3. Making Data Webby

This chapter covers

- Defining the principles of the Web
- How to plug your data into the Web
- Why relational databases aren't webby
- How to model data with triples
- How to infer meaning from data

On the World Wide Web's twentieth birthday, Sir Tim Berners-Lee, the Web's founder, wrote a piece for Scientific American, describing the state of the Web. He argued that "universality is the foundation" for the Web. That the Web was built on the fundamental principle that anyone should be able to say anything about any topic and that everyone should be able to access every Web page without buying into a proprietary system. We call these ideas free speech and free access. Free speech ensures ideas and information are defined from the bottom-up, by the people. Free access ensures ideas and information are allowed to spread quickly and easily among the people.

HTTP, URIs, and HTML – the standards that made the Web possible – have supported these ideals and afforded us the innovation, communication and world-wide transformations we all take for granted today. The Linked Data Web similarly relies on this philosophy of openness and interoperability and, as we saw in chapter one, the HTTP URIs that define the World Wide Web, are also at the heart of the Linked Data Web.

Recall that HTTP URIs give us three features necessary for such an open, interoperable Web: 1) they allow us to refer to unique data points without ambiguity, 2) they allow everyone to access these data points in an identical manner, 3) they allow us to link data points together so we can distribute our datasets across individuals, across organizations, and in fact, across the entire Web.

Webby data then, is data that has been liberated from proprietary or redundant technologies to disambiguate, to access, and to link their data points. We liberate our data so we can take advantage of the organic, distributed, ad-hoc nature of the Web. Over the following pages, we will see why modeling data for the Web means more than finding an efficient mechanism for storing and accessing information. We will discover that modeling data for the Web means documenting, organizing, and generating meaning from the World Wide Web we've all come to know and love.

WEBBY DATA IS DATA THAT HAS BEEN LIBERATED FROM PROPRIETARY OR REDUNDANT TECHNOLOGIES TO DISAMBIGUATE, TO ACCESS, AND TO LINK THEIR DATA POINTS.

3.1.On the Web Your Data Model Won't Last (Forever)

Every so often, we experience revolutions in thought that rock the foundations of those truths we hold to be self-evident. Even more often, we simply change our minds and decide that certain structures of thought, provenance of ideas, or relationships between entities no longer make sense. Whether exposing an Egyptian president as a dictator, redefining Pluto as a dwarf planet, or bringing Golf back to the Olympics after a century hiatus, our world seems to be in a constant flux of definitions and redefinitions. Unfortunately, data is often modeled in a static way, with predefined structure, provenance and relationships.

As a Web developer, you've no doubt discovered that the Web was clearly designed as an organic and mutable information space. Web pages and the links between them are easily changed and updated as new ideas surface. Therefore, a webby data model must be one that allows for the fluid fluctuations that permeate the Web. Over the following pages we will see Linked Data's approach to such a web-based data model.

3.1.1.Static Schemas Suck: How to plan for change

The Linked Data Web is built on the premise that change is good. Unfortunately, data managers have too often been afraid of change in their data and so have designed schemas to be static, unchanging creatures. Anyone who has been around during a merger or acquisition knows the difficulty in connecting such databases. The inevitable changes often happen in the code layer, and are usually solution-specific, so that another merger or acquisition means more code and more headaches. These problems are only exacerbated in the dynamic, organic Web. It may seem obvious that meshing data from multiple sources requires a dynamic schema, but we have found that it is too easy to resort to old habits, so here we explore a typical Linked Data mesh-up, and how to plan for change.

The Madison Metropolitan School District (MMSD) wants to provide a Web portal to help families better understand the MMSD schools, staff, and policies. The data for their portal will not only come from their own internal databases, but also from data provided by their local and state government, their individual schools, and more generic sources on the open Web. In order to gather all these data, they will need to do a good deal of data scraping and modeling. While we will leave the details of such work to later chapters, we can begin to understand the necessity and power of dynamic data models right now.

Modeling your data for the web:

1. *Lose the details*: Inventory your data sources and make a list of the general types of data you will use (location data, people data, event data, etc.)
2. *Query the market*: Search for other examples of such data on the Web and inventory the **properties**, **classes**, and **datatypes** they use.
3. *Plan your reuse*: Split your modeling plan into three categories: 1) those classes, properties, and datatypes that can be reused, 2) those that need to be extended from others, 3) those that need to be made up from scratch.

Step one (Figure 3.1.1.1) helps us abstract away the specifics of our data to better understand the essence of what we're modeling. For example, when we say a certain school resides at a certain location, we are using geodata information – the same type of data found in GPS devices and on Google Maps. It is often the case that about 80% of the data we model has already been modeled by others, so step one helps us abstract our data into a reusable form that will mesh well with other data models.

Figure 3.1.1.1: MMSD data sources and their associated types

Data	**Kind**
Schools and central office locations, etc.	Geodata
School officials, teachers, staff, etc.	People
School schedules, council meetings, etc.	Date/time
Parent/teacher groups, student extra-curricular activities	Organizations

We abstract our specific data into generic categories so that we can more easily find similar data models.

Step two (Figure 3.1.1.2) helps us find those structural components for our data that will allow it to stay dynamic over time. As the Web changes, so too can the meaning behind our data. If we trust the way GeoNames defines places (the fact that the US Geological Survey and the New York Times trust them is a good clue that we should too), we can simply describe places in the terms that they use; if we trust the way that Google defines events, we can reuse their definitions. We will soon hit that 80% mark of reused data terms and be thankful we didn't waste time reinventing the wheel.

Figure 3.1.1.2: Finding Similar Properties, Classes, and Datatypes

Kind of Data	Websites with similar data	Related Classes	Related Properties	Related Datatypes
Geodata	GeoNames	dbo:Park, dbo:Building, dbo:Lake	rich:address	xsd:String, xsd:Float
People	New York Times	foaf:Person	foaf:name, foaf:school	xsd:String, xsd:Integer
Events	Google	rich:Event	rich:startTime, rich:endTime	xsd:DateTime
Organizations	DBPedia, Freebase, FOAF files	foaf:Organization	foaf:homepage, fb: user.mdaconta.human_resources.employee.employedby	xsd:String, xsd:Integer

Here we create a list of the classes, properties, and datatypes that others have used when referring to similar data. See 3.2.3 for an explanation of the *prefix:term* syntax.

Step three (Figure 3.1.1.3) helps us define a webby data model without losing any unique definitions we might have. The beauty of the Linked Data Web is that it allows us to take advantage of other people's data models even while we add our own unique definitions. If we find a definition out there from an organization we trust, say, the W3C's definition of Person, we can simply reuse this concept without any changes. If instead we want to define a new concept like *Park With Playground*, we can still use the metadata associated with parks that we found via GeoNames (like the fact that they have an average annual temperature and a municipality code). Finally, if we have a concept like a School Board, which may not have similar data modeled on the Web, we can simply define our own class in our own namespace (explored in greater detail in chapter 5).

Figure 3.1.1.3: Reusing data without losing your voice

Data	Data Source	Class Name	Supertype
Geodata	GeoNames	ld4wdClass: ParkWithPlayground	dbo:Park
People	FOAF	foaf:Person	No Change
Events	Google	event:Hearing	event:Event
Organizations	N.A.	mmsd:SchoolBoard	New Concept

Now that we have found datatypes to copy, we can decide whether we can copy others' terms, whether we should build off them, or whether we need to define our own terms.

3.1.2.Data Provenance: Put on your reporter's cap

Data is often meaningless without context. Data provenance gives us the who, when, and how that any good reporter asks when confronted with the what and where of a news story. If you say that citizens are considered "British" (what) in Northern Ireland (where) you ought to clarify your remarks. Provenance tells you which group made that definition (who), the time period that the term has been considered appropriate (when) and the etymology of these decisions (how they came to be). By making such metadata explicit, we can be sure that our data are always used and reused within the proper context and under the proper circumstances.

Provenance metadata can be split into three categories:

1. *The Who*: Every bit of data can be traced to an assertion by a person or organization that something is true. It is, therefore, important that our webby data attribute their sources so that multiple versions of the truth can be presented and tracked.
2. *The When*: The validity of data changes over time – presidents have term limits, planets get redefined, Olympic sports go out of fashion – so it is important that all data we model are clearly marked with the dates under which they are valid.
3. *The How*: Even though we would like all data to have a single source and a single origin in time, we often find that data go through many transformations before they take on their present form. If you have ever played the game "telephone" you know that tracking provenance of messages entails tracking changes over time and over multiple sources. To truly understand the provenance of data, then, we need to know the timeline of the who's and when's: the explanation of how things came to be.

Fortunately, we will see in later chapters that Linked Data gives us provenance for free: the provenance of our data can be built explicitly into the Linked Data model itself, without relying on code to connect databases with meta-databases.

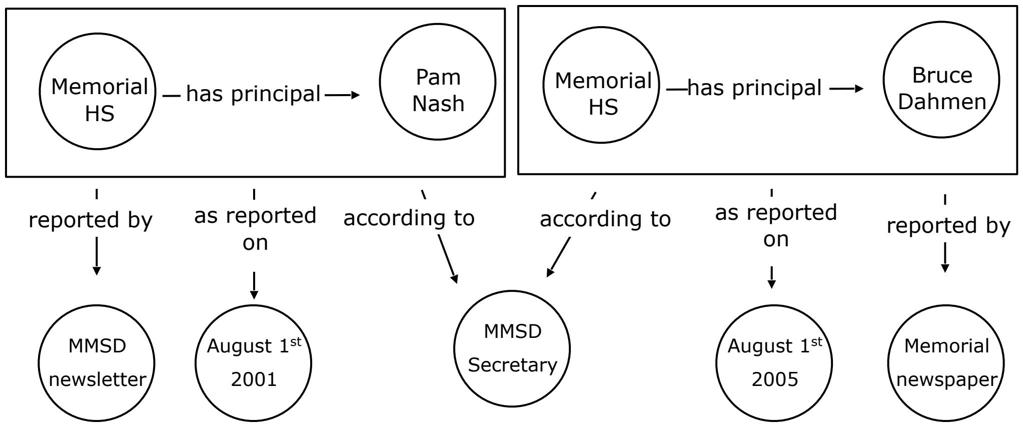
3.1.3.Allow anyone to say anything about any topic

On the Web, no one has a monopoly on truth. Because the Web was built on the fundamental principal that anyone can say anything about any topic (free speech), the Linked Data Web has a built-in respect for multiple versions of the truth. In fact, if we combine sections 3.1.1 and 3.1.2 we begin to see how this happens. In 3.1.1 we saw that webby data allows us to reuse other people's terms, extend their meaning, and, when necessary, make up our own definitions. In 3.1.2 we saw that the Linked Data Web requires us to track the provenance, or the "who, when and how" of our assertions. Combined, we can now say, for example, that "MMSD has a Hearing on startDate W, as assertedBy the president X on date Y in her StateOfTheDistrictAddress".

The power of reused terms with provenance is that we can all talk about the same things in different ways – or different things in the same way. And then we can sift through all these statements to model the complexity of our communities, businesses, and society. If there are reviews of the same product on multiple websites, Linked Data lets us mesh them together to provide a more holistic view for the user. Since every school and hospital has a street address, Linked Data lets us define these addresses identically so we could, for example, search for "elementary schools within 2 miles of a hospital". Finally, since we often want to redefine things over time (e.g. a school board member's list of responsibilities or a principal's name) without losing the old data, provenance allows us to qualify our statements.

For example, while it might sound contradictory to say both that "Memorial HS has principal Pam Nash" and that "Memorial HS has principal Bruce Dahmen", we can add provenance (qualifying data) to add a specific reference frame. So, instead we might say that "Memorial HS has principal Pam Nash *as reported on August 1st, 2001*" and "Memorial HS has principal Bruce Dahmen *as reported on August 1st, 2005*".

Figure 3.1.3.1: Qualifying statements with provenance



Linked Data allows us to define our own terms – such as *reported by*, *as reported on*, or *according to* – in order to answer the who, when, and how of data provenance.

Before moving on, we want to note that there is currently no standardized provenance vocabulary. While Linked Data allows you to define your own provenance terms (accordingTo, reportedBy, asProclaimedInHighEsteemByHisMajestyOnTheDate, etc.), they're most useful when they're reused. When computers reuse provenance terms, it builds a Web of Data where my *reportedBy* and your *reportedBy* mean the same thing. Section 3.2 will describe how triples allow such reuse, section 3.3 will show how meaning forms from this reuse, and Part II of this book will provide in-depth examples of reusing terms and data. See chapter 11 for a future outlook on research areas in provenance.

3.2.From Tabular and Relational Data to Triples

All data models are not created equal. Over the following pages, as we wade through tabular, relational, and triple-based data models, we will see why the abstractions afforded by each successive model provide us a more interoperable framework with which to distribute our data across the Web. We will explore the structure-first nature of tabular and relational databases, and contrast this with the triple-based model's organic structure, which allows you to take advantage of the ad-hoc nature of the Web. Finally, we will show why a triple-based data model is necessary for a Linked Data Web.

3.2.1.Tabular Data

Tabular, spreadsheet, or just plain table data is the most well-known data model on earth. Tables allow us to very clearly lay out a set of properties (the columns) and to list a set of entities (rows), which have values (cells) associated with these properties. Since data is solely defined by the cells in the table, we often refer to table data as property-value pairs. Such pairs are a powerful way of expressing data in a human-readable format. Take Figure 3.2.1.1, this table clearly displays data about the high schools in the Madison Metropolitan School District.

Figure 3.2.1.1: Madison Metropolitan School District High Schools

Name	**Location**	**Principal**
East High	2222 East Washington Ave.	Mary Kelley
James Madison Memorial High	201 S. Gammon Rd.	Bruce Dahmen
La Follette High	702 Pflaum Road.	Joe Gothard
West High	30 Ash St.	Ed Holmes

Tables with small amounts of information are useful and readable for most people. This table has three properties – Name, Location, and Principal – and four values associated with each property.

While data like that represented in Figure 3.2.1.1 is simple for humans, it can be confusing for computers:

1. Without a unique identifier for each row, we cannot guarantee that queries will be meaningful over time as values change.
2. Tables of any scale quickly become infeasible, because querying tabular data requires placing the entire table in the computer's memory before any calculations are made.
3. There is no way to recognize when values mentioned in multiple tables are referring to the same thing, because all values are given as plain text.

4. Properties (columns) are based on the table instead of the entities, so individual rows are unable to have custom properties.

Because of these problems and more, tabular data is usually reserved for small projects in which humans are the primary interface to the data.

3.2.2.Relational Data

You are probably already somewhat familiar with the relational database management system (RDBMS). Almost every Web site with a database relies on some version of the Structured Query Language, or SQL, whose sole function is to interface with such systems. RDBMSs coupled with SQL are powerful tools, which allow developers to expand on the property-value pairs of the tabular data model. In addition to providing a way of querying the data that you might find in a spreadsheet, the relational data model solves some of the problems we listed about tabular data earlier.

In practice, relational data often have a primary key associated with each row (or entity), which uniquely identifies entities without relying on data about an entity that may change over time. Relational databases can also be queried efficiently, without requiring memory for the entire database. Unfortunately, cross-table ambiguity and entity-specific properties (problems 3 and 4, above) live on in relational databases.

If the data from Figure 3.2.1.1 were in a relational database, we could query a piece of it with Listing 3.2.2.1.

Listing 3.2.2.1: SQL query, which could select "Bruce Dahmen" from the data in Figure 3.2.1.1

```
SELECT Principal
     FROM MMSD_HS
     WHERE PrimaryKey = "ABC123";
```

While the introduction of the primary key solves the problem of ambiguity (i.e., we can be sure that our query retrieves only data about the one and only Bruce Dahmen), this solution doesn't scale well. First off, in an RDBMS the uniqueness of our records are tied to the tables they inhabit. On the Web, where data comes from multiple websites, let alone multiple tables within a website, it is simply infeasible to rely on such a solution-specific uniqueness criteria. Second off, RDBMSs are not built to provide cross-Web access to their look-up tables. Because of this, there is no way to access data from a table without being given a solution-specific access point. While we could hack an RDBMS with URIs as our primary key – providing a globally unique name and access point – the RDB data model is not built for this so we would inevitably be writing solution-specific code that would be hard to update once new schemas were introduced into the system.

3.2.3.Triples and Quads

Unlike tabular and relational data, triples are built for change. Instead of boxy spreadsheets with predefined columns, the triple or Entity-Attribute-Value data model, allows each grouping of three terms to speak for itself. We call this the *Fun with Spot* data model. Saying

Spot walks Home follows the basic subject-predicate-object order of language. Such a statement can be easily modeled in a triple-based data model. And, as we will soon see, such data is designed to take advantage of the scalability of URIs.

If we instead wanted to model *Spot walks home* in a tabular or relational data model, we would have to predefine a structure of columns and rows, potentially adding an additional column to define a primary key for data about Spot.

Figure 3.2.3.1: Spot

Name	**Walks**	**Primary Key**
Spot	Home	ABC123

Creating an entire table for data about a single statement is overkill.

Instead, using triples, with the additional benefit of URIs, we can write:

<http://example.com/Spot> <http://example.com/walks> <http://example.com/Home>;

That's it! That's a complete and valid snippet of triple data. In fact, triple-based data are simply made of lots and lots of snippets just like this one.

Before we delve any further, we want to introduce you to one more feature of triple-based data: *qnames*. Qnames are simply abbreviations for URIs that give readers (and publishers) a break in deciphering (or printing) long and detailed identifiers. A qname is made up of two parts: a prefix and a term, with a colon in between. The prefix refers to some URI where data can be accessed, and the term refers to a class, property, or instance defined at that URI. For example, we could associate the prefix *ex* with *http://example.com/* and then refer to the triple above as *ex:Spot ex:walks ex:Home* (learn more about triple syntax in Chapter 4). The website http://prefix.cc can help you find commonly used prefix/URI mappings.

The triple-based data model not only allows us to quickly make statements about individuals, it also allows us to quickly add data to large datasets without any (!) modifications to overall structure.

Take, for example, this table of Spot and his friends' recent adventures:

Figure 3.2.3.2: Spot and friends represented in a table

Name	**Walks**	**Sleeps**	**Primary Key**
Spot	Home	In his bed	ABC123
Clifford	To the Circus	On the grass	ABC124
Shiloh	To the Woods	Under the stars	ABC125

When structure is predefined, every entity must share every property.

Notice that if we wanted to add that Clifford is a red dog, we would have to add a color column for every other dog. In contrast, a triple-based data structure allows us to start with a group of triples (Figure 3.7) and quickly add an attribute to a single entity (Figure 3.8).

Figure 3.2.3.3: Spot and friends represented in triples

```
@prefix ex: <http://example.com> .
ex:Spot ex:walks ex:Home .
ex:Clifford ex:walks "To the circus" .
ex:Shiloh ex:walks "To the Woods" .
```

Figure 3.2.3.4: Spot and friends represented in triples, with a red Clifford

```
@prefix ex: <http://example.com> .
ex:Spot ex:walks ex:Home .
ex:Clifford ex:walks "To the circus" .
ex:Shiloh ex:walks "To the Woods" .
ex:Clifford ex:hasFurColored ex:Red .
```

The benefits of the organic nature of triple-based data are multiplied when we venture into the Web. In the Web, references to popular people, places, and products not only span databases but also span domains, organizations, and even languages. Recognizing this reality, triples were designed to seamlessly link data across these divides. No longer are you forced to write code to describe relationships between data from multiple sources; triples build that logic right into the data itself (see section 3.3).

Say you're the US government and in your efforts to better understand your budget, you begin tracking how Americorps money is spent in the Boston area (Figure 3.2.3.5). You begin by looking at data held by the local Americorps-sponsored organization: the Massachusetts League of Community Health Centers (MLCHC). MLCHC gives you a list of those individuals who are salaried via tax dollars. Eager to understand what these members are doing to earn their stipends, you dig a bit deeper.

Figure 3.2.3.5: Distributed Americorps data using triples

Massachusetts League of Community Health Centers Database

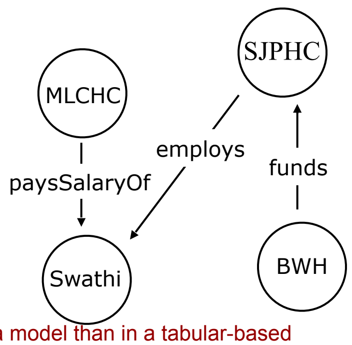
```
@prefix mlchc: <http://partners.org/mlchc/> .
mlchc:MLCHC mlchc:paysSalaryOf mlchc:Swathi .
```

Southern Jamaica Plain Health Center Database

```
@prefix mlchc: <http://partners.org/mlchc/> .
@prefix sjphc: <http://SJPHC.org/> .
sjphc:SJPHC sjphc:employs mlchc:Swathi .
```

Brigham Women's Hospital Database

```
@prefix sjphc: <http://SJPHC.org/> .
@prefix bwh: <http://BWH.org/> .
bwh:BWH bwh:funds sjphc:SJPHC .
```



Note that you can "follow the money" much easier in a triple-based data model than in a tabular-based one, since a graph provides a natural connect-the-dots structure.

First, you gather a list of Americorps-approved health centers in the Boston area from MLCHC. From this list, you venture to individual health centers and analyze their employee lists, where you find that Americorps member Swathi Damodaran is employed by the Southern Jamaica Plain Health Center (SJPHC) but salaried by MLCHC. You wonder if SJPHC has any sources of money outside the government and you find that, indeed, one of their funders is Brigham-Women's Hospital. Perhaps you know that Brigham-Women's Hospital performs abortions, and as a conservative senator you decide that you no longer want to put tax dollars toward clinics that are funded by such liberal institutions. Or perhaps you're a progressive senator who cares deeply about comprehensive sexual education, and upon learning that SJPHC is a leading voice, you now know that any money given to Brigham-Women's Hospital supports this effort.

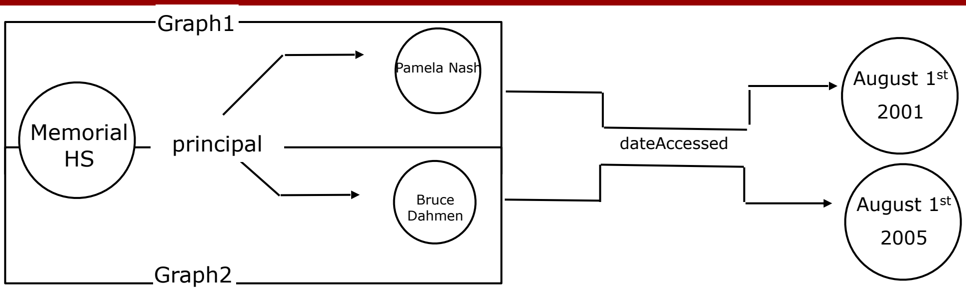
Such searches, that span multiple organizations and multiple databases, are the bread and butter of Linked Data. As you can see in Figure 3.2.3.5, triple-based data is built with simple and unambiguous connections between databases. Check out chapter 6 to discover how to query such distributed data.

Quads

Triples are great and useful, but if you were paying attention back in 3.1 you may be asking: "but how can we add provenance to our data if all our data is stuck in these individual subject-predicate-object snippets?" We're glad you asked! The most common way to add provenance to your data nowadays is to organize your triples in separate **named graphs** – groups of interlinked triples that are given the same who, when, and how provenance metadata. This is done via **quads**: triples that have an added URI at the end identifying their named graph.

For example, if you were scraping data about the administrative staff at Memorial High School, you would first find the relevant data on the school's website and map it into triples. Once you had all the triples associated with the administrative staff, you could place them all into a named graph by adding a fourth URI to each triple (Figure 3.2.3.6). The URI of the named graph could then be the subject of any provenance information you have about the triples in the graph.

Figure 3.2.3.6: Named Graphs with quad syntax allow you to add provenance to triples



```
@prefix memorial: <http://www.madison.k12.wi.us/145#> .
@prefix dbpedia: <http://dbpedia.org/property/> .
@prefix ld4wdGraph: <http://ld4wd.org/graphs/> .
@prefix ld4wdProp: <http://ld4wd.org/properties/> .

memorial:identifier dbpedia:principal "Pamela Nash" ld4wdGraph:Graph1 .
ld4wdGraph:Graph1 ld4wdProp:dateAccessed "August 1st, 2001" .
memorial:identifier dbpedia:principal "Bruce Dahmen" ld4wdGraph:Graph2 .
ld4wdGraph:Graph2 ld4wdProp:dateAccessed "August 1st, 2005" .
```

In this example, there are two quads and two triples. The two statements about the principals are placed inside named graphs, using the quad syntax, so that they can be given provenance information.

3.3.Expressing Meaning via Triples

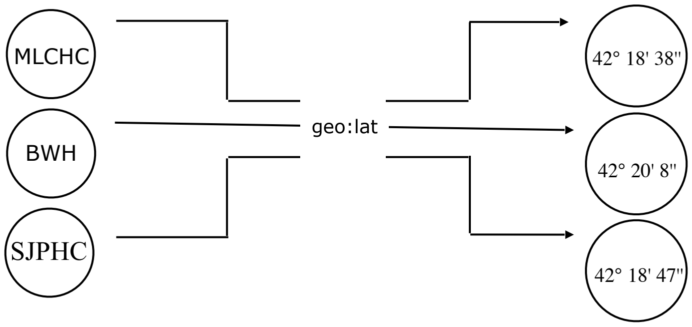
So far in this chapter we have gained an appreciation for the necessity of a messy Web and seen why a triple-based data structure helps to model it. Fortunately, triples were not only designed to model the Web's data, they were also designed to give meaning to this data. Meaning is important when you want to integrate or visualize data from multiple sources. In order to combine data from two or more sources, you have to understand the meaning of the entities you want to map to each other. Likewise, when you want to visualize data you often want to know the meaning of the data you are visualizing (location data for maps, weather data for forecasts, etc.) Over the next few pages, we will lay out how Linked Data and its webby, distributed nature affords meaning in your data. See chapter five for a more in-depth discussion of knowledge-creation via vocabularies.

3.3.1.It's the links, stupid

In 3.2 we saw the power of linking data together across databases using URIs. In fact, this is the basis for the use of the word "Linked" in Linked Data. Since every data point in the Web of Data is accessed and identified with an HTTP URI, you can follow the links from triple to triple – no matter who is hosting them – as easily as if they were all in the same table of an RDB. When the SJP health center (to take our earlier example) says that they employ Swathi (*sjp:SJP sjp:employs mlchc:Swathi*) the object of this triple (*mlchc:Swathi*) references a URI from a different database. Using the same technology we used to find the original triple (see SPARQL in chapter 6) we could also find *mlchc:MLCHC mlchc:paysSalaryOf mlchc:Swathi* (that Mass League pays Swathi's salary) from Mass League's database. Simply put, the power of Linked Data lies in its built-in support for referencing and accessing data *from* anywhere on the Web, *to* anywhere on the Web.

Just as we can share entities across databases we can also share properties. In fact, sharing properties is one of the most powerful uses of Linked Data, as it enables reuse of data in wide contexts. One of the most common cases of this is geodata. As we saw in section 3.1.1, GeoNames hosts geodata terms that we can reuse in our own data. By browsing data at GeoNames.org, we would find not only properties defined by GeoNames, but also properties that they have reused from other sources. For example, when GeoNames refers to latitude and longitude they use terms defined by the World Wide Web Consortium (W3C). When we use the same URIs for latitude and longitude as the W3C, we help our data mesh well with other geodata around the Web.

Figure 3.3.1.1: Linked Data allows shared properties across different datasets

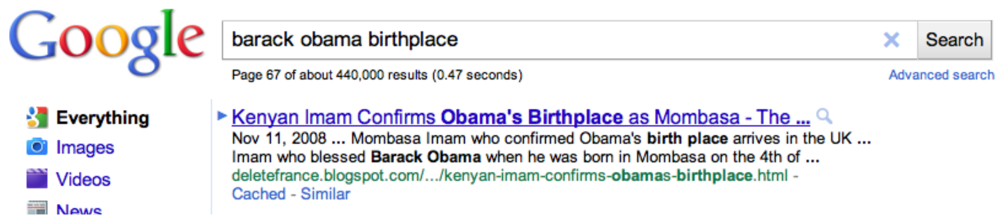


In this example, three separate organizations (MLCHC, BWH, and SJPHC) are able to use the *:lat* property defined at the URI *http://www.w3.org/2003/01/geo/wgs84_pos#* (represented with the prefix *geo*).

So what does this all have to do with adding meaning to our data? Well, on the Web just like in society, meaning comes from shared understandings. After all, if everyone spoke their own language, day-to-day life would be much more difficult. Linked Data, then, allows us to reuse terms directly in the data layer without writing code to translate or map data from multiple sources. In effect, Linked Data allows us to speak the same language as those around us – when it makes sense to do so (more on the subtleties of reuse in section 3.3.2).

The use of links to infer meaning is not a new subject on the Web. Take Google's famous PageRank algorithm (Figure 3.3.1.2). The reason that deletefrance.blogspot.com ranks low or that the New York Times ranks high in Google's algorithm is not because Google goes page-by-page defining the reputability of news sources. Instead, they rely on the Web of links that society has written: the number of hyperlinks, how long these links have been in effect, and the range of topics using these links.

Figure 3.3.1.2: Google uses links to determine importance of webpages



Page 67 of about 440,000 results (0.47 seconds)

Kenyan Imam Confirms **Obama's Birthplace** as Mombasa - The ...
Nov 11, 2008 ... Mombasa Imam who confirmed Obama's **birth place** arrives in the UK ... Imam who blessed **Barack Obama** when he was born in Mombasa on the 4th of ...
deletefrance.blogspot.com/.../kenyan-imam-confirms-**obamas**-**birthplace**.html - Cached - Similar

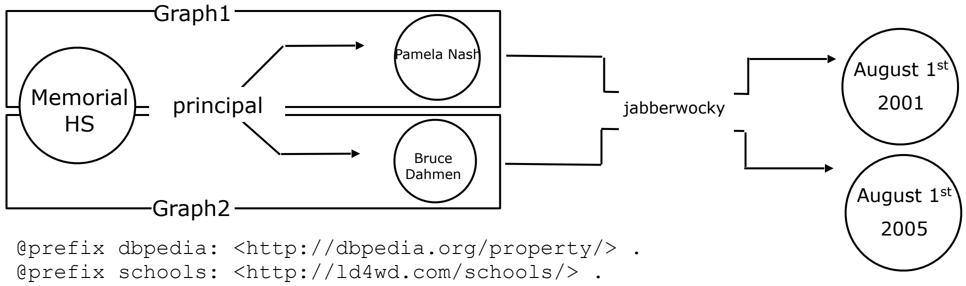
Note that this blatantly false statement was pushed to the 67th page of results in Google results for "Barack Obama birthplace", most likely because there are relatively few Web links pointing to the website or webpage.

The Web of Data expands on the power of the link by giving developers even more tools to determine the relationships or links between different data points or data sources. In section 3.1 we mentioned that provenance adds metadata allowing us to assert multiple versions of the truth depending on the who, when, and how of our data's origins. Then in 3.2 we saw how we can add such provenance to our triples using named graphs. We can now see how to infer meaning from such provenance through the reuse of URIs such as *ld4wdProp:dateAccessed*.

If you flip back to Figure 3.2.3.6, you will notice that we are stating two contradictory "truths": Memorial HS has a principal named Pam Nash and Memorial HS has a principal named Bruce Dahmen. We noted that we could do that because provenance allows us to append the time associated with each principal's tenure (one was principal in 2001 and the other in 2005). You may have wondered upon seeing us add the property *ld4wdProp:dateAccessed* how the computer knows that "August 1st, 2001" and "August 1st 2005" are dates, or even, given that they are dates what these dates have to do with the principals of Memorial. We want to reveal a secret: given the triples from 3.2.3.6, we might as well have called our property *ld4wd:jabberwocky*! After all, computers are very literal creatures; unless we define a vocabulary (coming in chapter 5), there is very little meaning they can infer from a URI. Little, but not none.

To understand the inferences that can be assumed from a simple URI, we have to remember the URI's virtues: 1) they are universally unambiguous, and 2) they can all be identically accessed (with HTTP). We are concerned with the first virtue here. Because a single URI can be assumed to be universally unambiguous, we know that it must mean the same thing in all contexts.

Figure 3.3.1.3: Shared URIs implies shared meaning



```
@prefix dbpedia: <http://dbpedia.org/property/> .
@prefix schools: <http://ld4wd.com/schools/> .
@prefix ld4wdGraph: <http://ld4wd.com/properties/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

schools:memorial dbpedia:principal "Pamela Nash" ld4wdGraph:Graph1 .
ld4wdGraph:Graph1 ld4wdProp:jabberwocky "August 1st, 2001" .
schools:memorial dbpedia:principal "Bruce Dahmen" ld4wdGraph:Graph2 .
ld4wdGraph:Graph2 ld4wdProp:jabberwocky "August 1st, 2005" .
```

In this example, it is not the name "jabberwocky" that defines the provenance of the two named graphs, but the fact that both graphs reuse this term. Because of this reuse, it is easy to query what separates two otherwise seemingly contradictory statements. We usually give properties more semantically meaningful names simply for the courtesy of us human users of the data.

As you can see in Figure 3.3.1.3, reusing properties gives us the freedom to infer that two assertions about principals can be differentiated if their provenance properties are shared. While there is a very powerful mechanism for defining meaning in Linked Data (see vocabularies in chapter 5), the simple act of linking to someone else's definition is often enough to inject meaning into our data. Even if no one has taken the time to write lengthy metadata, the simple act of reuse allows definitions to be implicitly learned from the data's use-cases. For example, if you use our property *ld4wdProp:jabberwocky*, and we have placed the values associated with *ld4wdProp:jabberwocky* on a calendar, you can be pretty sure that *ld4wdProp:jabberwocky* has to do with dates.

3.3.2. Finding and expressing equivalence

We have seen that links are powerful and that reusing them helps infer meaning, but sometimes reuse is simply not feasible. Maybe you're in an organization that wants to define all the terms in its own namespace. Or maybe when you begin your project you can't find anyone else with similar data, even though a month or year down the line there are many people referencing seemingly identical entities and properties. Fortunately, Linked Data supports more ad-hoc reuse.

There are a number of ways to assert the similarity of your data with other data on the web, we list five of the most common here:

1) owl:sameAs (Figure 3.3.2.1)
2) owl:equivalentProperty (Figure 3.3.2.2)
3) owl:equivalentClass (Figure 3.3.2.3)
4) rdfs:seeAlso (Figure 3.3.2.4)
5) foaf:homepage (Figure 3.3.2.5)

Figure 3.3.2.1: owl:sameAs asserts the equivalence of entities

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix schools: <http://ld4wd.com/schools/> .
@prefix ld4wdProp: <http://ld4wd.com/properties/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

schools:memorial ld4wdProp:principal "Bruce Dahmen" .
schools:memorial owl:sameAs dbpedia:James_Madison_Memorial_High_School .
```

Here, we are able to define our own *schools:memorial* entity as a reference to Memorial High School, while asserting that it is equivalent to the entity defined by dbpedia at dbpedia:James_Madison_Memorial_High_School. This makes integrating data from multiple sources about schools almost as trivial as if the exact URIs were reused.

Figure 3.3.2.2: owl:equivalentProperty asserts the equivalence of properties

```
@prefix dbpedia: <http://dbpedia.org/property/> .
@prefix ld4wdProp: <http://ld4wd.com/properties/> .
@prefix schools: <http://ld4wd.com/schools/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

memorial:identifier ld4wdProp:principal "Bruce Dahmen" .
ld4wdProp:principal owl:equivalentProperty dbpedia:principal .
```

Here, we are able to use our own *ld4wdProp:principal* property, while asserting that it is equivalent to the property that dbpedia uses to define school principals. This makes integrating data from multiple sources about principals almost as trivial as if the exact URIs were reused.

Figure 3.3.2.3: owl:equivalentClass asserts the equivalence of classes

```
@prefix dbpedia: <http://dbpedia.org/class/yago/> .
@prefix schools: <http://ld4wd.com/schools/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

schools:memorial rdf:type types:School .
types:School owl:equivalentClass dbpedia:School108276720 .
```

Here, we are able to use our own *types:School* entity type (known as a class), while asserting that it is equivalent to the type that dbpedia uses to refer to schools. This makes integrating data from multiple sources about schools almost as trivial as if the exact URIs were reused.

Figure 3.3.2.4: rdfs:seeAlso links related entities, but not necessarily identical ones

```
@prefix dbpedia: <http://dbpedia.org/class/yago/> .
@prefix schools: <http://ld4wd.com/schools/> .
@prefix schoolDistrict: <http://ld4wd.com/schoolDistrict/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

schools:memorial rdf:type dbpedia:School108276720 .
schools:memorial rdfs:seeAlso schoolDistrict:MMSD .
```

Here, we offer the Madison Metropolitan School District as a related entity to Memorial HS, which is a school in this district. The use of *rdfs:seeAlso* is fairly open to interpretation and is a nice way to connect two related entities without *owl:sameAs*, which implies that they are identical.

Figure 3.3.2.5: foaf:homepage refers to the human-readable webpage associated with the given entity

```
@prefix dbpedia: <http://dbpedia.org/class/yago/> .
@prefix schools: <http://ld4wd.com/schools/> .
@prefix types: <http://ld4wd.com/types/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://www.w3.org/2002/07/owl#> .

schools:memorial rdf:type dbpedia:School108276720 .
schools:memorial foaf:homepage <http://memorialweb.madison.k12.wi.us/> .
```

Here, we reference the homepage of Memorial HS. Because there is often confusion between a URI representing an organization, and a URI of a homepage about that organization, a good rule of thumb is: if you access the URI with an RDF browser (see chapter 7) will the server return RDF data? If no, then it is likely the *foaf:homepage* of some entity; if yes, then the URI likely represents an entity and can be the *owl:sameAs* of some other entity.

3.4.Summary

In this chapter we explored the dynamic, organic nature of the Web: how such a mercurial Web can be managed using provenance; how data can be modeled for the Web with graph-based "triples"; and how meaning can arise through data reuse and typed links. At the heart of this discussion has been the HTTP-based URI and its ability to liberate and link together entities and properties from the far reaches of the Web. Through such links we have built a model for Webby Data – data that is defined, exposed and structured exclusively using native Web technologies.