Linked Data in the Time of AI

Three days, from the first IRI to a knowledge graph an agent can be trusted to use. The standards are twenty years old. The reason to learn them is new: a language model can read anything, and it knows nothing about your data that your data does not say.

Why this, and why now

For most of their history, self-describing data and well-designed APIs were the right answer to a question few organizations felt they had to ask. A developer could always read the documentation, ask a colleague what the column meant, and hard-code the join. The cost of data that did not explain itself was paid slowly, in integration projects, and it was easy to mistake for the cost of doing business.

That consumer has changed. An agent does not ask a colleague. It reads what it is given, infers the rest, and acts — fluently, and at machine speed. Handed a field called `status` with the value `3`, it will guess, and the guess will read exactly as confident as a fact. **Data that carries its own identity, its own meaning and its own links removes the guess.** That has made APIs and self-describing data more useful than they have ever been: they are no longer only an integration convenience, they are the difference between an agent that is grounded and one that is improvising.

Agentic systems today look like the internet before the Web settled on its architecture — bespoke, brittle, point-to-point, with no coherent identity and constant collisions of terminology. Their hard problems are not AI problems. They are *context* problems, *identity* problems, *integration* problems and *authorization* problems, and the Web solved naming, identity and context resolution at scale a generation ago. Agentic systems will have to solve them again, or reuse what already works. This course is about reusing what already works.

Three days, six half-day sessions. The first day builds the model, the second makes it work for a living, and the third connects it to language models and agents. Days one and two stand on their own as a two-day Linked Data course for a team that is not there yet; day three assumes them.

One dataset, three sources that disagree

Every exercise runs against a dataset we supply: a small collection of people, organizations, publications and events, drawn from three sources that use different identifiers, different field names and, in a few planted places, different facts. It loads with one command and needs no external services.

This is deliberate. Linked Data earns its keep at the seams between systems, and a clean single-source example hides precisely the problem the technology exists to solve. The planted disagreements produce the same realizations in every room — the first time two graphs merge without a mapping table, and the first time a merged graph turns out to be confidently wrong. Participants apply the same techniques to their own data between sessions, where the stakes are real and the discussion is theirs.

Day 1 · The Model

DAY 1 | TWO HALF-DAYS | NO PREREQUISITES

For engineers and architects who have heard the words and never had a reason to care. The day replaces “the Semantic Web didn’t happen” with an accurate picture of what did: the parts of the stack that quietly won, where they are already running in production, and why they fit the present moment better than the one they were designed in.

THE SESSIONS

1 · Names, Not Addresses. The first step toward the Web’s model is to give things names — and to separate a thing’s identity from where it lives and from how it is represented. URLs, URNs and IRIs as one uniform way to name across schemes and protocols. The Richardson maturity levels read as an argument about data rather than about endpoints: identity, then interaction, then representation. Content negotiation, and why a person’s identifier must not resolve to a web page. Persistent identifiers that outlive the domain that minted them. Application-centric against data-centric architecture, and the open-world assumption that makes the second one possible.

Exercises: Name the Thing · Follow Your Nose — dereference a single identifier and see how far the links carry you.

2 · The Graph. RDF as the simplest data model that can absorb someone else’s data without a meeting: statements, graphs, and merge as a set union rather than a project. Turtle for people and JSON-LD for the developers who do not want to know it is RDF. Named graphs. Blank nodes, why they feel convenient, and why a graph you intend to merge, cache or cite should carry stable IRIs instead. Reusing vocabularies — schema.org, Dublin Core, FOAF, SKOS — before inventing one, and how to invent one when you must.

Exercises: Three Sources, One Graph · The Merge That Lied — the planted disagreement, found by the people who just merged it.

Day 2 · The Working Graph

DAY 2 | TWO HALF-DAYS | DAY 1 OR EQUIVALENT

For teams that need the graph to answer questions and hold its shape. A graph that accepts everything is a graph nobody can rely on. This day is about the two things that make one dependable: a query language that reaches across what used to be silos, and a way to say what *valid* means that a machine can check.

THE SESSIONS

3 · Asking Questions. SPARQL from the first pattern match to the queries that justify the whole exercise. Optional and negated patterns, filters and aggregates; `CONSTRUCT` as the way one graph becomes another; federation across endpoints you do not own. Property paths, which turn roll-up and drill-down over a hierarchy into a single expression instead of a recursive program. The payoff is the cross-silo question. When a commit, a pull request, a review finding, a work item and a human annotation share one identity scheme, *what is open against this file, by whom, across every repository* is one query — and it is a query no single-purpose tool can answer, because each of them owns only one of the five.

Exercises: Query Ladder · The Question No Tool Can Answer.

4 · Meaning, Shape and Publication. RDFS and just enough OWL to be useful without becoming a research program. SKOS for the vocabularies an organization already has and does not call vocabularies. Then the closed-world half that production systems need: SHACL shapes as the contract a graph must meet, validated mechanically, with reports a pipeline can act on. Vocabulary governance as a test rather than a policy — every term in use must exist in the published vocabulary, and the build says so. Publishing: JSON-LD in web pages and API responses, structured data that search engines and agents already consume, and hypermedia representations that tell a client what it can do next.

Exercises: Write the Shape, Break the Shape · Publish and Be Found — the participant's own profile as Linked Data, validated and dereferenceable.

Day 3 · In the Time of AI

DAY 3 | TWO HALF-DAYS | DAYS 1–2 OR EQUIVALENT

For teams putting language models and agents in front of their data. The day does not relitigate whether models hallucinate. It starts from the position that they are useful and unreliable in known ways, and asks the engineering question: which parts of the system should be the model's job, and what has to be true of the data for the rest to be deterministic.

THE SESSIONS

5 · Graphs and Language Models. What each is good at, stated without enthusiasm: a model is fluent, tolerant and unaccountable; a graph is exact, brittle and able to cite its sources. Retrieval-augmented generation and where similarity search runs out — questions that need a join, a count or a negation rather than a nearby paragraph. Graph-based retrieval and hybrid approaches. Using a model for what it is good at — extraction, entity linking, drafting a query from a question — with a shape validating everything it produces before anything trusts it. Provenance with PROV and named graphs, so that an answer arrives with where it came from and *why do you believe that* has a mechanical reply.

Exercises: Where Similarity Fails · Extract, Then Validate — a model populates the graph, and a SHACL shape decides what gets in.

6 · Agents on a Web of Data. Self-describing resources as the surface an agent works against, and why that beats a hand-curated tool list. The selection funnel that puts the model last: deterministic narrowing of what is possible, the model choosing only among what remains, a validation gate before anything executes. **Affordance should equal authorization** — a capability-scoped agent should not be refused an action, it should be unable to see it, so the attempt is never made and the refusal never has to be explained. State that lives outside any context window: a work ledger as a graph, so that what one session learned is there for the next, which is a different context entirely. And context itself as something resolved rather than passed around — the same name resolving differently by time, jurisdiction, role or circumstance, which is how a system answers *why was that decision made, and what was true when it was*.

Exercises: Narrow, Then Choose — build the funnel and measure how little is left for the model to decide · Design Review — a proposed agent integration, and the three places it is relying on a guess.

> **What this course is not.** It is not a tour of a graph-database product, and it is not an ontology-engineering course. The standards are taught because they are the portable part. Every exercise runs on open-source tooling, and nothing in the material depends on a vendor.

Who should come

Software engineers, data engineers and architects. No prior RDF, SPARQL or ontology experience is assumed on day one, and none is useful on day one — the most common thing to unlearn is a half-remembered account of why this did not work in 2008. Comfort with JSON, HTTP and a query language of any kind is enough.

Teams that already run a graph in production can begin at day two. Teams whose immediate problem is an agent in front of enterprise data sometimes ask to begin at day three; the recommendation is against it, because day three's answers are day one's identifiers and day two's shapes applied under pressure.

Practical requirements

What participants need. A machine they can install and run software on, and a text editor. The supplied environment brings its own triple store, query tooling and validator, runs locally, and needs no credentials. Day three's exercises use whichever language model the participant's organization already permits; they are written to produce comparable results across them, and the lessons do not depend on the model being a good one.

What is supplied. The dataset and environment; a participant handout for each session carrying the exercises, worked material and space to write; a reference card for Turtle, SPARQL and SHACL; and the solution graphs and queries, released at the end of each day.

Format. Designed for online delivery and equally deliverable in a room. Exercises are solo with discussion in chat or open floor. Each half-day runs about four hours including two breaks.

Adapting it. The syllabus is a starting point rather than a catalog item. It can be reshaped around a client's own domain and vocabularies, run as days one and two alone, or paired with *Securing and Designing APIs* — the other half of the same argument, about the interfaces this data travels through and how to connect an agent to them safely.

Next steps. This syllabus is a starting point. Courses are assembled per client — the emphasis, the examples and the length all move to fit the team in the room. To request more detail, or to schedule a discussion about running this course, get in touch at bosatsu.net/contact.html.

Brian Sletten · Bosatsu Consulting, Inc.
Semantic Web · Knowledge Graphs · RDF · SPARQL · SHACL · JSON-LD · AI & LLMs · agents