Knowledge Graphs vs. Property Graphs — A Brief Overview and Comparison
FOUNDATION
- TopQuadrant was founded in 2001
- Strong commitment to standards-based approaches to data semantics

MISSION
- Empower people and drive results — by making enterprise information meaningful

FOCUS
- Provide comprehensive data governance solutions using knowledge graph technologies
Today’s Agenda

I. A brief overview of:
   – Property Graphs
   – Knowledge Graphs

II. Differences in Terminology and Capabilities of these models

III. Strengths and Limitations of each approach

IV. Why Knowledge Graphs provide a strong foundation for data centric enterprise applications
   – E.g. enterprise data governance and metadata management

▪ Q & A
Some Definitions of a Knowledge Graph

- A **knowledge graph** (i) mainly describes real world entities and their interrelations, organized in a graph, (ii) defines possible classes and relations of entities in a schema, (iii) allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains.


- A **knowledge graph** acquires and integrates information into an ontology and applies a reasoner to derive new knowledge.


- A **knowledge graph** is a **knowledge base** that uses a graph-structured **data model** or topology to integrate knowledge and data. Knowledge graphs are often used to store interlinked descriptions of entities — real-world objects, events, situations or abstract concepts, including their semantics.

  *Wikipedia*
TopQuadrant: What Are Knowledge Graphs?

- A Knowledge Graph represents a knowledge domain
  - With “Active Models” that can be consulted at run-time

- It represents all information as a graph
  - A network of nodes and links
  - Not tables of rows and columns

- It represents facts (data) and models (metadata) in the same way – as part of a graph
  - Rich rules and inferencing

- It is based on open standards, from top to bottom
  - Readily connects to knowledge in private and public clouds

A knowledge graph, with its Active Models, grows and evolves over time.
Knowledge Graph Recipe

An Ontology is a data model (schema) with classes, properties, and rules - a precise blueprint for data
Property Graphs
aka Labeled Property Graphs

- **Caveat**: There is no standard property graph data model
  - Each implementation of a Property Graph is, thus, somewhat different
  - We will focus on the common aspects

- In general, the Property Graphs data model consists of three elements:
  - **Nodes** – entities in a graph
  - **Edges** – direct links aka relationships between nodes
  - **Properties** – key value pairs that can be associated with either nodes or edges
Property Graph by Example

```
"Name" The Post
"Released" 2017

"ID" 125
label: Movie

"ID" 10
"Role" Ben Bradlee

"ID" 123
"First Name" Tom
"Last Name" Hanks
"Year Born" 1956

label: Person
label: Director
label: Actor
```

- "ID" 125 is a movie node.
- "ID" 10 is a person node that has an "ACTED_IN" relationship with "ID" 125.
- "ID" 123 is another person node that also has an "ACTED_IN" relationship with "ID" 125.
Property Graph by Example
Property Graph by Example
RDF Knowledge Graph by Example
### Property Graph vs RDF Knowledge Graph

<table>
<thead>
<tr>
<th>Property Graph</th>
<th>Knowledge Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs are internal to a graph database, user has no control over them.</td>
<td>IDs are global – URIs, meant to be under users control to enable combining different graphs</td>
</tr>
<tr>
<td>Properties are literal values. They are fundamentally different from nodes and relationships.</td>
<td>Canonical structure. Everything is stored as nodes and links connecting them. A literal value is a node like any other. Property is any link – to a resource or a literal.</td>
</tr>
</tbody>
</table>
“Schema” as part of a Knowledge Graph
“Schema” as part of a Knowledge Graph
More "Schema"
Standards-based Ontologies and Vocabularies are Knowledge Graphs Increasingly Used across Industries

These ontologies uniquely identify and describe things and relationships in healthcare, web search services, manufacturing, financial services, …

Key enablers of transparency, interoperability and the emerging “cognitive enterprise”
## Property Graph vs RDF Knowledge Graph

<table>
<thead>
<tr>
<th>Property Graph</th>
<th>Knowledge Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs are internal to a graph database, user has no control over them.</td>
<td>IDs are global – URIs, meant to be under user’s control to enable combining different graphs.</td>
</tr>
<tr>
<td>Properties are literal values. They are fundamentally different from nodes and relationships.</td>
<td>Canonical structure. Everything is stored as nodes and links connecting them. A literal value is a node like any other. Property is any link - to a resource or a literal.</td>
</tr>
<tr>
<td><strong>Schema (semantics of data) is not a part of the graph.</strong></td>
<td><strong>Rich schemas, including rules are a part of the graph.</strong></td>
</tr>
</tbody>
</table>
RDF Knowledge Graphs: How to add information to triples
## Property Graph vs RDF Knowledge Graph

<table>
<thead>
<tr>
<th>Property Graph</th>
<th>Knowledge Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs are internal to a graph database, user has no control over them.</td>
<td>IDs are global – URIs, meant to be under user’s control to enable combining different graphs.</td>
</tr>
<tr>
<td>Properties are literal values. They are fundamentally different from nodes and relationships.</td>
<td>Canonical structure. Everything is stored as nodes and links connecting them. A literal value is a node like any other. Property is any link - to a resource or a literal.</td>
</tr>
<tr>
<td>Schema (semantics of data) is not a part of the graph.</td>
<td>Rich schemas, including rules are a part of the graph.</td>
</tr>
<tr>
<td>Each relationship (link or arch) uniquely identifies “node – link – node” combination. Relationships can be annotated with additional facts, but properties can’t be.</td>
<td>IDs of properties (links) are re-used. Thus, they do not uniquely identify a “node – link - node” combination that uses it. There is a way to give these triples identity. Any triple can be annotated with additional facts.</td>
</tr>
</tbody>
</table>
Property Graphs:
How to add information to a “property” value

“Name” White Plains
“Population” 58811

“ID” 127

label: Location
label: City

“Name” White Plains
“Population” (58811, 56853)

“ID” 127

label: Location
label: City

Can we say what year is the population for?
Property Graphs:
How to add information to a “property” value

Can’t be done. Must turn a “property” into a relationship

“Name” White Plains
“ID” 127
label: Location
label: City

type: POPULATION_SIZE

“Size” 58811
“Year” 2018
“ID” 131
label: Population

“Size” 56853
“Year” 2010
“ID” 132
label: Population

© Copyright 2020 TopQuadrant Inc.
RDF Knowledge Graph: How to add information to a triple with a literal object

Or, as it more commonly said, “to a property with a literal value”
Property Graph: How to use a node as a “property” value

Let’s say we want to capture information about Ben Bradlee so we want to turn the value of “role” into a node.
Property Graph: How to use a node as a “property” value

Must remove the edge annotation and create a relationship to a node labeled Person.
RDF Knowledge Graph: How to turn a literal value into a resource reference
RDF Knowledge Graph: How to turn a literal value into a resource reference

Figure 9: RDF Graph with Ben Bradlee as a Person
<table>
<thead>
<tr>
<th>Property Graph</th>
<th>Knowledge Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs are internal to a graph database, user has no control over them.</td>
<td>IDs are global – URIs, meant to be under user’s control to enable combining different graphs.</td>
</tr>
<tr>
<td>Properties are literal values. They are fundamentally different from nodes and relationships.</td>
<td>Canonical structure. Everything is stored as nodes and links connecting them. A literal value is a node like any other. Property is any link - to a resource or a literal.</td>
</tr>
<tr>
<td>Schema (semantics of data) is not a part of the graph.</td>
<td>Rich schemas, including rules are a part of the graph.</td>
</tr>
<tr>
<td>Each relationship (link or arch) uniquely identifies “node – link – node” combination. Relationships can be annotated with additional facts, but properties can’t be.</td>
<td>IDs of properties (links) are re-used. Thus, they do not uniquely identify a “node – link - node” combination that uses it. There is a way to give these triples identity. Any triple can be annotated with additional facts.</td>
</tr>
<tr>
<td>Changes in the graph design require restructure/re-load of the data and changes to all impacted queries.</td>
<td>Graphs can evolve and changes in the design can often be done with minimal impact on existing data and queries.</td>
</tr>
<tr>
<td>Product-specific query languages, variants of Cypher, increasingly, GraphQL support, introspection not integrated</td>
<td>Query standard – SPARQL. Increasingly, GraphQL support. In EDG: introspection and auto-generation of GraphQL Schemas.</td>
</tr>
</tbody>
</table>
Example of RDF Serialization

@prefix entertainment_data: <http://example.org/data-graphs/Entertainment_Data#> .
@prefix entont: <http://example.org/ontologies/Entertainment#> .
@prefix wikidata: <http://www.wikidata.org/entity/> .
@prefix schema: <http://schema.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

wikidata:Q2263

entont:Actor ;
entont:Director ;
entont:actedIn entertainment_data:ThePost

[[ entont:role "Ben Bradlee" ]] ;
schema:birthDate "1956-07-09"^^xsd:date ;
entont:directed entertainment_data:ALeagueOfTheirOwn ;
schema:familyName "Tom" @en;
schema:givenName "Hanks" @en;
rdfs:label "Tom Hanks" @en; .
## Property Graph vs RDF Knowledge Graph

<table>
<thead>
<tr>
<th>Property Graph</th>
<th>Knowledge Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs are internal to a graph database, user has no control over them.</td>
<td>IDs are global – URIs, meant to be under user’s control to enable combining different graphs.</td>
</tr>
<tr>
<td>Properties are literal values. They are fundamentally different from nodes and relationships.</td>
<td>Canonical structure. Everything is stored as nodes and links connecting them. A literal value is a node like any other. Property is any link - to a resource or a literal.</td>
</tr>
<tr>
<td>Schema (semantics of data) is not a part of the graph.</td>
<td>Rich schemas, including rules are a part of the graph.</td>
</tr>
<tr>
<td>Each relationship (link or arch) uniquely identifies “node – link – node” combination. Relationships can be annotated with additional facts, but properties can’t be.</td>
<td>IDs of properties (links) are re-used. Thus, they do not uniquely identify a “node – link - node” combination that uses it. There is a way to give these triples identity. Any triple can be annotated with additional facts.</td>
</tr>
<tr>
<td>Changes in the graph design require restructure/re-load of the data and changes to all impacted queries.</td>
<td>Graphs can evolve organically. Changes in the design can often be accomplished with minimal impact on existing data and queries.</td>
</tr>
<tr>
<td>Product-specific query languages, variants of Cypher, increasingly, GraphQL support, introspection not integrated</td>
<td>Query standard – SPARQL. Increasingly, GraphQL support. In EDG: introspection and auto-generation of GraphQL Schemas.</td>
</tr>
</tbody>
</table>

| No standard serialization for export.                                         | Standard serializations supported by all products – RDF/XML, Turtle, N3 and JSON-LD formats. |

© Copyright 2020 TopQuadrant Inc.
Knowledge Graphs are Self-Organizing – they can be composed
Knowledge Graphs are Self-Organizing – they can be composed

With consistent Identifiers for both Nodes and Links, Graph queries and rules can give insights and make new knowledge.
## Property Graph vs RDF Knowledge Graph

<table>
<thead>
<tr>
<th>Property Graph</th>
<th>Knowledge Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDs are internal to a graph database, user has no control over them.</td>
<td>IDs are global – URIs, meant to be under user’s control to enable combining different graphs.</td>
</tr>
<tr>
<td>Properties are literal values. They are fundamentally different from nodes and relationships.</td>
<td>Canonical structure. Everything is stored as nodes and links connecting them. A literal value is a node as any other. Property is any link - to a resource or a literal.</td>
</tr>
<tr>
<td>Schema (semantics of data) is not a part of the graph.</td>
<td>Rich schemas, including rules are a part of the graph.</td>
</tr>
<tr>
<td>Each relationship (link or arch) uniquely identifies “node – link – node” combination. Relationships can be annotated with additional facts, but properties can’t be.</td>
<td>IDs of properties (links) are re-used. Thus, they do not uniquely identify a “node – link - node” combination that uses it. There is a way to give these triples identity. Any triple can be annotated with additional facts.</td>
</tr>
<tr>
<td>Changes in the graph design require restructure/re-load of the data and changes to all impacted queries.</td>
<td>Graphs can evolve organically. Changes in the design can often be accomplished with minimal impact on existing data and queries.</td>
</tr>
<tr>
<td>Product-specific query languages, variants of Cypher, increasingly, GraphQL support, introspection not integrated.</td>
<td>Query standard – SPARQL. Increasingly, GraphQL support. In EDG: introspection and auto-generation of GraphQL Schemas.</td>
</tr>
<tr>
<td>No standard serialization for export.</td>
<td>Standard serializations supported by all products – RDF/XML, Turtle, N3 and JSON-LD formats.</td>
</tr>
<tr>
<td><strong>No partitioning of graphs nor merging or connectivity across graphs.</strong></td>
<td><strong>Named graphs as a partitioning approach. Graphs are “self-composing”, can be connected and merged.</strong></td>
</tr>
</tbody>
</table>
Knowledge Graphs support the move to Data Centric Systems and Architectures
<table>
<thead>
<tr>
<th>Application Centric</th>
<th>Data Centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data is a second-class citizen, and the application is the main thing</td>
<td>Data is the main asset, applications are “model-driven”</td>
</tr>
<tr>
<td>Data models are not self describing, can’t be understood without understanding application’s logic</td>
<td>Data is self-describing and can stand on its own</td>
</tr>
<tr>
<td>All business rules and semantics are hard coded in the applications</td>
<td>Data can be “active” with accessible semantics, business rules and even executable code</td>
</tr>
<tr>
<td>Data models are very rigid</td>
<td>Data models are flexible and evolvable</td>
</tr>
<tr>
<td>Changing application’s data model is expensive – requires changes to the application code due to tight coupling</td>
<td>Changing data models is inexpensive and may not require application changes</td>
</tr>
<tr>
<td>One data model per application</td>
<td>Multiple applications may share some parts of a data model, extending it as needed</td>
</tr>
<tr>
<td>Data sharing is an afterthought</td>
<td>Data sharing is “built in”</td>
</tr>
</tbody>
</table>
Why Self Describing Data is Important?

Open Data Example: Road Safety - Vehicles by Make and Model

We see the numbers, but what do they mean?

Data is only as good as its metadata
### Metadata adds Context & Definition – Making Data sharable and increasing its Value

**Customer**

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Company</th>
<th>City</th>
<th>Year Purchased</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Smith</td>
<td>Komputers R Us</td>
<td>New York</td>
<td>1970</td>
</tr>
<tr>
<td>Mary</td>
<td>Jones</td>
<td>The Lord’s Store</td>
<td>London</td>
<td>1999</td>
</tr>
<tr>
<td>Proful</td>
<td>Bishwal</td>
<td>The Lady’s Store</td>
<td>Mumbai</td>
<td>1998</td>
</tr>
<tr>
<td>Ming</td>
<td>Lee</td>
<td>My Favorite Store</td>
<td>Beijing</td>
<td>2001</td>
</tr>
</tbody>
</table>

Is this the city where the customer lives or where the store is located?

---

**Key Facts**

- **If metadata is not readily available, data can’t be understood.**
- **If metadata is kept separate from data, applications can’t access it.**
- **Further, data will change and metadata will get stale.**
- **If metadata is captured in a proprietary way, it is not open.**

**Definition**

Last Name represents the surname or family name of an individual.

**Business Rules**

In the Chinese market, family name is listed first in salutations.

**Format**

VARCHAR(30)

**Abbreviation**

LNAME

**Required**

YES

**More ...**

Numerous technical & business metadata including security, privacy, permissible values, etc.
TopBraid EDG – Composition of Knowledge Graphs to Support Adaptive Data Governance

- Enterprise Assets
- Technical Assets
- Big Data Assets
- Data Assets
- Glossaries
- Terminologies
- Governance Model

Compose UI Behavior, Queries, Rules and Constraints

Over 300 of pre-defined Asset Types
How Applications can be Model Driven – an example from TopBraid EDG

Models drive the User Experience (UX) i.e., what fields will appear on an (edit, search or browse) form, possible values, order of the fields, their division into sections and other relevant information.

An active model (ontology) describing a File Asset as it is captured in a Knowledge Graph – open to access at run time, sharable, making data self describing – for people and machines.

A form for capturing data about files – if a model changes, a form automatically adjusts.

The same form can display information about other assets – whatever the model says, it will display and will let you edit according to the definitions in the model.

We know the meaning of each data value because it is described in the ontology that is as accessible as the data.
Some Limitations of Property Graphs You are Likely to Face

- No Capture of Schema in a Graph
- Minimal Support for Validation and Data Integrity
- No Capture of Rich Rules in a Graph
- No Support for Inheritance and Inference in a Graph
- No Globally Unique Identifiers
- No Resolvable Identifiers
- No Inherent Connectivity Across Graphs
- Difficulties with Graph Evolvability
Steps for Transitioning from Property Graphs to a Knowledge Graph

1. Get your data out – “as-is”
   - It is fairly easy to generate one of the RDF standard serializations from a property graph. In fact, Neo4J offers a library for doing this.

2. Create a model for your data
   - You will not be able to get the semantics of the data; this is due to the fact that the data model only exists in your initial design sketches and, partially, within Cypher queries and programs.
   - TopBraid EDG can introspect the data to help you generate the initial model.
Steps for Transitioning from Property Graphs to a Knowledge Graph

3. Iterate
   – The initial model and data is likely to benefit from some adjustment as the structure of the graph data may be influenced by the specific limitations of the property graph data model and optimizations that were required due to its architecture

4. Migrate application code and queries
   – Take advantage of the data validation, reasoning and model-based introspection capabilities of the knowledge graphs – you will end up with smaller and more maintainable code base
   – If you already use GraphQL, this will likely simplify the migration
Summary of Key Take Away Points

- Increasingly graphs are seen as an excellent choice for data / metadata representation and management - for the data centric enterprise
  - Flexibility is one strong driver: heterogeneous data, integrating new data sources, analytics, and metadata management and governance all require flexibility and evolvability

- The two main graph data models are: **Property Graphs** and **Knowledge (RDF) Graphs**

- Property Graphs are offered in mature implementations (e.g. Neo4J), that are easy to get started with
  - but there are fundamental limitations that are not addressed in the design of property graphs
  - E.g., Property Graphs are not self-describing; the meaning of the data they store is not a part of a graph

- Knowledge Graphs support the move to Data Centric systems and architectures
  - by offering robust capabilities that support flexibility and evolvability
  - E.g., Knowledge Graphs capture not only data, but the meaning or semantics of data, including rich constraints and highly expressive rules
  - In the past, implementations of Knowledge Graphs were considered to be more academic, but this has changed with products like TopBraid EDG

- It is possible to move from a Property Graph to a Knowledge Graph in logical steps
  - Get your data out – “as-is”; *Create a model for your data*; iterate and optimize ...
Thank You!

... Questions?
Benefits of a Knowledge Graph based Platform for Data Governance 2.0

TopBraid Enterprise Data Governance (EDG):

- Is flexible and extensible, based on standards
- Integrates reasoning and machine learning
- Enables people (UI) and software (APIs/web services) to view, follow and query
- Bridges data and metadata “silos” for a seamless data governance
- Delivers Knowledge-driven data governance

As an enterprise knowledge graph infrastructure, TopBraid EDG supports Data Governance 2.0 and applications of AI / ML
To Learn More about TopBraid EDG and Knowledge Graphs:

**EDG Product Info:**
- [TopBraid Enterprise Data Governance](https://www.topquadrant.com/products/topbraid-enterprise-data-governance/) (TopBraid EDG)

**Contact us:** at [info@topquadrant.com](mailto:info@topquadrant.com) to:
- Discuss data governance and knowledge graphs
- Request a more targeted demo of TopBraid EDG
- Ask for a free EDG evaluation account
Latest Whitepaper

Knowledge Graphs versus Property Graphs: Similarities, Differences and Some Guidance on Capabilities

Download a copy on our website at: topquadrant.com/knowledge-assets/whitepapers/
More Webinar Recordings, Slides, Q&A:

Short Videos:
- TopBraid EDG “Quick Grok” Videos
  https://www.topquadrant.com/knowledge-assets/videos/
- TopBraid EDG Animated Video
  https://www.topquadrant.com/project/edg_agile_modular/

Blog:
- https://www.topquadrant.com/the-semantic-ecosystems-journal/

Data Governance White Papers
- https://www.topquadrant.com/knowledge-assets/whitepapers/