PhUSE CSS

Linked Data and Graph Database

Hands-on Workshop

EXERCISES

Version 2.0

October 2017

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

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

Contents

[Introduction 2](#_Toc489626907)

[Required Materials 2](#_Toc489626908)

[Symbols 2](#_Toc489626909)

[Server Login 3](#_Toc489626910)

[Exercises 4](#_Toc489626911)

[1. Neo4j 4](#_Toc489626912)

[1.1 Diagram the Data Model 4](#_Toc489626913)

[1.2 Transfer Diagram to Spreadsheet 6](#_Toc489626914)

[1.3 Upload to Neo4j 8](#_Toc489626915)

[1.4 Query and Visualize 9](#_Toc489626916)

[1.4.1 Find Study Participants 10](#_Toc489626917)

[1.4.2 Treatment for Specified Patient: Graph 10](#_Toc489626918)

[1.4.3 Treatment for Specified Patient: Value 10](#_Toc489626919)

[1.4.4 Explore Your Graph 11](#_Toc489626920)

[2. Resource Description Framework (RDF) 11](#_Toc489626921)

[2.1 Diagram the Data Model 11](#_Toc489626922)

[2.2 Transfer Diagram to Spreadsheet 12](#_Toc489626923)

[2.3 Create RDF (TTL) File 13](#_Toc489626924)

[2.4 Query and Visualize 14](#_Toc489626925)

[2.4.1 Find Study Participants 16](#_Toc489626926)

[2.4.2 Treatment for Specified Patient: Graph 16](#_Toc489626927)

[2.4.3 Treatment for Specified Patient: Value 17](#_Toc489626928)

[2.4.4 Explore your Graph 18](#_Toc489626929)

[Demonstrations 18](#_Toc489626930)

[Course Resources 19](#_Toc489626931)

# Introduction

Instructions in this document are specific to the PhUSE cloud server environment and methods used for this 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 workshop is comprised of two exercises and demonstrations (time permitting):

**Exercise 1:** Labeled Property Graph Neo4j

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

**Demonstrations:** SDTM data in Neo4j and RDF

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:
  + WorkshopExercises.pdf (this document)
  + Neo4jDiagram.pdf
  + RDFDiagram.pdf

## Symbols

Symbols are used to 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. |
| IdeaIcon_clean_20mm | Helpful tips and advice. |
| Image result for information icon | Additional information. |

# Server Login

Login to the cloud server using Remote Desktop. Instructions assume Windows 10 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.﻿
2. 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

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. Examine your copy of the **Neo4j Diagram** and 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. However:

|  |  |
| --- | --- |
| 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 need to start over, ask the instructor for a fresh copy of the diagram. |

* 1. 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 only. Neo4j nodes can exists without p:v pairs. However, the conversion script requires p:v pairs on nodes and it helps illustrate the model when later converting to RDF.* |

* 1. You may also add additional property:value pairs to the nodes that are 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, not Patient2 |

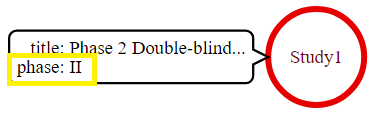
**Node P:V pairs**

|  |  |
| --- | --- |
| **Guideline** | **Examples/Explanation** |
| Each node must have at least one p:v pair. | Exercise requirement. |
| 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. |
| You may add new p:v pairs to existing nodes. | Example: Add gender, height to Person1. |

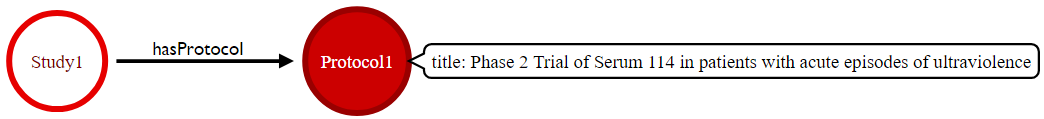
**Relations (Links/Edges)**

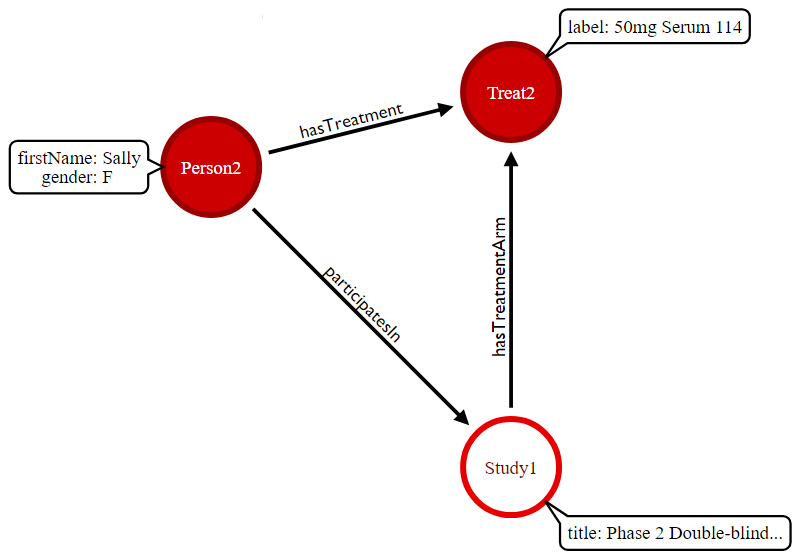
|  |  |
| --- | --- |
| **Guideline** | **Examples/Explanation** |
| No spaces or special characters ($"%&^!~…etc.) in the names | Special characters may have unanticipated effects in the conversion scripts. |
| Relationships have ***direction*** and ***describe*** how the nodes are related. Pick names that define the relation in a concise way. |  |
| No p:v pairs on the links. | Keep it simple for these exercises. Conversion scripts in the exercises to do not process p:v pairs on relations. |

**Examples**

Consider the following examples if you are having trouble defining new nodes and relations of your own.

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

Example 2: Add a protocol to the study. How would the protocol fit into a description of the clinical trials design process?

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.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | Consider other aspects of a clinical trial design and add those entities and relationships to your diagram, keeping in mind the suggested maximum of 6 new nodes, 12 new p:v pairs. |

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

# Transfer Diagram to Spreadsheet

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

1. **Sign in to the cloud server desktop** as described in the [Server Login](#_Server_Login) section.
2. Open Windows Explorer to the Linked Data folder location using the desktop shortcut
3. Navigate to 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 .**

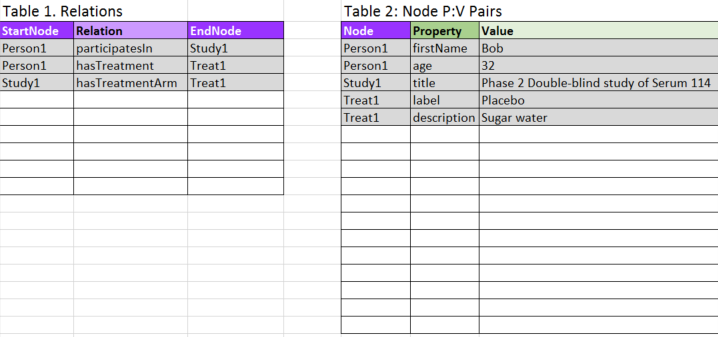


Figure 1 Neo4jModel.xlsx

**Table 1. Relations**

Table 1 contains the node-to-node ***relations***. 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** *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. Observe how Study1 is both a StartNode and EndNode.

1. Enter your new StartNodes, Relations, and EndNodes from your diagram by adding them under the rows shaded in grey that represent the original entities. 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. Every node listed in Table 1, as either a StartNode or an EndNode, must appear at least once in Table 2, since 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.

1. 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. 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.

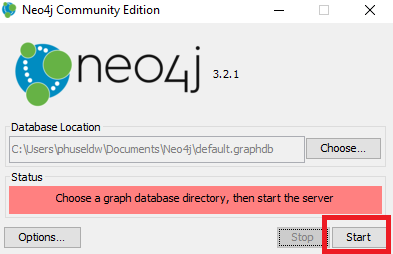
|  |  |
| --- | --- |
| **!** | **If a node name is present in Table 1 and absent in Table 2, the conversion process in the next section will fail.** |
| Image result for information icon | It is acceptable to have node names in the Table 2 without them appearing in the Table 1. These nodes will be present as isolated nodes with no relation to other nodes, which is completely acceptable (but does not match the purpose of our exercises). |

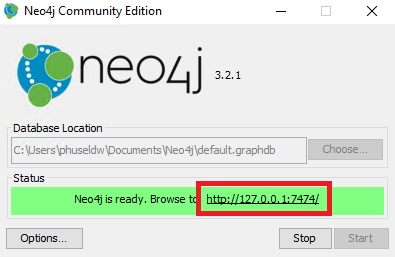
1. Save the spreadsheet using the **File | Save** menu. If prompted, ensure you use "Microsoft Excel Format" and not "ODF format."

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

# Upload to Neo4j

Use R to convert the convert the spreadsheet to graph data and insert it into the Neo4j database.

1. Start Neo4j by double-clicking on the application shortcut on the desktop. 
2. 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, then 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 **Neo4jFromSpreadsheet.R** to open it in 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 the table below for possible resolution and as for assistance if needed.

Neo4j Import Error Messages and Resolutions

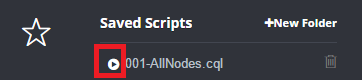
| **R Console Error Message** | **Cause and Resolution** |
| --- | --- |
| Error in curl::curl\_fetch\_memory(url, handle = handle) :  Couldn't connect to server | The Neo4j server is not started or is not available.  Return to steps 1-3 in Section 1.3 Upload to Neo4j to start Neo4j and confirm it is 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 needed. |

# Query and Visualize

View the graph in the Neo4j application.

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 application popup window displayed on the server desktop (see Section 1.3 Step 3).
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**. |



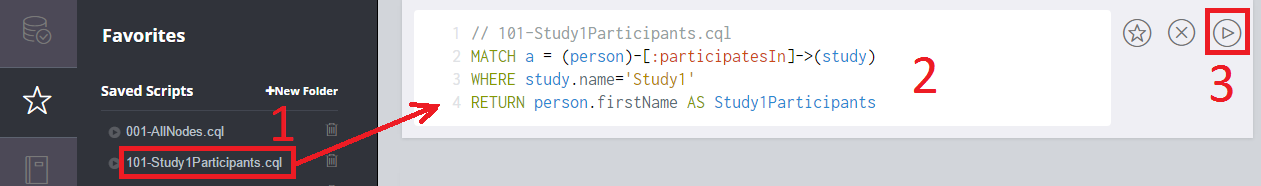
1. Click the play icon beside **001-AllNodes.cql** to show all nodes and relations in the database.
2. Explore the graph and compare it with your **Neo4j Diagram**.
3. Use drag-and-drop to position the nodes with your mouse.
4. Click on a node and view the node property:value pairs at the bottom of the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

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | 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 preexisting graphs during the upload process. |

# Find Study Participants

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 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 Study1. Return the result as a graph.

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

# Explore Your Graph

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.

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

|  |  |  |
| --- | --- | --- |
|  | 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.

# Diagram 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. Place your completed **Neo4j Diagram** beside the **RDF Diagram** provided by the instructor. Observe how the original nodes in the Neo4J model are represented in the RDF model.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | 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 must become S-P-O relations in your RDF graph model. |

1. Transfer the entities you created in your updated **Neo4j Diagram** to the **RDF Diagram** using pencil or pen.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | In the **RDF Diagram**, string values 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 the *capability* to link on to other nodes) are shown in red. This distinction becomes important when you transfer your diagram to the spreadsheet. You do not need to draw your nodes differently - just be aware of the different types of values you are representing. |
| **!** | Observe how the instructor translates the Neo4j model to the RDF model.  ***Ask for assistance if you are unsure how to add certain items to your RDF Diagram.*** |

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

# Transfer Diagram to Spreadsheet

You will now create a machine-readable representation of your RDF diagram by defining the nodes and relations in 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. Observe how the first rows of the spreadsheet contain the data from the initial model provided by the instructor. You will add your new data below the rows shaded in grey ( Figure 2 ) .

Each row in the spreadsheet represents an individual Subject --> Predicate --> Object relationship. Nodes like Person1 are listed multiple times because they are the Subject in multiple relations: *hasFirstName*, *hasAge*, *hasTreatment*, etc.

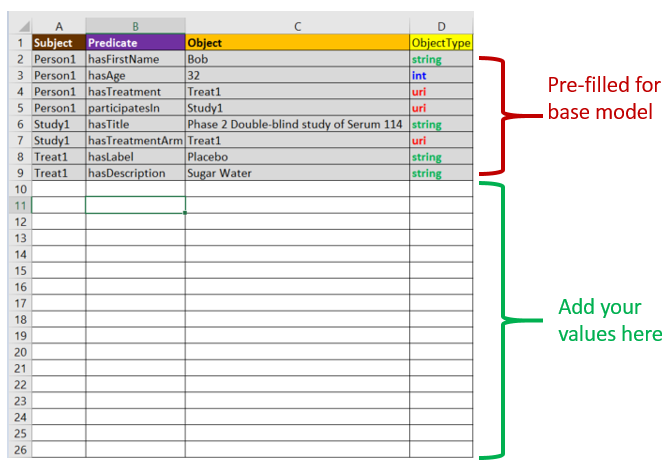


Figure 2 RDF Spreadsheet

1. Add your new Subject --> Predicate --> Object relations from the RDF Diagram to the spreadsheet.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | Only fill in the **Subject**, **Predicate**, and **Object** columns. You will complete the ObjectType column later. |

1. Confirm all the Subject --Predicate --> Object relations in the **RDF Diagram** are listed in your spreadsheet before moving to the next step.
2. You must now enter the **ObjectType** for the Object in each Subject --Predicate --> Object relation. For simplicity, this exercise uses only three ObjectTypes:

|  |  |
| --- | --- |
| **uri** | Object nodes that link to other nodes in your existing model, or represent things that ***could*** link to other nodes. These are Objects that become Subjects in another relation, or *could* 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 |

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | Many different data types are available in RDF. The exercises use only ***string***, ***int***, and ***uri*** types for simplicity. |
| **!** | Ask for assistance if you are unsure which ObjectType should be assigned to a node in your model. |

1. Save the spreadsheet using the **File | Save** menu.

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

# 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\r** , then double click on the file **RDFFromSpreadsheet.R** to open it into RStudio.
2. Execute the R script by clicking on the Source toolbar 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. |

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:

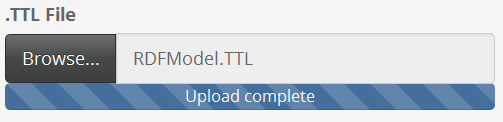
|  |
| --- |
| **@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .**  **@prefix phuse: <http://www.example.org/phuse/workshop/> .**  **@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .**  **phuse:Person1**  **phuse:hasAge "32"^^xsd:int ;**  **phuse:hasFirstName "Bob"^^xsd:string ;**  **phuse:hasTreatment phuse:Treat1 ;**  **phuse:participatesIn phuse:Study1 .**  **phuse:Study1**  **phuse:hasTitle "Phase 2 Double-blind study of Serum 114"^^xsd:string ;**  **phuse:hasTreatmentArm phuse:Treat1 .**  **phuse:Treat1**  **phuse:hasDescription "Sugar Water"^^xsd:string ;**  **phuse:hasLabel "Placebo"^^xsd:string .** |

1. 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. |  |

# Query and Visualize

You will now use an R Shiny app 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  .
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 4

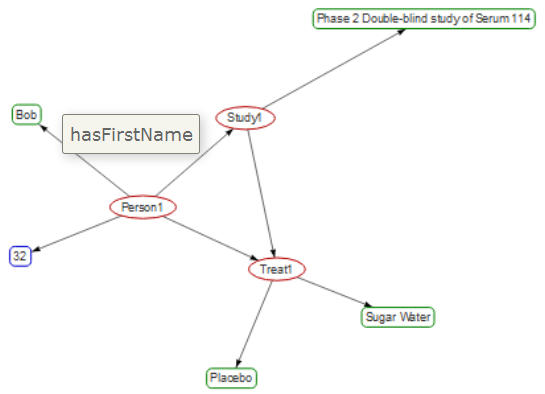


Figure 4 RDF as Force Network Graph

1. Explore your graph by clicking on nodes and mouse-over the links 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** |

# Find Study Participants

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.

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 .  } |

1. Click **Run query** to execute the query and view the first names of patients in Study1 in the **Query Result:** area of the app.

|  |  |
| --- | --- |
| **!** | 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 these exercises. |

# Treatment for Specified Patient: Graph

Determine the treatment Bob received in Study1. Return the result as triples and a graph.

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 **Query Result:** area and the **Visualize** tab. The query contains a number of BIND statements to provide the ?s, ?p, ?o result required to display the visualization.

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)  }  UNION  {  ?treat phuse:hasLabel ?label .  BIND (?treat as ?s)  BIND ( "phuse:hasLabel" AS ?p)  BIND ( ?label AS ?o)  }  } |

# Treatment for Specified Patient: Value

Determine the treatment Bob received in Study1. Return the result as triples (no visualization).

1. 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:hasFirstName "Bob" .  ?person phuse:hasTreatment ?treat .  ?treat phuse:hasLabel ?treatLabel .  } |

1. Execute the query to view the result. There is no visualization for this result, only the query result on the **Query** tab.

# Explore your Graph

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.
* Use the query /scripts/SPARQL/**200-EntireGraph.rq** to view your full graph to assist you with writing new queries.
* Note that the graph does not show the phuse: prefix for predicates (links/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 to all attendees in the days following the workshop.

# 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](https://github.com/phuse-org/LinkedDataWorkshop/tree/master/Annual2017-EU)

Course content is available at:

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