

## D9.6: Final Report on Testing of Graph Technologies

Author(s)	Jedrzej Rybicki (JUELICH)
Status	Final
Version	v1.0
Date	11/04/2017

Abstract: This document provides an assessment of the available graph database technologies, describe relevant use cases, and report on testing of these. It also includes recommendation for uptake of the graph database technologies.

Document identifier: EUDAT2020-DEL-WP9-D9.6	
Deliverable lead	JUELICH
Related work package	WP9
Author(s)	Jedrzej Rybicki
Contributor(s)	Anna Queralto(BSC), Paolo D'Onorio De Meo (CINECA) Stephan Kindermann (DKRZ), Vasily Bunakov (STFC)
Due date	01/03/2017
Actual submission date	11/04/2017
Reviewed by	Claudio Caccari (CINECA)
Approved by	PMO
Dissemination level	CONFIDENTIAL
Website	www.eudat.eu
Call	H2020-EINFRA-2014-2
Project Number	654065
Start date of Project	01/03/2015
Duration	36 months
License	Creative Commons CC-BY 4.0
Keywords	

*Copyright notice:* This work is licensed under the Creative Commons CC-BY 4.0 licence. To view a copy of this licence, visit <https://creativecommons.org/licenses/by/4.0>.



*Disclaimer:* The content of the document herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the document is believed to be accurate, the author(s) or any other participant in the EUDAT Consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the EUDAT Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the EUDAT Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

## TABLE OF CONTENT

<b>1. EXECUTIVE SUMMARY .....</b>	<b>4</b>
<b>2. INTRODUCTION .....</b>	<b>5</b>
<b>3. USE CASES .....</b>	<b>6</b>
3.1. B2FIND Data .....	6
3.2. Professional Network.....	7
3.3. Tracking Objects in B2SAFE.....	8
3.4. Modeling ENES Domain .....	8
3.5. Alternative View on B2SHARE Metadata.....	9
<b>4. PERFORMANCE AND SCALABILITY TESTING .....</b>	<b>10</b>
4.1. Performance and Scalability Testing of Storing Semantic Annotations.....	10
4.2. Performance Testing for the Professional Network Use Case.....	11
<b>5. ADOPTION PLANS.....</b>	<b>14</b>
<b>6. SUMMARY AND OUTLOOK.....</b>	<b>15</b>

## LIST OF FIGURES

Figure 1: Performance evaluation of neo4j v2 vs. v3.....	6
Figure 2: General idea of APOC-based partial updates .....	7
Figure 3: Comparison of MongoDB and neo4j performance.....	11
Figure 4: Data model including inferred relationships between authors.....	12
Figure 5: Results for query d, with and without materializing “isConnectedTo” .....	13

## 1. EXECUTIVE SUMMARY

This is the final report on testing of graph technologies in EUDAT2020 WP9.2. During the two years of the activity, we set off on evaluating if and how graph database technology can be applied to different EUDAT-inspired use cases. The general introduction into graph databases and description of our approach can be found in the initial report produced by this task (D9.3). In this document we report on the progress done and address the original hypothesis that graph databases can not only be used to substitute some of the technologies used currently in EUDAT but also offer additional benefits. We conclude this deliverable with a summary of possible uptakes of our results.

In short, we have shown that there is merit in using graph databases as they enable new usages of the data already collected in EUDAT domain. In some of the cases (e.g. professional network) the obtained results would be not possible with other technologies. We also provided evidence that the selected graph database technology is scalable and performs well.

Amid the positive results of evaluating both functionality and performance of the graph technology this task issues a clear recommendation to adopt the technology in the EUDAT CDI. A higher-level recommendation from our work, is to make the content collected in the EUDAT CDI a first-class citizen rather than focusing on the services and their features. If the content managed (even in most sophisticated and feature-full services) is not attractive to the users, it is a waste of resources. Graph databases can help in engaging users to explore available content and thus increase value of the EUDAT CDI.

## 2. INTRODUCTION

The WP9.2 was charged with a task to evaluate the applicability, usability, and scalability of graph databases in and for EUDAT infrastructure. To this end, we decided to split the efforts and follow EUDAT-inspired use cases, each task partner has identified and followed at least one use case. The goal of this parallel efforts was to identify interesting usages of the data already collected by EUDAT. We will report on this in the following section.

A cross-cutting concern expressed to us during the EUDAT Developers Meetings was a question if the neo4j (selected graph database technology) can reach sufficient performance and scalability in production setting. We conducted such evaluations, shared the results, and devoted a separate section to this subject in this report.

The reason for the task WP9.2 to end in M24, was to give enough time to uptake proposed solutions and suggestions by the operational infrastructure. This will be a longer process but in this deliverable we will report on its current status and perspectives. It should be stressed that the decision of using an established graph database as a basis for our work, resulted in high maturity level of the final solution.

### 3. USE CASES

Although a lot of work in this task happened in parallel to each other, we strived to produce one database that unifies most of the following use cases. The database is quite large with over 600 thousands nodes and 5 mln relations and covers B2FIND data, B2SHARE data, and additional relations from professional network use case. We envision this database to be adopted by EUDAT to offer alternative view on data gathered in EUDAT and facilitate new usages of the data.

#### 3.1. B2FIND Data

The development of B2FIND use case for the graph DB technology was defined by the next major version of neo4j (3.x) that came out in 2016. The first thing to look into was whether the new graph DB engine offers any performance gains compared to the older version. B2FIND graph, despite being substantial - in fact, the biggest of all graph DB use cases with 5 million relations in it – is still relatively small for performance testing being done with the types of queries corresponding to user actions in B2FIND graphic user interface. This is why the more challenging queries aimed at gathering database-wide statistics were used to compare the performance of neo4j v2.x versus v3.x over B2FIND graph.

The performance was first measured over a series of queries in the graph DB under neo4j v2.2.3, then the database was converted in a new format and the same series of queries were performed over the graph DB under v3.1.1.

The diagram below shows that for simpler requests like gathering counts of data creators by discipline, there is no noticeable performance difference between the versions. For more complex requests that involve what in the relational DB world would be multiple “joins” (and in the graph DB, multiple types of relations) as well as filtering out results by certain threshold values, the performance gains are significant for the new graph DB engine.

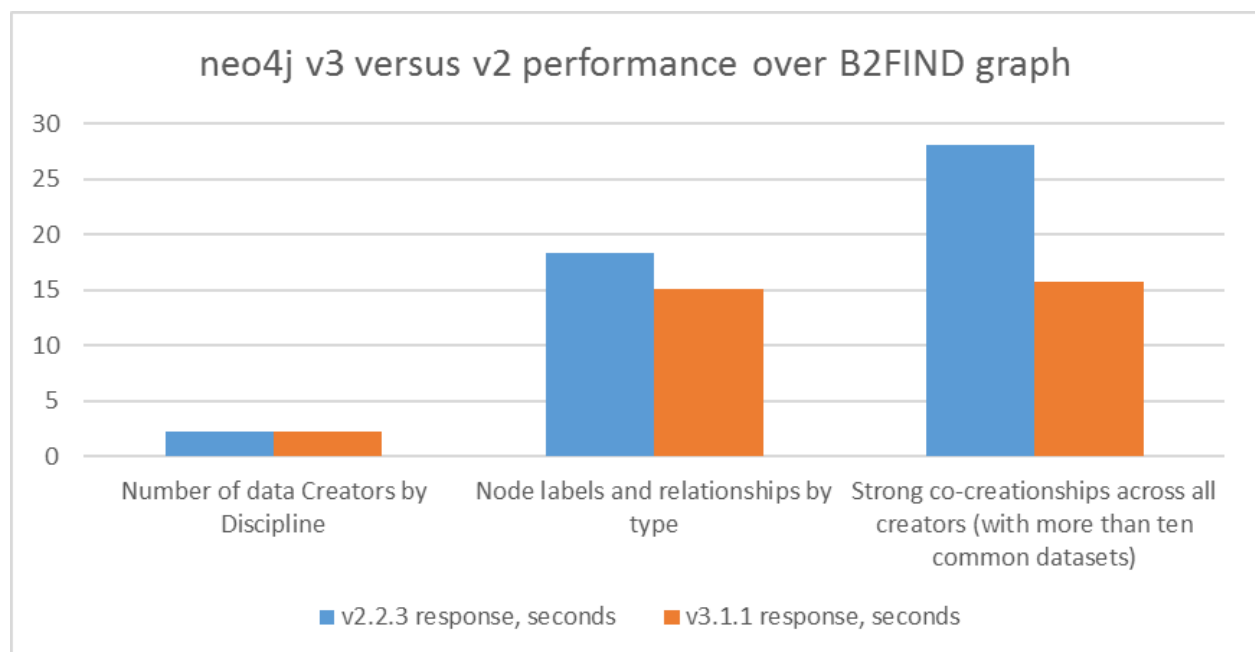


Figure 1: Performance evaluation of neo4j v2 vs. v3

Another novelty that version 3 of neo4j has brought about are the stored procedures and libraries of them. We evaluated this new feature for the case of B2FIND “delta” updates so that instead of bulk upload of the updated B2FIND export in neo4j, we could possibly use the on-the-fly requests from the graph DB instance to B2FIND, parsing responses and creating new or updated portions of the B2FIND graph.

For this purpose, we selected the APOC<sup>1</sup> library of procedures which is not a part of the standard neo4j distribution but has got an active community for its development and support. The APOC calls were tried as a wrapper for B2FIND API calls in order to, first, inquire B2FIND on the recent updates of the datasets, then create/update nodes and relations for the recently updated datasets. Below is the diagram for the workflow we considered.

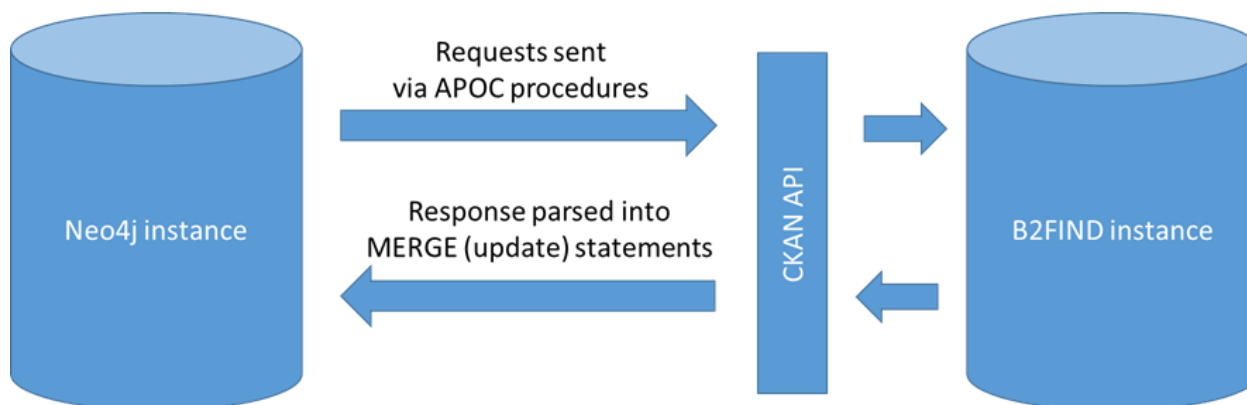


Figure 2: General idea of APOC-based partial updates

This exchange worked pretty well as a proof of concept, yet certain issues with the library used were discovered and reported back to the APOC community. Also, B2FIND instance configuration should be tuned in order to use this workflow in production as by default, only a limited number of recently updated data records is returned from B2FIND via its CKAN API.

Overall the technology of on-the-fly requests from the graph DB to the outer data sources looks promising and can be applied not only for “delta” updates of the particular graph (B2FIND graph in our case) but also as a flexible data integration technology, so that the graph DB instance can be used as a workbench capable of formulating very specific requests to external data sources and mashing up the received data into the graph of a desired shape, with desired semantics.

### 3.2. Professional Network

The idea behind this use case is to take advantage of graph technology to exploit the information currently stored in EUDAT services, namely B2FIND and B2SHARE. One of the main features of graph databases is that they are designed to efficiently perform traversals, which are very costly in other kinds of databases. We used the collected metadata for graph traversals to gain new insights about authors and the relationships between them. First, this kind of analysis can be useful for EUDAT users to establish new collaborations. Those can be established by analyzing the authorship of data object in EUDAT domain. Secondly, the same information about authors, their relationships and their scientific products can be used to analyze this data from a more global point of view. For instance, identifying the persons with more connections (either direct or indirect), or determining the average co-author degree in EUDAT.

With this goal, we defined a set of queries and executed them on the graph containing B2FIND and B2SHARE data. Despite being the kind of queries that graph databases are designed for, these executions showed that running the queries on demand, especially the global ones, was unfeasible already with the current amount of data. Global queries means that we did not use a particular starting point in graph, but rather tried to find all possible collaborations in given graph. Materializing some inferred relationships, that is, creating additional links between authors when a new data object appears, makes it possible to run the envisioned queries efficiently.

<sup>1</sup> <https://github.com/neo4j-contrib/neo4j-apoc-procedures>

In the future, the use case could be extended by integrating also B2NOTE data, once this service is integrated in EUDAT. In this way, information about the contents of the data objects could be taken into account in the queries instead of community or discipline, in order to specify more precise conditions on the expertise of the people searched for.

This use case showed that the metadata that is currently stored in EUDAT can be reused for other interesting purposes in addition to search and discovery of research data. Also, we found that if B2FIND information is to be used for this kind of explorations, it would be desirable that the values for the metadata fields were based on an ontology that would provide richer information. For instance, we could identify the relationships between disciplines and subdisciplines and take them into account to answer the queries in a more generic way, as opposed to the current version in which only exact matches can be considered.

### 3.3. Tracking Objects in B2SAFE

B2SAFE service aims at storing and distributing large volumes of data for a long-term access. While holding a consistent amount of datasets the community is often interested into the related metadata. To store it internally into the same service was the new goal to explore through this task, since graph databases are flexible enough to allow any community internal schema to be mapped into the B2SAFE data model.

Concurrently to a new B2SAFE automatic metadata extraction module<sup>2</sup>, a first prototype<sup>3</sup> was developed to design the best model: the goal was to keep it simple enough to save files and system metadata and associate them to extracted metadata. The results were optimal and pushed the B2SAFE team to adopt the model - with slight modifications - and the graph database into their upcoming feature “community metadata management”<sup>4</sup> as a beta release in March 2017.

Since the code rewriting of the B2STAGE HTTP API has become necessary in the same time to provide a widely-adopted standard for the community to access the B2SAFE datasets, the adoption of the graphdb has been considered as a part of the same process. We are going to officially provide the database within the HTTP API after the first production release<sup>5</sup>. All operations to the graphdb may be executed through the HTTP API service to make them consistent. The adoption of internal endpoints to internally process the stored metadata would also allow the B2SAFE service to be easier linked to B2FIND in the near future.

Performance evaluation for large data processing within the graphdb model is currently in progress. Future development on the same metadata local store would allow the community in depth searching for metadata relationship of their own datasets through the graphdb capabilities.

### 3.4. Modeling ENES Domain

The basic idea behind the ENES use case is the integration of information about data (collections) available in the ENES data infrastructure with information on ENES collections within the EUDAT domain. For this three graph generation and integration prototypes were developed<sup>6</sup>:

1. A graph generation script accessing the ENES (meta-)data index (solr / ESGF search API) to generate a neo4j graph representing the available data collections in the ENES data infrastructure
2. A graph generation script accessing the EUDAT B2FIND index (ckan API) to generate a neo4j graph representing the ENES data collections visible in the EUDAT domain.

<sup>2</sup> <https://github.com/EUDAT-B2SAFE/B2SAFE-metadata>

<sup>3</sup> <https://github.com/pdonorio/irods2graph>

<sup>4</sup> <https://confluence.csc.fi/display/EUDAT2/Data+provider+and+system+metadata+management>

<sup>5</sup> See Roadmap in <https://confluence.csc.fi/display/EUDAT2/HTTP+API+Development>

<sup>6</sup> [https://github.com/stephank16/enes\\_graph\\_use\\_case](https://github.com/stephank16/enes_graph_use_case)



3. A graph generation script accessing provenance records (referring to ENES data infrastructure ingested data collections) given in W3C prov standard representations (json or xml) to generate neo4j graph instances.

This way ENES community data sets available in the EUDAT domain representing curated data collections with DOIs assigned to them can be related to data collections in the ENES data infrastructure (e.g. newer or replicated versions of the same collections) and their data ingest history represented in W3C prov records. Discussions with EUDAT B2FIND lead to a new harvesting of ENES data sets, now making the DOI information accessible (opening up the future integration possibilities with authorship (e.g. ORCID) related information available in the graph.)

This use case, on one hand, showed the flexibility of the graph technology to integrate various related information entities. On the other hand the replication of the full information available in the ENES (metadata) catalogs is not feasible. Whereas the generation of the fine grained related graph model is possible (yet time consuming) - resulting in graph representations with millions of nodes and relations - the maintenance of the graph (synchronization with updates, deletions etc. in the ENES domain) is complex and not feasible. Thus, appropriate granularities of data collections representing nodes in the graph model have to be defined based on concrete envisaged usage scenarios. An opportunity to define concrete usage scenarios are integration and service pilots planned for current H2020 e-Infra 12 and e-Infra 21 calls.

### 3.5. Alternative View on B2SHARE Metadata

The use case was dealing with representing data currently stored in B2SHARE service. In the final months of the activity some model adjustment were conducted to make the B2SHARE data model compatible with the model from B2FIND use case. Furthermore, the integration of the data revealed some data inconsistency. For instance there were different ways of spelling names of objects or people in the existing data set. We have prepared a prototype to calculate semantic similarity between entity names in graph. This is perhaps not really a graph specific problem but it is strongly connected to the issue of data integration and was not addressed by EUDAT so far.

Since the changes in B2SHARE model and content were not very rapid, we devoted some of the effort to evaluate a task formulated by Technical Committee. TC wanted us to test if it is possible to make the data from graph database available through OAI-PMH interface which is currently used by B2FIND to harvest metadata. We created a prototype to facilitate such integration of neo4j with OAI-PMH interface. An exact description of the taken approach was documented, source code was made available, and passed to TC<sup>7</sup>.

Finally, there were some work on queries on the collected data. We emulated a situation of making suggestion to a user on data sets that might be relevant for his work (e.g. datasets with similar metadata tags). All this results were discussed with social network use case due to the similarity of the approach.

---

<sup>7</sup> <https://github.com/httpPrincess/neo-oai-experiments>

## 4. PERFORMANCE AND SCALABILITY TESTING

Database engines provide many ways to tune the performance for envisioned use cases (and graph databases are not an exception). Thus, in our work we did not intend to evaluate a generic performance and scalability of neo4j but rather test its suitability to handle EUDAT-specific use cases. Beyond that, the performance and scalability is a cross-cutting concern for all the use cases discussed above. Optimally we would compare the results for neo4j and technologies currently used in EUDAT. Unfortunately there were no performance and scalability tests conducted for basis EUDAT services like B2SHARE, B2SAFE, or B2FIND, so we lack a good benchmark for comparison.

### 4.1. Performance and Scalability Testing of Storing Semantic Annotations

One of the newly developed services in EUDAT is aiming at adding semantic annotations to data objects stored in the infrastructure. Current implementation of this service (B2NOTE) is using W3C Annotation Data Model and stores annotations in this format in MongoDB. The service is at its early development stage and the coding is focusing on the implementation of the required functionality and integration with existing services, rather than looking for the best possible storage backend. Thus, it makes sense to use this use case as a benchmark for evaluating neo4j and comparing it to MongoDB.

We have prepared an extensive report which was passed to both TC and the developers of B2NOTE service. It is also available with programs for evaluation in GitHub<sup>8</sup>. Here we just present excerpt of it. The test comprised of three types of operations:

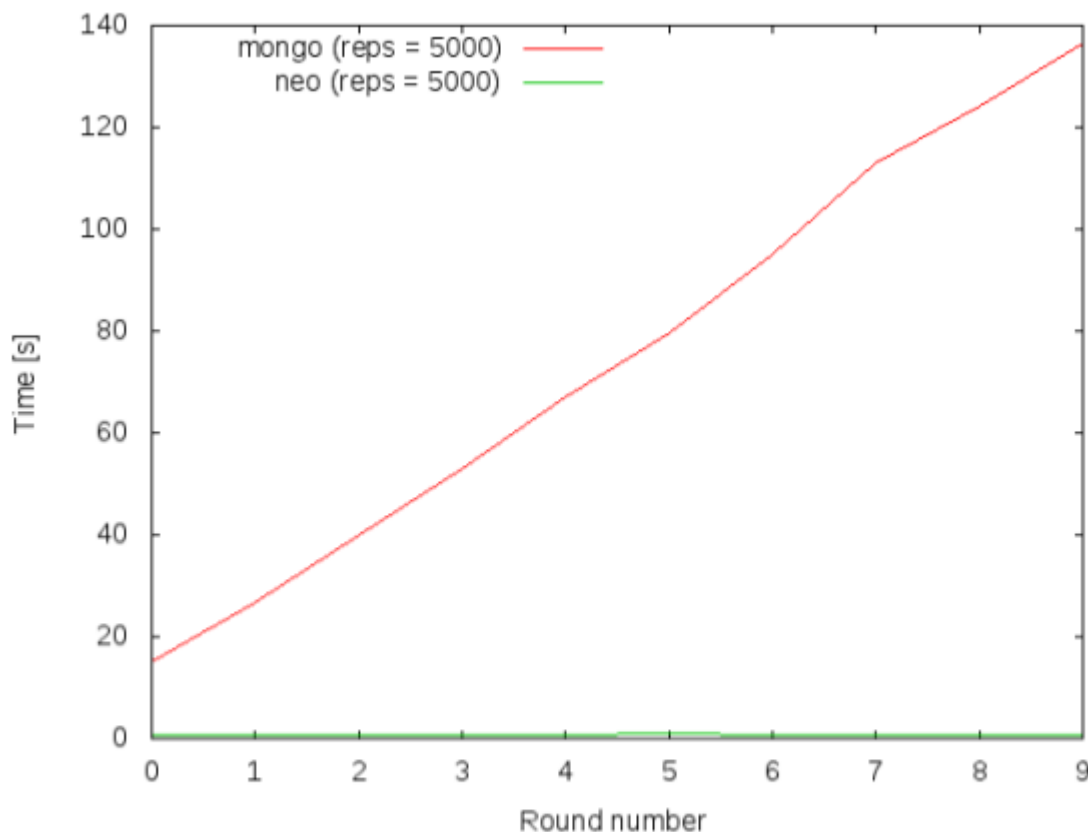
1. creation of new annotations,
2. retrieval for given body id,
3. retrieval for given target id.

The later ones could be translated into finding all the annotations for a given data object, and finding all the object with given annotations attached. We tested both performance and scalability of the graph database and compare it to the MongoDB. Since the retrieval of the data would be most probably the dominating operation we present a direct comparison of the selected products in this regard. The results are depicted on Figure 3 and show that MongoDB has problems in handling increasing size of the database (in each round new 5000 records are added).

At this point it is worth stressing that the testing procedure was prepared with help of Docker and docker-compose, in such a way that redoing the tests (for instance to verify them) is pretty simple.

---

<sup>8</sup> <https://github.com/httpPrincess/annotations-scalability>



Comparison of MongoDB and neo4j retrieval scalability. In each round reps=5000 records are added, and reps=5000 random records are retrieved.

Figure 3: Comparison of MongoDB and neo4j performance

## 4.2. Performance Testing for the Professional Network Use Case

The aim of this test is to check the feasibility of providing professional network functionalities in EUDAT using graph technologies. The queries needed to support this functionality can be classified into two types:

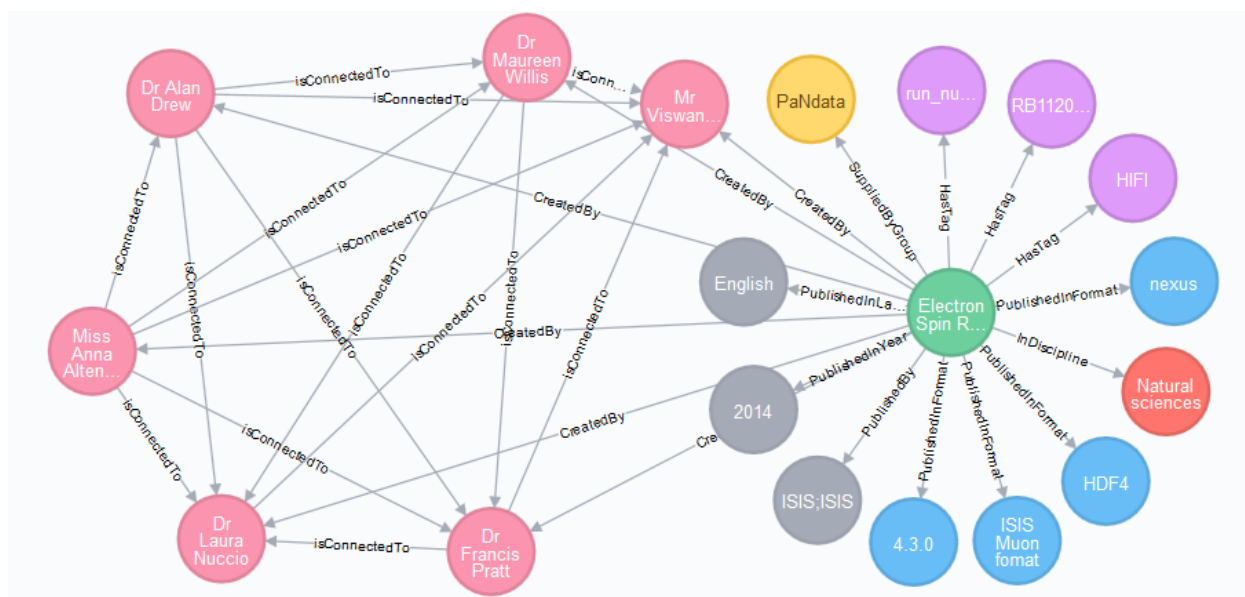
1. Personal queries: queries from the point of view of a specific author, in which the source of the traversal is fixed. For example: My collaborators (would be equivalent to connections in LinkedIn or friends in Facebook). A collaborator is someone that has co-authored a data object or collaborators from a given discipline. The relations can be weighted in neo4j, and the “strength” of a collaboration could be dependent on the number of common objects.
2. Global queries: queries aimed at analyzing the graph as a whole, not from the point of view of a given author. For example: Strong collaborators summary (pairs of authors with at least 10 common data objects), Average number of connections per author, up to a given distance, etc.

We tested the different queries in three graphs containing 7,000, 10,000 and 440,000 data objects (with their associated authors, disciplines, etc.). The last graph corresponds to the complete set of objects in B2FIND complemented with information from B2SHARE, with a total size of almost 630,000 nodes and 5.4 million relationships.

Queries in the first group, i.e., exploring local connections are not problematic from the point of view of performance, since they take a few seconds in the complete database. Examples of a holistic graph

analysis which can be performed quickly are aggregating queries like number of authors from given community. They rely on neo4j indexes and/or short graph traversals. However, holistic queries like authors with most collaborations up to given strength required much more runtime.

To solve this problem, we materialized the inferred relationship “*isConnectedTo*” to represent the connections between authors, as shown in Figure 4. This relationship has a property we called “distance” that represents the number of hops between the two authors. We created the relationships for distances 1 (meaning the two authors have collaborated in the same publication) to 3, since we believe that longer distances do not provide very useful information. Notice that, although the concept of connection is symmetric (i.e. if A is connected to B then B is also connected to A), we just need to create the relationships in one direction since Neo4j is able to traverse edges in both directions. This reduces the total number of relationships in the graph, and also simplifies the correct definition of the queries. The number of new relationships added to the original graph is 5.14 million.



**Figure 4: Data model including inferred relationships between authors**

Figure 5 shows the execution time in seconds of query d, as a representative of the group of costly queries, for the databases with 7,000, 10,000 and 440,000 nodes. The rest of queries have an analogous behavior. We have executed the query to find collaborators at distance 3 or closer, both calculating the paths on demand (corresponding to label “Calculated” in the figure), and also using the inferred “*isConnectedTo*” relationship (label “Materialized”). As can be seen, calculating the paths on demand is not feasible in the complete (440k) database, while pre-calculating and materializing the connections between authors keeps execution time very small (less than 18 seconds in average).

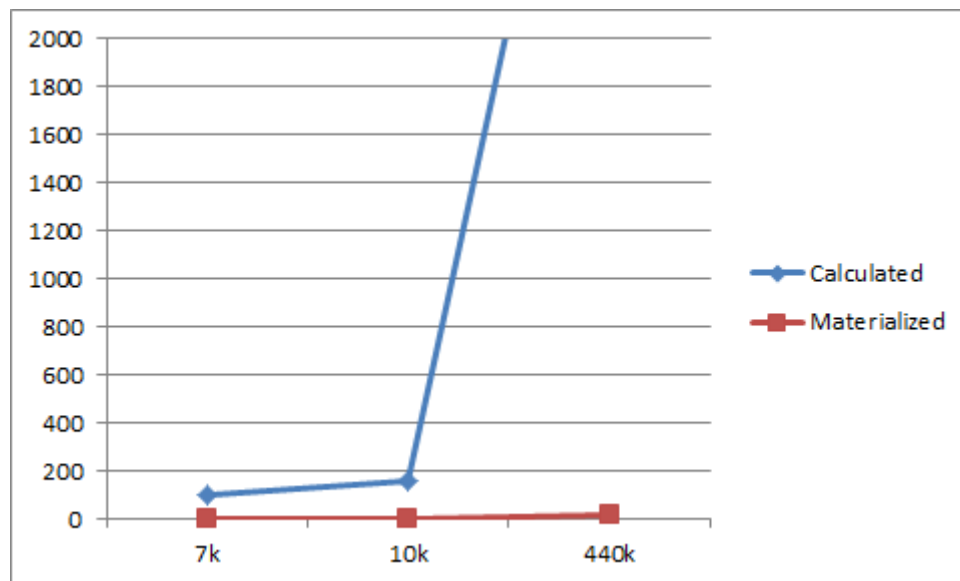


Figure 5: Results for query d, with and without materializing “isConnectedTo”

Having the relationships materialized requires calculating them every time a new data object is added to the database, so we need to ensure that this cost can be assumed when a new data object is inserted. The cost very much depends on the number of relationships of each author but, in average, according to our tests the creation of all the connections for a single author is done in less than one second. Thus, the overhead of adding these additional relationships when a new data object is created depends on the number of authors of the data object, which is currently 1.8. This means that we have to pay about 2 seconds to create the relationships for each insertion, but it will enable the execution of the queries in a few seconds.

More details about the evaluation can be found in the EUDAT wiki<sup>9</sup>. The analysis summarized here shows that providing a social network feature in EUDAT based on Neo4j would be feasible only if the needed inferred relationships are always kept up to date, which can be done at a negligible cost.

<sup>9</sup> <https://confluence.csc.fi/display/EUDAT2/Performance+testing+for+the+professional+network+use+case>

## 5. ADOPTION PLANS

The WP9.2 activity is due to end in M24 of the EUDAT project. The reason for finalizing this task so early is to enable Service Building activity to potentially uptake of the solutions and profit from the gained insights. To this end, we remained in contact with the WP5, WP8, and Technical Committee. There was a presentation of graph database technology prepared for the TC, which resulted in two specific tasks that WP9.2 should look into. In particular possibilities of exporting content of graph-database through OAI-PMH interface, and the performance and scalability evaluation of neo4j databases were requested and provided in time.

Amid the positive outcomes of those two tests, it was decided that neo4j can be used as an extension of B2SAFE to serve as local metadata store. The adoption plan in this case has been divided into two main steps. The first one required having one of the tasks of WP9.2 to prepare a graph model of the data stored in B2SAFE, used shortly after as a basis for the newly created local metadata store (LMS). The LMS has become necessary to save the automatically extracted metadata where communities explicitly their custom schema through files in XML format uploaded within the stored datasets. The second step is still ongoing: the HTTP-API development inside the B2STAGE service will help B2SAFE in managing the graphdb as a metadata store by testing performances on a large number of insertion and updates of the metadata modeled from the previous step in different scenarios. HTTP API would also become the main (and only) wrapper to operations of any service that insists on the LMS graphdb. Part of the benchmarks aims at testing an integration of a messaging queue to delay the effort of resources while processing transactions within the graphdb content.

A more sophisticated use case was addressed together with WP9.1.1 which deals with object stores. Object stores constitute an interesting point in the design space, offering efficient, scalable storage with low operation overhead. The main drawback of object stores is the very limited namespace functionality they offer. Thus, for many users a move from a file system with hierarchical namespaces of directories to a “flat” structure of objects in object stores might be perceived as a step back. In many cases, the hierarchy of directories serves as a metadata store, embodying additional information about the files stored. To this end, we looked at possibilities of connecting object stores with graph databases. The later ones should be then responsible for managing the namespace. The graph databases can not only mimic a hierarchical tree of directories but even accommodate for more sophisticated namespaces. The result of the joint work was a prototype of replicated object store passing information about changes in the location of objects to a flexible namespace built with help of a graph database. We plan to publish the results as a paper and make it available to EUDAT with clear recommendation for adoption as substitute for technologies (iRODS, EPIC PID) currently driving the B2SAFE service.

The extensive tests of the graph technology and identification of the potential benefits that cannot be easily achieved with other solutions justifies a question if the graph database should not be adopted as a central service, offering different view on the data gathered in the EUDAT infrastructures. We suggested this kind of change of EUDAT architecture to the TC. Far reaching goal of such a service would be to provide a holistic yet explorable view of the content collected in the EUDAT infrastructure. Potentially attracting more users to use the EUDAT services.

## 6. SUMMARY AND OUTLOOK

This deliverable is the final report on testing the graph technologies in WP9. We performed evaluation of both functionality and performance. With regard to functionality we identified use cases and queries which can run very well on the graph database containing integrated data from EUDAT domain. Also the performance evaluation produced satisfactory results, yet there were cases where selected graph database technology struggled. Those are the global queries evaluating holistic properties of a graph. We shown how this kind of queries can be speed-up, by materializing additional connections in the graph.

Given the positive results achieved by this task, we expressed a recommendation to the Technical Committee to adopt the graph database technology in the EUDAT infrastructure. The soft-skills obtained during the work can also be considered as an EUDAT asset and shared with communities and developers tackling graph-oriented problems.