PhUSE CSS

Linked Data and Graph Database

Hands-on Workshop

EXERCISES

Version 2.0

October 2017

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

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

Contents

[Introduction and Disclaimer 2](#_Toc484442224)

[Server Login and Preparation 2](#_Toc484442225)

[Exercises 3](#_Toc484442226)

[1. Neo4j 3](#_Toc484442227)

[1.1 Sketch the Data Model 3](#_Toc484442228)

[1.2 Transfer Model to Spreadsheet 5](#_Toc484442229)

[1.3 Upload to Neo4j 5](#_Toc484442230)

[1.4 Query and Visualize 5](#_Toc484442231)

[2. Resource Description Framework (RDF) 6](#_Toc484442232)

[2.1 Sketch the Data Model 6](#_Toc484442233)

[2.2 Transfer Model to Spreadsheet 8](#_Toc484442234)

[2.3 Create RDF (TTL) File 9](#_Toc484442235)

[2.4 Visualize the Data 9](#_Toc484442236)

[Demonstrations 11](#_Toc484442237)

[7. SDTM as LPG 11](#_Toc484442238)

[8. SDTM as RDF 11](#_Toc484442239)

# Introduction and Disclaimer

Instructions in this document are specific to the cloud server instance and files 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.

Your feedback is welcomed and encouraged. Please send your comments to:

**Tim Williams**

**tim.williams@PhUSE.eu**

# Server Login and Preparation

Login to the cloud server provided for the exercises. Instructions assume Windows OS on your local machine.

Connect to the server using Remote Desktop

1. In the search box on the taskbar, 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

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

**Section 1:** Labeled Property Graph Neo4j

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

**Demo 1:** SDTM data in Neo4j

**Demo 2:** SDTM data in RDF

# 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 the diagram in Figure 1 **.** 
   1. Note how individual nodes represent a Person, a Study, and a Treatment within a study.
   2. Property:value 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 property:value pairs on these links. To keep the exercises simple, ***you will not attach property:value pairs to the links in these exercises***.
2. Add nodes to the diagram in Figure 1 .
   1. Using a pen or pencil add new nodes and relations to the diagram. Pencil is preferred so you may adjust nodes and relations as you develop your data model.
   2. Add a few property:value pairs to your new nodes.
   3. 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).

***It is recommended to 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**

The suggestions here are to facilitate the translation and formatting of the data in both Neo4j and RDF. They are recommendations only for these exercises and not for Linked Data generally.

**Nodes**

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

**Node Property:Value pairs**

|  |  |
| --- | --- |
| **Guideline** | **Examples/Explanation** |
| Use property names consistent with existing ones for the same concept | age, not AgeYRS |
| Integers, characters, strings with spaces are all acceptable. |  |
| Avoid special characters ($"%&^!~, etc.) | Special characters may have unanticipated effects in the conversion scripts. |
| Add any new types of properties you wish (gender, height, etc.) to existing and new nodes. |  |

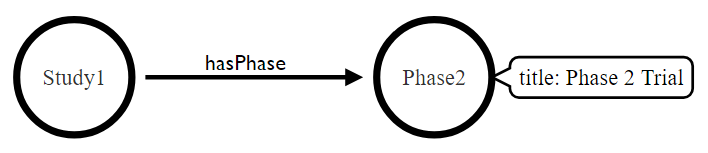
**Relations (Links/Edges)**

|  |  |
| --- | --- |
| **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. |  |
|  |  |

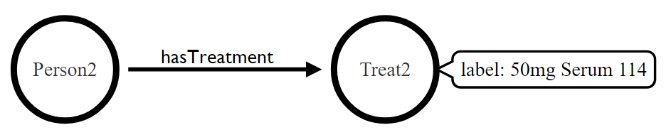
**Examples**

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

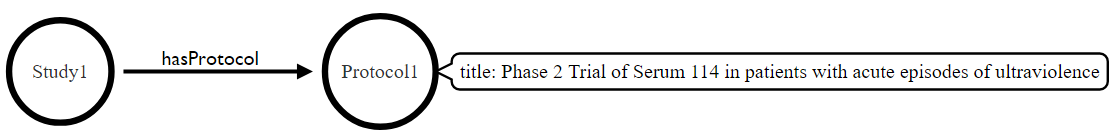
Example 1: Add study phase information to the existing study node.



Example 2: Add a new Person node (Person 2) and assign them to a new treatment node (Treatment2). Consider adding Gender value to the new person and attaching the new treatment node to the existing study node.



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



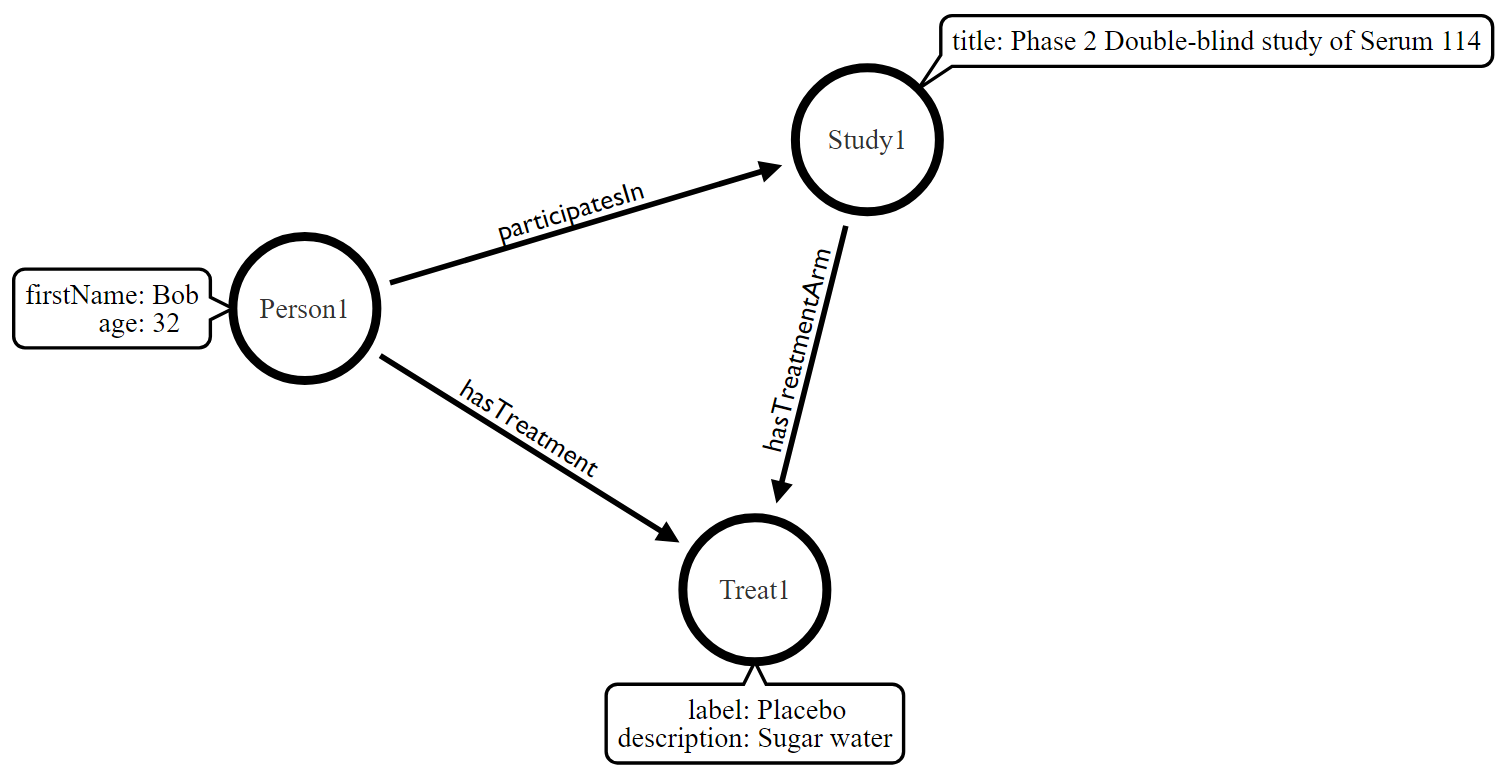


Figure 1 Neo4j Model Sketch

# Transfer Diagram to Spreadsheet

The diagram will now be translated into spreadsheet that records the nodes, property:value pairs, and relations. See Figure 2 for a screen shot of the initial 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 sections.
   1. **Relations**

The table on the left lists the Node- to-Node relationships. This section lists 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.

Examples

* Person 1 participates in Study 1 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. See Study1 which is both a StartNode and EndNode.
  1. **Node PV**

The table on the right lists the Property:Value pairs attached to each node. Each row lists one Property:Value Pair, so nodes like Person1 are listed on two rows: one for the **firstName** property, and again for the b property.

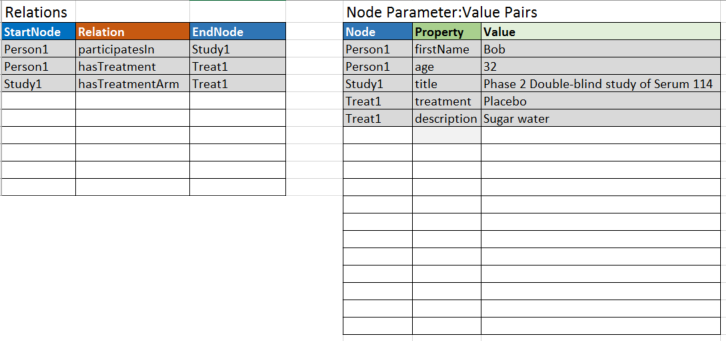


Figure 2 Neo4j Spreadsheet

1. Transfer the nodes, relations, and property:value pairs from your diagram to the Neo4jModel spreadsheet.

**Relations**

* 1. Start with the table on the left and enter your new Relations, listing the StartNode, Relation, and EndNode for each, using the rows under those shaded in grey that are reserved for the initial model. Ensure you capture all the relations, especially those where nodes participate in more than one relationship.

**Node PV**

* 1. Enter your new nodes Property:Value pairs in the table on the right. List each new Property:Value 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 on more than one row.

1. Cross-check the tables

**Relations**

* 1. Ensure the names in the StartNode and EndNode columns are named consistently, without error, and match the names in the diagram exactly.

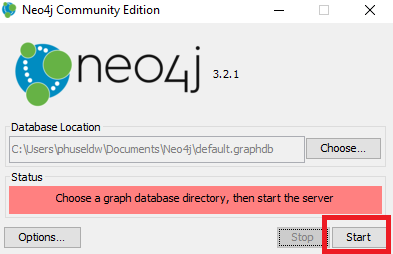
**Node PV**

* 1. Ensure the names in the Node column are named consistently, without error, and match the names in the diagram exactly.
  2. All unique StartNode and EndNode names in the Relations table must be present in the **Node PV** table. If a node is named in the4 Relations table without being present in the table on the right, the conversion process in the next section will fail.

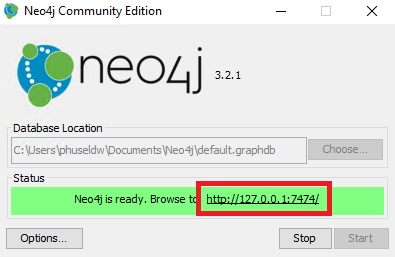
It is acceptable to have node names in the **Node PV** table without them appearing in the Relations table. These nodes will be present as isolated nodes with no relation to other nodes.

# Upload to Neo4j

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



1. 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 **Browse to** URL to launch Neo4j in a web browser.
3. There is currently no data in the database You will return to the web browser after you create and upload the spreadsheet using R.
4. Minimize the browser window to return to the desktop
5. Use Windows Explorer to navigate to the folder **C:\LinkedDataWorkshop\scripts** , then double click on the file **Neo4jFromExcel.R** to open it into RStudio.
6. Execute the R script by clicking on the Source toolbar button 
7. Review the R Console window.
8. If the script ran without error, you will see the message:

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

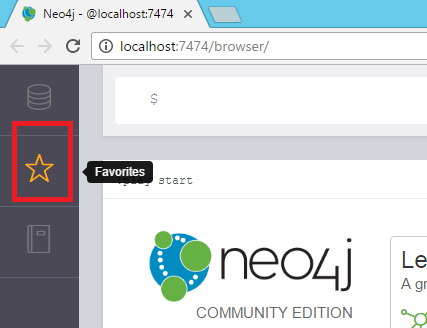
1. If an error occurred, use Table 1 to help diagnose the problem and contact the instructor or assistant for help if you need help.

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 the Relations table as a StartNode or EndNode is not defined in the Node PV table. The node name will be listed in the console message.  Possible resolutions include:  Adding the node name into the Node PV table  Correcting the node name in the Relations to match a name in the Node PV table. |
| 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 is listed in the Node PV table is absent from the Relations table. This situation results in a node that is not connected to other nodes.  Having an isolated node that is not connected to other nodes is perfectly acceptable. It may not have been intended for these exercises, so review your data model and spreadsheet. Re-run script if changes were needed to the spreadsheet. |

# Query and Visualize

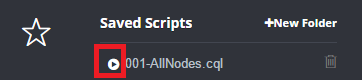
1. Return to the Neo4j Browser window you opened in a previous step. If you closed the browser you can re-open it by clicking the link in the Neo4j window.

****

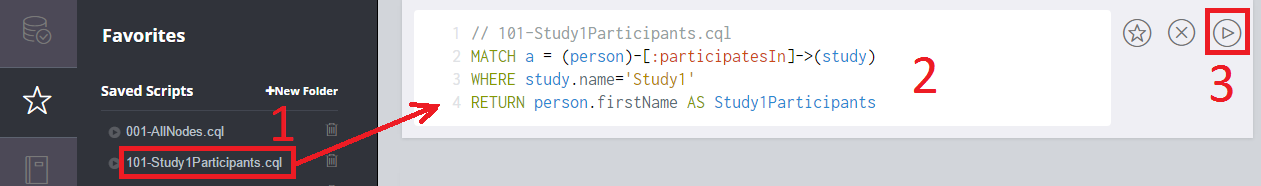
1. Click on the star icon in the application side bar to view the **Saved Scripts** prepared for these exercises.

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.



1. Explore the graph and compare it with your diagram in Figure 1
2. Use drag-and-drop to position the nodes with your mouse.
3. Click on a node and view the node property:value pairs at the bottom of the Neo4j screen.
4. Query the graph to find the first names of patients in Study1.
5. 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.

1. Determine the treatment Bob received in the study.
2. Repeat the same steps to load and execute the following query that returns a graph result of the treatment Bob received.

102-BobTreatmentGraph.cql

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

1. Execute the following to retrieve the same result as values, 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**

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.

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.

# Sketch 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).
2. 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.
3. Referring back to your sketch Figure 1, transfer those entities to the RDF model in Figure 3 by drawing them in the diagram. It may help to remove the staple from this document in order to place the diagrams side-by-side.
4. 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.

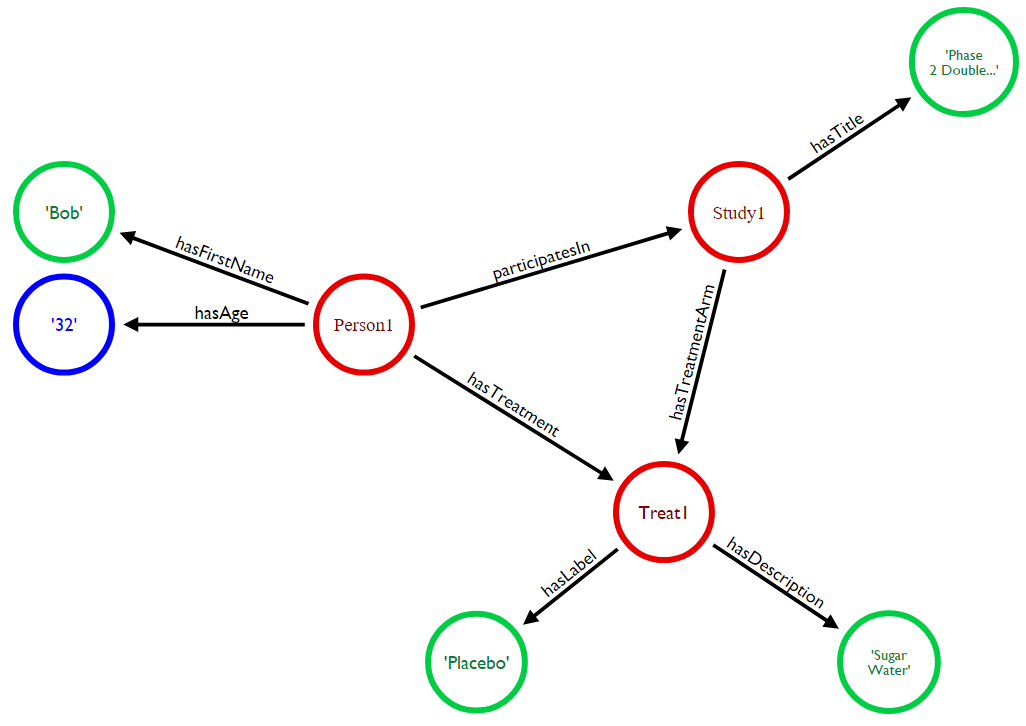


Figure 3 RDF Model Sketch

URI = red , string = green, integer = blue

# 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 sketch of the model 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.

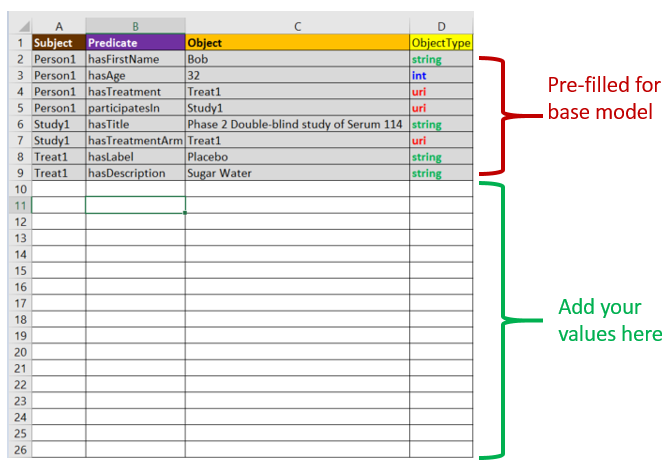


Figure 4 RDF Spreadsheet

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 Figure 5. Observe how the model in Figure 3 was translated to the data in Figure 4 and lastly into the RDF representation Figure 5 .

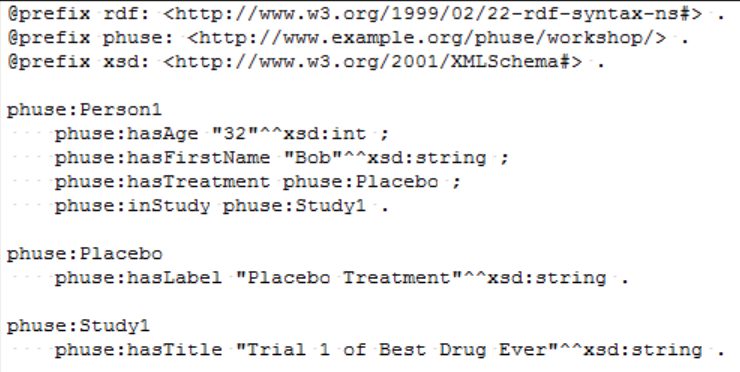


Figure 5 TTL file RDFModel.TTL

# Visualize the Data

A R script is used to load the TTL file and convert it to a force network graph.

1. From RStudio, open the file C:\LinkedDataWorkshop\**ViewRDFModel.R**
2. Execute the script to view the force network graph of the data. The graph will appear in the RStudio Viewer window and look similar to Figure 6

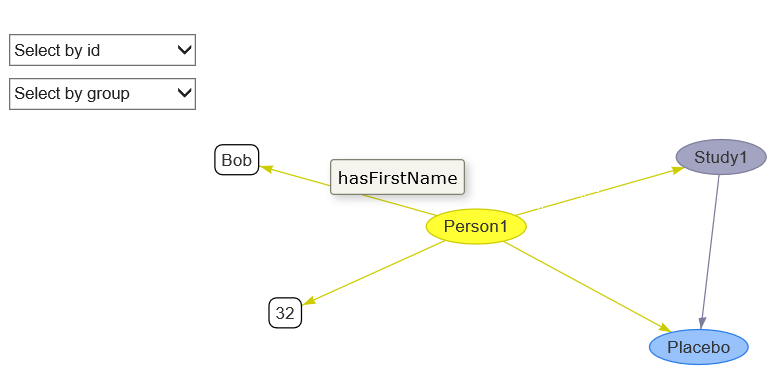


Figure 6 Force network graph of base model

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.

# Query the Data

<Details to be added for RSHINY app for opening and querying the TTL file.>

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

201-Study1Participants.rq

|  |
| --- |
|  |

1. Determine the treatment Bob received in the study.

202-BobTreatmentValue.rq

|  |
| --- |
|  |

**OPTIONAL**

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.

use the match(n) return(n) script to return the entire graph to remind you of the nodes, relations, and properties that

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

---- END OF EXERCISES ----

# Demonstrations

In this section the instructors will demonstrate example data from SDTM domains converted to the two different types of graph databases. Files are provided so you may follow along if you wish.

# SDTM as LPG

<content to be added>

# SDTM as RDF

<content to be added>

# Course Resources

<Add links for Neo4j and RDF>

**Neo4j**

**RDF**