Getting the best of Linked Data and Property Graphs: rdf2neo and the KnetMiner Use Case

Marco Brandizi[0000-0002-5427-2496], Ajit Singh[0000-0001-6239-9967], Christopher Rawlings[0000-0003-3702-1633] and Keywan Hassani-Pak[0000-0001-9625-0511]

Rothamsted Research, Computational and Analytical Sciences Department,
Harpenden, AL5 2JQ, UK
{name.surname}@rothamsted.ac.uk (chris.rawlings@... for CR).

Abstract. Graph-based modelling is becoming more popular, in the sciences and elsewhere, as a flexible and powerful way to exploit data to power world-changing digital applications. Compared to the initial vision of the Semantic Web, knowledge graphs and graph databases are becoming a practical and computationally less formal way to manage graph data. On the other hand, linked data based on Semantic Web standards are a complementary, rather than alternative, approach to deal with these data, since they still provide a common way to represent and exchange information. In this paper we introduce rdf2neo, a tool to populate Neo4j databases starting from RDF data sets, based on a configurable mapping between the two. By employing agrigenomics-related real use cases, we show how such mapping can allow for a hybrid approach to the management of networked knowledge, based on taking advantage of the best of both RDF and property graphs.

Keywords: semantic web, property graphs, Neo4j, knowledge networks, plant biology.

Availability: rdf2neo is available as open source software and shared on the GitHub code repository, at the URL https://github.com/Rothamsted/rdf2neo

1 Background

Sharing data, information and knowledge is ever more important in life sciences [1] and other human activities [2]. As recently highlighted by the well-known paper about the FAIR principles [3], we can achieve maximum benefits from data by sharing them according to the principles of Findability, Accessibility, Interoperability and Reusability. In the life science and other sciences the linked data approach [4] and the Semantic Web technologies [5] allow for the integration of heterogeneous knowledge and the study of phenomena involving interdependencies at multiple scales [6–8]. Over the years, science data practitioners have developed high quality data models, mostly based on ontology engineering and other highly formal modelling techniques [9]. While this gives advantages in many applications, for instance by allowing for advanced automatic reasoning and data validation [10], it is difficult for both software developers and domain experts to deal with these models and curate data that comply with them [11], especially when data are shared on a large scale [12]. For these reasons, knowledge graphs have appeared, especially in commercial contexts [13,14], where data are modelled in a more semi-formal way and logical inference is based on statistics and heuristic techniques, not only on formal logics [15]. At the infrastructural level, a number of data storage solutions have emerged in the last decade (or even conceived before the Semantic Web), including distributed relational-like databases, the so-called NoSQL databases [16,17] and graph databases based on the idea of property graphs (PG, [18–20]). Property graphs are different from triple stores based on the RDF's Semantic Web

standard, in three key ways [21]. Firstly, the PG data model is more coarse-grained than RDF, allowing the model to group all the datatype properties of an entity in a single node, whereas they require multiple triples in RDF. In a PG this is extended to the graph edges, i.e., binary relations linking entities, which can be enriched with attributes describing aspects like the provenance of a relation or its confidence score. Doing the same in RDF is known to be more complicated, since it requires graph patterns like reification [22]. On the other hand, the finer granularity of RDF makes it easier to do operations like merging attribute lists about an entity coming from different data sources. A second difference is in the underlying philosophy of PG projects like Neo4j (and, more in general, of NoSQL databases [20]), in the sense that many use cases are focused on providing stores for single applications and single organisations, with less commitment to, and support for standardisation, interoperability and sharing. This contrasts with the priorities in the Semantic Web World, where sharing and interoperability are primary motivations. A third difference is that triple stores are best at graph pattern searches, rather than more local searches based on graph-traversal algorithms, for which several PG databases are optimised [23].

1.1 Motivating Use Case

Our group has been involved in biological data integration and discovery for many years. In particular, we maintain the KnetMiner software suite [24], primarily based on web components, which can be used to explore large scale and semantically-rich knowledge networks, named Genome-Scale Knowledge Networks (GSKN [25]). Starting from keyword-based search, the user can explore networked knowledge that include genes, encoded proteins, phenotypes, disease, biological pathways and PUBMED citations. We use KnetMiner mostly to support the analysis of plant biology and agri-genomics-related organisms. Our published data sets include Arabidopsis, wheat, rice and maize. Our data types include ENSEMBL genes [26], AraCyc pathways [27] and Plant Ontology [28]. Recently we have started to redesign the backend architecture that KnetMiner is based on (see Figure 2 in [29]). In so doing, we aim to address three different objectives: i) better separation between data access services and data client applications, namely adopting web APIs and common graph-oriented query languages ii) management of data through common graph-oriented database servers, rather than ad-hoc solutions iii) Publicly sharing our data by means of well-known data standards. Based on a preliminary analysis of several options available [30] and further investigations presented in this paper, we propose that a “traditional” linked data approach can usefully be combined with the adoption of more recent property graph systems. In particular, we have developed BioKNO [29], a lightweight OWL-based ontology, falling under the category of application ontologies [31], which we are using as the reference data model to integrate biological data from external sources, available in different formats and data models. In order to facilitate data integration tasks as well as the development of Semantic Web-oriented applications, we have made a SPARQL endpoint available, based on the Virtuoso triple store [32]. Moreover, we have established a data pipeline to convert our RDF-based data (hence, after initial extraction, loading, transformation, or ELT) into Cypher instructions, to be used to populate instances of the Neo4j property graph database [33]. We have decided to add this additional data access channel, because our bioinformaticians consider Cypher and Neo4j useful tools for a number of tasks. For instance, we are using Cypher as a query language for what we call semantic motif searches [24], which are graph pattern-based searches tracking well-known relation paths between biological entities (e.g. Gene-> expresses->Protein-

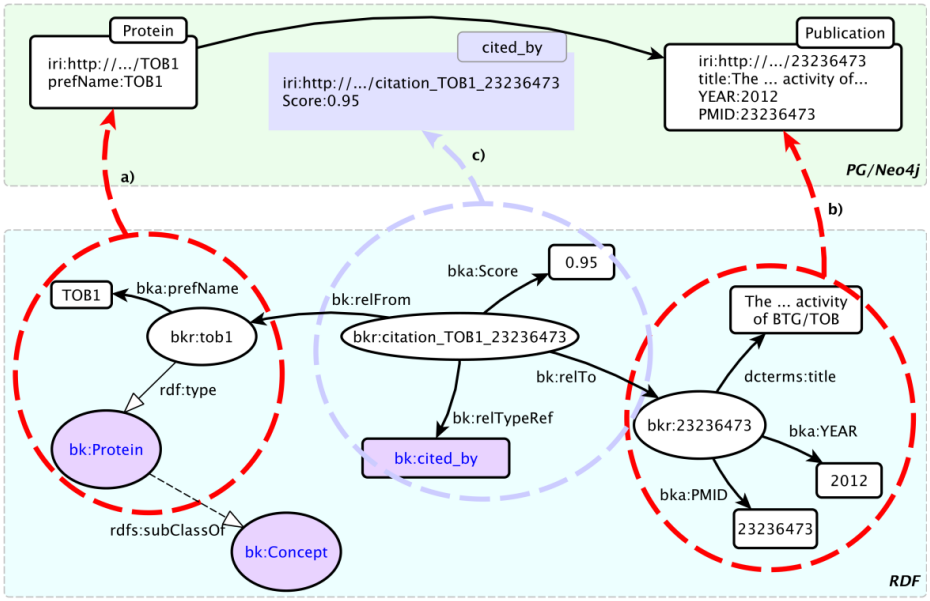
>published_in->Publication). We use semantic motifs for searching plant biology data, to implement term-based gene searches and to rank results based on scoring the relevance of a gene to the searches.
In order to support our use case, we have developed rdf2neo [34], a tool that can be flexibly used to map any RDF schema to a desired PG schema, so that it is possible to rely on RDF/OWL for data modelling and acquisition purposes and then deliver data through a number of useful formats and technologies. In particular, the configurable mapping available in our tool eases the definition of PG schemas aligned to the RDF-based model, which favours a data-centric application development.

2 Methods & Implementation

2.1 The rdf2neo Approach

rdf2neo is implemented as a Java library, which can be used either programmatically or by means of a command-line package. As shown in Fig. 1, its main function is mapping an RDF-based data model to a Property Graph data model, which is eventually created by a set of Cypher commands. SPARQL is used to select groups of values that make up the entities of a target property graph (nodes or relations). This works in two steps. Considering the case of PG nodes, first a query selects URIs[1] of RDF resources that correspond to PG nodes (e.g., *bkr:tob1* in the figure). Then, each of those URIs is used to instantiate two other SPARQL queries, both parameterised on a conventionally-named variable (*?iri*). One query is used to extract node labels (usually analogous to RDF/RDF-S/OWL classes), the other is used to extract pairs of property name + value for that node (e.g., *dcterms:title*, *bka:YEAR*). These node-related data are then used to issue Cypher instructions to a Neo4j server. Relationship mapping is similar to the node case: one SPARQL query, lists all the PG desired relationships, together with the URIs of the relationship origin and destination nodes, as well as a string about the relationship type (contrary to node labels, a relationship can have one type only in Cypher/Neo4j). Another SPARQL query, instantiated with the relationship's URI, lists the relationship property/value pairs, if any (*bka:Score* in our figure). In both node and relationship creation, the original node/relationship URI is always stored on the Neo4j side, by means of the property named 'iri'. This, in addition to being useful for future reference, is used for our operations, to refer nodes during the relationship creation stage.
The process above is implemented in the architecture shown in Fig. 2. This comprises a component that selects node URIs (same for relation pointers) and dispatches them to parallel threads, each fetching node (or relation) details, to eventually populate Neo4j via Cypher.
The SPARQL files we mentioned above are part of rdf2neo configuration, which is managed through Spring Beans configuration files [35]. This way, the task of translating a new RDF data set into a Neo4j database consists of defining new SPARQL mapping queries in a set of files and passing them to rdf2neo through a Spring configuration. This makes configuration flexible and convenient for the many developers who are familiar to Spring.

[1] In the code we use the term IRI in order to mean the general case, where practically IRIs can be considered as URIs with international support. Hereby, we call them with the better known URIs word and, for what matters us, the two acronyms can be considered synonyms.



a) b) Selection of node IRIs

```
SELECT ?iri {
  ?class rdfs:subClassOf* bk:Concept.
  ?iri a ?class.
}
```

a) b) Selection of node labels.
?iri is a parameter

```
SELECT ?label {
  ?iri a ?label.
}
```

a) b) Selection of node properties (ie, attributes).
Works similarly for relations (e.g., selection of bka:Score).
?iri is a parameter.

```
SELECT ?name ?value {
  { # Annotations of interest listed explicitly
    ?iri ?name ?value. VALUES ( ?name ) {
    (dcterms:identifier) (dcterms:title) ... }
  } UNION {
    # Annotations matching a criterion (eg, bka:prefName)
    ?iri ?name ?value.
    ?name rdfs:subPropertyOf* bk:attribute.}
}
```

c) Selection of relation IRIs/types/endpoints, reified relations

```
SELECT ?iri ?type ?fromIri ?toIri {
 ?iri a bk:Relation;
   bk:relTypeRef ?type;
   bk:relFrom ?fromIri;
   bk:relTo ?toIri.
}
```

c) (cont.), plain triples

```
SELECT ?iri ?type ?fromIri ?toIri {
  # Relations of interest listed explicitly
  VALUES ( ?type ) { ( bk:dataSource ) ( bk:evidence ) }
  ?fromIri ?type ?toIri.

  # ?iri is used in Cypher to assign an ID to each relation,
  # so we define reified IRIs here
  BIND(IRI(CONCAT(
    bkr:, MD5(CONCAT( ?type, ?fromIri ?toIri ))))
    AS ?iri )
}
```

Fig. 1. neo2rdf mapping approach. a,b) A SPARQL query selects URIs of resources to be turned into property graph nodes (instances of concepts in our case). Each of such URIs instantiates two parameterised queries, to pick up node's labels (i.e., types) and node's named properties (URIs are automatically shortened and turned into strings like 'Protein'). c) Relation pointers and their properties are selected in a similar way.

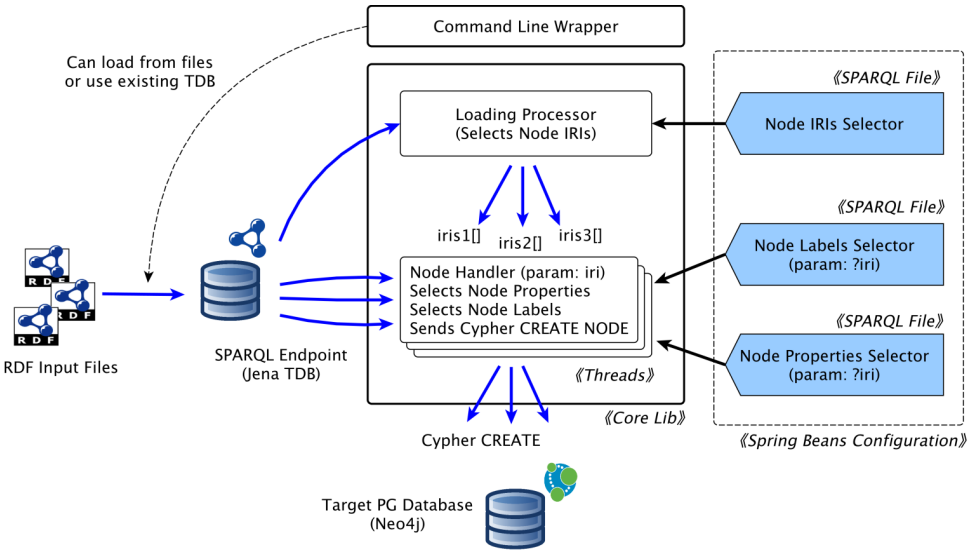


Fig. 2. Core components of the rdf2neo architecture, node case. In the main thread, a processor issues the SPARQL to select node URIs. Then URIs are sent to parallel instances of a node handler, which instantiates parameterised queries to get node labels and node name/value properties. The handler uses these results to build Cypher commands for node creation. Relations are created with similar components. SPARQL queries and other elements (e.g., Neo4j credentials) are configured via Spring Beans.

2.2 Formal Analysis

In order to ensure the correctness of the RDF/PG process described above, as well as study relevant properties, we have developed a formalisation of it (see Supplementary Document 1). One result we achieved from this analysis is that the computational complexity of our conversion procedures is mainly affected by the SPARQL mapping, and less by the overhead of Neo4j/Cypher interaction. Thus, the key factor for improving the performance of rdf2neo is by optimising the specific SPARQL queries used to realise such a mapping in a given use case [36,37].

3 Results

In order to assess the performance of the two types of graph databases we use, we have created a preliminary simple benchmark of specific aspects of interest to us. Namely, we compared Virtuoso to Neo4j. On the one hand, we made a qualitative evaluation of how easy it is to write queries in both SPARQL and Cypher languages. On the other hand, we assessed the speed of the respective query engines, when used with the typical data that we deal with in the KnetMiner project. This included an evaluation of scalability issues, both in the case of data loading and data querying. Results are summarised below.

Fig 3. Query performance of Virtuoso and Neo4j. Top: typical graph in tested data sets (other queries are about Gene Ontology and other ontology trees). Charts: Three test data sets, of increasing size, first two from the plant species Arabidopsis (A. thaliana), third from wheat (T. eestivum). Every data set was tested against Virtuoso and Neo4j. All execution times are in ms.

3.1 Test Settings

As shown in [38] in detail, we have made separate tests for Virtuoso and Neo4j. For each database, we tested three different data sets, having different sizes and degrees of data complexity in terms of types of entities and relations represented. The data sources are the same we use for powering the KnetMiner application (see section 1.1). In our tests, we focused mainly on biological pathways and gene ontology annotations from the Arabidopsis and wheat plant species. Fig. 3 shows a typical graph structure that

occurs in all the three data sets, which is explored by many of the queries we used for the tests. In writing the queries, we considered aspects like: a) basic functionality (e.g., 'cnt*, 'sel' queries in the cited document) b) typical data retrieval occurring in our applications, as said, these are mainly navigating biological pathways (e.g., 'join*', '*union*') and ontologies ('varPath*') c) complex queries, known to be challenging for most query engines ('pway', 'grp*', '*exist*', '*Ag'), including queries inspired by other benchmarks available in literature ('nestAg'). For each of these query types, we wrote both a Cypher and SPARQL version. Writing two similar versions for the same data retrieval semantics was fairly easy in most cases, thanks to the use of a common data model.

3.2 Performance of Data Loading and Querying

Figure 1 in [38] shows the loading times we obtained for the three tested data sets and the two graph databases. The higher loading times in the case of Neo4j are due to the fact that these include the rdf2neo conversion described above, which spends time loading BioKNO/RDF data into its embedded TDB instance, as well as in querying them using the SPARQL mapping queries. While we plan to improve the performance of our tool, its conversion operations cannot be made as fast as loading plain RDF into efficient triple stores. That said, these times seem reasonable for a batch operation, which is not needed very often (a few times per year, when we update our data sets). Moreover, as expected, rdf2neo scales well enough with the input size.
Results from the query performance are reported in Fig. 3. As expected, Neo4j performs well in graph traversal queries (up to 17x), where initial nodes are matched against the query and then the graphs connected to them are expanded. Property graph databases like Neo4j are known to be optimised for this kind of task. On the other hand, Virtuoso showed performance better than, or comparable to Neo4j for queries involving the union of graph branches (eg, '2union', 'nestAg', 'exist'), for queries having mostly 'tuple-like' patterns (eg, 3x for 'joinRe', 8x for 'joinReif'), where graph patterns have to be matched by searching several triple patterns and graph traversal does not bring many advantages, and, probably for similar reasons, for queries involving aggregation ('*grp*' and '*Ag' queries). Most queries scale well with the data set size. We were surprised to see significant differences (up to 14s versus less than 1ms) in queries like 'count all nodes' or 'select all instances of a given type'. Presumably, Neo4j is much faster at running them because it exploits indices of coarse-grained node structures, while Virtuoso indices cannot rely on node-level aggregation.

3.3 Qualitative Considerations

An area where the Semantic Web technologies still appear to be more suitable than property graphs is in data integration. For instance, in RDF systems the merging of data about the same entities happens almost for free, by ensuring that URIs generated in different data sets are the same when they represent the same real-world entities. A set of techniques for doing so are known (e.g., [39]) and tools like TARQL [40] exist, to translate from format like CSV to RDF. Such tools often rely on the same Semantic Web standards they target (e.g., TARQL uses SPARQL). Schema mapping and alignment is another area where Semantic Web technologies are useful. As a small example of that, see the federated search example mentioned in [29], which was made possible by mapping our BioKNO ontology to standards like BioPAX. On the other hand, a system like Neo4j appears to be particularly good in the more classical scenario where

a data repository is accessed by a limited set of applications and users, who can rely on simple and well-known schemas, without having the wider Semantic Web ambitions of data integration on a very large scale. As explained above, the semantic motif case is an example of that. In fact, the typical semantic motif query that we need to study agri-genomics-related knowledge networks is a graph pattern representing a chain of given node and relation types, a kind of pattern that is a first-class citizen in Cypher, while the same requires a more verbose syntax in the triple-oriented SPARQL. These query language considerations vary significantly when other use cases are considered. For instance, the query 2union1Nest from our benchmarking queries shows that matching multiple branches in a single SPARQL query is relatively straightforward, mostly thanks to the possibility to nest UNION constructs and define an overall graph pattern that reflects the queried data branching structure quite closely (e.g., see red and blue paths in the Fig. 3 graph). Doing something similar in Cypher can be challenging, even for simple cases, although this might improve in future [41,42].

3.4 Limits

The tests described here are not intended to be comprehensive or general, as in the case of works like [43,44]. Our evaluation is limited to the typical use cases we deal with in the KnetMiner project, that is, knowledge networks about molecular biology, mostly biological pathways and ontologies. Furthermore, we have only considered two popular, knowingly performant graph databases for RDF and property graphs, although there are many alternatives [45,46]. Finally, we have run the tests sending one query at a time only, without considering load issues (again, both Virtuoso and Neo4j are known for having good parallel performance).

4 Discussion and Conclusions

4.1 Related Work

Populating a property graph database like Neo4j from RDF-encoded data has been gaining interest in recent years, due to the opportunities offered by combining Semantic Web data standards, including the expressivity of ontologies, with the ease of use and efficiency of property graph technology [47]. A well-known tool to import RDF data into Neo4j has been implemented as a Neo4j plug-in [48]. While this defines a fixed 'natural' mapping between RDF graphs and property graphs, we allow for arbitrary correspondences between the two. This might be a valuable advantage, if one wants to rename ontology-oriented entities on the RDF side (e.g., 'continuant') in favour of more domain-specific terminologies, or to simplify reified property patterns. Ad-hoc mapping from RDF to PG/Neo4j has been employed in other life science projects, including protein structure exploration [49], heterogeneous data integration [50,51] and ontology management [52]. [53] is an interesting alternative way to combine linked data and property graphs through query languages, which we are interested in exploring in future. Works like [54,55] show the benefit of reducing the granularity of the RDF data models through mapping them onto more coarse-grained structures.

4.2 Discussion

Our approach increases the opportunities for knowledge sharing by offering diversified interfaces and means to access the data, which are all based on a common model. In our recent work [29] we describe in detail how rdf2neo is part of a new infrastructure to support plant biology and agriculture-related data, and how this contributes to the realisation of FAIR sharing principles. To summarise this aspect, anchoring multiple formats and technologies to a common model helps with data Accessibility and Interoperability. Aligning such models to standard bio-ontologies contributes to data Interoperability. Supporting multiple technologies and query languages ease data Reuse and Accessibility. Adopting services like SPARQL endpoints and resolvable URIs helps with Findability.

4.3 Conclusion

Considering the analysis presented in this paper, adopting a hybrid, linked data and property graph approach significantly eases the collaboration and connections between different information types, enabling diversified access to knowledge networks based on common data models. By so doing, one can usefully combine the best of the linked data world (data modelling standards like ontologies, technologies aimed at data integration and interoperability) with those provided by the Property Graphs (simple query languages, particularly suitable for knowledge navigation use cases, data models and implementations optimised for graph traversal). While such hybrid architecture is not without disadvantages, e.g., having to maintain a more complex infrastructure and to have multiple IT skills, adopting it is recommendable in situations similar to our use case. Automating tasks like data conversions and alignment is a relevant activity in maintaining such a hybrid architecture and rdf2neo can contribute significantly to it.
In future, we aim at improving the performance of rdf2neo, in particular by issuing fewer SPARQL queries during node/relation property fetching. Moreover, we plan to improve data interoperability by better integration with bioschemas [56]. Another objective we have is to extend our benchmarking to alternative triple stores and graph databases such as Gremlin [57], as well as a wider set of scientific data.

5 Acknowledgements

This work forms part of the Designing Future Wheat (DFW) strategic programme (BB/P016855/1) funded by the Biotechnology and Biological Sciences Research Council (BBSRC).

6 References

1. Mons B, van Haagen H, Chichester C, Hoen P-B 't, den Dunnen JT, van Ommen G, et al. The value of data. Nat Genet. 2011;43:281–3.
2. The data deluge [Internet]. The Economist; 2010. Available from: https://www.economist.com/node/15579717
3. Wilkinson MD, Dumontier M, Aalbersberg IjJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific data. 2016;3.
4. Ruttenberg A, Clark T, Bug W, Samwald M, Bodenreider O, Chen H, et al. Advancing translational research with the Semantic Web. BMC Bioinformatics. 2007;8:S2.

5. Wang X, Gorlitsky R, Almeida JS. From XML to RDF: how semantic web technologies will change the design of 'omic' standards. Nature Biotechnology. 2005;23:1099–103.
6. Belleau F, Nolin M-A, Tourigny N, Rigault P, Morissette J. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. Journal of Biomedical Informatics. 2008;41:706–16.
7. Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology. 2007;25:1251–5.
8. Jupp S, Malone J, Bolleman J, Brandizi M, Davies M, Garcia L, et al. The EBI RDF platform: linked open data for the life sciences. Bioinformatics. 2014;30:1338–9.
9. Staab S, Studer R, editors. Handbook on ontologies. 2. ed. Berlin: Springer; 2009.
10. Baader F, Horrocks I, Sattler U. Chapter 3 Description Logics. In: van Harmelen F, Lifschitz V, Porter B, editors. Foundations of Artificial Intelligence [Internet]. Elsevier; 2008 [cited 2018 Mar 7]. p. 135–79. Available from: http://www.sciencedirect.com/science/article/pii/S1574652607030039
11. James Malone, ICBO 2017 keynote [Internet]. Available from: https://www.slideshare.net/JamesMalone5/malone-icbo2017-keynote
12. Cai L, Zhu Y. The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. Data Science Journal [Internet]. 2015 [cited 2018 Mar 7];14. Available from: http://datascience.codata.org/articles/10.5334/dsj-2015-002/
13. Ehrlinger L, Wöß W. Towards a Definition of Knowledge Graphs. SEMANTiCS (Posters, Demos, SuCCESS). 2016.
14. Jesse W, Paul T. Facebook Linked Data via the Graph API. Semantic Web. 2013;245–250.
15. Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, et al. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. ACM Press; 2014 [cited 2018 Feb 22]. p. 601–10. Available from: http://dl.acm.org/citation.cfm?doid=2623330.2623623
16. Jing Han, Haihong E, Guan Le, Jian Du. Survey on NoSQL database. 2011 6th International Conference on Pervasive Computing and Applications [Internet]. Port Elizabeth, South Africa: IEEE; 2011 [cited 2018 Sep 26]. p. 363–6. Available from: http://ieeexplore.ieee.org/document/6106531/
17. Hecht R, Jablonski S. NoSQL evaluation: A use case oriented survey. 2011 International Conference on Cloud and Service Computing [Internet]. Hong Kong, China: IEEE; 2011 [cited 2018 Sep 26]. p. 336–41. Available from: http://ieeexplore.ieee.org/document/6138544/
18. Robinson I, Webber J, Eifrem E. Graph databases. Second edition. Beijing: O'Reilly; 2015.
19. Buerli M. The Current State of Graph Databases. 2012. Available from: https://www.semanticscholar.org/paper/The-Current-State-of-Graph-Databases-Buerli/5b5b6b80badccd291e3437460222e24326c65979?tab=abstract
20. Angles R. A Comparison of Current Graph Database Models. 2012 IEEE 28th International Conference on Data Engineering Workshops [Internet]. Arlington, VA, USA: IEEE; 2012 [cited 2018 Sep 26]. p. 171–7. Available from: http://ieeexplore.ieee.org/document/6313676/
21. Malhotra M, Nair TR. Evolution of Knowledge Representation and Retrieval Techniques. International Journal of Intelligent Systems and Applications. 2015;7:18–28.
22. Defining N-ary Relations on the Semantic Web [Internet]. Available from: https://www.w3.org/TR/swbp-n-aryRelations/
23. Holzschuher F, Peinl R. Querying a graph database – language selection and performance considerations. Journal of Computer and System Sciences. 2016;82:45–68.
24. Hassani-Pak K. KnetMiner - An integrated data platform for gene mining and biological knowledge discovery [PhD Thesis]. Universität Bielefeld; 2017.
25. Hassani-Pak K, Rawlings C. Knowledge discovery in biological databases for revealing candidate genes linked to complex phenotypes. Journal of integrative bioinformatics. 2017;14.

26. O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, McVeigh R, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Res. 2016;44:D733-745.
27. Mueller LA. AraCyc: A Biochemical Pathway Database for Arabidopsis. PLANT PHYSIOLOGY. 2003;132:453–60.
28. Cooper L, Walls RL, Elser J, Gandolfo MA, Stevenson DW, Smith B, et al. The Plant Ontology as a Tool for Comparative Plant Anatomy and Genomic Analyses. Plant and Cell Physiology. 2013;54:e1–e1.
29. Brandizi M, Singh A, Rawlings C, Hassani-Pak K. Towards FAIRer Biological Knowledge Networks Using a Hybrid Linked Data and Graph Database Approach. Journal of Integrative Bioinformatics [Internet]. 2018 [cited 2018 Sep 26];15. Available from: http://www.degruyter.com/view/j/jib.2018.15.issue-3/jib-2018-0023/jib-2018-0023.xml
30. Marco Brandizi. A Preliminary survey of RDF/Neo4j as backends for KnetMiner [Internet]. 2017. Available from: https://www.slideshare.net/mbrandizi/a-preliminary-survey-of-rdfneo4j-as-backends-for-knetminer
31. Menzel C. Reference Ontologies – Application Ontologies: Either/Or or Both/And? Proceedings of the KI2003 Workshop on Reference Ontologies and Application Ontologies [Internet]. Hamburg, Germany; 2003. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.2490
32. Erling O, Mikhailov I. RDF Support in the Virtuoso DBMS. In: Pellegrini T, Auer S, Tochtermann K, Schaffert S, editors. Networked Knowledge - Networked Media [Internet]. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009 [cited 2018 Sep 26]. p. 7–24. Available from: http://link.springer.com/10.1007/978-3-642-02184-8_2
33. Vukotic A. Neo4j in action. Shelter Island, NY: Manning Publications Co; 2015.
34. rdf2neo Tool, Code Repository [Internet]. 2018 [cited 2018 Sep 26]. Available from: https://github.com/Rothamsted/rdf2neo
35. Spring Beans Core Technologies [Internet]. Available from: https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html
36. Loizou A, Angles R, Groth P. On the formulation of performant SPARQL queries. Web Semantics: Science, Services and Agents on the World Wide Web. 2015;31:1–26.
37. 7 Steps to Fast SPARQL Queries - Stardog [Internet]. Available from: https://www.stardog.com/blog/7-steps-to-fast-sparql-queries/
38. Benchmarking kNetMiner data, Neo4j vs Virtuoso [Internet]. Available from: https://github.com/Rothamsted/graphdb-benchmarks/blob/master/README.md
39. Davidson P. Designing URI sets for the UK public sector [Internet]. Chief Technology Officer Council; 2009. Available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/60975/designing-URI-sets-uk-public-sector.pdf
40. Tarql: SPARQL for Tables [Internet]. Available from: http://tarql.github.io/
41. openCypher/CIP2016-06-22-nested-updating-and-chained-subqueries.adoc at CIP-nested-subqueries · petraselmer/openCypher [Internet]. Available from: https://github.com/petraselmer/openCypher/blob/CIP-nested-subqueries/cip/1.accepted/CIP2016-06-22-nested-updating-and-chained-subqueries.adoc
42. Neo4j : Post-UNION processing - Neo4j Graph Database Platform [Internet]. Available from: https://neo4j.com/developer/kb/post-union-processing/
43. Wu H, Fujiwara T, Yamamoto Y, Bolleman J, Yamaguchi A. BioBenchmark Toyama 2012: an evaluation of the performance of triple stores on biological data. Journal of Biomedical Semantics. 2014;5:32.
44. Guo Y, Pan Z, Heflin J. LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web. 2005;3:158–82.
45. Gubichev A, Then M. Graph Pattern Matching: Do We Have to Reinvent the Wheel? Proceedings of Workshop on GRAph Data management Experiences and Systems - GRADES'14 [Internet]. Snowbird, UT, USA: ACM Press; 2014 [cited 2018 Sep 27]. p. 1–7. Available from: http://dl.acm.org/citation.cfm?doid=2621934.2621944

46. Pobiedina N, Rümmele S, Skritek S, Werthner H. Benchmarking Database Systems for Graph Pattern Matching. In: Decker H, Lhotská L, Link S, Spies M, Wagner RR, editors. Database and Expert Systems Applications [Internet]. Cham: Springer International Publishing; 2014 [cited 2018 Sep 27]. p. 226–41. Available from: http://link.springer.com/10.1007/978-3-319-10073-9_18
47. Liebig T, Vialard V, Opitz M, Metzl S. GraphScale: Adding Expressive Reasoning to Semantic Data Stores. International Semantic Web Conference (Posters & Demos). 2015.
48. J.Barrasa. Importing RDF data into Neo4j [Internet]. Jesús Barrasa. 2016 [cited 2018 Mar 9]. Available from: https://jesusbarrasa.wordpress.com/2016/06/07/importing-rdf-data-into-neo4j/
49. Alocci D, Mariethoz J, Horlacher O, Bolleman JT, Campbell MP, Lisacek F. Property Graph vs RDF Triple Store: A Comparison on Glycan Substructure Search. Helmer-Citterich M, editor. PLOS ONE. 2015;10:e0144578.
50. Lysenko A, Roznovăţ IA, Saqi M, Mazein A, Rawlings CJ, Auffray C. Representing and querying disease networks using graph databases. BioData Min [Internet]. 2016 [cited 2018 Mar 15];9. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4960687/
51. Pareja-Tobes P, Tobes R, Manrique M, Pareja E, Pareja-Tobes E. Bio4j: a high-performance cloud-enabled graph-based data platform. 2015 [cited 2018 Oct 31]; Available from: http://biorxiv.org/lookup/doi/10.1101/016758
52. Jupp S, Burdett T, Leroy C, Parkinson HE. A new Ontology Lookup Service at EMBL-EBI. SWAT4LS. 2015. p. 118–119.
53. Thakkar H, Punjani D, Lehmann J, Auer S. Killing Two Birds with One Stone–Querying Property Graphs using SPARQL via GREMLINATOR. arXiv preprint arXiv:180109556. 2018;
54. Pham M-D, Passing L, Erling O, Boncz P. Deriving an Emergent Relational Schema from RDF Data. Proceedings of the 24th International Conference on World Wide Web - WWW '15 [Internet]. Florence, Italy: ACM Press; 2015 [cited 2018 Oct 31]. p. 864–74. Available from: http://dl.acm.org/citation.cfm?doid=2736277.2741121
55. Virgilio RD. Smart RDF Data Storage in Graph Databases. 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) [Internet]. Madrid, Spain: IEEE; 2017 [cited 2018 Oct 31]. p. 872–81. Available from: http://ieeexplore.ieee.org/document/7973793/
56. Gray AJ, Goble C, Jimenez RC. Bioschemas: From Potato Salad to Protein Annotation. 2017.
57. Rodriguez MA. The Gremlin Graph Traversal Machine and Language (Invited Talk). Proceedings of the 15th Symposium on Database Programming Languages [Internet]. New York, NY, USA: ACM; 2015. p. 1–10. Available from: http://doi.acm.org/10.1145/2815072.2815073