ENTERPRISE KNOWLEDGE

Global Knowledge & Information Management Services

Cutting Through the Noise: An Introduction to RDF & LPG Graphs

Definitions, Comparisons, and Common Questions

Benjamin Kass

Graph is good. From capturing business understanding to support standardization and data analytics to informing more accurate LLM results through Graph-RAG, knowledge graphs are an important component of how modern businesses translate data and content into actionable knowledge and information. For individuals and organizations that are beginning their journey with graph, two of the most puzzling abbreviations that they will encounter early on are RDF and LPG. What are these two acronyms, what are their strengths and weaknesses, and what does this mean for you? Follow along as this article walks through RDF and LPG, touching on these and other common questions.

Definitions

RDF

To paraphrase from our deep dive on RDF, the Resource Description Framework (RDF) is a semantic web standard used to describe and model information. RDF consists of "triples," or statements, with a subject, predicate, and object that resemble an English sentence; RDF data is then stored in what are known as "triple-store graph databases". RDF is a W3C standard for representing information, with common serializations, and is the foundation for a mature framework of related standards such as RDFS and OWL that are used in ontology and knowledge graph development. RDF and its related standards are queried using SPARQL, a W3C recommended RDF query language that uses pattern matching to identify and return graph information.

LPG

A Labeled Property Graph (LPG) is a data model for graph databases that represents data as nodes and edges in a directed graph. Within an LPG, nodes and edges have associated properties such as labels that are modeled as single value key-value pairs. There are no native or centralized standards for the creation of LPGs; however, the Graph Query Language (GQL), an ISO standardized query language released in April 2024, is designed to serve as a standardized query template for LPGs. Because GQL is a relatively recent standard, it is not yet adopted by all LPG databases.

What does this mean? How are they different?

There are a number of differences between RDF graphs and LPGs, some of which we will get into. At their core, though, the differences between RDF and LPG stem from different approaches to information capture.

RDF and its associated standards put a premium on defining a conceptual model, applying this conceptual model to data, and inferring new information using category theory and first order logic. They are closely tied to standards for taxonomies and linked data philosophies of data reuse and connection.

LPGs, by contrast, are not model-driven, and instead are more concerned with capturing data rather than applying a schema over it. There is less of a focus on philosophical underpinnings and shared standards, and more importance given to the ability to traverse and mathematically analyze nodes in the graph.

Specific Benefits & Drawbacks of Each

RDF

Plusses:

- Self-Describing: RDF describes both data and the data model in the same graph
- Data Validation: RDF can validate data and data models using SHACL, a W3C standard
- Expressivity: RDF and its larger semantic family is well suited to capturing the logical underpinnings and human understanding of a subject area.
- Flexible Modeling: RDF was originally designed for web use cases in which multiple data schemas / sources of truth are aggregated together. Due to this flexibility, RDF is useful in aligning schemas and querying across heterogeneous / different datasets, as well as metadata management and master data management
- Global Identifiers: Entities in the graph are assigned (resolvable) URIs. This has enabled the creation of open source models for both foundational concepts such as provenance and time, as well as domain specific models in complex subject areas like Process Chemistry and Finance that can be utilized and reused.
- Standardization: Wide standard implementation enables simple switching between vendor solutions
- Native Reasoning: OWL is another W3C standard built on RDF that enables logical reasoning over the graph using category theory

Minuses:

- High Cognitive Load: Due to the mathematical and philosophical underpinnings it can take more time to come up to speed on how to model in RDF and OWL
- Complexity of OWL Implementations: There are a number of different standards for how to implement OWL reasoning, and it is not always clear even to some experienced modelers which should be used when
- N-ary Structures: RDF cannot model many-to-many relationships. Instead, intermediary structures are required, which can increase the verbosity of the graph.
- Property Relations: Relationships cannot be added to existing properties in base RDF, restricting the kinds of statements that can be made. An RDF standard to extend this functionality, RDF*, is available in some triple-stores but is still under development and not consistently offered by vendors.

LPG

Plusses:

- Efficient Storage: LPGs are generally more performant with large datasets, and frequently updated data compared to RDF
- Graph Traversal: LPGs were designed for graph traversal to facilitate clustering, centrality, shortest path, and other common graph algorithms to perform deep data analysis.
- Analytics Libraries: There are a number of open source machine learning and graph algorithm libraries available for use with LPGs.
- Developer-Friendly: LPGs are often a first choice for developers since LPGs' data-first design and query languages more closely align to preexisting SQL expertise.
- Property Relations: LPGs support the ability to attach relationships on properties natively.

Minuses:

- No Formal Schema: There is not a formal mechanism for enforcing a data schema on an LPG. Without a validation mechanism to ensure adherence to a model, the translation of data into entities and connections can become fuzzy and difficult to verify for correctness.
- Vendor Lock-In: Tooling is often proprietary, and switching between LPG databases is difficult and inflexible due to the lack of a common serialization and proliferation of proprietary languages.
- Lack of Reasoning: There are no native reasoning capabilities for logical inferences based on class inheritance, transitive properties, and other common logical expressions, although some tools have plug-ins to enable basic inference.

Common Questions

Which do I use for a knowledge graph?

Although some organizations define knowledge graphs as being built upon RDF triple stores, you can use either RDF or LPG to develop a knowledge graph so long as you apply and enforce adherence to a knowledge model and schema over your LPG. Managing and applying a knowledge model is easier within RDF, so it is often the first choice for knowledge graphs, but it is still doable with LPGs. For example, in his book Semantic Modeling for Data, Panos Alexopoulos references using Neo4j, an LPG vendor, to represent and store a knowledge graph.

Is it easier to use an LPG?

LPGs have a reputation for being easier to use because they do not require you to begin by developing a model, unlike RDF, allowing users to quickly get started and stand up a graph. This does not necessarily translate to LPGs being easier to use over time, however. Modeling up front helps to solve data governance questions that will come up later as a graph scales. Ultimately, data governance and the need for a graph to reflect a unified view of the world, regardless of format, mean that the work which happens to model up-front in RDF also ends up happening over the lifetime of an LPG.

Which do I need to support an LLM with RAG?

Graph-RAG is a design framework that supports an LLM by utilizing both vector embeddings and a knowledge graph. Either an LPG or an RDF graph can be used to power Graph-RAG. Semantic RAG is a more contextually aware variant that uses a small amount of locally stored vector embeddings and an RDF data graph with an RDF ontology for its semantic inference capabilities.

Do I have to choose between RDF and LPG when creating a graph?

It depends. We have seen larger enterprises embrace both in instances where they want to take advantage of the pros of each. For example, utilizing an RDF graph for data aggregation across sources, and then pulling the data from the RDF graph into an LPG for data analysis. However, if you are within a single graph database tool/application, you will be required to choose which standard you want to use. Although there are graph databases that allow you to store either RDF or LPG, such as Amazon Neptune, these databases lock you into RDF or LPG once you select a standard to use for storage. Neptune does allow users to query over data using both SPARQL and property graph query languages, which bridges some of the gaps in RDF and LPG functionality. As of the time of writing, however, Neptune is less feature rich for RDF and LPG data management than comparable purely RDF or purely LPG databases such as GraphDB and Neo4J.

Can I use both?

You can use RDF and LPGs together, but there are manageability concerns when doing so. Because there are no formal semantic standards for LPGs in the same way as there are for RDF, it is generally destructive to move data from an LPG into an RDF graph. Instead, the RDF graph should be used as a source of logical reasoning information using constructs like class inheritance. Smaller portions of the RDF graph, called subgraphs, can then be exported to the LPG for use with graph-based ML and traversal-based algorithms.

See Appendix for a sample architecture that utilizes both RDF and LPG for entity resolution.

Which should I choose if I want to use programming languages like Python and Java?

Both RDF and LPG ecosystems offer robust support for both Java and Python, each with mature libraries and dedicated APIs tailored to their respective data models. For RDF, Java developers can leverage tools like RDF4J, which provides comprehensive support for constructing, querying (via SPARQL), and reasoning over RDF datasets, while Python developers benefit from RDFlib's simplicity in parsing, serializing, and querying RDF data. In contrast, LPG databases such as Neo4j deliver specialized libraries—Neo4j's native Java API and Python drivers like Py2neo or the official Neo4j Python driver—that excel at handling graph traversals, pattern matching, and executing graph algorithms. Additionally, these LPG tools often integrate with popular frameworks (e.g., Spring Data for Java or NetworkX for Python), enabling more sophisticated data analytics and machine learning workflows.

How should I choose between RDF and LPG?

How are you answering business use cases with the graph? What kind of queries will you be asking/running? That will determine which graph format best fits your needs. Regardless of model or standard, when defining a graph the first thing to do is to determine personas, use cases, requirements, and competency questions. Once you have these, particularly requirements and competency questions, you can determine which graph form best fits your use case(s). To help clarify this, we have a list of use case-based rules of thumb.

Use Case Rules of Thumb

See Appendix for guidance on use cases between LPG and RDF.

Conclusion

Both RDF and LPGs have relative strengths and weaknesses, as well as preferred use cases. LPGs are suited for big data analytics and graph analysis, while RDF are more useful for data aggregation and categorization. Ultimately, you can build a knowledge graph and semantic layer with either, but how you manage it and what it can do will be different for each. If you have more questions on RDF and LPG, reach out to EK with questions and we will be happy to provide additional guidance.

Appendix

Figure One: Sample Architecture

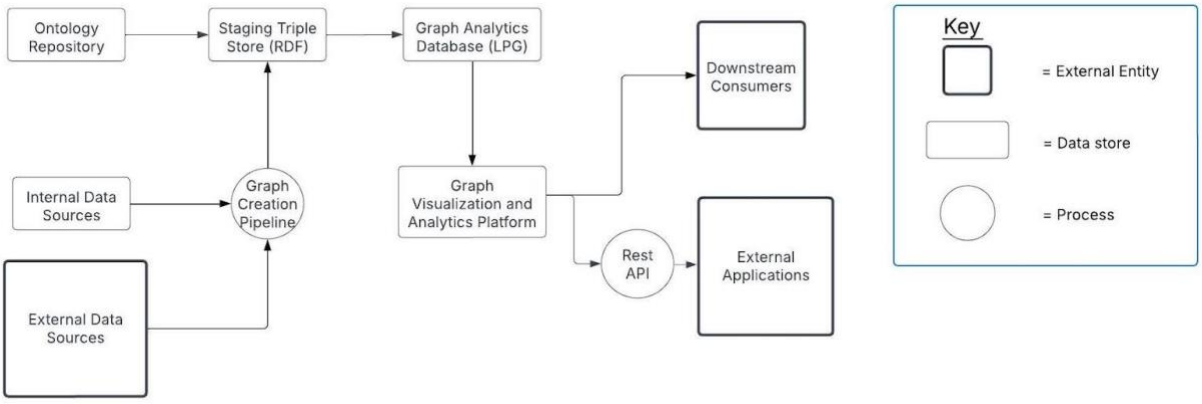

Figure Two: Use Case Rules of Thumb