Data integration at scale: Linked Data

Apply principles for connecting large, independent web data sets

Brian Sletten
June 22, 2015

Linked Data technologies effectively make one data set out of many. Now that you understand the Resource Description Framework (RDF) data model and the SPARQL Protocol and RDF Query Language (SPARQL), learn how those standards combine with the web's architecture to create and use easily consumable interconnected data.

In the first two articles in this series ("Creating webs of data with RDF" and "Query RDF data with SPARQL"), you learned about the Resource Description Framework (RDF) and the SPARQL Protocol and RDF Query Language (SPARQL) — two World Wide Web Consortium (W3C) standards for creating portable, queryable, network-friendly data. The graph model of RDF makes it easy to accumulate information about a topic from various sources. You now know how to pull RDF data to you via HTTP for local queries, or push a query to a standards-compliant server to avoid transferring unrelated data. In this Data integration at scale installment, you'll learn how RDF and SPARQL combine with the architecture of the web to create and use Linked Data.

Linked Data principles

To encourage consistency in how data is published on the web, Tim Berners-Lee defined four principles of Linked Data:

1. Use Uniform Resource Identifiers (URIs) as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL).
4. Include links to other URIs so that they can discover more things.

About this series

This series introduces, explores, and applies global standards that address large-scale data-integration challenges faced by developers, architects, and data managers daily. The cross-
platform, language- and application-independent technologies that the series covers enable
the integration of information in databases, documents, spreadsheets, and service APIs. The
data models and tools that you'll learn about can make your work easier and have a material
impact on your organization.

The motivation behind these principles shouldn’t require much explanation this far into the series,
but for clarity, I'll go through it quickly.

First, the purpose of a naming scheme is to make references in a shared context. These
references should be consistent, unambiguous, and collision-free. The URI standard provides a
naming-scheme scheme: a scheme for creating naming schemes. As long as you know how to
parse, represent, and potentially store URIs in your system, you can accept identifiers from any
other systems that abide by the standard. Such systems can include code written and deployed
today that accepts name references to new, URI-conforming names in the future.

**URNs**

The use of URIs doesn’t mean that ISBNs are useless for the Semantic Web. A Uniform
Resource Name (URN) is a type of URI that allows an external naming scheme to be
mapped into the URI space via a namespace prefix. A valid URN for a book might be:
urn:isbn:978-1608454303.

Other global naming systems exist. A common scheme is the **International Standard Book Number**
(ISBN). ISBNs have been crucial for standardizing references to books over the years. The
scheme's success is due mainly to the fact that support for the naming system has reduced costs
and errors for the book publishing and distribution markets. Unfortunately, ISBNs refer only to
books. Magazines, musical scores, and audiovisual products (movies, TV shows, broadcast
sporting events) all have separate identifier schemes. The subjects of books can be specified
using a hierarchical categorization scheme such as the **Dewey Decimal Classification** system,
but that’s another incompatible identifier system. Academic researchers can be identified via
**ORCID** identifiers, but nonacademics have no such system available. Therefore, to indicate that an
individual (academic) book was written by a specific researcher about a known topic would involve
not just three separate identifiers, but three separate schemes! Having a standard scheme to refer
to all of these things clearly makes sense.

Notice that the guidance from Berners-Lee isn’t that everyone needs to use the same URIs. You
gain basic interoperability simply by using the URI standard. It’s nice when people do agree on
what to call things, but they’re not required to. This is true for both the node and link identifiers in
RDF graphs.

Second, even though any URI-aware system can consume a reference to a URI identifier in an
external data set, users of the system might not recognize the identifier. An unfamiliar identifier
requires a means to look up what it points to. To find out anything about the named entity, the
ingesting system must know about such a service or have a means to discover it. As a result, the
dependencies and coupling that a consumer application has to support to use specific naming
schemes increase.

The second principle adds tremendous value for exchanging data. If your system can consume
URIs, if they are resolvable URIs (URLs), then — to learn more about what they refer to — you
can treat them like any other web resource and issue a GET request to them. No separate service needs to be discovered, and no new dependency exists beyond HTTP and its Uniform Interface. The name is both an identifier and a handle by which you can learn more.

The third principle clarifies that — in addition to whatever other custom formats you want to return when your resources are resolved — if you allow for standard serializations of standard data models, the resolving system doesn't need to know anything extra to parse the resulting structure. The system might not know what the identifiers are, but, by the second principle, it can resolve them whenever it wants to learn more. In addition to the standard serialization formats, supporting standard query mechanisms such as the SPARQL Protocol enables clients to ask questions of your data.

“Linked Data is a fundamentally different approach that works at a level of productivity, scale, and flexibility that's difficult to imagine if all you've ever had at your disposal were enterprise and programming-language-related solutions.”

Because the first principle doesn't require the use of standard identifiers — only standard identifier schemes — it's guaranteed that multiple names will be given to the same things in different data sets. This issue can be resolved in many ways, but I won't take the time to explain how in depth. In general, you can use higher-order semantic relationships such as owl:sameAs from the Web Ontology Language (OWL) to equate identifiers permanently. From then on, you can use any reasoning system that understands OWL semantics to query for any of the equivalent resources and get properties from all of them. The salient point here is that such mechanisms give you ways to connect your terms to other terms. Doing so enriches your data and helps support discoverability among the data sets.

Overall, the principles apply well to both public and private data. Don't think of all of these technologies as only being for free, public data that you want to give away. At the end of the day, they're all web resources and you can put them behind firewalls, paywalls, and authentication and authorization models. The goal is to break down many of the problems of connecting information among various data sources with technologies that work at scale. Meeting that goal helps drive down the cost of integration to almost nothing compared to more expensive, fragile, and time-consuming technologies not based on network-friendly standards.

You need look no further than the Linking Open Data community project to see these ideas implemented at scale.

**Linking Open Data project**

In 2007, a small group of people — the Linking Open Data (LOD) community project — started to connect a series of public data sets. In Figure 1, you see the first 12 data sets that were tied together — including DBpedia, GeoNames, and US Census information.
Figure 1. The Linking Open Data project cloud in 2007

I'll talk about DBpedia more in a minute. For now, start with the fact that information extracted from Wikipedia about the subject Auburn, California is available from DBpedia. Other information about Auburn might have been gathered in the 2000 U.S. Census and some might come from the GeoNames project. The three data sets use different identifiers for the same thing (Auburn), but with a little bit of poking around behind the scenes, you can see that DBpedia uses the OWL sameAs relationship to connect the terms. Now you can use any one of the three terms to query the data via an OWL-based reasoner and retrieve all of the results. (Again, how and why this works is beyond the scope of this article.)

In Listing 1, the URI in the GeoNames project for Auburn is equated to the Auburn DBpedia resource from the English-language context. I then connect the Freebase identifier for Auburn to the DBpedia resource. Finally, I connect the Auburn identifier from the Japanese DBpedia language context to the English one. At this point, all four of these names are equated to one another. Triples specified with any of them as the subject are now true about all of them.

Listing 1. Connecting identifiers with OWL

```
# Connecting the DBpedia resource for Auburn, CA to three other
# resources using owl:sameAs
@prefix owl:   <http://www.w3.org/2002/07/owl#> .

<http://rdf.freebase.com/ns/m.8r2rz> owl:sameAs  <http://dbpedia.org/resource/Auburn,_California> .
```

It's worth keeping in mind that these data sets come from different organizations and aren't necessarily produced by the members of the LOD project. But they're expressed using standards, which makes all of the difference for making the data consumable by a wide variety of clients. Some of the data is stored natively as RDF in files, some is stored in triple stores, some is stored in relational databases and projected as RDF as needed. The use of Linked Data technologies doesn't usually burden the sources of the information. Those technologies are merely conduits to emancipate the information and connect it to related content with ease. The linkage between data sets can be lumped in with the rest of the content or kept separate in a link set.

Remember from the previous article that you can pull information together from multiple data sources via SPARQL simply by referencing them with the FROM keyword. You can now imagine leaving the source data unadulterated but storing the identifier linkage in a file, as in Listing 1, and referring to that link set in a SPARQL query, as in Listing 2. For the purposes of the query, the connections among the terms in each of the sources of data will be included in the graph and available for reasoner-based integration.
Listing 2. SPARQL query with data sets and link sets

```
SELECT variable-list
FROM dataset1
FROM dataset2
FROM linkset
WHERE {
  graph pattern
}
```

The LOD project's initial 12 data sets were connected in this way. And then more were added. And then more. The project added new classes of data sets involving academic research citation; life sciences; government-produced data; information about actors, directors, movies, restaurants; and more. By 2014, 570 data sets representing billions of RDF triples were connected. You can see a summary of the LOD cloud diagram as of 2014 in Figure 2. You'll have more fun exploring an interactive version in an SVG-enabled browser. If you click through most of the individual data sets, you'll be taken to their corresponding Datahub pages.

**LOD cloud diagrams**

The LOD diagrams are released under a CC-BY-SA license and are available from various historical stages of the cloud. Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch, and Richard Cyganiak.

**Figure 2. The LOD project cloud in 2014**
Many of these data sets are described using — what else? — an RDF vocabulary for describing interlinked data: the Vocabulary of Interlinked Datasets (VoID). Who produced them? When were they last modified? How big are they? Where can you find link sets to connect them to other data? The VoID description answers these questions.

Let’s dig deeper into one of these data sources: DBpedia. DBpedia was one of the first attempts to provide structured metadata from Wikipedia. A VoID description of DBpedia would include metadata such as in Listing 3.

Listing 3. Example VoID description of DBpedia

```xml
@prefix void: <http://rdfs.org/ns/void#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix wv: <http://vocab.org/waiver/terms/norms> .
@prefix sd: <http://www.w3.org/ns/sparql-service-description#> .
@prefix : <#> .

:DBpedia a void:Dataset;
  dcterms:title "DBPedia";
  dcterms:description "RDF data extracted from Wikipedia";
  dcterms:contributor :FU_Berlin;
  dcterms:contributor :University_Leipzig;
  dcterms:contributor :OpenLink_Software;
  dcterms:contributor :DBpedia_community;
  dcterms:source <http://dbpedia.org/resource/Wikipedia>;
  dcterms:modified "2008-11-17"^^xsd:date;
.

:DBpedia_Geonames a void:Linkset
  ...
  ...

:FU_Berlin a foaf:Organization;
  rdf:label "Freie Universität Berlin";
  foaf:homepage <http://www.fu-berlin.de/>;
.
.
.
```

From the description you discover that DBpedia is information extracted from Wikipedia. Although most content on Wikipedia is unstructured, the site includes a tremendous amount of editorially controlled structure. The Infoboxes in the articles, in particular, are consistent and readily yield their information in nicely structured ways. As a consequence, more than 12.6 million things are uniquely described using 2.5 billion RDF triples from 119 localized language contexts, including:

- 830,000 people
- 640,000 places
- 370,000 creative works
- 210,000 organizations
- 226,000 species
- 5,600 diseases
Each one of these topics is its own resource with its own resolvable identifier. As you ponder the magnitude and variety of topics described here, remember that this multidomain data set is maintained and curated by volunteers. It includes 25 million links to images, 28 million links to documents, and 45 million links to other RDF datasets. Nearly three-quarters of the resources are organized by categories from multiple ontologies.

Each one of these resources has a logical identifier, an HTML-rendered page, and a direct link to an RDF/XML serialization:

- [http://dbpedia.org/resource/Auburn,_California](http://dbpedia.org/resource/Auburn,_California)  # logical identifier
- [http://dbpedia.org/page/Auburn,_California](http://dbpedia.org/page/Auburn,_California)  # HTML-rendered page
- [http://dbpedia.org/data/Auburn,_California.rdf](http://dbpedia.org/data/Auburn,_California.rdf)  # direct RDF link

If you follow the link to the logical resource, you're redirected to the HTML-rendered view. This happens because when you click that link, the browser requests the response with HTML as its preferred source. The DBpedia server redirects you to the rendered form. From there, you can explore Auburn's connections to related resources such as its newspaper, the county it is in, and famous people who were born there.

These URIs are all resource references, and each resource is described using RDF extracted from Wikipedia. What you see when you click is an HTML rendering of the RDF data, not the web page for that resource. For example, the Auburn Journal has its own web page, which can be discovered by following the [http://dbpedia.org/ontology/wikiPageExternalLink](http://dbpedia.org/ontology/wikiPageExternalLink) relationship off of the resource for the newspaper.

I mentioned that most of the DBpedia resources are categorized from multiple ontologies. What that means specifically is that the resources are instances of classes that are also RDF resources. If you look closely at Auburn's resource page, you'll see that it is an `rdf:type` of several classes including:

- [http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing](http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing)
- [http://schema.org/City](http://schema.org/City)
- [http://dbpedia.org/page/Category:Cities_in_Placer_County,_California](http://dbpedia.org/page/Category:Cities_in_Placer_County,_California)
- [http://dbpedia.org/page/Category:County_seats_in_California](http://dbpedia.org/page/Category:County_seats_in_California)

Note that these are different classes from different schemes. It's easy to see how more categories can be added at any time by asserting new `rdf:type` instance relationships to whatever makes sense. This is a set-membership relationship, however. That means that it's possible to ask for anything that is a member of that set (or instance of that class). If you click through the [http://dbpedia.org/page/Category:Cities_in_Placer_County,_California](http://dbpedia.org/page/Category:Cities_in_Placer_County,_California) category, you'll see other cities in Placer County including Loomis, Rocklin, and Roseville. Here, you are seeing a set of related cities based upon the containment relationship of their being part of the same county.

The [http://dbpedia.org/page/Category:County_seats_in_California](http://dbpedia.org/page/Category:County_seats_in_California) class includes a much larger set. Here, the seats of California's counties are categorized together, and you can get from one, through the relationship, to the others you know about. The links you are navigating are effectively implicit SPARQL queries handled behind the scenes. An equivalent query is:
SELECT ?s WHERE {
  ?s a <http://dbpedia.org/class/yago/CountySeatsInCalifornia> .
}

Because DBpedia supports the SPARQL Protocol, which I introduced in the previous article, this query can be turned into a direct link. The expanded form is:

http://dbpedia.org/snorql/?query=SELECT+%3Fs+WHERE+%7B+%3Fs+a+%3Chttp%3A+\%2F%2Fdbpedia.org%2Fclass%2Fyago%2FCountySeatsInCalifornia%3E+%7D

Now I'll combine some of the things that I have shown you into a new query:

SELECT ?s ?page WHERE {
  ?s a <http://dbpedia.org/class/yago/CountySeatsInCalifornia> ;
}

I'm adding one extra relationship to the previous query. What's being asked is now: "Show me all of the county seats of California and external web pages associated with them." This is a powerful query to be able to throw together against data automatically extracted from Wikipedia. You can see the results here.

Now, change one simple thing in the query. Instead of querying for resources that are members of the http://dbpedia.org/class/yago/CountySeatsInCalifornia class, use http://dbpedia.org/class/yago/CapitalsInEurope:

SELECT ?s ?page WHERE {
  ?s a <http://dbpedia.org/class/yago/CapitalsInEurope> ;
}

The results are available here. Changing nothing but the class name causes the results to now reflect external web pages associated with the capital cities of countries in the continent of Europe!

If I change the relationship I'm looking for linked to the resources that are categorized this way, I can ask another completely different question. This query asks for latitude and longitude information instead of external links:

SELECT ?s ?lat ?long WHERE {
  ?s a <http://dbpedia.org/class/yago/CapitalsInEurope> ;
  <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat ;
  <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?long .
} ORDER BY ?s

The results are available here.

It should be no great leap to imagine retrieving the information from such a query and displaying it on Google Maps. The result of doing so is shown in Figure 3, and you can interact with the result here. Think of how much code would have to change to find and visualize where all of the heads of state of European countries were born. (Hint: pretty much none.)
Now that you have the mechanics in place, it's not difficult to imagine how to ask other kinds of questions about arbitrary domains. My favorite DBpedia query (which I got from Bob DuCharme) is to find every chalkboard gag from every episode of "The Simpsons." When you follow that link, keep in mind that every episode is also a resource that contains links to the episode's director, special guests, featured character, and so on. Each episode is categorized as being a member of a set of television programs from a particular year. By following the member links to those classes, you can find other television episodes that aired at roughly the same time.

At this point, the sky is the limit on the kinds of questions you can ask of DBpedia. And keep in mind that DBpedia is only one of nearly 600 data sets that are part of the LOD cloud. Linked Data produces impressive results with relatively little human effort.

**Conclusion**

Consider how long it takes your organization to integrate a single new data source. Linked Data is a fundamentally different approach to the problem that works at a level of productivity, scale, and flexibility that's difficult to imagine if all you've ever had at your disposal were enterprise and programming-language-related solutions. Nothing about this approach limits its applicability to public-facing data. You can easily apply the same ideas behind your firewall.

**Linked Data Fragments**

A group of individuals has proposed a new approach, called Linked Data Fragments, to solving the problem of reliably querying linked data. This isn't a standardized approach yet, but it has many great ideas that I encourage you to investigate.

Linked Data isn't magic. Standard identifiers that resolve into standard serializations of standard data models is a straightforward (although perhaps nonintuitive) set of concepts. Supporting the SPARQL Protocol openly on the web, however, is an incredibly difficult thing to do from an engineering perspective. It's difficult to predict what kinds of loads random individuals will put on your servers. A significant effort has gone into keeping DBpedia up and running. You can read more about the process on the website.

In the next article, I'll introduce you to a software platform based upon these ideas and finally begin to introduce you to the Open Services Lifecycle Collaboration (OSLC) technologies that chose to head down this path.
Related topics

- **Linked Data**: Read about the LOD project's nearly 600 data sets, where they come from, how they are produced, and how they are linked.
- **Linked Data Fragments**: Read about a less burdensome way of allowing clients to query information remotely.
- **DBpedia**: Investigate a data set extracted from Wikipedia.
- **Wikidata**: Wikipedia has its own source of data inspired by the successes of DBpedia and now offering its own SPARQL endpoint.
- **Linked Data** (David Wood et al., Manning Publications, 2014): Check out another book on building systems around Linked Data concepts.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)