

5. Documentation

This chapter provides additional information about the application to users, administrators and programmers who might look to improve the application or would want to understand its inner workings better.

5.1 User documentation

This section describes how the user can connect to the application, utilize their solid pod and furthermore shows basic manipulation of the application data and view creation.

The reader is encouraged to navigate through the user guide¹ should they have any problems following specific steps setting up the solid environment.

5.1.1 Solid pod setup

This subsection describes the requirements necessary to enable the application to persist its data by utilizing the user's solid pod. While the application can be used without solid pods as illustrated in [section 2.4](#), no data and view management can be established, therefore severely limiting the functionality of the application. The user is encouraged to read through the documentation and follow all the steps outlined to familiarize themselves with the application and set up the environment properly.

Preliminaries

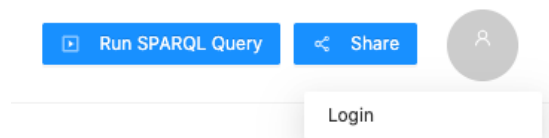
The following sections are written for Solid pods hosted at Inrupt². This application shall be usable with any Solid pod providing service but the details regarding its use might differ.

We assume the reader has their own Solid pod set up, if not, they are able to create a new one for free on the registration page³.

Logging in with Solid POD

To set up all permissions for the application, the user has to sign in to their Solid pod first via the application's avatar button in the top right corner.

Figure 5.1: Avatar menu



Upon clicking on the login button, the user is required to choose their solid pod provider either from the list provided or by specifying it themselves.

¹<https://github.com/solid/userguide>

²<https://inrupt.net/>

³<https://inrupt.net/register>

Log in to **jaresan.github.io**

After the provider is chosen, the user is able to authenticate via the provider's login screen.

Login

When the login is successful, the user is requested to grant permission to the application which would allow it to save and handle its data across the user's solid pod.

Authorize <https://jaresan.github.io/simplod/> to access your Pod?

By clicking Authorize, any app from <http://localhost:3000> will be able to:

- ☒ Read all documents in the Pod
- ☒ Add data to existing documents, and create new documents
- ☒ Modify and delete data in existing documents, and delete documents
- ☐ Give other people and apps access to the Pod, or revoke their (and your) access

If all of the above steps resolve correctly, the user is shown a positive feedback message and able to start working with the application fully.

With all of the steps outlined above complete, the application shall have all the permissions it needs to properly manage its data utilizing the user's solid pod.

Should the reader experience problems with the application data management, they are encouraged to resolve the problems manually directly in their Solid pod, as outlined in [Appendix A](#).

5.1.5 Examples

The following subsection outlines example scenarios which can be followed to introduce the user to the features of the application.

Every example is started from the new start of the application with **default settings**. The reader is welcome to use the accompanying deployment of this work on github⁵ by clicking **Demo**⁶

Nobel prize categories - graph

The first example is based on a data schema of Nobel prizes. This data schema represents the information about Nobel prizes and their laureates. Let's illustrate a simple example where we would like to know what Nobel prize categories there are.

First we have to load the data schema in the application, we can do that either by accessing the demo application⁷ with the data already encoded at or by following the steps below:

1. Click *File* → *New*
2. Create a New project with the following configuration:

Data Schema:

`https://jaresan.github.io/simplod/examples/nobel_prizes.ttl`

Endpoint: `http://data.nobelprize.org/sparql`

3. Click Create

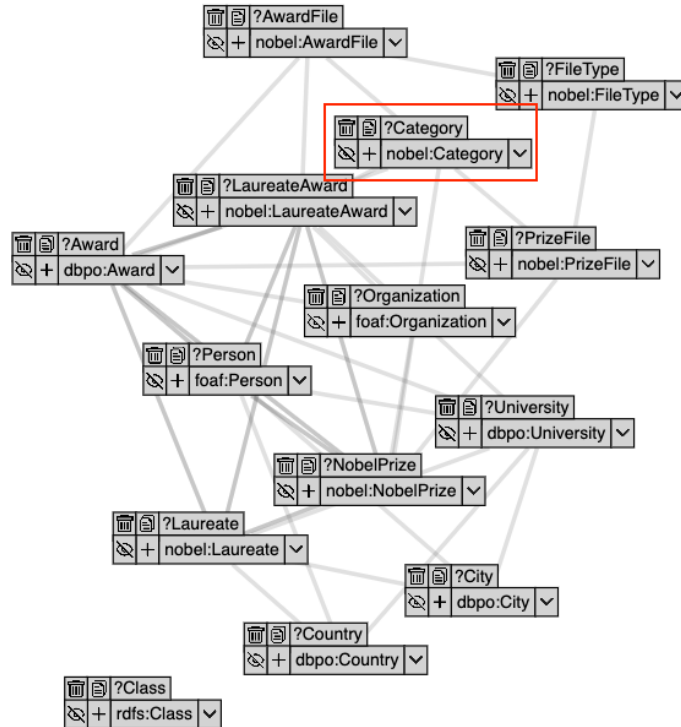
⁵`https://jaresan.github.io/simplod/`

⁶`https://jaresan.github.io/simplod/build/index.html`

⁷`https://jaresan.github.io/simplod/demo.html?schemaURL=https%3A%2F%2Fjaresan.github.io%2Fsimplod%2Fexamples%2Fnobel_prizes.ttl&endpointURL=http%3A%2F%2Fdata.nobelprize.org%2Fsparql`

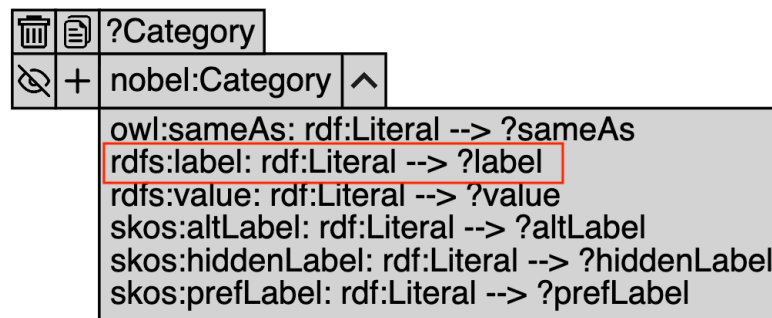
`https://jaresan.github.io/simplod/demo.html?schemaURL=https%3A%2F%2Fjaresan.github.io%2Fsimplod%2Fexamples%2Fnobel_prizes.ttl&endpointURL=http%3A%2F%2Fdata.nobelprize.org%2Fsparql`

Figure 5.41: Nobel prize example default view



When we open the application (depicted in [Figure 5.41](#)) we can notice the "Category" entity in the graph.

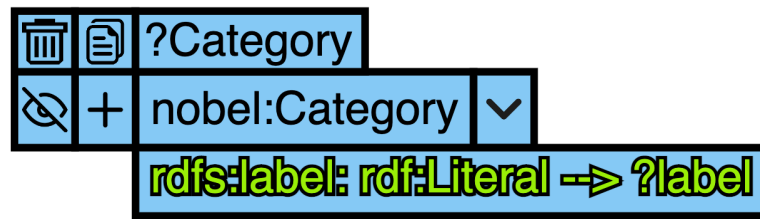
Figure 5.42: Category entity



Clicking on the category entity, we can expand its properties and see what it is linked to. Since we are looking for a textual description of the category, we will most likely be interested in the **rdfs:label**⁸ property. Let's go ahead and select it.

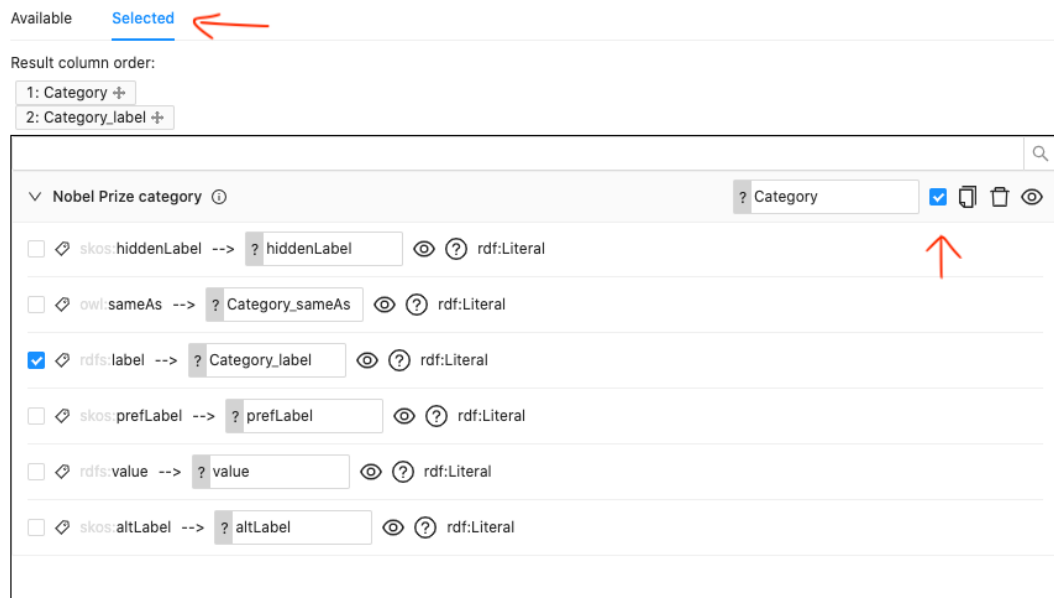
⁸https://www.w3.org/TR/rdf-schema/#ch_label

Figure 5.43: Selected property



For a more detailed overview of what we have selected, we can take a look at the list view under the graph or next to it in the tab **selected**.

Figure 5.44: List view



Here we can see we have selected the label property, named **Category_label**. Also the entity itself is selected (checkbox next to **Category**), which will provide us with the actual category IRI upon executing the query. Clicking the "Run SPARQL Query" at the top of the screen, the editor opens and we can see the fetched results:

Figure 5.45: Selected property

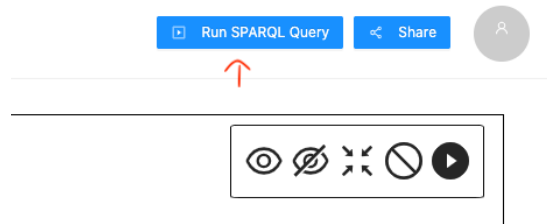





Figure 5.46: SPARQL Results



Query  





 http://data.nobelprize.org/sparql

```

1 PREFIX nobel: <http://data.nobelprize.org/terms/>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 SELECT DISTINCT ?Category ?label WHERE {
5   ?Category a nobel:Category.
6   ?Category rdfs:label ?label.
7 }

```

 Table  Response 6 results in 0.108 seconds Page size: 50  

	Category	label
1	<http://data.nobelprize.org/resource/category/Physiology_or_Medicine>	Physiology or Medicine
2	<http://data.nobelprize.org/resource/category/Literature>	Literature
3	<http://data.nobelprize.org/resource/category/Economic_Sciences>	Economic Sciences
4	<http://data.nobelprize.org/resource/category/Peace>	Peace
5	<http://data.nobelprize.org/resource/category/Physics>	Physics
6	<http://data.nobelprize.org/resource/category/Chemistry>	Chemistry




Showing 1 to 6 of 6 entries < 1 >


In [Figure 5.46](#) we can see the result created by our selection. First we have the **Category** which represents the actual IRI of the entity. Second we have the **Category_label** property, which is a textual representation of the entity itself, its label.


Considering we wanted to find what categories Nobel prizes are awarded in, this result suffices. However, should we want to just list the labels and display them somewhere, the entity IRIs would probably be unnecessary. We can go back to the list view to deselect them:


Figure 5.47: List view


Result column order:
1: Nobel_prize_category +


▼ Nobel Prize category ⓘ ? Category   


☐  skos:hiddenLabel --> ? hiddenLabel ⓘ ? rdf:Literal

☐  owl:sameAs --> ? Category_sameAs ⓘ ? rdf:Literal

☒  rdfs:label --> ? Nobel_prize_categ ⓘ ? rdf:Literal

☐  skos:prefLabel --> ? prefLabel ⓘ ? rdf:Literal

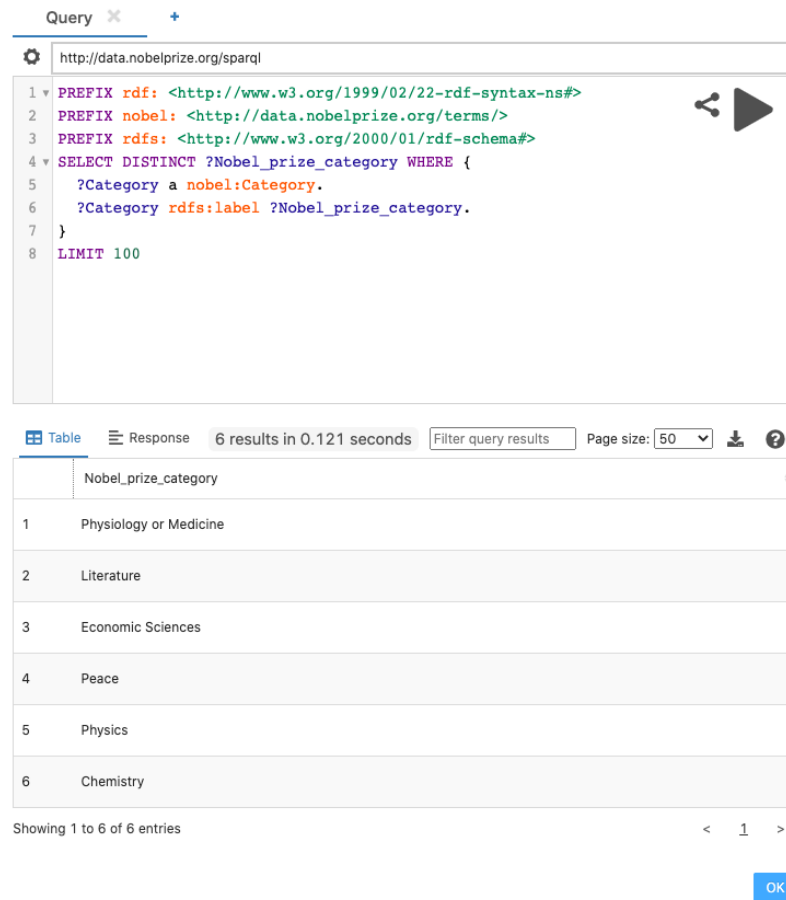
☐  rdfs:value --> ? value ⓘ ? rdf:Literal

☐  skos:altLabel --> ? altLabel ⓘ ? rdf:Literal

In [Figure 5.47](#) we deselect the entity IRI by unchecking the box on the entity row. We can also rename the rdfs:label property from **Category_label** to

Nobel_prize_category. Executing the query again, we get the following:

Figure 5.48: Cleaner results



The screenshot shows a query tool interface. At the top, there's a "Query" tab with a URL bar containing "http://data.nobelprize.org/sparql". Below the URL bar is a text area with a SPARQL query:

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX nobel: <http://data.nobelprize.org/terms/>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 SELECT DISTINCT ?Nobel_prize_category WHERE {
5   ?Category a nobel:Category.
6   ?Category rdfs:label ?Nobel_prize_category.
7 }
8 LIMIT 100
```

Below the query area, there are icons for "Table", "Response", and a status bar indicating "6 results in 0.121 seconds". There's also a "Filter query results" button and a "Page size: 50" dropdown. The results are displayed in a table with the following data:

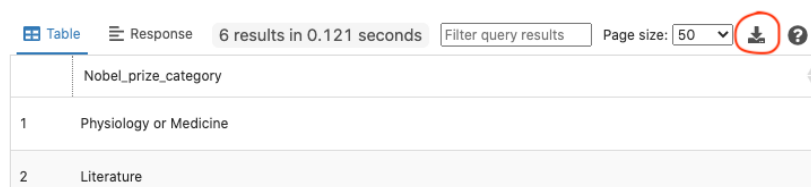
	Nobel_prize_category
1	Physiology or Medicine
2	Literature
3	Economic Sciences
4	Peace
5	Physics
6	Chemistry

At the bottom, there's a "Showing 1 to 6 of 6 entries" message and a blue "OK" button.

In [Figure 5.49](#) we can see that we have now only fetched the categories in a human readable format. Anybody shown these results can immediately understand what they represent.

Now that we've fetched the data, we might want to share them. We can do so by downloading the result directly in the CSV format and sharing that file:

Figure 5.49: Cleaner results



This screenshot is identical to the one in Figure 5.48, but with a red circle highlighting the download icon (a downward arrow) in the top right corner of the interface, next to the "Page size: 50" dropdown.



We can also share the data by sharing a link to a third party tool populated with our query. To do that, we can open the share menu via the "Share" button at the top of the screen. By clicking on the  icon for **YASGUI Query Tool**, we copy the URL with the query encoded into our clipboard and can then just paste it in the browser and view the result. We can also click the  icon to launch the tool directly.

Figure 5.50: Copy yasgui query URL

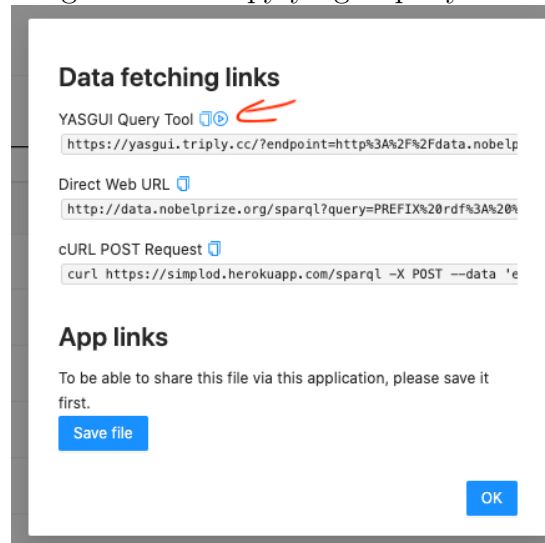
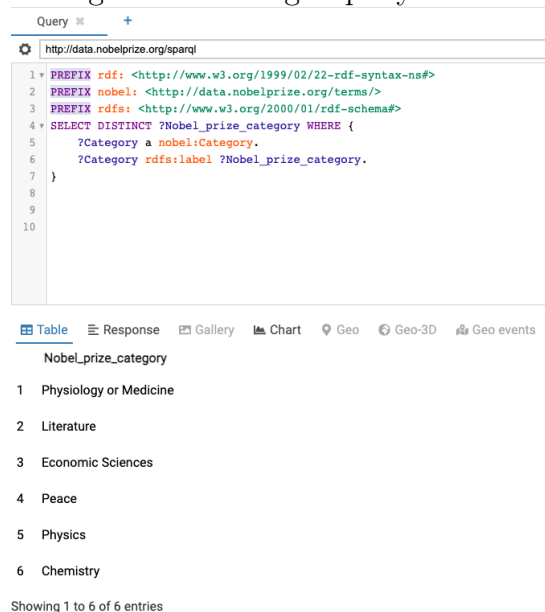


Figure 5.51: Yasgui query results



We can also use "Direct access URL" which returns the results directly, or get the cURL POST request to use in the terminal. All of the share options are described in [Figure 5.1.2](#).

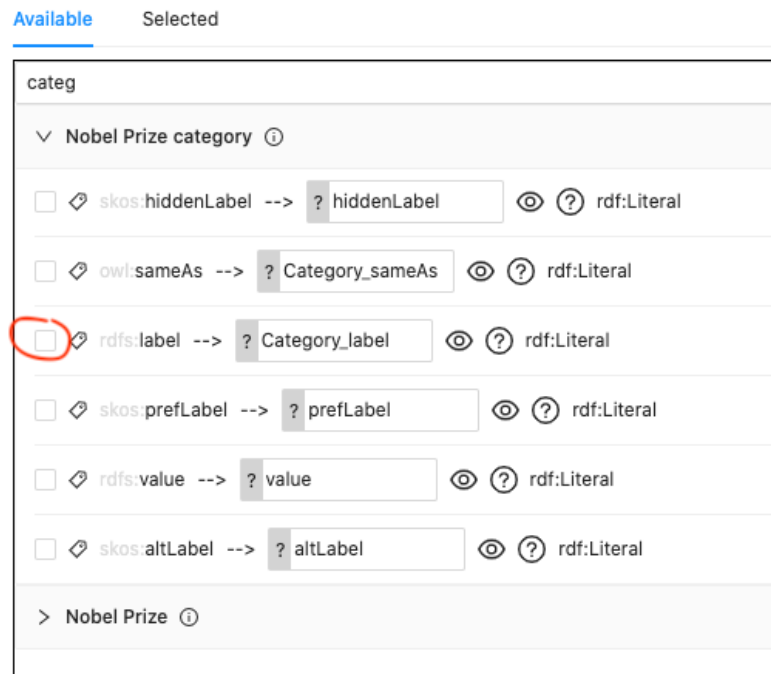
Nobel prize categories - list view

What if we don't want to navigate through the graph because it might seem too clunky?

We can use the search functionality in the list view. We are looking for category. By typing "category" in the search field, we can see entity rows being filtered out based on their matching text. When we type "categ", we can see the top result is "Nobel Prize category" which is what we are looking for. Same

way as in the graph, here we can see **rdfs:label** which is the property we are interested in.

Figure 5.52: List view



In this case, we can select the property directly by checking the box on its left side. The rest of the steps is the same as the end for the graph variant. The graph and list are connected and new changes are reflected in both of these components.

Nobel prize laureates

Continuing with the example of nobel prizes, let's try an example where we'd like to get nobel laureates with some additional info about them. Let's begin with the default view by following the same steps as previously, either accessing the example directly⁹ or by following the steps below:

1. Click *File* → *New*
2. Create a New project with the following configuration:

Data Schema:

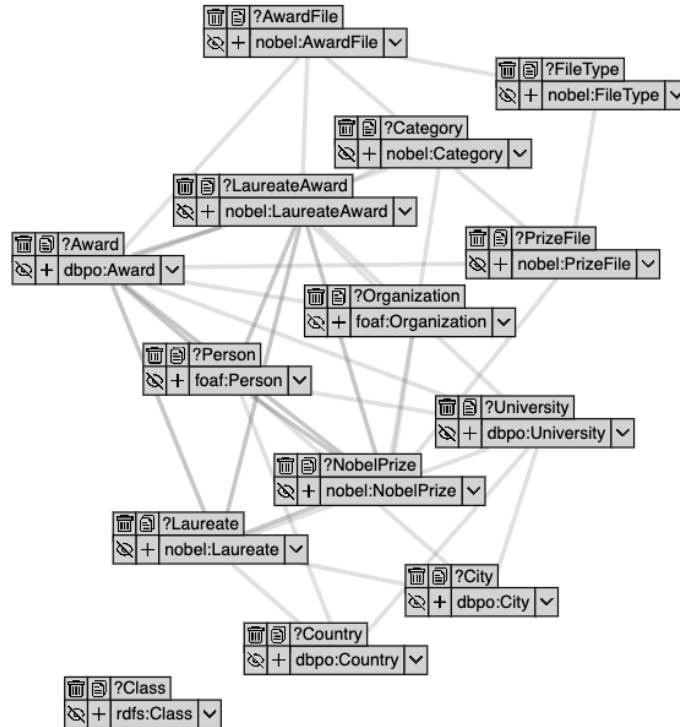
https://jaresan.github.io/simplod/examples/nobel_prizes.ttl

Endpoint: <http://data.nobelprize.org/sparql>

3. Click Create

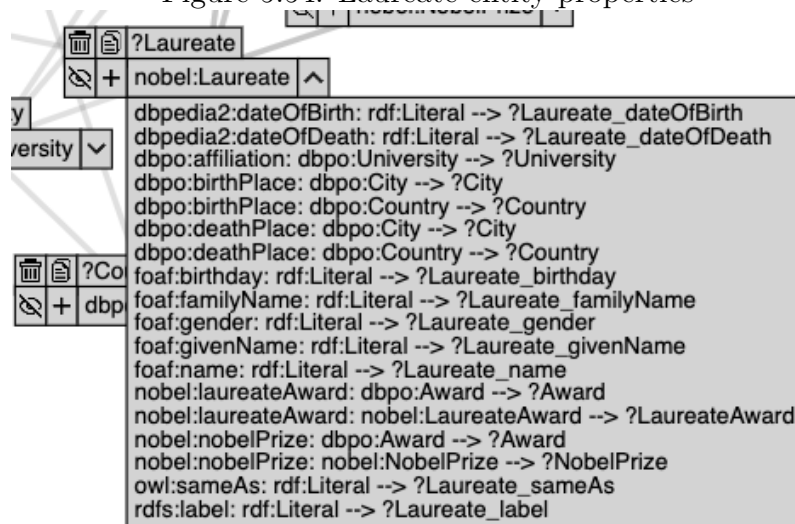
⁹https://jaresan.github.io/simplod/demo.html?schemaURL=https%3A%2F%2Fjaresan.github.io%2Fsimplod%2Fexamples%2Fnobel_prizes.ttl&endpointURL=http%3A%2F%2Fdata.nobelprize.org%2Fsparql

Figure 5.53: Nobel prize example default view



Since we are interested in nobel prize laureates, we can click the **Laureate** entity to see what relationships there are:

Figure 5.54: Laureate entity properties



Let's say we're interested in the laureate's birth country, their name and additional information about the prize they received, from the properties, the corresponding ones would be **dbpo:birthPlace** (**dbpo:Country**), **foaf:name** and **nobel:nobelPrize** (**nobel:NobelPrize**) respectively. The selection would look as follows:

Figure 5.56: All properties of interest

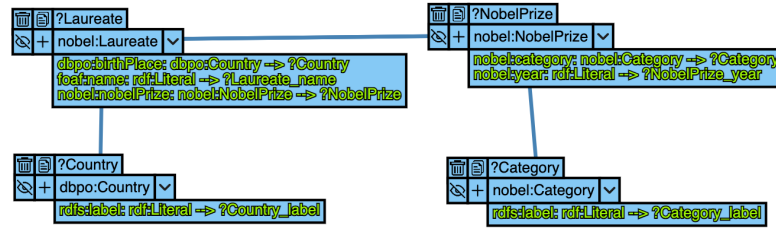
The graph illustrates a network of entities and their relationships. Key entities and their associated properties are as follows:

- ?File** (grey): `nobel:FileType`
- ?Category** (blue): `nobel:Category`, `rdflabel: rdfs:Literal => ?Category_label`
- ?Organization** (grey): `foaf:Organization`
- ?Award** (grey): `dbpo:Award`
- ?PrizeFile** (grey): `nobel:PrizeFile`
- ?NobelPrize** (blue): `nobel:NobelPrize`, `nobel:category: nobel:Category => ?Category`, `nobel:year: rdfs:Literal => ?NobelPrize_year`
- ?Laureate** (blue): `nobel:Laureate`, `dbpo:birthPlace: dbpo:Country => ?Country`, `foaf:name: rdfs:Literal => ?Laureate_name`, `nobel:nobelPrize: nobel:NobelPrize => ?NobelPrize`
- ?Country** (blue): `dbpo:Country`, `rdflabel: rdfs:Literal => ?Country_label`
- ?Class** (grey): `rdfs:Class`

Other entities and properties shown include `?File`, `?Category`, `?Organization`, `?Award`, `?PrizeFile`, `?NobelPrize`, `?Laureate`, `?Country`, and `?Class`. The graph also includes various literals and labels, such as `rdflabel: rdfs:Literal => ?Category_label`, `rdflabel: rdfs:Literal => ?Country_label`, `nobel:category: nobel:Category => ?Category`, `nobel:year: rdfs:Literal => ?NobelPrize_year`, `dbpo:birthPlace: dbpo:Country => ?Country`, `foaf:name: rdfs:Literal => ?Laureate_name`, and `nobel:nobelPrize: nobel:NobelPrize => ?NobelPrize`.

After hiding the data, let's arrange the nodes better by moving them around. After cleaning the graph up a bit, we get something that could look as follows:

Figure 5.57: Cleaned up selection




This is already a valid selection. Using the graph quick action toolbar in the top right, we can hit the  button and execute the query for the following results (the order might differ based on the specific order of selecting the properties):

Figure 5.58: Execute query button



Figure 5.59: Result set with IRIs

	Country	Category	Laureate	NobelPrize	Laureate_name	Country_label	NobelPrize_year	Category_label
1	<http://data.nobelprize.org/resource/country/Argentina>	<http://data.nobelprize.org/resource/category/Physiology_or_Medicine>	<http://data.nobelprize.org/resource/laureate/431>	<http://data.nobelprize.org/resource/nobelprize/Physiology_or_Medicine/1984>	César Milstein	Argentina	"1984" <http://www3.org/2001/XMLSchema#integer>	Physiology or Medicine
2	<http://data.nobelprize.org/resource/country/Germany>	<http://data.nobelprize.org/resource/category/Physiology_or_Medicine>	<http://data.nobelprize.org/resource/laureate/430>	<http://data.nobelprize.org/resource/nobelprize/Physiology_or_Medicine/1984>	Georges J.F. Köhler	Germany	"1984" <http://www3.org/2001/XMLSchema#integer>	Physiology or Medicine
3	<http://data.nobelprize.org/resource/country/United_Kingdom>	<http://data.nobelprize.org/resource/category/Physiology_or_Medicine>	<http://data.nobelprize.org/resource/laureate/429>	<http://data.nobelprize.org/resource/nobelprize/Physiology_or_Medicine/1984>	Niels K. Jerne	United Kingdom	"1984" <http://www3.org/2001/XMLSchema#integer>	Physiology or Medicine

While this query is valid, a person interested in this information might not want the IRIs to be present in the result set, the textual representation provided via labels might be sufficient. To get rid of the IRIs in the result set, we have to deselect the entities via the list view as follows:

Figure 5.60: Selected entities













> country	? Country	<input checked="" type="checkbox"/>			
> Nobel Prize category ⓘ	? Category	<input checked="" type="checkbox"/>			
> Laureate ⓘ	? Laureate	<input checked="" type="checkbox"/>			
> Nobel Prize ⓘ	? NobelPrize	<input checked="" type="checkbox"/>			

Figure 5.61: Deselected entities

> country	? Country	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
> Nobel Prize category ⓘ	? Category	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
> Laureate ⓘ	? Laureate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
> Nobel Prize ⓘ	? NobelPrize	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

This effectively removes the IRIs from the query and displays only the queried properties (in this case the labels we selected). The result is more concise and shorter:

Figure 5.62: Result set without IRIs

906 results in 0.576 seconds					
	Laureate_name	Country_label	NobelPrize_year	Category_label	
1	César Milstein	Argentina	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	
2	Georges J.F. Köhler	Germany	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	
3	Niels K. Jerne	United Kingdom	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	

Nobel prize laureates - part 2

We have retrieved information about Nobel prize laureates and about the awards they received. Let's extend the search by querying for the places where the laureates passed away.

Checking [Figure 5.54](#), we can see there are two properties of interest, namely **deathPlace** (**dbpo:Country**) and **dateOfDeath**. Let's query for **deathPlace** (**dbpo:Country**) then. Understandably this will result in a data set of **only deceased laureates**, since laureates with no such property will be omitted from the result set, as the property is marked as required, not optional. Marking the property as optional would allow to search for both living and deceased laureates with place of death filled in where applicable.

Updated selection and the results with death place under **Country_label** would look as follows:

Figure 5.63: Selection with place of death

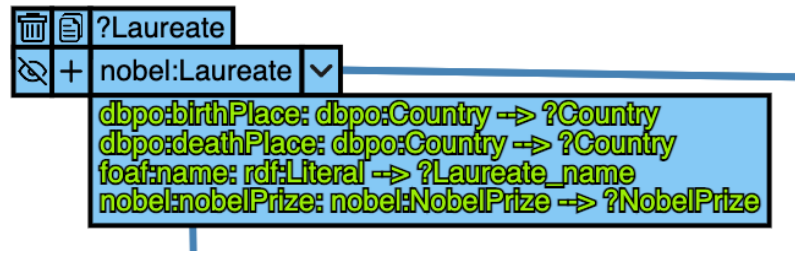



Figure 5.64: Results with place of death

Table Response 362 results in 0.416 seconds

	Laureate_name	Country_label	NobelPrize_year	Category_label
1	Georges J.F. Köhler	Germany	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine
2	Daniel Nathans	USA	"1978"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine
3	Alexis Carrel	France	"1912"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine

You might notice the new result set doesn't contain more columns than [Figure 5.62](#). This is because we are querying for a **single country**. Our query actually translates to **find laureates who were born and died in the same country** due to the links/edges being pointed to the same **Country** entity. While this is not an invalid query and can have its uses, it is not what we are looking for. This is where we need to use the  icon on the **Country** node in the graph (or use the same on in the list view) and create a separate entity instance for **Country** to introduce a distinction between the death place and birth place.


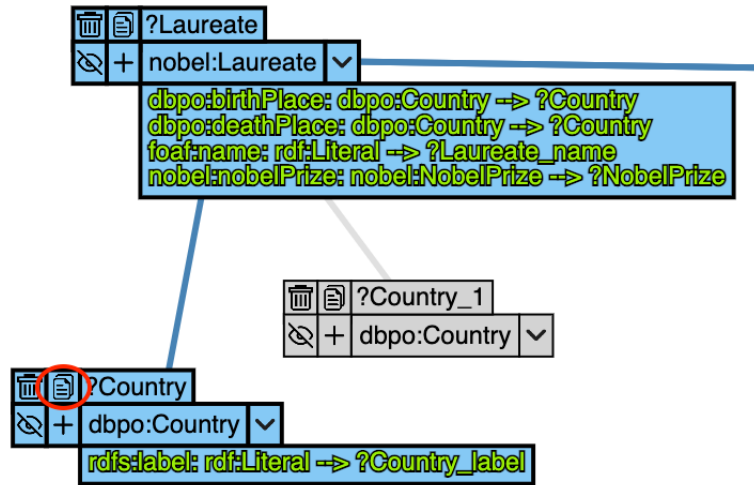
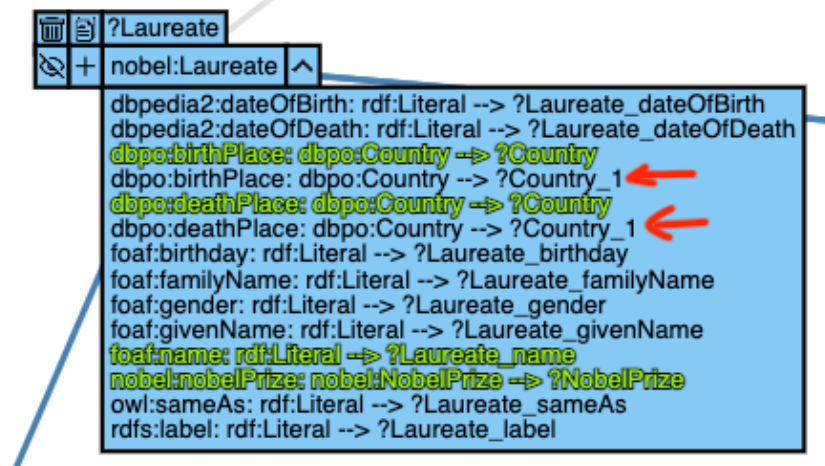
Clicking the  icon on the **Country** entity, we get the following:

Figure 5.65: Selection with cloned Country entity



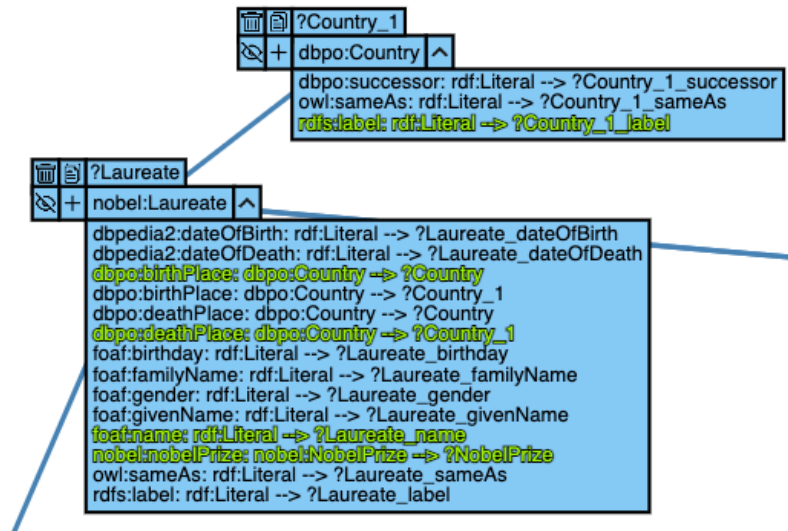
By copying the node, the **Laureate** entity has new properties added that target the new **Country** entity:

Figure 5.66: Newly listed properties



Next we just have to remove the old **deathPlace** and use the new one:

Figure 5.67: Proper selection



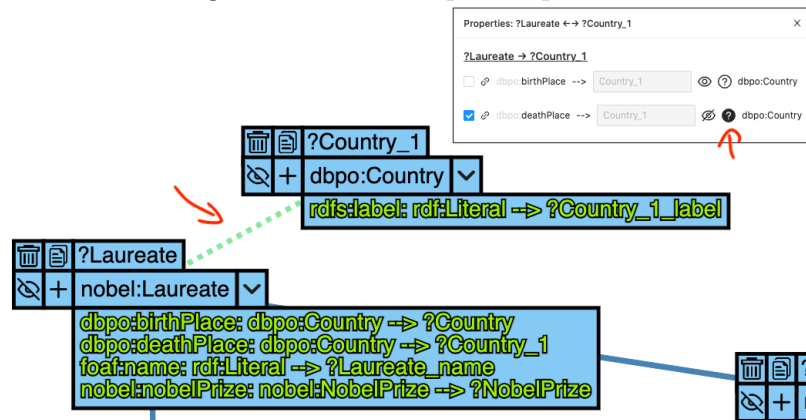
We again pick **label** on the newly copied **Country_1** and remove the IRI specification by deselecting the checkbox in the list view for **Country_1**. Executing the query via **▶** now yields results even if the laureate wasn't born and died in the same country:

Figure 5.68: Proper death place results

Table Response 604 results in 0.849 seconds					
	Laureate_name	Country_label	NobelPrize_year	Category_label	Country_1_label
1	César Milstein	Argentina	"1984"***<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	United Kingdom
2	Georges J.F. Köhler	Germany	"1984"***<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	Germany
3	Niels K. Jerne	United Kingdom	"1984"***<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	France

If we were looking for all laureates (living and dead) and just adding the information of their death place if it exists, we could mark the **deathPlace** property as optional via the list view or clicking the edge:

Figure 5.69: Death place optional



Which in turn returns all laureates, living and dead, with the country of their death if it's specified:

Figure 5.70: Results with death place optional

Table Response 100 results in 0.137 seconds

	Laureate_name	Country_label	NobelPrize_year	Category_label	Country_1_label
1	César Milstein	Argentina	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	United Kingdom
2	Georges J.F. Köhler	Germany	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	Germany
3	Niels K. Jerne	United Kingdom	"1984"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	France
4	Hamilton O. Smith	USA	"1978"^^<http://www.w3.org/2001/XMLSchema#integer>	Physiology or Medicine	

Another nice example would be to query only for **living laureates**. However, such an example would require the ability to constrain the values of properties, which is not a feature implemented in this work.

German books - Save & Load example

Let us finish with an example that will show the loading and saving capabilities of the application. This example is based on B3Kat cataloguing platform¹⁰ which we will use to fetch data about books and their relevant information.

Unlike in the previous examples, here we will be starting from an already curated example project file. Files like these can be created by appointed users to separate bigger data sets into smaller, better manageable chunks. The data schema used in this example has been curated directly via the application from a *ttr* file available at github¹¹.

We will also be utilizing a Solid pod for data persistence. Before proceeding further, it is important we follow the steps outlined in subsection 5.1.1 and have the Solid pod set up with the necessary application permissions up correctly.

Let's open the application Demo - <https://jaresan.github.io/simplod/build/index.html>.

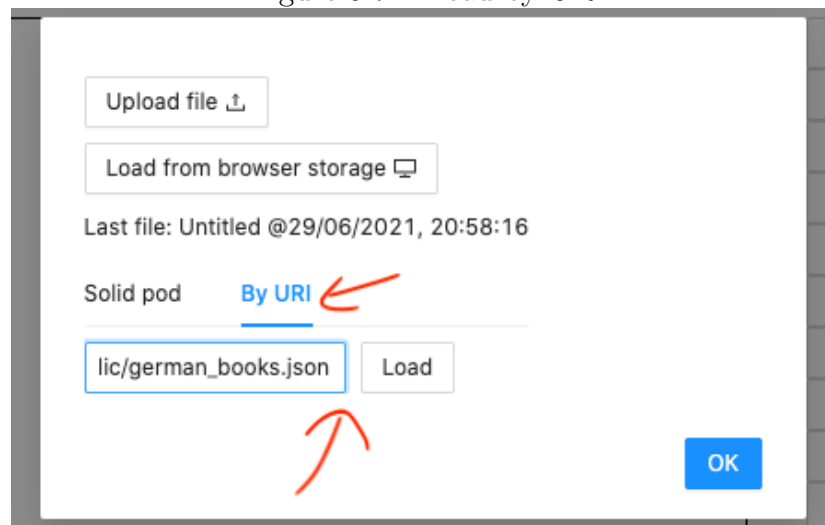
To start with the example, we will first load the appropriate project file by navigating to *File* → *Load* → *By URI*.

We put https://jaresan.github.io/simplod/examples/german_books.json in the input field and press **Load**.

¹⁰<https://www.kobv.de/services/katalog/b3kat/>

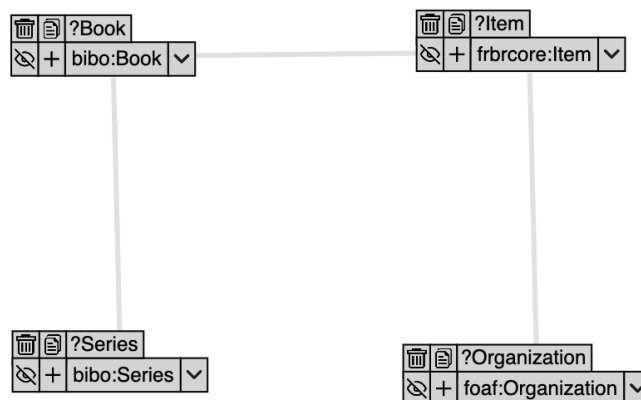
¹¹https://raw.githubusercontent.com/jaresan/simplod/master/public/german_books.ttr

Figure 5.71: Load by URI



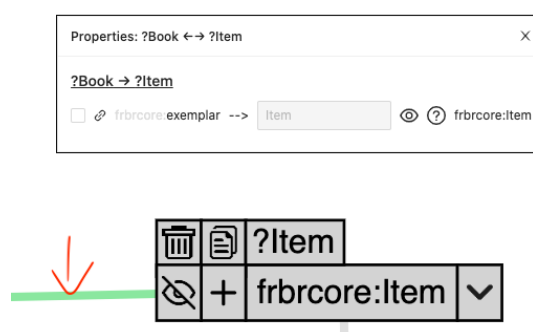
Upon loading the project file, we can see the default view for the curated book data set:

Figure 5.72: Graph loaded



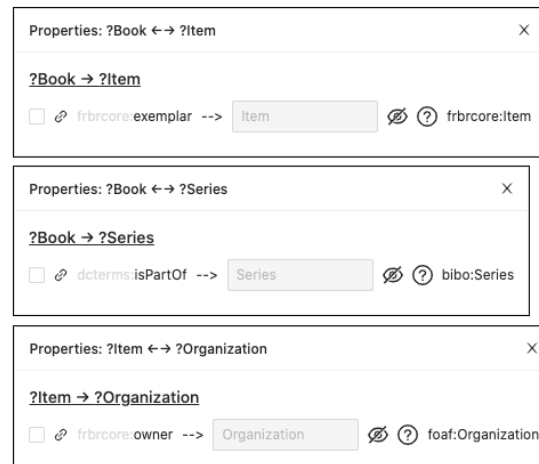
By clicking on the edges between the nodes, we can inspect the relationships they represent:

Figure 5.73: Edge descriptions



By inspecting all three edges presented, we will get the following information for the existing relations in the data schema:

Figure 5.74: Edge descriptions



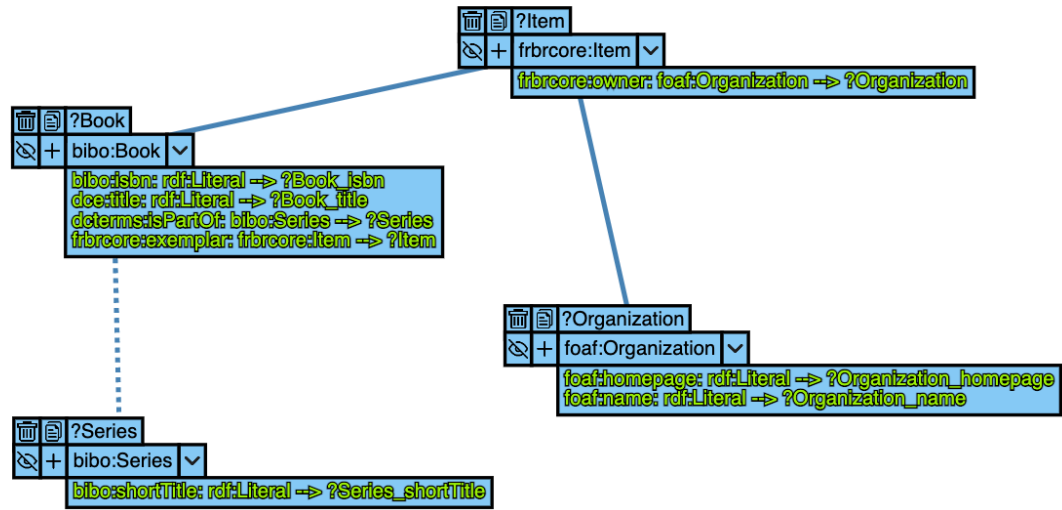
From this, we can gather that a book can have an exemplar item which in turn is owned by an organization. A book can also be a part of a series.

Let's find books with their respective series, should they be a part of one. If we are looking for books, we might also be interested where we could borrow them from. We are then going to query their respective exemplar **Items**, and for the items, we will select the **Organizations** that represent the owner.

As usual, we want to get the labels or titles for each item. Organizations in this case also contain the property **homepage** which could be useful as a reference to the owner as well. With all of this data in mind, the selection would look as follows:

- Book
 - bibo:isbn** - ISBN of the book
 - dce:title** - Title of the book
 - dcterms:partOf**, Optional - Book series
 - frbrcore:exemplar** - Exemplar of the book
- Series
 - bibo:shortTitle** - Title of the series
- Item
 - frbrcore:owner** - Organizational owner of the exemplar
- Organization
 - foaf:homepage** - Homepage of the organization
 - foaf:name** - Name of the organization

Figure 5.75: Graph selection



Notice the dashed edge, this means the relationship between book and its series is optional, meaning we will query for all books and return their respective series, if existing. Not marking this edge as optional, we would only query for books that exist in a series. In the list selection, we can deselect the entities themselves to omit the IRIs from displaying in the result set:

Figure 5.76: List selection

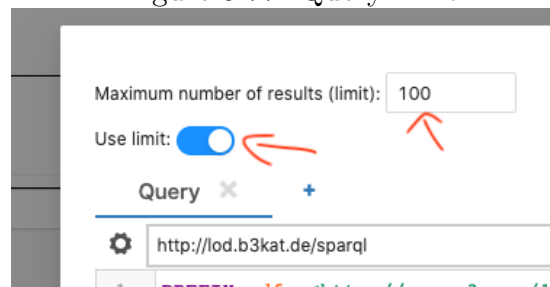
Result column order:

- 1: Book_isbn +
- 2: Book_title +
- 3: Series_shortTitle +
- 4: Organization_name +
- 5: Organization_homepage +

> bibo:Book	? Book	<input type="checkbox"/>			
> bibo:Series	? Series	<input type="checkbox"/>			
> foaf:Organization	? Organization	<input type="checkbox"/>			
> frbrcore:Item	? Item	<input type="checkbox"/>			

We can run the query via the icon. For the demonstration purposes of this example, it would be a good idea to limit the maximum number of results we can retrieve to 100. The data set provided by B3Kat spans over 25 million titles and querying across them all might take a significant amount of time.

Figure 5.77: Query limit




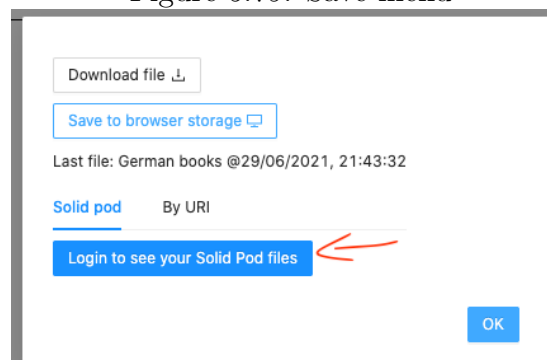
After changing the limit and running the query again by pressing the , we can get the following results:

Figure 5.78: Results

	Book_isbn	Book_title	Series_shortTitle	Organization_name	Organization_homepage
5	0122897455	Plant resources of arid and semiarid lands		Hochschulbibliothek Amberg	< http://bibliothek.oth-aw.de >
44	0231054769	The elements of cinema		Hochschulbibliothek Amberg	< http://bibliothek.oth-aw.de >
45	0231054777	The elements of cinema		Hochschulbibliothek Amberg	< http://bibliothek.oth-aw.de >
62	0387117830	Systemtechnik		Hochschulbibliothek Amberg	< http://bibliothek.oth-aw.de >
3	0387158782	Meß- und Prüftechnik	Halbleiter-Elektron.	Hochschulbibliothek Amberg	< http://bibliothek.oth-aw.de >
78	0444861858	Handbook of econometrics		Hochschulbibliothek Amberg	< http://bibliothek.oth-aw.de >

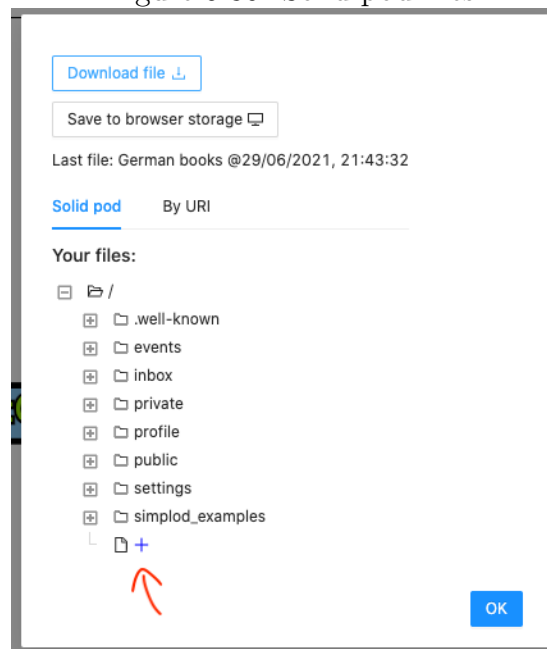
What if we want to save this result to our Solid pod? We just go to *File* → *Save*. If we are logged in, we already see our Solid pod files. If we are not logged in, we can log in either directly through the button in this menu, or through the top right avatar menu.

Figure 5.79: Save menu



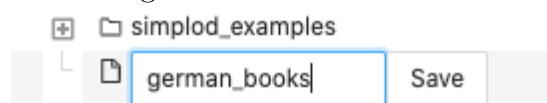
After successfully logging in, we can see our the list of our Solid pod files. We can list through the folders and in each one click the **+** icon to save the file in that location. Let's go with the root folder and click the **+** icon.

Figure 5.80: Solid pod files



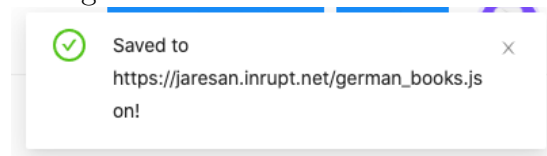
After clicking the **+** icon, we just have to pick a file name. Let's type in *german_books* and click on save or hit enter.

Figure 5.81: New filename



If the file got saved correctly, we are greeted with a notification confirming the new save location:

Figure 5.82: File saved notification



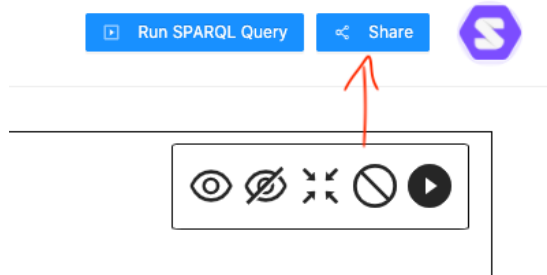
We can also see the new file location in the status bar. Hitting **ctrl+s** now saves the changes to the new remote location.

Figure 5.83: Status bar after remote save



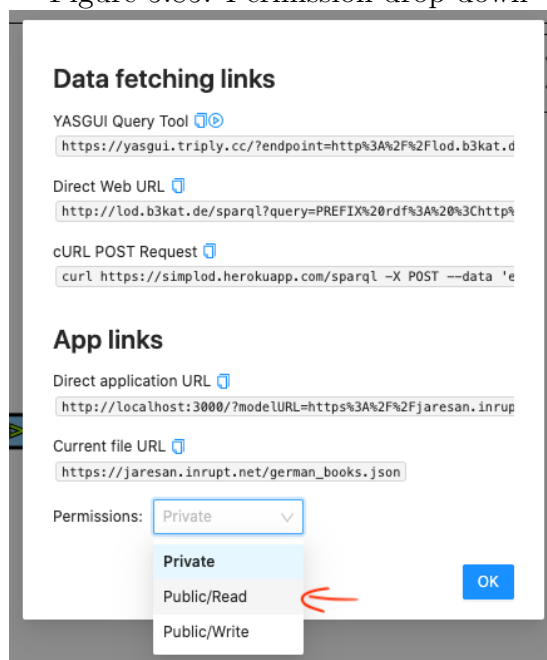
Finally, we might want to share this project file among other users. To do that, first we need to set its permission appropriately from the **Share** menu in the top-right.

Figure 5.84: Permission drop-down



From the drop-down list, we can select **Public/read**, so that every user will be able to read this project file, but only we, as the owner, will be able to edit it.

Figure 5.85: Permission drop-down




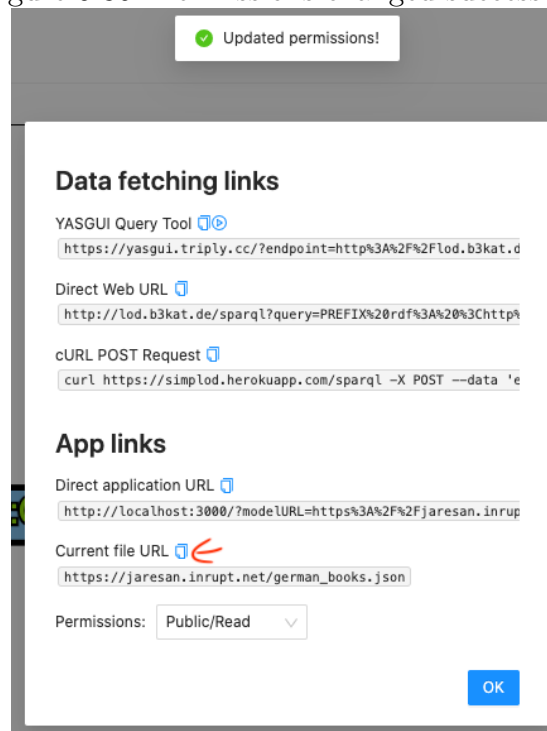
If everything goes correctly, the action is confirmed by a notification and we can proceed to copy the file URL via the  icon and share it among other users.

Figure 5.86: Permissions changed successfully



The copied URL (in this case https://jaresan.inrupt.net/german_books.json) leads directly to the model file in our Solid pod. Same way as in the first step of this example this URL can be directly loaded in the application by *File* → *Load* → *By URI*.

A. Solid pod troubleshooting

This chapter describes manual setup of solid pods in detail, should the steps outlined in [subsection 5.1.1](#) fail.

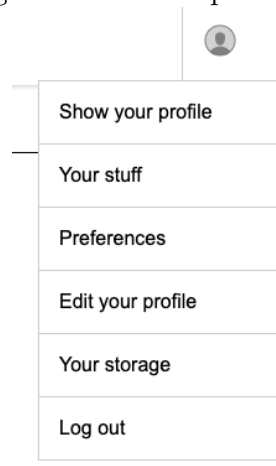
A.1 Logging in with Solid POD

To set up all permissions for the application, the user has to sign in¹ to their Solid pod first.

After signing in, the user can navigate in the application utilizing a tooltip menu by hovering over their avatar in the top right corner.

In the following sections navigating to a certain part of the solid pod takes this menu into account and references it.

Figure A.1: Solid pod menu



A.2 Enabling trusted apps

To allow the application to manipulate data in the solid pod, it has to be added to the solid pod's trusted sources. The application can be marked as trusted by following these steps:

- Go to "Preferences" in the menu
- Go to "Manage your trusted applications" section and add the URL where the application is hosted, as a demo example `https://jaresan.github.io` can be used as the application URL

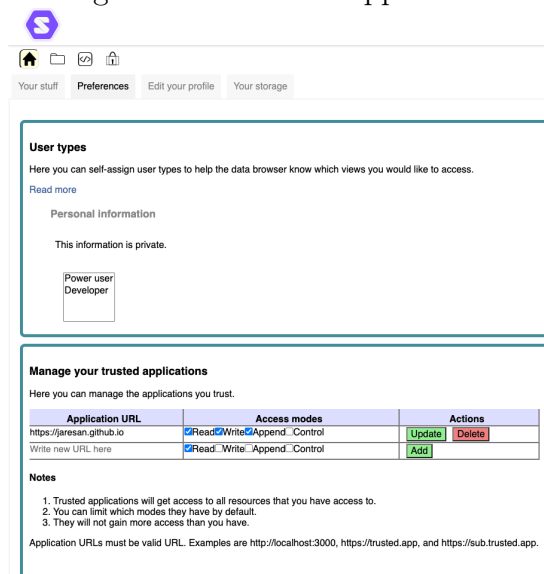
Note: Make sure you omit the trailing slash, i.e. `https://hosting.com`, not `https://hosting.com/`

- Check "Read", "Write", "Append" privileges
- Click on "Add"

¹<https://solid.community/login>

This allows the application to manipulate data in the user's solid pod when they are logged in via their solid pod providing entity.

Figure A.2: Trusted applications



A.3 Creating a save destination

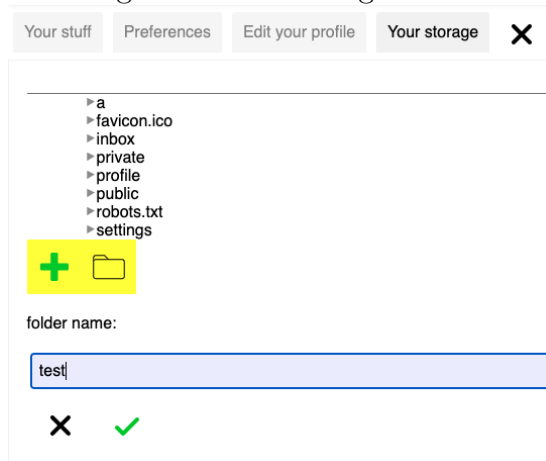
To allow the application to manipulate the views, the user has to provide a destination where the data can be manipulated. This can be done by creating a folder in the user's solid pod as follows:

- Go to "Your storage" in the menu
- Click the green plus button
- Select the folder icon and choose new folder name
- Click the green check button
- The newly created folder can be found at

`https://USERNAME.solid.community/FOLDER_NAME`

where USERNAME and FOLDER_NAME represent the user's solid pod login and the folder name chosen in the previous steps respectively

Figure A.3: Creating a folder



A.4 Linking the save destination

The application uses the solid pod's settings file to determine the location of the folder it can utilize to store and manage its data.

The location of the settings file is specified in the user's profile in their solid pod, e.g. <https://USERNAME.solid.community/profile/card>.

In the following part, it is assumed the user's settings file is hosted at the default provided URL at <https://USERNAME.solid.community/settings/prefs.ttl>.

For demonstration purposes, the folder considered in the following steps is considered to be created in the root of the user's solid pod, therefore being hosted at https://USERNAME.solid.community/FOLDER_NAME. If that is not the case for the reader, they are encouraged to replace these URLs with the URLs of their own resources. To correctly set up the save destination, the user can follow these steps:

- Go to <https://USERNAME.solid.community/settings/prefs.ttl>
- Click the source button
- Click edit
- Add the following line:

```
<https://APPLICATION_DOMAIN>  
a <http://www.w3.org/ns/auth/acl#agent>;  
<http://www.w3.org/ns/pim/space#storage>  
<https://USERNAME.solid.community/public/FOLDER_NAME>
```

- Save the changes