Module 1.1

RDF vs LPG, side by side

Reading companion

An honest survey of the canonical sources that ground sub-module 1.1.
For personal study, talk preparation, or a quick orientation before drafting.

The most-asked question, taken seriously

The question

Anyone arriving at RDF from Neo4j (or arriving at Neo4j from RDF) asks the same thing: which one should I be using, and what's the actual difference?

Most existing answers are partisan. The vendor pieces lean one way; the semantic-web defenders lean the other. The honest read is that both stacks are correct for their context — and that the choice has cascading consequences.

What this deck leaves you with

- A reading order that builds from foundation to nuance
- A summary and reading-time estimate for each source
- An honest note on what each source is good for and where it falls short
- A synthesis slide that names the one conceptual move that ties everything together
- Direct links for working through each piece

Eight sources, two tiers

Four primary readings for the conceptual core. Four supporting pieces that surface the practitioner debate.

Primary

Allemang, Hendler & Gandon — Ch 1–3
RDF, RDFS, the conceptual core. ~4 hours.

Hogan et al. — Section 3
Data models in the broader landscape. Free, online. ~2 hours.

DuCharme — Ch 1–2
SPARQL fundamentals. Reading anchor for ergonomics. ~3 hours.

W3C RDF 1.1 Primer
The spec, written for newcomers. ~1 hour.

Supporting

Hartig et al. — Edge-labelled vs Property Graphs
arXiv 2022. User study, not opinion. ~1 hour to skim.

Cagle — RDF 1.2 vs Neo4j/OpenCypher
Substack 2026. Opinionated, recent. ~30 min.

DeBellis — Semantic Web vs Property Graphs
Personal blog. Balanced practitioner take. ~30 min.

Neo4j — RDF vs Property Graphs
Vendor blog. Read for the case against RDF. ~20 min.

Total reading time: ~10 hours primary, ~2.5 hours supporting

primary

Semantic Web for the Working Ontologist, Ch 1–3

Allemang, Hendler & Gandon · 3rd ed., 2020 · Book chapters · ~4 hours

What it is

The conceptual foundation. Allemang has spent two decades helping enterprises actually deploy this stack; the book is shaped by what he's seen go wrong. Chapters 1–3 build RDF from first principles: triples, URIs, the four serializations, and RDFS as the first layer of meaning on top of raw triples. The 3rd edition was substantially updated for the modern knowledge-graph era.

Key takeaways

- Triples are the substrate; everything else is convention layered on top
- URIs do real work: global identity is the feature, not the verbosity
- RDFS gives you typed nodes and edges without committing to OWL's complexity
- The book voice is honest about pain points — uncommon in textbooks

Read with this in mind

Allemang earns his framing through examples, not declarations. Notice where he switches between abstract spec and concrete deployment story — that's where the practical judgment lives.

https://workingontologist.org/

primary

Knowledge Graphs, Section 3 (data models)

Hogan et al. · 2021 · Open-access book / Synthesis Lecture · ~2 hours

What it is

Eighteen of the world's top knowledge-graph researchers wrote a single coherent treatment of the field. Section 3 covers data models in the broader landscape — including both RDF and property graphs as siblings, not rivals. This is the most useful single source for the RDF-vs-LPG framing because Hogan and co-authors treat both with respect and explain their differences in formal terms.

Key takeaways

- RDF and LPG are both formalizations of directed graphs, with different primitives
- Embedding-based methods (knowledge graph embeddings) sit beside both as a third tradition
- The vocabulary used across the section gives you a shared frame for talking to both communities
- Free, online, citable in your own writing without permission hassle

Read with this in mind

Hogan et al. write like academics; the prose is denser than Allemang. Read Section 3 after you've internalized Allemang's chapters — the academic frame makes more sense once the concepts are familiar.

primary

Learning SPARQL, Ch 1–2

DuCharme · 2nd ed., 2013 · Book chapters · ~3 hours

What it is

DuCharme writes with unusual clarity for a technical book. Chapters 1–2 are the SPARQL-side counterpart to Allemang — they get you reading and writing basic queries against real data. The book is dated (2013) but the SPARQL spec has not changed in ways that age the content. The companion site provides every example as downloadable files for your own Fuseki.

Key takeaways

- SPARQL is graph-pattern matching, not table operations — the mental shift from SQL is real
- Triple patterns with variables are the primitive; everything else composes from them
- Reading SPARQL is easier than writing it; DuCharme builds up to writing with patience
- His blog at bobdc.com is the active follow-on; subscribe for current practice

Read with this in mind

DuCharme assumes you can already read Turtle. If Turtle still feels foreign, finish Allemang Ch 2 before opening this book. Otherwise the example queries will feel like noise.

https://www.learningsparql.com/

primary

RDF 1.1 Primer

W3C · 2014 · Web specification (primer-style) · ~1 hour

What it is

The official W3C introduction to RDF. Unlike most specs, the Primer is written deliberately for newcomers — it's the document the working group wrote for people who needed to onboard onto RDF for the first time. Read it after Allemang Ch 1 for the formal anchor on the concepts you've just been introduced to informally.

Key takeaways

- The four serializations explained with the same example in each format
- Datatypes, language tags, and blank nodes covered with care
- Concise — read end to end in one sitting
- The reference text to point clients at when they ask 'where is this written down officially'

Read with this in mind

The Primer is the gentle on-ramp. The full RDF 1.1 Concepts document is the formal spec — don't read that for the first pass; bookmark it as a reference for later.

https://www.w3.org/TR/rdf11-primer/

supporting

Edge-labelled vs Property Graphs: a user-perspective comparison

Hartig et al. · arXiv, 2022 · Academic paper · ~1 hour (skim)

What it is

A user study comparing how people actually model the same domain in RDF-star and Cypher. Most existing RDF-vs-LPG comparisons are opinion pieces written by partisans of one side. This one collects evidence from real users on five modelling problems and analyzes the patterns. It's the closest thing the field has to a controlled comparison — and the findings don't always favor the side the authors might prefer.

Key takeaways

- Both formalisms can express the same domain, with different ergonomic costs
- Users introduce structural inconsistencies in both — but different kinds
- Reification (talking about a triple) is where ergonomics diverge sharpest
- Empirical anchor when you need to push back on partisan claims

Read with this in mind

Skim, don't deep-read. The methodology section is detailed; you don't need it. The findings section and tables are where the value is.

https://arxiv.org/pdf/2204.06277

supporting

RDF 1.2 vs Neo4j / OpenCypher

Kurt Cagle · Substack, 2026 · Practitioner blog post · ~30 min

What it is

Cagle is opinionated, prolific, and reads the field constantly. This post is one of the most up-to-date comparisons of where the two stacks actually sit in 2026 — including the recent RDF 1.2 changes that close some of the historical ergonomics gaps and the practical state of named graphs versus property-bag edges. Read for the recency more than the balance.

Key takeaways

- RDF 1.2 changes the comparison materially — older posts miss this
- Named graphs and quoted triples now do work that previously required reification gymnastics
- Neo4j's element IDs improved but still aren't stable application identifiers
- Cagle's framing is partisan-but-informed; useful as a current-state snapshot

Read with this in mind

Cagle is an RDF partisan. Read with that in mind. The factual claims about the spec are reliable; the conclusions about Neo4j's weaknesses sometimes overstate.

https://ontologist.substack.com/p/rdf-12-vs-neo4jopencypher

supporting

Semantic Web vs Property Graphs

Michael DeBellis · Personal blog (updated 2025) · Practitioner blog post · ~30 min

What it is

DeBellis is the author of the canonical new Protégé pizza tutorial — a working ontology engineer with deep practitioner experience. This piece is the most balanced practitioner-side comparison you'll find. He covers the genuine wins of each stack without overselling, and the writing is concrete enough to be useful for thinking about your own modeling decisions.

Key takeaways

- Property graphs handle n-ary relations naturally; OWL needs a design pattern
- The n-ary relation pattern is so well-known that the convenience gap is smaller than it appears
- Reasoning support is where the semantic-web stack still has a clear unique offering
- DeBellis's writing on SHACL elsewhere is the natural follow-on for Module 3

Read with this in mind

DeBellis comes from the semantic-web side but writes fairly. The most useful framing: where does each stack let you do something the other can't do at all?

https://www.michaeldebellis.com/post/owlvspropgraphs

supporting

RDF vs Property Graphs: Choosing the Right Approach

Neo4j (vendor) · 2024 · Vendor blog post · ~20 min

What it is

The vendor's case for property graphs over RDF. Read deliberately as advocacy — Neo4j has a clear commercial interest. The post overstates RDF's weaknesses (the n-ary point in particular treats a design-pattern workaround as a fundamental limitation) but accurately captures the developer-experience friction that drives real adoption decisions.

Key takeaways

- Vendor framing reveals what they think their customers care about — useful market signal
- The 'first-class relationships' argument is the strongest one; take it seriously
- GQL standardization (2024) closes one historical gap RDF held over LPG
- Read what's NOT said — formal semantics, reasoning, vocabulary reuse barely appear

Read with this in mind

This is the post your client may have read before talking to you. Knowing the argument structure and where it stretches is consulting positioning.

https://neo4j.com/blog/knowledge-graph/rdf-vs-property-graphs-knowledge-graphs/

THE CONCEPTUAL MOVE

RDF and LPG model the same kinds of facts using different primitives — and the consequences cascade.

RDF

Predicates are global URIs

Every term is universally identifiable. Vocabulary reuse is the unlock; verbosity is the cost. The interop story has actual teeth when external graphs (ESCO, FIBO, Wikidata) already speak your terms.

LPG

Relationships are local, with their own properties

Edges carry attributes natively. No reification gymnastics for time, weight, or provenance. Ergonomics for path traversal are the unlock; cross-system integration is the cost.

Neither is a superset. The choice depends on what your application actually does — and on what you can defend, in detail, to the engineer who'll inherit it.