Incremental View Maintenance for Property Graph Queries

Gábor Szárnyas
Budapest University of Technology and Economics
Department of Measurement and Information Systems
MTA-BME Lendület Research Group on Cyber-Physical Systems
szarnyas@mit.bme.hu

ACM Reference Format:
Gábor Szárnyas. 2018. Incremental View Maintenance for Property Graph Queries. In *Proceedings of 2018 International Conference on Management of Data (SIGMOD'18).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3183713.3183724

1 PROBLEM AND MOTIVATION

Graph processing challenges are common in modern database systems, with the property graph data model gaining widespread adoption [29]. Due to the novelty of the field, graph databases and frameworks typically provide their own query language, such as Cypher for Neo4j [27], Gremlin for TinkerPop [28] and GraphScript for SAP HANA [24]. These languages often lack a formal background for their data model and semantics [1]. To address this, the openCypher initiative [21] aims to standardise a subset of the Cypher language, for which it currently provides grammar specification and a set of acceptance tests to allow vendors to implement their openCypher compatible engine.

Incremental view maintenance has been used for decades in relational database systems [4]. In the graph domain, numerous use cases rely on complex queries and require low latency, including financial fraud detection, source code analysis [32] and checking integrity (or well-formedness) constraints in databases [30]. While these could benefit from incremental evaluation, currently no property graph system provides incremental views. Our research investigates the incremental view maintenance for openCypher queries. A key challenge is that the property graph data model includes lists and maps, and queries can return arbitrarily nested data structures.

We propose three desirable properties for an incremental property graph query engine: (IVM) incremental view maintenance, (FGN) fine granularity update operations on nested data structures, (ORD) ordering. Previous research showed that IVM and FGN together are possible [19]. However, as stated in [8], *"incremental view maintenance [IVM] strategies for data models that preserve order [ORD] remain an open problem to date"*. While removing support for *ordering* might seem a plausible workaround, it would pose serious limitations: (1) queries that require top-k results are common [17] and (2) even more importantly, Cypher handles paths as an alternating list of vertices and edges, which must be kept ordered. Therefore, we investigate the following research question: *Which practical fragment of the openCypher language is incrementally* maintainable?

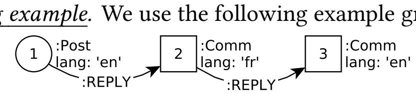
SIGMOD'18, June 10–15, 2018, Houston, TX, USA
© 2018 Copyright held by the owner/author(s).
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of 2018 International Conference on Management of Data (SIGMOD'18)*, https://doi.org/10.1145/3183713.3183724.

2 PRELIMINARIES

Data model. A *property graph* is $G = (V, E, st, L, T, \mathcal{L}, \mathcal{T}, P_v, P_e)$, where V is a set of vertices, E is a set of edges and $st : E \to V \times V$ assigns the source and target vertices to edges. Vertices are labelled from L by function $\mathcal{L}$ and edges are typed from T by function $\mathcal{T}$. Let $D = \cup_i D_i$ be the union of atomic domains D_i. P_v is a set of vertex properties. A vertex property $p_i \in P_v$ is a partial function $p_i : V \to D_i$. Edge properties P_e can be defined similarly.

Given a property graph G, relation r is a *graph relation* if the following holds [13]: $\forall A \in \text{sch}(r) : \text{dom}(A) \subseteq V \cup E \cup D$, where $\text{sch}(r)$ is the schema of r (a list containing attribute names), $\text{dom}(A)$ is the domain of attribute A, and V/E are the vertices/edges of G.

Running example. We use the following example graph:



We use an example query that lists Posts p, along with threads t that contain (transitive) reply Comm[ent]s that are written in the same lang[uage] as the Post. The result is shown on the right. (For conciseness, edges are omitted from paths throughout the paper.)

```
MATCH t = (p:Post)-[:REPLY*]->(c:Comm)
WHERE p.lang = c.lang
RETURN p, t
```

p	t
1	[1, 2, 3]

GRA. Graph queries can be formulated in *graph relational algebra* (GRA) [20], which introduces two graph-specific operators: (1) the *get-vertices* nullary operator $\bigcirc_{(\mathsf{v}:\mathsf{V})}$, which returns vertices v with a label V to serve as a base relation for later operators, (2) the *expand-out* unary operator $\uparrow\,^{(\mathsf{w}:\mathsf{W})}_{(\mathsf{v})}\,[\mathsf{:E}]\,(r)$ that navigates from v on an edge typed E to a vertex w with label W. The expand-out operator can also define transitive closure patterns, denoted by the $*$ symbol. GRA allows nested data structures, i.e. if x is an attribute of a graph relation, $x.p$ accesses the value of property p in x [13].

NRA. To allow precise formalisation of nested data structures, we use *nested relational algebra* (NRA) [7, 14], which allows arbitrary nesting of relations. To access nested values, attribute A of a nested relation r can be *unnested* using the operator $\mu_A(r)$. Nested relations can also represent properties of vertices/edges along with collections such as lists and maps. We present two nested relations α and β that store the vertices and edges of the graph, respectively:

α

<table>
<tr><th>id</th><th>label</th><th>properties</th></tr>
<tr><td>1</td><td>Post</td><td><table><tr><th>key</th><th>value</th></tr><tr><td>lang</td><td>en</td></tr></table></td></tr>
<tr><td>2</td><td>Comm</td><td>…</td></tr>
</table>

β

s	t	type	properties
1	2	REPLY	
2	3	REPLY	

We define operators formally as $\bigcirc_{(\mathsf{v}:\mathsf{V})} \equiv \pi_{\mathsf{id}\to\mathsf{v}}\sigma_{\alpha.\mathsf{label}=\mathsf{V}}(\alpha)$ and $\uparrow\,^{(\mathsf{:W})}_{(\mathsf{v})}\,[\mathsf{:E}]\,(r) \equiv \sigma_{r.v=\beta.\mathsf{s}\wedge\beta.\mathsf{type}=\mathsf{E}\wedge\beta.\mathsf{t}=\alpha.\mathsf{id}\wedge\alpha.\mathsf{label}=\mathsf{W}}\,(r \bowtie \beta \bowtie \alpha)$.

3 RELATED WORK

Cypher. Due to its novelty, there are only a few research works on the formalisation of (open)Cypher. An early attempt to provide a framework for the theoretical representation of openCypher queries was published in [13]. In [20], we published a formalisation of a subset of openCypher that mapped queries to GRA. The *Cypher for Apache Spark* project is an ongoing effort to adapt the Cypher language to Spark [22]. None of these works considers IVM. Graphflow [15] is an active graph database for incremental openCypher queries. However, it does not support nested data structures.

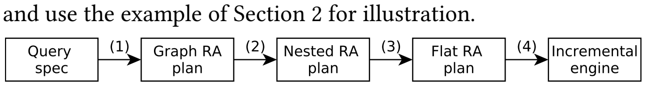
IVM of graph queries. The VIATRA framework [33] provides an incremental query engine over the object-oriented Eclipse Modeling Framework. However, it does not support FGN or ORD. Strider [26] is a system supporting continuous SPARQL queries. As the RDF data model does not handle collections as first class citizens (only head-tail style lists are supported), FGN is not supported.

Querying nested data structures. Paper [16] presents a method for incremental view maintenance in object-oriented databases, but ordering is not supported. Recently, the authors of [5, 6] formalised the language of the MongoDB document store using nested relational algebra, including ordering. However, IVM was not considered. An approach for incremental calculation of XQuery expressions is presented in [9] and its accompanying technical report [8].

4 APPROACH AND CONTRIBUTIONS

As discussed in Section 1, order-preserving lists are required to store paths. Henceforth, we propose a property graph query model that only allows (unordered) *bags*, except for paths that are still stored *as a list* but can only be updated as an atomic unit (i.e. the previous path has to be deleted and the new one has to be inserted). We argue that the distinction between collection properties and paths makes sense from a practical point of view: collection properties often receive updates, while paths only benefit from incremental updates in rare cases (e.g. when a single transaction deletes an edge in the path but adds another one that keeps the path from deleting).

Overview. We propose the following workflow for compiling property graph queries to an incrementally maintainable expression and use the example of Section 2 for illustration.



(1) Compile the queries to GRA. A mapping from openCypher was given in our earlier work [20]. The example query results in:

$$\pi_{\mathsf{p,t}}\sigma_{\mathsf{c.lang=p.lang}}\left(\uparrow^{\mathsf{(c:Comm)}}_{\mathsf{(p)}}[\mathsf{:REPLY*}]\left(\bigcirc_{\mathsf{(p:Post)}}\right)\right)$$

(2) Transform GRA to NRA, which is the key step to allow incremental maintenance. As expand operators cannot be maintained incrementally, they are replaced with joins. For this, we introduce the nullary *get-edges* operator $\Uparrow^{\mathsf{(w:W)}}_{\mathsf{(v:V)}}[\mathsf{e:E}]$ that returns triples (v, e, w) for each edge e of type E between v of label V and w of label W. Using this, each *expand-out* is replaced with *natural joins*:

$$\uparrow^{\mathsf{(w:W)}}_{\mathsf{(v)}}[\mathsf{:E}]\,(r) \equiv r \bowtie \Uparrow^{\mathsf{(:E)}}_{\mathsf{(v:V)}}[\mathsf{w:W}]$$

Similarly, *transitive expand-outs* are replaced with *transitive joins*:

$$\uparrow^{\mathsf{(w:W)}}_{\mathsf{(v)}}[\mathsf{:E*}]\,(r) \equiv r \bowtie^{*} \Uparrow^{\mathsf{(:E)}}_{\mathsf{(v:V)}}[\mathsf{w:W}]$$

Unlike relational databases, property graphs do not have a predefined schema. Hence, we slightly modify the *unnest* operator (Section 2) so that defines specific attribute(s) to be unnested from the nested relation. For example, $\mu_{\mathsf{c.lang\to cL}}$ extracts the lang property of c. Using these rules, the example is transformed to: $\pi_{\mathsf{p,t}}\sigma_{\mathsf{cL=pL}}\mu_{\mathsf{c.lang\to cL,\,p.lang\to pL}}\left(\bigcirc_{\mathsf{(p:Post)}} \bowtie^{*} \left(\Uparrow^{\mathsf{(c:Comm)}}_{\mathsf{(p:Post)}}[\mathsf{:REPLY}]\right)\right)$

(3) Transform NRA to FRA following the approaches presented in [7, 25]. However, a key difference is that due to their schema-free nature, *the schema of the nested relations is not known for property graphs in advance* and has to be inferred based on the query. Therefore, this step includes pushing down nested attributes to the $\bigcirc$ and $\Uparrow$ operators. On the example, this results in $\pi_{\mathsf{p,t}}\sigma_{\mathsf{cL=pL}}\left(\bigcirc_{\mathsf{(p:Post\{lang\to pL\})}} \bowtie^{*} \left(\Uparrow^{\mathsf{(c:Comm\{lang\to cL\})}}_{\mathsf{(p:Post)}}[\mathsf{:REPLY}]\right)\right)$,

where the notation $\{\mathsf{lang} \to \mathsf{pL}\}$ represents a property that must be included in the base relation returned by the $\bigcirc$ or $\Uparrow$ operator.

(4) Create an incremental view for the FRA expression. Incremental view maintenance algorithms for FRA are well studied both from a theoretical perspective [2, 4, 10, 11] and implementation-wise, with many practical tools [12, 33] and research prototypes [15, 26, 31]. While they are not expressible in first-order logic, it is possible to evaluate transitive operations incrementally [3, 23].

Based on this approach, we propose that a fragment of the openCypher language, with unordered bags (instead of lists) and atomic paths (which can only be inserted or deleted, and lose their ordering when unnested), can be evaluated using relational IVM techniques.

Evaluation. The presented approach allows IVM for property graph queries while allowing FGN and some degree of ORD (for paths). In particular, the proposed fragment still allows returning paths and *path unwinding* [1], a feature that permits the query to iterate over the nodes of a path variable. The main tradeoff of the approach is that it does not allow users to use lists in the data model and queries. It is also not possible to specify top-k style queries, e.g. get the top 3 messages, based on the number of replies received.

Summary of contributions. Up to our best knowledge, our research is the first to investigate challenges of *incremental view maintenance for property graph queries*. We put a particular emphasis on handling nested data structures and ordering; and propose to limit the usage of ordering for (atomic) paths. Formulating the queries in NRA and flattening it to an FRA expression allows us to infer the *minimal schema* required by each operator, based on the query specification. Our approach does not require a priori knowledge of the data schema, unlike the *schema cleanup* algorithm of [34] (defined in the context of evaluating XQuery expressions on XML documents) and the *schema merging* algorithm of [18] (defined for consolidating multiple schemas into a mediated one).

Limitations and future work. Property graph queries present numerous additional challenges not presented in this paper. In particular, aggregations, the `OPTIONAL MATCH`, `WITH`, `SKIP` constructs were omitted, and are discussed (for non-incremental queries) in our earlier work [20]. Expressions were also left for future work.

ACKNOWLEDGMENTS

This work was partially supported by NSERC RGPIN-04573-16 and the MTA-BME Lendület Cyber-Physical Systems Research Group.

REFERENCES

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

[2] Gábor Bergmann. 2013. *Incremental Model Queries in Model-Driven Design.* Ph.D. dissertation. Budapest University of Technology and Economics, Budapest.

[3] Gábor Bergmann, István Ráth, Tamás Szabó, Paolo Torrini, and Dániel Varró. 2012. Incremental Pattern Matching for the Efficient Computation of Transitive Closure. In *ICGT (Lecture Notes in Computer Science)*, Vol. 7562. Springer, 386–400. https://doi.org/10.1007/978-3-642-33654-6_26

[4] José A. Blakeley, Per-Åke Larson, and Frank Wm. Tompa. 1986. Efficiently Updating Materialized Views. In *SIGMOD*. 61–71. https://doi.org/10.1145/16894.16861

[5] Elena Botoeva et al. 2016. A Formal Presentation of MongoDB (Extended Version). *CoRR* abs/1603.09291 (2016). http://arxiv.org/abs/1603.09291

[6] Elena Botoeva et al. 2016. OBDA Beyond Relational DBs: A Study for MongoDB. In *Description Logics*.

[7] Jan Van den Bussche. 2001. Simulation of the nested relational algebra by the flat relational algebra, with an application to the complexity of evaluating powerset algebra expressions. *Theor. Comput. Sci.* 254, 1-2 (2001), 363–377. https://doi.org/10.1016/S0304-3975(99)00301-1

[8] Katica Dimitrova, Maged El-Sayed, and Elke A. Rundensteiner. 2003. *Order-Sensitive View Maintenance of Materialized XQuery Views.* Technical Report. Computer Science Department, Worcester Polytechnic Institute. WPI-CS-TR-03-17.

[9] Katica Dimitrova, Maged El-Sayed, and Elke A. Rundensteiner. 2003. Order-Sensitive View Maintenance of Materialized XQuery Views. In *ER*. 144–157. https://doi.org/10.1007/978-3-540-39648-2_14

[10] Timothy Griffin and Leonid Libkin. 1995. Incremental Maintenance of Views with Duplicates. In *SIGMOD*. 328–339. https://doi.org/10.1145/223784.223849

[11] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. 1993. Maintaining Views Incrementally. In *SIGMOD*. 157–166. https://doi.org/10.1145/170035.170066

[12] Red Hat. 2017. Drools. http://www.drools.org/. (2017).

[13] Jürgen Hölsch and Michael Grossniklaus. 2016. An Algebra and Equivalences to Transform Graph Patterns in Neo4j. In *GraphQ at EDBT/ICDT*.

[14] Gerhard Jaeschke and Hans-Jörg Schek. 1982. Remarks on the Algebra of Non First Normal Form Relations. In *PODS*, Jeffrey D. Ullman and Alfred V. Aho (Eds.). ACM, 124–138. https://doi.org/10.1145/588111.588133

[15] Chathura Kankanamge et al. 2017. Graphflow: An Active Graph Database. In *SIGMOD*. 1695–1698. https://doi.org/10.1145/3035918.3056445

[16] Harumi A. Kuno and Elke A. Rundensteiner. 1998. Incremental Maintenance of Materialized Object-Oriented Views in MultiView: Strategies and Performance Evaluation. *IEEE Trans. Knowl. Data Eng.* 10, 5 (1998), 768–792. https://doi.org/10.1109/69.729731

[17] LDBC Social Network Benchmark task force. 2018. *LDBC Social Network Benchmark (SNB).* Technical Report. Linked Data Benchmark Council. https://ldbc.github.io/ldbc_snb_docs/ldbc-snb-specification.pdf.

[18] Xiang Li, Christoph Quix, David Kensche, Sandra Geisler, and Lisong Guo. 2011. Automatic generation of mediated schemas through reasoning over data dependencies. In *ICDE*. 1280–1283. https://doi.org/10.1109/ICDE.2011.5767913

[19] Jixue Liu, Millist W. Vincent, and Mukesh K. Mohania. 1999. Incremental Maintenance of Nested Relational Views. In *IDEAS*. 197–205. https://doi.org/10.1109/IDEAS.1999.787268

[20] József Marton, Gábor Szárnyas, and Dániel Varró. 2017. Formalising openCypher Graph Queries in Relational Algebra. In *ADBIS*. 182–196. https://doi.org/10.1007/978-3-319-66917-5_13

[21] Neo Technology. 2018. openCypher Project. http://www.opencypher.org/. (2018).

[22] openCypher. 2018. CAPS: Cypher for Apache Spark. https://github.com/opencypher/cypher-for-apache-spark. (2018).

[23] Chaoyi Pang, Guozhu Dong, and Kotagiri Ramamohanarao. 2005. Incremental maintenance of shortest distance and transitive closure in first-order logic and SQL. *ACM Trans. Database Syst.* 30, 3 (2005), 698–721. https://doi.org/10.1145/1093382.1093384

[24] Marcus Paradies et al. 2017. GraphScript: implementing complex graph algorithms in SAP HANA. In *DBPL*. 13:1–13:4. https://doi.org/10.1145/3122831.3122841

[25] Jan Paredaens and Dirk Van Gucht. 1992. Converting Nested Algebra Expressions into Flat Algebra Expressions. *ACM Trans. Database Syst.* 17, 1 (1992), 65–93. https://doi.org/10.1145/128765.128768

[26] Xiangnan Ren et al. 2017. Strider: An Adaptive, Inference-enabled Distributed RDF Stream Processing Engine. *PVLDB* 10, 12 (2017), 1905–1908. http://www.vldb.org/pvldb/vol10/p1905-ren.pdf

[27] Ian Robinson, Jim Webber, and Emil Eifrém. 2015. *Graph Databases* (2nd ed.). O'Reilly Media.

[28] Marko A. Rodriguez. 2015. The Gremlin graph traversal machine and language (invited talk). In *DBPL*. 1–10. https://doi.org/10.1145/2815072.2815073

[29] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. 2017. The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. *PVLDB* 11, 4 (2017), 420–431. http://www.vldb.org/pvldb/vol11/p420-sahu.pdf

[30] Gábor Szárnyas, Benedek Izsó, István Ráth, and Dániel Varró. 2017. The Train Benchmark: Cross-Technology Performance Evaluation of Continuous Model Validation. *Softw. Syst. Model.* (2017).

[31] Gábor Szárnyas, János Maginecz, and Dániel Varró. 2017. Evaluation of Optimization Strategies for Incremental Graph Queries. *Periodica Polytechnica Electrical Engineering and Computer Science* 61, 2 (2017), 175–192. https://doi.org/10.3311/PPee.9769

[32] Zoltán Ujhelyi et al. 2015. Performance comparison of query-based techniques for anti-pattern detection. *Information & Software Technology* 65 (2015), 147–165. https://doi.org/10.1016/j.infsof.2015.01.003

[33] Dániel Varró, Gábor Bergmann, Ábel Hegedüs, Ákos Horváth, István Ráth, and Zoltán Ujhelyi. 2016. Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework. *Softw. Syst. Model.* 15, 3 (2016), 609–629. https://doi.org/10.1007/s10270-016-0530-4

[34] Xin Zhang, Bradford Pielech, and Elke A. Rundensteiner. 2002. Honey, I shrunk the XQuery!: an XML algebra optimization approach. In *WIDM at CIKM*. 15–22. https://doi.org/10.1145/584931.584936