THREE

THE TECHNOLOGY

Today when we say "technology" it is often shorthand for "computer technology." The Technology section of a newspaper reports on Silicon Valley news and reviews the latest consumer gadgets that are powered by bits and bytes. Of course this is not the only technology in our lives, but it is the one that defines our modern age. A century and a half ago, the defining technology was electricity and all things electric. The light bulb was literally the bright idea of the day. Today we have LED light bulbs that we can control with a smartphone app, turning on the lights when we are still on our way home, or creating a romantic atmosphere by changing the color and intensity of the light at the touch of a screen.

If we move back in time we see ages defined by their technological innovations: steam power, water power, or the precision use of metals that made it possible to create accurate timepieces and to automate the production of fine cloth. We can go back to the printing press, clearly a defining technology for all that came after it. Printing technology depended both on innovations with metals and also on

the development of paper-making techniques that greatly improved on previous writing surfaces, like sheepskin, papyrus, wax, clay, and stone.

Basically, it's technology all of the way back—back to fire and the first stone axes. We naturally take for granted the technologies that precede our own age, and we marvel at the ones that are new.

Libraries of course have been technology-based from the beginning of their history. The earliest libraries that we know of were furnished with writings in the form of scrolls. Medieval libraries held bound manuscripts. The big leap forward was the Gutenberg revolution and the concomitant increase in the production of copies of texts. The number of books not only increased but they also become more affordable as a result of their abundance. Other technologies also had effects on libraries, such as the aforementioned development of electric lighting, which reduced the threat of fire and allowed readers to make use of the library outside of daylight hours.

In the eighteenth and nineteenth centuries, not only were more copies of books produced than ever before, but the numbers of new writings and new editions also grew. Library holdings thus increased as well, which led to difficulties in keeping up with an inventory of the items held by the library. Today we assume that every library has a catalog, but even in the 1800s some libraries had no actual record of their holdings or relied on a brief author list. Much "finding" done in libraries at the time relied on the memory of the librarian. Charles Ammi Cutter, writing about the catalog of the Harvard College Library in 1869, took pity on the librarian overseeing a collection of 20,000 books without a proper catalog, who had to attempt to answer subject-based queries using only his own knowledge of the content of the collection.

The library catalog technology of Cutter's day was a printed book. Printed book catalogs had the same advantages as books themselves: they could be produced in multiple copies and were highly portable. A library could give a copy of its catalog to another library, thus making it possible for users to discover, at a distance, that a library had the item sought. The disadvantages of the printed book catalog, however, became more serious as library collections grew and the rate of growth increased. A library catalog needed near-constant updating. Yet the time required to produce a printed book catalog in an era in which printing required that each page be typeset meant that the printed catalog could be seriously out of date as it came off the printing press. Updating such a catalog meant reprinting it in its entirely, or staving off an expensive new edition by producing supplementary volumes of newly acquired works, which then made searching quite tedious.

In the mid-1800s the library card catalog was already winning hearts and minds. Cutter attributed the development of the card catalog to Ezra Abbot, head of the Harvard College Library, in 1861 (Cutter 1869). Although neither the book catalog nor the card catalog meets all needs as efficiently as one would desire, the card catalog had already proven itself as an up-to-date instrument for library users and librarians alike. German professor Markus Krajewsky, in his book on the history of card files, *Paper Machines* (2011), shows that cards on paper slips had been used in earlier times, in particular by the early bibliographers and encyclopedists who needed to create an ordered presentation of a large number of individual entries. It was libraries, however, that demonstrated how useful and flexible the card catalog could be.

Cards were lauded by Melvil Dewey in his introduction to early editions of his Decimal Classification, although his classification and "relativ index" in no way required the use of a card system. However, the "Co-Operation Committee" of the newly formed American Library Association announced its decision on the standardization of the catalog card in *Library Journal* in 1877; not coincidentally, Dewey's library service company, The Library Bureau, founded in 1876, was poised to provide the cards to libraries at a cost lower than custom-produced card stock. The Library Bureau soon branched out into the provision of catalog furniture and a variety of card-based products for a growing business records market. In fact, before long providing cards to libraries was only a small portion of The Library Bureau's revenue as businesses and other enterprises in the United States and Europe turned to card systems for record-keeping. Krajewski considers these card systems the early precursors of the computerized database because of the way that they atomized data into manipulatable units, and also allowed the reordering of the data for different purposes.

It should be obvious that both the book catalog and the card catalog were themselves technologies, each with different affordances. They also were affected by related technological developments, such as changes in printing technologies. The typewriter brought greater uniformity to the card catalog than even the neatest "library hand" could, and undoubtedly increased the amount of information that one could squeeze into the approximate 3" x 5" surface. When the Library of Congress developed printed card sets using the ALA standard size and offered them for sale starting in 1902, the use of the card catalog in US libraries was solidified.

After Melvil Dewey, the person who had the greatest effect on library technology was Henriette Avram, creator of the Machine Readable Cataloging (MARC) format. This was not only an innovation in terms of library technology, it was generally innovative in terms of the computing capability of the time. In the mid-1960s, when MARC was under development, computer capabilities for handling textual data were very crude. To get an idea of what I mean, look at the mailing label on any of your magazines. You will see upper-case characters only, limited field sizes, and often a lack of punctuation beyond perhaps a hash mark for apartment numbers. This is what all data looked like in 1965. However, libraries needed to represent actual document titles and author names, and languages other than English. This meant that the library data record needed to have variable length fields, full punctuation, and diacritical marks. Avram delivered a standard that was definitely ahead of its time.

Although the primary focus of the standard was to automate the printing of cards for the Library of Congress's card service, Avram worked with staff at Library of Congress and other libraries involved in the project to leverage the MARC record for other uses, such as the local printing of "new books" lists. To make these possible the standard included non-text fields (in MARC known as "fixed fields") that could be easily used by simple sort routines. The idea that the catalog could be created as a computerized, online access system from such records was still a decade away, but Librarian of Congress L. Quincy Mumford announced in his foreword to Avram's 1968 document *The MARC Pilot Project* that MARC records would be distributed beginning in that year, and that this "should facilitate the development of automation throughout the entire library community." And it did.

Melvil Dewey did not anticipate the availability of the Library of Congress printed card service when he proposed the standardization of the library catalog card, yet it was precisely that standardization that made it possible for libraries across America to add LC printed cards to their catalogs. Likewise, Henriette Avram did not anticipate the creation of the computerized online catalog during her early work on the MARC format, but it was the existence of years of library cataloging in a machine-readable form that made the OPAC a possibility.

The next development in library catalog technology was the creation of that computerized catalog. It would be great to be able to say that the move from the card catalog to the online catalog was done mainly with the library user's needs

in mind. That wasn't my experience working on the University of California's online catalog in the early 1980s. The primary motivators for that catalog were the need to share information about library holdings across the entire state university system (and the associated cost savings), and to move away from the expense and inefficiency of card production and the maintenance of very large card catalogs. At the time that the library developed the first union catalog, which was generated from less than a half dozen years of MARC records created on the systems provided by the Ohio College Library Center (later known solely as OCLC) and the Research Libraries' Group's RLIN system, the larger libraries in the University of California systems were running from 100,000 to 150,000 cards behind filing into their massive card catalogs. This meant that cards entered the catalog about three months after the book was cataloged and shelved. For a major research library, having a catalog that was three months out of date, and only promising to get worse as library staffing decreased due to budget cuts, made the online catalog solution a necessity.

We, and by "we" I mean all of us in library technology during this time, created those first systems using the data we had, not the data we would have liked to have. The MARC records that we worked with were in essence the by-product of card production. And now, some thirty-five years later, we are still using much the same data even though information technology has changed greatly during that time, potentially affording us many opportunities for innovation. Quite possibly the greatest mistake made in the last two to three decades was failing to create a new data standard that would be more suited to modern technology and less an imitation of the library card in machine-readable form. The MARC record, designed as a format to carry bibliographic data to the printer, was hardly suited to database storage and manipulation. That doesn't mean that databases couldn't be created, and to be sure all online catalogs have made use of database technology of some type to provide search and display capabilities, but it is far from ideal from an information technology standpoint.

The real problem is the mismatch of the models between the carefully groomed text of the catalog entry and the inherent functionality of the database management system. The catalog data was designed to be encountered in an alphabetical sequence of full headings, read as strings from left to right; strings such as "Tolkien, J. R. R. (John Ronald Reuel), 1892–1973" or "Tonkin, Gulf of, Region—Commerce—History—Congresses." Following the catalog model of which Charles Cutter was a primary proponent, the headings for authors, titles, and subjects are designed to be filed together in alphabetical order in a "dictionary catalog."

Database management systems, which are essential to permit efficient searching of large amounts of data, work on an entirely different principle from the sequential file. A database management system is able to perform what is called "random access," which is the ability to go seemingly directly to the entry or entries that match the query. (The actual internal mechanism of this access is quite operationally complex.) These entries are then "retrieved," which means that they are pulled from the database as a set. A set of retrieved entries may be from radically different areas of the alphabetical sequence, and once retrieved are no longer in the context intended by the alphabetical catalog.

Database management systems include the ability to treat each word in a sentence or string as a separate searchable unit. This has been accepted as a positive development by searchers, and is now such a common feature of searching that today most do not realize that it was a novelty to their elders. No longer does a search have to begin at the same left-anchored entry determined by the library cataloging rules; no longer does the user need to know to search "Tonkin, Gulf of . . ." and not "Gulf of Tonkin." Oddly enough, in spite of the overwhelming use of keyword searching in library catalogs, which has been shown to be preferred by users even when a left-anchored string search was also available, library cataloging has continued its focus on headings designed for discovery via an alphabetical sequence. The entire basis of the discovery mechanism addressed by the cataloging rules has been rendered moot in the design of online catalogs, and the basic functioning of the online catalog does not implement the intended model of the card catalog. Parallel to the oft-voiced complaint that systems developers simply did not understand the intention of the catalog, the misunderstanding actually goes both ways: significant difference in retrieval methods, that is, sequential discovery on headings versus set retrieval on keywords, did not lead to any adaptation of cataloging output to facilitate the goals of the catalog in the new computerized environment. Library systems remain at this impasse, some three-and-a-half decades into the history of the online catalog. The reasons for this are complex and have both social and economic components.

It is not easy to explain why change was not made at this point in our technology history, but at least one of the factors was the failure to understand that cataloging is a response to technical possibilities. Whether the catalog is a book, a card file, or an online system, it can only be implemented as an available technology. Unlike most other communities, the library community continues to develop some key data standards that it claims are "technology neutral." It is, however, obvious that any data created today will be processed by computers, will be managed by database software, will be searched using database search

capabilities, and will be accessed by users over a computer network. One ignores this technology at great peril.

THE PRESENT AND FUTURE

We have made the error in the past of moving to new technologies without examining the fit between our data and the new technology. A perfect example of this is the development of an XML version of the MARC record. There are indeed similarities between MARC and XML, primarily that both can be used to mark up or encode machine-readable documents. Both can also encode structured data, although the MARC use of fixed fields is less flexible than XML, which allows variable-length data throughout. MARCXML was developed as a pure serialization of the MARC format. "Serialization" means that the data encoding of MARC was translated directly to XML without any related transformation of the data itself. Although this produced a record that could be managed with XML-aware software, it did nothing to improve the kind of data that could be conveyed in library bibliographic records. It also did nothing to address some of the limitations of the MARC record. The MARCXML standard is kept one-to-one with the original MARC record, with the single exception that field and record sizes are not enforced. (MARC fields are limited to a four-character length, thus to 9,999 bytes; the record itself cannot exceed 99,999 bytes.) But the limitation on the number of subfields to a field remains, even though there are fields that have no open subfields available for expansion. Other inconveniences also remain, such as the non-repeatability of the MARC fixed field information, which then forces some repeatable elements like languages and dates to be coded in more than one field to accommodate repeatability. MARCXML was never allowed to develop as its own technology, and therefore did not present a change. Library data in XML, rather than in MARCXML, could have represented a real change in capabilities. It might also have provided a better transition to new technology than we now have, because we could have resolved some of the more awkward elements of MARC over a decade or more, with a gradual update to the library systems that use this data. Today we either have to carry those practices on to our future data, or we need to make a great leap forward and break with our past.

We missed the XML boat, but now some are hoping to get on board the latest ship sailing by: the Semantic Web and its base technology, the Resource Description Framework (RDF). It should be noted that there is one other data technology development that could have been considered between XML and

RDF, that of object-oriented design (OO). By the early 1990s, when the FRBR Study Group was being formed, relational technology was no longer new and object-oriented technology was taking its place in many implementations. Programming languages like Java and Python are object-oriented, and data and databases can also be "OO." Library data is leap-frogging over this technology, or it will if it adopts RDF for its data, as it appears it might.

Unlike most of the data models that preceded it, from entity-relation to object-oriented, RDF does not arise from the world of business that prompted our previous technology upgrades. The Semantic Web, as the name implies, comes out of web technology. This is a significant difference from, for example, database technologies, because the web is an open platform and is the place where we put publicly accessible data, whereas databases are private and closed, housed within enterprises and often highly controlled in terms of access. This means that many of the design assumptions that drive the Semantic Web standards are quite different from those encountered in business data processing.

First, let's look at where the Semantic Web comes from and what is meant by "semantic." The Semantic Web comes out of a combination of web technology, with linking and identifying as primary requisites, and the artificial intelligence (AI) community, with smart "bots" as its goal. Where most of us read the term "semantic" as meaning "meaning," in the AI world "semantic" refers to a computable axiom, such as:

If A = B, and B = C, then A = C

Obviously, machine intelligence and human intelligence are significantly different. AI attempts to model human thinking by defining the world as information about things and rules that can be used to "understand" those things. As we know from the overly confident promises that have come out of the AI community since the dawn of computing, the world and how we humans understand it is more complex than it seems. Human intelligence is still a marvel that is unchallenged by machines, in spite of gains in such algorithm-rich areas like the game of chess.

Artificial intelligence on the World Wide Web is a more tractable problem than creating a robot that can navigate stairs, recognize human faces, and pass a Turing test, because the web is already a data abstraction with some distance from the sense-experienced world, and therefore more amenable to computation. The Semantic Web was introduced in an article in *Scientific American* in 2001 by Tim Berners-Lee, founder of the World Wide Web and director of the World Wide Web Consortium, and his associates James Hendler and Ora Lassila. The article told the story of a helpful bot that could find an available doctor, check

your calendar, and make an appointment that fit into your schedule. Creating such technology over the web would require much less effort than creating this technology as a stand-alone system; the web already had solved the problem of a large distributed system capable of handling heterogeneous data and billions of users. The trick was to include in the web the kind of coding that would allow data to be used alongside the current web of documents and media files.

The technology to achieve this is all based on the Resource Description Format (RDF), which itself is a deceptively simple model of things and relationships that can be used to express very complex data. There are some particular aspects of RDF that are both essential and notably different from the technology that most of us have worked with during our careers. There isn't space here to fully elucidate the technology that is RDF, but some points are key to the analysis in the second part of this book. Let's begin with identifiers.

Everything being described in RDF must have a standard identifier that begins with "http://" followed by a domain name (e.g., "ala.org/") and a precise path (conference2015). That might seem confusing, because that is the same prefix that is used for a uniform resource locator (URL), which is the address of something on the web. RDF is using the same standard for its identifiers for a couple of reasons: first, the mechanism to create and manage domain names on the web already exists, which means that it will be easy to create these identifiers; second, the combination of identification with location means that information about the thing identified can be stored on the web at that location without any change in technology.

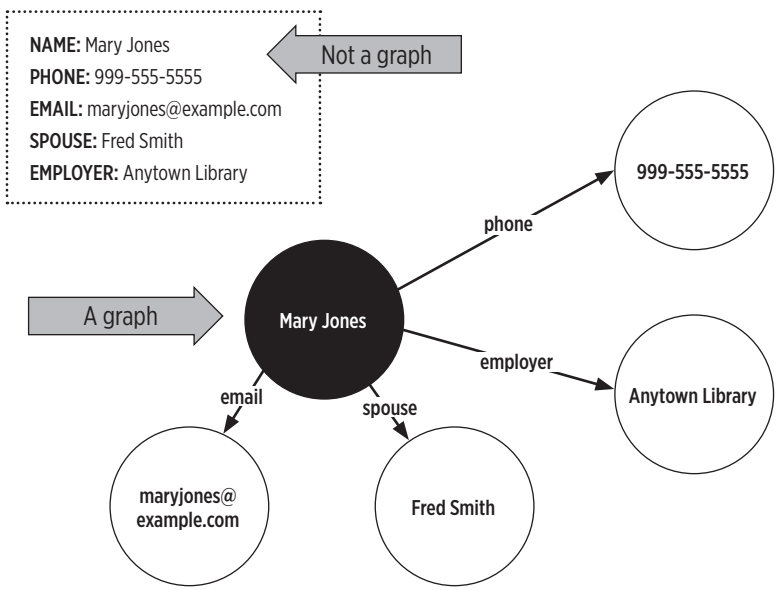
RDF identifiers are intended for machines, not humans. No one wants to read, much less type, "http://id.loc.gov/authorities/subjects/sh85038796" for the Library of Congress subject heading "Dogs." All identifiers can have human-readable labels, and the assumption is that in every situation where a human is interacting with the data, the human-readable label will be the one displayed. This includes input, which in many data creation scenarios in business applications already makes use of textual pull-down lists for easy and accurate input. Thus a cataloger will choose a subject heading, such as "Dogs in literature," from a list and the data stored will be "http://id.loc.gov/authorities/subjects/sh85038823."

Identifiers are in a sense merely a substitution of the normalized text we use today, often in the form of a formatted heading, with a particular string in the URL format. Other changes required in the shift to RDF are more radical. One of the ones that is most difficult to understand is that RDF data about resources is not stored as separate records; instead, information about a thing is in the form of a graph of statements. Graphs have no boundaries; they can grow and they

can interconnect with other graphs where their data intersects (figure 3.1). To give a simple example, the identified author in a library catalog description can interlink with the author information page on Amazon or with the encyclopedic entry about the author on Wikipedia. This assumes that these systems have knowledge of each other's identifiers, but that is increasingly the case: library authority identifiers are already found in Wikipedia entries, so this connection can be made. Data in RDF resembles synapses, with multiple connections that allow new information paths to be created as more information is added (figure 3.1).

FIGURE 3.1

A graph



The next key piece of information about RDF is actually about the nature of the World Wide Web itself. The web is an open space where millions of people and corporations and governments can put information that they wish to make public. Most contributors to the web also have other information stored in private data repositories. Although these private repositories may be in some way connected to the Internet, they are protected by user accounts and passwords, and some are protected through layers of digitally locked doors. The Semantic Web has an emphasis on the public information space, although its technology can also be used for privately held data.

There are three main principles that govern the Semantic Web that are important for understanding the rules that are applied to Semantic Web data:

- the Open World Assumption
- the Non-Unique Name Assumption
- "anyone can say anything about anything"

The Open World Assumption describes the nature of the web, which is that the web is never complete, never done, and it may not be possible to have access to all of it at any one given time. What this means is that web applications must not rely on completeness. If your bibliographic description on the open web has no title, it doesn't mean that there will never be a title, or that there hasn't ever been one. You can assume that a title exists, just not in your current view. Contrast this to a database application that has strict control over input and output, and where rules governing the data are enforced: that title must be there. In a database, a bibliographic description with no author means that the resource has no author attribution. In the web environment, that negative cannot be assumed from the absence of the element.

The Non-Unique Name Assumption (NUNA) states that any identified thing can be identified with more than one identifier. This is like real web life, where I am identified by more than one e-mail address (one at kcoyle.net and another at gmail.com), an IRC handle, and a Twitter name, in addition to my social security number, passport number, driver's license number, and so on, in "real life." On the web you cannot assume that each identifier represents a unique entity. To avoid chaos, there are ways to code identifiers as identifying to the "same" or "different" resources, but the Non-Unique Name Assumption rules any identifier pairs without explicit relationships, such that you cannot draw conclusions from identifiers alone.

The statement that "anyone can say anything about anything" is as true for today's World Wide Web as it is for the Semantic Web: there is no technical restriction on who can put information on the web. There is also no restriction on who can link to resources on the web. You may exercise content control over a web site that you create, but you cannot stop anyone else from linking to it. The same is true on the Semantic Web, where anyone can create links to your data. There is, however, a difference in the effect of linking on the Semantic Web as compared to the web of web pages, because RDF links are more meaningful than links between web pages. Links between web pages have a single meaning, which is simply "this links to that." Semantic Web links carry a meaning to the link,

such as "this is a sub-class of that," or "this is the same as/different from that." These are conditions that you should keep in mind when designing your data. To the extent that you can predict how your data might interact with other data in that vast data space, you need to design your data to "play well with others."

The basic technology of the Semantic Web is RDF. Other technologies build on that. One of these is the Web Ontology Language, OWL, which is the language developed for the creation of Semantic Web vocabularies. First, yes, OWL should be WOL, but it is OWL. Second, the RDF documentation uses the terms *vocabulary* and *ontology* interchangeably. The term *ontology* comes out of the artificial intelligence community and it implies a level of rigor in the definition of terms and their relationships. OWL is to the Semantic Web what a metadata schema has been for us in the past: OWL is how you define the terms of your domain and how you will use those terms to create your data.

OWL is a difficult standard to understand if you are not familiar with certain aspects of artificial intelligence decision-making. Many of the features that are defined in OWL sound familiar but in fact mean something different from what most of us are accustomed to. OWL is designed for a particular Semantic Web function called "inferencing." Inferencing allows you to draw conclusions from data that is present. Thus if:

Every man is a mammal
Fred is a man
Therefore, Fred is a mammal

OWL is quite a bit more sophisticated than this example implies, and includes concepts such as "inverse functional object property" and "negative data property assertion," among many others. The purpose of OWL is to define a vocabulary that can be used in complex artificial intelligence work. It also includes the ability to define some common features of metadata languages, such as cardinality (mandatory, repeatable) and equivalence (same as or different from). Unfortunately, what these features mean in OWL can be quite different from what they mean in metadata standards with which we are familiar.

The meaning of the OWL terms is governed by the RDF concept of classes, in which things being described acquire their membership in a class from the terms that define them. In our simple example above, Fred acquires "mammal-ness"

because he is described by the term "man," which itself has been defined as being of class "mammal." In artificial intelligence this mimics the human brain's ability to draw conclusions from information in the environment, generalizing from knowledge gained in one experience to apply in other situations. The Semantic Web builds up knowledge from atoms of learning, which is the opposite of the top-down approach that is common in classifications of knowledge.

There have been controversies about OWL since its inception, because it is so very complex and also so easily misunderstood. Depending on your application, you can ignore much of that complexity, but for any OWL assertion that you do use you must make sure that you understand the consequences of its use. In particular, many of the OWL declarations about terms and classes seem identical to functions in familiar programming languages. A simple example mentioned above is cardinality. Cardinality in programming languages declares the minimum and maximum allowed occurrences of a data element. If the minimum cardinality of the element is "1," that element is required—it must occur at least one time. If it is "0," then the element is optional. If the maximum cardinality is "1," the element is not repeatable, but any other number defines the number of times it can repeat in your data. In most programming situations, data that violates these rules is considered to be in error.

OWL has minimum and maximum cardinality, but their meaning has a different interpretation due to the application of the Open World Assumption and the Non-Unique Name Assumption. You can define your data as having, for example, a single creator for each given resource; the maximum cardinality of your creator element is therefore "1." If you create or encounter data that has more than one creator for a single resource, this is neither an error nor even an inconsistency in the data. Instead, applying the rules of the Semantic Web, applications that interpret OWL data will conclude that all of the creator identifiers identify a single entity because your rule says that there is only one such entity, and that entity can have any number of identifiers. At times this OWL rule may come in handy because you want to find equivalent identities, but that presumes that the data has all been coded correctly, something that most of us have learned is rarely the case. This is the big "gotcha" of OWL. OWL-based software can examine data that exists and can return a response that the data either does or does not conform to the OWL rules that have been defined for those data elements. But OWL cannot control the creation of data that meets its rules; it examines but that it does not enforce, in large part because "anyone can say anything about anything" and because OWL is intended to function in an open world that is always in flux.

This aspect of OWL generally confuses people because the OWL rules so closely resemble the rules that other programming languages use for a very different purpose: data quality control. In fact, because people often want to use OWL rules in the same way that they use programming rules in closed and controlled environments, there is now software that applies the OWL rules in closed environments, treating identifiers as uniquely identifying a single entity. This reverses two of the main truths of the Semantic Web, which are the Open World Assumption and the Non-Unique Name Assumption. It also operates on data stores where "anyone can say anything about anything" is definitely not allowed. In other words, a mirror copy of the OWL language is being used in the same way that we have always used programming languages, but not in the way intended for the Semantic Web.

Within your own closed environment, such as a local database, you clearly can do whatever you want with your data and you can impose any kinds of rules and controls that serve you and your organization. But if you open that same data to the web, the meaning of those rules will be interpreted using the Semantic Web standard meanings, which means that the Open World Assumption and the Non-Unique Name Assumption will be applied. The actual meaning of your data will be radically different in those two different environments, and operations like searches could yield very different results. The upshot of this is that the same OWL-defined vocabulary should not be used in both the closed and the open worlds.

This conflict between the controlled data stored in one's personal or corporate database and the open environment of the web is one of the hardest for data designers coming from other technology environments to overcome. There obviously is a real need to perform quality control on data, but the basis of the Semantic Web is one of discovery, not control. This is a conflict that, as of this writing, is unresolved, both in code and in terms of best practices. One possible solution, proposed by the Dublin Core Metadata Initiative (DCMI), the same people who develop the Dublin Core metadata terms, is to separate the controlling aspect of the vocabulary from its basic semantics. This isn't different from many existing metadata implementations: terms to be used are defined for their meaning, and a separate structure and rules are developed that turn those terms into a metadata record.

Dublin Core (DC) is a good example of this. Dublin Core terms are defined apart from their use in metadata. Dublin Core's element "title" is defined simply as "the name of the resource." Whether it is mandatory or optional, and whether or not it is repeatable, is not part of the definition of the term itself. Those rules would be defined in a metadata schema or in what the DCMI calls an

"application profile," which is a definition of the metadata structure and rules for a particular application. The term can be used in different ways in different metadata implementations, and the DC terms are indeed used in a wide variety of situations. However, in all uses the term retains the same meaning. This separation of meaning from rules results in maximum flexibility that allows the same terms to be used in many different applications, as Dublin Core terms are today. That flexibility is the positive outcome of this method. The negative outcome is that the separation of meaning from rules results in maximum flexibility, so that data sharing requires some adjustment between communities. The application profile, if provided in a machine-readable form, can be the basis for data sharing because communities can easily understand the structure of data created by others. Through all of that, however, a Dublin Core title remains "the name of the resource" even if some communities allow only one, some more than one, and for some the element may be optional.

We can contrast this to the primary metadata standard used in libraries today, MARC 21. This standard defines the meaning of terms and also the rules for data quality in a single standard. This is not uncommon as a data creation and management approach, however, it is undeniably a definition of a closed data world. Anyone who would use the base MARC record structure and data elements with a different set of rules governing term meanings and cardinality would simply not be creating MARC 21 data, and there would be no expectation that one could successfully combine data created under such different sets of rules.

The final aspect of the Semantic Web that I'll cover here is classes. We're all familiar with the concept of a class from scientific taxonomy and classification systems. In those systems we assign things to classes to give them the meaning of the class, putting ourselves and cats in the class "mammals," and books on mammals in one of the sub-classes of biology. Classes have a different meaning and work differently in the Semantic Web; they are not categories or boxes to put things into, but are meaningful information about things that can be used in various contexts. Classes are not exclusive in their nature, and anyone or anything can have the qualities of more than one class. This is much like the real world, where a person can be an employee in one context, a parent in another, and a volunteer firefighter in yet another. Rather than assigning a thing to a class, the class is deduced based on how something is described. Our rules may say that persons with paychecks are employees, those with children are parents, and those who are members of Volunteer Brigade 7 are volunteer firefighters, and anyone can be all three. By attributing characteristics to the thing we are describing, we build up our world by describing it. This, too, fits into the methods of artificial intelligence where their creations must be able to make deductions about newly

encountered things in the world based on information, as we do in real life. We recognize chairs as chairs even if we haven't seen a particular chair before. We understand that a person is a police officer because anyone wearing that uniform is a police officer, even if we haven't seen that person before. We are moved to open the door for a person carrying packages because we know that it's hard to open a door when your hands are full, in spite of not having been in this exact situation (same person, same door, same packages) before. All of this computation happens quickly and naturally in the human brain, and some of it can be imitated through code if the right information is given about things we describe on the web.

The preceding describes some of the fundamentals of the Semantic Web. The Semantic Web is implemented as *linked data,* a set of common practices for data on the web. One of these practices, the use of http-based identifiers, has been discussed above. Other practices have to do with making sure that your data can be used in the open environment of the World Wide Web. There are standard ways to define your metadata so that others can understand it and potentially use it. Linked data is a mix-and-match technology, and people are encouraged to make use of metadata definitions that exist rather than inventing their own. Any description can be made up of metadata from a number of different sources, and can use descriptive elements found anywhere on the open web.

From this description you can undoubtedly conclude that a future library data standard using linked data would look considerably different from the data we have today. The purpose of linked data is both discovery, through hyperlinks, and new knowledge creation, by linking between previously separate communities and their data stores. Those looking at linked data for libraries are focused on the library catalog and its discovery function. Our current catalog data is very different in its goals and content from data that would play well in a linked data environment. The challenge for us is to make this transition intelligently, and in a way that serves library users. The remainder of this book looks at current efforts with that challenge in mind.