Paper TT09

SDTM domains by query - is it possible?

Johannes Ulander, S-Cubed, Copenhagen, Denmark

ABSTRACT

This paper will demonstrate what happens when turning your existing SDTM data into a linked data graph, where the structural boundaries of the data points in different domains are removed. It will look at our existing data from a different angle and show how it is possible to solve things that cannot be solved in a relational database. In the future, we will not be focused on trying to find the best model to store our data, but finding the best way of representing our knowledge.

INTRODUCTION

Within the pharmaceutical industry, there is a growing interest in trying to find new ways of working with our clinical data, as it seems that we always end up in the same situations. We have multiple mappings of one data point from one data structure to another, with a constant need to express the relationship between data points and its metadata. We are constantly searching for the perfect relational database model, so that the small subtle changes in our collected data can be handled without interfering with the overall model. Maintaining our standards and releasing new versions is very complex, and when we are ready to release a new version of a standard, it is probably already outdated compared with what needs to be collected.

Other industries, e.g. companies creating social network apps for your mobile phone, usually release new apps every other week. And more interestingly, you usually never notice any difference although something must have happened. These companies use linked data and graph databases to a very high degree, which has triggered my curiosity and need to understand if there is something that we in the pharmaceutical industry can learn from this technology.

The examples will use the CDISC SDTM and SDTMIG to represent tables in relational databases, as it is the most commonly used and understood tabular structure for using clinical data, but the discussions is at a general level on the differences between relational databases and linked data. Some of the example SDTM data comes from the CDISC *SDTM/ADaM Pilot Project* [1], but some small changes to the content have been made to make queries easier to read (e.g. USUBJID has been changed from “01-701-1015” to “1”.)

SDTM DOMAIN BY QUERY

The structural foundation for how we deal with clinical data in our day to day jobs is a relational database, and the tables we create in them are designed to answer specific questions, e.g. “Which subjects participated in the study?” and ”When did the subjects take their medication?”. So whenever we get new data to collect or something changes in the original question, we need to use all our experience and knowledge to figure out how it will work in our current model and decide whether it fits our existing standards or if there is a need to change it. Quite often we have to change the business rules (or even break them) to be able to squeeze new pieces of information into our existing model, although we sometimes find some good work arounds. But all the time we are surrounded by these boundaries of variables and tables in our relational database models.

Is it possible to remove the boundaries? Can linked data provide a benefit to our way of working?

SHORT INTRODUCTION TO LINKED DATA

In this paper, I will use the term linked data as something that is built up by triples of information. These triples can be used in graph databases or triple stores. It doesn’t really matter on a high level which one is used, the basic principles still apply for what is discussed in this paper.

The triples contain *nodes* (think of them as “things”), *relationships* and *properties*, and they either express:

a) The relationship between two nodes (i.e. “Links” them together), or

b) The property of a node (i.e. an attribute attached to a node and does not link further to another node)

Let’s start with the two simple tables below. The first table contains a relationship between an *Investigator* and a *site* at which he/she is working. The second table contains the properties of the *Investigator* (*Name*) and the *site* (*Site ID* and *City*.)

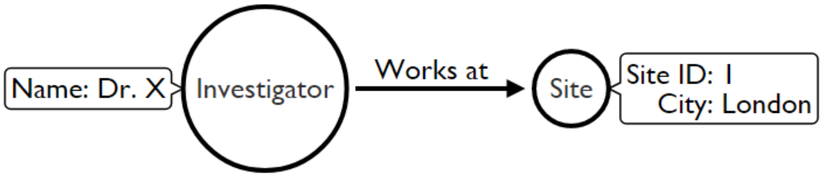
Node	***Relation***	***Node***
Investigator	Works at	Site

Example of a triple expressing the relationship between two nodes

Node	***Property***	***Value***
Investigator	Name	Dr. X
Site	Site ID	1
Site	City	London

Example of triples expressing properties of nodes [1]

The above triples can then be visualised like the below graph, using circles for *nodes*, arrows for *relationships* and call outs for the *properties*.

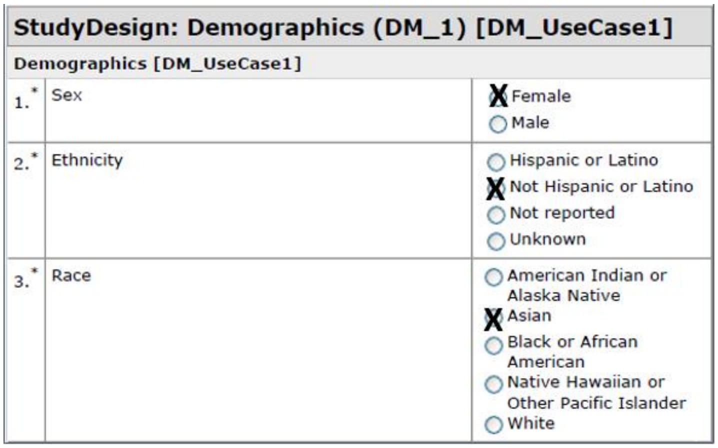


Example graph visualisation of the above tables

The above graph visualisation tells us that the investigator, Dr. X, works at a site in London (with Site ID = 1.) Please note that there is no difference in the content of the triples in the table and the visualised graph, they both contain exactly the same information. So for the rest of the paper, I will only show the graph representation of the triples. And when I write code examples, it will be in Cypher [2] because it has a very simple and short syntax.

SIMPLE START: DEMOGRAPHICS

To put this knowledge in practice, let's start with a simple demographics CRF example from the *CDASH Library of examples* [3]. The form is filled in with *Sex* = *Female*, *Ethnicity* = *Not Hispanic or Latino* and *Race* = *Asian*.



StudyDesign: Demographics (DM_1) [DM_UseCase1]

Demographics [DM_UseCase1]

1.*	Sex	☒ Female ○ Male
2.*	Ethnicity	○ Hispanic or Latino ☒ Not Hispanic or Latino ○ Not reported ○ Unknown
3.*	Race	○ American Indian or Alaska Native ☒ Asian ○ Black or African American ○ Native Hawaiian or Other Pacific Islander ○ White

Collected Demographic CRF

Everyone working with SDTM knows that the above form is stored in the SDTM DM domain like this[2]:

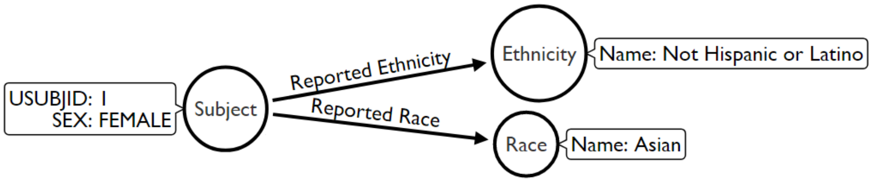
USUBJID	***SEX***	***RACE***	***ETHNIC***
1	F	ASIAN	NOT HISPANIC OR LATINO

Collected Demographic CRF data in SDTM DM domain

[1] There is nothing that prevents me from putting all data in the same table. I've chosen this approach because I think it is easier to understand the difference between a relation and a property. A relationship ends in another node (which can have properties of its own) and a property ends in a literal value which does not contain any further relationships (aka. edge.) In RDF you would typically express everything in a single table as <*Subject*>-<*Predicate*>-<*Object*> triples.

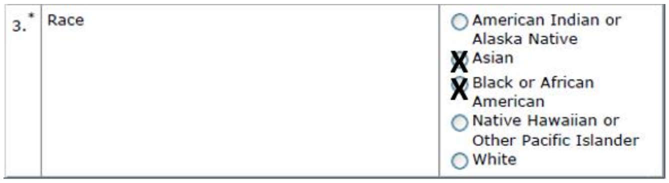
[2] Only showing USUBJID and the collected information from the CRF.

In a graph, we could represent the same collected data like this, using nodes for *Subject*, *Ethnicity* and *Race,* relationships for *Reported Ethnicity* and *Reported Race*, and properties for *USUBJID*, *Sex*, *Not Hispanic or Latino* and *Asian*.



Collected Demographic CRF data in a graph

This simple case works very well in SDTM, and there isn't a clear benefit of using a graph, as the DM domain in SDTM has been designed to answer this specific question and enable listing the demography of each subject participating in a trial. But in reality we know that we will often get multiple responses to one and the same question, e.g. *Race*.[3] So let's add another reported Race to the equation, and get the following CRF collection of race.



3.* Race

○ American Indian or Alaska Native
☒ Asian
☒ Black or African American
○ Native Hawaiian or Other Pacific Islander
○ White

Excerpt of collected Demographic CRF with multiple responses to reported Race question

In SDTM we have a pretty neat trick to handle this situation, by using supplemental qualifiers. Instead of having the reported races in the *Race* variable, we instead introduce a new term *MULTIPLE*, and add two records in the supplemental DM domain (SUPPDM) with both the reported races.

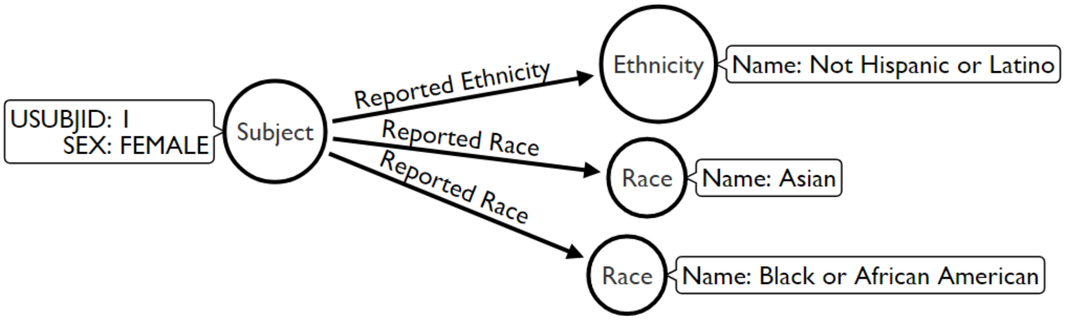
USUBJID	***SEX***	***RACE***	***ETHNIC***
1	F	MULTIPLE	NOT HISPANIC OR LATINO

+

QNAM	***QVAL***
RACE1	ASIAN
RACE2	BLACK OR AFRICAN AMERICAN

Collected Demographic CRF data in SDTM DM domain with multiple responses to Race question

With the addition of another response to race, we can clearly see that the way the data is collected is impacting how it is represented in a tabular structure, and that the representation in SDTM probably also gets further away from how this is represented in the data entry tool. It will also impact mapping tools and any programs that need to be adapted to how they should work depending on how data is collected. Let's add the same knowledge to our graph, i.e. that the subject has a reported race of *Black or African American* as well as *Asian*.



Collected Demographic CRF data in a graph with multiple responses to Race question

All we've done is adding this new triple of knowledge: *Subject* (with USUBJID=1) has a *Reported Race* of *Black or African American*. There is no change to any structure and we didn't need to change or introduce any new terminology from the originally collected values (like we did with *MULTIPLE* in SDTM above.)

[3] And FDA even encourages companies to collect multiple responses for race. [4].

In fact, both graphs of demographics above are generated with the same query:

Cypher Query in Neo4j [2]

```
MATCH (s:Subject {USUBJID:"1"})-[rel]->(node)
WHERE node:Race OR node:Ethnicity
```

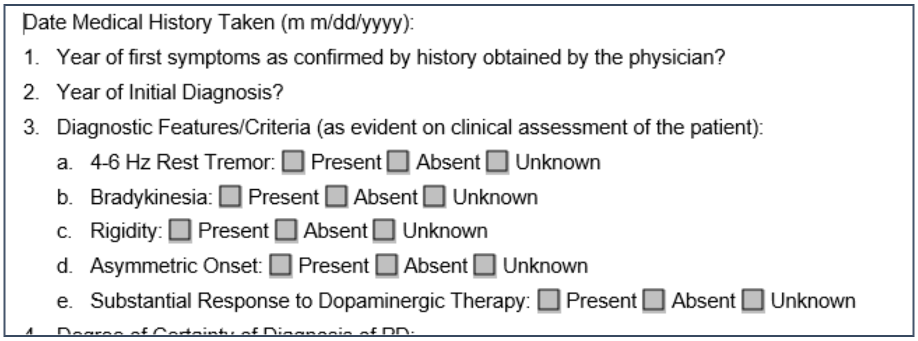
Explanatory pseudo code

Find *s* (a node of type *Subject* with *USUBJID=1*) and the relationships (*rel*) to any *node* of type Race or Ethnicity

When the query is executed with only one reported race for the subject, it results in the first visualisation. And when executed after the second race has been added, it is the second graph visualisation.

ADDING COMPLEXITY: MEDICAL HISTORY

The first attempt with demographics seems to have gone well, and it seems like the graph was able to handle the multiple responses to the same question without problem. So let's try with something more complex, Medical History. The below CRF excerpt is from the NINDS Common Data Elements for Parkinson's Disease [5], which looks like a very typical pre-specified medical history CRF. It contains two different dates related to the same disease (*Year of first symptoms* and *Year of Initial Diagnosis*) as well as diagnostic features and/or criteria that are known to be related to diagnosing the disease.

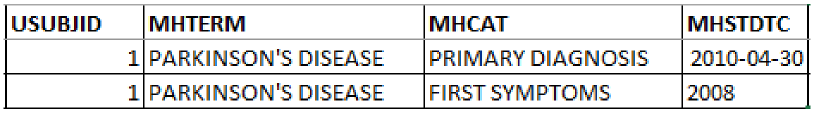


Date Medical History Taken (m m/dd/yyyy):

1. Year of first symptoms as confirmed by history obtained by the physician?
2. Year of Initial Diagnosis?
3. Diagnostic Features/Criteria (as evident on clinical assessment of the patient):
   a. 4-6 Hz Rest Tremor: □ Present □ Absent □ Unknown
   b. Bradykinesia: □ Present □ Absent □ Unknown
   c. Rigidity: □ Present □ Absent □ Unknown
   d. Asymmetric Onset: □ Present □ Absent □ Unknown
   e. Substantial Response to Dopaminergic Therapy: □ Present □ Absent □ Unknown
4. Degree of Certainty of Diagnosis of PD:

Excerpt from NINDS CDE example CRF for Medical History

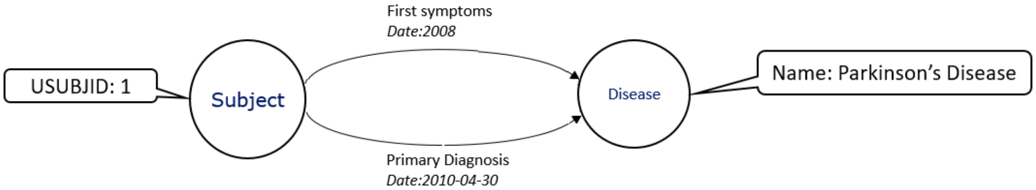
As we will get both the date of first symptoms and year of initial diagnosis and it only exists one start date in the SDTMIG Medical History domain, we will need to make a design decision on how to represent this information. (E.g. make two rows with differing category, add a supplemental qualifier or use Findings About. All of them work, but it has a great impact on mapping and output programs.) This is what it could look like with two rows and a different categories:



USUBJID	MHTERM	MHCAT	MHSTDTC
1	PARKINSON'S DISEASE	PRIMARY DIAGNOSIS	2010-04-30
1	PARKINSON'S DISEASE	FIRST SYMPTOMS	2008

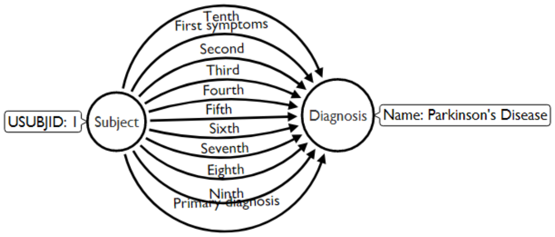
Collected Medical History CRF data in *SDTM* MH domain with two *dates*

Adding the same knowledge to the graph, that the subject had the first symptoms in 2008 and was diagnosed with Parkinson's disease on 2010-04-30, could look like this:



Collected Medical History CRF data in SDTM MH domain with two dates

Just like in the multiple race example above, there are no structural issues. I just added my new knowledge and queried the linked data to find the answer to my question. And again, I don't need to adapt my question if there is one, two or even 10 relationships between the subject and the disease. But I will need to work on the visualisation, if I want to show them all at the same time.

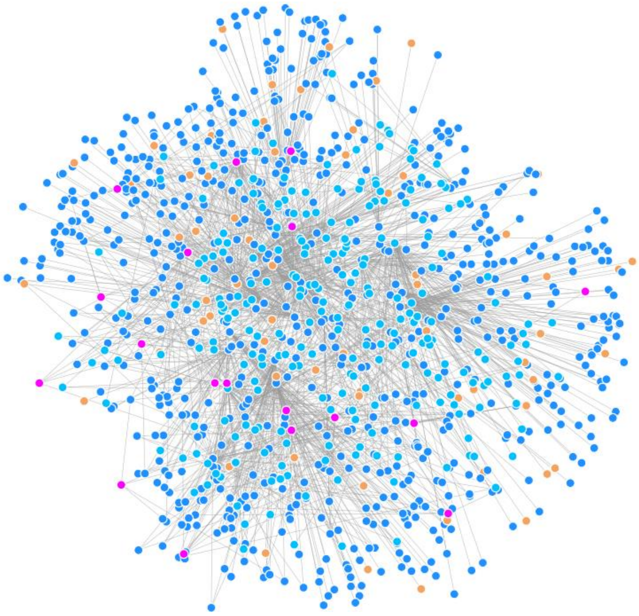


Messy graph showing 10 relationships between a subject and Parkinson's Disease

Maybe you have already noticed, but I have not once indicated to which SDTM domain my linked data belongs. The reason is quite simple: there are no structures or tables in the linked data, only the information and knowledge I have stored.[4]
So where is my SDTM domain in linked data?

WHAT IS AN SDTM DOMAIN IN LINKED DATA?

As there are no tables in the linked data (at least not as we know them from relational databases), queries are fundamental to view the data as a human. We can search the whole database directly, to show the full extent of the linked data universe. The below picture from the Glandon MDR shows all the metadata from CDISC SDTM and SDTMIG as a linked data graph.



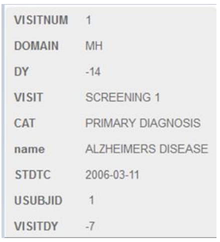
The linked data galaxy of CDISC SDTM and SDTMIG

And by using the properties of the linked data, we can find what we need, e.g. the nodes which have the property stating that it belongs to Medical History for a subject.

Cypher Query in Neo4j
`MATCH (n) WHERE n.DOMAIN = "MH" AND n.USUBJID = "1"`
Explanatory pseudo code
Find nodes *n* which have the property DOMAIN = "MH" and USUBJID = "1"

The result of the query is visualized in the below graph (on the left) and in a tabular view (to the right) showing all the properties of each individual node (N.B only the selected nodes' properties are shown).

[4] Some of the graph database tools, e.g. Neo4j, even takes care of the storage for you, so I have no idea on how the data itself is represented on the hard drive. You just create knowledge and query the database.



VISITNUM	1
DOMAIN	MH
DY	-14
VISIT	SCREENING 1
CAT	PRIMARY DIAGNOSIS
name	ALZHEIMERS DISEASE
STDTC	2006-03-11
USUBJID	1
VISITDY	-7

***Left image**: "SDTM Domain MH" as linked data* ***Right image**: The properties of the selected node to the left*

And if we want to create the SDTM MH domain that we are familiar to, we can change the query to include the study node to get STUDYID and return it as a table:

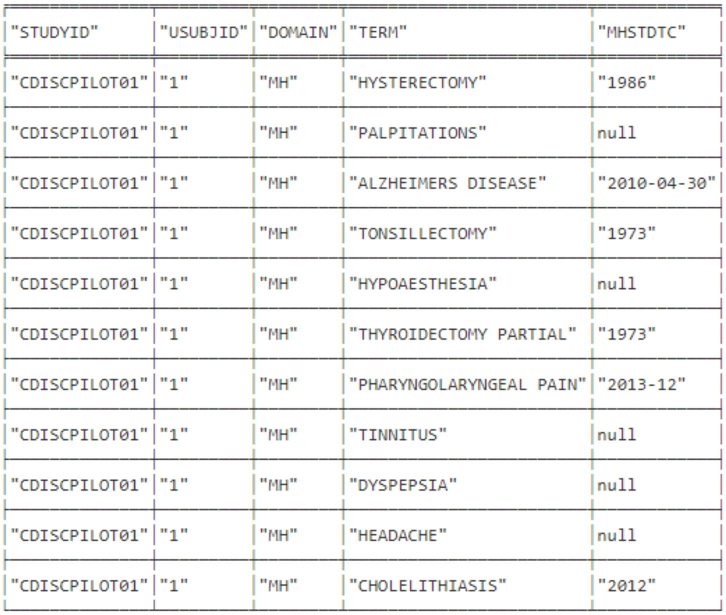
Cypher Query

```
MATCH (s:Subject)-[]->(n), (s:Subject)-[]-(study:Study)
WHERE s.USUBJID = '1' AND n.DOMAIN = 'MH'
RETURN study.name as STUDYID,
       s.USUBJID  as USUBJID,
       n.DOMAIN   as DOMAIN,
       n.name     as MHTERM,
       n.STDTC    as MHSTDTC
```

Explanatory pseudo code

Find subject node *s* with *USUBJID* = "1" and the related nodes *n* which have the property *DOMAIN* = "MH", also find the study node *study*. Return *study*.*name* as STUDYID, *s*.*USUBJID* as USUBJID, *n*.*DOMAIN* as DOMAIN, *n*.*name* as MHTERM and *n*.*STDTC* as MHSTDTC

Which will generate the text output below:



```
╒═════════════╤═════════╤════════╤═════════════════════════╤════════════╕
│"STUDYID"    │"USUBJID"│"DOMAIN"│"TERM"                   │"MHSTDTC"   │
╞═════════════╪═════════╪════════╪═════════════════════════╪════════════╡
│"CDISCPILOT01"│"1"     │"MH"    │"HYSTERECTOMY"           │"1986"      │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"PALPITATIONS"           │null        │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"ALZHEIMERS DISEASE"     │"2010-04-30"│
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"TONSILLECTOMY"          │"1973"      │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"HYPOAESTHESIA"          │null        │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"THYROIDECTOMY PARTIAL"  │"1973"      │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"PHARYNGOLARYNGEAL PAIN" │"2013-12"   │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"TINNITUS"               │null        │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"DYSPEPSIA"              │null        │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"HEADACHE"               │null        │
├─────────────┼─────────┼────────┼─────────────────────────┼────────────┤
│"CDISCPILOT01"│"1"     │"MH"    │"CHOLELITHIASIS"         │"2012"      │
└─────────────┴─────────┴────────┴─────────────────────────┴────────────┘
```

SDTM MH Domain for subject 1 created by a query on linked data

Using tabular format for the output will of course get us back to the problems discussed earlier, e.g. representing multiple races and dates in SDTM, whereas exporting it to other linked data formats is easier as we don't have the tabular restrictions.

THE LINKED DATA UNIVERSE

In the linked data universe there is actually not a distinction between a collected value and a derived value, and you will be able to directly express the relationship between collected and derived values. If we want to calculate the average Systolic Blood pressure for subject “1”, we can do it with the below query.

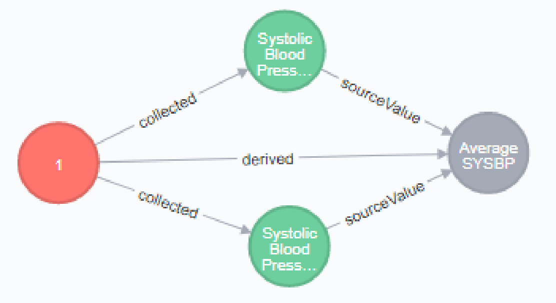
Cypher Query

```
MATCH (sbpNode:StandardisedResult {name:"Systolic Blood Pressure"})<-[]-(subjectNode:Subject
{USUBJID:"1"})
CREATE (averageNode:Average {name:"Average SYSBP"})
WITH COLLECT(sbpNode) as sbpNodes, averageNode, AVG(sbpNode.STRESN) as theAverageValue,
subjectNode
FOREACH (node in sbpNodes | CREATE (node)-[source:sourceValue]-(averageNode) )
SET averageNode.value = theAverageValue
CREATE (averageNode)<-[:derivedValue]-(subjectNode)
RETURN sbpNodes, averageNode, subjectNode
```

Explanatory pseudo code

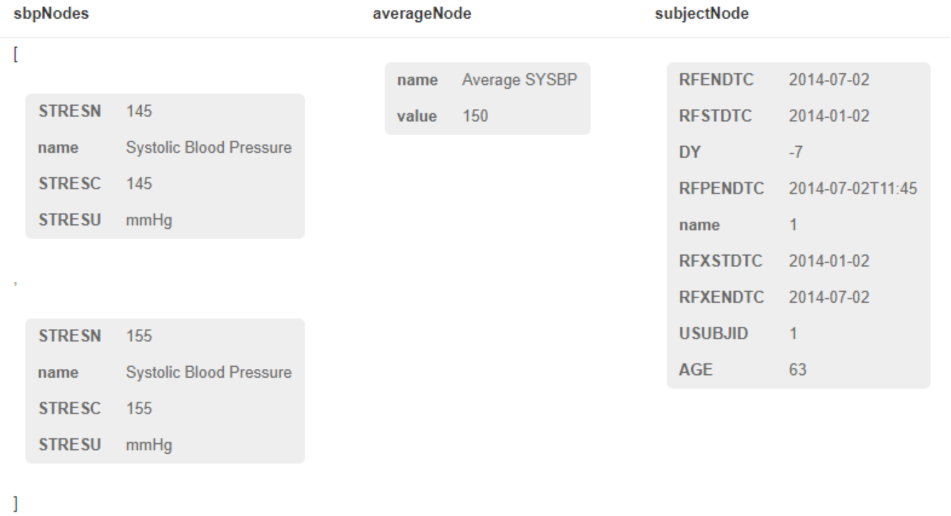
Find all Systolic Blood Pressure nodes (*sbpNode*) for subject 1 (*subjectNode*).
Create a node *averageNode* to store the average value.
For every *sbpNode* found, create a relationship *sourceValue* from the *averageNode* to the *sbpNode*.
Set *averageNode.value* to the average Systolic Blood Pressure.
Create a relationship *derivedValue* between the *subjectNode* and the *averageNode*.

The result from the above code is this:



Average Systolic Blood Pressure derived from collected values

And in textual representation, showing the properties of the nodes:



sbpNodes	averageNode	subjectNode
[STRESN 145 name Systolic Blood Pressure STRESC 145 STRESU mmHg , STRESN 155 name Systolic Blood Pressure STRESC 155 STRESU mmHg]	name Average SYSBP value 150	RFENDTC 2014-07-02 RFSTDTC 2014-01-02 DY -7 RFPENDTC 2014-07-02T11:45 name 1 RFXSTDTC 2014-01-02 RFXENDTC 2014-07-02 USUBJID 1 AGE 63

Average Systolic Blood Pressure derived from collected values in textual representation

As you can see in the above textual representation, we have the properties for the two Systolic Blood Pressure nodes to the left, the properties for the subject node to the right and the newly created average node in the middle. The average node has a property value = 150, which is calculated from the STRESN properties of the Systolic Blood Pressure nodes on the left.

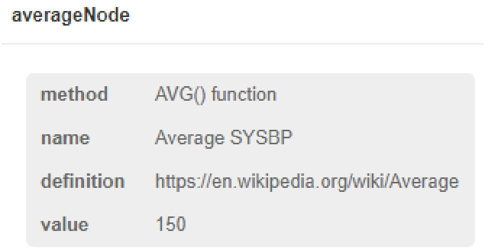
In the graph representation you also see the relationships, which show the traceability between our newly derived average node to the source nodes which are used as input for the calculation.

And if we tweak the SET statement in the previous code like this:

Cypher Query

```
   SET averageNode.value = theAverageValue,
       averageNode.definition = "https://en.wikipedia.org/wiki/Average",
       averageNode.method = "AVG() function"
```

I have also added the definition (from Wikipedia) and method of calculation to the average node:



averageNode

method	AVG() function
name	Average SYSBP
definition	https://en.wikipedia.org/wiki/Average
value	150

Average Systolic Blood Pressure node updated with definition and method of calculation

By adding the method of calculation and a link to the definition, we have also removed the boundary of separating data from metadata, and the data is getting closer to truly being a part of the linked data universe.

CONCLUSION

By the examples in this paper I have showed that linked data has a lot to offer to the pharmaceutical industry.

- It is possible to create an SDTM domain by query
- Linked data can solve many of the problems that we are dealing with today in relational databases
- Linked data excels at handling complex information, and we can focus more on how to store our knowledge, rather than discussing which variables and structures best fits future needs

I hope this can inspire more people to embark on a linked data journey, because clinical data is not getting less complex. We need to take control over our data and be able to define more precise standards, which is necessary for being part of the linked data universe.

REFERENCES

1. Updated Version of Pilot Submission Package (2013)
   https://www.cdisc.org/sdtmadam-pilot-project
2. https://neo4j.com/developer/cypher/ (retrieved 2017-08-25)
3. CDASH_USER GUIDE V1-1.1 LIBRARY OF EXAMPLE CRFS
   https://www.cdisc.org/standards/foundational/cdash
4. Collection of Race and Ethnicity Data in Clinical Trials
   https://www.fda.gov/downloads/regulatoryinformation/guidances/ucm126396.pdf
5. Medical History of Parkinson's Disease Version 1.1 Date 08/10/2012
   https://www.commondataelements.ninds.nih.gov/Doc/PD/F0750_Medical_History_of_Parkinsons_Disease.docx

ACKNOWLEDGMENTS

Thanks to AJ De Montjoie and Dave Iberson-Hurst for support, discussions and inspiration.

CONTACT INFORMATION

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

Johannes Ulander
S-Cubed ApS,
Lille Strandstraede 20C, 5
Copenhagen / 1254
Web: http://www.s-cubed-global.com/