

## Interactive Visualization of Linked Data

Tim Williams, UCB Biosciences Inc., Raleigh, USA

### ABSTRACT

Linked Data facilitates traceability and data traversal by design, taking us to the next level of data exploration and analysis. However, many data consumers are unable to take advantage of these features due to their lack of experience with graph database query languages. Visualization conquers this shortcoming by providing intuitive, interactive interfaces to the underlying data.

Linked Data as Resource Description Framework (RDF) is a natural fit for web-based applications because of its storage format: HTTP Uniform Resource Identifiers (URIs). JavaScript libraries like D3js provide visually rich and interactive displays that can link directly back to the underlying web of data.

Data models in graph databases like Neo4j are represented just as they would be drawn on a whiteboard. In contrast to RDF, both nodes and relations can be assigned labels and properties. The visual model *is* the data model, enabling intuitive queries and visualization.

### INTRODUCTION

By "mapping data into visual properties" (1) we leverage our brain's highly developed ability to detect visual patterns and trends. *Interactive* visualization further engages the viewer through dynamic displays and controls. The use of web browsers for data visualization is on the rise. Long-established web technologies are given new life with recent updates such as HTML5 and CSS3. When coupled with the availability of high-quality JavaScript presentation libraries like D3js (D3), Plotly, and others, these technologies are fueling a revolution in data presentation and exploration.

Applications increasingly provide the capability to drill-down to data underlying plots, charts and summaries. Facilitating traceability and data traversal *by design*, graph database solutions have an advantage over relational data stores, where values (nodes) are connected by links (also commonly referred to as edges). In Resource Description Framework (RDF), values and relations are stored using Uniform Resource Identifiers. These HTTP URI's are a natural fit for display and navigation within web browsers.

Several PhUSE CSS projects are exploring the vast potential of RDF within the clinical trials space for both data and metadata (2). CDISC terminology is also now available as RDF (3). Both the complexity of the technology and a lack of widely-available tools for data exploration and analysis present barriers for new users. This is starting to change as pharmaceutical companies move to adopt Linked Data approaches and as training resources become available (4).

The "Property Graph" Database Neo4j lowers many of the barriers to adoption. An intuitive data model, interface, and query language has new users up-and-running quickly, making it a good choice for many Linked Data projects.

### PLOTS AND CHARTS VERSUS GRAPHS

The terms plot, chart, and graph are often used interchangeably. As noted by Alberto Cairo in his recent book, "The Truthful Art":

A chart is a display in which data are encoded with symbols that have different shapes, colors, or proportions. In many cases, these symbols are placed within a Cartesian coordinate system. The word "plot" is a [commonly used] synonym of "chart"... - Alberto Cairo (1)

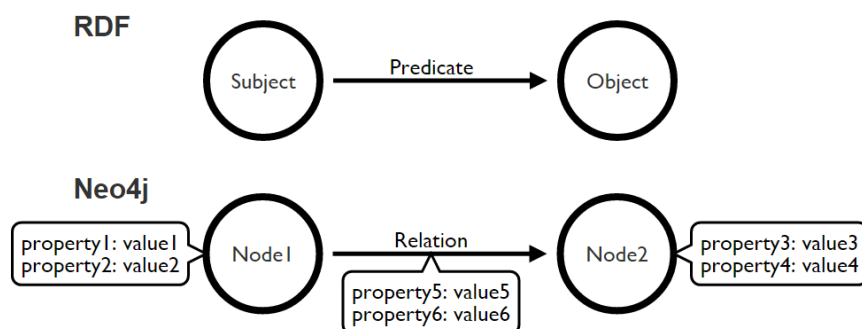
Cairo further observes that the term *graph* is claimed by practitioners of graph theory. This paper respects that assertion, reserving the word *graph* for the interlinked data in graph and RDF databases, which may in turn be displayed and accessed from interactive charts and plots. There is some unavoidable confusion when graph data is displayed using *network* and *force network graphs*. Use of "graph" in this context is appropriate terminology for a visualization that illustrates the connected nature of graph data.

## DATA REPRESENTATION IN PROPERTY GRAPHS AND RDF TRIPLESTORES

This section provides an abbreviated introduction to Neo4j and RDF. More information is available on the web, the PhUSE wiki (2), and the PhUSE "Semantics 101 for Pharma" Workshop (4).

Both Neo4j and RDF represent data as nodes (values) joined by links (edges) (**Figure 1**). Neo4j is a "Labeled Property Graph" that consists of nodes, relationships, properties, and labels. Nodes may contain properties as key-value pairs and can be tagged with one or more labels for identification and grouping. Labels typically indicate the role that a node plays within the dataset. Nodes are connected by relationships that contribute to the overall structure of the graph. Relationships are *directed*; they have a start node and an end node. The relationships themselves may have properties as key-value pairs. This relationship metadata can also be used as part of a query.

In contrast to Neo4j, RDF stores data in triples defined as Subjects related to Objects by a Predicate. Similar to Neo4j, the relationship is directional, but in RDF the direction is always from the Subject to the Object. The Object of one relation may be the Subject of another, and *vice versa*. The Predicate may become a Subject or an Object in other relations, leading to a web of interconnected data. RDF does not have the concept of properties on nodes and edges. Rather, labels and properties and other metadata are added to a graph using additional triples as part of the data model. See the RDF Data Cube (5) for examples. A discussion of RDF ontologies, reasoning, and inferencing is beyond the scope of this paper.



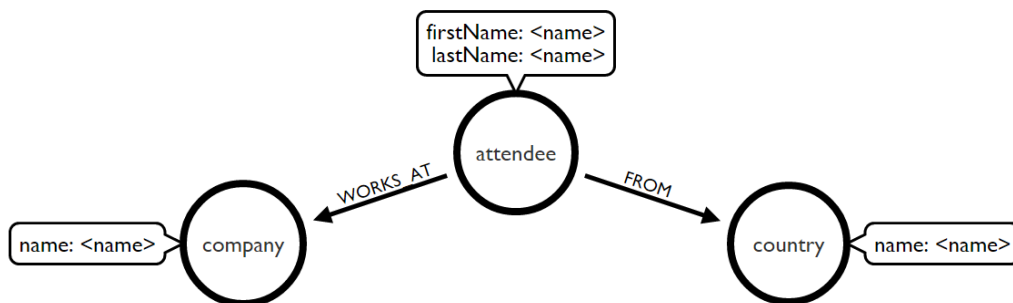
**Figure 1** Comparison of Neo4j and RDF Nodes and Relations

Both property graphs and RDF triplestores can be used for clinical trials data. The confidential nature of clinical trials poses a challenge when presenting at conferences and online. This paper therefore relies on data from the list of PhUSE 2016 conference attendees and other data fabricated solely for the purposes of illustration.

### NEO4J

A common first step when developing a data model is to draw the relationships in a diagram, a process often called *white boarding*. The advantage for Neo4j is that the model defined in the whiteboard sketch is almost exactly the same as the model in the database. It then follows that the database query language also corresponds closely to the whiteboard model. In this way, the real world model is represented in the database, the query language, and values returned by the query. This consistent and intuitive representation is facilitating the rapid growth of Neo4j across multiple industries.

For illustrative purposes, consider a graph of attendees at the PhUSE annual conference. A similar approach can be applied to other data sources, like patients in a clinical trial. The data in this example is very basic. Attendee nodes have property:value pairs for the first name (*firstName*) and last name (*lastName*) while both company and country nodes only have a name (*name*) property. The relations *FROM* and *WORKS\_AT* were not assigned properties in this example. The graph model in **Figure 2** is simple: the attendee *WORKS\_AT* a company, and the attendee is *FROM* a country.



**Figure 2** Neo4j whiteboard model of PhUSE attendees.

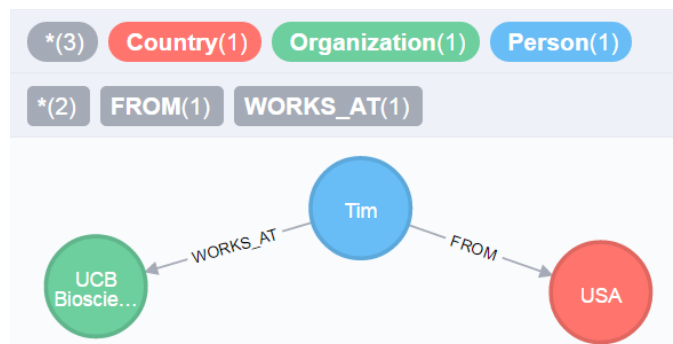
## PhUSE 2016

The white board model in **Figure 2** is reflected in the query in **Figure 3**. The pattern, written in the Neo4j query language Cypher, describes the path that connects the values of interest. Cypher queries are often called *ASCII art* because of how the query format visually matches the nodes and relations path.

```
MATCH a = (company) -[:WORKS_AT]- (attendee) -[:FROM]- (country)
WHERE attendee.firstName='Tim' AND attendee.lastName='Williams'
RETURN a;
```

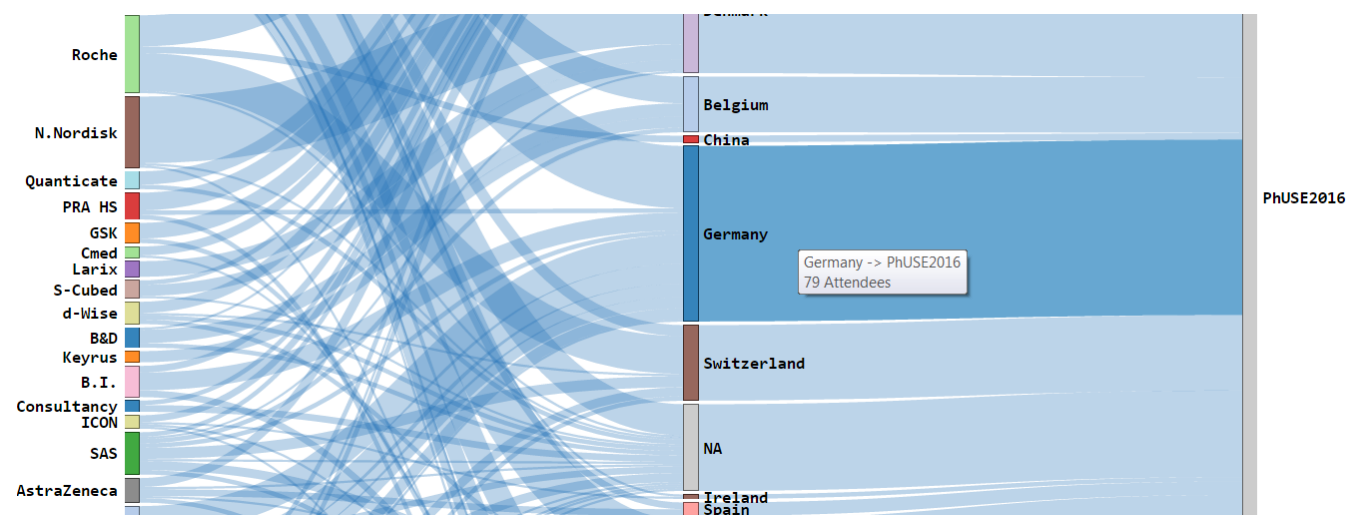
**Figure 3** Cypher query to return the graph for the PhUSE attendee Tim Williams.

The query result returned to the Neo4j browser (**Figure 4**) closely matches both the underlying data model and the Cypher query. Users may interact with the visualization to explore the data. For example, expanding the "USA" country node shows all attendees from the United States. Expanding the company node displays all attendees from the company "UCB Biosciences".



**Figure 4** Query result in the Neo4j browser for PhUSE attendee Tim Williams.

Visualization of Neo4j is not limited to the browser that comes with the application. The R package `RNeo4j` (6) provides R programmers with a familiar interface for querying and interacting with the data. Visualizations may then be piped into packages like `Rvisnetwork` for attractive, interactive displays. "Graph Visualization for Neo4j" (7) identifies a number of additional visualization options. JSON is a common intermediary data format for D3 visualizations. JSON was used in the interactive Sankey plot of PhUSE 2016 attendees, an excerpt of which is shown in **Figure 5**. A mouseover of each node in the plot reveals the number of attendees from a company or country. For clinical trials data, Sankey plots are an excellent choice to show treatment changes during the course of a study.



**Figure 5** Sankey plot of PhUSE 2016 attendees (excerpt).

## RESOURCE DESCRIPTION FRAMEWORK (RDF)

Much like property graphs, RDF is a labeled, directed multigraph (8). RDF provides machine-interpretable data and metadata and is present on both the World Wide Web and in knowledge management systems behind company firewalls. In contrast to Neo4j, a strength of RDF is in the specification of the vocabulary, with standardized Classes and Properties defined in RDF and RDF Schema (RDFS). A large number of ontologies (9) are available for standardizing terminology, representing

knowledge, and creating rule sets using Web Ontology Language (OWL). All of these properties of RDF make it well-suited for data and knowledge storage in the clinical trials space.

The challenges of learning RDF and SPARQL can be overcome with the use of appealing, interactive visualizations that serve as an interface to the underlying data, where the inherent connectivity of RDF within triplestores can be exposed. RDF is represented using HTTP URI's (10), making it a natural fit for browser-based display and interaction. The displays can provide hyperlinks directly into the faceted browser view of the Virtuoso quadstore<sup>1</sup>, with "clickable" navigation through the data. This allows users to experience the "connectedness" of Linked Data without the need to write a query.

The D3 JavaScript library is an excellent choice for Linked Data visualization and is one of the most popular visualization libraries across multiple industries, aided in part by a large number of examples available online (11). Despite the impediment of the pharmaceutical industry's historic dependence on established (and often very expensive) software, D3 is gaining popularity in our industry (12) (13) with a potential that has yet to be realized. This culture is starting to change with the rapidly expanding use of R and the adoption of other free and open source application suites.

**Figure 6** illustrates the code necessary to link from a D3 display to a URI in a triplestore. The example uses a key combination of a left mouse click ("click") while simultaneously holding down the control key (`ctrlKey`). Multiple key combinations like these are preferred because mouse-click events alone are often used to interact with visualizations in other ways.

```
var nodes = svg.selectAll("g.node")
    .data(dataset.nodes)
    .enter()
    .append("g")
    .on("click", function (d) {
        if (d3.event.ctrlKey) { location.href = d.uri; }
    })
```

**Figure 6** D3 Node Definition with 'control key+click' function referencing a URI value in a triplestore.

An example URI value for the code in **Figure 6** provides the address for a remote endpoint that contains data for a clinical trial with an NCT ID of NCT00175890:

```
http://lod.openlinksw.com/describe/?uri=http://bio2rdf.org/clinicaltrials:NCT00175890
```

When the user *ctrl+clicks* the node in a D3 chart that represents that trial, the JavaScript instructs the browser to describe the URI `http://bio2rdf.org/clinicaltrials:NCT00175890` at the site `http://lod.openlinksw.com`, where data from ClinicalTrials.Gov is hosted as linked data using Virtuoso.

If you are running a local instance of Virtuoso you can instruct the code to point to *localhost*:

Example 1:

```
http://localhost:8890/CTGOV/describe/?uri=http://bio2rdf.org/clinicaltrials:NCT00175890
```

Example 2:

```
http://localhost:8890/RDFCUBE/describe/?uri=http://www.example.org/dc/dm/ds/obs01
```

The first http address above would display information about the clinical trial in a faceted browser view, similar to **Figure 7**, while the second would display data from an observation in an RDF data cube. The faceted browser view contains hyperlinks the user may follow to continue exploring the data, traversing the graph from one node value to the next.

About: A Placebo-controlled Study of Levetiracetam In Children (1mo to 4yrs of Age) With Partial Onset Seizures.

An Entity of Type : [http://bio2rdf.org/clinicaltrials\\_vocabulary:Resource](http://bio2rdf.org/clinicaltrials_vocabulary:Resource), within Data Space : [lod.openlinksw.com](http://lod.openlinksw.com) associated with source [document\(s\)](#)

Type: [clinicaltrials resource \[clinicaltrials\\_vocabulary:Resource\]](#) ▼ [New Facet based on Instances of this Class](#)

---

Attributes	Values
<a href="#">type</a>	<a href="#">Clinical Study [clinicaltrials_vocabulary:Clinical-Study]</a> <a href="#">clinicaltrials resource [clinicaltrials_vocabulary:Resource]</a>
<a href="#">label or name</a>	A Placebo-controlled Study of Levetiracetam In Children (1mo to 4yrs of Age) With Partial Onset Seizures. [clinicaltrials:NCT00175890]
<a href="#">Title</a>	A Placebo-controlled Study of Levetiracetam In Children (1mo to 4yrs of Age) With Partial Onset Seizures.
<a href="#">Identifier</a>	clinicaltrials:NCT00175890
<a href="#">in dataset</a>	<a href="http://bio2rdf.org/clinicaltrials_resource:bio2rdf.dataset.clinicaltrials.R3">http://bio2rdf.org/clinicaltrials_resource:bio2rdf.dataset.clinicaltrials.R3</a>
<a href="#">identifiers.org URI</a>	<a href="http://identifiers.org/clinicaltrials/NCT00175890">http://identifiers.org/clinicaltrials/NCT00175890</a>
<a href="#">Bio2RDF uri</a>	<a href="http://bio2rdf.org/clinicaltrials:NCT00175890">http://bio2rdf.org/clinicaltrials:NCT00175890</a>
<a href="#">Bio2RDF identifier</a>	NCT00175890
<a href="#">Bio2RDF namespace</a>	clinicaltrials
<a href="#">intervention brows...ervention-browse]</a>	<a href="#">Piracetam [clinicaltrials_resource:3352605dc14d7f0e09469973fc77bc61]</a> <a href="#">Etiracetam [clinicaltrials_resource:61e5650c36f3b7bbecd755cb46761545]</a>

Figure 7 Virtuoso faceted browser view of clinical trial information from ClinicalTrials.Gov, hosted on OpenLinkSW.

The network graph in **Figure 8** displays information about studies that contributed to two data pools used for submissions. Study phase and contribution are easily identified in the display, with drill-down to study information available by clicking on the study nodes. The legend is also interactive. A click on "Phase 1" opens a faceted browser view listing all Phase 1 studies. From there the user can follow the hyperlinked data to obtain details about each of Phase 1 study.

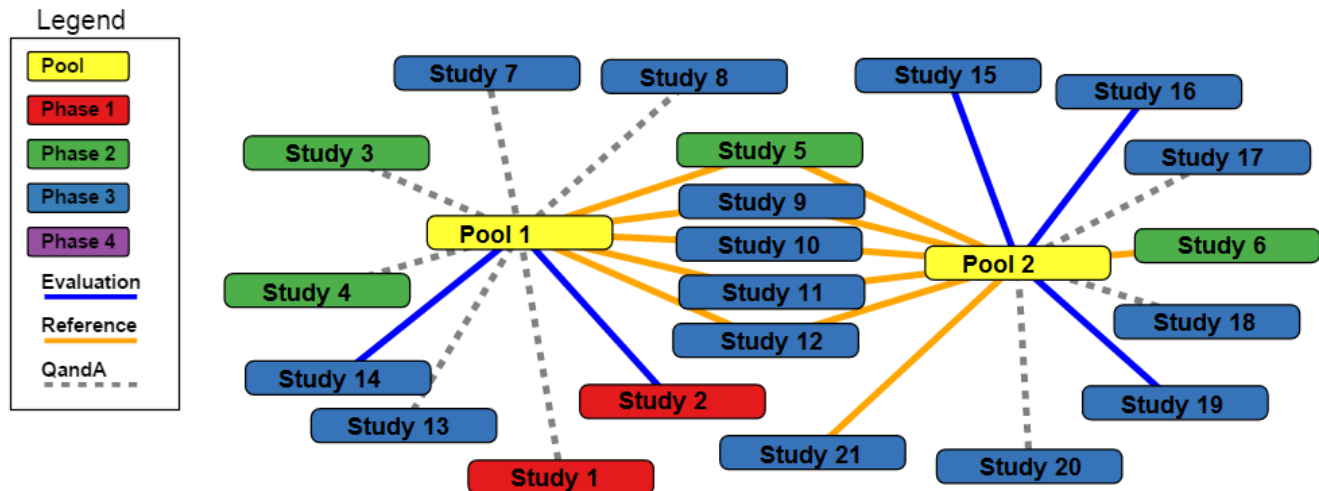


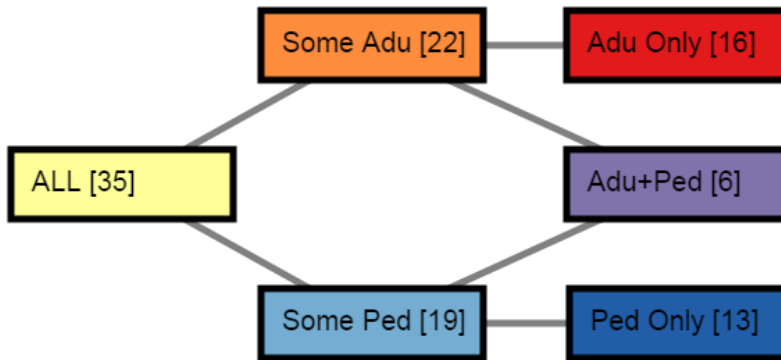
Figure 8 Force network graph showing studies that contribute to data pools.

Interactions in D3 web pages are not limited to following a hyperlink into faceted browser views. `d3sparql.js` (14) is a JavaScript library that enables execution of SPARQL queries, then transforms the result into JSON for display using D3. The example in **Figure 9** shows the result of clicking on the category "Some Ped" to display studies that contain pediatric patients. When the user holds down the Shift key and clicks a node (**Shift+Click**), a SPARQL query is executed on the triplestore and the result is returned to the table below the diagram. Once again, the user needs no knowledge of the underlying query language in order to display data from the triplestore. The hyperlinks in the "Obs" column lead the user back into the RDF triplestore where the data for that observation is displayed in the faceted browser view of Virtuoso, similar to **Figure 7**.

## PhUSE 2016

### Age Categories

<Shift+Click> to List Studies in each Age Category:



Click "Obs" for study details

Obs	Study	AgeGroup	MinAge	MaxAge	Phase	Control	Randomized
<a href="http://www.example.org/kmd/obs5">http://www.example.org/kmd/obs5</a>	Study 9	Pediatric	1 month	less than 4 years	Phase 3	Placebo	Yes
<a href="http://www.example.org/kmd/obs7">http://www.example.org/kmd/obs7</a>	Study 3	Pediatric	4 years	12 years	Phase 2	None	Yes
<a href="http://www.example.org/kmd/obs10">http://www.example.org/kmd/obs10</a>	Study 6	Pediatric	1 month	less than 4 years	Phase 2	Not Applicable	No
<a href="http://www.example.org/kmd/obs11">http://www.example.org/kmd/obs11</a>	Study 17	Adult+Pediatric	4 years	65 years	Phase 3	Placebo	Yes

Figure 9 Query an RDF Endpoint from a D3 web page using pre-defined categories. Table truncated for display.

A final example shows how interactive displays can be created directly in R, without the need for either intermediary data formats like JSON or coding the display using HTML and D3. A local triplestore graph of publications data was queried by executing SPARQL within R using the `rrdf` package (15). The graph was visualized using the `visNetwork` package (16), which brings the `vis.js` JavaScript library into the R environment. A mouseover of the publications shows the title, author, and publication date.

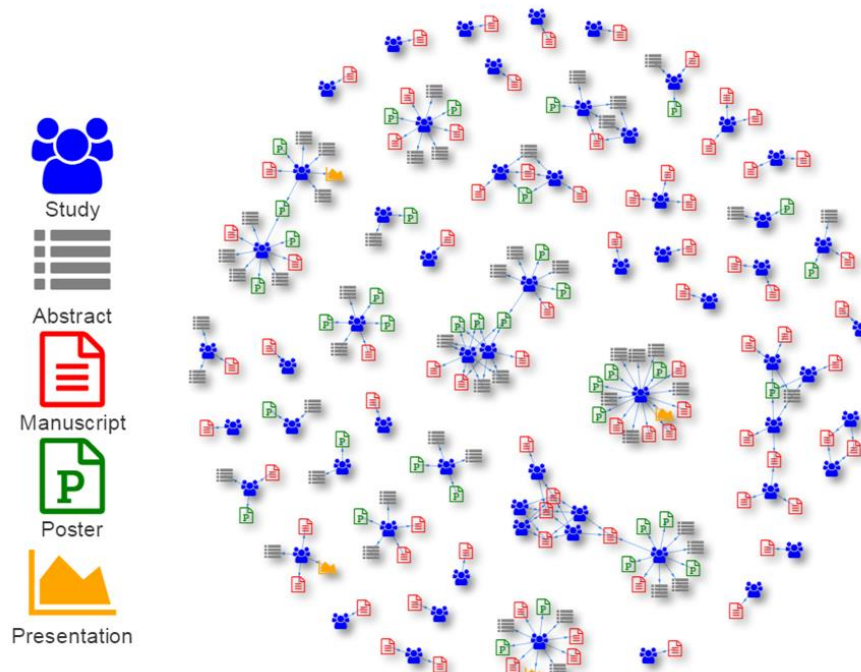


Figure 10 Study data linked to various types of publications

## CONCLUSION

The vast number of visualizations available within the D3 library means you are only limited by your imagination and choice of the plots and charts most appropriate for your data. The increasing availability of free and open source tools is shifting the landscape away from expensive, proprietary tools towards freely available and highly flexible toolsets. Many JavaScript

libraries have both free and commercial licensing, allowing you to recreate and improve upon visualizations previously only available in proprietary applications.

The future of clinical trials data is the graph. User-friendly, aesthetically pleasing, interactive visualizations will open up new ways to explore the web of clinical trials data.

### REFERENCES

1. **Cairo, Alberto.** *The Truthful Art. Data, charts, and maps for communication.* s.l. : New Riders, 2016.
2. PhUSE Semantic Technology Working Group Overview. [Online] PhUSE. [Cited: 07 27, 2016.] [http://www.phusewiki.org/wiki/index.php?title=Semantic\\_Technology](http://www.phusewiki.org/wiki/index.php?title=Semantic_Technology).
3. **CDISC.** CDISC Standards in RDF. *CDISC.* [Online] [Cited: 08 23, 2016.] <http://www.cdisc.org/rdf>.
4. *PhUSE Workshop: Semantics 101 for Pharma.* **Anderson, Marc and Williams, Tim.** Barcelona : PhUSE, 2016.
5. **Cygniak, R, Reynolds, D. and Tennison, J.** The RDF Data Cube Vocabulary. W3C Recommendation 16 January 2014. [Online] [Cited: 01 16, 2014.] <http://www.w3.org/TR/vocab-data-cube>.
6. **White, Nicole.** RNeo4j . *Github.* [Online] [Cited: 07 26, 2016.] <http://github.com/nicolewhite/RNeo4j>.
7. Graph Visualization for Neo4j. [Online] Neo4j. [Cited: 08 23, 2016.] <http://neo4j.com/developer/guide-data-visualization>.
8. Resource Description Framework. [Online] Wikipedia. [Cited: 08 23, 2016.] [http://en.wikipedia.org/wiki/Resource\\_Description\\_Framework](http://en.wikipedia.org/wiki/Resource_Description_Framework).
9. Linked Open Vocabularies (LOV). [Online] Open Knowledge Foundation. [Cited: 07 26, 2016.] <http://lov.okfn.org/dataset/lov/>.
10. Uniform Resource Identifier. *Wikipedia.* [Online] [https://en.wikipedia.org/wiki/Uniform\\_Resource\\_Identifier](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier).
11. D3 Gallery. *GitHub.* [Online] [Cited: 07 26, 2016.] <http://github.com/d3/d3/wiki/Gallery>.
12. *Interactive Data Visualization of Adverse Events Clinical Trial Data with the D3.js script library.* **Laurent, Johann and McDevitt, Hugh.** London : PhUSE, 2014.
13. *Create interactive web graphics out of your SAS or R datasets.* **Warnat, Patrick.** Vienna : PhUSE, 2015.
14. d3sparql.js. [Online] Github. [Cited: 07 27, 2016.] <http://github.com/ktym/d3sparql>.
15. **Willighagen, Egon.** R Package: rrdf. *Github.* [Online] [Cited: 07 27, 2016.] <http://github.com/egonw/rrdf>.
16. **Thieurmél, B.** Introduction to visNetwork. *CRAN.* [Online] [Cited: 07 27, 2016.] <http://cran.r-project.org/web/packages/visNetwork/vignettes/Introduction-to-visNetwork.html>.
17. **Panzarino, Onofrio.** *Learning Cypher.* s.l. : Packt Publishing, 2014.
18. PhUSE CSS Project: Analysis Results & Metadata. *PhUSE Wiki.* [Online] [Cited: 08 23, 2016.] <http://bit.ly/2bf7Pk0>.
19. **Hunger, Michael.** Charting Neo4j 3.0. *Neo4j Blog.* [Online] [Cited: 08 23, 2016.] <http://neo4j.com/blog/charting-neo4j-3.0>.
20. **Van Bruggen, Rik.** Learning Neo4j. [Online] 2016. [Cited: 08 23, 2016.] <http://neo4j.com/book-learning-neo4j/>.
21. **Robinson, Ian, Webber, Jim and Eifrem, Emil.** Graph Databases. [Online] [Cited: 08 23, 2016.] <http://neo4j.com/book-graph-databases/>.
22. Using Virtuoso Open-Source on Windows. [Online] [Cited: 08 23, 2016.] <http://bit.ly/2bRNx6h>.
23. **Fossati, Marco.** The complete tutorial for RDF data ingestion in Virtuoso. [Online] Confluence. [Cited: 08 23, 2016.] <http://bit.ly/2bL3Ok1>.
24. **W3C.** SPARQL 1.1 Query Language. W3C Recommendation 21 March 2013. [Online] W3C. [Cited: 08 23, 2016.] <http://www.w3.org/TR/sparql11-query/>.
25. **Houssain, Monsur and Hausenblas, Michael.** CORS on Virtuoso. [Online] [Cited: 08 23, 2016.] [http://enable-cors.org/server\\_virtuoso.html](http://enable-cors.org/server_virtuoso.html).
26. **Bostock, Mike.** D3 : Data Driven Documents. [Online] [Cited: 08 23, 2016.] <https://d3js.org/>.
27. The Big List of D3.js Resources. *Dashing D3.js.* [Online] [Cited: 08 23, 2016.] <http://www.dashingd3js.com/d3-resources/the-big-list-of-d3-resources>.
28. **W3C.** SPARQL 1.1 Federated Query. W3C Recommendation. [Online] W3.ORG. [Cited: 08 23, 2016.] <http://www.w3.org/TR/sparql11-federated-query>.

### ACKNOWLEDGEMENTS

The author is indebted to the following people, organizations, and companies:

- Mike Bostock for D3js
- OpenLink Software for Virtuoso
- Neo4j
- R-project, Egon Willighagen for rrdf, and Nicole White for RNeo4j.




## PhUSE 2016

This paper is largely based on free, open-source software and the efforts of volunteers in PhUSE working groups. Please support those who donate their time and expertise through your own collaboration, participation, and promotion of these activities.

### CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Tim Williams  
UCB BioSciences, Inc  
Raleigh, NC  
tim.williams@ucb.com (work)       @NovasTaylor  
NovasTaylor@gmail.com (personal)  
<https://www.linkedin.com/in/timpwilliams>

Brand and product names are trademarks of their respective companies.

---

<sup>1</sup> Virtuoso is a quadstore because it stores graph names in addition to the Subjects, Predicates, and Objects present in traditional triplestores. Triplestore is used throughout the paper to refer to the storage of RDF triples using Virtuoso or other RDF databases.