Advantages and Myths of RDF

by

Michael K. Bergman
Structured Dynamics LLC

April 22, 2009

A 10th Birthday Salute to RDF's Role in Powering Data Interoperability

There has been much welcomed visibility for the semantic Web and linked data of late. Many wonder why it has not happened earlier; and some observe progress has still been too slow. But what is often overlooked is the *foundational* role of RDF — the Resource Description Framework.

From my own perspective focused on the issues of data interoperability and data federation, RDF is the single most important factor in today's advances. Sure, there have been other models and other formulations, but I think we now see the Goldilocks "just right" combination of expressiveness and simplicity to power the foreseeable future of data interoperability.

So, on this 10th anniversary of the birth of RDF **[1]**, I'd like to re-visit and update some much dated discussions regarding the advantages of RDF, and more directly address some of the mis-perceptions and myths that have grown up around this most useful framework.

A Simple Intro to RDF

RDF is a data model that is expressed as simple *subject-predicate-object* "triples". That sounds fancy, but just substitute verb for *predicate* and noun for *subject* and *object*. In other words: *Dick sees Jane*; or, the *ball is round*. It may sound like a kindergarten reader, but it is how data can be easily represented and built up into more complex structures and stories.

A *triple* is also known as a "statement" and is the basic "fact" or asserted unit of knowledge in RDF. Multiple statements get combined together by matching the *subjects* or *objects* as "nodes" to one another (the *predicate*s act as connectors or "edges"). As these node-edge-node triple statements get aggregated, a network structure emerges, known as the RDF *graph*.

The referenced "resources" in RDF triples have unique identifiers, IRIs, that are Web-compatible and Web-scalable. These identifiers can point to precise definitions of predicates or refer to specific concepts or objects, leading to less ambiguity and clearer meaning or semantics.

In my own company's approach to RDF, basic instance data is simply represented as attribute-value pairs where the *subject* is the instance itself, the *predicate* is the attribute, and the *object* is the value. Such instance records are also known as the ABox. The structural relationships within RDF are defined in ontologies, also known as the TBox, which are basically equivalent to a schema in the relational data realm.

RDF triples can be applied equally to all structured, semi-structured and unstructured content. By defining new types and predicates, it is possible to create more expressive vocabularies within RDF. This expressiveness enables RDF to define controlled vocabularies with exact semantics. These features make RDF a powerful data model and language for data federation and interoperability across disparate datasets.

There are many excellent introductions or tutorials to RDF; a recommended sampling is shown in the endnotes **[2]**.

Is RDF a Framework, Data Model or Vocabulary?

Well, the answer to the rhetorical question is, all three!

The RDF data model provides an abstract, conceptual framework for defining and using metadata and metadata vocabularies. See: We were able to use all three concepts in a single sentence!

> *The RDF model draws on well-established principles from various data representation communities. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources and an RDF model can therefore resemble an entity-relationship diagram. . . . In object-oriented design terminology, resources correspond to objects and properties correspond to instance variables.* **[1]**

But, actually, because RDF is simultaneously a framework, data model and basis for building more complex vocabularies, it is both simple and complex at the same time.

It is first perhaps best to understand basic RDF as a data model of triples with very few (or unconstrained) semantics **[3]**. In its base form, it has no range or domain constraints; has no existence or cardinality constraints; and lacks transitive, inverse or symmetrical properties (or *predicates*) **[4]**. As such, basic RDF has limited reasoning support. It is, however, quite useful in describing static things or basic facts.

In this regard, RDF in its base state is nearly adequate for describing the simple instances and data records of the world, what is called the ABox in description logics.

RDFS (RDF Schema) is the next layer in the RDF stack designed to overcome some of these baseline limitations. RDFS introduces new predicates and classes that bound these semantics. Importantly, RDFS establishes the basic constructs necessary to create new vocabularies, principally through adding the *class* and *subClass* declarations and adding *domain* and *range* to *properties* (the RDF term for *predicates*). Many useful vocabularies have been created with RDFS and it is possible to apply limited reasoning and inference support against them.

The next layer in the RDF stack is OWL, the Web Ontology Language. It, too, is based on RDF. The first versions of OWL were themselves layered from OWL Lite to OWL DL to OWL Full. OWL Lite and OWL DL are both decidable through the first-order logic basis of description logics (the basis for the acronym in OWL DL). OWL Full is not decidable, but provides an OWL counterpart to fragmented RDF and RDFS statements that are desirable in the aggregate, with reasoning applied where possible.

OWL provides sufficient expressive richness to be able to describe the relationships and structure of entire world views, or the so-called terminological (TBox) construct in description logics. Thus, we see that the complete structural spectrum of description logics can be satisfied with RDF and its schematic progeny, with a bit of an escape hatch for combining poorly defined or structural pieces via using OWL Full **[5]**.

However, RDF is ***NOT*** a particular serialization. Though XML was the original specified serialization and still is the defined RDF MIME type (application/rdf+xml; other serializations take the form text/turtle or text/n3 or similar), it is not necessary to either write or transmit RDF in the XML syntax.

In any event, depending on its role and application, we can see that RDF is a foundation, in careful expressions based in description logics, that lends itself to a clean expression and separation of concerns. With RDF and RDFS, we have a data model and a basis for vocabularies well suited to instance data (ABox). With RDFS and OWL, we have an extended schema structure and ontologies suitable for describing and modeling the relationships in the world (TBox). Thus, RDF is a framework for modeling all forms of data, for describing that data through vocabularies, and for interoperating that data through shared conceptualizations (ontologies) and schema.

Rationale for a Canonical Data Model

In the context of data interoperability, a critical premise is that a single, canonical data model is highly desirable. Why? Simply because of 2N v N^2. That is, a single reference ("canon") structure means that fewer tool variants and converters need be developed to talk to the myriad of data formats in the wild. With a canonical data model, talking to external sources and formats (N) only requires converters to the canonical form (2N). Without a canonical model, the combinatorial explosion of required format converters becomes N^2 **[6]**.

Note, in general, such a canonical data model merely represents the agreed-upon internal representation. It need not affect data transfer formats. Indeed, in many cases, data systems employ quite different internal data models from what is used for data exchange. Many, in fact, have two or three favored flavors of data exchange such as XML, JSON or the like.

In most enterprises and organizations, the relational data model with its supporting RDBMs is the canonical one. In some notable Web enterprises – say, Google for example – the exact details of its internal canonical data model is hidden from view, with APIs and data exchange standards such as GData being the only visible portions to outside consumers.

Generally speaking, a canonical, internal data standard should meet a few criteria:

- Be expressive enough to capture the structure and semantics of any contributing dataset
- Have a schema itself which is extensible
- Be performant
- Have a model to which it is relatively easy to develop converters for different formats and models
- Have published and proven standards, and
- Have sufficient acceptance so as to have many existing tools and documentation.

Other desired characteristics might be for the model and many of its tools to be free and open source, suitable to much analytic work, efficient in storage, and other factors.

Though the relational data model is numerically the most prevalent one in use, it has fallen out of favor for data federation purposes. This loss of favor is due, in part, to the fragile nature of relational schema,

which increases maintenance costs for the data and their applications, and incompatibilities in standards and implementation.

Though still comparatively young with a smaller-than-desirable suite of tools and applications support **[7]**, RDF is perhaps the ideal candidate for the canonical data model. To understand why, let's now switch our discussion to the advantages of RDF.

Advantages of RDF

It is surprisingly difficult to find a consolidated listing of RDF's advantages. The W3C, the developer of the specification, first published on this topic in the late 1990s, but it has not been updated for some time **[8]**. Graham Klyne has a better and more comprehensive presentation, but still one that has not been updated since 2004 **[4]**.

I believe data interoperability to be RDF's premier advantage, but there are many, many others.

Another advantage that is less understood is that RDF and its progeny can completely switch the development paradigm: data can now drive the application, and not the other way around. Frankly, we are just at the beginning realizations of this phase with such developments as linked data and even whole applications or application languages being written in RDF **[9]**, but I think time will prove this advantage to be game-changing.

But, there are many perspectives that can help tease out RDF's advantages. Some of these are discussed below, with the accompanying table attempting to list these 'Top Sixty' advantages in a single location.

Standard, Open and Expressive

In its ten year history, RDF has spawned many related languages and standards. The W3C has been the shepherd for this process, and there are many entry locations on the World Wide Web Consortium's Web site to begin exploring these options **[10]**. These standards extend from the RDF, RDFS and OWL vocabularies and languages noted above that give RDF its range of expressiveness, to query languages (*e.g.*, SPARQL), transformation languages (*e.g.*, GRDDL), rule languages (*e.g.*, RIF), and many additional constructs and standards.

The richness of this base of standards is only now being tapped. The combination of these standards and the tools they are spawning is just beginning. And, because it is so easily serialized as XML, a further suite of tools and capabilities such as XPath or XSLT or XForms may be layered onto this base.

Moreover, one is not limited in any way to XML as a serialization. RDF itself has been serialized in a number of formats including RDF/XML, N3, RDFa, Turtle, and N-triples. Also, RDF's simple *subject-predicate-object* data model can readily convert human-readable and easily authored instance records (*subject*) written in the style of attribute-value pairs (*predicate-object*). As such, RDF is an excellent conversion target for all forms of naïve data structs **[11]**.

Data Interoperability

Indeed, it is in data exchange and interoperability that RDF really shines. Via various processors or extractors, RDF can capture and convey the metadata or information in unstructured (say, text), semi-structured (say, HTML documents) or structured sources (say, standard databases). This makes RDF almost a "universal solvent" for representing data structure.

> *"The semantic Web's real selling point is URI-based data integration."*
> — Harry Halpin **[12]**

Because of this universality, there are now more than 100 off-the-shelf 'RDFizers' for converting various non-RDF notations (data formats and serializations) to RDF **[13]**. Because of its diversity of serializations and simple data model, it is also easy to create new converters. Generalized conversion languages such as GRDDL provide framework-specific conversions, such as for microformats.

Once in a common RDF representation, it is easy to incorporate new datasets or new attributes. It is also easy to aggregate disparate data sources as if they came from a single source. This enables meaningful composition of data from different applications regardless of format or serialization.

Simple RDF structures and predicates enable synonyms or aliases to also be easily mapped to the same types or concepts. This kind of semantic matching is a key capability of the semantic Web. It becomes quite easy to say that your *glad* is my *happy*, and they indeed talk about the same thing.

What this mapping flexibility points to is the immense strengths of RDF in representing diverse schema, the next major advantage.

Schema Unbound

The single failure of data integration since the inception of information technologies — for more than 30 years, now — has been schema rigidity or schema fragility. That is, once data relationships are set, they remain so and can not easily be changed in conventional data management systems nor in the applications that use them.

Relational database management (RDBM) systems have not helped this challenge, at all. While tremendously useful for transactions and enabling the addition of more data records (instances, or rows in a relational table schema), they are not adaptive nor flexible.

Why is this so?

In part, it has to do with the structural "view" of the world. If everything is represented as a flat table of rows and columns, with keys to other flat structures, as soon as that representation changes, the tentacled connections can break. Such has been the fragility of the RDBMS model, and the hard-earned resistance of RDBMS administrators to schema growth or change.

Yet, change is inevitable. And thus, this is the source of frustration with virtually all extant data systems.

RDF has no such limitations. And, for those from a conventional data management perspective, this RDF flexibility can be one of the more unbelievable aspects of this data model.

As we have noted earlier, RDF is well suited and can provide a common framework to represent both instance data and the structures or schema that describe them, from basic data records to entire domains or world views. In fact, whatever schema or structure that characterizes the input data — from simple instance record layouts and attributes to complete vocabularies or ontologies — also embodies domain knowledge. This structure can be used at time of ingest as validity or consistency checks.

As a framework for data interoperability, RDF and its progeny can ingest all relations and terminology, with connections made via flexible predicates that assert the degree and nature of relatedness. There is no need for ingested records or data to be complete, nor to meet any prior agreement as to structure or schema.

Increment, Evolve, Extend, Adapt . . .

Indeed, the very fluidity of RDF and structures based on it is another key strength. Since a basic RDF model can be processed even in the absence of more detailed information, input data and basic inferences can proceed early and logically as a simple *fact basis*. This strength means that either data or schema may be ingested and then extended in an incremental or partial manner. Partial representations can be incorporated as readily as complete ones, and schema can extend and evolve as new structure is discovered or encountered.

This is revolutionary. RDF provides a data and schema representation framework that can evolve and adapt to what data exists and what structure is known. As new data with new attributes are discovered, or as new relationships are found or realized, these can be added to the existing model *without any change whatsoever to the prior existing schema*.

This very adaptability is what enables RDF to be viewed as data-driven design. We can deal with a partial and incomplete world; we can learn as we go; we can start small and simple and evolve to more understanding and structure; and we can preserve all structure and investments we have previously made.

And applications based on RDF work the same way: they do not need to process or account for information they don't know or understand. We can easily query RDF models without being affected whatsoever by unreferenced or untyped data in the basic model.

By replacing the rigid relational data model with one based on RDF, we gain robustness, flexibility, universality and structural persistence over fragility.

Existing technologies such as SQL and the relational model were devised without the specific requirements of disparate, uncontrolled, large-scale integration. Though the relational model enabled us to build efficient data silos and transaction systems, RDF now enables us to finally federate them.

'Top Sixty' Benefits of RDF

- A foundation based in description logics that lends itself to clean expression and separation of concerns regarding instance data (ABox) and schema structure (TBox)
- RDF's unique identifiers, IRIs, are Web-compatible and are Web-scalable
- Potential use of inferencing to contextually broaden search, retrieval and analysis
- Potential use of its structure to automatically drive applications and tools, including populating context-relevant dropdown lists and auto-completion
- Based on open source, languages and standards
- A comparatively complete suite of specifications including languages, schema and tools (*e.g.*, RDF, RDF Schema, OWL, RIF, SPARQL, GRDDL, etc.)
- A choice of a variety of serializations and notations, including RDF/XML, N3, RDFa, Turtle, N-triples, as well as possible expression in many non-RDF notations
- Instance records in human-readable, easily authored attribute-value formats can be readily converted to the *s-p-o* RDF "triple" data model
- Can capture metadata and structure from unstructured, semi-structured and structured data
- More than 100 off-the-shelf 'RDFizers' exist for converting various non-RDF

'Top Sixty' Benefits of RDF

notations (data formats and serializations)
- Easy and cost-effective incorporation of new datasets wherein only new attributes require a structure update; all others simply get mapped
- Aggregate processing of disparate sources as if they came from a single source
- A ready structure for synonym and alias matching when merging or matching datasets
- In converting non-RDF data, the ability to bring a more formal class structure to the description of things
- Common framework and vocabulary for representing instance data
- Common framework and vocabulary for representing data structures and schema
- Can describe simple data structs to complete vocabularies/ontologies to processing and inferencing rules
- Schema can be calculated from the ingested triples; thus, can either generate schema from scratch or be used to cross-check prior schema
- Can accept and store data with different structure in a general RDF container (*e.g.*, all animals v a specific bird)
- Eliminates the trade-off between good design and performance for related structure (*e.g.*, full names *v* first and last names)
- Untyped relations can still have single operations performed against them
- More formal RDF structures (*e.g.*, ontologies) embody domain expertise within their subject structure
- Readily extensible with schema that are also machine readable, bringing about a high degree of automation
- Allows data that is structured slightly differently to be stored together in the lowest common denominator of an RDF statement
- No need for upfront schema agreement; can evolve, extend and adapt
- Allows the schema to change independently of the data without requiring any existing data to be thrown away or padded with NULLs
- The basic RDF model can be processed in the absence of more detailed information as a simple *fact basis*
- Schema based on RDF can be extended and grown incrementally without impacting the existing datastore
- As a corollary, development based on RDF can also be incremental, reducing the need to "design it at once" or "design it right" up front
- RDF models and apps lend themselves to experimentation and agile development
- Information can be gathered incrementally from multiple sources
- Data and schema can be ingested, represented and conveyed in "partial" form
- Structure and schema can evolve incrementally in concert with new understandings and new data
- All prior investments in structure and schema can be maintained
- Because of conceptual closeness to the relational data model, it is possible to represent RDF in a relational database and vice versa
- RDF thus has the ability to take advantage of historical RDBMs and SQL query optimizations

'Top Sixty' Benefits of RDF

- Ability to create RDF "views" or wrappers over relational schema that can be queried via SPARQL
- A common storage format based on the triple or quad; suitable for datastore hosting by relational database management systems
- The use of untyped relations reduces the total number of relations to be handled, with operations over them only needed once
- Relational systems can serve instance data *in situ* (ABox) while interoperability is provided by an RDF structural and schema layer (TBox)
- Ability to do specialized work, such as inferencing
- Use of a set-based semantics and queries
- Via its SPARQL query language, easy mechanisms to drive faceted search and other browsing and viewing tools
- Because of how RDF works it is possible to query a dataset without knowing anything about the data in advance
- Ability to generalize selection, viewing and publishing tools driven solely from the RDF structure; as the structure changes, tools automatically reflect those changes (*e.g.*, plug-and-play)
- Can easily create and apply inferencing tables over RDF datastores **[14]**
- The RDF graph brings all the advantages and generality of structuring information using graphs
- A graph is, itself, a unique form of data type with unique algorithms and analytic features
- Graphs are modular and can be readily combined or broken apart
- Graphs can be used for scalable, parallelized information processing
- Unique types of search and discovery can occur with RDF graphs
- Graphs provide the ability to visualize and navigate large network structures
- Queries are unaffected by unreferenced values in the source data
- Emerging *lingua franca* of the semantic Web and 'Web of data'
- Strong compatibility with "linked data" based on Web access (HTTP) and IRI identifiers
- RDF is readily adaptable to the open-world assumption (OWA)
- Relation to the semantic Web means much global information and data can be admixed with local content
- Across all global sources the potential for powerful data "mesh-ups" conjoining related information
- Network effects such as shared vocabularies, shared background knowledge, collective authoring, annotating and curating, and
- RDF is an emergent data model.

Yet, Still Kissing Cousins with the Relational Model

Despite these differences in fragility and robustness, there are in fact many logical and conceptual affinities between the relational model and the one for RDF. An excellent piece on those relations was written by Andrew Newman a bit over a year ago **[15]**.

RDF can be modeled relationally as a single table with three columns corresponding to the *subject-predicate-object* triple. Conversely, a relational table can be modeled in RDF with the *subject* IRI derived from the primary key or a blank node; the *predicate* from the column identifier; and the *object* from the cell value. Because of these affinities, it is also possible to store RDF data models in existing relational databases. (In fact, most RDF "triple stores" are RDBM systems with a tweak, sometimes as "quad stores" where the fourth tuple is the *graph*.) Moreover, these affinities also mean that RDF stored in this manner can also take advantage of the historical learnings around RDBMS and SQL query optimizations.

Just as there are many RDFizers as noted above, there are also nice ways to convert relational schema to RDF automatically. OpenLink Software, for example, has its RDF "Views" system that does just that **[16]**. Given these overall conceptual and logical affinities the W3C is also in the process of graduating an incubator group to an official work group, RDB2RDF **[17]**, focused on methods and specifications for mapping relational schema to RDF.

What is emerging is one vision whereby existing RDBM systems retain and serve the instance records (ABox), while RDF and its progeny provide the flexible schema scaffolding and structure over them (TBox). Architectures such as this retain prior investments, but also provide a robust migration path for interoperating across disparate data silos in a performant way.

Data-driven Applications

As developers, one of our favorite advantages of RDF is its ability to support data-driven applications. This makes even further sense when combined with a Web-oriented architecture that exposes all tools and data as RESTful Web services **[18]**.

Two tool foundations are the RDF query language, SPARQL **[19]**, and inferencing. SPARQL provides a generalized basis for driving reports and templated data displays, as well as standard querying. Utilizing RDF's simple triple structure, SPARQL can also be used to query a dataset without knowing anything in advance about the data. This provides a very useful discovery mode.

Simple inferencing can be applied to broaden and contextualize search, retrieval and analysis. Inference tables can also be created in advance and layered over existing RDF datastores **[14]** for speedier use and the automatic invoking of inferencing. More complicated inferencing means that RDF models can also perform as complete conceptual views of the world, or knowledge bases. Quite complicated systems are emerging in such areas as common sense (with OpenCyc) and biological systems **[20]**, as two examples.

RDF ontologies and controlled vocabularies also have some hidden power, not yet often seen in standard applications: by virtue of its structure and label properties, we can populate context-relevant dropdown lists and auto-complete entries in user interfaces solely from the input data and structure. This ability is completely generalizable solely on the basis of the input ontology(ies).

A Graph Representation

As the intro noted, when RDF triples get combined, a graph structure emerges. (Actually, it can most formally be described as a directed graph.) A graph structure has many advantages. While we are seeing much starting to emerge in the graph analysis of social networks, we could also fairly argue that we are still at the early stages of plumbing the unique features of graph ("network") structures.

Graphs are modular and can be both readily combined and broken apart. From a computational standpoint, this can lend itself to parallelized information processing (and, therefore, scalability). With specific reference to RDF it also means that graph extractions are themselves valid RDF models.

Graph algorithms are a significant field of interest within mathematics, computer science and the social sciences. Via approaches such as network theory or scale-free networks, topics such as relatedness, centrality, importance, influence, “hubs” and “domains”, link analysis, spread, diffusion and other dynamics can be analyzed and modeled.

Graphs also have some unique aspects in search and pattern matching. Besides options like finding paths between two nodes, depth-first search, breadth-first search, or finding shortest paths, emerging graph and pattern-matching approaches may offer entirely new paradigms for search.

Graphs also provide new approaches for visualization and navigation, useful for both seeing relationships and framing information from the local to global contexts. The interconnectedness of the graph allows data to be explored via contextual facets, which is revolutionizing data understanding in a way similar to how the basic hyperlink between documents on the Web changed the contours of our information spaces **[21]**.

Many would argue (as do I), that graphs are the most “natural” data structure for capturing the relationships of the real world. If so, we should continue to see new algorithms and approaches emerge based on graphs to help us better understand our information. And RDF is a natural data model for such purposes.

Open World Applications and the Semantic Web

Ultimately, data interoperability implies a global context. The design of RDF began from this perspective with the semantic Web.

This perspective is firstly grounded in the open-world assumption: that is, the information at hand is understood to be incomplete and not self-contained. Missing values are to be expected and do not falsify what is there. A corollary assumption is there is always more information that can be added to the system, and the design should not only accommodate, but promote, that fact.

As the *lingua franca* for the semantic Web, using RDF means that many new data, structures and vocabularies now become available to you. So, not only can RDF work to interoperate your own data, but it can link in useful, external data and schema as well.

Indeed, the concept of linked data now becomes prominent whereby RDF data with unique IRIs as their universal identifiers are exposed explicitly to aid discovery and interlinking. Whether internal data is exposed in the linked data manner or not, this external data can now be readily incorporated into local contexts. The Linking Open Data movement that is promoting this pattern has become highly successful, with billions of useful RDF statements now available for use and consumption **[10]**.

The semantic Web and RDF is enabling the data federation scope to extend beyond organizational boundaries to embrace (soon) virtually all public information. That means that, say, local customer records can now be supplemented with external information about specific customers or products. We are really just at the nascent stages of such data “mesh-ups” with many unforeseen benefits (and, challenges, too, such as privacy and identity and ownership) likely to emerge.

At Web scales, we will see network effects also emerge in areas such as shared vocabularies, shared background knowledge, and collective authoring, annotating and curating. To be sure the traditional

work of trade associations and standards bodies will continue, but likely now in much more operable ways.

Myths of RDF

Throughout the years, a number of myths have grown up around RDF. Some, unfortunately, were based on the legacy of how RDF was first introduced and described. Other myths arise from incomplete understanding of RDF's multiple roles as a framework, data model, and basis for vocabularies and conceptual descriptions of the world.

The accompanying table lists the "Top Ten" of myths I have found to date. I welcome other pet submissions. Perhaps soon we can get to the point of a clearer understanding of RDF.

'Top Ten' Myths of RDF

1. ***RDF is equivalent to XML*** — perhaps the biggest PR error in RDF's first introduction was to tie RDF so closely with XML. RDF is a data model as described herein that has no dependence on XML and exists in abstract form separate from it
2. ***RDF is written or expressed in XML*** — in a related way, RDF can be serialized (expressed) in many forms other than XML
3. ***RDF and OWL are independent*** — OWL is a language grounded in RDF and a natural extension of the RDF "stack"; OWL is at the other end of the expressiveness spectrum **[22, 23]**
4. ***RDF is a serialization*** — no; XML is a serialization, RDF is a data model, framework and basis for constructing vocabularies
5. ***Basic RDF has no semantics*** — though limited and purposefully free, basic RDF in fact has extremely well considered semantics; an essential document for any practitioner is **[3]**
6. ***RDF is too complex*** — it depends, right? At the level of the basic triple, RDF is extremely simple and is the best place to start learning about RDF
7. ***RDF is too simple*** — it depends, right? At the level of OWL ontologies, RDF can capture virtually any relationship and aspect of the world; see **[5]** for a great start
8. ***RDF is useful for "large" datasets only*** — the real purpose of RDF is data interoperability, which is needed ***any*** time two or more datasets are combined, regardless of size
9. ***(Conversely and paradoxically), RDF is not scalable*** — this premise is still being tested, but we now have very large-scale experience with the government and in the Billion Triples Challenge
10. ***RDF is not performant*** — daily we keep learning more about optimizations, query and re-write strategies, and the like. Orri Erling **[24]** does some of the best work around in this area and writes lucid explanations on his blog. Moreover, RDF systems are easily embedded in WOA architectures, which prove themselves daily at global Web scales.

Conclusion

Emergence is the way complex systems arise out of a multiple of relatively simple interactions, exhibiting new and unforeseen properties in the process. RDF is an emergent model. It begins as simple "fact" statements of triples, that may then be combined and expanded into ever-more complex structures and stories.

As an internal, canonical data model, RDF has advantages over any other approach. We can represent, describe, combine, extend and adapt data and their organizational schema flexibly and at will. We can explore and analyze in ways not easily available with other models.

And, importantly, we can do all of this without the need to change what already exists. We can augment our existing relational data stores, and transfer and represent our current information as we always have.

We can truly call RDF a disruptive data model or framework. But, it does so without disrupting what exists in the slightest. And that is a most remarkable achievement.

---

[1] Actually, it is just a few weeks past. The first RDF specification was published as: Ora Lassila and Ralph R. Swick, eds., 1999. "Resource Description Framework (RDF) Model and Syntax Specification," *W3C Recommendation*, 22 February 1999; see http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. Of course, RDF had been in development under various names for some time. To my knowledge, the first public explanation specific to the RDF name was by Tim Bray, "RDF and Metadata," on XML.com, June 09, 1998; see http://www.xml.com/pub/a/98/06/rdf.html. I'm measuring RDF's birthday in relation to it being published as an official standard (recommendation) per the first reference.

[2] I first recommend an older introduction by Ian Davis, http://research.talis.com/2005/rdf-intro/. There is a more recent, shorter version by Davis and Tom Heath, *The 30 Minute Guide to RDF and Linked Data,* at http://www.slideshare.net/iandavis/30-minute-guide-to-rdf-and-linked-data. Also, Joshua Tauberer's write-up at http://www.rdfabout.com/intro/? is quite excellent.

[3] Patrick Hayes, 2004. "RDF Semantics," a *W3C Recommendation*, February 2004. See http://www.w3.org/TR/rdf-mt/.

[4] Graham Klyne, 2004. "Semantic Web and RDF," on the *Nine by Nine* Web site (http://www.ninebynine.net/), 4 May 2004; see http://www.ninebynine.org/Presentations/20040505-KelvinInsitute.pdf.

[5] The soon-to-be-released recommendation of OWL 2 is best introduced through the recent: OWL 2 Working Group, eds., 2009. "OWL 2 Web Ontology Language: Document Overview," *W3C Working Draft*, 27 March 2009; see http://www.w3.org/TR/owl2-overview/.

[6] The canonical data model is especially prevalent in enterprise application integration. An interesting animated visualization of the canonical data model may be found at: http://soa-eda.blogspot.com/2008/03/canonical-data-model-visualized.html.

[7] Still, my own Sweet Tools listing of RDF and -related tools now contains nearly 800 items.

[8] The RDF Advantages Page; see http://www.w3.org/RDF/advantages.html.

[9] See, for example, Neno, the Semantic Web Programming Environment, at: http://neno.lanl.gov/Home.html; and Ripple, at http://code.google.com/p/ripple/. The developers of these systems are now combining efforts.

[10] Here are some useful starting points for RDF at the World Wide Web Consortium (W3C): Begin at the W3C's ESW wiki. The Linking Open Data community maintains its own people and projects listings as well. Current topics are discussed on the W3C's semantic Web mailing lists. The W3C maintains a good general semantic Web tools, with specific listings of RDF Triplestores.

[11] Michael Bergman, 2009. "'Structs': Naïve Data Formats and the ABox," on the *AI3 blog*, January 22, 2009; see http://www.mkbergman.com/?p=471. And, Ibid, 2009. "'Making Linked Data Reasonable using Description Logics, Part 4," on the *AI3 blog*, February 23, 2009; see http://www.mkbergman.com/?p=478.

[12] Harry Halpin, video interview with Marcos Caceres, "GRDDL, Bridging the Interwebs?," August 4, 2008, on StandardsSuck.org. See http://standardssuck.org/grddl-bridging-the-interwebs.

[13] See, for example, these Virtuoso RDF cartridges (http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSponger) or listing of RDFizers (http://simile.mit.edu/wiki/RDFizers).

[14] OpenLink Software, 2009. "17.6. Inference Rules & Reasoning,", part of the online Virutoso User Manual; see: http://docs.openlinksw.com/virtuoso/rdfsparqlrule.html.

[15] Andrew Newman, 2007. "A Relational View of the Semantic Web," published on XML.com, March 14, 2007; see http://www.xml.com/pub/a/2007/03/14/a-relational-view-of-the-semantic-web.html.

[16] OpenLink Software, 2009. "17.4.3. RDF Views over RDBMS Data Source," part of the online Virutoso User Manual; see: http://docs.openlinksw.com/virtuoso/rdfsparqlintegrationmiddleware.html#rdfviews. Also see OpenLink Software, 2007. *Virtuoso RDF Views – Getting Started Guide*, v1.1, June 2007; see http://virtuoso.openlinksw.com/Whitepapers/pdf/Virtuoso_SQL_to_RDF_Mapping.pdf.

[17] W3C, 2009. *RDB2RDF Working Group Charter*, revised February 24, 2009; see http://www.w3.org/2005/Incubator/rdb2rdf/WG-draft-charter/.

[18] See further my various blog posts on Web-oriented architecture (WOA).

[19] Especially recommended as an introductory tutorial is: Lee Feigenbaum, 2008. "SPARQL By Example: A Tutorial," Sept. 17, 2008; see http://www.cambridgesemantics.com/2008/09/sparql-by-example.

[20] Many disciplines are embracing RDF. But, in biology, some exemplar projects are the Bio2RDF genomics project; the Linking Open Drug Data (LODD) initiative, which is a sub-project of the W3C's broader Health Care and Life Sciences Interest Group (HCLSIG); the Neurocommons project; and the RDF branches of the Open Biomedical Ontologies (OBO) project and foundry.

[21] A very nice visualization of graph-driven structures in relation to information discovery and navigation is provided by Rama Hoetzlein, 2007. Quanta: The Organization of Human Knowedge: Systems for Interdisciplinary Research, a Master's Thesis, University of California, Santa Barbara, June 2007; see http://www.rchoetzlein.com/quanta/index.htm.

[22] The original phrasing of this *Myth* used the term "distinct", which Ted Thibodeau Jr rightly questioned. This myth goes to the heart of what I think is a false separation of the RDF and OWL "camps". As the intro noted, I see a natural progression from RDF → RDFS → OWL, with the transition representing more precise semantics and expressiveness. Describing simple things simply, especially for linked data as mostly practiced, works well in RDF and RDFS. Once world views and conceptual schema are desired for inter-relating these things, RDFS and OWL become the better option. OWL Full (including OWL 2, see **[23]**) is fully grounded in RDF semantics. However, since OWL Full is not decidable, a subset of that, OWL DL, is still expressible as RDF but now consistent with description logics. This approach can provide more inferencing and reasoning power, at the slight cost of greater care in the semantics used and relationships asserted. In the end, the "distinction" between RDF and OWL is really a difference in use cases and intentions, imo.

[23] Michael Schneider, ed., 2009. *OWL 2 Web Ontology Language RDF-Based Semantics*, W3C Working Draft 21 April 2009; see http://www.w3.org/TR/owl2-rdf-based-semantics/.

[24] See Orri Erling's Weblog at: http://www.openlinksw.com/weblogs/oerling/.