Semantic University

Introduction

This set of lessons is an introduction to RDF, the core data model of the Semantic Web and the foundation of all other Semantic Web technologies. The lessons introduce RDF, present additional details, and then place RDF in the context of other technologies such as XML and JSON. Coverage is meant for beginners new to RDF, but a technical background will very much help you follow the material.

If you haven't already completed the lessons in Semantic Technologies Applied, now's the time to do so. They provide critical context for this set of lessons.

RDF 101

RDF (Resource Description Framework) is one of the three foundational Semantic Web technologies, the other two being SPARQL and OWL.

In particular, RDF is the data model of the Semantic Web. That means that all data in Semantic Web technologies is represented as RDF. If you store Semantic Web data, it's in RDF. If you query Semantic Web data (typically using SPARQL (_wp_link_placeholder)), it's RDF data. If you send Semantic Web data to your friend, it's RDF.

In this lesson we will introduce RDF.

Objectives

In this lesson you will learn:

- What RDF is and how it fundamentally differs from XML and relational databases
- What is meant by a "graph data model"
- How RDF is typically represented visually
- The importance of the URI, and the significance (or lack thereof) of identity "universality"

Today's Lesson

RDF is the foundation of the Semantic Web and what provides its innate flexibility. All data in the Semantic Web is represented in RDF, including schema describing RDF data.

RDF is not like the tabular data model of relational databases. Nor is it like the trees of the XML world. Instead, RDF is a graph.

RDF Graphs

In particular, it's a labeled, directed graph. I don't mean "graph" as in "charts and graphs" but rather as in "dots and lines."

Therefore you can think of RDF as a bunch of nodes (the dots) connected to each other by edges (the lines) where both the nodes and edges have labels.

The term labeled, directed graph will mean a lot to the mathematicians in the audience, but for the rest of you I've included a simple example here.

This is a complete, valid, visual representation of a small RDF graph. Show it to any Semantic Web practitioner and it will be immediately obvious to her what it represents.

The nodes of the graph are the ovals and rectangles (ovals and rectangles are a convention that we'll get to shortly). The edges are labeled arrows that connect nodes to each other. The labels are URIs (this is very important, and we'll cover it in more detail in a bit).

Note: the graph nature of RDF is why the logos of Semantic Web companies almost universally have some reference to a graph. See if you can spot the graph in the logos of Revelytix, Ontoprise, Ontotext, Apache Jena, and Cambridge Semantics, not to mention the RDF logo, pictured at the top if this lesson.

There are three kinds of nodes in an RDF directed graph:

- Resource nodes. A resource is anything that can have things said about it. It's easy to think of a resource as a thing vs. a value. In a visual representation, resources are represented by ovals.
- Literal nodes. The term literal is a fancy word for value. In the example above, the resource is http://www.cambridgesemantics.com/people/about/rob (once again, a URI) and the value of the foaf:name property is the string "Rob Gonzalez". In a visual representation, literals are represented by rectangles.
- Blank nodes. A blank node is a resource without a URI. Blank nodes are an advanced RDF topic that we'll cover in another lesson. I usually recommend avoiding them in general, especially if you're new to the space. They are listed here simply for completeness.

Edges can go from any resource to any other resource, or to any literal, with the only restriction being that edges can't go from a literal to anything at all.

Think about this for a second.

This means that anything in RDF can be connected to anything else simply by drawing a line.

This idea is key. When we talk about Semantic Web technologies being fundamentally more flexible than other technologies (XML, relational databases, BI cubes, etc.), this is the reason behind it. In the abstract, you're just drawing lines between things. Moreover, creating a new thing is as easy as drawing an oval.

If you compare this mentally to the model you might know from working with a relational database, it's starkly different. Even for basic relationships, such as many-to-many relationships, the abstract model of a relational database gets complicated. You end up adding extra tables and columns (think foreign keys, join tables, etc.) just to work around the inherent rigidity in the system.

The ability to connect anything together, any time you want, is revolutionary. It's like hyperlinking on the Web, but for any data you have! The following video was in Introduction to the Semantic Web, but I'll include it here as well since it underscores this point.

This linking between things is the fundamental capability of the Semantic Web, and is enabled by the URI.

The Central Importance of the URI

If you want to connect two things in a relational database you have to add foreign keys to tables (or, if you have a many-to-many relationship, create join tables), etc. If you want to link things between databases, you need an ETL job using something like Informatica. It's just not easily done.

If you consider the XML world, the same thing is true. Connecting things within an XML document is possible, if tedious. Connecting things between XML documents requires real work. Unless you're one of the very few who just loves XSLT, you're not doing that very often.

The fundamental value and differentiating capability of the Semantic Web is the ability to connect things.

The URI is what makes this possible.

URI stands for Universal Resource Identifier. The universal part of that is key. Instead of making ad hoc IDs for things within a single database (think primary keys), in the Semantic Web we create universal identities for things that are consistent across databases. This enables us to create linkages between all things (hold the skepticism for a second; we'll get to it!).

In RDF, resources and edges are URIs. Literals are not; they are simple values. Blank nodes are not (this is what the "blank" means in the name). Everything else is, including the edges.

If you look at our example above, there are several examples of URIs.

- http://www.cambridgesemantics.com/
- http://www.cambridgesemantics.com/people/about/rob
- foaf:member (this is shorthand for http://xmlns.com/foaf/0.1/member)
- foaf:name (again, shorthand for http://xmlns.com/foaf/0.1/name)

The first one is the URI for the company Cambridge Semantics. The second is a URI for Rob, the author of this article. The other two are URIs for the edges that connect the resources (we'll say more about URIs of edges in a minute).

You should notice that a couple of the URIs above are URLs. You can click on them. They are valid Web addresses. So what makes them a URI?

In short, all URLs are URIs, but not all URIs are URLs. It's a little confusing, for sure, but the vast majority of Semantic Web practitioners stick to using URLs for all of their URIs. See the article URL vs. URI vs. URN for a succinct explanation of the differences if you're curious.

Back to the concept of universality of identity. If I have a database that contains information about myself, I would use the URI http://www.cambridgesemantics.com/people/about/rob to refer to any data relating to me. If you have another database that has other information about me, you would also use that same URI. That way, if we wanted to find all facts in both databases about me, we could query using the single universal URI.

Easy, right?

Like all theoretically simple models, things are different in the real world.

On the Limits of Universality and RDF Schema Design

There is a major problem with the concept of universality presented above.

It's impossible to get everyone everywhere to agree on a single label for every specific thing that ever was, is, or shall be.

If you read the introductions to Semantic Web technologies around the web, you'll see lots of people focus on the importance of the URI. After all, how can you connect things if you don't agree on their labels? The focus on URI definition is especially true for those creating RDF vocabularies.

What we mean by an RDF Vocabulary is essentially the set of URIs for the edges that make up RDF graphs. The edges are what relates the things in graph, and are what give it meaning. Using specific URIs is like speaking in a specific language—hence the term vocabulary. For example, in order for two Semantic Web applications to share data, they must agree on a common vocabulary. (Note: we're going to be covering RDF vocabulary and schema creation in future lessons on RDFS and OWL.)

So if two applications have to agree on vocabulary for all concepts, then it stands to reason that all vocabularies must be set ahead of time, right? Fortunately, the a priori existence of share vocabulary turns out to be helpful, but far from necessary. In our example, foaf:name is not the first URI ever created that represents the name concept, and it's OK that another URI for the name concept wasn't reused.

Fortunately, as you'll see in the SPARQL tutorials, it is very easy to translate RDF written in one vocabulary to another vocabulary. The Semantic Web technologies were built under the assumption that different people in different applications written for different purposes at different times would create related concepts that overlap in any number of ways, and therefore there are provisions and methods to make it all work together with little effort. There is no such provision in the XML or relational database worlds.

Said another way, you do not have to agree on all URIs for all things up front. In fact, it's much easier not to do so. Reuse vocabulary when possible and convenient, and don't worry too much about that when it doesn't work out.

This same universal identity conundrum also happens for resources. For example, you could consider my Linked In profile URL to be a URI representing me. This is clearly distinct from the URI that Cambridge Semantics uses, but, again, the Semantic Web offers very simple ways to merge identical concepts so that they appear as one universally.

We'll cover the details of how this works in future lessons, but for now let's return to RDF basics.

Statements and Triples

Now that you get the basics, I have to introduce some community jargon that will help you understand material you read on the Web about Semantic Web technologies.

Rather than talk in the language of nodes and edges, Semantic Web practitioners refer to statements or triples, which are representations of graph edges.

A statement or triple (they are synonymous) refers to a 3-tuple (hence triple) of the form (subject, predicate, object). This linguistic, sentential form is why RDF schemas are often called vocabularies.

As mentioned, the subject is a URI, the predicate is a URI, and the object is either a URI or a literal value.

If we represent our graph example as a set of triples, they would be:

- (csipeople:rob, foaf:name, "Rob Gonzalez")
- (csipeople:rob, foaf:member,http://www.cambridgesemantics.com/)

(Note that for brevity I'm using the namespace alias csipeople for the URI namespace http://www.cambridgesemantics.com/people/about/).

RDF graphs therefore are simply collections of triples. An RDF database is often called a triple store for this reason.

However, Semantic Web practitioners found it very difficult to deal with large amounts of triples for application development. There are lots of reasons that you would want to segment different subsets of triples from each other (simplified access control, simplified updating, trust, etc.), and vanilla RDF made segmentation tedious.

At first the community tried using reification to solve this data segmentation problem (we'll cover reification in another lesson, but the concept is essentially triples about triples), but today everyone has converged on using named graphs.

Named Graphs and Quads

A named graph is simply a collection of RDF statements that has a name (which, as you should have guessed, is a URI).

Modern triple stores all support named graphs, and they are built into SPARQL 1.1, the latest SPARQL query language specification.

When referring to a triple in a named graph, you would often use 4-tuple notation instead of 3-tuple notation. The 4-tuple is of the form:

(named graph, subject, predicate, object)

For this reason, a triple store that supports named graphs is often called a quad store, though, somewhat confusingly, triple stores themselves are often quad stores anyway. That is, if an RDF database bills itself as a triple store it probably supports named graphs. The term quad store isn't that important.

Looking at the 4-tuples, it's pretty obvious that the same statement can exist in multiple named graphs. This is by design and is a very important feature. By organizing the statement into named graphs, a Semantic Web application can implement access control, trust, data lineage, and other functionality very cleanly.

Exactly the best ways to segment triples in your application is an advanced topic that will be covered in future lessons, and is a large part of the value brought by Semantic Web platforms which will often hide the details and logic of named graph creation and segmentation to simplify application development. For now, it's simply important to know that named graphs are the segmentation and organizational mechanism of the Semantic Web.

Conclusion

This is a lot of information to cover in a single lesson, especially at this level of detail. However, it boils down to a very simple summary that will become second nature to you if you spend any time implementing Semantic Web technologies:

- RDF is a graph data model.
- RDF data are directed, labeled graphs.
- A single edge in an RDF graph is a 3-tuple that is called either a statement or triple.
- Triples are organized into named graphs, forming 4-tuples, or quads.
- RDF resources (nodes), predicates (edges), and named graphs are labeled by URIs.
- Although preferable to reuse URIs when possible, Semantic Web technologies, including OWL and SPARQL, make it easy to resolve URI conflicts, as we'll see in future lessons.

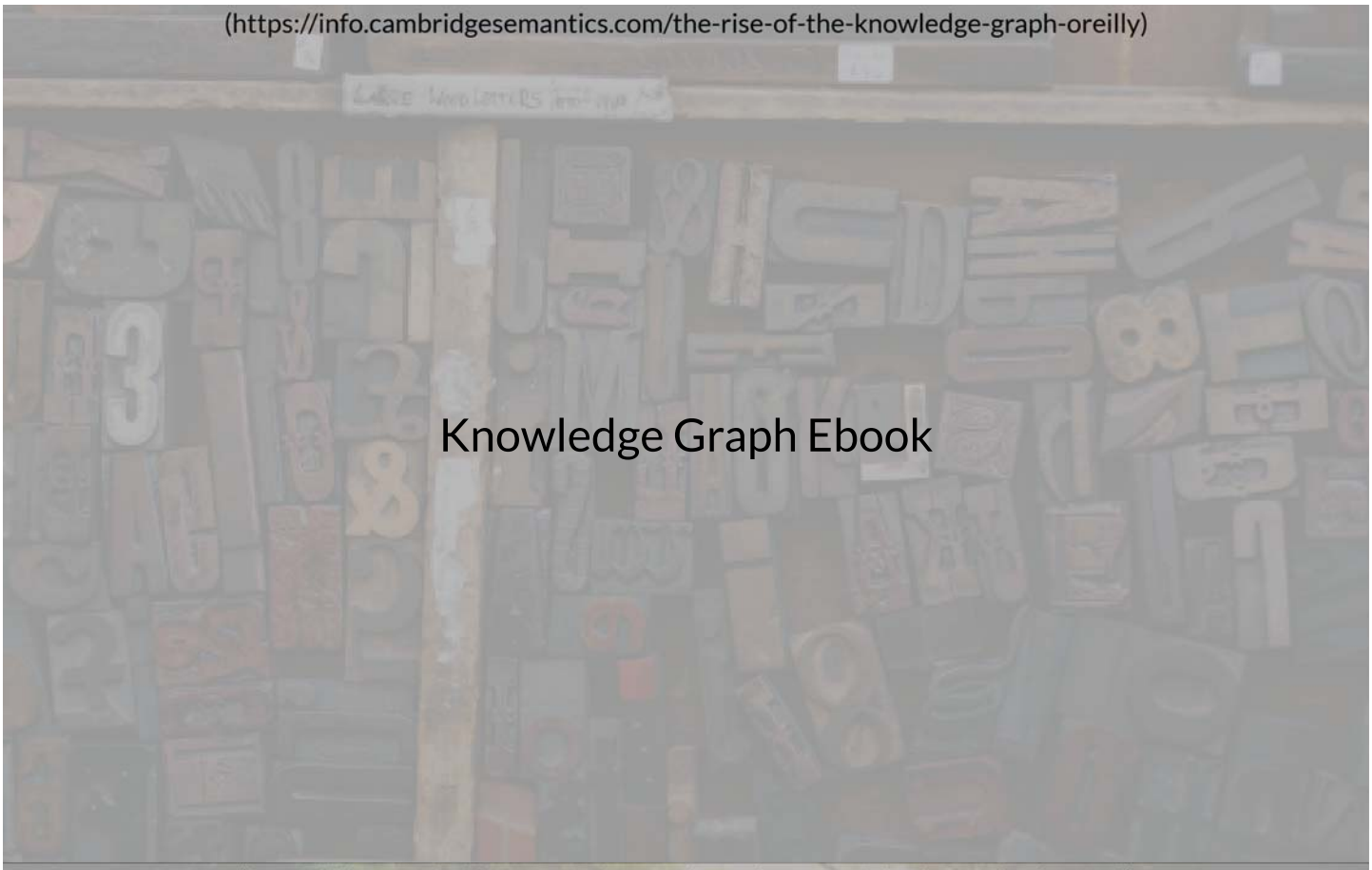
‹ PREVIOUS LESSON (HTTPS://CAMBRIDGESEMANTICS.COM/BLOG/SEMANTIC-UNIVERSITY/SEMANTIC-TECHNOLOGIES-APPLIED/SEMANTIC-WEB-ENTERPRISE/)

NEXT LESSON › (HTTPS://CAMBRIDGESEMANTICS.COM/BLOG/SEMANTIC-UNIVERSITY/LEARN-RDF/RDF-NUTS-BOLTS/)

(https://info.cambridgesemantics.com/an-integrated-data-enterprise)

(https://info.cambridgesemantics.com/what-is-data-fabric-design)

(https://info.cambridgesemantics.com/the-rise-of-the-knowledge-graph-oreilly)



Knowledge Graph Ebook

(https://blog.cambridgesemantics.com/sparql-one-standard-to-rule-them-all)

SPARQL - One Standard to Rule Them All

CAMBRIDGE SEMANTICS

1 Beacon St | 15th Floor | Boston, MA 02108