PhUSE 2016

TT06

Reflect on your data

Dr Peter Tormay, Capish Nordic AB, Malmö, Sweden

Henrik Drews, Capish Nordic AB, Malmö Sweden

Abstract

In order to take full advantage of well curated, integrated and harmonized data it needs to be paired with an appropriate information platform that enables end users to access the data without barriers. If this is not the case, all efforts of data integration will simply result in another data silo that doesn't live up to its full potential.

Working with an object oriented information platform that stores data in an unbiased way and creates a web of information could be a solution. Here individual information blocks can be linked through meaningful relationships and data can be traversed without knowledge of the underlying database model enabling end users to view and evaluate data from many different vantage points dependent on the questions asked. Queries and filters do not need to be translated into a logical and physical model of the database but rather work in a more natural way.

Introduction

It is no secret that improving productivity in the R&D arena is pivotal for the industry's future [1]. On a high level this is dependent on the number of drug candidates under development and the probability of success on the one hand and the cost and the time it takes to develop an individual drug on the other hand. Moreover, in today's world it is not just a question of efficiency, i.e. the time it takes to develop a drug but also of effectiveness, i.e. the value a drug brings to the table in relation to the current standard of care.

On a more granular level, drug innovation depends on a better understanding of disease mechanisms, the identification of potential drug targets and how these targets can be hit. Improving the probability of success requires a good understanding of whether a drug candidate operates as expected, whether it has the required efficacy and safety profile as early as possible [2]. In short R&D performance 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 [3].

The importance of data has been realised and a lot of focus has been put on harmonising and integrating data through standardisation, the implementation of common terminologies and vocabularies, thus improving semantic interoperability [4]. Although a consistent approach is a prerequisite for the evaluation and analysis of data it is not sufficient. To fulfil the value proposition that standardised data offers, it needs to be supported by a well curated, optimized data repository that meets the requirements of the end user.

The main challenge is to provide a structured framework 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. Data needs to be stored in an unbiased way, which maintains the context in which the data was collected, while enabling the pooling and grouping of data in a way that supports the evaluation and analysis of the data in different ways. As more and more data becomes available and more and more data sources need to be integrated it is important that any infrastructure is scalable without the need to remodel the whole data repository.

Big Data, data types and scalability

No discussion about data integration and analytics can avoid the concept of Big Data. We are all familiar with the 3 V's (which sometimes have been extended to 4, 5 or more V's). From a clinical development perspective, the key challenges are associated with the different type of data (variety), the quality of data (veracity) as well as data variability.

With the explosion of data and the notion that this data could become relevant in the future, the question around scalability of existing information systems becomes more and more relevant. Scalability is affected by two dimensions [5]:

- **Data volume:** Can the system cope with the increase in data volumes, i.e. handle more data of the same type efficiently.
- **Data complexity:** Can the system cope with increasing data complexity, i.e. can the system effectively handle the proliferation of data types? In order to effectively handle complex data, the system not only needs to be able to add these data types into the mix but also needs to be able to connect these different data types with each other.

The discussion about data complexity also includes the discussion about structured and unstructured data. On the one side you have data that can fit neatly into tables and thus can easily be "structured". On the other side you have unstructured data that doesn't fit neatly into tables and thus is more difficult to model. Having said that, unstructured data is not really unstructured it only means that it follows a pattern that doesn't lend itself to tables or is flexible and even evolves as more data becomes available. The classic example of unstructured data is free text. Nobody will deny though that text is highly structured. Text is governed by grammar. In addition, the meaning in text often depends on context, something that is lost when you try to structure it using the classic table approach.

Data volume, data complexity and whether data is structured or unstructured and how these factors impact on scalability has profound implications on how to store data effectively and efficiently for easy access, data retrieval and contextual evaluation. Continued innovation is necessary to find new information platforms and systems that are suited for the new paradigm of connected data.

Relational databases – the Swiss army knife of data management

Relational databases have been the Swiss army knife of the industry for decades. One of the key strength of relational databases is the ability to manage transactions. In many ways the data we collect during a clinical trial fit into this category: a new patient – a new row in the patient table, a new adverse event – a new row in the adverse event table, new laboratory data go in the laboratory data and so on. Each of these tables in themselves are rather easy to construct.

The challenge arises when the data in these tables need to be connected for further analysis. Despite the "relational" in the name, relational databases are not very good when it comes to connecting these tables. Relational simply means that keys – primary and foreign keys can be created in each table to connect them. This is where the challenge starts. First of all, these "relationships" in themselves have no meaning and do not provide any information on how and why these tables are connected. As the number and complexity of the tables increases it is not just a simple matter of connecting individual tables. Consider a situation where several entries in one table need to be connected to several entries in another table – so called many to many relationships. In order to achieve this, separate join tables which are completely unrelated to the actual data need to be implemented.

Another concept that comes into play is normalisation. In order to avoid redundant information, meaning information that would be repeated on several lines and might be prone to inconsistency is split out in new tables with so called one to many relationships. There are other factors to consider such as circular references as well as multiple key fields (i.e. several columns providing a composite key) that complicate the matter.

In order to accommodate all these constraints a defined, rigid database schema needs to be created before any data can actually be entered into the system (a simple example is shown in Figure 5 on page 7). This is a complex undertaking as the original conceptual model – the way we perceive the data and how it is connected - needs to be translated into a logical model - the way the database can understand these connections – which in turn needs to be converted into a physical model that can actually be programmed. With increasing data complexity this becomes increasingly difficult.

In order to improve performance data has to be rearranged in the context of the specific analysis. As a result, multiple data warehouses need to be created for different analysis. As a consequence, with every change of perspective the data needs to be remodelled, a costly undertaking.

Another approach to improve performance, is OLAP (online analytical processing) where the data is converted into multi-dimensional data cubes that can be easily sliced and diced. Again all these cubes are predefined in the context of a given analysis which requires the creation of new cubes in the case the perspective changes.

Data lakes

The idea behind data lakes is simple. Rather than trying to squeeze data into a predefined schema upfront, data is stored in its raw format until it is needed. Only when the data is required for analysis is it transformed. The benefits of such an approach are clear [6].

- Rather than spending a lot of time and effort integrating every bit of data into a model, data is only accessed when it is needed. This makes sense, when the relevance of that data might not be known or even worse make the overall integration project fail as it proves impossible to create the appropriate data model.
- Since you maintain access to the original raw data throughout the lifecycle of the data it is possible to evaluate the data in different contexts at a later stage.

The rise of Hadoop with its distributed filesystem, where any data can be stored effortlessly and cheaply make this approach a compelling proposition.

At the same time great care needs to be taken to keep track of all the data that goes in the data lake. In order to be able to make use of this data at a later stage everything about this data needs to be recorded – it needs to be clear in which context the original data was collected, what each data point means and how the data is stored (i.e. the original data schema). This requires a sophisticated metadata repository. Without it you run the risk of misinterpreting the data at a later stage or being unable to use it at all. In the worst case scenario, you might not even remember that you have the data in the first place.

It could also be argued that the data lake merely postpones the challenges encountered with relational database systems. At some point the data needs to be modelled which could still be an issue when data complexity is high.

This is well recognised and the Hadoop ecosystem is constantly adding new applications into the mix to deal with these challenges. This makes the simple idea of modelling your data only when you need it a rather complex undertaking. Gartner speaks of the fallacy of data lakes in this context [7]. They fully recognise the potential benefits but point out that data lakes are not aimed at the end user but rather the data scientists who need to make sense of the data and still provide smaller well curated DataMart to end users for further evaluation.

Graph databases

Graph databases as the name says represents data in a graph format, i.e. network of information. The main difference is that each data point (node, vertex) is explicitly connected to other data points via relationships (edges)[8].

As a group, graph databases provide a clear benefit over relational databases when it comes to connected or linked data for the following reasons:

- **The conceptual data model translates directly into the graph database model**. There is no need for transformations. 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. 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 means that data can already be exploited or evaluated before the final data model has emerged. Repurposing of the data is also much easier as the perspective and focus can be adjusted.
- **Metadata can be stored directly as part of the data**. Graph databases make it easy to store any data about the data as part of the model. This makes it much easier for users to understand the model and work with it.
- **Data can be evaluated in different contexts**. Data can be explored starting from any node within the network of data points.

Graph databases come essentially in two flavours. RDF and property 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 below.

RDF or triple stores

RDF is based on so called triple stores or simple statements comprising of a subject, predicate and object [9]. The predicate in this statement establishing a relationship between subject and object. While subjects are always so called uniform resource identifiers (URI), objects can either be another URI or a scalar value. The popularity of RDF can be attributed to the following benefits.

- RDF is built upon W3C's linked data technology resulting in a simple and uniform data model
- Due to the implementation of namespaces different graphs can be easily combined.
- SPARQL as a standardised query language
- Inferencing

While triple stores effectively create a network of linked data once the data is processed, the underlying structure is not a graph but rather a table optimised for horizontal scalability.

One further drawback of triple stores is the lack of properties. Although properties can be implemented as triples themselves they clearly differ from real relationships. Just consider the two statements:

- Alan – has age – 38 (property)
- Alan – participates – clinical study (relationship).

Property graph databases

Property graph databases are so called "native graph databases" as the database represents the data as a graph internally. In addition, nodes and (also) relationships can have properties (also called attributes, fields etc.) which are comprised of an arbitrary number of key value pairs [8].

In this respect each node is a small collection of information that is connected to other pieces of information through explicit relationships. Nodes also can have labels identifying them as a particular type of information unit. An example is given in Figure 1.

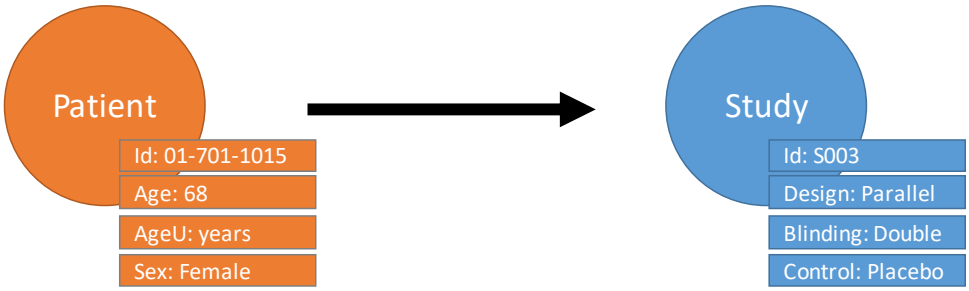


Figure 1 Simple representation of two nodes labelled as Patient and Study and their relationship

How these pieces of information and their relationships are modelled depends very much on the purpose of the database. The underlying data model and the actual database need to go hand in hand. While nodes and relationships are equal partners in a graph database, data models can focus on nodes or relationships. The difference is shown in Figure 2.

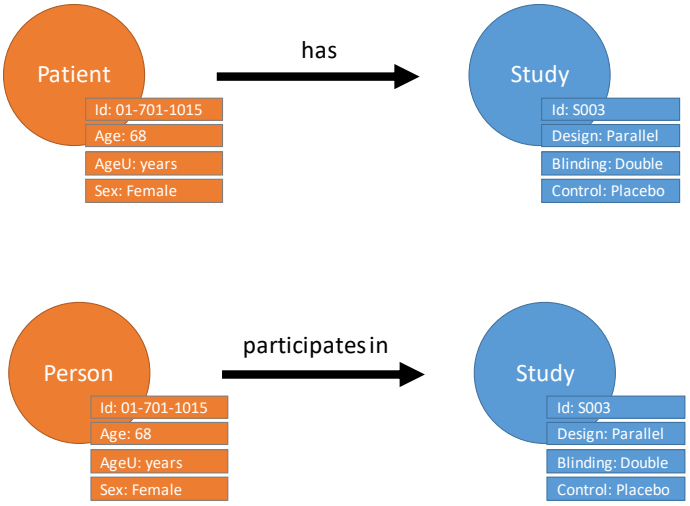


Figure 2 Different approaches to data modelling

In the top representation the relationship purely works as a connector between the two nodes while all the meaning is effectively represented in the node labels. Even without the relationship label it is clear that the person referenced on the left as a patient is in fact a patient in the study on the right. In the bottom example the relationship provides the context and only by knowing the relationship of "participated in", it becomes evident that the person on the left is a patient in the study.

The data model to choose depends on the questions you want to ask and whether the aim is to analyse the graph as a whole for analytical purposes or whether local transactions are the primary concern.

Graph databases and analytical processing

In the past graph databases have been considered to be transactional in nature and particular useful when it comes to showing connected data and traversing the database from one node to the next. The classic example is the friends and friends of friends traversal from one person node to the next. In this context data is primarily modelled with a focus on the relationships [10]. Therefore, graph databases were not considered to be ideal for analytical processing as the traversal process would result in performance issues.

We suggest a new approach where by modelling data with a focus on the node types, graph databases can be converted into powerful analytical systems eliminating the performance issues associated with the "classic" approach.

This new approach allows for the creation of several indices, representing:

a) individual nodes (identity index)

b) node types (node type identity index)

c) existence of indirect relationships (relation index)

d) property values (property index)

In particular the relation index provides a new concept and can be used to efficiently search using indirect relationships or search for specific criteria (via property indices) and retrieve a broader result set.

The implementation of the relation index in conjunction with the other indices therefore allows to answer complex questions such as:

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) What adverse events have been reported for patients with elevated liver values (and who received drug A)

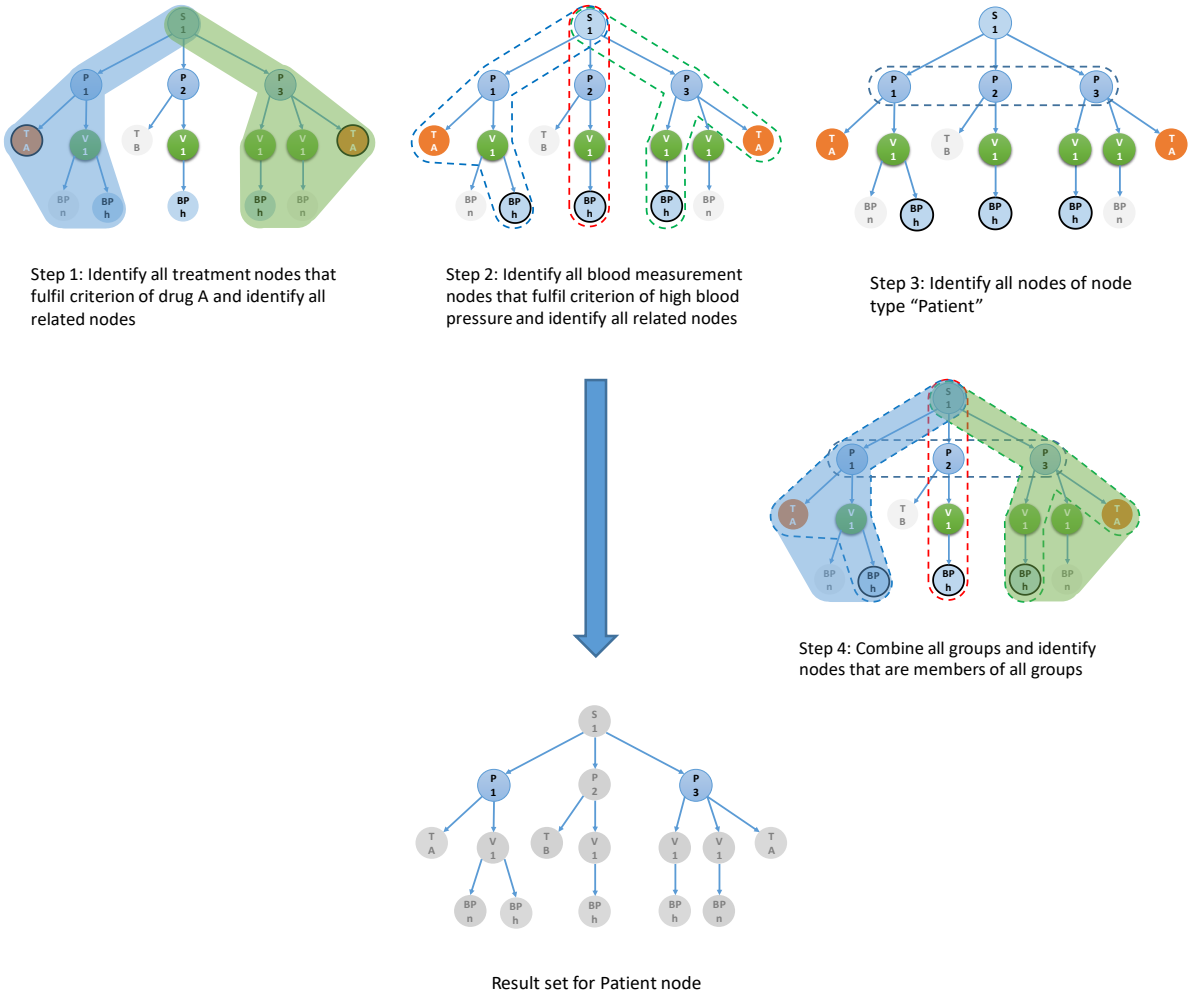


Figure 3 Identification of patients with high blood pressure receiving Drug A

In the case of the first example question, the various indices would be used as follows. In the first step all treatment nodes fulfilling the criterion of drug A are identified and using the relation index all related nodes returned. In the second step the same process is done for high blood pressure measurements. In the third step all nodes of the type patient would be returned. Combining the results of all steps, all patients who are present in all of the subsets are returned. In the particular case Patients 1 and 3 fulfil all criteria and this is shown in Figure 3.

In addition, the relation index is used to identify all data connected to, for instance, a particular patient in a single step. In this way data subsets (cohorts) can easily be created.

In the current example, all data for Patients 1 and 3 can be accessed looking at the relation index and returning all related nodes. This is shown in Figure 4.

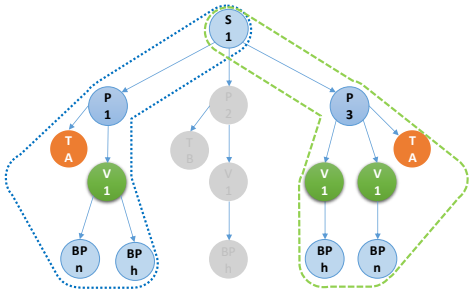


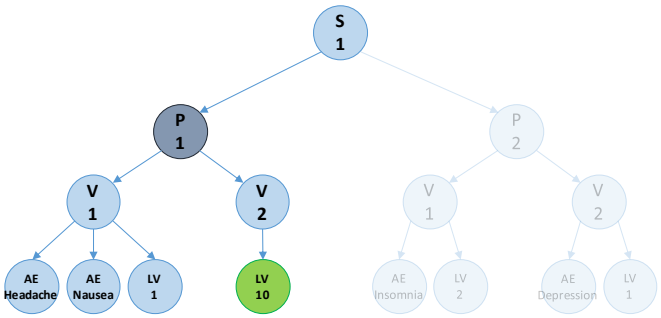
Figure 4 Creation of cohorts by identifying all related nodes

Graph database and intuitive querying

The implementation of a graph database in conjunction with various indices also improves the way data can be queried, making the data querying process faster and more intuitive for the end user.

A comparison of how a query would look like for the second example question mentioned above (see page 5) is given in Figure 5. This is shown in comparison to a relational database implementation of the same information and the resulting SQL query.

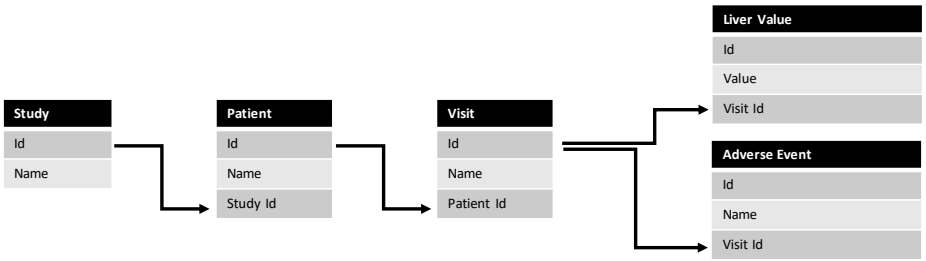
A)



Select data:
Type of Node: "Patient"
With (Type of Node: "Liver Value", Property: "Value > 5")

Fetch:
Type of Node:"Adverse Event", property:"Name"

B)



Study	
Id	Name
1	1

Patient		
Id	Name	Study Id
1	1	1
2	2	1

Visit		
Id	Name	Patient Id
1	1	1
2	2	1
3	1	2
4	2	2

Liver Value		
Id	Value	Visit Id
1	1	1
2	10	2
3	2	3
4	1	4

Adverse Event		
Id	Name	Visit Id
1	Headache	1
2	Nausea	1
3	Insomnia	3
4	Depression	4

```
SELECT ae.Name
FROM Adverse Event ae
INNER JOIN Visit v ON v.Id = ae.Visit Id
INNER JOIN Patient p ON p.id = v.Patient Id
WHERE p.Id IN (
  SELECT p2.Id
  FROM Patient p2
  INNER JOIN Visit v2 ON v2.Patient Id = p2.Id
  INNER JOIN Liver Value lv ON lv.Visit Id = v2.Id
  WHERE lv.Value > 5
)
```

Figure 5 intuitive querying in a graph database (A) compared to a relational database (B)

In case of the graph representation, the construction of the query only requires knowledge of the different nodes, in this case - adverse event, patient and liver value. Moreover, the overall query process is a simple two step process, where the data is first reduced to the relevant patients using the property indices and relation index and subsequently filtered for the adverse events under investigation.

In order to construct the equivalent query for the relational database representation intricate knowledge of all tables and keys connecting individual tables, is required. As the number of tables and thus the number of joins increases, the time it takes to join the different tables increases exponentially resulting in increased response times for the query.

Figure 5 also provides a good example of how the data model and the actual data representation are identical for the graph database whereas data tables and data model require independent representations for the relational database.

Conclusion

This paper explores a way to exploit the advantages of a graph oriented data repository. Combining this with a standardised approach to data integrity and harmonization in a well-defined ontology provides an easy to use approach for end users to merge, rearrange and put data into meaningful context for the analysis of patterns and correlations.

In this paper we describe a new approach to graph databases where individual nodes can be easily linked to all other related nodes whether directly or indirectly and the available data can be evaluated without intricate knowledge of the underlying database design. In fact, this approach makes data access easier and querying and filtering more straight forward. This logic supports creation of cohorts and subpopulations for evaluation of data in different contexts and from different perspectives, so that end users can explore data for example from a patient perspective or study perspective.

Query performance is greatly improved as related nodes can be accessed in a single step avoiding costly join operations needed for relational database implementations or node traversal as in other graph database implementations.

This new approach to data analytics provides more actionable and relevant answers to many of our traditional questions in a timely manner.

References

1. Paul, S.M., et al., *How to improve R&D productivity: the pharmaceutical industry's grand challenge.* Nature Reviews Drug Discovery, 2010. **9**: p. 203-214.
2. David, E., T. Tramontin, and R. Zemmel, *Pharmaceutical R&D: the road to positive returns.* Nature Reviews Drug Discovery, 2009. **8**: p. 609-610.
3. Tormay, P., *Big Data in Pharmaceutical R&D: Creating a Sustainable R&D Engine.* Pharmaceutical Medicine, 2015. **29**: p. 87-92.
4. Thessen, A.E. and D.J. Patterson, *Data issues in the life sciences.* ZooKeys, 2011: p. 15-51.
5. Moniruzzaman, A.B.M. and S.A. Hossain, *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison.* arXiv preprint arXiv:1307.0191, 2013.
6. Stein, B. and A. Morrison. *Rethinking integration: Data Lakes and the promise of unsiloed data: PwC*. PWC Technology Forecast [cited 2014 2014-08-21]; Available from: http://www.pwc.com/us/en/technology-forecast/2014/cloud-computing/features/data-lakes.html.
7. Heudecker, N. and A. White. *The Data Lake Fallacy: All Water and Little Substance*. 2016-09-06]; Available from: https://www.gartner.com/doc/2805917/data-lake-fallacy-water-little.
8. Harrison, G., *Next generation databases: NoSQL, NewSQL, and Big Data: what every professional needs to know about the future of databases in a world of NoSQL and Big Data*. The expert's voice in Oracle. 2015, New York: Apress (IOUG). 235.
9. Allemang, D. and J.A. Hendler, *Semantic Web for the working ontologist: effective modeling in RDFS and OWL*. 2nd ed ed. 2011, Waltham, MA: Morgan Kaufmann/Elsevier. 354.
10. Robinson, I., J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*. 2015: " O'Reilly Media, Inc.".

CONTACT INFORMATION

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

Dr Peter Tormay

Capish Nordic AB

Carlsgatan 3

211 20 Malmö

Work Phone: +46 40 10 88 80

Email: peter.tormay@capish.com

Web: www.capish.com