openCypher over RDF: Connecting Two Worlds

Michael Schmidt*, Brad Bebee, Willem Broekema, Mohamed Elzarei, Carlos Manuel Lopez Enriquez, Marcin Neyman, Florian Schmedding, Andreas Steigmiller, Bryan Thompson, Geo Varkey, Gregory Todd Williams and Amanda Xiang

Amazon Neptune Team, Amazon Web Services, Seattle, WA

Abstract

Today's graph database space is divided into two – for the most part, separated – technology stacks: Labeled Property Graphs (LPGs) and RDF. As the team working on Amazon Neptune, a graph database that supports both technologies, we aim to give our customers the choice and flexibility they need to address their graph use cases. What we learned when working with them on their graph applications – from social networks, recommendations, fraud detection, to Knowledge Graphs and LLM grounding – is that the two technology stacks each have their unique strengths. RDF, with its standardized serialization formats, global identifiers, and the availability of Linked Open Data sets, is of particular value for data architects who seek to build, integrate, and interchange graph data. Application development teams, on the other hand, often prefer LPG query languages to interact with graphs, due to their intuitive syntax, the maturity of developer ecosystems (client drivers, programming language integration, etc.), and graph-specific features such as built-in support for path extraction and algorithms. In this demo we will present our recent work on openCypher over RDF, which aims to combine the strengths of the two worlds by (a) allowing our customers to load LPG and RDF data into a single, connected graph and (b) querying this single graph using the openCypher query language. From a conceptual perspective, this functionally is achieved by an overarching graph metamodel called *OneGraph*, which encompasses both data models and provides LPG and RDF specific views that implicitly define query semantics. We will utilize open data sets to showcase how we combine LPG and RDF data into a single data graph, demonstrate how this unified graph can be queried and modified using openCypher, and discuss concepts, design decisions, as well as remaining challenges in aligning the LPG and RDF stacks.

1. Introduction

The Semantic Web stack, which comes with the W3C standardized Resource Description Format (RDF) as its foundational data model, originates from the vision of enhancing the Web with a globally connected, machine-understandable network of knowledge [1]. RDF decomposes data into (*subject*, *predicate*, *object*) triples that build upon globally unique identifiers for resources, so-called IRIs [2]. This clears the way for data serialization, exchange, and interlinking at global scale. Consequently, companies often choose RDF when defining a broader, top-down information architecture that aims to connect data across heterogeneous domains and feeds multiple applications. Labeled Property Graphs (LPGs), on the other hand, were driven by companies, Open Source initiatives like Apache TinkerPop, and non-profit organization such as LDBC [3] alike. As a result, different flavors of the LPG data model and different query language

Posters, Demos, and Industry Tracks at ISWC 2024, November 13–15, 2024, Baltimore, USA

*Corresponding author: schmdtm@amazon.com.

have emerged. As a common denominator, LPGs abstract graphs as vertices connected through edges, both of which can be described by properties (essentially, key-value pairs) and may be tagged with labels. The absence of built-in semantics and a looser set of constraints gives LPG users a larger degree of freedom, e.g. when it comes to questions like what labels are used for, or whether identifiers are globally unique vs. application scoped. While this can be seen as a downside of LPGs for building and interlinking graphs in Open World scenarios, it lowers complexity and makes LPGs an attractive choice especially for closed application scenarios.

Differences in coverage, usability, and expressiveness can also be found when comparing LPG and RDF query languages. SPARQL, the W3C standardized language for RDF, comes with a concise semantics and is well-suited for declarative extraction of subgraphs via pattern matching. While it offers some unique features compared to LPG query languages – built-in federation across SPARQL endpoints, convenient querying of schema information, and reasoning support – it lacks other functionality that is commonly found in LPG query languages, most notably when it comes to path queries. The SPARQL 1.1 extensions introduced support for property paths, but important graph use cases (e.g., binding paths to variables and returning them as query output) remain hard, in some cases impossible, to achieve. In contrast, LPG languages like Gremlin [4], openCypher [5], and GQL [6] treat paths as first-class citizens, coming with built-in data types to represent and user-defined functions to process paths.

The key take-away is that both LPG and RDF have unique characteristics which, from an end user perspective, translate into pros and cons for choosing either of the two stacks. When users build graph applications today, they typically opt into either of the two worlds – a decision with profound consequences that is not only hard to make, but also expensive to revert if requirements change. To address this situation, we proposed *OneGraph* [7] as an initiative to bring the two worlds together and allows graph users to benefit from the best of each world.

2. OneGraph

OneGraph aims to achieve seamless interoperability between LPGs and RDF by providing users the ability to load and manage LPG and RDF data in a unified data graph. It can be understood as a meta model that *comprises both RDF and LPGs*. The conceptual architecture depicted in Fig. 2 illustrates the basic idea (components marked in bold font are in scope for this demo). When loading data, LPG and RDF are both mapped into OneGraph's internal data model. This facilitates the co-existence of (possibly interlinked) data from both stacks inside the same database. The idea of mapping both LPG and RDF into an overarching model, which comes with the merit of a lossless representation for both formats, is conceptually different from the majority of prior work in this space, which mostly focused on exploring *direct* mappings at data model [8, 9] or query language layer (e.g., [10, 11, 12]). Approaches that propose a unifying meta model like OneGraph were only introduced recently, either developed in parallel to OneGraph (such as *multilayer graphs* [13]) or directly inspired by OneGraph (e.g., *statement graphs* [14]).

While a detailed formalization of the OneGraph model is out of scope for this demo, the basic idea behind OneGraph is to define data model agnostic elements that both LPG and RDF data can be mapped to. As an example, two such element types are *relationship statements*, which allow to represent links between resources, and *property statements*, to describe properties of

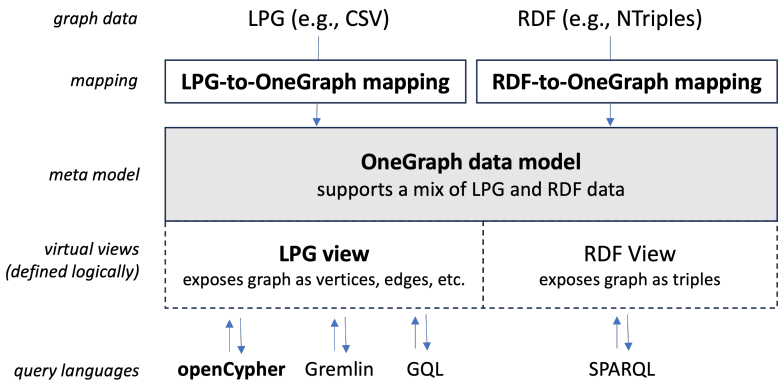

Figure 1: Conceptual overview of OneGraph and how it is used in Neptune Analytics

resources. Relationship statements, for instance, arise from the RDF-to-OneGraph mapping for RDF triples that have resources in object positions, and also from the LPG-to-OneGraph mapping for edges in property graphs. Similarly, RDF triples with literals in object position are mapped to property statements, and so are property graph vertex properties.

On top of its internal format, OneGraph then allows to query the unified graph using an LPG or RDF query language of choice. Conceptually, this is achieved through logical views – virtually defined mappings from the OneGraph model back into either LPG or RDF that implicitly define the semantics of read queries for LPG and RDF languages. Relationship statements, for instance, are mapped to property graph edges, no matter if they originated from RDF or LPG data.

To bridge the gap between identifiers, OneGraph also allows for the co-existence of IRIs (as globally scoped identifiers originating from RDF dataset) and so-called *local resource identifiers* (LRIs) originating from LPG data. In the RDF view, LRIs are exposed as IRIs that are prefixed with a system local namespace; vice versa, in LPG languages full IRIs are identified through syntactic conventions that set them apart from LRIs (we provide an example below in Section 3).

3. Demo

In the demo, we sketch the life cycle of building, querying, and maintaining graphs composed from both LPG and RDF data using an interactive Jupyter notebook running over Neptune Analytics – our memory-optimized graph database engine – which implements OneGraph concepts to support openCypher over RDF. The demo scenario uses a mix of public LPG and RDF graphs: Air Routes[1], an LPG dataset describing airports and flight routes, contextualized with RDF graphs from GeoNames[2] and Wikidata[3]. We will highlight serialization formats (CSV for LPG, N-Triples for RDF) and showcase mechanisms to integrate data into a connected graph.

The central part of the demo will be the interactive execution of openCypher queries against the unified graph. We will highlight that our design supports RDF querying purely via syntactic

[1]Available at https://github.com/krlawrence/graph/tree/master/sample-data (last accessed: Sep 2, 2024)
[2]Available for download at https://www.geonames.org/ontology/documentation.html (last accessed: Sep 2, 2024)
[3]See https://www.wikidata.org/ (last accessed: Sep 2, 2024).

conventions within openCypher, allowing users to reuse existing tooling (such as syntax validators) without modifications. A focus will be on the syntax to disambiguate local identifiers (as found in Air Routes) and global IRIs (stemming from GeoNames and Wikidata). Complementary, we will discuss minor syntactic extensions to openCypher for prefixed IRI support as an opt-in feature for increased readability. To provide a simplistic example of an openCypher query that leverages these syntactic extensions, the following query matches Wikidata airports (where the prefixed Wikidata IRI `entity::Q644371` identifies the class "Airport") and returns their IRIs as well as their opening dates (identified through Wikidata property `prop::P1619`):

```
PREFIX entity : <http://www.wikidata.org/entity/>
PREFIX prop   : <http://www.wikidata.org/prop/direct/>
MATCH (airport : entity::Q644371)
RETURN id(airport) AS airportIri, airport.prop::P1619 AS openingDate
```

Beyond such simple queries that highlight syntactic aspects, our demo will also cover areas in which openCypher as a query language for RDF provides benefits over SPARQL. This includes (a) classes of openCypher path queries that cannot be expressed in SPARQL today, (b) support for composite types (e.g., lists and maps) and result transformations such as folding and unfolding, and (c) built-in support for stored procedures and graph analytics, which we demonstrate via queries that invoke graph algorithms like Breadth First Search and Page Rank. In addition to read queries, we will include openCypher update queries, with a focus on interoperability aspects between LPG and RDF such as connecting LPG and RDF subgraphs, introducing edge properties over RDF data, or linking LPG subsets of the graph against RDF ontologies.

Last but not least, on the conceptual side we will highlight the idea behind the *OneGraph* data model, how it helps Neptune Analytics to manage LPGs and RDF in a unified way, and discuss key challenges that we faced when designing openCypher over RDF. We will make the demo publicly available to users who want to explore the functionality outside of the conference.

4. Conclusion and Future Work

We see the ability to manage integrated RDF and LPG graphs and query them with openCypher as a first step towards the broader vision behind *OneGraph*. Directions that we plan to explore in the future include extended feature coverage in openCypher for RDF (e.g., named graph and RDF blank node support), LPG-RDF interoperability for other query languages (like SPARQL, Gremlin, and GQL), as well as combining of fragments from different query languages within a single query against OneGraph. We are also engaged in standardization activities that aim to bridge the worlds between RDF and LPGs. Examples include composite type support for RDF and SPARQL [15], which aims to close usability and expressiveness gaps between the RDF and LPG data model and query languages, as well the RDF-star Working Group [16], which seeks to extend RDF with user-friendly reification mechanisms similar to LPG edge properties.

We do believe that interoperability between LPGs and RDF contains an interesting set of open research challenges, especially for the Semantic Web community, and are convinced that researching and building technology to overcome the separation of the two worlds is a unique opportunity to accelerate the adoption of Semantic Web technology in industry. We would like to encourage researchers that are interested in this space to reach out to us.

References

[1] O. Lassila, J. Hendler, T. Berners-Lee, The Semantic Web, Scientific American 284 (2001) 34–43.

[2] M. Duerst, M. Suignard, RFC 3987: Internationalized Resource Identifiers (IRIs), 2005.

[3] P. Boncz, LDBC: Benchmarks for Graph and RDF Data Management, in: Proceedings of the 17th International Database Engineering & Applications Symposium, 2013, pp. 1–2.

[4] M. A. Rodriguez, The Gremlin Graph Traversal Machine and Language (invited talk), in: Proceedings of the 15th Symposium on Database Programming Languages, 2015, pp. 1–10.

[5] A. Green, M. Junghanns, M. Kießling, T. Lindaaker, S. Plantikow, P. Selmer, openCypher: New Directions in Property Graph Querying., in: EDBT, 2018, pp. 520–523.

[6] A. Deutsch, N. Francis, A. Green, K. Hare, B. Li, L. Libkin, T. Lindaaker, V. Marsault, W. Martens, J. Michels, et al., Graph Pattern Matching in GQL and SQL/PGQ, in: Proceedings of the 2022 International Conference on Management of Data, 2022, pp. 2246–2258.

[7] O. Lassila, M. Schmidt, O. Hartig, B. Bebee, D. Bechberger, W. Broekema, A. Khandelwal, K. Lawrence, C. M. Lopez Enriquez, R. Sharda, et al., The OneGraph Vision: Challenges of Breaking the Graph Model Lock-in, Semantic Web 14 (2023) 125–134.

[8] H. Chiba, R. Yamanaka, S. Matsumoto, G2GML: Graph to graph mapping language for bridging RDF and property graphs, in: International Semantic Web Conference, Springer, 2020, pp. 160–175.

[9] S. Khayatbashi, S. Ferrada, O. Hartig, Converting property graphs to RDF: a Preliminary Study of the Practical Impact of Different Mappings, in: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), 2022, pp. 1–9.

[10] R. Mutharaju, A SPARQL to Cypher Transpiler, Ph.D. thesis, Indraprastha Institute of Information Technology New Delhi, 2022.

[11] Z. Zhao, X. Ge, Z. Shen, C. Hu, H. Wang, S2CTrans: Building a bridge from SPARQL to Cypher, in: International Conference on Database and Expert Systems Applications, Springer, 2023, pp. 424–430.

[12] H. Thakkar, D. Punjani, J. Lehmann, S. Auer, Two for one: Querying property graph databases using SPARQL via gremlinator, in: Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), 2018, pp. 1–5.

[13] R. Angles, A. Hogan, O. Lassila, C. Rojas, D. Schwabe, P. Szekely, D. Vrgoč, Multilayer graphs: A unified data model for graph databases, in: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), 2022, pp. 1–6.

[14] E. Gelling, G. Fletcher, M. Schmidt, Bridging graph data models: RDF, RDF-star, and property graphs as directed acyclic graphs, arXiv preprint arXiv:2304.13097 (2023).

[15] O. Hartig, G. Williams, M. Schmidt, O. Lassila, C. M. L. Enriquez, B. Thompson, Datatypes for Lists and Maps in RDF Literals, in: European Semantic Web Conference, Springer, 2024.

[16] World Wide Web Consortium, RDF-star Working Group (2024). URL: https://www.w3.org/groups/wg/rdf-star/, last accessed: Sep 2, 2024.