# The Resource Description Framework (RDF)

Simply defined, a knowledge graph is a network of entities, their attributes, and how they're related to one another. While these networks can be captured and stored in a variety of formats, most implementations leverage a graph based tool or database. However, within the world of graph databases, there are a variety of syntaxes or flavors that can be used to represent knowledge graphs. One of the most popular and ubiquitous is the Resource Description Framework (RDF), which provides a means to capture meaning, or semantics, in a way that is interpretable by both humans and machines.

## What is RDF?

The Resource Description Framework (RDF) is a semantic web standard used to describe and model information for web resources or knowledge management systems. RDF consists of "triples," or statements, with a subject, predicate, and object that resemble an English sentence. For example, take the English sentence: "Bess Schrader is employed by Enterprise Knowledge." This sentence has:

- A subject: Bess Schrader
- A predicate: is employed by
- An object: Enterprise Knowledge

Bess Schrader and Enterprise Knowledge are two entities that are linked by the relationship "employed by." An RDF triple representing this information would look like this:


## What is the goal of using RDF?

RDF is a **semantic** web standard, and thus has the goal of representing meaning in a way that is interpretable by both humans and machines. As humans, we process information through a combination of our experience and logical deduction. For example, I know that "Washington,

D.C." and "Washington, District of Columbia" refer to the same concept based on my experience in the world – at some point, I learned that "D.C." was the abbreviation for "District of Columbia." On the other hand, if I were to encounter a breathing, living object that has no legs and moves across the ground in a slithering motion, I'd probably infer that it was a snake, even if I'd never seen this particular object before. This determination would be based on the properties I associate with snakes (animal, no legs, slithers).

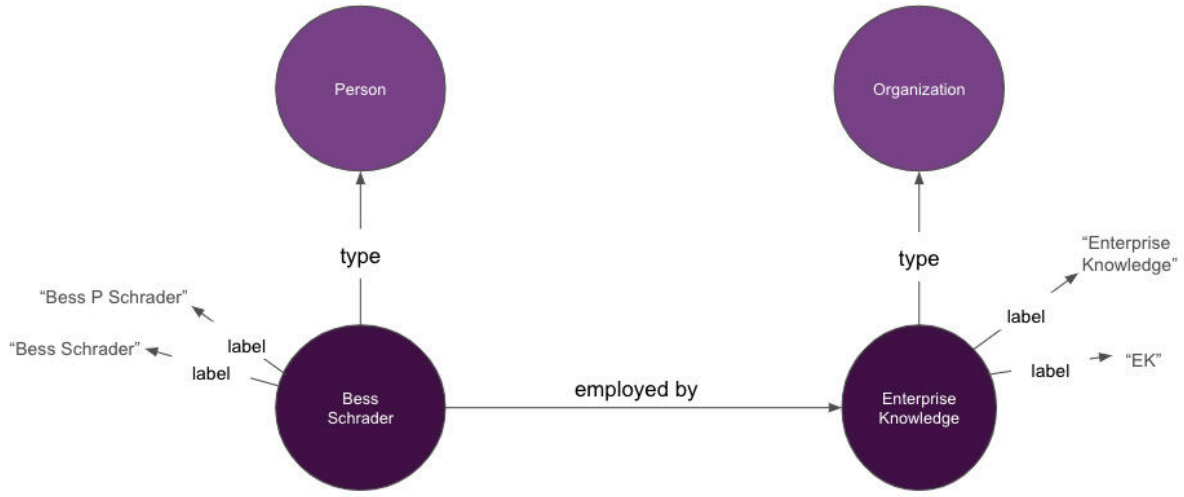
Unlike humans, machines have no experience on which to draw conclusions, so everything needs to be explicitly defined in order for a machine to process information this way. For example, if I want a machine to infer the type of an object based on properties (e.g. "that slithering object is a snake"), I need to define what a snake is and what properties it has. If I want a machine to reconcile that "Washington, D.C." and "Washington, District of Columbia" are the same thing, I need to define an entity that uses both of those labels.

RDF allows us to create robust semantic resources, like ontologies, taxonomies, and knowledge graphs, where the meaning behind concepts is well defined in a machine readable way. These resources can then be leveraged for any use case that requires context and meaning to connect and unify data across disparate formats and systems, such as semantic layers and auto-classification.

# How does RDF work?

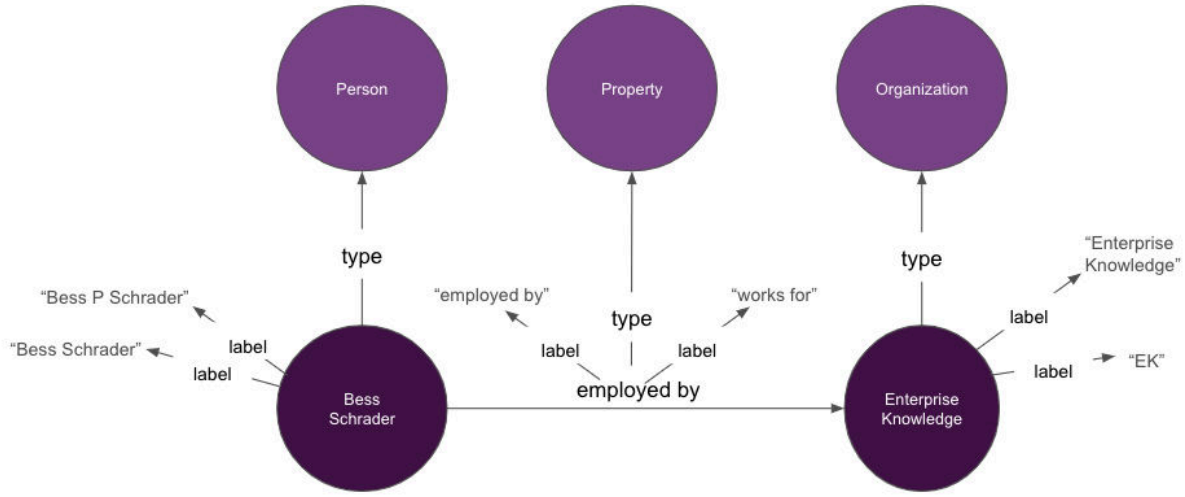
Let's go back to our single triple representing the fact that "Bess Schrader works at Enterprise Knowledge."


We can continue building out information about the entities in our (very small) knowledge graph by giving all of our subjects and objects types (which indicate the general category/class that an entity belongs to) and labels (which capture the language used to refer to the entity).




These types and labels are helping us define the semantics, or meaning, of each entity. By explicitly stating that “Bess Schrader” is a person and “Enterprise Knowledge” is an organization, we’re creating the building blocks for a machine to start to make inferences about these entities based on their types.

Similarly, we can create a more explicit definition of our relationship and attributes, allowing machines to better understand what the “employed by” relationship means. While the above diagram represents our predicate (or relationship) as a straight line between two entities, in RDF, our predicate is itself an entity and can have its own properties (such as type, label, and description). This is often referred to as making properties “first class citizens.”



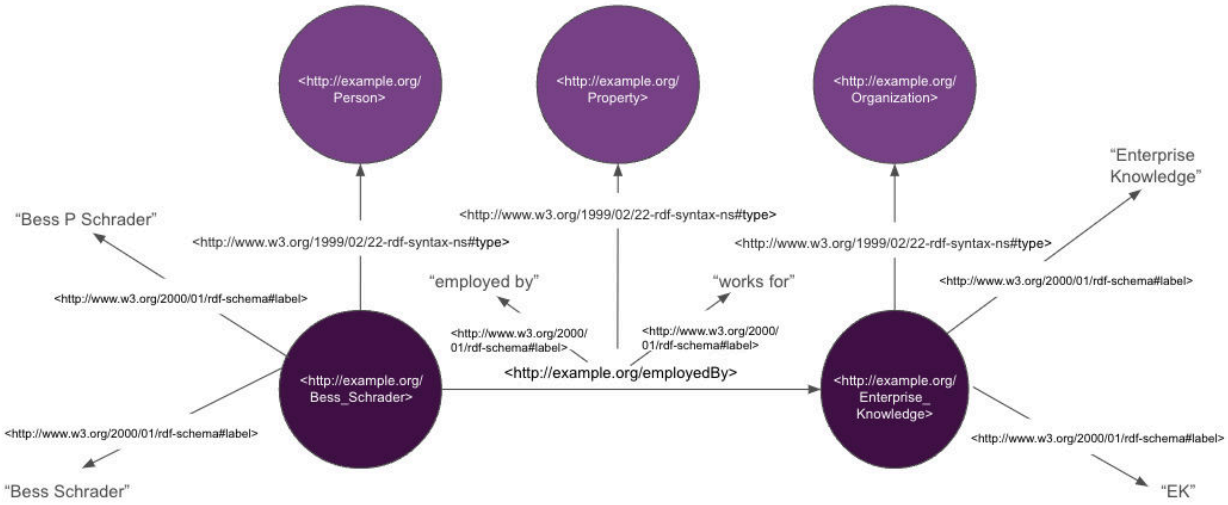
## Uniform Resource Identifiers (URIs)

But how do we actually make this machine readable? Diagrams in a blog are great in helping humans understand concepts, but machines need this information in a machine readable format. To make our graph machine readable, we'll need to leverage unique identifiers.

One of the key elements of any knowledge graph (RDF or otherwise) is the principle of "things, not strings." As humans, we often use ambiguous labels (e.g. "D.C") when referring to a concept, trusting that our audience will be able to use context to determine our meaning. However, machines often don't have sufficient context to disambiguate strings – imagine "D.C." has been applied as a tag to an unstructured text document. Does "D.C." refer to the capital city of the US, the comic book publisher, "direct current," or something else entirely? Knowledge graphs seek to reduce this ambiguity by using entities or concepts that have unique identifiers and one or more labels, instead of relying on labels themselves as unique identifiers.

RDF is no exception to this principle – all RDF entities are defined using a Uniform Resource Identifier (URI), which can be used to connect all of the labels, attributes, and relationships for a given entity.

Using URIs, our RDF knowledge graph would look like this:




These URIs make our triples machine readable by creating unambiguous identifiers for all of our subjects, predicates, and objects. URIs also enable interoperability and the ability to share information across multiple systems – because these URIs are *globally* unique, any two systems that reference the same URI should be referring to the same entity.

# What are the advantages to using RDF?

The RDF Specification has been maintained by the World Wide Web Consortium (W3C) for over two decades, meaning it is a stable, well documented framework for representing data. This makes it easy for applications and organizations to develop RDF data in an interoperable way. If you create RDF data in one tool and share it with someone else using a different RDF tool, they will still be able to easily use your data. This interoperability allows you to build on what's already been done — you can combine your enterprise knowledge graph with established, open RDF datasets like Wikidata, jump-starting your analytic capabilities. This also makes data sharing and migration between internal RDF systems simple, enabling you to unify data and reducing your dependency on a single tool or vendor.

The ability to treat properties as "first-class citizens" with their own properties allows you to store your data model along with your data, explaining what properties mean and how they should be used. This reduces ambiguity and confusion for both data creators, developers, and data consumers. However, this ability to treat properties as entities also allows organizations to standardize and connect existing data. RDF data models can store multiple labels for the same property, enabling them to act as a "Rosetta Stone" that translates metadata fields and values across systems. Connecting these disparate metadata values is crucial to being able to effectively retrieve, understand, and use enterprise data.

Many implementations of RDF also support inference and reasoning, allowing you to explore previously uncaptured relationships in your data, based on logic developed in your ontology. This reasoning capability can be an incredibly powerful tool, helping you gain insights from your business logic. For example, inference and reasoning can capture information about employee expertise – a relationship that's notoriously difficult to explicitly store. While many organizations attempt to have employees self-select their skills or areas of expertise, the completion rate of these self-selections is typically low, and even those that do complete the selection often don't keep them up to date. Reasoning in RDF can leverage business logic to automatically infer expertise based on your organization's data. For example, if a person has authored multiple documents that discuss a given topic, an RDF knowledge graph may infer that this person has knowledge of or expertise in that topic.

# What are the disadvantages to using RDF?

To fully leverage the benefits of RDF, entities must be explicitly defined (see best practices below), which can require burdensome overhead. The volume and structure of these assertions, combined with the length and format of Uniform Resource Identifiers (URIs), can make getting started with RDF challenging for information professionals and developers used to working with more straightforward (albeit more ambiguous) data models. While recent advancements in generative AI have great potential to make the learning curve to RDF less onerous via

human-in-the-loop RDF creation processes, learning to create and work with RDF still poses a challenge to many organizations.

Additionally, the “triple” format (subject - predicate - object) used by RDF only allows you to connect two entities at a time, unlike labeled property graphs. For example, I can assert that “Bess Schrader -> employed by -> Enterprise Knowledge,” but it’s not very straightforward in RDF to then add additional information about that relationship, such as what role I perform at Enterprise Knowledge, my start and end dates of employment, etc. While a proposed modification to RDF called RDF* (RDF-star) has been developed to address this, it has not been officially adopted by the W3C, and implementation of RDF* in RDF compliant tools has occurred only on an ad hoc basis.

# What are some best practices when using RDF to create a knowledge graph?

RDF, and knowledge graphs in general, are well known for their flexibility – there are very few restrictions on how data must be structured or what properties must be used for their implementation. However, there are some best practices when using RDF that will enable you to maximize your knowledge graph’s utility, particularly for reasoning applications.

## All concepts should be entities with a URI

The guiding principle is “things, not strings”. If you’re describing something with a label that might have its own attributes, it should be an entity, not a literal string.

## All entities should have a label

Using URIs is important, but a URI without at least one label is difficult to interpret for both humans and machines.

## All entities should have a type

Again, remember that our goal is to allow machines to process information similarly to humans. To do this, all entities should have one or more types explicitly asserted (e.g. “Washington, D.C” might have the type “City”).

## All entities should have a description

While using URIs and labels goes a long way in limiting ambiguity (see our “D.C.” example above), adding descriptions or definitions for each entity can be even more helpful. A well written description for an entity will leave little to no question around what this entity represents.

Following these best practices will help with reuse, governance, and reasoning.

Want to learn more about RDF, or need help getting started? Contact us today.