Reliable Context

RELIABLE CONTEXT / RESEARCH & PRACTICAL GUIDANCE

RDF vs Property Graph vs Relational for Knowledge Graphs

> Reliable Context is a kernel of truth: a system of record for organizations that fuse private and public information and must defend every value they act on. One canonical record per real thing, a cited source and an observation date behind every value, unknowns declared rather than left blank, and any past state of the record reproducible.

Published by Reliable Context · 2026-10-29 · rdf vs property graph · knowledge graph rdf or lpg · enterprise knowledge graph model · gql iso 39075 · rdf vs neo4j

Original article: https://reliablecontext.com/articles/rdf-vs-property-graph-vs-relational-enterprise-knowledge-graph



In this article

1. The short answer
2. Three ways to store connected facts
3. RDF: triples, IRIs, graphs
4. Property graphs: nodes, edges, properties
5. Relational: tables and joins
6. One fact, three models, with provenance
7. Query languages
8. Schema and validation
9. Interchange and identity
10. Decision table by requirement
11. Sources

The short answer

All three models can store the same connected facts. They differ in what they make easy to say *about* a fact, how they combine data from different sources, and which standards govern exchange and query. If your requirement is to merge data from many publishers and exchange it under open standards, RDF fits best. If your requirement is traversal-heavy application queries over a graph you own, a property graph is usually more direct. If your facts are already relational, highly regular and governed by SQL tooling, relational tables with a graph view (SQL/PGQ) may be enough. For a system of record whose central need is provenance on every statement, the deciding question is: **how does the model attach evidence to one edge, and does that attachment survive export?**

This article explains the three models, shows one fact modelled three ways with its provenance, compares query and validation standards as of 28 October 2026, and ends with a decision table.

Three ways to store connected facts

A **knowledge graph** here means a set of entities and typed relationships between them, used as a shared reference across applications. Hogan et al.'s survey describes the main graph data models as directed edge-labelled graphs (of which RDF is the standard instance), property graphs, and graph datasets that manage several named graphs together (Hogan et al., §2.1). The relational model is the third option in practice, because most enterprise facts already live in tables.

	RDF	Property graph	Relational
Unit of data	Triple: subject, predicate, object	Node or edge, each with a label and key-value properties	Row in a table with a fixed set of columns
Identity	IRIs, which have global scope	Database-internal identifiers	Primary keys, local to a database

	RDF	Property graph	Relational
Edge attributes	Not native in RDF 1.1; reification, named graphs, or RDF 1.2 triple terms	Native: properties on edges	Native: columns on a join table
Schema	Optional; RDFS/OWL for meaning, SHACL for validation	Optional or vendor-specific; GQL defines graph types	Required; DDL with constraints
Default assumption about missing facts	Typically open world	Typically closed world in practice	Closed world
Standard query language	SPARQL (W3C)	GQL (ISO/IEC 39075:2024); Cypher and others	SQL; SQL/PGQ for graph queries

RDF: triples, IRIs, graphs

RDF 1.1 defines an RDF graph as "a set of RDF triples", each made of a subject, predicate and object. Nodes are IRIs, literals or blank nodes. An RDF dataset is "a collection of RDF graphs" with exactly one unnamed default graph and zero or more named graphs (RDF 1.1 Concepts §3-4).

The property that matters most for enterprise integration is identity. "By design, IRIs have global scope. Thus, two different appearances of an IRI denote the same resource" (RDF 1.1 Concepts §1.3). Two RDF graphs that use the same IRI for a company can be merged by taking their union, and the statements line up without a mapping step. The corollary is that identity mistakes also merge silently: if two publishers use one IRI for two different companies, the union asserts that one thing has both sets of properties.

RDF is usually interpreted under an **open world assumption**: the absence of a statement does not mean it is false, only that it is not known (Hogan et al., §3.1.1). That is appropriate when combining partial sources. It is a poor fit for questions such as "list suppliers with no sanctions hit", where an absent statement has to be distinguished from a search that was performed and found nothing.

Property graphs: nodes, edges, properties

A property graph "allows a set of property–value pairs and a label to be associated with both nodes and edges" (Hogan et al., §2.1.3). The model was proposed for modelling flexibility: a relationship can carry its own data (a start date, a weight, a source) without being turned into an intermediate node.

Hogan et al. make a point that is easy to lose in vendor comparisons: "property graphs can be translated to/from directed edge-labelled graphs without loss of information", and "often the choice of model will be secondary to other practical factors, such as the implementations available" (Hogan et al., §2.1.3). The models are equivalent in what they can represent. They differ in what is convenient, standardized and portable.

Property graphs became an ISO standard target only recently. ISO/IEC 39075:2024 (GQL) "defines data structures and basic operations on property graphs" and a language for "creating, accessing, querying, maintaining, and controlling property graphs" (ISO/IEC 39075:2024). ISO/IEC 9075-16:2023 (SQL/PGQ) "defines ways for the SQL language to represent property graphs and to interact with them" (ISO/IEC 9075-16:2023), which lets a relational database expose tables as a graph for querying.

Relational: tables and joins

A relational schema can represent any graph: one table per entity type, one table per relationship type, with foreign keys. Edge attributes are columns on the relationship table. Strengths are mature constraints (keys, foreign keys, checks), transactions, and a vast tool ecosystem. Weaknesses for knowledge graphs are schema rigidity when new relationship types appear weekly, and the effort of writing variable-length path queries, which SQL/PGQ now addresses inside SQL.

One fact, three models, with provenance

The example is fictional. **Halden Therapeutics AG and Halden Biologics GmbH are fictional companies.**

The fact: *Halden Therapeutics AG owns 60% of Halden Biologics GmbH*, read from the 2025 annual report, page 112, on 6 March 2026, by an analyst. The requirement is that the ownership edge carries its source, page, quote and read date, because the percentage will be used in a risk calculation.

RDF 1.1: a named graph per assertion

```
# TriG: the statement lives in its own named graph; the graph is described in the default graph
@prefix ex:   <https://example.org/kg/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

ex:assertion-771 {
    ex:halden-therapeutics ex:ownsShareOf ex:halden-biologics .
}
ex:assertion-771 prov:wasDerivedFrom ex:annual-report-2025 ;
    ex:sharePercent "60"^^xsd:decimal ;
    ex:page 112 ;
    ex:quote "Halden Therapeutics AG holds 60% of the shares of Halden Biologics GmbH." ;
    prov:generatedAtTime "2026-03-06T10:12:00Z"^^xsd:dateTime .
```

The named graph is an entity you can describe, which is exactly how PROV bundles work. The cost: the percentage sits on the graph rather than on the edge, and one graph per statement multiplies graph names.

RDF 1.2: a reifier for the triple

RDF 1.2 introduces **triple terms**: "an RDF triple used as an RDF term within another triple". A reifying triple uses `rdf:reifies` to point from a **reifier** to a triple term, and "it is expected that the reifiers (rather than the triple terms) will be used in further statements" (RDF 1.2 Concepts §1.5). The specification notes that a triple term on its own is not asserted: in its own example, the proposition "is not claimed to be true".

```
# RDF 1.2 Turtle (draft syntax): assert the triple and annotate it through a reifier
ex:halden-therapeutics ex:ownsShareOf ex:halden-biologics ~ ex:claim-771 {|
    ex:sharePercent 60 ;
    prov:wasDerivedFrom ex:annual-report-2025 ;
    ex:page 112 ;
```

```
    prov:generatedAtTime "2026-03-06T10:12:00Z"^^xsd:dateTime
|} .
```

That separation between an asserted triple and a quoted proposition is useful for evidence: a record can hold a claim from a source without asserting it, which is what a conflicting or unverified value should look like. RDF 1.2 Concepts was a W3C Candidate Recommendation Snapshot dated 7 April 2026, and RDF 1.2 Turtle a Working Draft dated 23 October 2026, so the syntax above may still change (RDF 1.2 Concepts).

Property graph: properties on the edge

```
(:Company {name: "Halden Therapeutics AG"})
  -[:OWNS_SHARE_OF {percent: 60, source: "annual-report-2025", page: 112,
                    quote: "Halden Therapeutics AG holds 60% ...",
                    read_at: datetime("2026-03-06T10:12:00Z")}]->
(:Company {name: "Halden Biologics GmbH"})
```

This is the most direct form. Two limits appear as requirements grow. First, an edge property is a literal; if the source is itself an entity with its own provenance (who fetched it, which version), it needs to become a node, and the edge needs to point to it, which a plain edge cannot do. Many designs then promote the relationship to a node (`(:Ownership)`), which is reification by another name. Second, the evidence is portable only as far as the export format carries edge properties.

Relational: a relationship table with evidence columns

ownership_id	owner_id	owned_id	percent	source_id	page	quote	read_at
771	C-001	C-014	60.0	DOC-2025-AR	112	Halden Therapeutics AG holds 60% ...	2026-03-06 10:12Z

The relational form is the most rigid and the most enforceable: `source_id` can be a non-null foreign key to a documents table, so a row without a source cannot exist. That is a constraint the other two models express only through a separate validation layer.

Query languages

Language	Model	Standard and status (28 October 2026)
SPARQL 1.1	RDF	W3C Recommendation, 2013
SPARQL 1.2	RDF 1.2	W3C Working Draft, 25 October 2026
GQL	Property graph	ISO/IEC 39075:2024, published
SQL/PGQ	Relational tables viewed as a property graph	ISO/IEC 9075-16:2023, published
Cypher and other vendor languages	Property graph	Vendor or community specifications

SPARQL queries across named graphs, so the RDF 1.1 pattern above is queryable: select the triple and the metadata of the graph that contains it. Property graph languages query edge properties directly. SQL/PGQ lets a relational system answer path queries without moving the data.

Schema and validation

RDF separates meaning from validation. OWL and RDFS describe what terms mean and support inference under the open world assumption. SHACL, a W3C Recommendation since 20 July 2017, is "a language for validating RDF graphs against a set of conditions"; the conditions are written as shapes in a "shapes graph", and the graph being checked is the "data graph" (SHACL). A shape can require that every ownership claim has exactly one source and one read date, which closes the world locally for the purpose of validation. A SHACL 1.2 Core Working Draft was published on 25 October 2026 (SHACL 1.2 Core). The site's comparison of OWL entailment and SHACL constraints covers the distinction in detail.

Property graph databases offer constraints that vary by product; GQL defines graph types, but implementations differ in which constraints they enforce. Relational databases have the strongest built-in enforcement.

Interchange and identity

Concern	RDF	Property graph	Relational
Standard serialization	Turtle, TriG, N-Triples, JSON-LD, RDF/XML	No single standard exchange format widely shared across products	CSV and SQL dumps; schema travels separately
Merging two sources	Union of graphs; shared IRIs align automatically	Requires an ID mapping step	Requires key mapping and schema alignment
Risk on merge	Wrong shared IRI silently fuses two things	Duplicate nodes until resolved	Duplicate rows until resolved
Statement-level provenance in export	Named graphs (1.1) or reifiers (1.2) travel in standard formats	Travels if the format carries edge properties	Travels as columns

Global identifiers make RDF strong for integration, and they move the hard problem to the point where identifiers are assigned. Entity resolution has to happen before IRIs are shared, not after. Linking two records with `owl:sameAs` lets a reasoner combine the data for both as one resource (Hogan et al., §3.2.1), which is the wrong tool when two entities are merely related.

Decision table by requirement

If your dominant requirement is ...	Prefer	Because
Exchanging facts with external parties under open standards	RDF	Standard serializations and global identifiers

If your dominant requirement is ...	Prefer	Because
Merging many partially overlapping sources	RDF	Union semantics; open world
Provenance on every statement, exported intact	RDF (named graphs now, reifiers when RDF 1.2 is final) or relational with evidence tables	Statement-level metadata in a standard or enforced form
Deep traversal queries in an application you control	Property graph	Native edge properties; path queries
Hard guarantees that every fact has a source	Relational, or any model plus SHACL or equivalent validation	Non-null foreign keys or shape constraints
Graph queries over data already in SQL	Relational with SQL/PGQ	No data movement
Reasoning over class hierarchies	RDF with OWL	Standard entailment

Two observations hold regardless of choice. First, the model does not supply provenance; it only offers places to put it. A property graph with a `source` property on every edge is better evidenced than an RDF graph without named graphs. Second, model choice does not settle identity. All three models need an entity resolution step before facts from different sources can be combined safely, as the site's note on the RDF 1.2 provenance migration test and its knowledge graph glossary entry discuss.

Whichever model is chosen, the property to test is the one Reliable Context publishes as a guarantee: one canonical record per real thing, with a cited, verbatim, dated source behind every value.

Sources

1. Hogan et al., Knowledge Graphs, ACM Computing Surveys 54(4), 2021 (arXiv:2003.02320) - graph data models; translation between property graphs and edge-labelled graphs; open world assumption; owl:sameAs.
2. RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, 25 February 2014 - triples, graphs, datasets, global scope of IRIs.
3. RDF 1.2 Concepts and Abstract Data Model, W3C Candidate Recommendation Snapshot, 7 April 2026 - triple terms, reifiers and reifying triples.
4. Shapes Constraint Language (SHACL), W3C Recommendation, 20 July 2017 - validation of RDF data graphs against shapes graphs.
5. SHACL 1.2 Core, W3C Working Draft, 25 October 2026 - status of the next SHACL version.
6. SPARQL 1.1 Query Language, W3C Recommendation, 21 March 2013 - standard RDF query language.
7. SPARQL 1.2 Query Language, W3C Working Draft, 25 October 2026 - status of SPARQL 1.2.
8. ISO/IEC 39075:2024, Information technology — Database languages — GQL - published property graph query language; abstract.
9. ISO/IEC 9075-16:2023, SQL Part 16: Property Graph Queries (SQL/PGQ) - property graphs within SQL; abstract.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://arxiv.org/abs/2003.02320
2. https://www.w3.org/TR/rdf11-concepts
3. https://www.iso.org/standard/76120.html
4. https://www.iso.org/standard/79473.html
5. https://www.w3.org/TR/rdf12-concepts
6. https://www.w3.org/TR/sparql11-query
7. https://www.w3.org/TR/sparql12-query
8. https://www.w3.org/TR/shacl
9. https://www.w3.org/TR/shacl12-core

THE PUBLISHER

About Reliable Context

Reliable Context is a system of record for organizations that fuse private and public information and must defend every value they act on.

The guarantees

One canonical record per real thing: duplicates resolve by rule, ambiguity is shown rather than hidden, and related organizations relate without ever being merged. A verbatim, cited source and an observation date sit behind every value. Unknowns are declared as not found, blocked or not applicable with a reason, never left as an empty cell. Any past state of the record is reproducible, together with what changed since.

Where it runs

It runs inside the customer's perimeter, one database per organization. Permitted sources, model routing, reader clearance and retention are set as the customer's own policies, and every value is exportable with its chain of custody for an auditor or regulator.

What it is not

It is not a vector database or a RAG framework, not a data catalog, MDM or data-quality suite, not a CRM or data-enrichment vendor, and not a chatbot or AI assistant. Where a reader needs one of those, the honest answer is to say so and point them there.

Visit

See Reliable Context.

Reliable Context is in production use in two sectors; more than six thousand of its identity decisions were made by rule and none has been reversed. Articles published here explain general practice, cite their primary and academic sources, and name the jurisdiction of any regulatory rule they describe. They are not legal, regulatory or compliance advice for a specific organization, and requirements change.

Website: https://reliablecontext.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.