PhUSE EU Connect 2018

Paper PP05

New Approach to Graph Databases

Anna Berg, Capish, Malmö, Sweden
Henrik Drews, Capish, Malmö, Sweden
Catharina Dahlbo, Capish, Malmö, Sweden

ABSTRACT

Graph databases have, during the past few years, become more common when it comes to managing large amounts of data. The data is primarily modelled with focus on the relationships but has been considered not to be ideal for analytical processing. In order to take advantage of well-structured data, Capish suggests a new approach to support an information model, presented as a graph database with the focus on nodes instead of relationships. The approach allows the creation of several indices, which can be used to efficiently search using indirect relationships.

This paper discusses the differences between graph databases associated to the "classic" approach and the new approach Capish suggests, together with requirements and consequences using one or another.

INTRODUCTION

The R&D performance of pharmaceutical companies is dependent on a good understanding of all relevant data, not only in isolation, but in its totality. Data also needs to be evaluated in context against a backdrop of ever changing and increasing data volumes [1]. This sets requirements on *flexibility* and *scalability* of the solution, without having the need to remodel the whole data repository when encountering new data.

A prerequisite for an improved evaluation and analysis of data is *semantic interoperability*. This can be achieved by a consistent approach to harmonization and integration of data, through standardization and implementation of common terminologies [2]. To fulfil the value proposition that standardized data offers, data also needs to be well curated and meet the quality requirements of the end user.

Furthermore, data needs to be modelled in an unbiased way, which maintains the *context* in which the data was collected. The context carries the meaning or the purpose of the data, which enables users to properly communicate, analyze, draw conclusions and make decisions [3]. Technology systems that do not provide context require the human brain to bring context to the data, with a risk of introducing potential human error and limits the number of people being able to properly analyze and evaluate the data. Addition of context also enables pooling of data in a way that makes the user aware of potential differences in the meaning of the data.

Another challenge is to provide a data repository that enables *instant access* to all available data, without the constant need for time consuming data extraction, transformation and load procedures, every time end users want to explore a new hypothesis.

NoSQL (not only SQL) solutions and especially *graph databases* have become more common when it comes to managing large amounts of data. Graph databases, as the name implies, represent data in a graph format, i.e. a network of information. The main difference to traditional relational databases is that each data point (node, vertex) is *explicitly connected* to other data points, via relationships (edges).

As a group, graph databases provide a clear benefit over relational databases for the following reasons [7]:

- **The conceptual data model translates directly into the graph database model**. Most data and how the data is connected can be described as a graph. In reality this is how our brain works, making it rather easy to understand. This also means there are no constraints on individual joins, as with relational databases.
- **Graph databases are flexible and easily expandable**. New nodes and relationships can be added without disturbing the existing graph. As a consequence, the data model can evolve with our understanding of the

respective domain. This allows to exploit or evaluate the data before the final data model has emerged. Repurposing of data is also much easier as the perspective and focus can be adjusted.

- **Metadata can be stored directly as part of the data**. In graph databases, it is easy to add a context to the data or to store metadata as part of the model, making the graph self-describing. This ensures that data and the data about the data are never separated, maintaining the value of the data and making it possible to analyze the available data, not just now, but also in the future.
- **Data can be evaluated in different contexts**. Data can be explored starting from any node within the network of data points. In combination with enough context and metadata to explain the data, this gives the user a possibility to explore new hypotheses and create new insights.

CLASSIC GRAPH DATABASES

All graph databases have one thing in common; they consist of nodes connected by edges (relations). However, how the nodes and their relations are modelled and stored depends very much on the purpose of the database. Two classic graph database techniques are RDF (Resource Description Framework) graph and property graph databases.

RDF GRAPH DATABASES

An RDF graph database models and stores the data in an atomic way, with no internal structure of its components. This means that the nodes and relations only consist of a unique label, where the nodes often are primitive data types like string, integer, date, etc.

RDF is based on so-called “triples” or simple statements comprising of a subject, predicate and object. The predicate in this statement represents a relationship between the subject and object. While the subject always is a URI, objects can either be another URI or a literal. A database built up by such triples is called a “triple store”, see Figure 1.

Subject	*Predicate*	Object
URI_1	*hasName*	Mary
URI_1	*knows*	URI_2
URI_2	*hasName*	John

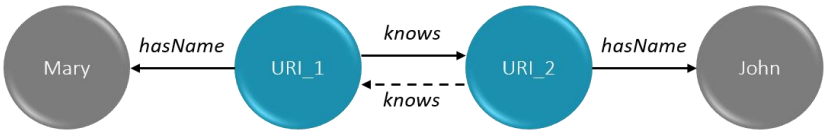

Figure 1. Example of three triples in RDF, represented both as a triple store and as a graph. Note that the dashed relation can only be inferred if inference rules have been set up.

RDF is built upon W3C’s linked data technology, which has been developed for the semantic web. The idea behind the semantic web is to publish data in a structured format with well-defined semantics, to enable computers to use it in meaningful ways [4]. For this reason, the items in an RDF database are identified by URIs (Uniform Resource Identifiers), as a way of “namespacing” the identifiers. Therefore, use cases of RDF are often to exchange and publish data.

Semantics can be defined as the meaning and understanding of data. One interpretation of this could be the *vocabulary-based semantics*, which is frequent in RDF graph databases. Because of the identification with URIs, common vocabularies can be defined and used in a data sharing situation. This covers most of the usage of RDF today [5]. Another interpretation of semantics is *inference-based semantics*, which is rule-based and often achieved by using an ontology, e.g. OWL (Web Ontology Language) for RDF. Inference is used to infer facts, do reasoning or check the consistency of the dataset. Inference has to be made explicit in a rule, often by specifying which other relations exist in the data if a specific relation is present. An example could be a rule defining the relation called ‘knows’ as symmetric, with the conclusion that if Mary knows John, John also knows Mary, see example in Figure 1.

Since RDF graph databases are triple stores, traversals become quite cumbersome as the database increases. Therefore, there are performance issues with RDF databases, that do not exist in the same way with property graph databases, as discussed later. These issues can be solved by introducing indices, but at the cost of flexibility.

The popularity of RDF can be attributed to a facilitated exchange of data, providing:

- A possibility to do ”friend of a friend” traversals
- A possibility to do inference
- Facilitated data sharing between many parties, due to an easy handling of many different vocabularies and namespaces

PROPERTY GRAPH DATABASES

A property graph database has a possibility to hold an internal structure of its components, achieved by adding properties, types and labels to the nodes and relations. The components of this internal structure can also be called attributes, fields, etc., which are comprised of an arbitrary number of key value pairs [5]. Since the nodes can have labels, they might also be used to identify them as a particular *type* of information unit.

The possibility to have information as properties instead of separate nodes makes property graphs much more compact than RDF graphs. An example of a property graph is given in Figure 2.

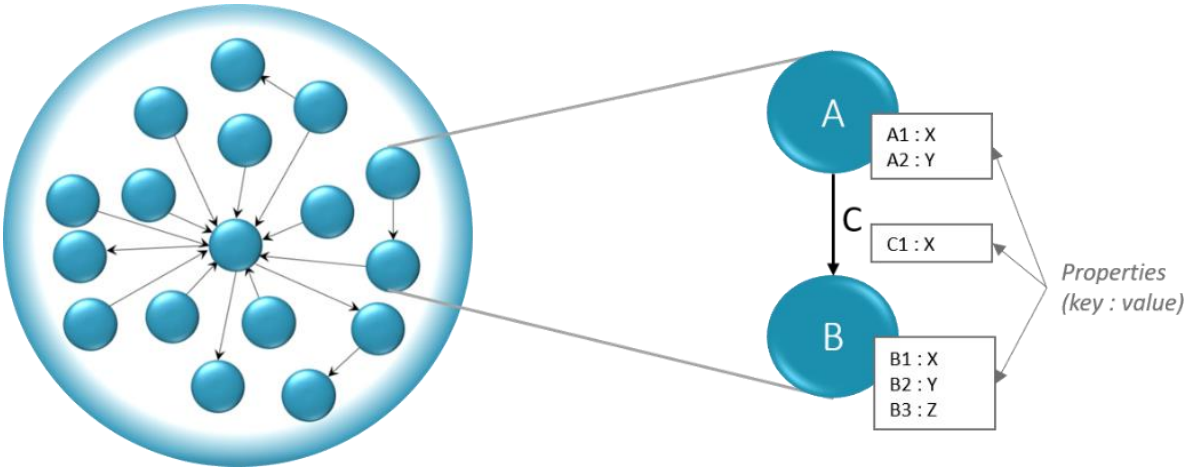

Figure 2. Simple representation of two nodes in a labelled and directed property graph.

How the nodes and their relationships are modelled in a property graph depends very much on the purpose of the database. While nodes and relationships are equal partners in a graph database, models can choose to focus on either nodes or relationships. The model to choose depends on the questions you want to ask and whether the aim is to analyze the graph as a whole for analytical purposes or whether local transactions are the primary concern. The modelling of property graphs is further discussed in the section “New Approach to Graph Databases – Modelling Principles”.

Property graph databases are so called “native graph databases”, as the database store the data as a graph internally. Examples of property graphs are the Capish® database and the Neo4j database.

In contrast to the use cases of RDF graphs to exchange and publish data, property graph databases are primarily developed to:

- Achieve an efficient storage
- Allowing for fast querying
- Allowing for fast traversals

NEW APPROACH TO GRAPH DATABASES

While much emphasis in the clinical world has been on RDF or triple stores, property databases offer some benefits that will help evaluating clinical data or data in the life sciences, as discussed above. Capish has advanced the classic property graph database and developed a new approach to graph databases by:

1. Utilizing an **ontology** as an information model and to describe the data
2. Establishing certain **modelling principles**
3. Implementing several **indices**

1. ONTOLOGY

An ontology is a *formal representation* of a knowledge domain, describing its entities (e.g. events and processes) as well as the relationships connecting these entities. Ontologies are often used to add *semantics* to the data in a graph. Semantics in this context refer to the meaning and understanding of the data. No graph database can in itself provide semantics to its components. Instead, semantics have to be provided explicitly, often by the use of an ontology that describes the components of the graph.

The Capish ontology consists of an *information model,* defining how the nodes and relations of the graph should be modelled, as well as a *terminology,* describing and defining all the components of the data.

The information model is based on a network of information units, so-called “Holons”, and their relations. In contrast to relational databases, data that is tightly connected are stored together in the Holon. Examples of such data could be variables belonging to an individual event, process or entity, e.g. a measurement result stored together with its unit, the date it was measured and its original unit (if changed). Each Holon should be self-described; therefore, it is carrying all the information necessary to understand and evaluate this particular event, process or entity. An example of a couple of Holon definitions in the Capish ontology and their representation in the database (instances) can be seen in figure 3.

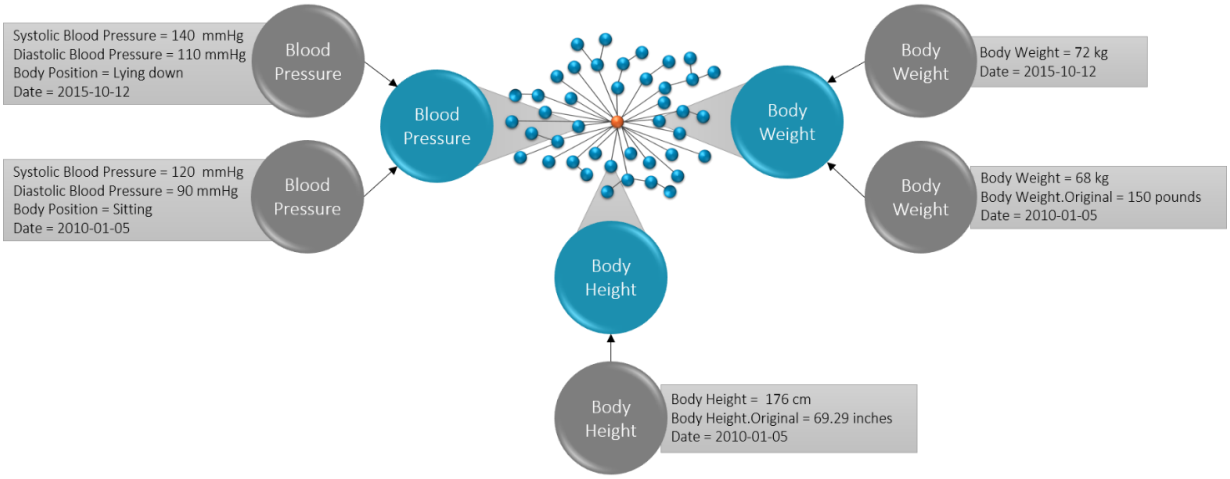

Figure 3. Simple representation of five Holons (gray nodes) and their properties (so-called fields and values) corresponding to three Holon definitions (blue nodes) in the Capish ontology.

The terminology part of the Capish ontology defines the components of the graph, for example:

- Naming and synonyms
- Knowledge domain (medicine, law, geography, etc.)
- Data type and content scale (time, string, decimal, ordinal, narrative, etc.)
- Time dependency (no time dependency, point in time or interval)
- Physical dimension (allowed units)
- Standard value lists

The terminology also holds ways to represent certain kind of data in a way making it easy for any user or software to find and use it. Examples of such data includes links to documents, reference ranges, original values, notes and content validity.

The main purpose of the Capish ontology is to describe and unify data, which has the benefit of allowing for curation of data from disparate data sources into a single, well defined graph. This makes queries across all available data feasible. The ontology also provides enhanced data quality, since the data is tightly curated and modelled.

2. MODELLING PRINCIPLES

The establishment of modelling principles is very useful to guarantee *consistency* of the model, both in different contexts and over time. A consistent way of modelling is of great importance when integrating data from different domains and when there is a need to extend an existing graph with new data.

Capish has developed several principles to ensure modelling consistency. One principle that becomes important when comparing graph databases is that the relation in the Capish approach is not allowed to add any additional meaning to the data. This principle has the consequence that there can never be more than one relation between any two Holons. If there are more than one, each relation must mean something different.

The rationale behind this modelling principle is that the properties of a relation are not allowed to change how the data should be interpreted. In other words, the name of a relation is simply used as a nice way to read out the relation and must not be required to interpret the relation. What is most important is that the interpretation of a Holon shall not be dependent on the route leading to it, since each Holon has to be self-described. Information needed to interpret the connection is instead given in so-called “relation Holons”, which tie two other Holons together in a specific context, or during a specific point in time. This modelling principle is a prerequisite for the indexing process, described below.

See figure 4 for a simplified example of how this principle makes the same data modelled differently in the Capish graph and the classic property graph. In the Capish graph, dates and other related data are modelled as properties of the relation Holons, whereas in the classic property graph, they are properties of the relations.

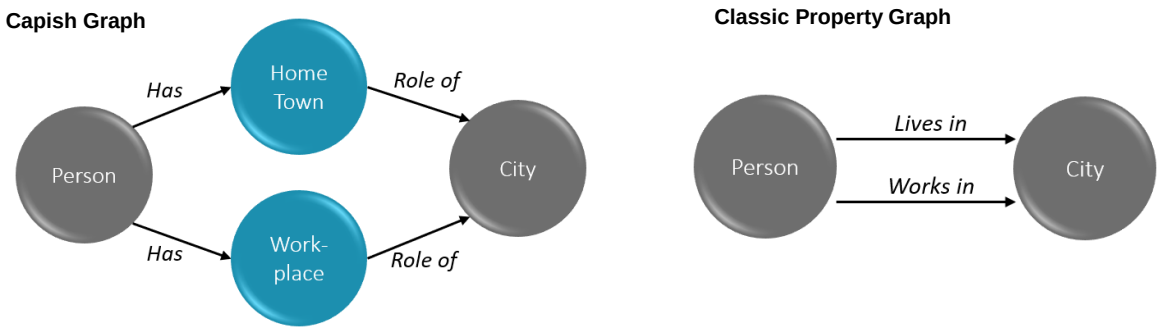

Figure 4. An example of two ways of modelling where a person lives and works during different times of his/her life. When adding data to these models, the number of nodes and relations increases, since one and the same person can move and change work many times during his/her life. In the Capish graph, dates and other related data are properties of the relation Holons, whereas in the classic property graph, they are properties of the relations. Gray nodes represent ordinary nodes (Holons) and blue nodes represent the relation Holons in the Capish graph.

In the classic property graph database approach, there is a gray area whether a specific data point should end up as a node or as a relation. The fact that all data points end up in the nodes (the Holons) in the Capish ontology-based information model makes it easier to create and maintain a consistent way of modelling.

3. INDEXING

The new approach also depends on the creation of several indices [6], represented by:

- **Identity index** – is used to identify a Holon in the database
- **Relation index** – is used to identify all Holons related to a specific Holon (e.g. a patient) in a single step
- **Holon definition identity index** – is used to identify the Holon definition (type of Holon), thereby finding which Holons of a specific Holon definition that are present in the database

The indexing procedure facilitates analytical processing by avoiding the costly node traversals. They also provide a more intuitive data querying process, since the user does not need to know the chain of relations. In particular, the relation index provides a new concept and can be used to efficiently search using indirect relationships, or search for specific criteria and retrieve a broader result set.

In addition, the relation index is used to set up "reflection points". A reflection point is a Holon for which the user might be interested in finding all related data. Having a reflection point in, for instance, the 'Patient' Holon can be used to identify all data connected to a particular patient, in one single step. The possibility to add a reflection point enables users to analyze the database from different perspectives and easily create data subsets (cohorts).

The implementation of the indices allows to answer complex questions in a fast and intuitive way. Examples of such questions are:

A. Find all blood pressure measurement that are considered high and that are connected to patients that were given a dose of the drug A.

B. Find the adverse events that have been reported for patients with elevated liver values and who received drug A.

COMPARISON

To be able to do a fair comparison between the different types of graph databases, more than one use case have to be evaluated. Some of the most common use cases are listed and discussed below. For a summary of use cases and the graph database that is suggested as the best approach for each case, see table I.

SEMANTICS

The atomic modelling of RDF in combination with the URIs provide a good base for semantics. Combined with an ontology to define inference rules, RDF graph databases become particularly advantageous for semantics use cases, compared to the property graph databases.

DATA & METADATA SHARING

One of the original use cases for RDF graphs was in data sharing situations, solved by the easy handling of many different vocabularies and namespaces, as discussed above. This also makes RDF advantageous for storing common metadata standards and terminologies, e.g. the CDISC SDTM format.

GRAPH TRAVERSAL

RDF databases have been useful when it comes to showing connected data and traversing the database from one node to the next. The classic example is the "friend of a friend" traversal, from one person node to the next. The property graph databases further improve the traversal times by drastically decreasing the number of nodes to pass in the graph. Since the purpose of the Capish approach is not to optimize the traversal times, but rather to enable an easy-to-use platform for exploring data, traversing the graph is not a use case.

TRANSACTIONS

RDF databases were never meant to be used for operational and transactional use cases. They should instead be used in mostly additive, typically slow-changing datasets, holding relatively static information about the world. For example, capitals of countries, names of diseases, etc. Conversely, the classic property graph databases are useful in highly dynamic scenarios and with more transactional use cases [5], whereas the Capish database is not optimized for such use cases.

DATA STORAGE

When it comes to storage of large amounts of data, e.g. from a clinical trial, the property graph databases are better suited. This is mainly because of the faster and easier ways to access and query the data.

ANALYTICAL PROCESSING

RDF graph databases are not considered to be ideal for analytical processing, as the traversal process would result in performance issues. Property graph databases have an improved traversal performance, but they still need to traverse the graph to find a specific node. By adding indices, traversals are avoided, and the desired nodes can be found faster, thereby increasing the analytical performance. Combined with an ontology to define and model data, as done in the Capish approach, drastically improves analytical processing [7].

QUERYING & FILTERING

Because of the atomic decomposition of the data in RDF, the patterns are typically rather long when performing queries. Again, the performance is increased by using a property graph, since it is more compact. The implementation of various indices further improves the way data can be queried, making the data querying process faster and more intuitive for the end user.

Table I. Summary of use cases and the graph database that is suggested as the best approach for each case.

Use Case	Best Approach
Semantics	RDF Graph
Data & Metadata Sharing	RDF Graph
Graph Traversal	Property Graph
Transactions	Property Graph
Data Storage	Property Graph
Analytical Processing	Capish Property Graph
Queries & Filtering	Capish Property Graph

EXAMPLE

As discussed, the same data can be modelled very differently – either as a relational database, an RDF database or any kind of property graph database. To give an example of this, clinical trial data from two CDISC SDTM domains is modelled both as an RDF graph and according to the Capish approach to a property graph, see figure 5.

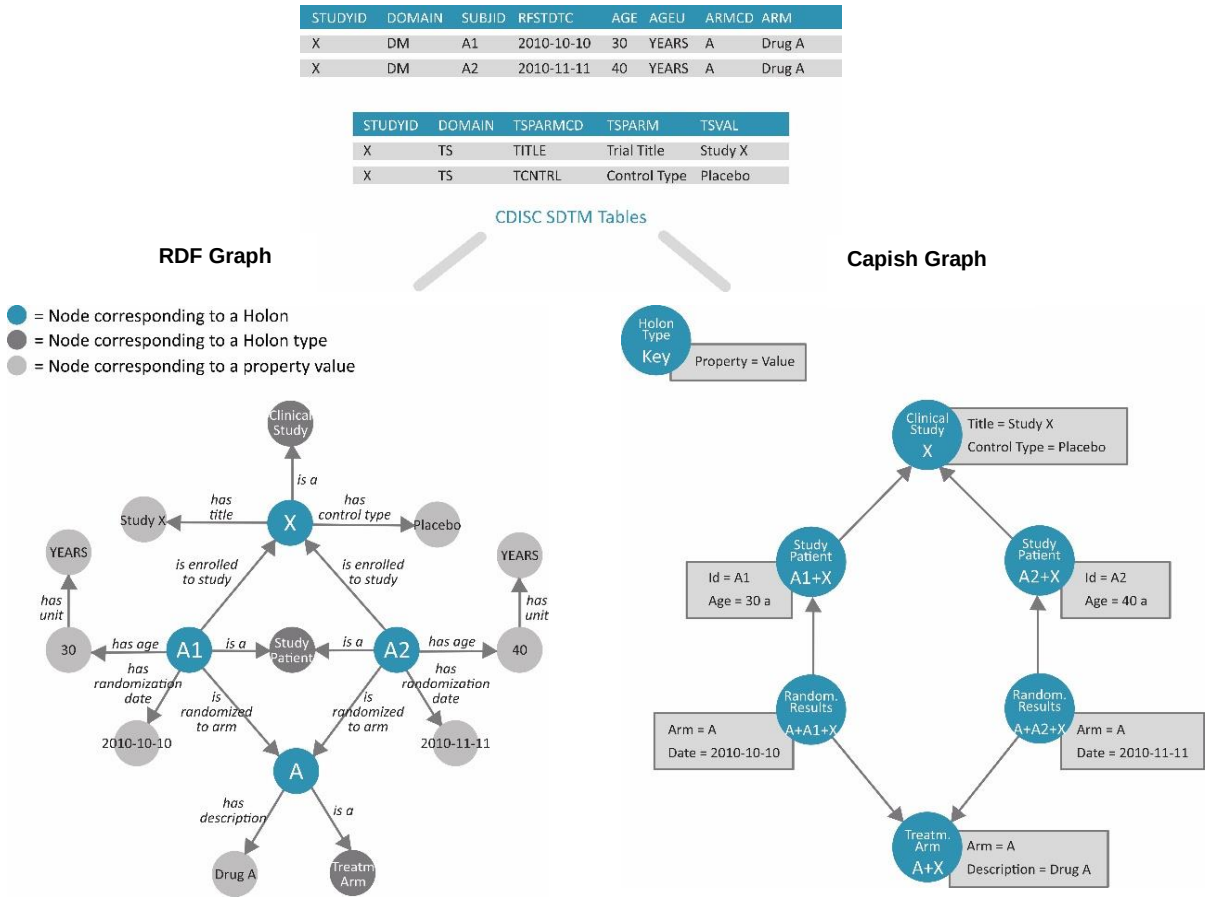

Figure 5. Comparison between a relational database (CDISC SDTM) on the top, RDF to the left and the Capish database to the right. The data used in this comparison is clinical trial data from the demography (DM) and trial summary (TS) domains in the CDISC SDTM standard.

This example clearly illustrates the difference between the atomic way of modelling in RDF and the more compact model of a property graph. It also illustrates how the properties of a property graph can be used to hold both identifiers or keys (e.g. A1+X) and actual values of variables.

CONCLUSION

The differences between graph databases boil down to how the graphs are stored and modelled. Which one to use depends on the actual use case.

Even though much emphasis in the clinical world has been on classical graph databases approaches, they are not considered to be ideal for analytical processing. By using a property graph together with an ontology and an indexing process, many of the performance issues associated with the classic approach could be eliminated and graph databases can be converted into powerful analytical systems.

REFERENCES

1. P. Tormay, "*Big Data in Pharmaceutical R&D: Creating a Sustainable R&D Engine".* Pharmaceutical Medicine, 2015. 29: p. 87-92.
2. A.E. Thessen, and D.J. Patterson, "*Data issues in the life sciences".* ZooKeys, 2011: p. 15-51.
3. I. Fleming and J. Leveille, *"TT10 – The Technology of Context"*, presented at the annual PhUSE conference, 2015.
4. T. Berners-Lee, J. Hendler and O. Lassila, *"The Semantic Web"*. Scientific American, 2001. 5: p. 29-37.
5. J. Barrasa, *"RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?"*, Neo4j blog.
6. C. Dahlbo, A. Workneh and H. Drews, *"PP04-Unique Technology for the Future"*, presented at the PhUSE EU Connect, 2018.
7. P. Tormay and H. Drews, *"TT06 - Reflect on your data"*, presented at the annual PhUSE conference, 2016.

CONTACT INFORMATION

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

Anna Berg
Capish Nordic AB
Carlsgatan 3
SE-211 20 Malmö, Sweden

+46 40 10 88 80
anna.berg@capish.com

www.capish.com