PhUSE CSS

Linked Data and Graph Database

Hands-on Workshop

EXERCISES

Version 2.0

October 2017

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Contents

[Introduction 2](#_Toc485734202)

[Required Materials 2](#_Toc485734203)

[Symbols 2](#_Toc485734204)

[Server Login 3](#_Toc485734205)

[Exercises 4](#_Toc485734206)

[1. Neo4j 4](#_Toc485734207)

[1.1 Diagram the Data Model 4](#_Toc485734208)

[1.2 Transfer Diagram to Spreadsheet 6](#_Toc485734209)

[1.3 Upload to Neo4j 7](#_Toc485734210)

[1.4 Query and Visualize 9](#_Toc485734211)

[1.4.1 Find Study Participants 9](#_Toc485734212)

[1.4.2 Treatment for Specified Patient: Graph 10](#_Toc485734213)

[1.4.3 Treatment for Specified Patient: Value 10](#_Toc485734214)

[1.4.4 OPTIONAL 10](#_Toc485734215)

[2. Resource Description Framework (RDF) 11](#_Toc485734216)

[2.1 Draw the Data Model 11](#_Toc485734217)

[2.2 Transfer Model to Spreadsheet 11](#_Toc485734218)

[2.3 Create RDF (TTL) File 12](#_Toc485734219)

[2.4 Query and Visualize 13](#_Toc485734220)

[2.5 Additional Queries and Visualization 15](#_Toc485734221)

[Demonstrations 17](#_Toc485734222)

[8. Course Resources 18](#_Toc485734223)

# Introduction

Instructions in this document are specific to the PhUSE cloud server environment, files, and methods used for the PhUSE workshop. The exercises represent one of many possible approaches to the material and make no claim to be best or recommended method. The instructor will complete the exercises during the session, along with the class.

The exercises are presented in two sections followed by two demonstrations:

**Section 1:** Labeled Property Graph Neo4j

**Section 2:** Resource Description Framework (RDF )

**Demonstrations:** SDTM data in Neo4j and RDF [Time Permittting]

Your feedback is welcomed and encouraged. Please send your comments to: **tim.williams@PhUSE.eu**

## Required Materials

* Pencil + eraser or pen
* Laptop with Remote Desktop capability
* Printed copies of the following will be supplied by the instructor:
  + WorkshopExercises.pdf (this document)
  + Neo4jDiagram.pdf
  + RDFDiagram.pdf

## Symbols

The following symbols are used in this document.

|  |  |
| --- | --- |
| **!** | Cautions and warnings. Failure to follow these steps may lead to unanticipated results and problems. |
| IdeaIcon_clean_20mm | Helpful tips and advice, including guidance that may only apply to these exercises (and not generally applicable to Linked Data in other contexts). |
| Image result for information icon | Additional information. |

# Server Login

Login to the cloud server using Remote Desktop. Instructions assume Windows OS on your local machine.

1. In the search box on the taskbar on your laptop, type **remote desk,** then select **Remote Desktop Connection** from the Programs section.﻿



1. Click **Show Options** if needed to show the fields for **Computer:** and **User name:**

You will be provided with an IP Address for the Computer: field. Everyone will use the same User name:

**Computer:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Provided by Instructor)

**User name:** phuseldw

**Password:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Provided by Instructor)

1. Click **Connect** after entering the Computer IP address and your username for the session.
2. Enter the password supplied by the instructor and click **OK**.

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

# Exercises

# Neo4j

# Diagram the Data Model

A basic model containing a small number of entities in a clinical trial is provided as starting point. You will add additional nodes and relations to the diagram before translating it into the Neo4j Labeled Property Graph.

1. Examine your copy of the Neo4j Diagram printoutand observe the following:
   1. **Nodes** represent a Person, a Study, and a Treatment within a study.
   2. **Property:value pairs** (p:v pairs) on each node contain additional information, such as the person's age and the title of the study.
   3. **Relationships between the nodes**, often called *links* or *edges*, are shown with arrows that contain labels describing the type of relationship. Neo4j allows p:v pairs on these links.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | To keep our exercises simple, ***you will not attach p:v pairs to the links in these exercises***. |

1. Add nodes to the diagram.
   1. Use a pen (or preferably a pencil) to add new nodes and relations to the diagram.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | Pencil & eraser makes it easier to adjust the nodes and relations as you develop your data model. If you wish to start over, ask the instructor for a fresh copy of the diagram. |

1. Add a few p:v pairs to your new nodes.
2. You may also add additional property:value pairs to the nodes that were already present in the diagram (Example: Add Gender to the Person1 node).

|  |  |
| --- | --- |
| **!** | ***Add no more than six additional nodes to the model and fewer than twelve property:value pairs. This will keep the model manageable for later exercises.*** |

**Guidelines for adding nodes and relations**

Follow these guidelines when creating the nodes, p:v pairs, and relations.

|  |  |
| --- | --- |
| Image result for information icon | Many of these recommendations serve to simplify the exercises and are not applicable to Linked Data in real-world applications. |

**Nodes**

|  |  |
| --- | --- |
| **Guideline** | **Examples/Explanation** |
| Short node names | Person2 not "Person 2 in the Clinical Study" |
| No spaces or special characters ($"%&^!~…etc.) in the names | Person2, not "Person 2" |
| Follow naming conventions already used in the diagram | Person2, not Person\_2 |

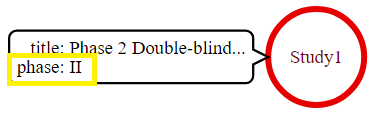
**Node P:V pairs**

|  |  |
| --- | --- |
| **Guideline** | **Examples/Explanation** |
| Follow naming conventions consistent with those in the initial diagram. | age, not AgeYRS |
| Integers, characters, strings with spaces are all acceptable. | title:Phase 2 Double blind… |
| Avoid special characters ($"%&^!~, etc.) | Special characters may have unanticipated effects in the conversion scripts. |
| Add any new p:v pairs you wish (gender, height, etc.) to existing and new nodes. |  |

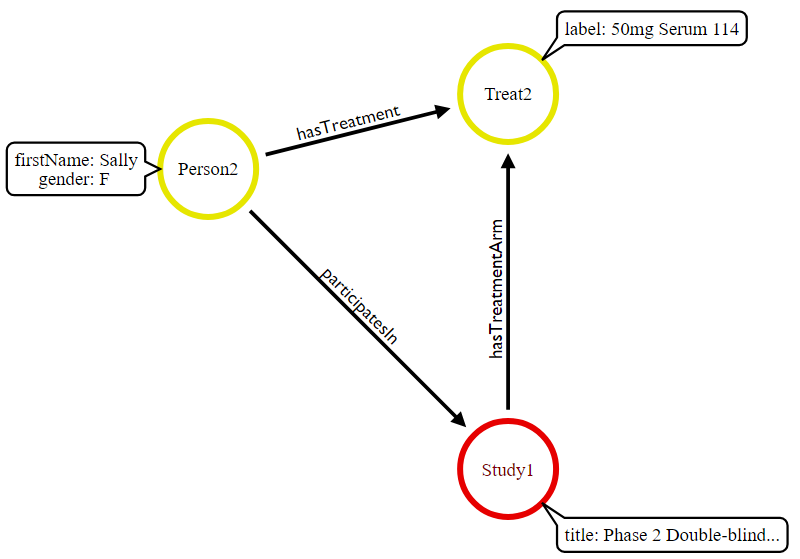
**Relations (Links/Edges)**

|  |  |
| --- | --- |
| **Guideline** | **Examples/Explanation** |
| No spaces or special characterse ($"%&^!~…etc.) in the names | Special characters may have unanticipated effects in the conversion scripts. |
| Relationships have ***direction*** and ***describe*** how the nodes are related. Pick names that define the relation in a concise way. |  |
| Remember: no p:v pairs on the links. | Keeping it simple for these exercises. |

**Examples**

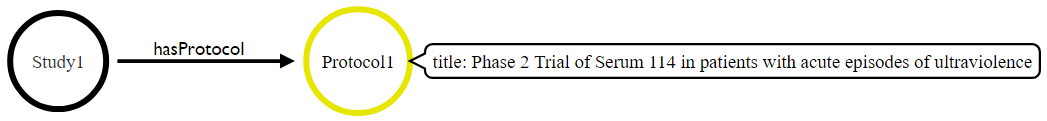
Consider the following examples if you are having trouble imagining new nodes and relations.

Example 1: Add study phase information to the existing Study1 node.



Example 2: Add a new female person node (Person2) and assign them to a new treatment node (Treat2). Remember to add relations from Person2 to Study1, and Study1 to Treat2.

Example 3: Add a protocol to the study. How would the protocol fit into a description of the clinical trials design process?



|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | Consider other aspects of clinical trial design and add those entities and relationships to your diagram, keeping in mind our 6-node, 12 p:v suggested limit. |

# Transfer Diagram to Spreadsheet

You will now create a machine readable version of your diagram by copying the nodes, relations, and p:v pairs into a spreadsheet.

1. Open Windows Explorer to the Linked Data folder location using the desktop shortcut 
2. Navigate to the **…/data** subfolder and double click on the file **Neo4jModel.xlsx** to open the spreadsheet.
3. Observe how the spreadsheet is divided into two tables as show in **Figure 1 .**

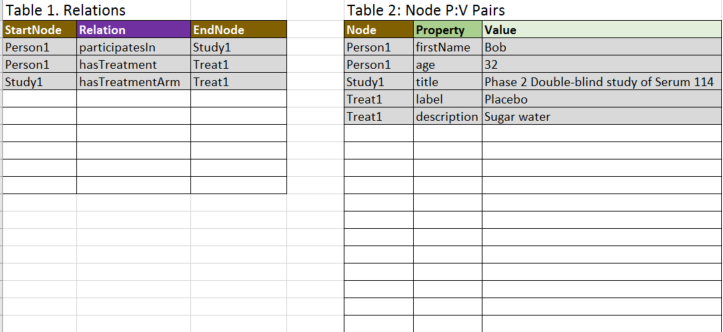


Figure 1 Neo4jModel.xlsx

**Table 1. Relations**

Table 1 contains the Node-to-Node relationships. Each row stores the node at the start of a relationship (**StartNode**), the label for the relation (**Relation**), and the node at the end of the relationship (**EndNode**). Nodes are listed once for each relationship in which they participate, so a single node is often listed more than once.

Examples

* Person 1 *participatesIn* Study1 and *hasTreatment* Treat1, so Person1 is listed twice.
* A node at the end of one relation (EndNode) can also be the StartNode in another relation. Observer how Study1 is both a StartNode and EndNode.

1. Enter your new StartNodes, Relations, and EndNodes that you added to your diagram, entering them under the rows shaded in grey that represent items in the initial diagram. Ensure you capture all the relations, especially those where nodes participate in more than one relationship.

**Table 2: Node P:V Pairs**

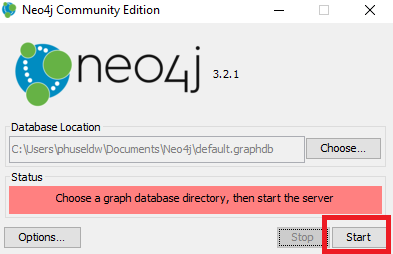
Table 2 lists the property:value pairs attached to each node. Each row lists one p:v pair, so nodes like Person1 are listed on two rows: one for the **firstName** property, then again for the **age** property.

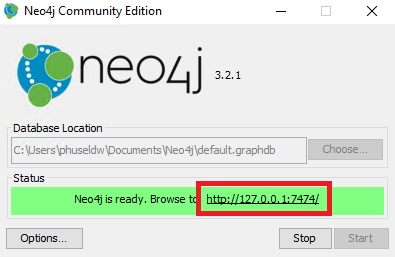
1. Enter your new nodes Property:Value pairs into Table 2. List each new p:v pair on a new row in the table, along with the Node that contains that property. Nodes that have more than one property:value pair will be listed more than once.
2. Cross-check the tables:
   1. **Table 1. Relations**

* Confirm the values for **StartNode** and **EndNode** are named consistently, without error, and match the names in the diagram exactly.
  1. **Table 2. Node P:V Pairs**
* Confirm the values in the **Node** column are named consistently, without error, and match the names in the diagram exactly.
* All StartNode and EndNode names in Table 1 ***must*** be present in the Table 2.

|  |  |
| --- | --- |
| **!** | **If a node name is present in Table 1 and absent in Table 2, the conversion process in the next section will fail.** |
| Image result for information icon | It is acceptable to have node names in the Table 2 without them appearing in the Table 1. These nodes will be present as isolated nodes with no relation to other nodes, which is completely acceptable (although a bit odd for our exercises!) |

# Upload to Neo4j

1. Sign in to the cloud server desktop.
2. Start Neo4j by double-clicking on the application shortcut on the desktop. 
3. Accept the default **Database Location** shown in the dialog box and click **Start**.

1. There will be a delay while the database initiates. The red bar changes to yellow and finally to green when the database is ready. The green bar contains the address of the Neo4j instance.
2. Click on the http:: address in the **Browse to** fieldto launch Neo4j in a web browser.

|  |  |
| --- | --- |
| Image result for information icon | The database is now running and contains no data. You will insert your data into the database from the spreadsheet, using R. |

1. Minimize the browser window to return to the desktop
2. Use Windows Explorer to navigate to the folder **C:\LinkedDataWorkshop\scripts\r** , then double click on the file **Neo4jFromExcel.R** to open it into RStudio.
3. Execute the R script by clicking on the Source toolbar button 
4. Review the R Console window.
5. If the script ran without error, you will see the message:

Success! Neo4j data available at http://localhost:7474/browser/

1. If an error occurred, use Table 1 to help diagnose the problem and contact the instructor or assistant for help.

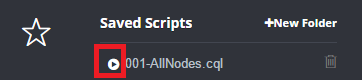
Table 1 Error Messages and Resolutions

| **R Console Error Message** | **Cause and Resolution** |
| --- | --- |
| Error in curl::curl\_fetch\_memory(url, handle = handle) :  Couldn't connect to server | The Neo4j server is not started or is not available.  Return to steps 1-3 in Section 1.3 Upload to Neo4j to start Neo4j and confirm it is running, then execute the R Script again using the Source toolbar button. |
| ERROR: Spaces in node names not permitted in this exercise!  ERROR: Fix node names, then re-run script.  Error in eval(expr, envir, enclos) : | Review the values in the **StartNode**, **EndNode**, and **Node** columns for spaces in the names. Spaces are not permitted for these exercises.  Correct the node names (including on the diagram so it matches the spreadsheet), then re-run the R Script. |
| ERROR: Node found in relation is not a defined node. | A node defined in Table 1. Relations as a StartNode or EndNode is not defined in Table 2. The node name will be listed in the console message.  Possible resolutions include:   * Add the node name to Table 2. * Correct the node name in Table 1 to match the name in Table 2. |
| R script fails to execute after corrections made to spreadsheet. | Ensure spreadsheet was saved. Re-run script.  Ask for assistance. |
| WARNING: Node not used in any relation: | A node listed in Table 2 is absent from Table 1. This situation results in a node that is not connected to other nodes. It may be an unintended omission.  Action may or may not be required. Review your diagram against the spreadsheet and ask for assistance if need. |

# Query and Visualize

1. Return to the Neo4j browser window you opened in a previous step. If you closed the browser you can re-open it by clicking the link in the Neo4j window.
2. Click on the star icon in the application side bar to view the **Saved Scripts** prepared for these exercises.

|  |  |
| --- | --- |
| **!** | Ask for assistance if you do not see the **Saved Scripts**. |

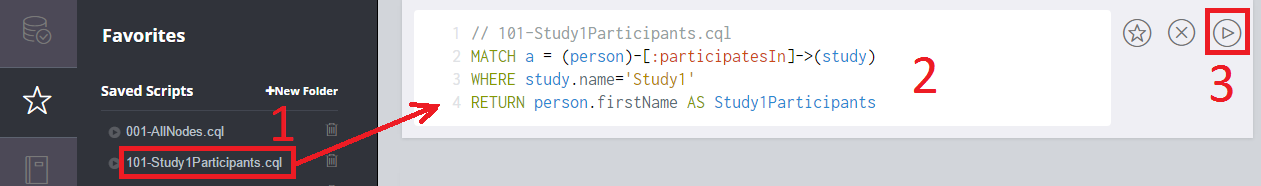


1. Click the play icon beside **001-AllNodes.cql** to show all nodes and relations in the database.
2. Explore the graph and compare it with your **Neo4j Diagram**.
3. Use drag-and-drop to position the nodes with your mouse.
4. Click on a node and view the node property:value pairs at the bottom of the Neo4j screen.

# Find Study Participants

Query the graph to find the first names of patients in Study1.

1. Click on the **name** of the Saved Script (1). This places the cypher code in the execution window (2).



1. Review the script to understand how it queries the path of graph data.

101-Study1Participants.cql

|  |
| --- |
| **MATCH a = (person)-[:participatesIn]->(study)**  **WHERE study.name='Study1'**  **RETURN person.firstName AS Study1Participants** |

1. Click on the execution button to the right of the query text (3) to execute the query.
2. Observe the result of query in the results window.

# Treatment for Specified Patient: Graph

Determine the treatment Bob received in the study by returning the result as a graph.

1. Load and execute the query **102-BobTreatmentGraph.cql**.

102-BobTreatmentGraph.cql

|  |
| --- |
| **MATCH a = (person)-[:hasTreatment]->(treat)**  **WHERE person.firstName='Bob'**  **RETURN a** |

# Treatment for Specified Patient: Value

Determine the treatment Bob received in the study by returning the result as a value.

1. Load and execute the query **103-BobTreatmentValue.cql** to retrieve the same result as value, not as a graph.

103-BobTreatmentValue.cql

|  |
| --- |
| **MATCH a = (person)-[:hasTreatment]->(treat)**  **WHERE person.firstName='Bob'**  **RETURN person.firstName AS Name, treat.label as Treatment, treat.description AS Description** |

# Optional Activity

While you wait for the class to catch up:

1. Try writing and executing your own queries on the data you created.
2. You can load one of the saved scripts as a starting point for your query.
3. Use the **001-AllNodes.cql** script to return the entire graph to remind you of the nodes, relations, and properties that can be queried.

|  |  |
| --- | --- |
| Image result for information icon | Neo4j is capable of queries that are much more complicated than shown here. See the course Resources to learn more about Neo4j and its query language Cypher. |

This is the end of exercises for Neo4j. RDF will be introduced before returning to the exercises in the next section.

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

# Resource Description Framework (RDF)

The same concepts you modeled using Neo4j will now be represented using RDF.

# Draw the Data Model

An initial model is once again provided as a starting point, to which you will add the same values you created in the Neo4j exercises.

1. Compare the base model for RDF in Figure 3 with the base model for Neo4j (Figure 1)

RDF does not use property:value pairs on nodes and edges. Observe how *Subjects – Predicate --> Object* relations attach values like the age (32) to the Person1 node. Property:Value pairs in Neo4j must become S-P-O relations in your RDF graph model.

1. Transfer the entities from the Neo4j Figure 1 model to the Figure 3 RDF model by drawing them into the diagram. It may help to remove the staple from this document to allow you to place the diagrams side-by-side.
2. In the diagram, strings are indicated in green and integers in blue. These types of nodes are not capable of linking to other nodes - the path ends at those nodes. Nodes that link to other nodes, or have a capability of doing so, are shown indicated in red. The distinction will become important when transferring the diagram to the spreadsheet.

# Transfer Model to Spreadsheet

You must now convert the model to a machine-readable representation so it can be converted to RDF triples. This is accomplished by transforming your diagram into a spreadsheet, similar to the Neo4j exercise.

1. Use Windows Explorer to navigate to the **…/data** subfolder and double click on the file **RDFModel.xlsx** to open the spreadsheet**.**
2. The first rows of the spreadsheet contain the data from the base model, before you added your own data. You will add your new data below the rows that are shaded in grey ( Figure 4 ) .
3. You must determine the ObjectType for the Object in each Subject --Predicate --> Object relations. Many different data types are available in RDF. The exercises use only ***string***, ***int***, and ***uri*** types for simplicity.

Recall how it is possible for the Object in one relation to be the Subject in another relation, forming a series of node-to-node relationship paths. In this exercise, these "connection" nodes are coded as the 'uri' ObjectType because they can form a path of connected links.

|  |  |
| --- | --- |
| **string** | Character values with no outbound link.  Examples: "Bob", "Protocol for Study 123", "Male" |
| **int** | Integer values with no outbound link. Example: 32 |
| **uri** | Object nodes that link to other nodes in your existing model, or represent things that ***could*** participate in another link relationship. |

Ask the instructor for assistance if you are unsure which ObjectType should be assigned to a node in your model.

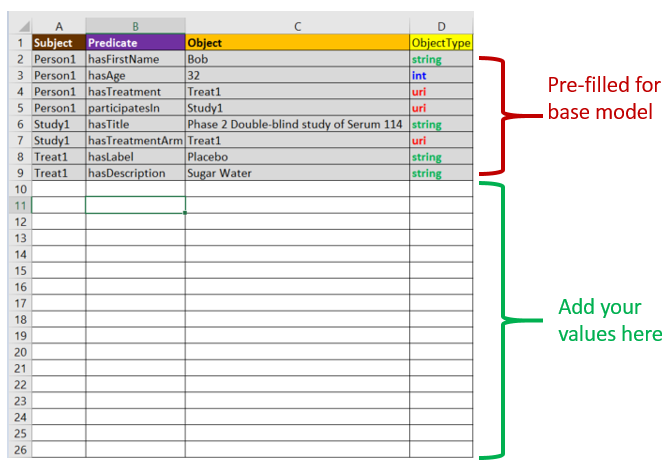


Figure 2 RDF Spreadsheet

1. Confirm all nodes and relations are present in your spreadsheet. You may have several nodes listed as both Subject and Object.
2. Save the file using the **File | Save** menu.

# Create RDF (TTL) File

An R script is used to convert the spreadsheet data into RDF and save the result in a file with a .TTL extension (N3 Turtle serialization).

1. Use Windows Explorer to navigate to the folder **C:\LinkedDataWorkshop\scripts** , then double click on the file **SpreadsheetToRDF.R** to open it into RStudio.
2. Execute the R script by clicking on the Source toolbar button 

|  |
| --- |
| The R script also validates the TTL file. You receive no message if the file is valid RDF. Error messages follow the format that identifies the location and type of error:  ERROR [line: *n*, col: *n*] *type of error*  Contact the instructor or assistant if you encounter an error message. Open the **RDFModel.TTL** file in the /data folder into Notepad++ for troubleshooting. |

1. The script outputs RDF as the file: **C:\LinkedDataWorkshop\data\RDFModel.TTL**
2. Double click the TTL file to open it into Notepad++. The file will appear similar to **Error! Reference source not found.**. Observe how the model in
3. Figure 3 was translated to the data in Figure 4 and lastly into the RDF representation **Error! Reference source not found.** .

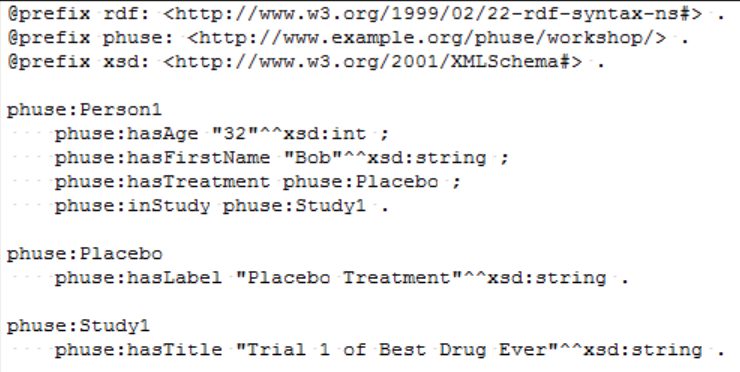
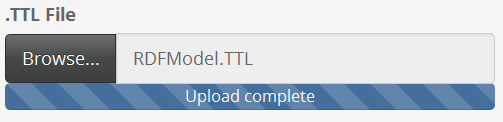


Figure 3 TTL file RDFModel.TTL

# Query and Visualize

An R Shiny app is used to query and visualize the RDF TTL file.

1. From RStudio, open the file C:\LinkedDataWorkshop\scripts\r\**SelectTTLToQuery.R**
2. Run the app by clicking the RunApp icon  or using the key combination ***Ctrl+Alt+R*** .
3. Load your TTL file into the application by clicking **Browse** under**.TTL File** and navigate to the file C:\LinkedDataWorkshop\data\**RDFModel.TTL** . Double-click the file to load it into the app.
4. A default query is already available within the app. Click **Run query** to execute the query and view the result in the **Query Result:** area.



1. Review the Subject, Predicate, Object values in the Query Result and compare them to your model.
2. Click on the **Visualize** tab at the top of the app to view a network graph of the query result. The graph will look similar to Figure 6

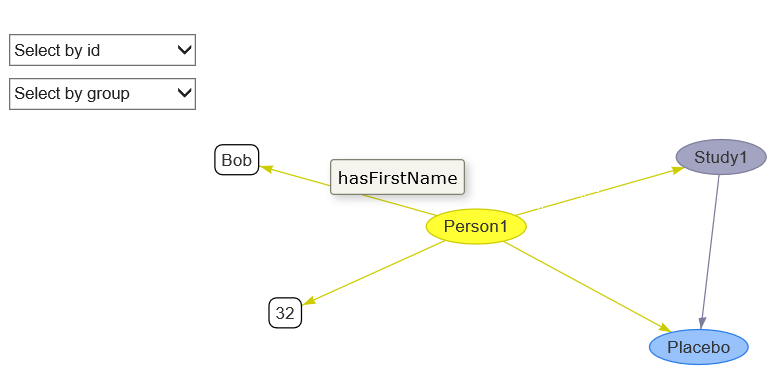


Figure 4 Force network graph of base model

1. Explore the graph by clicking on nodes and using mouse-over to show the relations. You may also use the drop-down selections for highlighting node categories and groups.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | You can return to the full graph visualization at any time by loading and executing the script /scripts/SPARQL/**200-EntireGraph.rq** |

# Additional Queries and Visualization

**Find Study Participants**

1. Click on the **Query** tab in the app.
2. Click **Browse** under **OPTIONAL:.RQ Query File** and navigate to the file C:\LinkedDataWorkshop\scripts\SPARQL\**201-Study1Participants.rq** . Double-click the file to load it into the app.
3. Click Browse under the Load the query **201-Study1Participants.rq** **.** Query the graph to find the first names of patients in Study1.

201-Study1Participants.rq

|  |  |
| --- | --- |
| # 201-Study1Participants.rq  PREFIX phuse: <http://www.example.org/phuse/workshop/>  SELECT ?name  WHERE {  ?person phuse:participatesIn phuse:Study1 .  ?person phuse:hasFirstName ?name .  } | |
| **!** | | The **Visualize** tab in the R Shiny app requires results to be returned as ?s, ?p, ?o to illustrated Subject-Predicate-Object relations. It does not accommodate single nodes or SPARQL results with names other than s,p, o. When a query result does not meet these conditions, the Visualize tab displays:  **Error: incorrect number of dimensions** | |

**Treatment for a Specified Patient: Graph**

Determine the treatment Bob received in the study.

1. Use the field **OPTIONAL:.RQ Query File** to load the query file **202-BobTreatmentGraph.rq** into the query window.
2. Execute the query and view the result in the Visualize tab. The query contains a number of BIND statements to provide the ?s, ?p, ?o result format needed to display a graph result in the app.

202-BobTreatmentGraph.rq

|  |
| --- |
| # 202-BobTreatmentGraph.rq  # Two queries to get different parts of the graph returned as s,p,o  # as needed for the visualization.  PREFIX phuse: <http://www.example.org/phuse/workshop/>  SELECT ?s ?p ?o  WHERE {  {  # Get the first set of triples for the graph:  # Person - hasFristName - Bob  ?person phuse:hasFirstName "Bob" .  BIND ( ?person as ?s)  BIND ( "phuse:hasFirstName" AS ?p)  BIND ( "Bob" AS ?o)  }  UNION  {  # Get Bob's treatment.  ?person phuse:hasFirstName "Bob" .  ?person phuse:hasTreatment ?treat .  BIND (?person as ?s)  BIND ( "phuse:hasTreatement" AS ?p)  BIND ( ?treat AS ?o)  }  } |

**Treatment for a Specified Patient : Value**

1. Use the field **OPTIONAL:.RQ Query File** to load the query file **203-BobTreatmentValue.rq** into the query window.
2. Execute the query to view the result. There is no visualization for this result, only the query result on the Query tab.

204-BobTreatmentValue.rq

|  |
| --- |
| # 203-BobTreatmentValue.rq  # Return Bob's treatment. No visualization  PREFIX phuse: <http://www.example.org/phuse/workshop/>  SELECT ?treatment  WHERE {  ?person phuse:hasFirstName "Bob" .  ?person phuse:hasTreatment ?treat .  # Make the result pretty  BIND (strafter(str(?treat), "workshop/") AS ?treatment)  } |

**OPTIONAL**

While you wait for the class to catch up:

* Try writing and executing your own queries on the data you created. You may use the saved queries as a starting point or write your own from scratch.
* Use the script /scripts/SPARQL/**200-EntireGraph.rq** to view your entire graph as a guide to writing new queries.
* Remember that the graph does not show the phuse: prefix for predicates (edges).

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

---- END OF EXERCISES ----

# Demonstrations

If time allows, SDTM data domains will be shown as both Labeled Property graph and RDF. Files and slides will be provided.

# Course Resources

<Add links for Neo4j and RDF>

**Neo4j**

* [Recommended Overview](https://neo4j.com/developer/graph-database)

https://neo4j.com/developer/graph-database

* [Overview of graph db and Neo4j](https://youtu.be/U8ZGVx1NmQg)

https://youtu.be/U8ZGVx1NmQg

* [Intro to Cypher](https://www.youtube.com/watch?v=1TSBXZMv6tc)

https://www.youtube.com/watch?v=1TSBXZMv6tc

* [Graph Modeling](https://www.youtube.com/watch?v=AaJS-DGBQX4)

https://www.youtube.com/watch?v=AaJS-DGBQX4

* [RDF in Neo4j](http://guides.neo4j.com/rdf-graphs)

http://guides.neo4j.com/rdf-graphs

**RDF**

* [Introduction to Semantic Web](http://www.cambridgesemantics.com/semantic-university/introduction-semantic-web)

http://www.cambridgesemantics.com/semantic-university/introduction-semantic-web

* [What is Linked Data?](http://www.cambridgesemantics.com/semantic-university/what-linked-data)

http://www.cambridgesemantics.com/semantic-university/what-linked-data

* [Introduction to Linked data](http://www.cambridgesemantics.com/semantic-university/introduction-linked-data)

http://www.cambridgesemantics.com/semantic-university/introduction-linked-data

* [Learning SPARQL](http://www.learningsparql.com/)

<http://www.learningsparql.com/>

[Hands-on Workshop Github Repository](https://github.com/phuse-org/LinkedDataWorkshop/tree/master/Annual2017-EU)

All files and content associated with this course.

https://github.com/phuse-org/LinkedDataWorkshop/tree/master/Annual2017-EU