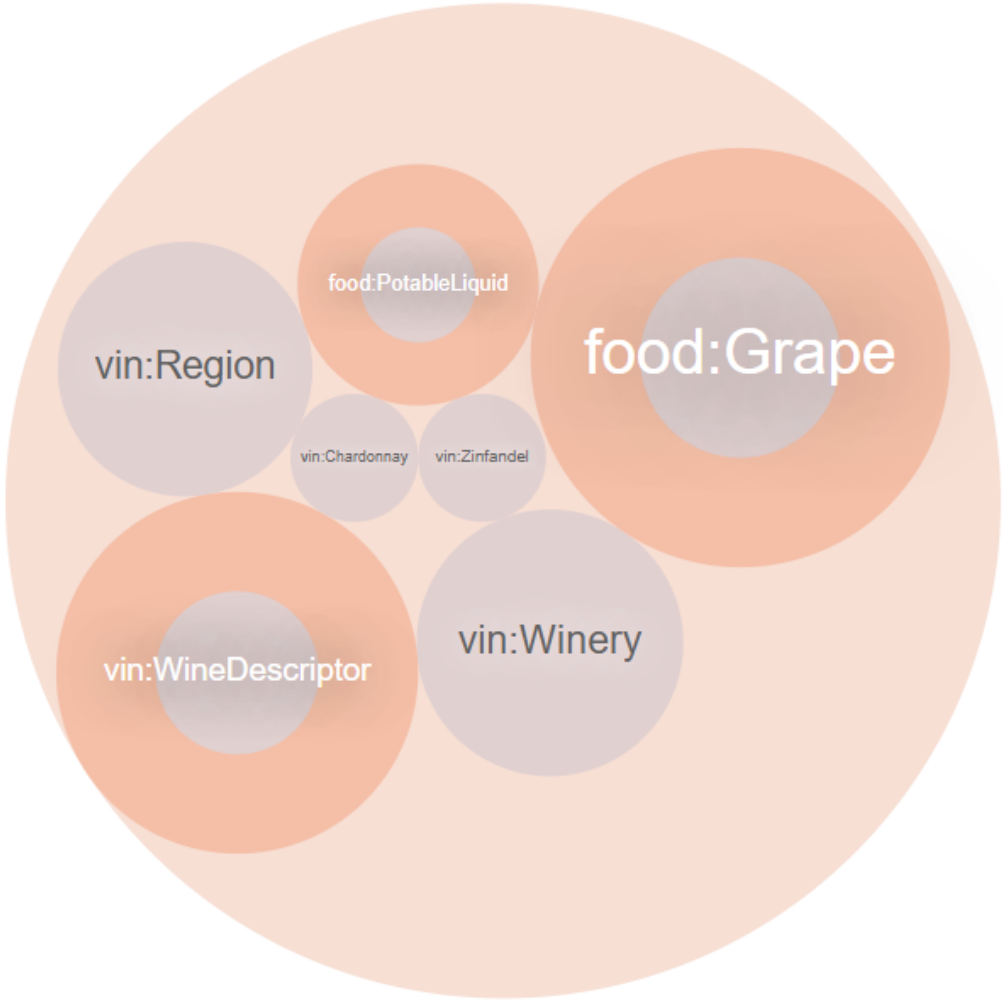
Class hierarchy

Class Count

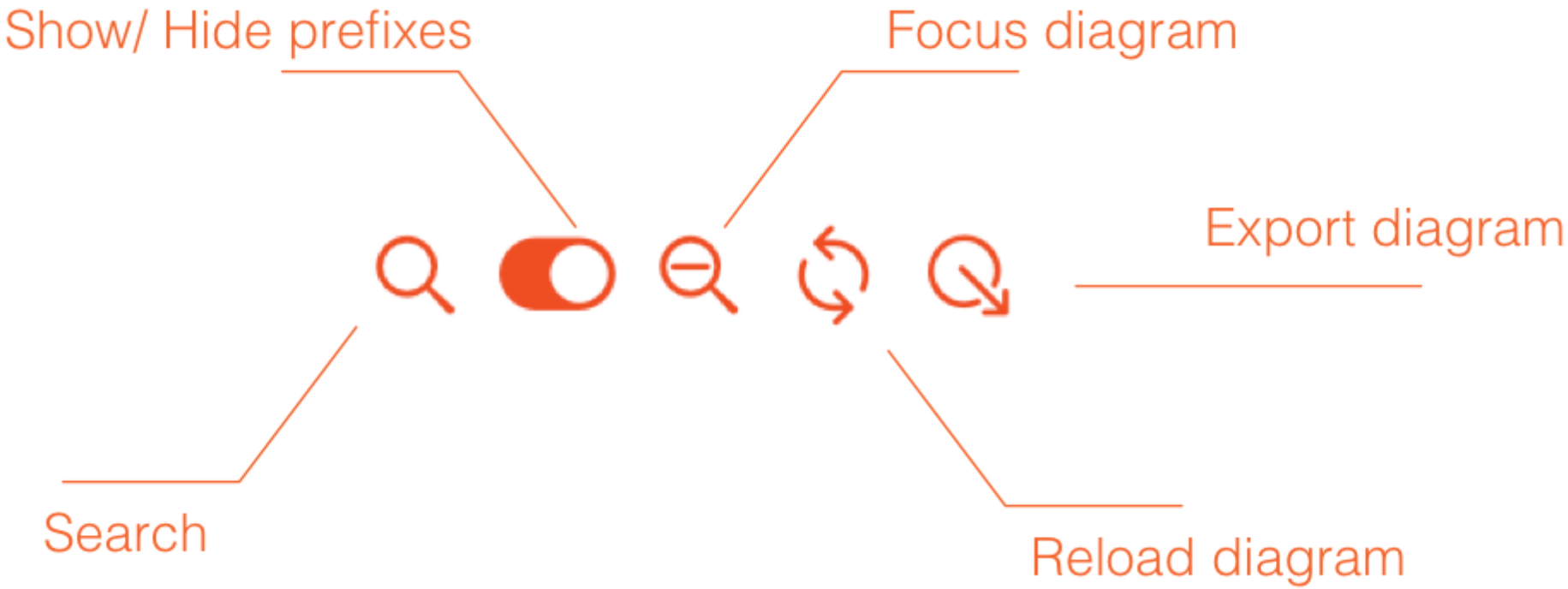
50

10

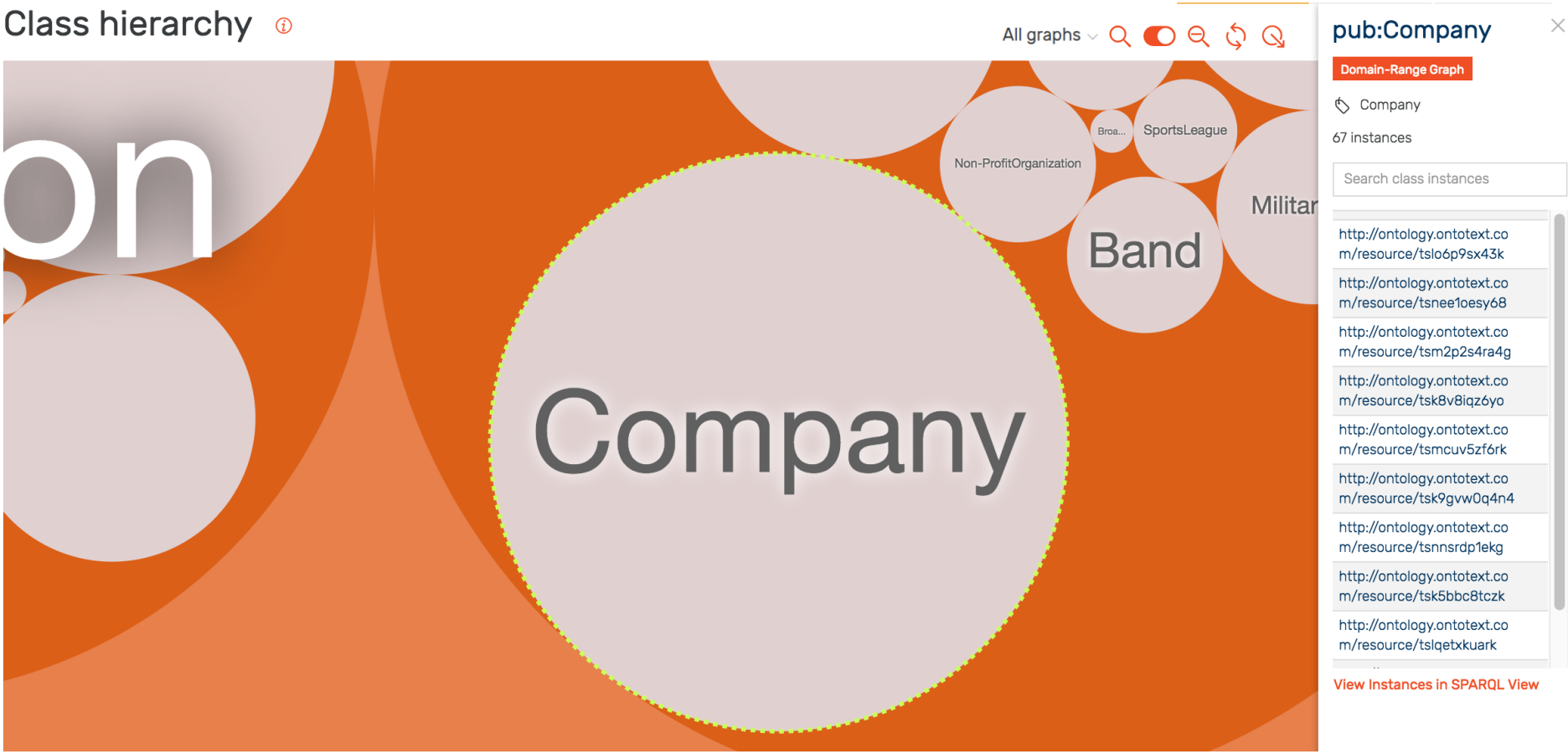
1



Class Hierarchy: Illustration [Contd.]



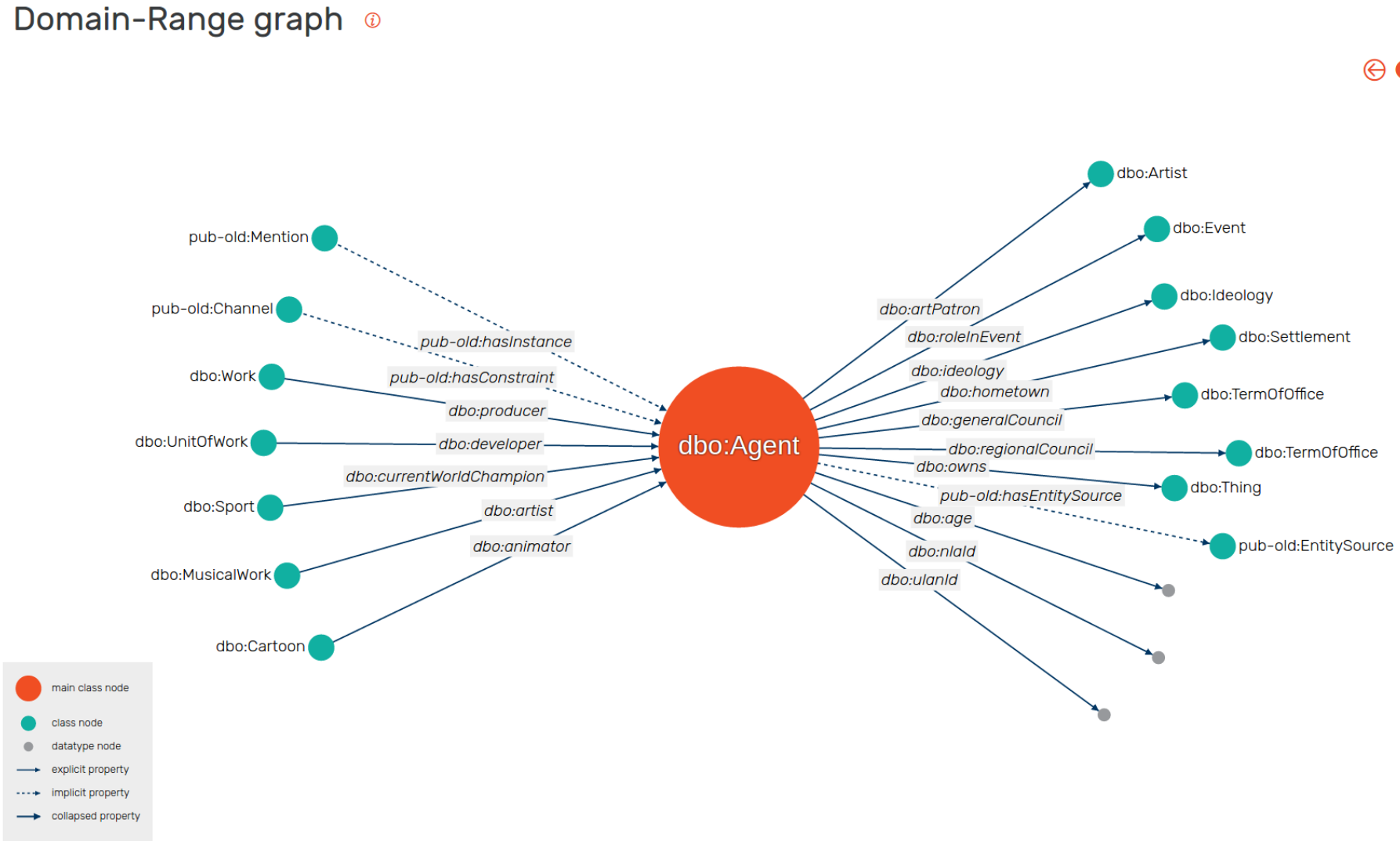
Class Hierarchy: Illustration [Contd.]



Domain-Range Graph

- To explore the connectedness of a given class, double click the class circle or the *Domain-Range Graph* button from the side panel.
- You can see a diagram that shows this class and its properties with their domain and range, where domain refers to all subject resources and range - to all object resources.
- You can also further explore the class connectedness by clicking:
  - the green nodes (object property class)
  - the labels - they lead to the View resource page, where you can find more information about the current class or property

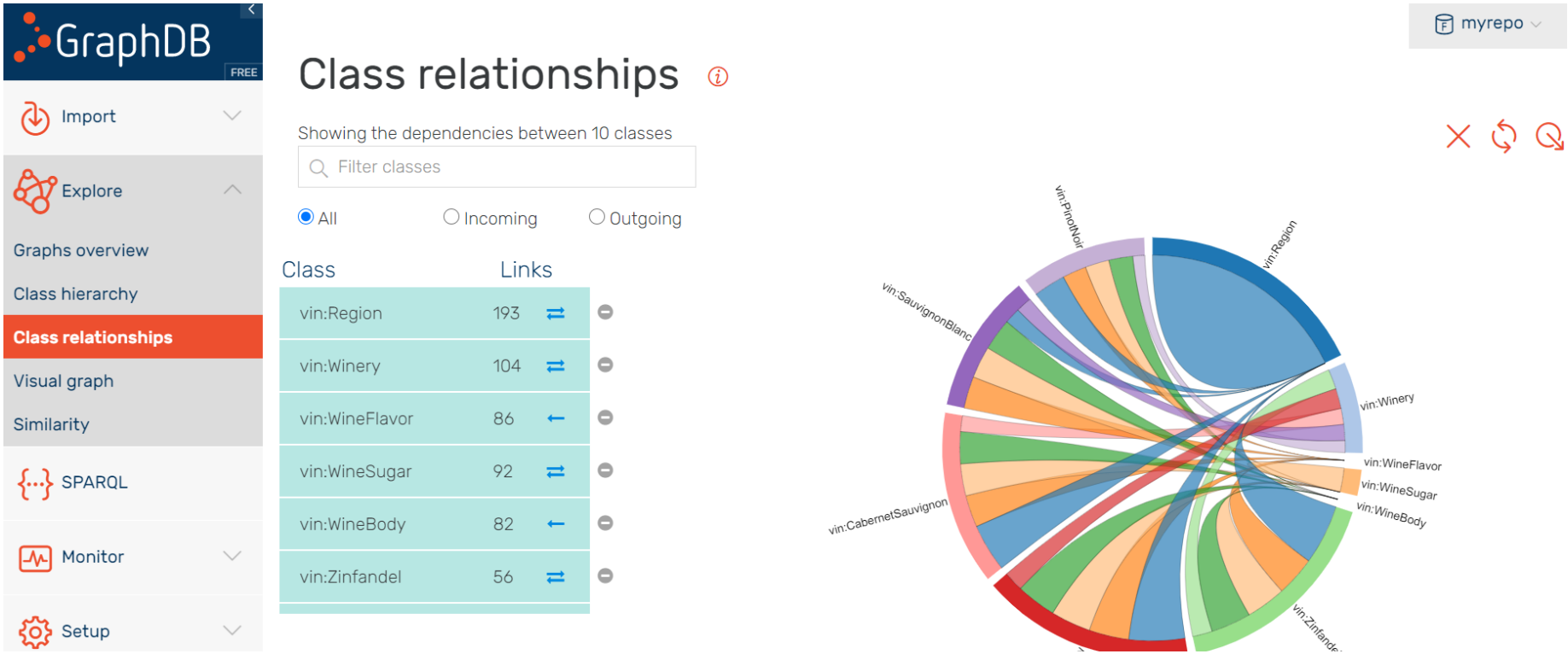
Domain-Range Graph: Illustration



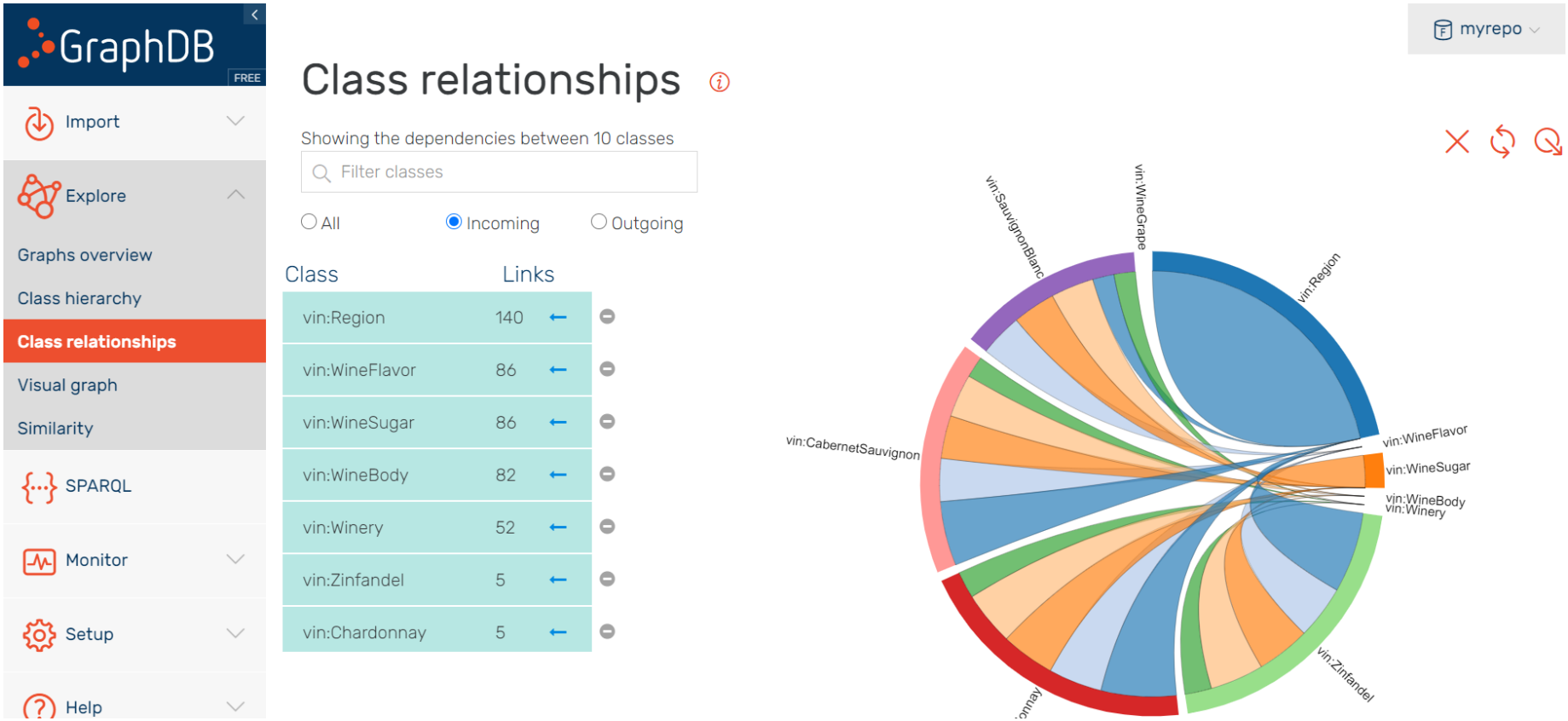
Class relationships

- To explore the relationships between the classes, navigate to *Explore -> Class relationships*.
- You can see a complicated diagram showing only the top relationships, where each of them is a bundle of links between the individual instances of two classes.
- Each link is an RDF statement, where the subject is an instance of one class, the object is an instance of another class, and the link is the predicate.
- Depending on the number of links between the instances of two classes, the bundle can be thicker or thinner and gets the color of the class with more incoming links.
- These links can be in both directions. To control which classes to display in the diagram, use the add/remove icon next to each class.

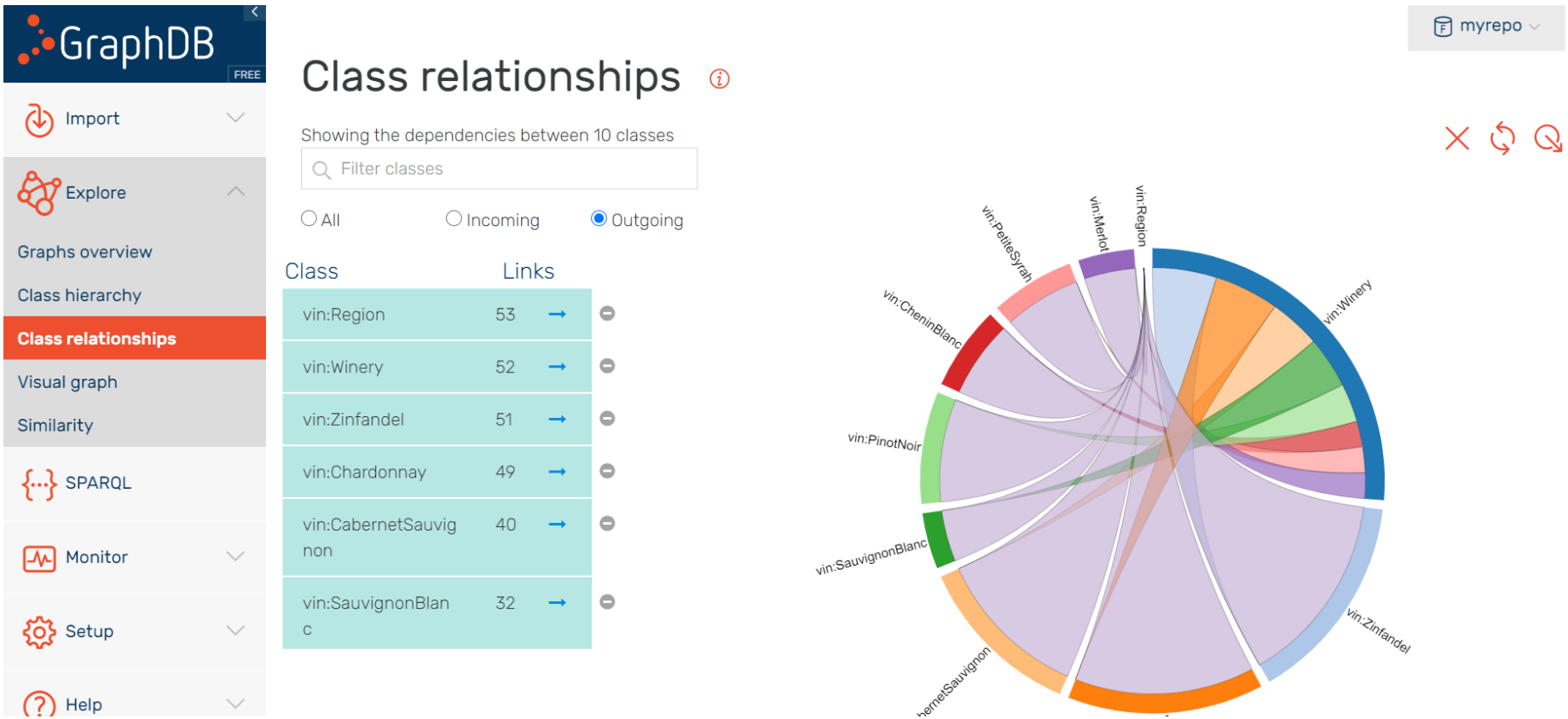
Class relationships: Illustration



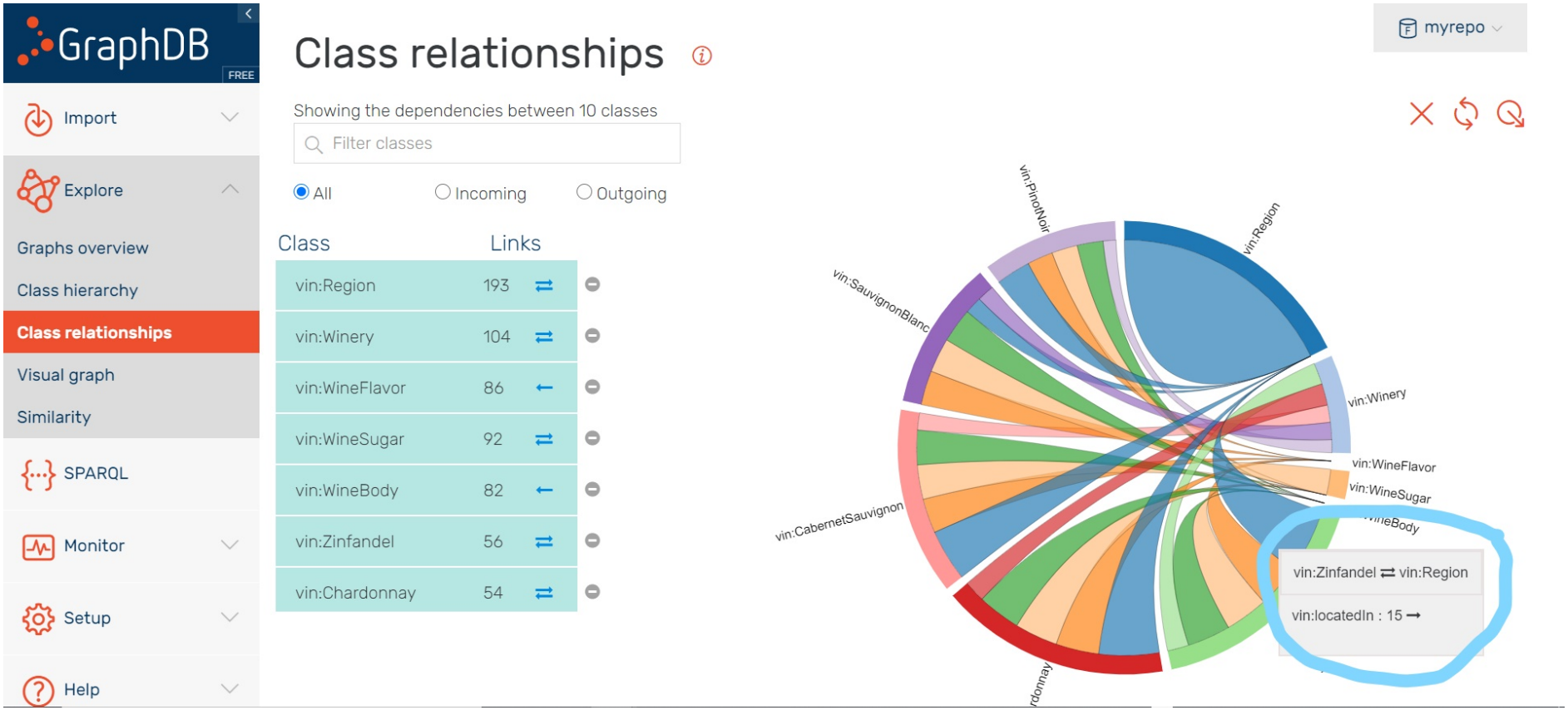
Class relationships: Illustration



Class relationships: Illustration



Class relationships: Illustration



SPARQL

SPARQL is a query language for RDF graph databases with the following types:

- **SELECT** - returns tabular results;
- **CONSTRUCT** - creates a new RDF graph based on query results;
- **ASK** -returns YES if the query has a solution, otherwise “NO”;
- **DESCRIBE** - returns RDF data about a resource;
- **INSERT** - inserts triples into a graph;
- **DELETE** - deletes triples from a graph (etc.)

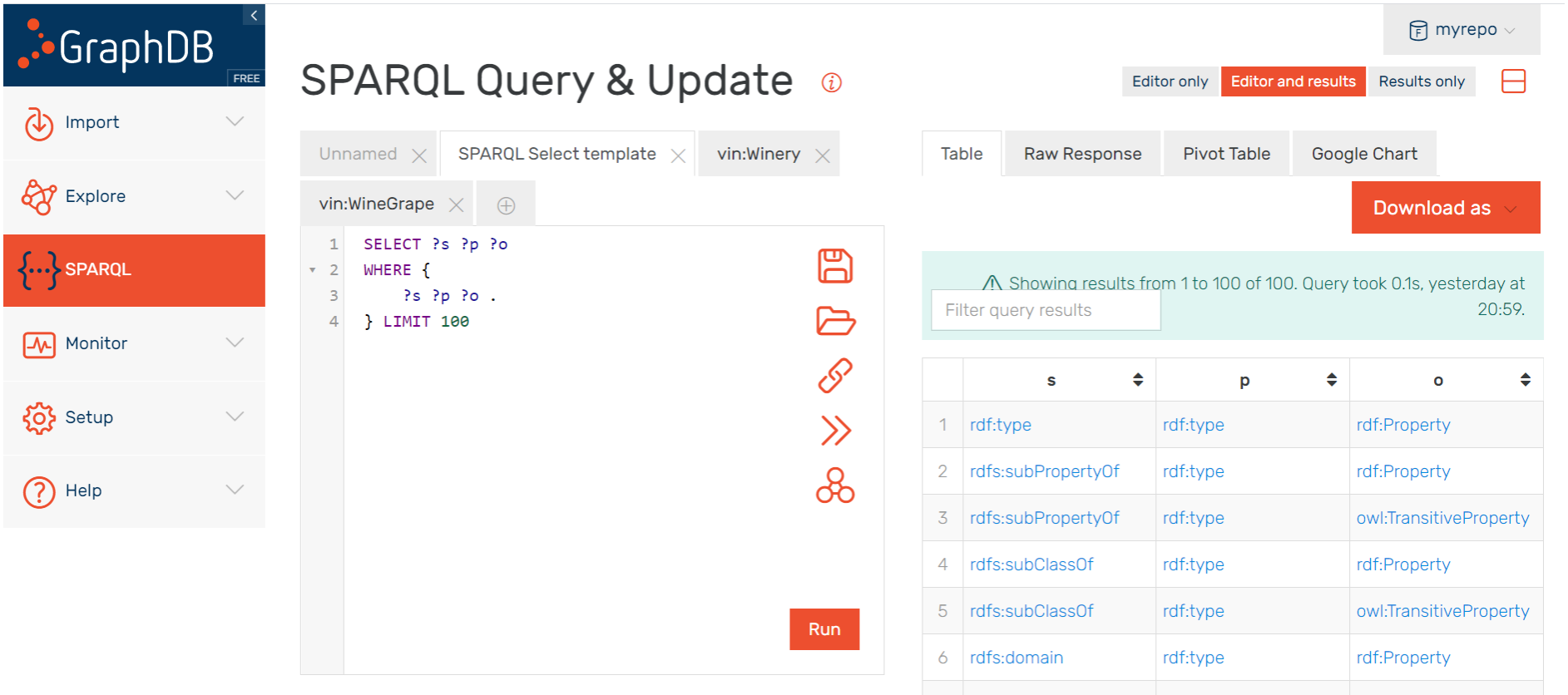
You have to use SPARQL to formalize CQs and explore the KG.

SPARQL

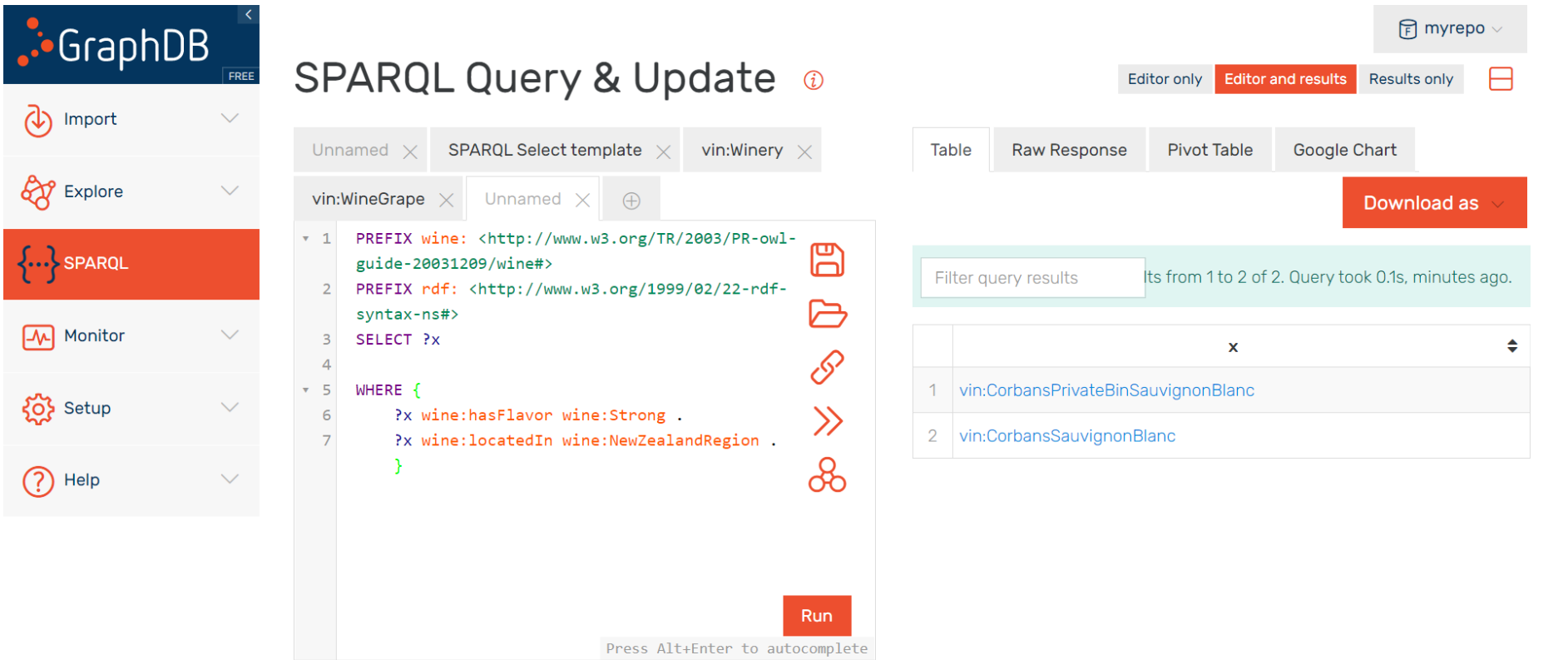
For learning SPARQL, please visit the following resources (**highly recommended**):

- SPARQL Introductory Tutorial
- How to Query RDFS SPARQL
- Complex Queries with SPARQL
- SPARQL 1.1 Query Language
- SPARQL playground

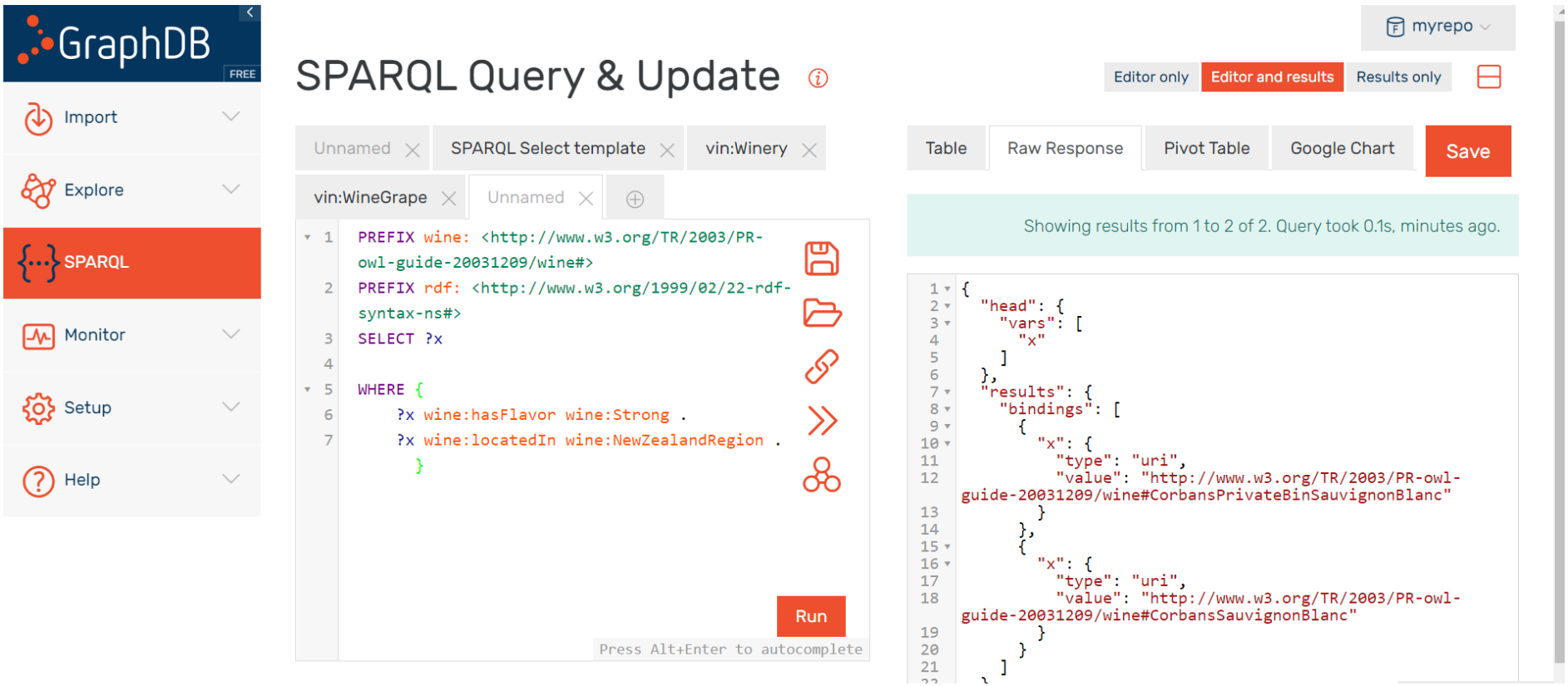
Query data through the Workbench: Illustration



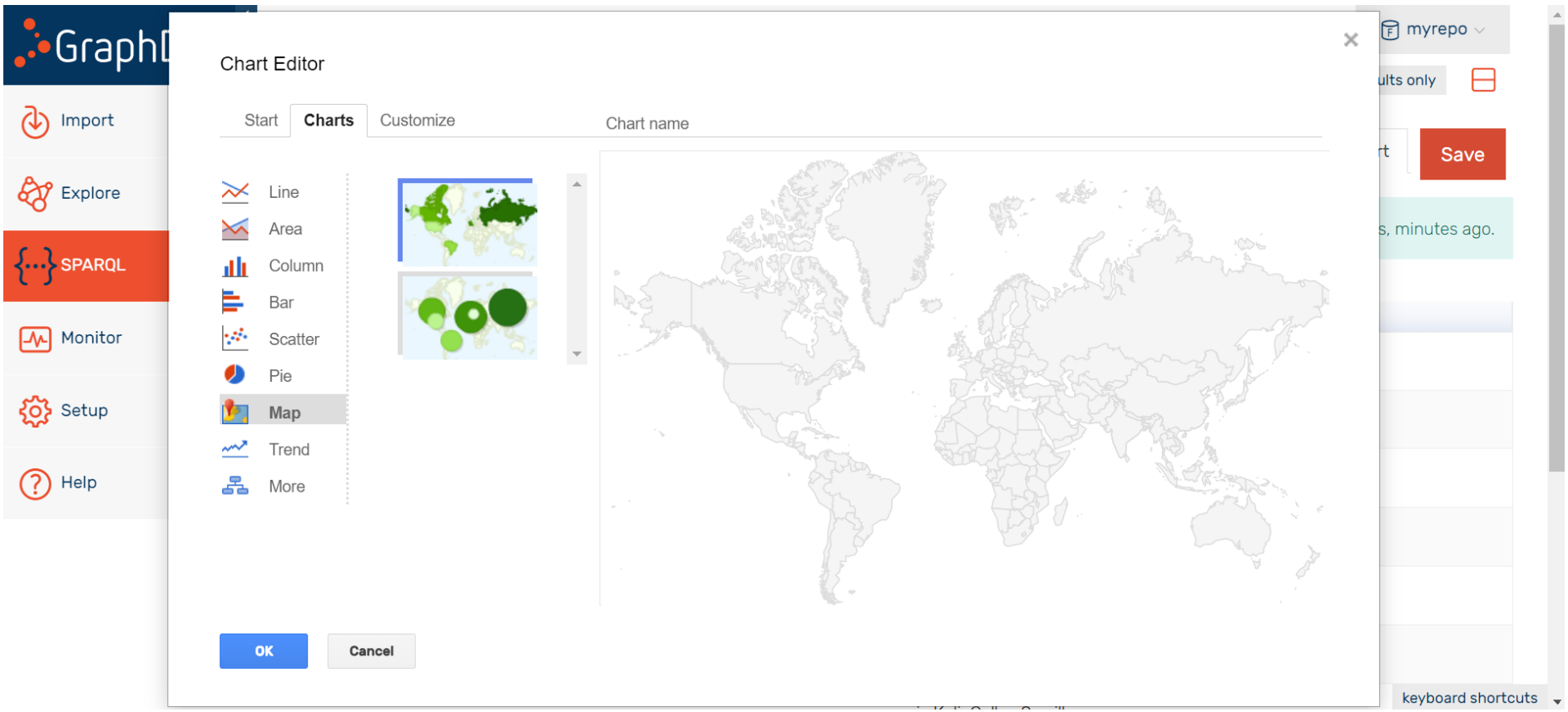
Query data through the Workbench: Illustration [Contd.]



Query data through the Workbench: Illustration [Contd.]



Query data through the Workbench: Illustration [Contd.]



References

Please visit the following reference resources (**highly recommended**):

- GraphDB Free Documentation
- GraphDB Fundamentals
- SPARQL in 11 minutes
- GraphDB Workbench Tutorials

Thank you for listening!

KDI : Knowledge and Data Integration

W10.L19.M6.T19.2.2

GraphDB