Profiling Graphs: Order from Chaos

Aidan Hogan
Institute for the Foundations of Data
DCC, Universidad de Chile
ahogan@dcc.uchile.cl

ABSTRACT

Graphs are being increasingly adopted as a flexible data model in scenarios (e.g., Google's Knowledge Graph, Facebook's Graph API, Wikidata, etc.) where multiple editors are involved in content creation, where the schema is ever changing, where data are incomplete, where the connectivity of resources plays a key role—scenarios where relational models traditionally struggle. But with this flexibility comes a conceptual cost: it can be difficult to summarise and understand, at a high level, the content that a given graph contains. Hence profiling graphs becomes of increasing importance to extract order, a posteriori, from the chaotic processes by which such graphs are generated. This talk will motivate the use of graphs as a data model, abstract recent trends in graph data management, and then turn to the issue of profiling and summarising graphs: what are the goals of such profiling, the principles by which graphs can be summarised, the main techniques by which this can/could be achieved? The talk will emphasise the importance of profiling graphs while highlighting a variety of open research questions yet to be tackled.

CCS CONCEPTS

• **Information systems → Graph-based database models**;

KEYWORDS

graphs, profiling, schema

ACM Reference Format:
Aidan Hogan. 2018. Profiling Graphs: Order from Chaos. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3184558.3191647

1 WHY GRAPHS?

The relational model has proven rather useful for managing data in digital form. Order is imposed in this model by the presence of the relational schema. With the resulting order, one can avail of a number of benefits, including integrity constraints, access control primitives, indexing schemes, query optimisation techniques, transactions, ..., not to mention a detailed blueprint of what data may or may not be contained in the model. To avail of this order – and all of its benefits – one simply has to answer, up front, some basic questions about the data domain: What types of entities will be described? What are the relations and their multiplicities? What are the attributes and what are the functional dependencies?

Achieving agreement on such questions is straightforward – if not at least worthwhile – in the context of, for example, a bank or a hospital. However, in situations where the domain is more open – where potentially reality itself could be modelled [8] – arriving at dependable *a priori* answers to such questions becomes a lot more difficult. While it may seem safe, for example, to assert in a schema that MAYOR is a relation between a PERSON and a PLACE, the people of Sunol, California could, at some unspecified point in the future, take exception to that definition having had a DOG ("Bosco") as MAYOR for over a decade. To any sufficiently-specific *a priori* schema, reality is sure to hold difficult exceptions.

In settings where the domain is more open, the growing trend is thus moving away from relational models and towards more flexible alternatives; one such alternative is that of the graph model. The idea of structuring data as directed labelled graphs has been around for at least as long as the present author [6]. While more exotic flavours of graph data models have been proposed since then – including graphs where nodes can themselves be graphs (*hypernodes*), or where edges can connect any number of nodes (*hyperedges*), or where edges can be labelled with attribute–value pairs (*property graphs*) [1, 2] – the core ideas remain the same: graphs offer a natural way to represent (and query) the connections between elements of the data, and offer a more flexible alternative to, e.g., the rigid relational model governed by a relational schema.

Such characteristics of graph models have become increasingly valued for environments where data are incomplete and/or where the schema remains fluid—a natural example being scenarios involving management of Web data. While the Semantic Web community has long championed graph data models through the foundational RDF standard, the recent hype around industry-driven initiatives – such as Google's Knowledge *Graph*, Facebook's *Graph* API or the Open *Graph* Protocol – indicate that graph data models are becoming more and more mainstream: viewing (and querying) your data through the lens of a graph is no longer necessarily seen as an act of relational heresy, but rather something that could be considered natural when dealing with diverse "semi-structured" data.

2 WHY PROFILE GRAPHS?

While graphs offer a natural way to model and query incomplete data with a fluid schema, their use comes at a cost. The relational schema plays a key role in traditional data management scenarios but it has no direct analogue in the world of graphs. When a user first wishes to query a relational database, to understand what content it contains, they might first ask for the list of tables or some other description of the schema; what should they ask for if they wish to query a graph? The lack of such a schema also

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191647

complicates various data management operations, such as indexing, query optimisation, transactions, validation, and so forth.

The question then is: can we have the best of both worlds? Can we maintain the flexibility that graph models provide *and* the benefits of having a relational schema to impose order on the data? The simple answer is probably "no", at least in the general case: if we lose some measure of order and regularity in our data model, then we lose the benefits that would have come with it for data management. We cannot have our schema and break it too.

Perhaps for this reason, a lot of focus in terms of structuring graphs has been on defining "upfront schemas". For example, in the Semantic Web community, the RDF Schema [3] standard allows to define the semantics of terms that can be used in the graph, such as to state that MAYOROF has the domain HUMAN; however, such a schema does not help to understand the legacy data of a graph, which may or may not use the terms defined in RDFS. Another standard more recently proposed along these lines is the SHACL language [5], which allows for specifying some constraints that an RDF graph should follow; however, this standard aims to enforce order upfront and again does not help to understand legacy data.

But even without an explicit upfront schema, most real-world graphs do have an inherent order. Most mayors are still, after all, human. But to exploit this order, we need methods to analyse and distil that order from graph-structured datasets. That order can then be used to help users formulate queries, to help summarise the content of a graph for the purposes of data retrieval or federation, to understand the processes by which the graphs evolve and change, as well as to enable or otherwise optimise low level data-management operations, such as query optimisation, compression, indexing, quality and completeness assessment, and so forth. Distilling such order from graphs is (often) known as "*profiling*" [4], and may involve analysing and describing a graph along a variety of dimensions for a variety of applications.

3 HOW SHOULD WE PROFILE GRAPHS?

While there has been quite a lot of work proposed to profile graphs, much of this work has been as heterogeneous as the graphs that the works aim to profile [4]: different techniques have been proposed to study the dynamics of graphs, or the quality of graphs, or the completeness of graphs, or to index graphs, or summarise them, or integrate them. This talk will argue that all of these aspects of profiling graphs are inherently related and can all benefit from better ways to analyse and summarise the structure of graphs from a more theoretical perspective and with a more general approach.

Along these lines, we argue that the area of graph profiling could greatly benefit from – and should actively seek – an appropriate notion of data-driven schema for graphs: something that plays the role of the relational schema but is extracted from the data rather than being imposed upfront. Thereafter, many profiling tasks could be done not over the "raw" graphs, but rather at the schema level.

4 A GENERAL SCHEMA FOR GRAPHS?

While this idea sounds good in principle, in practice, there is no unique natural notion of schema for graphs—natural in the sense, for example, of the relational schema for tabular data. Rather there are a wide variety of proposed methods to summarise and extract high-level structures from graphs [7]. Many such proposals, however, are based on common principles: for example, if one assumes that a schema for graphs should be connected (as a graph), then creating such a schema must involve grouping sets of nodes into a single node and/or grouping sets of edges into a single edge, be it through a direct equivalence relation, or some clustering or community detection method. A problem we face in this line of research is not a lack of possibilities for graph schemas, but rather the opposite: how can we decide between them for the purposes of profiling?

To present some ideas along these lines, we give some example desiderata for a data-driven graph schema:

- **Scalability:** Given that some knowledge graphs are in the order of millions of nodes and edges, a suitable notion of schema should be computable from graphs of that size.
- **Stability:** A minor change in the underlying graph should not be able to effect a major change in the corresponding schema.
- **Conciseness:** The schema should be significantly smaller than the graph that it describes.
- **Connectivity:** The schema should not simply describe the nodes of the graph, but should capture information on how the graph is connected.
- **Readability:** The schema should be human-interpretable, meaning that its structure can be directly understood rather than representing abstract objects without direct significance.

In this talk, we will then use this list of desiderata to guide a discussion of possible research directions towards defining what a general notion of graph schema could look like (if such a holy grail exists) and how we could define benchmarks for such.[1]

Acknowledgements. *This work was supported by Fondecyt Grant No. 1181896.*

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.* 50, 5 (2017), 68:1–68:40. DOI: https://doi.org/10.1145/3104031

[2] Renzo Angles and Claudio Gutiérrez. 2008. Survey of graph database models. *ACM Comput. Surv.* 40, 1 (2008), 1:1–1:39. DOI: https://doi.org/10.1145/1322432.1322433

[3] Dan Brickley, R.V. Guha, and Brian McBride. 2014. RDF Schema 1.1. W3C Recommendation. (25 Feb. 2014). http://www.w3.org/TR/rdf-schema/.

[4] Mohamed Ben Ellefi, Zohra Bellahsene, John G. Breslin, Elena Demidova, Stefan Dietze, Julian Szymanski, and Konstantin Todorov. RDF Dataset Profiling - a Survey of Features, Methods, Vocabularies and Applications. *Semantic Web Journal* (????). To appear.

[5] Holger Knublauch and Dimitris Kontokostas. 2014. Shapes Constraint Language (SHACL). W3C Working Group Note. (24 June 2014). http://www.w3.org/TR/rdf11-primer/.

[6] Gabriel M. Kuper and Moshe Y. Vardi. 1984. A New Approach to Database Logic. In *ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*. 86–96. DOI: https://doi.org/10.1145/588011.588026

[7] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *CoRR* abs/1612.04883 (2018). arXiv:1612.04883 http://arxiv.org/abs/1612.04883

[8] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[1] Of course, this list of desiderata is far from complete. We may also consider more application-oriented requirements; e.g., it may be useful for schemas to be composable, such that given the schemas of two independent graphs, it should be feasible to compute the schema of the union of the two graphs.