Declarative Graph Querying in Practice and Theory

Tutorial Abstract

George H. L. Fletcher
Technische Universiteit Eindhoven
g.h.l.fletcher@tue.nl

Hannes Voigt
Technische Universität Dresden
hannes.voigt@tu-dresden.de

Nikolay Yakovets
Technische Universiteit Eindhoven
hush@tue.nl

ABSTRACT

With the recent resurgence of interest in graph data management, there has been a flurry of research on the design and engineering of graph query languages. On the design side, there is a large body of theoretical results that have been obtained regarding graph languages. On the engineering side, many sophisticated scalable solutions for graph query processing have been developed and put into practice. While both areas are focusing on the study of graph query languages, there has been relatively little work bridging the results on both sides. This tutorial will survey the state of the art in this landscape with a particular focus on uncovering and highlighting indicative research issues that are ripe for collaboration and cross-fertilization between the engineering and theoretical studies of graph database systems.

1. MOTIVATION

The mathematical concept of a graph is somewhat a rediscovered old friend in the database community. Predating relational database systems, the CODASYL network data model resembles essentially graph data. In the 1980s and early 1990s, with the rise of object-oriented programming and advent of object-oriented database systems, research considered graph-based data models and graph query languages [2]. With the continued dominance of relational DBMSs, none of these efforts got any sustainable traction in industry. In the last decade, however, the graph concept has a considerable revival with three major trends driving it.

The first driver is the Semantic Web movement [8]. The idea of the semantic web gave rise to the RDF [38] data model, which structures data as a labeled graph. This propelled the publication and maintenance of thousands of open RDF datasets on the internet, most famously DBpedia [3]. It also sparked research in every corner of the database community – ranging from works investigating the fundamental properties of query languages for labeled graphs to the design of storage structures and query engines for RDF data.

The second driver is agility with respect to the management of data. New application domains (e.g. [16, 39]) as well as novel development methods [7] increased the demand for data models that are less rigid and schema-oriented but more ad-hoc and data-oriented. Graph data models typically excel in this regard as new nodes and edges can be added anytime, regardless of their properties. This propelled the proliferation of the Property Graph model and corresponding DBMSs, such as Neo4j[1] and Apache TinkerPop Blueprints[2] implementations. By now also major DBMS vendors such as IBM and Oracle have put their weight behind the Property Graph model and are developing Property Graph-based data management solutions.

The third driver is a shift in interest of analytics from merely reporting towards data-intensive science and discovery [17]. One major method in this discipline is network analysis, which puts the focal point of interest on the connectivity of entities. The toolbox of network analysis offers a rich set of algorithms and measures. These tools give incentives to consider the graph structure of data collections in a wide range of application fields, further increasing the demand for scalable graph data management solutions.

Today, graph data management has become a major topic in the database community, in research as well as industry. There are several new challenges of graph data management which fundamentally distinguish it from tabular or nested (XML, JSON) data. An exemplification of how much traction graph data management has gained is the Linked Data Benchmark Council (LDBC).[3] In LDBC, research and industry are jointly developing standardized benchmarks for graph data management workloads to accelerate the maturing of graph management systems by increasing competition.

A rather new LDBC initiative is the Graph Query Language Standardization Task Force. The query language is one of the most crucial elements of a DBMS. It defines the functionality of a DBMS and how it is exposed to the user. At the same time, it sets the tone for the DBMS implementation by requiring certain functionality. Establishing a standardized graph query language, such as SQL for relational systems, is the next step towards more competition and progress. The task force brings together a group of researchers from engineering and theory as well as developers and representatives from industry.

One early lesson learned in the task force is that there exist two disparate bodies of work surrounding graph query lan-

[1]http://neo4j.com/
[2]http://tinkerpop.apache.org/
[3]http://ldbcouncil.org/

guages. One body of work focuses on the foundational issues arising in the design of graph query languages, their functionality, semantics, and formal properties such as decidability, query complexity, and containment. The other body of work focuses primarily on the engineering of systems, considering the design of storage and indexing solutions and scalable query processing engines for graph data. Due to this divide in research, it is clear that we will not get the best systems possible with the knowledge available. With both sides rather oblivious to more recent advances of the other, particularly challenges at their intersection often remain untouched.

This tutorial aims at uncovering and highlighting indicative research issues that are ripe for collaboration and cross-fertilization. In the core fields of design of declarative query languages and query processing, we will give an overview of recent advances. We will also point out their rich connections and new research challenges which arise from bringing theory and engineering together. Overall, we aim to motivate and stimulate such bridges, towards a broader coherent understanding and further improvements in the design and engineering of graph database systems.

2. SCOPE OF THE TUTORIAL

Audience. The tutorial is relevant for EDBT as well as ICDT attendees. The intended audience of the tutorial includes:

- Researchers interested in novel open challenges in graph data management or who are particularly interested in collaboration and cross-fertilization between theory and practice and want to have a kick start.
- Professionals that work in graph database system engineering and graph query language design and are interested in foundational background and broadening their scope of interesting query features.

Apart from basic knowledge about graph and database concepts there are no special requirements for this tutorial.

Coverage. The attendees of this tutorial will take home:

(i) an overview of the landscape of declarative graph query languages covering the most important features with their different functionality and properties from a practical as well as theoretical standpoint;
(ii) a survey of the foundations and recent advances accomplished by engineering and theory in graph query processing and optimization; and,
(iii) insights into open challenges for both foundational and engineering work and in particular for research topics at the intersection of both.

Scope. We scope the tutorial to core topics in graph query language design and processing. We look at the selected topics from a data management perspective, i.e., the focus is on concepts and techniques relevant to the engineering of graph data management system with a declarative query language interface. With this specific focus, there is already a wealth of results and open research challenges.

In particular, we will not discuss the design of streaming, distributed, federated, or parallel processing solutions. We will also not cover analytical topics such as graph search, graph clustering, graph pattern mining, etc. It would be impractical to also cover these topics in a focused 3-hour tutorial. Furthermore, there have been excellent tutorials on these topics recently (e.g., [19, 20, 21, 42]).

3. TUTORIAL OUTLINE

After a short illustrative introduction to the distinctive properties of graph data and graph queries, the tutorial will cover two core research areas in graph query languages: language design and query processing including data representation and query optimization aspects. Within each area, we will present the current state of the art in both theory and engineering and discuss important bridges between the bodies of work in these areas.

3.1 Graph query languages

Advances in theory. As indicated already, there is a long history of the study of graph query languages. The theoretical study of graph query languages (expressive power, evaluation complexity) has advanced ahead of the engineering of graph databases, e.g., the study of regular path queries since the 1980s. We will give a systematic presentation of the current design space of graph query languages in the theory community, including a historical perspective on this development. Major languages here include subgraph matching queries, path algebras, regular path queries, and reachability [4, 6, 10, 12, 13, 24, 34, 35, 40, 43].

Advances in engineering. With the recent proliferation of graph database system such as Neo4j, Virtuoso[4], and many others in industry and open source communities, there is a zoo of graph query languages available today. All of them offer some flavor of subgraph matching and reachability querying functionalities. We will give a structured overview of the major players in the field such as SPARQL 1.1, openCypher[5], declarative pattern matching in Gremlin, and PGQL [37] and point out their main functional and distinctive features. A look at the LDBC benchmark [11] queries will complement this to a summary of the functional features available and required from a practical, use case-driven standpoint.

Challenges. By contrasting the theoretical design space with practical query languages and use cases, we point out certain matches and mismatches, that give opportunities for knowledge transfer or give rise to new research challenges. Recently, for instance, practical query languages such as SPARQL 1.1 and openCypher have introduced support for regular path queries, which are very well studied in theory, while there is much room for aligning practical languages with this literature. Another area where many open research challenges remain is in aligning recent engineering efforts centered around the Property Graph model, on the one hand, with theoretical results applying mainly to labeled graphs, on the other. In particular, the impact which operations on graph properties might have on fundamental language properties must be considered. Finally, practical querying languages demand for functionalities is not considered in-depth from a theory perspective yet, such as aggregation queries, top-k queries, or diversity in path queries. Other aspects we will cover are: closedness/composability versus views; and, path logics versus traversal DSLs.

[4]http://virtuoso.openlinksw.com/
[5]http://www.opencypher.org/

3.2 Graph query processing

Advances in theory. The complexity of static query analysis (query containment, equivalence) is well understood for variations of graph path queries [9, 22, 34]. We will provide tutorial participants with an overview of the fundamental results for the languages surveyed in Section 3.1, with a particular focus on the demarcation between decidable and undecidable extensions of regular path queries. Tractable language-independent characterizations of graph query languages have been established in terms of the structure of a given graph instance. These characterizations are the basis for index data structures for path query evaluation/acceleration. We will provide an overview of recent advances in the theory of structural indexing and compression methods for graph data and their formal connections to the graph query languages [15, 25]. We will also discuss recent advances in the theory of worst-case optimal joins algorithms, as applied to graph query processing [1, 29].

Advances in engineering. In query processing, algorithms and graph representation go hand in hand. Node and edge tables, compressed sparse row format, and triple tables are the most common techniques used for primary graph data representation. Recent works considered various refinements of these techniques to increase efficiency by compression [1, 28], partitioning [31], triple indexing [23, 28, 44], and path indexing [14, 36]. Other advances concern the updatability of the data structures used [26, 28, 44]. Within the tutorial, we will give a crisp intro into the common base techniques and provide an overview of main ideas of the refinements. On the algorithm side, we will concentrate on advances in join processing for graph queries, since these advances are relevant for many of the query classes from the design space. In the tutorial, we will cover automata [41] and two-way join-based approaches [23, 26, 28, 33] as well as approaches that improve the utilization of high combined selectivities in graph queries, such as sideways information passing [27] and worst-case optimal n-way joins [1, 30]. We also highlight current challenges in scalability and efficiency [5].

Challenges. Research on worst-case optimal joins actually stretches from theory to engineering and excellently exemplifies the benefits of bridging both realms. We will use this example to illustrate to the tutorial participants how bridging effort can result in coherent understanding and advanced solutions. With this motivation in place, we point out further bridging challenges. For instance, while structural indexing is a very promising method from theory it has not been echoed much in system implementation. In engineering, though, updatability is an important concern, which theory is challenged to give more consideration. Further bridging challenges we will point out are n-way joins with multiset semantics and algorithmic applications of static query analysis.

3.3 Looking ahead

We will round out the tutorial with a discussion of promising research advances which have not yet bridged the gap between the theory and engineering of graph query languages. These include topics such as the decidability, complexity, and containment of graph query languages involving node and edge creation [18, 32], features which are particularly appropriate in the context of the Property Graph data model. These developments have good potential for research impact in the intersection of engineering and theoretical investigations of graph query languages.

The tutorial will conclude with a recap of the major areas that we see for collaboration and cross-fertilization between engineering and theory.

4. BIOGRAPHY OF THE PRESENTERS

George Fletcher is an associate professor of computer science at Eindhoven University of Technology. He obtained his PhD from Indiana University Bloomington in 2007. His research interests span query language design and engineering, foundations of databases, and data integration. His current focus is on management of massive graphs such as social networks and linked open data. He was a co-organizer of the EDBT Summer School on Graph Data Management (2015) and is currently a member of the LDBC Graph Query Language Standardization Task Force.

Hannes Voigt is a post-doctoral researcher at the Dresden Database Systems Group, Technische Universität Dresden and obtained his PhD from the same university in 2014. He worked on various database topics such as physical design, management of schema-flexible data, and self-adapting indexes. From 2010 to 2011, he worked at SAP Labs, Palo Alto contributing to a predecessor of SAP HANA Graph Project. His current research focuses on database evolution and versioning, declarative graph query languages and efficient graph processing on NUMA in-memory storage systems. He is also member of the LDBC Graph Query Language Standardization Task Force.

Nikolay Yakovets is an assistant professor of computer science at Eindhoven University of Technology. He obtained his PhD from Lassonde School of Engineering at York University in 2016. He worked on various database topics at IBM CAS Canada and Empress Software Canada. His current focus is on design and implementation of core database technologies, management of massive graph data, and efficient processing of queries on graphs.

5. REFERENCES

[1] C. R. Aberger, A. Nötzli, K. Olukotun, and C. Ré. EmptyHeaded: Boolean Algebra Based Graph Processing. *CoRR*, abs/1503.02368, Mar. 2015.

[2] R. Angles and C. Gutiérrez. Survey of Graph Database Models. *ACM Computing Surveys*, 40(1), 2008.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC/ASWC*, pages 722–735, Busan, Korea, 2007.

[4] P. B. Baeza. Querying graph databases. In *PODS*, pages 175–188, New York, NY, 2013.

[5] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat. gMark: Schema-driven generation of graphs and queries. *IEEE Trans. Knowl. Data Eng.*, in press, 2017.

[6] P. Barceló, G. Fontaine, and A. W. Lin. Expressive path queries on graphs with data. In *LPAR*, pages 71–85, Stellenbosch, South Africa, 2013.

[7] K. Beck et al. Manifesto for Agile Software Development. http://agilemanifesto.org/, 2001.

[8] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Sci. Amer.*, pages 34–43, May 2001.

[9] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

[10] M. P. Consens and A. O. Mendelzon. Graphlog: a visual formalism for real life recursion. In *PODS*, pages 404–416, Nashville, 1990.

[11] O. Erling, A. Averbuch, J. Larriba-Pey, H. Chafi, A. Gubichev, A. Prat-Pérez, M. Pham, and P. A. Boncz. The LDBC Social Network Benchmark: Interactive Workload. In *SIGMOD*, pages 619–630, 2015.

[12] D. Figueira and L. Libkin. Path logics for querying graphs: Combining expressiveness and efficiency. In *LICS*, pages 329–340, Kyoto, 2015.

[13] G. H. L. Fletcher, M. Gyssens, D. Leinders, D. Surinx, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs. *Inf. Sci.*, 298:390–406, 2015.

[14] G. H. L. Fletcher, J. Peters, and A. Poulovassilis. Efficient regular path query evaluation using path indexes. In *EDBT*, pages 636–639, Bordeaux, 2016.

[15] G. H. L. Fletcher et al. Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations. *J. Log. Comput.*, 25(3):549–580, 2015.

[16] M. J. Franklin, A. Y. Halevy, and D. Maier. From Databases to Dataspaces: A New Abstraction for Information Management. *SIGMOD Record*, 34(4):27–33, 2005.

[17] T. Hey, S. Tansley, and K. M. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, 2009.

[18] J. Hidders. *A graph-based update language for object-oriented data models.* PhD thesis, Eindhoven University of Technology, 2001.

[19] Y. Ke, J. Cheng, and J. X. Yu. Querying large graph databases. In *DASFAA*, pages 487–488, 2010.

[20] A. Khan and S. Elnikety. Systems for big-graphs. *PVLDB*, 7(13):1709–1710, 2014.

[21] A. Khan, Y. Wu, and X. Yan. Emerging graph queries in linked data. In *ICDE*, pages 1218–1221, 2012.

[22] E. V. Kostylev, J. L. Reutter, and D. Vrgoc. Containment of data graph queries. In *ICDT*, pages 131–142, Athens, Greece, 2014.

[23] Y. Luo, F. Picalausa, G. H. L. Fletcher, J. Hidders, and S. Vansummeren. Storing and indexing massive RDF datasets. In R. De Virgilio, F. Guerra, and Y. Velegrakis, editors, *Semantic Search over the Web*, pages 31–60. Springer Berlin Heidelberg, 2012.

[24] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo. Strong simulation: Capturing topology in graph pattern matching. *ACM Trans. Database Syst.*, 39(1):4, 2014.

[25] S. Maneth and F. Peternek. A survey on methods and systems for graph compression. *CoRR*, abs/1504.00616, 2015.

[26] B. Motik, Y. Nenov, R. Piro, I. Horrocks, and D. Olteanu. Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems. In *AAAI*, pages 129–137, 2014.

[27] T. Neumann and G. Weikum. Scalable join processing on very large RDF graphs. In *SIGMOD*, pages 627–640. ACM, 2009.

[28] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113, 2010.

[29] H. Q. Ngo, C. Ré, and A. Rudra. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Record*, 42(4):5–16, 2013.

[30] D. T. Nguyen, M. Aref, M. Bravenboer, G. Kollias, H. Q. Ngo, C. Ré, and A. Rudra. Join Processing for Graph Patterns: An Old Dog with New Tricks. In *GRADES*, pages 2:1–2:8, 2015.

[31] M. Paradies, W. Lehner, and C. Bornhövd. GRAPHITE: An Extensible Graph Traversal Framework for Relational Database Management Systems. In *SSDBM*, pages 29:1–29:12. ACM, 2015.

[32] J. Paredaens, P. Peelman, and L. Tanca. G-Log: A graph-based query language. *IEEE Trans. Knowl. Data Eng.*, 7(3):436–453, 1995.

[33] R. Raman, O. van Rest, S. Hong, Z. Wu, H. Chafi, and J. Banerjee. PGX.ISO: Parallel and Efficient In-Memory Engine for Subgraph Isomorphism. In *GRADES*, pages 1–6. ACM, 2014.

[34] J. L. Reutter, M. Romero, and M. Y. Vardi. Regular queries on graph databases. In *ICDT*, pages 177–194, Brussels, 2015.

[35] S. Santini. Regular languages with variables on graphs. *Inf. Comput.*, 211:1–28, 2012.

[36] J. Sumrall, G. H. L. Fletcher, A. Poulovassilis, J. Svensson, M. Vejlstrup, C. Vest, and J. Webber. Investigations on path indexing for graph databases. In *PELGA*, Grenoble, 2016.

[37] O. van Rest, S. Hong, J. Kim, X. Meng, and H. Chafi. PGQL: a property graph query language. In *GRADES*, Redwood Shores, CA, 2016.

[38] W3C. RDF 1.1 Concepts and Abstract Syntax. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, Feb. 2014.

[39] H. Werner, C. Bornhövd, R. Kubis, and H. Voigt. MOAW: An Agile Visual Modeling and Exploration Tool for Irregularly Structured Data. In *BTW*, pages 742–745, 2011.

[40] P. T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, 2012.

[41] N. Yakovets, P. Godfrey, and J. Gryz. Query planning for evaluating SPARQL property paths. In *SIGMOD*, pages 1875–1889, San Francisco, 2016.

[42] D. Yan, Y. Bu, Y. Tian, A. Deshpande, and J. Cheng. Big Graph Analytics Systems. In *SIGMOD*, pages 2241–2243, 2016.

[43] J. Yu and J. Cheng. Graph reachability queries: A survey. In C. C. Aggarwal and H. Wang, editors, *Managing and Mining Graph Data*, Advances in Database Systems, pages 181–215. Springer US, 2010.

[44] L. Zou, M. T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao. gstore: a graph-based SPARQL query engine. *VLDB J.*, 23(4):565–590, 2014.