OSTHUS

Let's talk about Graph Data models and why they are relevant for your Data and IT Strategy

AUTHOR

Harsh Thakker, PhD.

in

Issued Date

August 2022

+49 241 94313 0

office@osthus.com

www.osthus.com

Table of Content

Objective

To all graph enthusiasts, researchers, students and professionals out there, who would love to learn more about Graph Data models: This whitepaper is for you.

Feel free to use it as a reference for your own papers, as training material or use cases – We hope the content serves you well!

But let me be a bit more precise: The objective of this white paper is to familiarize the readers with the graph data model and its advantages; specifically the Resource Description Framework (RDF) and Property graphs (PG) models which are the two most popular graph-data modeling approaches amongst the commercial and open-source community. In doing so, we introduce and illustrate the concepts of Resource Description Framework (RDF), Property graphs (PG), Ontology, Semantics, and other relevant terminology.

We conclude this article by presenting a selected list of use cases/scenarios where adopting a graph-data model (compared to OR in conjunction with an existing relational database setup) will open doors to reaping several application specific design and performance benefits.

So let’s dive in.

Graph Data Models – Real world fact capturing at its best

Looking at the evolution of information technology, one can observe a trend from data models and knowledge representation techniques that are tightly coupled to the capabilities of the underlying hardware to more intuitive and natural methods that resemble human-style information processing.

This evolution began with machine assembly languages, progressed to procedural programming, object-oriented methods, and resulted in the increasingly loose coupling of data and code with relational databases, declarative query languages, and object-relational mapping (ORM). In recent years, another step in this evolution has been observed: graph-based data models that organize information into conceptual networks. When choosing formalisms for modeling real-world scenarios such as biological, transportation, communication, and social networks, graphs are particularly popular because of their intuitive data model.

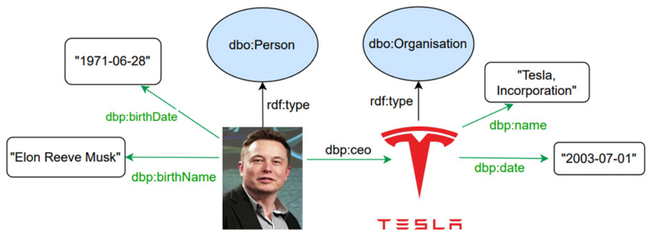
Resource Description Framework (RDF) and Property Graph (PG) data models are two most popular approaches for data management that are based on modeling, storing, and querying graph-like data. Several database systems/vendors based on these models are gaining relevance in the industry due to their use in several domains where graphs and network analytics are required. Below is the landscape of large-scale adoption of graph-based data models by a variety of public and private organizations as consolidated by Frank Van Harmelen in 2019.

The RDF Data Model – Real world facts put in a data model

The RDF data model is a well-known W3C standard, which is used for data modeling and encoding machine-readable content on the Web and within intranets. Using RDF one can capture real-world facts (data, e.g. Elon Musk is the CEO of Tesla Inc.) and annotate statements about those facts (so called meta-data, e.g. Elon Musk is CEO of Tesla Inc. since 2008). These facts are represented in RDF as RDF triples. An RDF triple is a tuple (s, p, o) where s is called the subject, p is the predicate, and o is the object. Here, the subject and object can be interpreted as entities (nodes) and the predicate represents the relationship between them (edge). An example of RDF triples is shown on the right capturing the relationship between Elon Musk and Tesla Inc. in turtle format:

The figure below shows a graphical representation of an RDF graph of the data on the right.

```
@prefix dbr: <http://dbpedia.org/resource/>.
@prefix dbo: <http://dbpedia.org/ontology/>.
@prefix dbp: <http://dbpedia.org/property/>.
dbr:Elon_Musk dbp:ceo dbr:Tesla_Inc .
dbr:Elon_Musk dbp:birthDate “1971-06-28” .
dbr:Elon_Musk dbp:birthName “Elon Reeve Musk” .
dbr:Tesla_Inc rdf:type dbo:Organisation .
dbr:Tesla_Inc dbp:name “Tesla Incorporation” .
dbr:Tesla_Inc dbp:date “2003-07-01” .
```



i

We use green color to represent the relationships in RDF (i.e. edges and property keys in PG) and blue color to represent classes (and nodes in PG). The literals (or property values in the PG) are represented in black color.

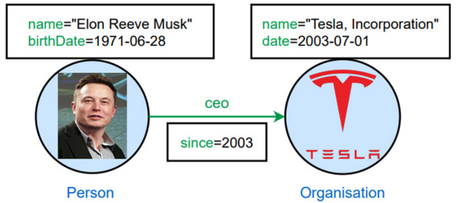
The Property Graph Data Model

A Property Graph (PG) is defined as a labeled directed multigraph whose main characteristic is that nodes and edges can contain a set (possibly empty) of name-value pairs referred to as properties.

From the point of view of data modeling, each node represents an entity, each edge represents a relationship (between two entities), and each property represents a specific characteristic (of an entity or a relationship). An example of a Property Graph (data), corresponding to the RDF example above, in GraphML format is shown on the next page.


```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<key id="a0" for="node" attr.name="label" attr.type="string"/>
<key id="a1" for="node" attr.name="name" attr.type="string"/>
<key id="a2" for="node" attr.name="birthDate" attr.type="Date"/>
<key id="a3" for="node" attr.name="age" attr.type="int"/>
<key id="a4" for="node" attr.name="birthPlace" attr.type="string"/>
<key id="a5" for="node" attr.name="founded" attr.type="string"/>
<key id="a6" for="node" attr.name="hq" attr.type="string"/>
<key id="a7" for="node" attr.name="employees" attr.type="int"/>
<key id="a8" for="edge" attr.name="label" attr.type="string"/>
<key id="a9" for="edge" attr.name="since" attr.type="int"/>
<graph id="G" edgedefault="directed">
<node id="n0">
<data key="a0">Person</data>
<data key="a1">Elon Reeve Musk</data>
<data key="a2">1971-06-28</data>
<data key="a3">46</data>
<data key="a4">South Africa</data>
</node>
<node id="n1">
<data key="a0">Organisation</data>
<data key="a1">Tesla, Incorporation</data>
<data key="a5">2003-07-01</data>
```

The figure below presents a graphical representation of a Property Graph corresponding to the Elon Musk example. The circles represent nodes, the arrows represent edges, and the boxes contain the properties (key-value pairs) for nodes and edges.



Semantics

Capturing the meaning of data

(Web) Semantics is the study of the meaning of concepts and their representation, through which one can add context to statements about entities.

Ontology and RDF Schema

An Ontology, is a model that captures an aspect (including meaning of things) of the real world. Ian Horrocks defines it as follows -- An ontology:

Introduces a vocabulary that is relevant to the domain

i.e often includes names of classes and relationships

Specifies an intended meaning of the vocabulary (semantics) of the vocabulary

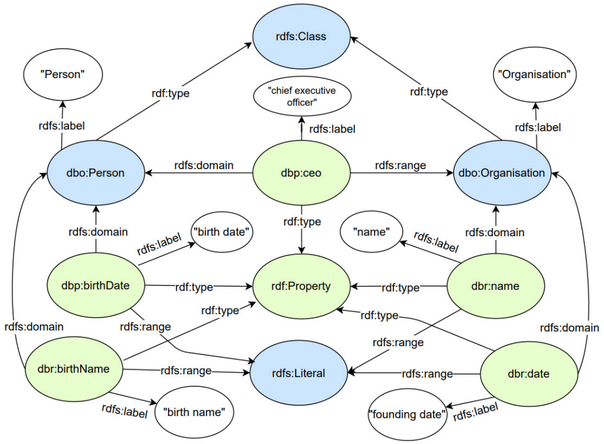
Formalized using a suitable logic, e.g. OWL (Web Ontology Language) formalized using description logic

Consists of two parts

1. Set of axioms describing the structure of the model (using e.g. RDF Schema)
2. Set of facts describing a particular concrete situation (e.g. Elon Musk is a Person)

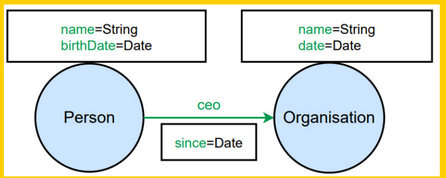
RDF Schema (RDFS) defines a standard data modeling vocabulary (i.e., a set of terms, each having a well-defined meaning), which enables the description of resource classes and property classes.

From a database perspective, RDF Schema allows us to define the structure of the data in an RDF graph, i.e., a schema for RDF data. The figure below shows the schema of the running example capturing the relationship between Elon Musk and Tesla Inc. resemble human-style information processing.



PG Schema

In the context of property graphs, there isn't a notion of schema in practice, as is in the case of RDF and relational databases. Although, there exist some systems that use the notions of node types and edge types. As with RDF databases, the schema allows us to define and validate the structure of a property graph.
The notion of a schema in PG is best illustrated by the following figure.



The Key Differences between RDF and PG Data Models

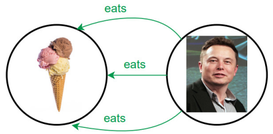
Despite both RDF and PG being graph-based data models, there are some key differences between the two. Some of these are at the fundamental level and others at the vendor implementation level:

RDF doesn't allow unique identification of relationship instances of the same type

- In RDF, it's not possible to uniquely identify instances of a relationship. Meaning that, it's not possible to have connections of the same type between the same pair of nodes because that would represent exactly the same triple, with no extra information.e.g. You cannot have the triple with predicate "eats" (Elon eats ice_cream) three times in RDF (as it is essentially the same triple); whereas in PG it is possible to have the same type of edge three times between the same pair of nodes.

- This makes a huge difference while answering a question like "How many times does Elon eat ice cream?"; Using SPARQL COUNT() you will get 1; whereas using count() function in a language like Gremlin or Cypher you get the answer 3[2].


```
dbr:Elon_Musk dbp:eats dbr:Ice_cream
dbr:Elon_Musk dbp:eats dbr:Ice_cream
dbr:Elon_Musk dbp:eats dbr:Ice_cream
```




```
g.V(elon).addE('eats').to(icecream)
g.V(elon).addE('eats').to(icecream)
g.V(elon).addE('eats').to(icecream)
```

[2] Please note there is a caveat around this in RDF by introducing meta-data using reification, which allows you to capture the information about the eating counts. The idea here is to demonstrate that it isn't supported without using additional triples.

- Unlike RDF there is no standard data format for encoding PGs (GraphML, GRYO, YARS-PG, GML, etc.). This presents a serious challenge for supporting interoperability between several graph systems.
- Most RDF serializations are triple-centric, while most PG serializations represent graphs as lists of nodes and edges
- RDF can have multivalued properties and the labeled property graph can have arrays.
  - In RDF you can have multiple triples with the same subject and predicate and different objects. In PG, this is equivalent to using arrays (the same key having multiple values)

SPARQL

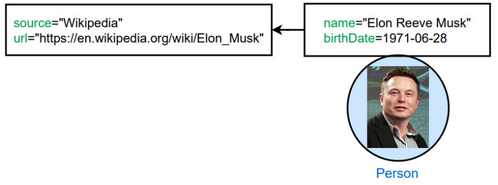
```
PREFIX dbp: <http://www.dbpedia.org/property>
INSERT DATA {
dbr:Elon_Musk dbp:birthName "Elon Reeve Musk" .
dbr:Elon_Musk ex:likes "Space Travel" .
dbr:Elon_Musk ex:likes "Nikola Tesla" . }
```

birthName="Elon Reeve Musk"
likes=["Space Travel","Nikola Tesla"]

GREMLIN

```
v = g.addV().property('birthName','Elon Reeve
    Musk').property('likes','Space
    Travel').property('likes','Nikola
    Tesla').next()
```

- The RDF data model allows metadata about properties, i.e. it allows for edges between edges (a predicate can be the subject in another triple). Although it must be noted that this feature is not common in real-world use cases. A PG does not allow edges between edges. However, graph data serializations such as .gryo (used by Apache Tinkerpop) allow supporting of multi-level metadata by supporting meta-properties. This allows one to annotate – for instance provenance data, about a particular fact expressed by a property. In the example below, we can see that the birth date information about Elon Musk is sourced from the corresponding Wikipedia page about him.



Why should you adopt a RDF/PG-based Approach vs Relational Data Model?

For the sake of Schema Flexibility and Metadata Management

If your business problem or use case requires that the data schema evolves with the maturity of the application and it's requirements (over time). Unlike relational databases which demand that the schema exists upfront and the data has to be consistent (or satisfy) the schema constraints, Graph data models offer a very flexible notion of schema.

To ensure FAIR Data Principles, Data, and Terminology Standardization

If the goal of your business problem or use case is to adopt, enforce and adhere to the FAIR data principles - making your data and data assets Findable, Accessible, Interoperable, and Reusable;

You might want to consider leveraging the RDF technology stack along with your existing data warehouse/lakehouse/etc. infrastructure. RDF, SPARQL, and OWL being W3C standards are widely adopted by commercial vendors and cloud-solution providers. Furthermore, the whole vision behind semantic web (RDF, SPARQL, OWL, etc. technology stack) is to uniquely identify and describe resources on the web. This design feature allows eliminating redundancy in the naming, identification, accessibility, and reusability of existing resources. Web Ontologies provide a standardized and concrete framework to classify, associate, and describe concepts and terminologies in both closed- and open domains. For e.g., the popular and publicly available Chemical Entities of Biological Interest (ChEBI) ontology, can be used to identify and associate chemical compounds in a chem- or bio-informatics use case.

If you like high performance in your complex analytical Queries

Unlike the execution of SQL over relational databases, graph query languages such as Gremlin or Cypher do not have to perform a large number of "joins" in order to evaluate a complex query. Thus, if your business problem or use case envisions queries of analytical nature (analytical operators) over a large number of different classes (nodes or entities), adopting a PG-based data model would be your best bet. Consider building a recommendation engine for finding the nearest match for a chemical or biological compound across your vast data landscape. Graph traversals are order(s) of magnitude faster than a traditional relational data model (and SQL) approach.

You are looking for a 360-degree holistic view of your data and business assets, aka a Centralized Knowledge/ Data Management Layer

If you would like to design and develop an overarching central data layer (or knowledge management) for your enterprise in order to allow a 360-degree holistic view of your data and business assets, you might want to consider the deployment of an enterprise knowledge graph. Using standardized vocabularies/taxonomies/ontologies one can create a multi-layered data architecture for enabling integrated global data governance, data insight, and business intelligence ecosystem.

Build and leverage the power of AI/ML/NLP based Applications

If your business problem or use case is building an application based on machine learning, natural language processing (or targeting any other artificial intelligence-related problem), such as chatbots, open-or-closed domain question answering systems, or NLP-based systems, check out the current state of the art in graph data-based machine learning solutions. Both RDF and PG databases are catering to these needs providing native support for building modular pipelines using a custom implementation of various AI, ML, NLP libraries, and plugins.

Summary

To summarize, the adoption of a graph data model based approach is of paramount importance, both performance and modeling flexibility wise, if the goal of the use case/application is to provide a standardized means of data production, consumption, governance, and lineage. It goes without saying that graph databases, be it RDF triplestores or PG databases, are not a silver bullet for addressing all data and IT strategy challenges. For instance, while graph databases provide native support for high performance graph computation, discovery and lineage style queries (highly relevant for data cataloguing, governance, standardization/stewardship, adopting FAIR principles), they still do lack a tried and tested commercial support for ACID properties (critical for transactional systems/use cases)[3].

They should not be looked as a one-stop replacement for existing data warehouses, data lakes, data marts, etc.; instead they should be looked at as an natural extension layer in for supporting analytical and machine learning based use cases/applications on top of the existing data & IT infrastructure.

To put it in a simpler language – one should use a sharpened knife to chop vegetables, fruits, etc. and a top-notch axe to chop down trees[4]. Using a combination of both tools gives your organization a wider set of possibilities and competence to exploit. Therefore, when driving organizational data and IT strategy one should consider the entire toolkit and leverage the best available tool to tackle a particular use case based on its merits.

If you are interested in learning more about the RDF and PG data models, their real-world use cases (who uses them and for what?), and a variety of cloud-based solutions you can use to address your complex data-driven business needs, stay tuned to check out our upcoming second white paper on this topic.

[3] Neo4J does support transactional capabilities to the best out understanding, but here we refer to the broader spectrum of graph vendors covering both RDF and PG data models.
[4] We obviously do not recommend that you chop down trees! It is to make a point here ;-)

OSTHUS

Get In Touch

+49 241 94314 0
www.osthus.com
office@osthus.com
Eisenbahnweg 9-11,
52068 Aachen