Multilayer graphs: A unified data model for graph databases

Renzo Angles
DCC, Universidad de Talca & IMFD
Chile
rangles@utalca.cl

Aidan Hogan
DCC, University of Chile & IMFD
Chile
ahogan@dcc.uchile.cl

Ora Lassila
Amazon Web Services
USA
ora@amazon.com

Carlos Rojas
IMFD
Chile
cirojas6@uc.cl

Daniel Schwabe
Visiting Researcher USC Information Sciences Institute
USA
dschwabe@gmail.com

Pedro Szekely
USC Information Sciences Institute
USA
dvrgoc@ing.puc.cl

Domagoj Vrgoč
PUC Chile & IMFD
Chile
dvrgoc@ing.puc.cl

ABSTRACT

In this short position paper, we argue that there is a need for a unifying data model that can support popular graph formats such as RDF, RDF* and property graphs, while at the same time being powerful enough to naturally store information from complex knowledge graphs, such as Wikidata, without the need for a complex reification scheme. Our proposal, called the *multilayer graph model*, presents a simple and flexible data model for graphs that can naturally support all of the above, and more. We also observe that the idea of multilayer graphs has appeared in existing graph systems from different vendors and research groups, illustrating its versatility.

1 INTRODUCTION

Recent years have seen renewed interest in using graphs for modeling, managing, querying and analyzing data, particularly in scenarios involving diverse data, incomplete knowledge, multitudinous sources, and so forth. This interest stems from the growing realization in various communities – such as Databases [6], Semantic Web [15], Machine Learning [9], and more recently Knowledge Graphs [16] – that graphs provide a flexible, lightweight and intuitive abstraction well-suited to many complex, diverse domains [1].

Within these different communities, various abstract and concrete graph-based data models have been proposed.[1] Perhaps the simplest such model is that of a directed labeled graph, which is simply a set of triples, where each triple forms a directed labeled edge. This forms the basis of concrete data models, such as the Resource Description Framework (RDF) [7]. Such an abstraction is also used by the Machine Learning community for topics such as knowledge graph embeddings [25] and in the Networks community for topics such as community detection [17]. It is also popular in the Database community, particularly in the theoretical literature, where such graphs are often simply called *graph databases* [27].

However, in practice, directed labeled graphs are sometimes considered *too simple*. What if, for example, we want to add data that describe edges themselves, or graphs themselves? While more complex data can be modeled in directed labeled graphs using various forms of reification [14], the result can often be verbose and unintuitive [1]. Hence a wide range of graph-based models have emerged [3]: the (labeled) property graph model [1, 26] has gained significant popularity in the Database community, while models such as named graphs [10] and RDF-star (RDF*) [12] have been proposed within the Semantic Web community.

With several abstract and concrete graph models now available, the question becomes how to make these models *interoperable*. How can we integrate data from both RDF and property graphs? How can we design a graph database engine that can seamlessly ingest, integrate and query data from any such model? One possibility is to take the simplest model – directed labeled graphs – as our base and use reification [14] to represent more complex models, but as mentioned before, reification is too verbose. Another possibility is to take a more complex model – say property graphs – as our base [4, 11], but this would add complexity to higher levels when we think of graph queries, analytics, learning, etc.

Here we discuss an intermediate solution. Specifically we define *multilayer graphs*: an abstract graph model that extends directed labeled graphs with edge ids. This model largely removes the need for reification when modeling complex data, and yet adds minimal complexity versus directed labeled graphs. Adding edge ids to directed labeled graphs has already been proposed independently in the context of systems developed by the authors of the present paper [18, 19, 24]. Our contribution here is to formally define and motivate this abstract model, comparing it to other graph models.

[1]By an *abstract data model*, herein we refer to a structure used to represent data (e.g., the relational model). A *concrete data model* further adds details important in practice, such as the types of terms allowed, syntaxes, etc. (e.g., SQL's data model).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GRADES & NDA'22, June 12, 2022, Philadelphia, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9384-3/22/06…$15.00
https://doi.org/10.1145/3534540.3534696

We remark that there have been other proposals for formalizing graphs, such as multi-attributed relational structures (MARS) [20], which have been used to model knowledge graphs such as Wikidata [22]. These languages, while promising, go beyond a mere data model, capturing logic formulae over graphs. What we propose here is much simpler: a concise data model for graphs.

We will first introduce existing graph models, and their strengths and weaknesses. We then formally define the multilayer graph model, and show how other graph models can be represented within it. We further discuss practical benefits of the model, and how it is currently being used. We conclude with some future directions.

2 EXISTING GRAPH DATA MODELS

One of the simplest graph models is based on *directed labeled graphs*, which consist of a set of edges of the form $\boxed{a} \xrightarrow{b} \boxed{c}$, where a is called the source node, b the edge label, and c the target node. Such graphs are a staple in the theoretical literature on graph databases [5], and they form the basis of the RDF data model [7], where the source node, edge label and target node are called *subject*, *predicate* and *object*, respectively. In the context of knowledge graphs, nodes are used to represent entities and edges represent binary relations. As an example, the edge $\boxed{\text{Michelle Bachelet}} \xrightarrow{\textit{position held}} \boxed{\text{President of Chile}}$, tells us that Michelle Bachelet was (or is) the president of Chile.

However, directed labeled graphs are sometimes *too* simple. They elegantly represent binary relations, but higher arity relations can be problematic. Take, for example, the two Wikidata knowledge graph [23] statements from Figure 1. Both statements claim that Michelle Bachelet was a president of Chile, and both are associated with *qualifiers* that provide extra information: in this case a start date, an end date, who replaced her, and whom she was replaced by. There are two statements, indicating two distinct periods when she held the position. Also the ids for objects (for example, Q320 and P39) are shown; any positional element can have an id and be viewed as a node in the knowledge graph (for instance `start date`, identified by `P580`, can be a source node of another statement).

Representing statements like this in a directed labeled graph requires some form of *reification* to decompose n-ary relations into binary relations [14]. Figure 2 shows a graph where e_1 and e_2 are nodes representing n-ary relationships. The reification is given by the use of the edges labeled as `source`, `label` and `target`. For simplicity, we use human-readable nodes and labels, where in practice, a node $\boxed{\text{Sebastián Piñera}}$ will rather be given as the identifier $\boxed{\text{Q306}}$, and an edge label `replaces` will rather be given as `P155`. While using reification is a valid solution, its main drawbacks are that: (i) it can easily become cumbersome and inefficient for querying; and (ii) it introduces semantics into graph data (requiring that an edge labeled `replaces` has a particular meaning for instance).

To circumvent this issue, a number of graph models have been proposed to capture higher-arity relations more concisely, including property graphs [8] and RDF* [12, 13]. However, both have limitations that render them incapable of modeling the statements shown in Figure 1 without resorting to reification [14].

On the one hand, property graphs allow labels and property–value pairs to be associated with both nodes and edges, where the statements of Figure 1 can be represented as the property graph in Figure 3. Though more concise than reification, labels, properties

Michelle Bachelet [Q320]

position held [P39]	President of Chile [Q466956]	
	start date [P580]	2014-03-11
	end date [P582]	2018-03-11
	replaces [P155]	Sebastián Piñera [Q306]
	replaced by [P156]	Sebastián Piñera [Q306]
position held [P39]	President of Chile [Q466956]	
	start date [P580]	2006-03-11
	end date [P582]	2010-03-11
	replaces [P155]	Ricardo Lagos [Q331]
	replaced by [P156]	Sebastián Piñera [Q306]

Figure 1: Wikidata statement group for Michelle Bachelet

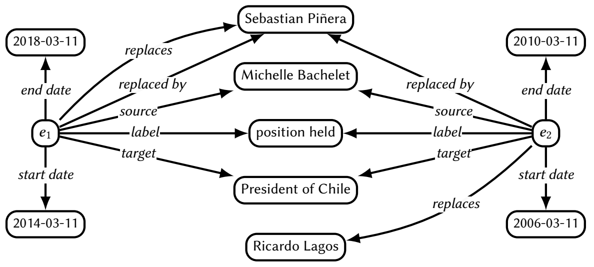
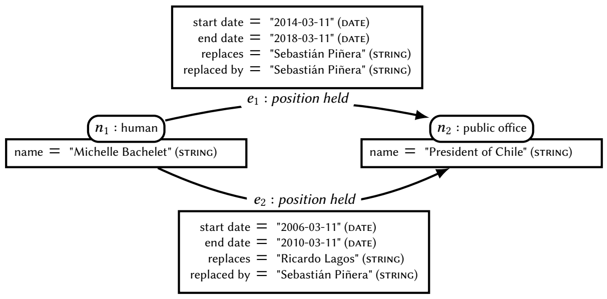


Figure 2: Directed labeled graph reifying Figure 1



start date = "2014-03-11" (DATE)
end date = "2018-03-11" (DATE)
replaces = "Sebastián Piñera" (STRING)
replaced by = "Sebastián Piñera" (STRING)
e_1 : position held
n_1 : human
n_2 : public office
name = "Michelle Bachelet" (STRING)
name = "President of Chile" (STRING)
e_2 : position held
start date = "2006-03-11" (DATE)
end date = "2010-03-11" (DATE)
replaces = "Ricardo Lagos" (STRING)
replaced by = "Sebastián Piñera" (STRING)

Figure 3: Property graph for Figure 1, with extra node labels

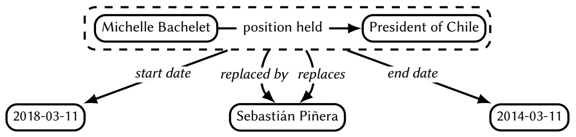


Figure 4: RDF* graph for the first statement of Figure 1

and values are strings, not nodes; for example, "Ricardo Lagos" is not a node, but a string, which complicates, for example, querying for the parties of presidents that Bachelet replaced.

On the other hand, RDF* supports *quoted triples*, which represent an edge as a node [13]. For example, the first statement of Figure 1 can be represented in RDF* as shown in Figure 4. However, we can only represent one of the statements in this way, as we can only have one distinct node per edge; if we add the qualifiers for both statements, we will not know which start date pairs with which end date, for example. A proposed workaround involves adding intermediate nodes to denote different *occurrences* of quoted triples, requiring a reserved term, amongst other complications [13].

3 MULTILAYER GRAPHS

Per the previous section, a key feature needed to model complex statements – such as multiple presidencies of a particular person – is the ability to refer to the entire statement repeatedly. More precisely, we need to be able to reference an edge as it if were a node (or possibly multiple nodes). A multilayer graph captures this feature through edge ids [14, 18, 19].

3.1 Defining multilayer graphs

We will now define multilayer graphs in two equivalent ways: as a function, and as a relation.

Functional definition. Assume a universe Obj of objects (strings, numbers, IRIs, etc.). We define multilayer graphs as follows:

Definition 3.1. A *multilayer graph* $G = (O, \gamma)$ consists of a finite set of objects $O \subseteq \text{Obj}$ and a partial mapping $\gamma : O \to O \times O \times O$.

Intuitively, O is the set of objects that appear in our graph database, and γ models directed, labeled and identified edges between objects. If $\gamma(e) = (n_1, l, n_2)$, this states that the edge (n_1, l, n_2) has id e, label l, and links the source node n_1 to the target node n_2.

We stated that multilayer graphs allow us to capture higher-arity relations more directly. So how do we represent the Wikidata statements from Figure 1? One possible representation is given in Figure 5; here we only show edge ids as needed (all edges have ids).

Relational definition. We can also define a multilayer graph as a single relation: MGraph(source,label,target,eid), where eid (edge id) is a primary key of the relation. An edge $\gamma(e) = (n_1, l, n_2)$ becomes a tuple MGraph(n_1, l, n_2, e). This definition shows how the model can be supported using relational database systems and techniques, as was done for example in KGTK [18], or MillenniumDB [24].

We remark that multilayer graphs are closely related to the graph theoretic notion of *labeled quivers.*

3.2 The layering

The layers in multilayer graphs result from the nested use of edge ids. Given a multilayer graph $G = (O, \gamma)$, the *layer* for an object $o \in O$, denoted as layer(o), is defined as follows. If o is not an edge id (not in the domain of γ), then layer(o) = 0. Otherwise, if $\gamma(o) = (n_1, l, n_2)$; then layer($o$) = max{layer($n_1$), layer($l$), layer($n_2$)} + 1. We call a multilayer graph G an n-layer graph where n is the maximum layer of an edge id in G. Figure 6 is a 2-layer graph, as depicted in Figure 5, where each layer forms a directed labeled graph, where the first layer does not use edge ids as nodes, and where each edge in all subsequent layers uses an edge id from the previous layer as a node. A hypothetical third layer might, for example, add external references to support the specific dates claimed on layer 2.

In some cases, layers may not be clearly defined. Let $D = (V, E)$ denote the dependence graph of a multilayer graph $G = (O, \gamma)$, where V is the codomain of γ, $E \subseteq V \times V$, and $(e_1, e_2) \in V$ if and only if $\gamma(e_1)$ returns a triple containing e_2. We say that the layers of G are *well-defined* if and only if D is acyclic. For example, if $\gamma(e) = (e, l, n)$, then the layers are not well-defined as the layer of e depends recursively on itself, and thus modifies itself. We say that such edge ids are on layer ∞, giving rise to a ∞-layer graph.

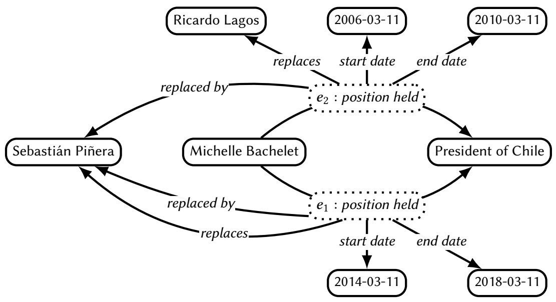


Figure 5: Multilayer graph for Figure 1

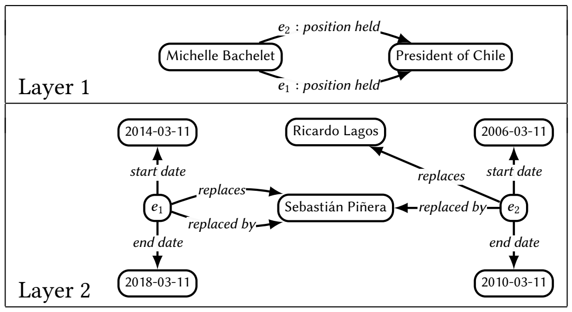


Figure 6: Two layers of Figure 5

The notion of layering lies inherent in other graph models. In Wikidata, qualifiers can be seen as forming a layer 2 graph (per Figure 6). The properties (attribute–value pairs) in property graphs are akin to layer 2 metadata, though strictly speaking the values do not form nodes. In RDF*, quoted triples can be nested arbitrarily, allowing for arbitrary layers; again, however, RDF* does not directly support quoting the same triple multiple times.

3.3 Concrete models using multilayer graphs

A number of concrete data models proposed in different settings already adhere to the multilayer graph model. Hernández et al. [14] propose using singleton named graphs – effectively multilayer graphs in an RDF setting – in order to represent Wikidata qualifiers, placing one triple in each named graph, such that the name acts as an id for the triple. The Knowledge Graph Toolkit (KGTK) [18] for applying analytics over knowledge graphs has likewise used a concrete format conforming to multilayer graphs in order to abstract different forms of graph into an intuitive tabular format that allows for the application and composition of different analytical tasks. The Amazon Neptune graph database further proposes the *one graph* (1G) data model – a concrete version of multilayer graphs– as an underlying model that support RDF, RDF*, property graphs, etc., with often similar rationale to that provided here [19]. MillenniumDB [24] also uses the multilayer graph model – and an extension to support external annotation – as the basis of its open source graph database. These four initiatives have aimed to support diverse graph data models, and all have converged towards the idea of building a unified graph model by using edges as nodes, which we have herein generalized as the multilayer graph model.

4 MULTILAYER VS. OTHER GRAPHS

In Appendix A, we discuss how legacy graph data models can be mapped to the multilayer graph model. Here we compare the models. Table 1 summarizes the features that are directly supported by the respective graph models without requiring *reserved terms*, which would include, for example, source, label and target in Figure 2. Reserved terms may add indirection [14], require special semantics, or increase tuple counts. The features are as follows:

- *Edge label*: assign a label to an edge.
- *Node label*: assign labels to nodes.
- *Edge annotation*: assign attribute–value pairs to an edge.
- *Node annotation*: assign attribute–value pairs to a node.
- *External annotation*: nodes/edges can be annotated without adding new nodes or edges.
- *Edge as node*: an edge can be referenced as a node (this allows edges to be connected to nodes of the graph).
- *Edge as nodes*: an edge can be referenced as multiple nodes.[2]
- *Arbitrary layers*: an edge involving an edge node can itself be referenced as a node, and so on, recursively.
- *Graph as node*: a graph can be referenced as a node.
- *Quotation*: edges referenced as nodes can be distinguished as being either quoted (not asserted) or asserted.

We use MG, to denote the concrete data models described in Section 3.3 that instantiate multilayer graphs. Some blanks in Table 1 are more benign than others; for example, *Node label* requires a reserved term (e.g., rdf:type), but no extra tuples; on the other hand, *Edge as node* requires reification, using at least one extra tuple, and at least one reserved term. All features except *External annotation* can be supported in all models with reserved vocabulary. Furthermore all features can be supported with a single reserved term, except for *Graph as node* and *Quotation*.

The *Edge as nodes* feature is important for many use-cases, such as for modeling the Wikidata example shown in Figure 1 (note: values such as Ricardo Lagos are themselves nodes, which is why property graphs are not considered as supporting this feature). Only named graphs and multilayer graphs can model such examples without reserved terms. Comparing named graphs and multilayer graphs, the latter sacrifices the "*Graph as node*" ability, which, as aforementioned, requires a reserved `graph` term to model in a multilayer graph. However, conceptually speaking, multilayer graphs and named graphs aim to serve different purposes. We view a multilayer graph as modeling one graph, whereas named graphs are intended to represent multiple graphs. In order to manage multiple graphs, one could also potentially consider a *multilayer named graph* model, which allows for multiple multilayer graphs to be named with an additional (fifth) element.

Comparing property graphs with multilayer graphs, we see that property graphs directly support certain features, such as node labels and external annotation. Node labels are trivial to support via a reserved `label` term, but external annotation is not supported. To highlight the issue, compare the property graph of Figure 3 and the multilayer graph of Figure 5: in the property graph, there is no node for Sebastian Piñera, but rather a string; in the multilayer graph, there is a node for Sebastian Piñera that is connected via an edge id.

[2]For example, Michelle Bachelet –*position held*→ President of Chile can be referenced by multiple nodes involved in different edges representing two presidencies.

Table 1: Features supported without reserved terms
(NG = Named Graphs, PG = Property Graphs, MG = Multilayer Graphs)

	RDF	RDF*	NG	PG	MG
Edge type/label	✓	✓	✓	✓	✓
Node label				✓	
Edge annotation		✓	✓	✓	✓
Node annotation	✓	✓	✓	✓	✓
External annotation				✓	
Edge as node		✓	✓		✓
Edge as nodes			✓		✓
Arbitrary layers		✓	✓		✓
Graph as node			✓		
Quotation		✓			

This can change, for example, the results returned for queries. If external annotation were deemed essential to support, one could consider a *multilayer property graph* model, which would add labeling and property functions to a multilayer graph, or equivalently, two additional relations alongside the base MGraph relation, namely Label(object, label) and Prop(object, attribute, value). This would enable stricter compatibility with legacy property graph data and systems. We expand on this in Appendix B.

The final feature, quotation, is useful to describe edges without asserting them; for example, to state that an edge is untrue, uncertain, refuted, disputed, obsolete, etc. For example, Wikidata contains the edge Pluto –*instance of*→ planet, which is assigned a deprecated rank due to being, respectively, obsolete and widely disputed. The RDF-star W3C Community [13] has defined that quoted edges, are not asserted by default, but must rather be asserted separately (a shortcut syntax is provided to quote and assert an edge). In other models, there is no direct way to indicate whether an edge is asserted or not asserted without reserved terms. In multilayer graphs this could be solved by splitting edge ids into asserted and unasserted ones. Appendix E discusses this further.

5 CONCLUSIONS

Multilayer graphs adopt the natural idea of adding edge ids to directed labeled edges in order to concisely model higher-arity relations in graphs without the need for reserved vocabulary or reification. They can naturally represent popular graph models, such as RDF and property graphs, and allow for combining the features of both models in a novel way. Multilayer graphs are inspired by concrete data models developed by several independent groups aiming to support legacy graph models: *singleton named graphs* used to query Wikidata with SPARQL [14], the *KGTK format* used for knowledge graph processing & analytics [18], the *one graph* (1G) model proposed for the Neptune database [19], and the *domain graph model* supported by the MillenniumDB database [24]. Our goal has been to formalize, discuss and draw attention to the abstract data model shared by all these developments, and to highlight its value for unifying different graph-based data models.

ACKNOWLEDGMENTS

Angles, Hogan, Rojas and Vrgoč were supported by ANID – Millennium Science Initiative Program – Code ICN17_002. Vrgoč was also supported by Fondo Puente UC 011/2021. D. Schwabe was partially supported by a grant from CNPq, Brasil.

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter A. Boncz, George H. L. Fletcher, Claudio Gutiérrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan F. Sequeda, Oskar van Rest, and Hannes Voigt. 2018. G-CORE: A Core for Future Graph Query Languages. In *SIGMOD International Conference on Management of Data*. 1421–1432.

[2] 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. https://doi.org/10.1145/3104031

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

[4] Renzo Angles, Harsh Thakkar, and Dominik Tomaszuk. 2020. Mapping RDF Databases to Property Graph Databases. *IEEE Access* 8 (2020), 86091–86110. https://doi.org/10.1109/ACCESS.2020.2993117

[5] Pablo Barceló Baeza. 2013. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*. 175–188. https://doi.org/10.1145/2463664.2465216

[6] Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. *Querying Graphs*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00873ED1V01Y201808DTM051

[7] Richard Cyganiak, David Wood, and Markus Lanthaler. 2014. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation. https://www.w3.org/TR/rdf11-concepts/

[8] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. ACM, 1433–1445. https://doi.org/10.1145/3183713.3190657

[9] William L. Hamilton. 2020. *Graph Representation Learning*. Morgan & Claypool Publishers. https://doi.org/10.2200/S01045ED1V01Y202009AIM046

[10] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. 2013. SPARQL 1.1 Query Language. W3C Recommendation. https://www.w3.org/TR/sparql11-query/

[11] Olaf Hartig. 2014. Reconciliation of RDF* and Property Graphs. *CoRR* abs/1409.3288 (2014). arXiv:1409.3288 http://arxiv.org/abs/1409.3288

[12] Olaf Hartig. 2017. Foundations of RDF★ and SPARQL★ (An Alternative Approach to Statement-Level Metadata in RDF). In *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017 (CEUR Workshop Proceedings)*, Vol. 1912. CEUR-WS.org. http://ceur-ws.org/Vol-1912/paper12.pdf

[13] Olaf Hartig, Pierre-Antoine Champin, Gregg Kellogg, Andy Seaborne, Dörthe Arndt, Jeen Broekstra, Bob DuCharme, Ora Lassila, Peter F. Patel-Schneider, Eric Prud'hommeaux, Ted Thibodeau Jr., and Bryan Thompson. 2021. RDF-star and SPARQL-star. W3C Draft Community Group Report. https://w3c.github.io/rdf-star/cg-spec/2021-07-01.html

[14] Daniel Hernández, Aidan Hogan, and Markus Krötzsch. 2015. Reifying RDF: What Works Well With Wikidata?. In *International Workshop on Scalable Semantic Web Knowledge Base Systems (CEUR Workshop Proceedings)*, Vol. 1457. CEUR-WS.org, 32–47. http://ceur-ws.org/Vol-1457/SSWS2015_paper3.pdf

[15] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. 2010. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press. http://www.semantic-web-book.org/

[16] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *ACM Comput. Surv.* 54, 4 (2021), 71:1–71:37. https://doi.org/10.1145/3447772

[17] San-Chuan Hung, Miguel Araujo, and Christos Faloutsos. 2016. Distributed community detection on edge-labeled graphs using Spark. In *International Workshop on Mining and Learning with Graphs (MLG)*.

[18] Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Milo Rogers, Ronpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe, and Pedro A. Szekely. 2020. KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis. In *International Semantic Web Conference (ISWC)*. Springer, 278–293.

[19] Ora Lassila, Michael Schmidt, Brad Bebee, Dave Bechberger, Willem Broekema, Ankesh Khandelwal, Kelvin Lawrence, Ronak Sharda, and Bryan B. Thompson. 2021. Graph? Yes! Which one? Help! *CoRR* abs/2110.13348 (2021). arXiv:2110.13348 https://arxiv.org/abs/2110.13348

[20] Maximilian Marx, Markus Krötzsch, and Veronika Thost. 2017. Logic on MARS: Ontologies for Generalised Property Graphs. In *International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 1188–1194. https://doi.org/10.24963/ijcai.2017/165

[21] Vinh Nguyen, Olivier Bodenreider, and Amit P. Sheth. 2014. Don't like RDF reification?: making statements about statements using singleton property. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel (Eds.). ACM, 759–770. https://doi.org/10.1145/2566486.2567973

[22] Peter F. Patel-Schneider and David Martin. 2020. Wikidata on MARS. *CoRR* abs/2008.06599 (2020). arXiv:2008.06599 https://arxiv.org/abs/2008.06599

[23] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85. https://doi.org/10.1145/2629489

[24] Domagoj Vrgoc, Carlos Rojas, Renzo Angles, Marcelo Arenas, Diego Arroyuelo, Carlos Buil Aranda, Aidan Hogan, Gonzalo Navarro, Cristian Riveros, and Juan Romero. 2021. MillenniumDB: A Persistent, Open-Source, Graph Database. *CoRR* abs/2111.01540 (2021). arXiv:2111.01540 https://arxiv.org/abs/2111.01540

[25] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. 29, 12 (Dec. 2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[26] Jim Webber. 2012. A programmatic introduction to Neo4j. In *Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH)*. ACM, 217–218. https://doi.org/10.1145/2384716.2384777

[27] Peter T. Wood. 2012. Query languages for graph databases. *SIGMOD Rec.* 41, 1 (2012), 50–60. https://doi.org/10.1145/2206869.2206879

APPENDIX

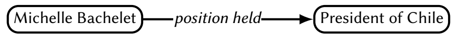
In this appendix we further elaborate on some points made in the paper. In particular, we discuss:

- How to map RDF, RDF*, and property graphs into multilayer graphs (Appendix A).
- We discuss multilayer property graphs in a bit more detail in Appendix B.
- In Appendix C, we discuss how two multilayer graphs can be merged into a single one.
- Appendix D is dedicated to an abstract model for querying multilayer graphs.
- We include a detailed discussion on quoted edges in Appendix E.

A FROM LEGACY TO MULTILAYER GRAPHS

Herein we discuss how RDF, RDF* and property graphs can be mapped to the multilayer graph graph model.

RDF and RDF.* The multilayer graph data model naturally subsumes the RDF graph model, as well as RDF*. To show how RDF is modeled in multilayer graphs, consider the following edge, claiming that Michelle Bachelet was the president of Chile.



We can encode this triple in a multilayer graph by storing the tuple (Michelle Bachelet, position held, President of Chile, e) in the MGraph relation, where e denotes a unique (potentially auto-generated) edge id, or equivalently stating that:

$$\gamma(\mathsf{e}) = (\text{Michelle Bachelet, position held, President of Chile}).$$

The edge id e can be automatically generated. One can thus automatically load an RDF graph into a multilayer graph, by assigning a new edge id to each triple. The id of the edge itself is not needed for RDF, but can be used for modeling RDF-star (RDF*) graphs.

If we wished to use the multilayer graph model to capture named graphs or RDF Datasets [7], we could introduce a reserved term, say `graph`, to assign an edge to a graph. Then a quad (s, p, o, g) would become $\gamma(e_1) = (s, p, o)$, $\gamma(e_2) = (e_1, \texttt{graph}, g)$. Edges without a `graph` edge could be considered as part of the default graph (or we could add a reserved `default` value, if desired. Alternatively, they could be assigned to a `global` default graph that includes all edges.); optionally, *named multilayer graphs* could be considered in the future to support multiple multilayer graphs using a fifth element (quins).

Property graphs. In order to encode (labeled) property graphs as a multilayer graph, we need to be able to represent labels and properties (attribute–value pairs) on both nodes and edges. Edges already have precisely one label associated with them (as permitted by Neo4j [26]), while node ids can also double as a single label. Nodes (and edges) could, if needed, have zero-or-more labels associated with them using a reserved `label` or `type` term, per RDF(S). Properties can be represented as outward edges from a given node or edge id, per, e.g., the *start date*, *replaces*, etc., edges in Figure 5. If we wanted to maintain external annotation as a feature (where attribute values are not nodes, but rather strings), we would need the *multilayer property graph* model discussed in Section 4.

B PROPERTY MULTILAYER GRAPHS

Here we formally define multilayer property graphs.

Functional definition. The functional definition of this concept is as follows:

Definition B.1. A *multilayer property graph* is defined as a tuple $G = (O, \gamma, \texttt{lab}, \texttt{prop})$, where:

- (O, γ) is a multilayer graph;
- $\texttt{lab} : O \to 2^O$ is a function assigning a finite set of labels to an object; and
- $\texttt{prop} : O \times O \to O$ is a partial function assigning a value to a certain property of an object.

Moreover, we assume that for each object $o \in O$, there exists a finite number of properties $p \in O$ such that $\texttt{prop}(o, p)$ is defined.

Relational definition. On the other hand, the relational definition can be defined in terms of the following three relations:

- MGraph(source,label,target,$\underline{\text{eid}}$)
- Label(object,label)
- Prop($\underline{\text{object,attribute}}$,value).

Notice that we require the value of an attribute for a single object to be unique (i.e. (object,attribute) is a key for Prop). On the other hand, an object can have multiple labels. We could then model Figure 3 with tuples such as (here abbreviating terms slightly):

MGraph(n_1, *pos held*, n_2, e_1)	MGraph(n_1, *pos held*, n_2, e_2)
Label(n_1, *human*)	Label(n_2, *public office*)
Label(e_1, *pos held*)	Label(e_2, *pos held*)
Prop(n_1, *name*, `"M.Bachelet"`)	Prop(n_2, *name*, `"P. of Chile"`)
Prop(e_1, *start*, `"2014-03-11"`)	Prop(e_1, *end*, `"2018-03-11"`)
···	

and so forth for other properties. The *replaces* and *replaced by* values could be either added as external annotations to the Prop relation, or as links to nodes representing the people in the MGraph relation. With this model, one could also simplify the MGraph, and exclude the label component, since this can be assigned by the relation Label. However, having a unique edge label is very handy for indexing purposes, and models precisely what is required by the Wikidata knowledge graph.

C MERGING MULTILAYER GRAPHS

Assume we wish to merge two multilayer graphs $G_1 = (O_1, \gamma_1)$ and $G_2 = (O_2, \gamma_2)$. We must take care since the most natural definition, $G_1 \cup G_2 = (O_1 \cup O_2, \gamma_1 \cup \gamma_2)$, may not yield a multilayer graph if there is a clash of edge ids, i.e., if there exists $o \in O_1 \cap O_2$ such that $\gamma_1(o) \neq \gamma_2(o)$. In this case, $\gamma_1(o) \cup \gamma_2(o)$ will no longer even be a function. We see two general solutions for this. First, we can define edge ids to be local (similar to blank nodes in RDF) and then rewrite the edge ids in (say) G_2 to distinguish them from all terms in G_1; this is done by MillenniumDB [24]. Second, we can define edge ids to be global (i.e., externally referenceable) and leave it to the application to reconcile edge id clashes, which may involve removing or renaming edge ids (and their corresponding data) from one multilayer graph, or both; this is done by KGTK [18].

D QUERYING MULTILAYER GRAPHS

Graph query languages are typically founded on basic graph patterns [2, 10]. We can define a basic graph pattern for multilayer graphs as a pair $Q = (X, \xi)$, where $X \subseteq \mathsf{Obj} \cup \mathsf{Var}$ is a set of objects and variables, and $\xi : X \to X \times X \times X$. Given a multilayer graph $G = (O, \gamma)$, let $\mu : \mathsf{Var} \cap X \to O$ denote a mapping from the variables of X to O. Then we define the evaluation $Q(G) = \{\mu \mid \mu(\xi) \subseteq \gamma\}$, where $\mu(\xi)$ is the image of ξ under μ. Path queries can be evaluated on the directed labeled graph that forms the codomain of γ. Other relational features can be layered on top of these base queries to transform or combine sets of solution mappings [2].

E QUOTED EDGES

Here we elaborate a bit more on quoting edges, and provide some examples. Quotations allow us to to describe edges without asserting them; for example, to state that an edge is untrue, uncertain, refuted, disputed, obsolete, etc. For example, Wikidata contains edges claiming that $\boxed{\text{Pluto}}\xrightarrow{\textit{instance of}}\boxed{\text{planet}}$ and $\boxed{\text{Earth}}\xrightarrow{\textit{shape}}\boxed{\text{disk}}$, both of which are assigned a deprecated rank due to being, respectively, obsolete and widely disputed. These claims would be best represented as quoted edges that are defined as deprecated, but not asserted, such that, for example, if we query for instances of $\boxed{\text{planet}}$, we should not receive $\boxed{\text{Pluto}}$ as a result. On the other hand, $\boxed{\text{Michelle Bachelet}}\xrightarrow{\textit{position held}}\boxed{\text{President of Chile}}$ could be asserted, such that, for example, if we query for people who have held the position of $\boxed{\text{President of Chile}}$, we would return $\boxed{\text{Michelle Bachelet}}$. The RDF-star W3C Community [13] has defined that quoted edges, such as $\boxed{\text{Michelle Bachelet}}\xrightarrow{\textit{position held}}\boxed{\text{President of Chile}}$ in Figure 4, are not asserted by default, but must rather be asserted separately (a shortcut syntax is provided to quote and assert an edge). In other models, there is no direct way to indicate whether an edge is asserted or not asserted without reserved terms. A potential solution to support quotation of unasserted edges in multilayer graphs without reserved vocabulary would be to define two disjoint sets of edges ids as objects: asserted and unasserted edge ids.