

PhUSE CSS
Linked Data and Graph Database
Hands-on Workshop
EXERCISES

Version 2.0
October 2017

Contents

Introduction	2
Required Materials.....	2
Symbols	2
Server Login.....	3
Exercises	4
1. Neo4j.....	4
1.1 Diagram the Data Model.....	4
1.2 Transfer Diagram to Spreadsheet	6
1.3 Upload to Neo4j	8
1.4 Query and Visualize.....	9
1.4.1 Who enrolled in Study 1?.....	9
1.4.2 What treatment did Bob receive? (Graph result)	10
1.4.3 What treatment did Bob receive? (Value result).....	10
1.4.4 Graph Exploration	10
2. Resource Description Framework (RDF)	11
2.1 RDF Model Spreadsheet to the RDF Diagram	11
2.2 Assign ObjectType.....	12
2.3 Create RDF (TTL) File	13
2.4 Query and Visualize.....	14
2.4.1 Who enrolled in Study 1?.....	15
2.4.2 What treatment did Bob receive? (Triples + Graph result)	16
2.4.3 What treatment did Bob receive? (Value result).....	16
2.4.4 Graph Exploration	16
Demonstrations	17
Appendix 1: Error Messages and Resolutions.....	18
A1.1 Neo4jFromSpreadsheet.R : Error Messages and Resolutions	18
A1.2. RDFFromSpreadsheet.R : Error Messages and Resolutions	18
A1.3 Unprotect .XLSX Spreadsheet in Libre Office.....	19
Appendix 2: Course Resources.....	19

Introduction

Instructions in this document are specific to the PhUSE cloud server environment and workshop methods. 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 attendees.

The workshop is comprised of the following:

Exercise 1: Labeled Property Graph Neo4j

Exercise 2: Resource Description Framework (RDF)

Demonstrations: SDTM data in Neo4j and RDF (time permitting)

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

Required Materials

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

Symbols

Symbols provide important information:



Stop and wait. Do not proceed to the next section until instructed.



Cautions and warnings. Failure to follow these steps may lead to unanticipated results and problems.



Helpful tips and advice.

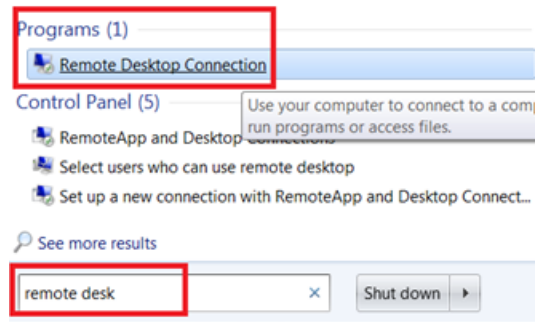


Additional information.

Server Login

Login to the cloud server using Remote Desktop. Instructions assume Windows 10 operating system.

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



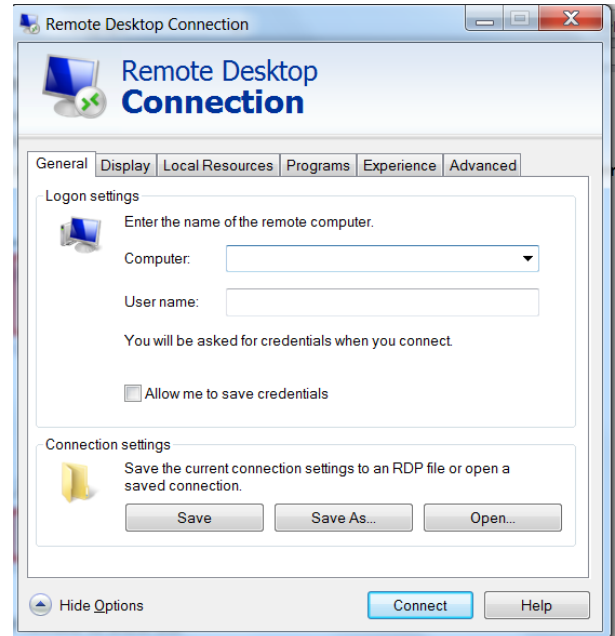
2. Click **Show Options** if needed to show the fields for **Computer:** and **User name:**

The instructor will provide the password and an IP Address for the Computer: field.


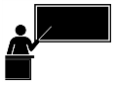

Computer: _____
(IP Address provided by Instructor)

User name: phusel dw

Password: _____
(Provided by Instructor)



3. Entering values in the **Computer:** and **User name:** fields, then click **Connect**.
4. When prompted, enter the password supplied by the instructor and click **OK**.

	<p>Stop here and wait for the instructor.</p> <p> Presentation follows</p>	
---	---	---

Exercises

1. Neo4j

1.1 Diagram the Data Model

An initial 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. Observe the following in your copy of the **Neo4j Diagram**:
 - a. **Nodes** represent a person, a study, and a treatment within a study. Node names are in *UPPERCASE*.
 - b. **Property:value pairs** (p:v pairs) on each node contain information such as the person's age and the title of the study. Property names should be all *lowercase*. Values can be *mixed case*.
 - c. **Relationships between the nodes**, often called *links* or *edges*, have directional arrows and labels that indicate the meaning of the relationship. Neo4j enables p:v pairs on links. However:



To keep our exercises simple, **do not attach p:v pairs to links**.

2. Add nodes to your diagram, following advice from the instructor and the section "[Guidelines for adding nodes and relations](#)" (below).
 - a. Use a pen (or preferably a pencil) to add new nodes and relations to the diagram.



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

- b. Add a few p:v pairs to your new nodes.



Each node must have at least one property:value pair. This is an exercise requirement to aid later comparison with RDF.

- c. You may also add new property:value pairs to the nodes already present in the initial mode. For example, adding **gender** to the **PERSON1** node.



To keep the model manageable for later exercises, please add no more than six additional nodes to the model and less than twelve new property:value pairs.

Guidelines for adding nodes and relations

Follow these guidelines when creating the nodes, p:v pairs, and relations. Mandatory requirements are shown in bold.



Many of these recommendations serve to simplify the exercises and do not apply outside of the workshop.

Nodes

Guideline	Examples/Explanation
Use UPPERCASE letters	PERSON1, not Person1 or person1. Exercise requirement only.
No spaces or special characters (\$"%&^!~...etc.) in the names	PERSON2, not "Person 2"
Short node names	PERSON2 not "Person 2 in the Clinical Study"
Follow naming conventions already used in the diagram	PERSON2, not Person_2, not Patient2

Node property:value pairs

Guideline	Examples/Explanation
Each node must have at least one p:v pair.	Exercise requirement.
property: all lowercase value: Can be mixed case.	Exercise requirement.
Follow naming conventions consistent with those in the initial diagram.	age, not ageinyears or hasage
Value: Integers, characters, strings with spaces are all acceptable.	title:Phase 2 Double blind...
No special characters (\$"%&^!~, etc.)	Special characters may have unanticipated effects in the conversion scripts.
You may add new p:v pairs to existing nodes.	Example: Add gender, height to PERSON1.

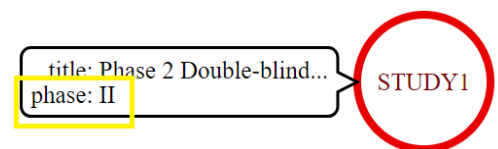
Relations (Links/Edges)

Guideline	Examples/Explanation
Use lowercase letters	treatmentarm, not TREATMENTARM or treatmentArm. Helps to differentiate from Nodes.
No spaces or special characters (\$"%&^!~...etc.) in the names	Special characters may have unanticipated effects in the conversion scripts.
Relationships have <i>direction</i> and <i>describe</i> how the nodes are related. Pick names that define the relation concisely.	
No p:v pairs on the links.	Keeping it simple for these exercises. Conversion scripts in the exercises to do not process p:v pairs on relations so this data will be lost.

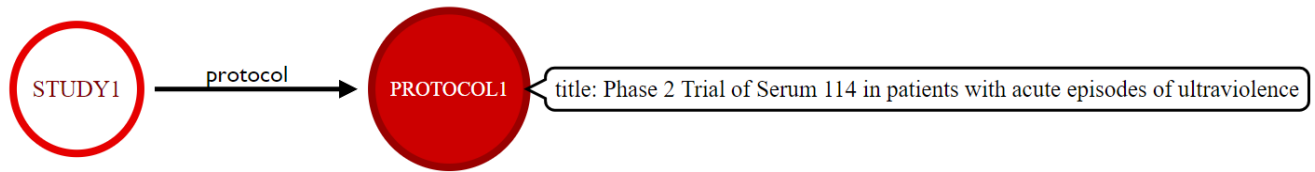
Examples

Consider the following examples if you are having trouble inventing new nodes and relations. New nodes are in solid colors.

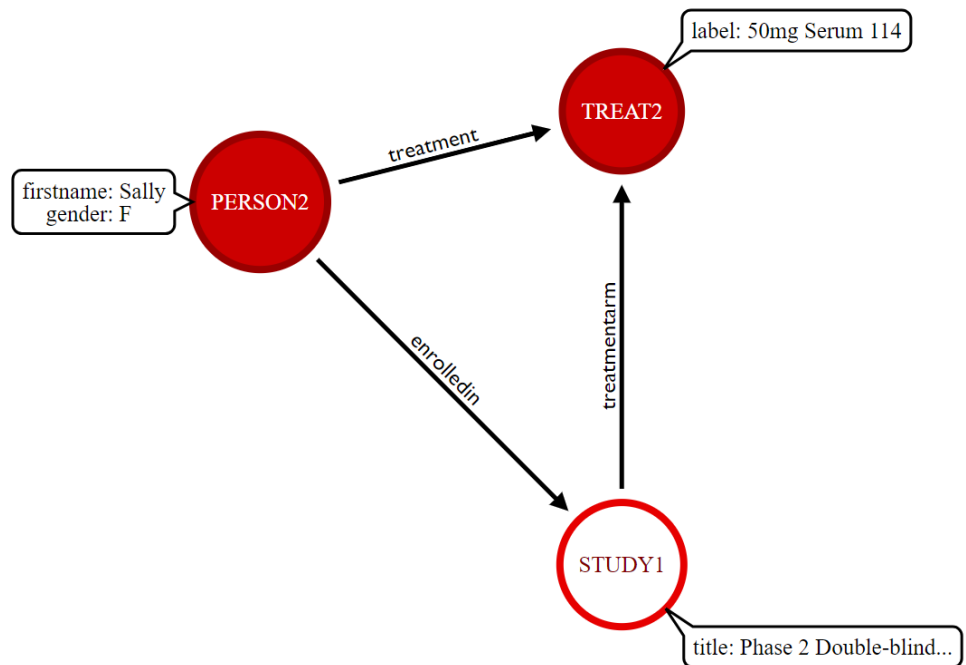
Example 1: Add study phase to the existing STUDY1 node.



Example 2: Add a protocol to the study. Consider how the protocol node may relate to other nodes that are a part of the clinical trials design.



Example 3: 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.

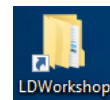


	Stop here and wait for the instructor.	
--	--	--

1.2 Transfer Diagram to Spreadsheet

You will now create a machine readable representation of your diagram by transferring the nodes, relations, and p:v pairs to a spreadsheet.

1. **Sign in to the cloud server** as described in the [Server Login](#) section.
2. Open Windows Explorer to the Linked Data folder location using the desktop shortcut
3. Navigate to the content within the **.../data** subfolder and double click on the file **Neo4jModel.xlsx** to open the spreadsheet.
4. Observe how the spreadsheet is divided into two tables as shown in [Figure 1](#).



	A	B	C	D	E	F	G
1	Table 1. Nodes and Relations				Table 2: Node P:V Pairs		
2	StartNode	Relation	EndNode		Node	Property	Value
3	PERSON1	enrolledin	STUDY1		PERSON1	firstname	Bob
4	PERSON1	treatment	TREAT1		PERSON1	age	32
5	STUDY1	treatmentarm	TREAT1		STUDY1	title	Phase 2 Double-blind study of Serum 114
6					TREAT1	label	Placebo
7					TREAT1	description	Sugar water
8							
9							
10							
11							
12							

Figure 1 Neo4jModel.xlsx

Table 1. Nodes and Relations

Table 1 is used to list each node in the diagram and the relationships between nodes. Rows shaded in grey represent entities in the original model provided by the instructor. Each row contains 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

- **PERSON1** is *enrolledin* **STUDY1** and also has *treatment* **TREAT1**, so PERSON1 is listed twice.
- A node at the end of one relation (EndNode) can also be the StartNode in another relation. Observe how STUDY1 is both an EndNode and a StartNode.

5. Add your new StartNodes, Relations, and EndNodes under the shaded rows in Table 1. Make sure 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. Every node listed in Table 1, as either a StartNode or an EndNode, must appear at least once in Table 2, because the exercises require nodes to have at least one p:v pair. Many nodes have more than one p:v pair and so are listed on multiple rows. For example, PERSON1 has a row for the **firstname** property and a second row for the **age** property.

6. Enter your new node 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.

Review

7. Review the values you entered in each table.

a. Table 1. Nodes and Relations

Confirm the values you entered:

- Match your diagram.
- Follow the naming conventions, including capitalization.


b. Table 2. Node P:V Pairs

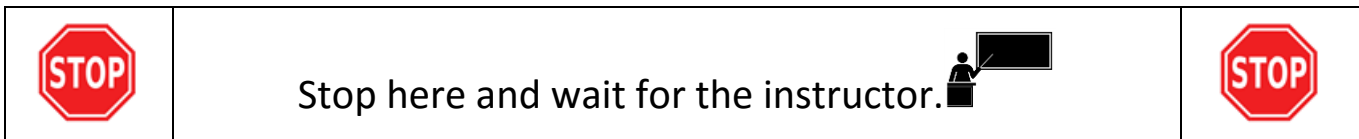
Confirm the values you entered:

- Match your diagram.
- Follow the naming conventions, including capitalization.
- 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.

i 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 (but does not fit the purpose of our exercises).

8. Save the spreadsheet using the **File | Save** menu or disk icon  and minimize (or close) the spreadsheet application.

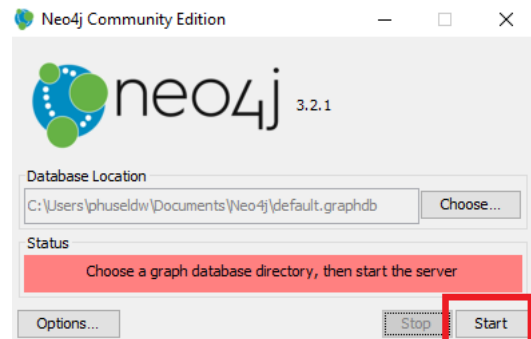


1.3 Upload to Neo4j

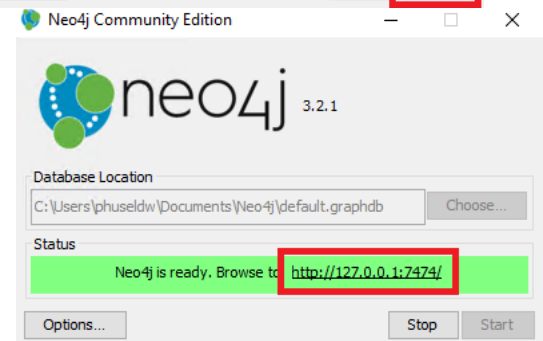
Use R to convert the spreadsheet to graph data and insert it into the Neo4j database. You must first initialize an empty Neo4j database.

1. Start Neo4j by double-clicking on the application shortcut. 

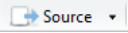
2. Accept the default **Database Location** shown in the dialog box and click **Start**.



3. The red bar changes to yellow, then to green when the database is ready.
4. Click on the http:: address in the **Browse to** field (within the green bar) to launch Neo4j in a web browser.
5. Minimize the browser window to return to the desktop.



6. Use Windows Explorer to navigate to the content within the folder **C:\LinkedDataWorkshop\scripts\r** , then double click on the file **Neo4jFromSpreadsheet.R** to open it in RStudio.

7. Execute the R script by clicking on the Source button .
8. Review the R Console window.
 - a. If the script ran successfully you will see the message:
 Success! Neo4j data available at <http://localhost:7474/browser/>
 - b. If an error occurred, consult [Appendix 1](#) for possible resolutions. Ask for assistance if needed.

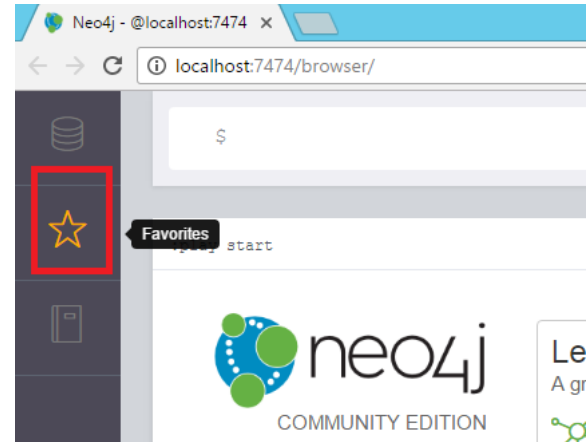
1.4 Query and Visualize

View the graph in the Neo4j application.

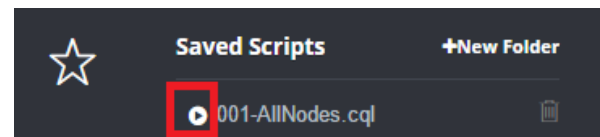
1. Return to the Google Chrome browser where Neo4j was opened in a previous step. If you closed the browser you can re-open it by clicking the link in the Neo4j application popup window displayed on the server desktop (see Section 1.3 Step 4).
2. Click on the star icon in the application side bar to view the **Saved Scripts** prepared for these exercises.



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



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

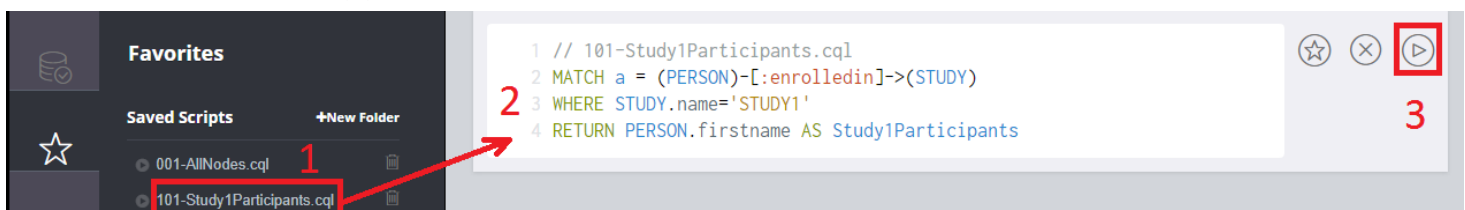


Correct any errors or omission in your Neo4j graph by returning to section 1.2 to change the spreadsheet followed by the steps in section 1.3 to reload the data into Neo4j. The load process deletes the graph from the previous run before updating with new data.

1.4.1 Who enrolled in Study 1?

Query the graph to find the first names of patients in STUDY1.

1. Click on the name of the Saved Script "**101-StudyParticipants.cql**" (1). This places the Cypher query in the execution window (2).



2. Review the query to understand how it follows the path of graph data to return the result.

101-STUDY1Participants.cql

```
MATCH enrollGraph = (PERSON)-[:enrolledin]->(STUDY)
WHERE STUDY.name='STUDY1'
RETURN PERSON.firstname AS STUDY1Participants
```

3. Click on the execution button to the right of the query text (3) to execute the query.
4. Observe the outcome in the results window.

1.4.2 What treatment did Bob receive? (Graph result)

Determine the treatment Bob received in STUDY1. Return the result as a graph.

1. Execute the query **102-BobTreatmentGraph.cql**.

102-BobTreatmentGraph.cql

```
MATCH treatGraph = (PERSON)-[:treatment]->(TREAT)
WHERE PERSON.firstname='Bob'
RETURN treatGraph
```

1.4.3 What treatment did Bob receive? (Value result)

Determine the treatment Bob received in STUDY1. Return the result as a value.

1. Execute the query **103-BobTreatmentValue.cql**. The result is similar to the previous query but this time a value is returned instead of a graph.

103-BobTreatmentValue.cql

```
MATCH treatGraph = (PERSON)-[:treatment]->(TREAT)
WHERE PERSON.firstname='Bob'
RETURN PERSON.firstname AS Name, TREAT.label as
Treatment, TREAT.description AS Description
```

1.4.4 Graph Exploration

Explore your graph while you wait for the class to catch up.

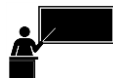
- Try writing and executing your own queries on the data you created.
- You can load and modify one of the saved scripts to create your own query.
- Use the **001-AllNodes.cql** script to display the entire graph to remind you of the nodes, relations, and properties that can be queried.



Neo4j is capable of queries that are much more complicated than those shown here. See the [Course Resources](#) to learn more about Neo4j and the Cypher query language.

This is the end of exercises for Neo4j.

Stop here and wait for the instructor.



Presentation follows



2. Resource Description Framework (RDF)

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

2.1 RDF Model Spreadsheet to the RDF Diagram

In section [1.3](#) you ran an R script that uploaded a spreadsheet into the Neo4j database. This script also inserted the values into a second spreadsheet that will be used to create RDF.

1. Navigate to the content within the **.../data** subfolder and double click on the file **RDFModel.xlsx** to open the spreadsheet.
2. Observe how the values in the rows shaded in grey (**Figure 2**) are represented in the **RDF Diagram** supplied by the instructor. These values are from the initial model, before you added your content. Your values appear below the grey rows.

	A	B	C	D
1	Subject	Predicate	Object	ObjectType
2	PERSON1	firstName	Bob	string
3	PERSON1	age	32	int
4	PERSON1	treatment	TREAT1	uri
5	PERSON1	enrolledIn	STUDY1	uri
6	STUDY1	title	Phase 2 Double-blind study of Serum 114	string
7	STUDY1	treatmentArm	TREAT1	uri
8	TREAT1	label	Placebo	string
9	TREAT1	description	Sugar Water	string
10	PERSON2	enrolledin	STUDY1	
11	PERSON2	firstname	Sally	
12	PERSON2	gender	F	
13	PERSON2	treatment	TREAT2	
14	PROTOCOL1	title	Phase 2 Trial of Serum 114 in patients with acute episodes of ultraviolence	
15	STUDY1	phase	II	
16	STUDY1	protocol	PROTOCOL1	
17	STUDY1	treatmentarm	TREAT2	
18	TREAT2	label	50mg Serum 114	
19				

Figure 2 RDFModel.XLSX with values from instructor's example.



Recall how RDF does not use property:value pairs on nodes and edges. Observe how *Subject --> Predicate --> Object* relations attach values like the age (32) to the PERSON1 node. Property:value pairs in Neo4j become S-P-O relations in RDF.

3. Starting below the grey rows, transfer at least some of your Subject, Predicate, Object values from the spreadsheet to the RDF Diagram using pencil or pen. The objective of this step to show how your values are represented on the whiteboard.



In the RDF Diagram, note how the nodes supplied by the instructor are shown in **green** for string values and in **blue** for integers. 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 the *capability* to

link to others) are shown in red. This distinction becomes important when you assign ObjectType values in the spreadsheet. You do not need to draw your nodes differently - just be aware of the different types of values you are representing.

4. If you have many new rows in your spreadsheet, only transfer a small number of them until you understand the relationship between the whiteboard and the spreadsheet. This will also help you in the next steps where you will assign an ObjectType to each Object.
5. Compare your **RDF Diagram** with your **Neo4j Diagram**. What are the similarities and differences?

2.2 Assign ObjectType


You must now enter the **ObjectType** value for the Object in each Subject --Predicate --> Object in the spreadsheet. RDF supports many data types. For simplicity, this exercise uses only three:


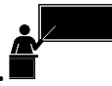

uri	Object nodes that link to other nodes in your existing model, or represent things that <i>could</i> link to other nodes. These Objects would become Subjects in another relation, or <i>could</i> become Subjects if more data was available.
string	Character values with no outbound link. Examples: "Bob", "Protocol for Study 123", "Male", "11-Oct-2017", etc. Dates can be represented as strings for this exercise.
int	Integer values with no outbound link. Example: 32

1. Compare the values for ObjectType already assigned in the grey rows of the spreadsheet to the Object nodes in the RDF diagram.
2. Assign ObjectType values to your new Objects in the spreadsheet.
 - Always assign the same ObjectType value if you are repeating an existing node. Example: If you enrolled a second person in Study 1 (PERSON2 -- enrolledin -- STUDY1), STUDY1 is previously defined in the spreadsheet with ObjectType = uri, so ensure you use the same type (uri) for this object.
 - Be consistent with existing types. For example, if you added a new Person (PERSON2), or a second treatment (TREAT2), these receive the same objectType as the corresponding PERSON1 and TREAT1 nodes (ObjectType= uri) because they are the same type of entity.
 - Labels are given a value of ObjectType=string.
 - Blank ObjectTypes are not allowed in the exercises and will cause the conversion script to fail.



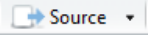
Ask for assistance if you are unsure which ObjectType value should be assigned to a node in your model.

3. Save the spreadsheet using the **File | Save** menu or disk icon  and minimize (or close) the spreadsheet application.

	Stop here and wait for the instructor. 	
---	--	---

2.3 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.

1. Use Windows Explorer to navigate to the content within the folder **C:\LinkedDataWorkshop\scripts\r**, then double click on the file **RDFFromSpreadsheet.R** to open it into RStudio.
2. Execute the R script by clicking on the Source button .
3. Review the R console window for errors.

The R script runs validation checks on your TTL file using Apache RIOT. You receive no message if the file is valid RDF. Error messages identify the location and type of error:

ERROR [line: *n*, col: *n*] *type of error*

Ask for assistance if you encounter an error message. Double click the **RDFModel.TTL** file in the /data folder to open it into Notepad++ for troubleshooting.

4. The script outputs RDF as the file: **C:\LinkedDataWorkshop\data\RDFModel.TTL**
5. Double click the TTL file to open it into Notepad++. The file will appear similar to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix phuse: <http://www.example.org/phuse/workshop/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

phuse:PERSON1
  phuse:age "32"^^xsd:int ;
  phuse:enrolledin phuse:STUDY1 ;
  phuse:firstname "Bob"^^xsd:string ;
  phuse:treatment phuse:TREAT1 .

phuse:STUDY1
  phuse:phase "II"^^xsd:string ;
  phuse:protocol phuse:PROTOCOL1 ;
  phuse:title "Phase 2 Double-blind study of Serum 114"^^xsd:string ;
  phuse:treatmentarm phuse:TREAT1, phuse:TREAT2 .

phuse:TREAT1
  phuse:description "Sugar Water"^^xsd:string ;
  phuse:label "Placebo"^^xsd:string .
```

6. Observe how the nodes and relations in the RDF Diagram were translated into the data in the RDFModel.TTL file. Note the S,P,O relations and the representation of URI's, Integer values (xsd:int), and strings (xsd:string).

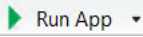


Stop here and wait for the instructor.

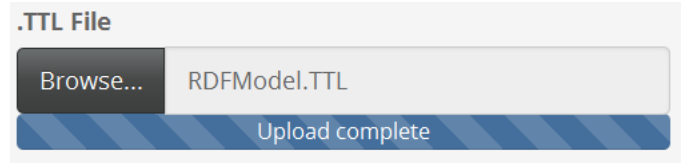


2.4 Query and Visualize

You will now use an R Shiny app to query and visualize the RDF file.

1. Open the file C:\LinkedDataWorkshop\scripts\r\SelectTTLToQuery.R into RStudio.
2. Run the app by clicking the **Run App** icon .

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.

SPARQL Query

```
# Default Query
PREFIX phuse: <http://www.example.org/phuse/workshop/>
SELECT *
WHERE {
  ?s ?p ?o
  FILTER (REGEX(STR(?s), 'phuse'))
}limit 10
```

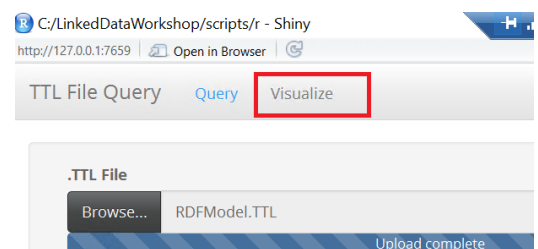
Run query

Query Result:

Show entries Search:

s	p	o
phuse:STUDY1	phuse:treatmentarm	phuse:TREAT2
phuse:STUDY1	phuse:treatmentArm	phuse:TREAT1
phuse:STUDY1	phuse:title	Phase 2 Double-blind study

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



7. The graph will look like **Figure 3**.

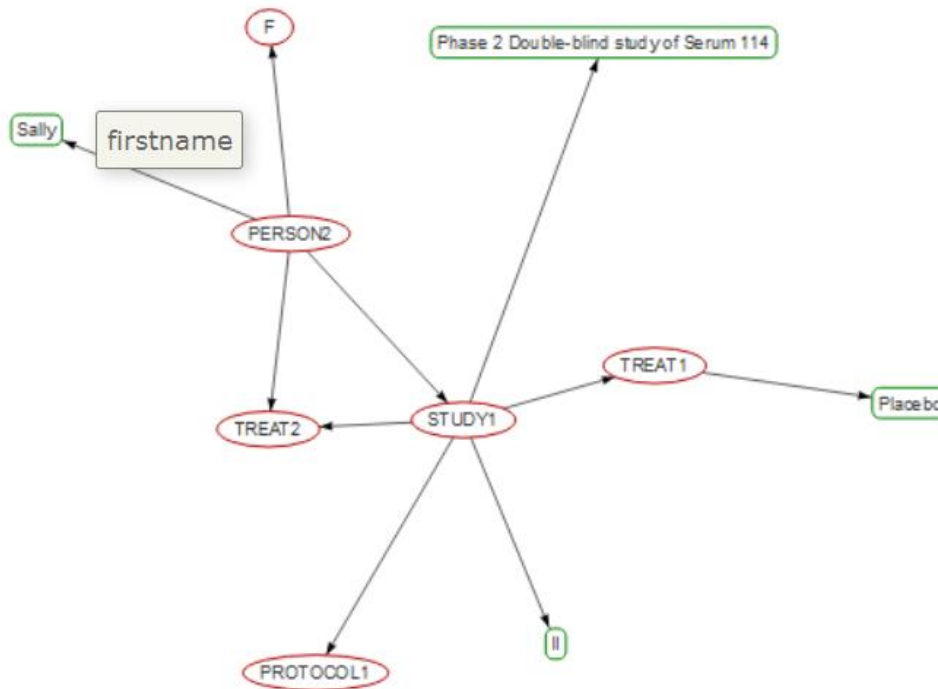


Figure 3 RDF as Force Network Graph

8. Use your mouse or touchpad to pan and zoom the graph.
9. If not all nodes are present, increase the value of the LIMIT statement in the query, re-execute the query, and click the Visualize tab again.
10. Explore your graph by clicking on nodes. Mouse-over the links to show the relationship labels. You may also use the drop-down selections for highlighting node categories and groups.



Return to the full graph visualization at any time by loading and executing the script `/scripts/SPARQL/200-EntireGraph.rq`

2.4.1 Who enrolled in Study 1?

Query the graph to find the first names of patients in STUDY1.

1. Click on the **Query** tab in the R Shiny 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.

TTL File Query **Query** Visualize

.TTL File
Browse... RDFModel.TTL
Upload complete

OPTIONAL: .RQ Query File
Browse... No file selected

SPARQL Query

201-STUDY1Participants.rq

```
# 201-STUDY1Participants.rq
PREFIX phuse: <http://www.example.org/phuse/workshop/>
SELECT ?name
WHERE {
  ?person phuse:enrolledin phuse:STUDY1 .
  ?person phuse:firstname ?name .
}
```

- Click **Run query** to execute the query and view the first names of patients in STUDY1 in the **Query Result** area of the app. **There is no visualization for this result.**



The R Shiny app requires query results to be returned in the ?s, ?p, ?o format in order to visualize the Subject-->Predicate-->Object relations in the **Visualize** tab. The visualization does not accommodate single nodes or SPARQL results that return anything other than ?s, ?p, ?o. When a query result does not meet these conditions, the **Visualize** tab displays:

Error: incorrect number of dimensions

This is a limitation of the R Shiny app created for the workshop.

2.4.2 What treatment did Bob receive? (Triples + Graph result)

Determine the treatment Bob received in STUDY1. Return the result as triples and a graph.

- Use the field **OPTIONAL: .RQ Query File** to load the query file **202-BobTreatmentGraph.rq** into the query window.
- Run the query and view the result in the **Visualize** tab. The query is a union of statements necessary to format the result in the s,p,o format required for the visualization, so the query text is not presented here in the exercises for examination. It is more important that you understand the query and result in section, [2.4.3](#).

2.4.3 What treatment did Bob receive? (Value result)

Determine the treatment Bob received in STUDY1. Return the result as triples (no visualization).

- Use the field **OPTIONAL: .RQ Query File** to load the query file **203-BobTreatmentValue.rq** into the query window.

203-BobTreatmentValue.rq

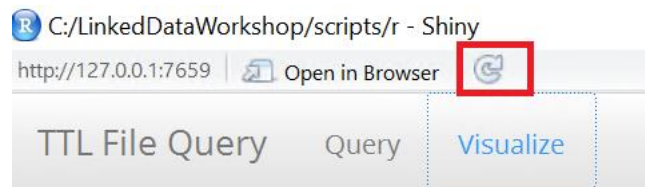
```
# 203-BobTreatmentValue.rq
# Return Bob's treatment. No visualization
PREFIX phuse: <http://www.example.org/phuse/workshop/>
SELECT ?treatLabel
WHERE {
  ?person phuse:firstname "Bob" .
  ?person phuse:treatment ?treat .
  ?treat phuse:label ?treatLabel .
}
```




- Execute the query to view the result. There is no visualization for this result.

2.4.4 Graph Exploration

Explore your graph 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.
- The query /scripts/SPARQL/**200-EntireGraph.rq** shows the full graph to assist you with writing new queries.
- Note that the visualization does not show the phrase: prefix for predicates (links/edges). You must use the prefix in SPARQL query predicates.
- If you receive the error "**Incorrect number of dimensions.**" and are sure that your query structure is correct, try reloading the Shiny App using the reload icon. Then reload the TTL and try the query again. Note that the reload will remove your query, so make a copy of it prior to reloading the app!



	<p>Stop here and wait for the instructor.</p>  <p>Presentation follows</p>	
---	---	---

Congratulations! You've made to the

---- END OF THE EXERCISES ----

Demonstrations

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

Files and slides will be provided to all attendees in the days following the workshop.

Appendix 1: Error Messages and Resolutions

A1.1 Neo4jFromSpreadsheet.R : 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 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. Nodes and 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: <ul style="list-style-type: none"> • 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 needed.
Error: FileNotFoundException (Java): data\RDFModel.xlsx (The process cannot access the file because it is being used by another process)	You have the spreadsheet RDFModel.xlsx open and in a locked state. Close RDFModel.xlsx and re-run Neo4jFromSpreadsheet.R

A1.2. RDFFromSpreadsheet.R : Error Messages and Resolutions

R Console Error Message	Cause and Resolution
Error! Same Object defined as different ObjectTypes Object:STUDY1 ObjectTypes: string,uri Correct in spreadsheet then re-run this script.	An Object is defined as more than one ObjectType in the spreadsheet. Assign the same type to the Object name. Save the spreadsheet and execute the R script again.
ERROR: At least 1 ObjectType value is missing. Make corrections in the source spreadsheet Followed by: Error in if ((RDFModel[i, "Object"] == RDFModel[i - 1, "Object"]) & (RDFModel[i, : missing value where TRUE/FALSE needed	An ObjectType value is missing. Add the ObjectType value. Save the spreadsheet and execute the R script again.

A1.3 Unprotect .XLSX Spreadsheet in Libre Office



You should only need to change values in the ObjectType column. All other cells are protected. If you must change other cells (for example, to change row height to view values) you must unprotect the sheet using these steps:

1. Right-click on the sheet name **RDFModel**.
2. Choose **Protect Sheet**.
3. In the Unprotect Sheet dialog window, enter the password **phuseldw**.
4. You will now be able to alter cells in the spreadsheet. USE CAUTION: Making changes outside of the ObjectType column may cause problems!

Appendix 2: Course Resources

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](#)

Course content is available at:

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