Learning By Example: Designing and Developing Linked Data Application

Karim Tharani
University of Saskatchewan

Abstract
According to constructivist theory of learning, new knowledge is acquired on the basis of what is already known by learners. Learning to build Linked Data applications challenges traditional web technologists to think differently at every stage of the design and development process, starting from data modeling all the way to presenting data on the Linked Data web. Thus to understand and adopt an emerging and transformative web technology such as Linked Data, it is useful for web technologists to learn it in the context of prevalent web tools and technologies. This paper presents a comparative and illustrative example of designing and developing Linked Data application using the traditional and familiar LAMP (Linux Apache MySQL and PHP) web technology stack.

Introduction

In the current web, information is encoded in Hypertext Markup Language (HTML) to present information in web browsers for users. The HTML markup language was specifically designed to parcel information as documents for easy access and consumption by humans. Two central constructs make HTML functional and popular—hyperlinks and webpages. *Hyperlinks* provide a way for humans to access information whereas *webpages* provide a way to parcel and present information in manageable volume. These two constructs, however, are neither intrinsic to the web nor the Internet but are byproducts of how information is encoded using HTML. While HTML encoding makes information discernable to humans by presenting it as a collection of documents, it also makes information non-actionable for machines (i.e. special purpose software and hardware designed and dedicated to parsing and processing large volumes of information intelligibly) from a computing point of view. What if information could exist and be accessed without being committed to a webpage? What if we could make advanced judgments about information to be served without having to click on hyperlinks? Linked Data makes this possible by proposing an alternate markup language for encoding information over the web called Resource Description Framework (RDF). Linked Data is part of the broader *Semantic Web* movement to transform the current *web of documents* to a *web of data* (Shadbolt, Hall, & Berners-Lee, 2006). In essence, Linked Data is an emerging and open set of principles and praxis geared toward transforming the current web into an open database usable by both humans and machines (Bizer, Heath, & Berners-Lee, 2009).

How does Linked Data work, anyway?

A more intuitive way to understand how Linked Data and RDF work together is to think of web resources (such as a relevant piece of information, book, document, etc.) as *nodes* of a graph in space interconnected or linked to each other by multiple *edges* with meaningful labels or descriptors. Unlike HTML, RDF allows links between resources to be described using controlled vocabularies or ontologies. The end result of using RDF is that, by virtue of all the descriptors surrounding resources (or nodes), there is a deeper

and more accurate awareness about the nature of knowledge resources not only for humans but also for machines. This awareness of data by machines could allow the web to "function more like relational databases, providing much more accurate search results—the ability to distinguish between a book that is written *about* a person, as opposed to a book that is written *by* a person, for example" (Byrne & Goddard, 2010). In order for Linked Data to deliver on this vision, however, web technologists and information professionals need to follow a set of rules for publishing information on the web. These rules were outlined by Berners-Lee, the founder of the World Wide Web, and have now come to be known as the Linked Data Principles:

1. Use URIs (or Uniform Resource Identifier) as names for things
2. Use HTTP URIs so that people can look up those names
3. URIs should provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things

The use of these principles to encode information makes it possible for us to tag or name hyperlinks, a practice also known as *typed linking*. Because hyperlinks are not typed or named in the current web, the only way to find related content or know about relationship between a webpage and other related webpages is via follow-the-link method i.e. by physically clicking on hyperlinks and consuming information being presented. But with with RDF, we can know the nature of content and relationship among various pieces of information in advance. In other words, with RDF we can harvest the information (or metadata) about these relationships about knowledge resources to improve search results and discovery of new and relevant knowledge.

RDF Markup and Bibliographic Framework

As mentioned before, the use of RDF makes information understandable and usable for machines. This is because RDF markup requires three basic elements for any information to be described or linked on the web: subject-predicate-object. For this reason metadata records in RDF format are sometimes referred to as *triples*. For example, "an RDF link that connects information about a person with information about publications in a bibliographic database might state that a person is the author of a specific paper" (Bizer et al., 2008, p. 1265). Accordingly a RDF triple or record describing this relationship between a particular author and a book could be stated as illustrated in the figure below (Figure 1).

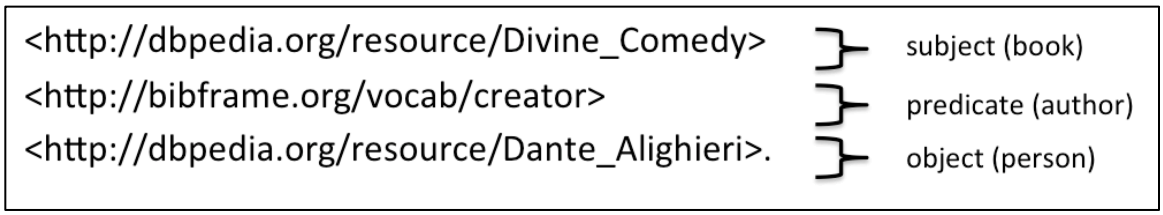

Figure 1: RDF Markup Illustration

The subject and object components are typically two entities or resources that are related. The name and nature of the relationship between them is facilitated through the *predicate* component using standardized and controlled vocabularies. Since library systems (such as catalogues, finding aids, and discovery systems, etc.) are treasure troves of metadata describing content and relationships pertaining to information resources and entities (title, author, publisher, subject, location, etc.), libraries are extremely well positioned to embrace Linked Data and influence the evolution the current web to the *Linked Data Web*. This means that with Linked Data and RDF, the library metadata can now be read, interpreted, and contextualized by machines, opening up a new and exciting automated ways of discovering knowledge resources over the web. Not surprisingly and recognizing this reality, the Library of Congress took the step of introducing Bibliographic Framework or BIBFRAME as a new approach to bibliographic control. In other words, "the BIBFRAME model is the library community's formal entry point for becoming part of a much larger web of data" (Miller, Ogbuji, & MacDougall, 2012, p. 4).

Historically, the FRBR (Functional Requirements for Bibliographic Records) model has been influential in theorizing and communicating the intricacies of bibliographic relationships, but attempts to implement it at libraries have not fared well in practice due to lack of implementation standards and guidelines. In contrast, BIBFRAME is very deliberate in balancing the theoretical and the practical aspects to close this gap. For example, with BIBFRAME, the four layers of intellectual and artistic endeavors as defined by FRBR Group 1 entities (work, expression, manifestation, and item) have been collapsed into just two entities: 1) *Work*, a resource reflecting a conceptual essence of the cataloging item and 2) *Instance*, a resource reflecting an individual, material embodiment of the Work. All other bibliographic authority resources (including people, subjects, organizations, etc.) have been rolled into a single entity called *Authority*. Any other related information pertaining to a resource such as holdings information, reviews, cover art, and alike are grouped under the *Annotation* entity. Thus, BIBFRAME presents a very simple, flattened, and above all, practical view of the bibliographic relationships using only four entities as illustrated in the entity relationship diagram (ERD) below (Figure 2).

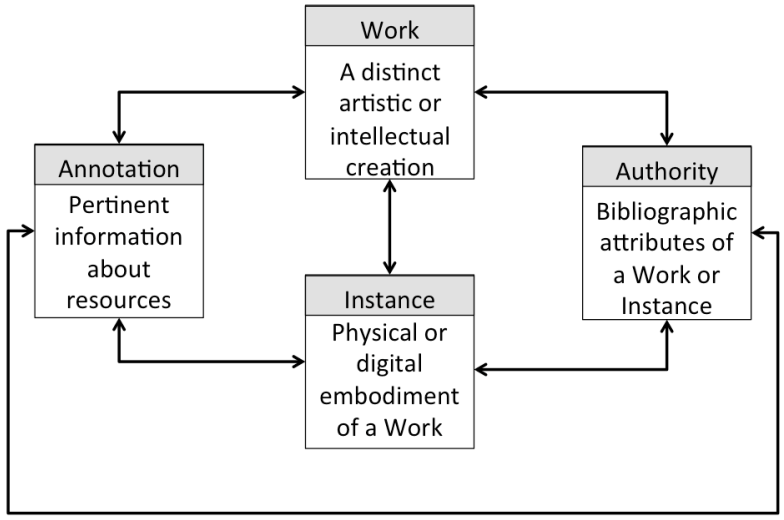

Figure 2: BIBFRAME Entity Relationship Diagram

In the traditional web application development paradigm, having a logical ERD would generally be a trigger for developing a physical relational database model for an application. The functional simplicity of BIBFRAME, however, does not easily permeate to implementation because BIBFRAME was developed specifically to serve as a functional overlay for Linked Data technology as opposed to an application architecture. One of the fundamental premises of Linked Data is that any unit of information (known as *resource*) can be linked to any other resource over the web. What this means for traditional technologists is that the database fields for storing relationships among resources (such as author, date, etc.) are no longer pre-defined in the database. This poses grave challenges in conceiving a database design that allows such a high level of flexibility to store, manage, and database fields. Thus, what Linked Data essentially entails is the separation of entities from their attributes. Furthermore, these attributes, or ontologies in Linked Data terms, are developed, maintained, and shared among various organizations to allow open information exchange.

GIST 3.0 – An example of Linked Data application developed using LAMP

Developing Linked Data applications in libraries challenges traditional web technologists to think differently at every stage of the design and development process, starting from data modeling all the way to presenting data over the web. This was certainly true for the development team responsible for enriching the existing Ginan Index and Search Tool or GIST with Linked Data principles. GIST was initially developed at the University of Saskatchewan Library in collaboration with the Harvard University Library. It was launched in 2013 as a web-based finding aid for a collection of primary sources of *ginans* (devotional hymns) housed at the Harvard University Library. Since its launch, however, the user community made several requests to expand GIST to include ginanic collections from other institutions and sources. Given that GIST was essentially conceived as a web index of works and primary sources, it seemed opportune to revamp the existing LAMP-based GIST application using Linked Data principles. The technical discussion below is informed by the challenges and decisions encountered in transforming the existing GIST application into GIST 3.0 using Linked Data principles implemented in LAMP (Linux Apache MySQL and PHP) technology stack. The LAMP technology stack is the most commonly used platform for open source for web development (Ware, 2002). Since traditional web applications are typically designed and developed using three-tiered architecture (Eckerson, 1995), the technical details of designing and implementing GIST 3.0 is structured using these familiar tiers, namely: database, application, and presentation.

Database Tier

As mentioned before, the promise of Linked Data is that resources such as books and articles, at least in theory, can have infinite number of attributes, which need not be predicted or defined before hand. Since it is not possible or practical to create new database tables or fields every time new relationships and links among resources are identified, this necessitates designing a database structure that facilitates describing, storing, and managing resources and their relationships in very generic terms. Linked Data facilitates linking any piece of data to any other piece of data through *typed linking*, the ability to define and name relationships through predicates using controlled

vocabulary or ontologies rather than by virtue of traditional database fields and tables. The role of ontologies in Linked Data is crucial since ontologies essentially become the attributes that users and administrators use to define their resources rather than the database fields that a technologist may have created in traditional design. How can we design such a database? In traditional database design, entities of interest such as books, authors, and alike often end up as tables with specific attributes to describe them. In the Linked Data realm, all entities of interest are known as resources and no distinction is made among these resources other than the unique URIs they possess. Thus the database tables of interest for Linked Data applications such as GIST 3.0 may include *resources, namespaces, predicates, and triples*.

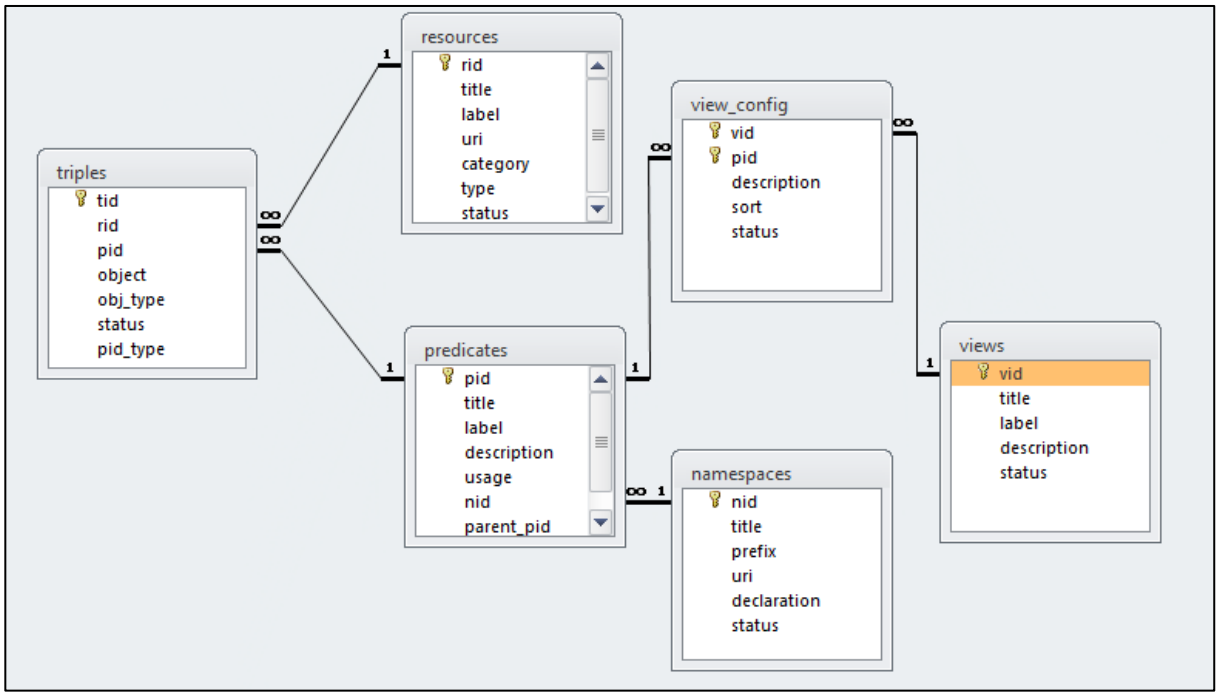

Figure 3: Database design for implementing GIST 3.0

When storing works or units of information in the *resources* table, the table itself should have a few administrative fields for distinguishing and grouping them for presentation, which will be discussed below. Any data stored in MySQL database is not automatically shared as Linked Data. In order to share data as Linked Data, appropriate RDF triples must be explicitly defined and stored in the *triples* table. Attributes that define relationships among resources of interest by way of typed linking are stored in the *predicates* table. These attributes or predicates typically belong to a particular set of vocabulary or *namespace*. Much like resources, all predicates in Linked Data must also have their own unique and accessible URIs leading to their descriptions. For instance, the namespace for BIBFRAME vocabulary is declared and retrieved using http://bibframe.org/vocab/ and a predicate within BIBFRAME such as "creator" can be dereferenced (i.e. made accessible) using http://bibframe.org/vocab/creator. It is advisable to use existing predicates from known and published vocabularies rather than

building new or custom vocabulary to avoid duplication and redundant typed links. Some of the common and popular Linked Data vocabularies or ontologies include Web Ontology Language (OWL), Friend of a Friend (FOAF), etc.

At times it may also be necessary to embed certain attributes in a context of another attribute. A typical example is the imprint information of a book, which is a combination of multiple attributes like publisher name, publication year, city, etc. Creating a parent_pid field in the *predicates* table and populating it with the appropriate identifiers, is one way of accommodating this functionality. Depending on which RDF serialization method is chosen, nested predicates may become an important consideration for generating triples as well as presenting data over the web. Nested RDF serialization formats such as RDF/XML are generally considered more human-friendly than some of the others formats such as n-triples. Once the database design is implemented in MySQL, the next step is to populate it with the necessary data from library source systems such as catalogues, digital repositories, finding aids, etc.

Application Tier

In the library realm, MARC (MAchine-Readable Cataloging) encoding standard was developed to make bibliographic information electronic and machine processible, which transformed library cataloguing praxis of the day. Since then, MARC has been widely accepted as metadata standard. Over half a century later, the needs and demands of the cataloguing world have changed such that MARC is now quickly becoming a liability for libraries. The initial challenges faced by many libraries to ingest and share catalogue records for electronic resources can, to some extent, be attributed to constraints imposed by past library practices and standards steeped in managing physical resources. Much like MARC, RDF is also a machine-readable encoding standard. This is a result of concerted effort to realize the fundamental goal of Linked Data to make information on the web machine-processible.

In order to comply with the requirement that Linked Data be machine and human friendly, it is necessary for a Linked Data application to be able to detect if the incoming data requests to access resources are from humans by way of web browsers or machines that host web crawlers or run automated indexers. This helps in deciding if the requested resource needs to be formatted for browsers or simply served as raw RDF. Since Linked Data principles embrace HTTP, an existing mechanism called *content negotiation* for detecting incoming requests was used for GIST 3.0. With this mechanism, HTTP headers are inspected to ascertain if the requesting client prefers data in HTML or not. In the LAMP technology paradigm, the logic to detect the type of request could be implemented using the directory-level configuration file (aka .htaccess file) on Apache. There are ways to simulate non-browser requests in order to test if the detection and delegation logic implemented through the above mechanism is effective. One of the ways to do so is to use the Linked Data validator known as Vapour. When given a URI, Vapour simulates HTML and RDF requests and presents a visual report and underlying details of its requests. Alternatively, cURL can be used for testing content negotiation in a command-line environment as follows:

```
curl -I -H "Accept: application/rdf+xml" "http://dbpedia.org/resource/Asturias"
```

Once the desired content for the request is detected, appropriate application logic can be put in place to fetch and format requested data. For example, in the case of DBPedia (which is the Linked Data version of Wikipedia) requests for a resource from web-browsers (which are indicative of human users) are redirected to a different path than the requests originating from non-browser ones (which are indicative of machines such as web bots and crawlers). This simple and effective mechanism to have a logical distinction between "page" for humans and "data" for machines in DBPedia was also deployed for GIST 3.0.

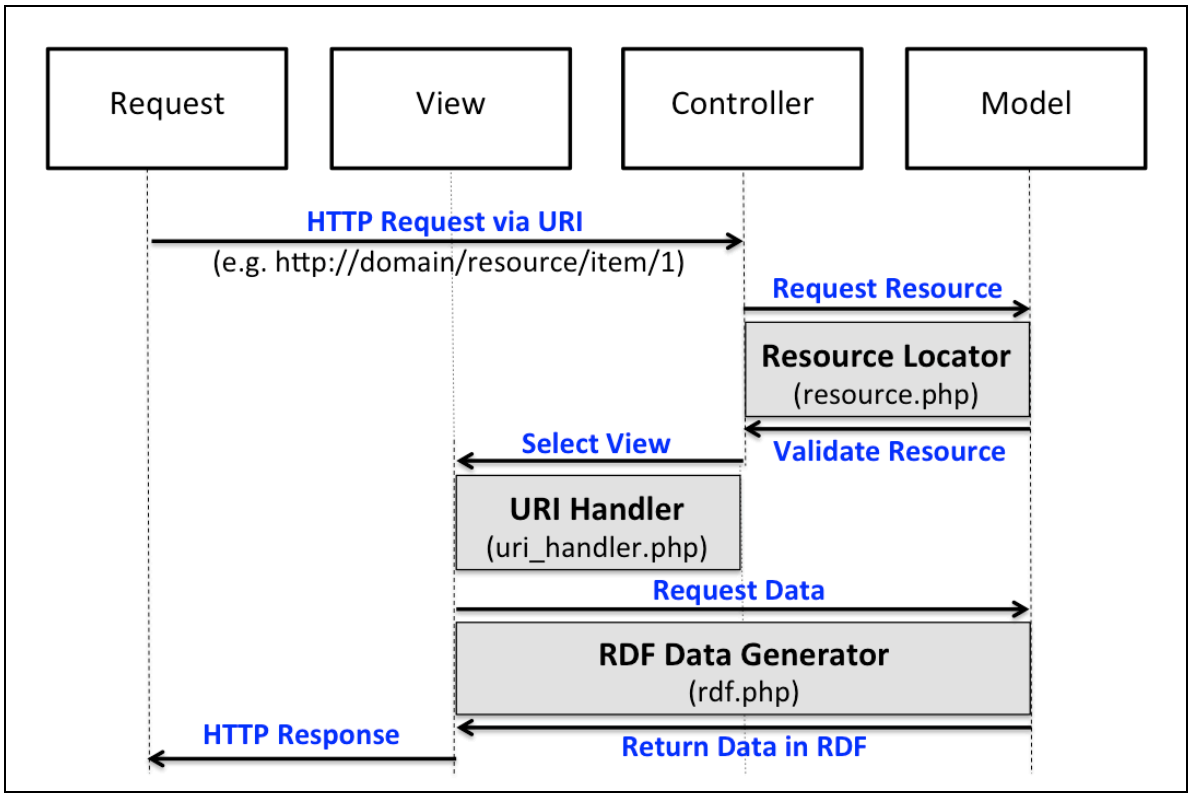

Figure 4: GIST 3.0 Application control flow in MVC framework

We can use the Model View Controller (or MVC) pattern (Leff & Rayfield 2001) as a framework for understanding how the application logic has been implemented for GIST 3.0 using LAMP. Figure 4 depicts the control flow of GIST 3.0 implemented using PHP. The Controller receives incoming HTTP requests in the form of URIs, which are translated into action and forwarded to the Model. The Model ascertains the validity of the requested resource and returns the result back to the Controller. Upon hearing back from the Model, the Controller selects the appropriate path to engage the View tier, which is responsible for communicating and presenting the results stemming from the request. The View layer, in turn, has the ability to connect with the Model directly to fetch the data it needs to fulfill the request it received from the Controller. In the Linked Data context, this is where the triples are generated.

Presentation Tier

As is the case in many bibliographic applications, there is typically *list* and *detail* views of resources. In GIST 3.0, the list view is used to present available resources with links to corresponding detail views. The detail view essentially presents a profile of the desired resource such as an organization or a collection. For ease of navigation, a link to return to the list view is also provided on all detail views in GIST 3.0 (see Figure 5). Just as the number and content of webpages are pre-determined in traditional web applications, the presentation views for Linked Data applications must also be determined beforehand based on the purpose of applications and user needs.

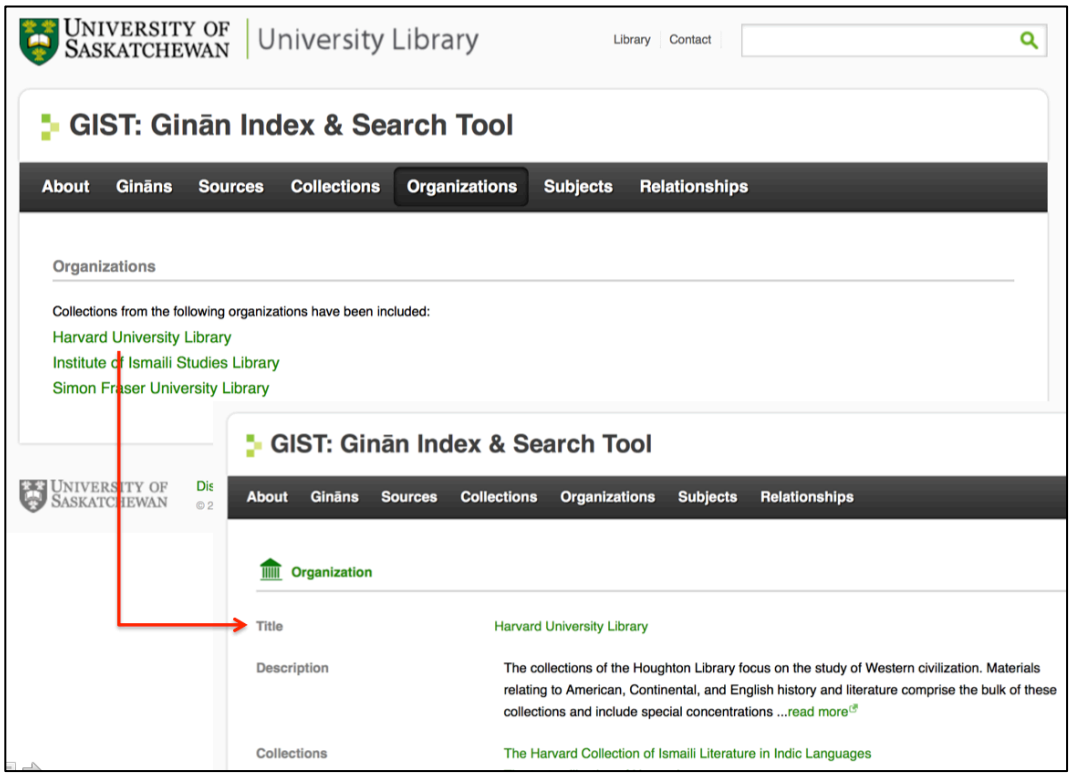

Figure 5 – Summary and Detail views in GIST 3.0

The GIST 3.0 application also uses two tables —*views* and *view_config*— to manage data and presentation for various views. The *view* table is used to identify and enlist views of desired resources. The *view_config*, in turn, table is used to identify and associate relevant predicates with each view. For example, one of the views in GIST 3.0 is the *organizations* view in GIST 3.0 that lists organizations whose collections have been included and indexed (see Figure 5). This view must be identified in the *views* table and given a unique identifier. In the *view_config* table, relevant predicates associated with the organizations view are also identified and stored (see Figure 6). The data returned from the database for a chosen view is then used to serialize RDF triples in a preferred format (such as Turtle, N-Triples, RDF/MXL, etc.). The choice of format also impacts the level sophistication needed for the RDF serialization program. A human-friendly format requires nested nodes and attributes than others. There are also free utilities (from W3C, EasyRDF, etc.) that provide conversion and crosswalk between these formats.

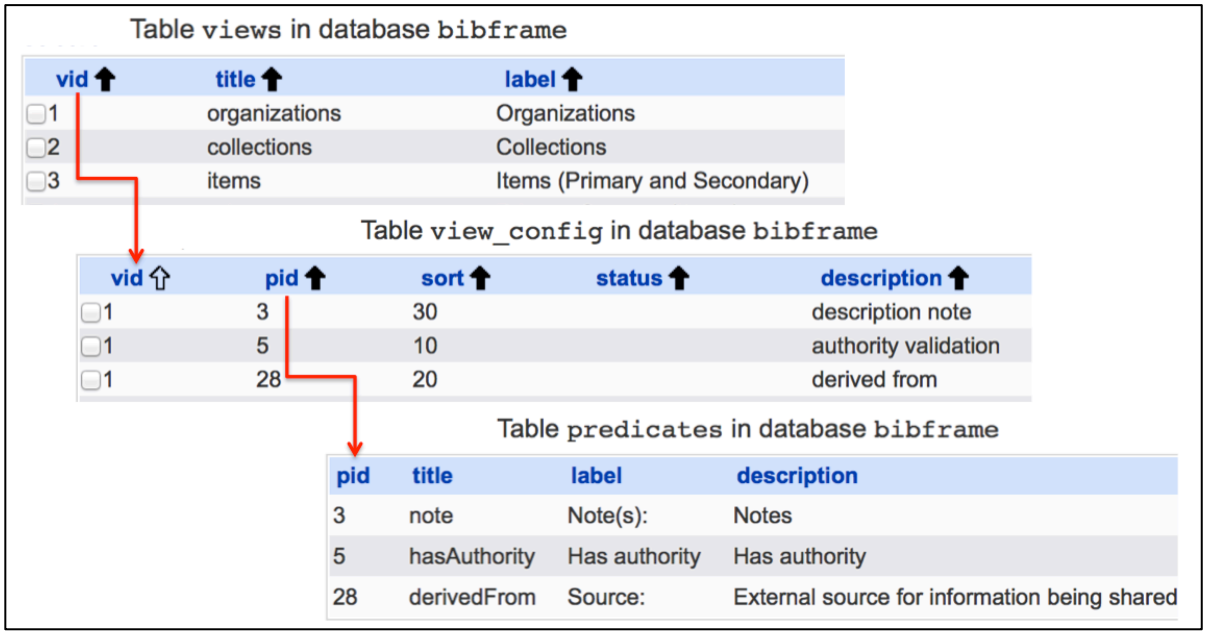

Figure 6: Administrative tables used for presentation in GIST 3.0

In terms of the application control flow, the only difference between serving requests from web bowsers (for humans) versus non-browsers (for machines) is that for humans data are presented and formatted in HTML using stylesheets before being sent back to browsers. For all non-browser requests, the data are presented in raw RDF format. If HTML formatting is needed, appropriate stylesheets are used for generating the desired view. Since the raw data for GIST 3.0 are in RDF/XML format, the XSL stylesheets used for formatting RDF make extensive use of XSLT to parse and display data, which is a programming language that may also challenge traditional PHP programmers. The merging of stylesheet (presentation) with XML (data), known as transformation, is achieved with the XSLTProcessor suite of built-in procedures in PHP.

In practice, when the data have stabilized and new data are not being added to the application, it may also be possible to present data using static data files. One advantage of static data files is that they save database hits and processing cycles for generating underlying triples on-demand. In the GIST 3.0 beta version, since the application data set was relatively small, it was possible to generate a single file with all the necessary triples to serve various views. The GIST 3.0 data file is periodically refreshed to keep the static file current and in synch with the database. Another advantage of having static files is that they make the entire application portable and resilient to network unavailability. In fact with little adjustment, the entire application could potentially be run on a local machine in a client mode.

Conclusion

With Linked Data and use of RDF, the *web of data* is not only readable for humans but also actionable for machines. This awareness of data by machines makes it possible to transform the current web into a global *database* capable of answering sophisticated queries without the need of any proprietary software or skills. From a practical point of view, use of Linked Data enables information professionals to be ready to integrate, consume and share metadata across many sources and in various formats while decreasing duplicate data and effort. From a philosophical point of view, the promise behind Linked Data resonates with the fundamental mission of the academia and libraries to make hidden knowledge known and accessible based on open standards. The Linked Data movement is gaining attention and momentum in academia, and it is imperative for web technologists to start getting involved and experimenting with Linked Data from an early stage. This article is based on our initial experimentation with Linked Data at the University of Saskatchewan Library using existing library infrastructure and skills, a necessary starting point to be able to embrace and rationalize adopting new and emerging technologies.

Acknowledgements
University of Saskatchewan's President's SSHRC program provided the funding for this project. The author gratefully acknowledges contributions of Faham Negini in his capacity as the project's graduate research assistant.

About the author
Karim Tharani is a tenured faculty and Head of the Library Systems and Information Technology unit at the University Library, University of Saskatchewan in Canada. His research interests include digital libraries, digital access and preservation, digital humanities, metadata harvesting, and use of digital technologies in collaborative and community contexts.

References

Berners-Lee, T. (2011). Linked data-design issues (2006). *Retrieved from http://www.w3.org/DesignIssues/LinkedData.html*.

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. *International journal on semantic web and information systems*, *5*(3), 1–22.

Bizer, C., Heath, T., Idehen, K., & Berners-Lee, T. (2008, April). Linked data on the web (LDOW2008). In *Proceedings of the 17th international conference on World Wide Web* (pp. 1265–1266). ACM.

Byrne, G., & Goddard, L. (2010). The strongest link: Libraries and Linked Data. *D-Lib Magazine*, *16*(11/12). Retrieved from http://www.dlib.org/dlib/november10/ byrne/11byrne.html

Eckerson, W. W. (1995). Three tier client/server architectures: achieving scalability, performance, and efficiency in client/server applications. *Open Information Systems*, *3*(20), 46–50.

Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International* (pp. 118–127). IEEE.

Miller, E., Ogbuji, U., Mueller, V., & MacDougall, K. (2012). Bibliographic Framework as a Web of Data: Linked Data Model and Supporting Services. In *Washington, DC: Library of Congress*. Retrieved from http://www.loc.gov/bibframe/pdf/marcld-report-11-21-2012.pdf.

Shadbolt, N., Hall, W., & Berners-Lee, T. (2006). The semantic web revisited. *Intelligent Systems, IEEE*, *21*(3), 96–101.

Ware, B. (2002). *Open Source Development with LAMP: Using Linux, Apache, MySQL and PHP*. Addison-Wesley Longman Publishing Co., Inc.