Epimorphics
CONNECTING DATA

WHAT IS LINKED DATA?

Linked data is an approach to publishing and sharing data on the web.

What does that mean? Surely if I just put an Excel or PDF file on a web site then I'm publishing data? What makes linked data different and what's the benefit?

Yes, it is certainly possible to publish data by simply putting a document, a PDF or HTML file, on a web site. That does mean that you can provide a narrative to explain the data. If your only purpose is for people to read your report then that's a fine way to publish. However, any data behind such a document is locked away and inaccessible. People can't take that data and analyse and re-present it. They can't combine it with other data to provide new services or discover new insights.

Publishing the data as structured files like spreadsheets does allow the data to be analysed and used in applications but leads to a series of static data silos. The structure and coding of the data is typically not explained in machine processable form, making it hard to use the data safely without help.

There is no connection between the information in one spreadsheet and the next, each is a separate data island. The data is static; if a new data entry is available an hour later do you have to re-download and figure out what has changed? If you only want a slice of the data can you do that without having to download the whole spreadsheet?

APPLYING PRINCIPLES OF THE WEB

Linked data is about applying the principles of the web to sharing data, and doing so at a deeper level that just putting up one big monolithic file. Instead we give each thing in the data an individual identity or URI.
For example if we were publishing a set of information about the addresses of schools in the UK we would have a URI for each school. Then we can publish information about each school as small sets of statements, for example stating its name, its latitude/longitude or the administrative district it falls in. If the things we link to, such as the administrative district, are themselves represented by URIs then we are creating a web of linked information.
The district URI might give information on the boundary of the district and in turn link to other information on the current political makeup of the administration and its education budget.

This gives us a number of benefits:

- The data is placed in context, each item has a web address through which it can be annotated and referenced, allowing explanations and implications to be linked back directly to the data
- The data is linked (to its information model and to related data) enabling information to be combined across silos, enhanced by combination with third party data sources and contextualized
- The data is accessible at a fine grain over the web so that downstream applications can run from the live data, ensuring it is up to date, while not preventing them using static data dumps if preferred

The linked data approach builds upon a number of key web standards:

URIs
URIs are used to identify anything of interest in the data, including the entities the data is about (e.g. a particular school), the classes or concepts involved in the data (e.g. the notion of a School) and the properties that are available to describe schools (e.g. its name, location, or controlling authority). While we tend to talk about URIs, which include various different identifier schemes, the recommendation is to use http URLs so that standard web client software can fetch (GET) from those URLs and hope to discover useful information on them and onward links to other related data.

RDF
The recommended approach to representing the data itself is to use RDF the Resource Description Framework. RDF presents information as a series of simple statements. Each statement (sometimes also called a triple) says that some subject has some property with some value. The subject of the statement is normally identified by a URI, as is the property or predicate being described. The value of the property, the object of the statement, can either be a literal value (a string or number for example) or it can be another URI.

RDF EXAMPLE

Where *so:* `401874` represents the URI http://education.data.gov.uk/id/school/401874 and **`rdfs:label`** represents the URI http://www.w3.org/2000/01/rdf-schema#label The things being described and linked in these statements are typically called resources, hence the name Resource Description Framework. Thus RDF is intrinsically well-suited to the linked data approach of linking together things identified by URIs.

Apart from its simplicity and the fact that it is fundamentally based on identifying resources through use of URIs, RDF has one other key characteristic that makes linked data work well – it is schemaless or open world. When representing data using object oriented modelling such as UML or in common database design methods one thinks in terms of data as being held in containers (objects or rows of values). To publish, store or query such data you need to know what the allowed values are in the containers. In contrast in RDF everything is represented as statements. Whether the statements are about attributes, such as a label, or relational links such as `hasAdministrator`, there is no similar notion of a fixed shape container. This means there is no requirement to do a global, top-down schema design to agree, for example, everything that can be said about a school. Instead different authorities are free to publish different statements about the same school using their own sets of properties. This gives an intrinsic flexibility and resilience to data published this way. However, we still need some way of publishing agreed on vocabularies of terms that can be used.

VOCABULARY STANDARDS – RDFS, OWL, SKOS

To complement this open world use of terms we do need some way to publish vocabularies that define what terms are available and how those terms relate. These might range from simple controlled lists of terms, such as might be used in a library for categorizing documents, through to rich logical models of a domain. The formal term for such a shared, well specified vocabulary is an ontology. Though in linked data the level of formality and depth of modelling can vary widely according to the needs of a particular application.

Of course, we identify the class by a URI such as http://education.data.gov.uk/def/school/School not by its label.

We commonly shorten URIs by defining a common prefix for a set of terms, so that if we use school: to represent the prefix http://education.data.gov.uk/def/school/ then the earlier URI would be shown as school:School.

At heart this approach to modelling is quite straightforward. We want to represent the types or categories into which our resources can be grouped, these are called classes.

So for example in defining an ontology to represent school information we might have a class called School. Similarly we need properties that can be used to link a resource to another resource or to a simple literal value. Again we use URIs to identify these properties, for example the rdfs:label used in the above example is standard way of attaching a name or text label to any resource.

The standards that support this provide for a range of sophistication from simply listing classes and properties, through to expressing rich axioms that define relationships between the classes and properties. RDFS, the RDF Vocabulary Language (misleadingly the 'S' originally stood for 'schema'), provides a base foundation. In particular it defines terms such as rdf:type which links a resource to its class. Then OWL, the Web Ontology Language, provides more sophisticated capabilities on top of this.

In some situations we simply want a set of controlled terms that we are going to use as symbolic labels. They are not intended to model the domain in the way that the classes and properties do in RDFS and OWL. We simply want some way to denote a category separate from its textual label. The standard for that situation is SKOS or Simple Knowledge Organisation System.

SPARQL

The final key piece of the puzzle is query. If we publish data as linked data then there is a certain amount that can be achieved by following your nose – that is starting from the URI of one resource, fetching it and following links to other resources. However, in many situations we want to actually query some aggregation of this data – to find all resources matching some pattern. For this we need a query language suited to this graph-like connected web of data. That query language is SPARQL.

SEMANTIC WEB

This technology stack was originally developed as part of a W3C initiative called the semantic web. The initiative was sometimes mis-interpreted as rather top down, AI-like, approach to the whole web – which was not the case. Some people may use semantic web and linked data interchangeably, others emphasise that linked data is a particular pragmatic way of applying the technology stack. In our view the technology stack is a very solid, practical way of modelling, sharing and working with data on the web – whatever label you want to give it.