**RDF**

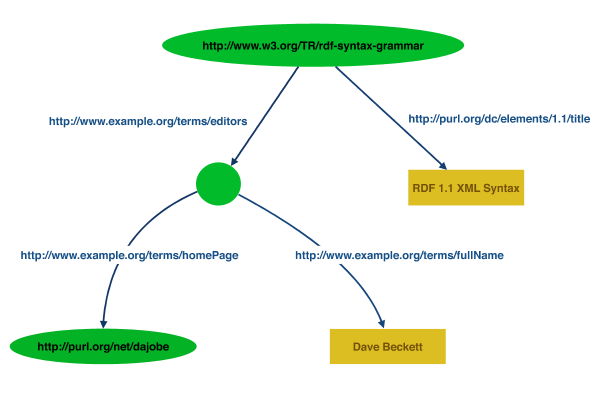
If you're eager to understand what the semantic web is and how it works, you must first understand how it stores data. We start from the ground up by outlining the graph database - the data storage model used by the semantic web.

|  |
| --- |
| RDF is a common acronym within the semantic web community because it forms one of the basic building blocks for forming the web of semantic data. What it defines is a type of database which you may not be immediately familar with: something called a **graph database**.  Although it might not be familiar to you, it is the type of database that builds the semantic web, globally. We will learn why in these tutorials.  During this lesson, you will learn what a graph database is, how RDF defines one, and visualise graph data so you can get a feel of what it looks like. We will begin by comparing hierarchical, relational, and graph databases to see how they are different. 1.1 Introducing The Graph Database Relational database vs graph database  For most types of data storage, there is the concept of some elements of data (whether they be for example data nodes or data tables) having more precedence, or importance, over other elements.  For example, take an XML document. An XML document typically contains nodes of information each with a parent node. At the root of the document is the highest level node, which has no parent.  Take a look at the illustration above. In a data graph, there is no concept of roots (or a hierarchy). A graph consists of resources related to other resources, with no single resource having any particular intrinsic importance over another. An Example Of A Data Graph It's easiest first to look at a series of statements about how things relate to each other and to visualize these as a graph before looking at how these relationships might be expressed in RDF. Look at the following statements describing the relationship between a dog (called Bengie) and a cat (called Bonnie):  Bengie is a dog. Bonnie is a cat. Bengie and Bonnie are friends.  Using these three simple statements, let's turn this into a data graph:    The relationships implied by this graph are fairly intuitive but to be thorough let's review them. We can can see that our two things - identified by "Thing 1" and "Thing 2" - have the properties **name**, **animalType** and **friendsWith**.  From this, we can see that "Thing 1"'s name is Bengie, and "Thing 2"'s name is Bonnie. "Thing 1" is a dog, and "Thing 2" is a cat. And finally, both are friends with each other (implied by the friendsWith property pointing in both directions).  **Important Point** The arrows in the above diagram are properties, sometimes in RDF terminology called predicates. Remember for now that the terms property and predicate are interchangable, and that it is the arrows that describe the properties in the graph. 1.2 A Starting Example Of RDF <?xml version="1.0" encoding="UTF-8"?>  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:dc="http://purl.org/dc/elements/1.1/"  .xmlns:region="http://www.country-regions.fake/">  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Oxford">  <dc:title>Oxford</dc:title>  <dc:coverage>Oxfordshire</dc:coverage>  <dc:publisher>Wikipedia</dc:publisher>  <region:population>10000</region:population>  <region:principaltown rdf:resource="http://www.country-regions.fake/oxford"/>  </rdf:Description>  </rdf:RDF> 1.3 The RDF Statement (Triple) The RDF/XML above (between the <rdf:Description> tags inclusive) is called an RDF statement, or sometimes called an RDF triple. Of the two, triple is the most helpful term as it describes the breaking of the statement into its three constituent parts: the subject, predicate, and object of the statement.  It is easiest first to illustrate these terms in the form of a simple graph. Look at the following graph of data describing the color of a T-shirt:  Example graph showing color property of a T-shirt  In terms of the simple graph above, the:   * **Subject** is the T-shirt * **Predicate** (property) is the color * **Object** is white   **Important Point** RDF, whilst the foundation of defining data structures for the semantic web, does not in itself describe the semantics, or meaning, behind the data. This will come later when we introduce RDFS (RDF Schema) and OWL (Web Ontology Language). Don't worry about these for now. First, we need to learn how RDF structures data and relationships and how that differs from the more familiar ways of storing data - we need to paradigm shift from the relational or hierarchical means of modelling data to a graph model.  Let's look at this in terms of a simple RDF/XML statement:  <?xml version="1.0" encoding="UTF-8"?>  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:feature="http://www.linkeddatatools.com/clothing-features#">  <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">  <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/  </rdf:Description>   </rdf:RDF> |

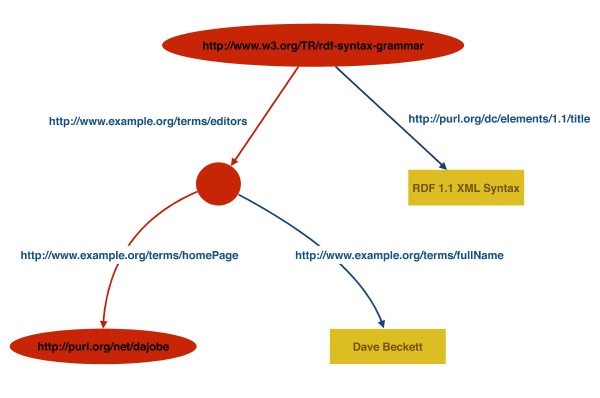
### how it defined [statements](http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-Concepts) comprising a subject, a predicate (property), and an object. The **subject**->**predicate**->**object** relationship is called a triple.

|  |
| --- |
| If the graph data model is the model the semantic web uses to store data, RDF is the format in which it is written. In the first lesson, we looked at graph data and introduced RDF. Now, we go further and describe the groundwork you need to write graph data yourself using RDF/XML - one of the most popular RDF formats on the web.  We then looked briefly at RDF (Resource Description Framework) format, and saw how it defined [statements](http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-Concepts) comprising a subject, a predicate (property), and an object. The **subject**->**predicate**->**object** relationship is called a triple.  During this lesson, you will learn step-by-step how to build your own RDF statements and understand them visually in a graph. This will then allow you to start thinking about adding semantics (meaning) to your data and understand the profound advantages this offers.  To do this best, let's build a simple RDF document, step-by-step. 2.1 Building An RDF DocumentAdd The RDF Document Root Tag First, add the RDF root node:  1.<rdf:RDF  2.xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">  4.<!-- Body Code Omitted -->  6.</rdf:RDF>  If you look at line 02, you will see the standard W3.org namespace **http://www.w3.org/1999/02/22-rdf-syntax-ns#**. This namespace tells any machine reader that the enclosing document is an RDF document, and that the rdf:RDF tag resides in this namespace.  This namespace, and the RDF node, forms the root of all RDF documents. Add A Statement An RDF document can contain more than one statement. For simplicity, we'll only add one. Start by adding a an rdf:Description tag, which in RDF/XML can contain one or more statements about the same subject:  01.<rdf:RDF  02.xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">  04.<rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">  06.<!-- Statement Code Omitted -->  08.</rdf:Description>  10.</rdf:RDF>  The rdf:Description tag simply means "I'm going to describe something (a subject) and I'm giving it the unique ID **http://www.linkeddatatools.com/clothes#t-shirt**". Add Predicates There's no point in saying you're going to describe something, give it a unique ID but then not describe anything about it. RDF statements describe the characteristics of their subjects using properties, or predicates in RDF terminology.  For simplicity, let's start by adding one predicate: the size of our T-shirt.  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:feature="http://www.linkeddatatools.com/clothing-features#">  <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt"  <feature:size>12</feature:size>  </rdf:Description>  .</rdf:RDF>  See line 07. This simply says "The subject has a property with name **feature:size** which has the literal value 12". In RDF terminology, this is a statement.  Finally, let's add one more predicate: the color of the T-shirt.  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:feature="http://www.linkeddatatools.com/clothing-features#">  <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">  <feature:size>12</feature:size>  <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>  </rdf:Description>  </rdf:RDF>  You'll notice this one isn't quite the same as the last one. Whereas the last one had the literal value 12, this one is referring to the subject (ID) of another statement. That's right - objects in RDF can refer (reference) the subjects of other statements.  **Remember** A subject in an RDF document may also be referenced as a object of a property in another RDF statement (in the resource attribute). This can be a confusing concept for those starting out with RDF.  So this simply says "The subject has a property with name **feature:color** with object referring to the statement with ID **http://www.linkeddatatools.com/colors#white**". 2.2 Breaking Down The Statement Now we've looked at a simple example of an RDF document, let's formalize what we've learned and break the statement into its component parts:  1.<rdf:Description rdf:about="subject">  2.<predicate rdf:resource="object" />  3.<predicate>literal value</predicate>  4.<rdf:Description>  Here you see the subject of the statement (what the statement is about), and the two forms of predicates (literal values and resources, which reference other RDF statements).  **Note** The **rdf:Description** RDF/XML element allows you to group one or more statements into a single container. The above general form actually contains two statements referring to the same subject, but with two predicates and objects: a resource and a literal. 2.3 A More Thorough Example <?xml version="1.0" encoding="UTF-8"?>  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:dc="http://purl.org/dc/elements/1.1/"  xmlns:region="http://www.country-regions.fake/">  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Oxford">  <dc:title>Oxford</dc:title>  <dc:coverage>Oxfordshire</dc:coverage>  <dc:publisher>Wikipedia</dc:publisher>  <region:population>10000</region:population>  <region:principaltown rdf:resource="http://www.country-regions.fake/oxford"/>  </rdf:Description>  </rdf:RDF>  To test your understanding and point out any areas where you need to recap, see if you can identify on the RDF document above the:   * **Subject** of the statement * **Predicates** of the statement - including whether they are resources or literals * **Objects** referenced by the resource predicates   Once you have understood RDF documents, and how they relate to data graphs, you are ready for our next lesson introducing the idea of modeling semantics into RDF graph data. 2.4 A Quick Recap Of URIs And XML Namespaces Unique IDs that we've been using so far such as **http://www.linkeddatatools.com/clothes#t-shirt** are called [Uniform Resource Identifiers](http://www.w3.org/Addressing/URL/URI_Overview.html), or URIs for short. We've been using URIs to give a unique ID to the subjects, predicates or objects of statements so far without really saying why.  Because URIs are so crucial to the driving purpose behind RDF - to make data exchangable globally - we will make a quick recap of URIs now. If you already understand URIs, you can skip this section. XML Namespace URIs Look back again at our example RDF document we've built. See that the T-shirt size predicate has the name **feature:size** on line 07 below:  01.<rdf:RDF  02.xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  03.xmlns:feature="http://www.linkeddatatools.com/clothing-features#">  05.<rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">  07.<feature:size>12</feature:size>  08.<feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>  10.</rdf:Description>  12.</rdf:RDF>  On line 03, notice that we've defined the XML namespace **feature** by giving it the namespace URI **http://www.linkeddatatools.com/clothing-features#**.  The purpose of this namespace is simply to avoid name conflicts with tags of the same name: other tags with the name "size" could be defined with other namespace URIs, and an RDF reader would still be able to tell that they were different properties even though they had the same tag name.  To get a fully qualified URI for **feature:size**, simply substitute the prefix **feature:** with its namespace URI, obtaining the full name **http://www.linkeddatatools.com/clothing-features#size**.  **Note** XML namespace URIs in RDF are used to distinguish between properties with the same (tag) name. To get the fully qualified URI, simply substitute the namespace prefix with the namespace URI.  Now we can state the graph with fully qualified URIs.  Graph showing fully qualified URIs  As you can see, it's not always easy or convenient to represent URIs in full in RDF graph diagrams. Often, shorthand versions are used instead (i.e. only using the namespace prefix).  **You have completed this lesson. You should now understand the following:**   * How to write your own basic RDF documents in RDF/XML * Understand how and why RDF uses URIs to identify subjects, predicates and objects * How to relate an RDF document to a corresponding data graph with fully qualified UR |

This document defines an XML syntax for RDF called RDF/XML in terms of Namespaces in XML, the XML Information Set and XML Base. In order to encode the graph in XML, the nodes and predicates have to be represented in XML terms — element names, attribute names, element contents and attribute values. RDF/XML uses XML [QNames](http://www.w3.org/TR/1999/REC-xml-names-19990114/" \l "NT-QName) as defined in Namespaces in XML [[XML-NAMES](https://www.w3.org/TR/rdf-syntax-grammar/#bib-XML-NAMES)] to represent IRIs. All QNames have a [namespace name](http://www.w3.org/TR/1999/REC-xml-names-19990114/#dt-NSName) which is an IRI and a short [local name](http://www.w3.org/TR/1999/REC-xml-names-19990114/#NT-LocalPart). IA graph can be considered a collection of paths of the form node, predicate arc, node, predicate arc, node, predicate arc, ... node which cover the entire graph. In RDF/XML these turn into sequences of elements inside elements which alternate between elements for nodes and predicate arcs. This has been called a series of node



An RDF graph is given in [Figure 1](https://www.w3.org/TR/rdf-syntax-grammar/#figure1) where the nodes are represented as ovals and contain their IRIs where they have them, all the predicate arcs are labeled with IRIs and string literals nodes have been written in rectangles.



1. Node with IRI http://www.w3.org/TR/rdf-syntax-grammar
2. Predicate Arc labeled with IRI http://example.org/terms/editor
3. Node with no IRI
4. Predicate Arc labeled with IRI http://example.org/terms/homePage
5. Node with IRI http://purl.org/net/dajobe/

**Striped RDF/XML (nodes and predicate arcs)**

<rdf:Description>

<ex:editor>

<rdf:Description>

<ex:homePage>

<rdf:Description>

</rdf:Description>

</ex:homePage>

</rdf:Description>

</ex:editor>

</rdf:Description>

The [Figure 2](https://www.w3.org/TR/rdf-syntax-grammar/#figure2) graph consists of some nodes that are IRIs (and others that are not) and this can be added to the RDF/XML using the rdf:about attribute on node elements to give the result in [Example 2](https://www.w3.org/TR/rdf-syntax-grammar/#example2):

EXAMPLE 2

**Node Elements with IRIs added**

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">

<ex:editor>

<rdf:Description>

<ex:homePage>

<rdf:Description rdf:about="http://purl.org/net/dajobe/">

</rdf:Description>

</ex:homePage>

</rdf:Description>

</ex:editor>

</rdf:Description

**Complete description of all graph paths**

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">

<ex:editor>

<rdf:Description>

<ex:homePage>

<rdf:Description rdf:about="http://purl.org/net/dajobe/">

</rdf:Description>

</ex:homePage>

</rdf:Description>

</ex:editor>

</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">

<ex:editor>

<rdf:Description>

<ex:fullName>Dave Beckett</ex:fullName>

</rdf:Description>

</ex:editor>

</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">

<dc:title>RDF 1.1 XML Syntax</dc:title>

</rdf:Description>

Taking [Example 3](https://www.w3.org/TR/rdf-syntax-grammar/#example3), there are two node elements that can take multiple property elements. The subject node with IRI http://www.w3.org/TR/rdf-syntax-grammar has property elements ex:editor and ex:title and the node element for the blank node can take ex:homePage and ex:fullName.

**Using multiple property elements on a node element**

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">

<ex:editor>

<rdf:Description>

<ex:homePage>

<rdf:Description rdf:about="http://purl.org/net/dajobe/">

</rdf:Description>

</ex:homePage>

<ex:fullName>Dave Beckett</ex:fullName>

</rdf:Description>

</ex:editor>

<dc:title>RDF 1.1 XML Syntax</dc:title>

</rdf:Description>

Completing the Document: Document Element and XML Declaration

rdf:RDF element is also used to declare the XML namespaces that are used, although that is not required. When there is only one top-level node element inside rdf:RDF, the rdf:RDF can be omitted although any XML namespaces must still be declared

<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:dc="http://purl.org/dc/elements/1.1/"

xmlns:ex="http://example.org/stuff/1.0/">

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"

dc:title="RDF1.1 XML Syntax">

<ex:editor>

<rdf:Description ex:fullName="Dave Beckett">

<ex:homePage rdf:resource="http://purl.org/net/dajobe/" />

</rdf:Description>

</ex:editor>

</rdf:Description>

</rdf:RDF>

XML Literals: rdf:parseType="Literal"

<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:ex="http://example.org/stuff/1.0/">

<rdf:Description rdf:about="http://example.org/item01">

<ex:prop rdf:parseType="Literal" xmlns:a="http://example.org/a#">

<a:Box required="true">

<a:widget size="10" />

<a:grommit id="23" />

</a:Box>

</ex:prop>

</rdf:Description>

</rdf:RDF>

RDF has a set of container membership properties and corresponding property elements that are mostly used with instances of the rdf:Seq, rdf:Bag and rdf:Alt classes which may be written as typed node elements.

<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<rdf:Seq rdf:about="http://example.org/favourite-fruit">

<rdf:\_1 rdf:resource="http://example.org/banana"/>

<rdf:\_2 rdf:resource="http://example.org/apple"/>

<rdf:\_3 rdf:resource="http://example.org/pear"/>

</rdf:Seq>

</rdf:RDF>

This is a few lines from a CD-list:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Title** | **Artist** | **Country** | **Company** | **Price** | **Year** |
| Empire Burlesque | Bob Dylan | USA | Columbia | 10.90 | 1985 |
| Hide your heart | Bonnie Tyler | UK | CBS Records | 9.90 | 1988 |

<?xml version="1.0"?>

<rdf:RDF

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description

rdf:about="http://www.recshop.fake/cd/Empire Burlesque">

<cd:artist>Bob Dylan</cd:artist>

<cd:country>USA</cd:country>

<cd:company>Columbia</cd:company>

<cd:price>10.90</cd:price>

<cd:year>1985</cd:year>

</rdf:Description>

<rdf:Description

rdf:about="http://www.recshop.fake/cd/Hide your heart">

<cd:artist>Bonnie Tyler</cd:artist>

<cd:country>UK</cd:country>

<cd:company>CBS Records</cd:company>

<cd:price>9.90</cd:price>

<cd:year>1988</cd:year>

</rdf:Description>

.

.

.

</rdf:RDF>