Boca: an open-source RDF store for building Semantic Web applications

Lee Feigenbaum, Sean Martin, Matthew N. Roy, Benjamin Szekely and Wing C. Yung

Received (in revised form): 23rd February 2007; Accepted: 6th April 2007

Abstract
This article presents the design goals and features of the open-source Boca RDF server in the context of a community of cancer-tumor modeling investigators. Boca supplements the desirable data features of the Semantic Web with important enterprise and application features to power a new generation of Semantic-Web-based applications. The data features enable the integration and retrieval of tremendous quantities of diverse data. The enterprise features promote data integrity, fidelity, provenance and robustness. The application features provide for collaborative applications and dynamic user interfaces.

Keywords: Semantic Web; data integration; open source; database; modeling; RDF

For many years, the Semantic Web has been little more than a dream. It began as a vision put forth by World-Wide-Web creator Tim Berners-Lee and was built as a collection of young and untested technology standards. To biologists, in particular, the Semantic Web has promised—but not yet delivered—solutions to challenges of massive data integration, structured and directed search and knowledge management. In the past 2 years, however, this situation has begun to change. The standards have matured, the catalog of available software tools and infrastructure has exploded, and the focus has shifted dramatically to the applications: What specific problems can we solve with Semantic Web technologies, and how can we build the applications to solve these problems? The Boca RDF server has been designed and constructed with applications as a first priority.

Semantic Web stores derive their value from the flexibility and expressivity that come from working with semantic content. There are many existing storage systems providing RDF storage, including Hewlett-Packard’s Jena [1], Aduna Software’s OpenRDF [2], Redland [3] and Oracle 10 g [4]. These stores are used to store and query semantic data. The stores typically boast the ability to work with large amounts of data and feature highly optimized engines to answer queries in milliseconds. But a scalable and well performing semantic data store is only the first step towards the ability to realize the full potential of Semantic Web applications. On its own, a store does not address the challenges of multiple users with differing levels of access, collaborative user experiences or offline application access, all of which are required for adoption by users in many industries, including the life sciences. Boca goes beyond traditional semantic stores by providing application developers with these features and several other benefits, paving the way for the next generation of semantically enhanced software.

Corresponding author. Lee Feigenbaum, 1 Rogers St., Floor 3N, Cambridge, MA 02142, USA. Tel: 1-617-894-6223, E-mail: feigenbl@us.ibm.com

All authors work for IBM’s Internet Technology Group in Cambridge, MA. The team currently focuses on researching and developing strategies and software for leveraging Semantic-Web technologies within enterprises.

Lee Feigenbaum is a software engineer with the team and maintains a technical blog at http://thefigtrees.net/lee/blog/. Lee is the chair of the W3C’s RDF Data Access Working Group, chartered to deliver standards for the SPARQL protocol and query language.

Sean Martin is a Senior Technical Staff Member with the team which he represents on the W3C’s Health Care and Life Sciences interest group.

Matthew Roy is a software engineer with the team and is Boca’s lead developer.

Benjamin Szekely is a software engineer with the team and is the author of the popular Jastor, an RDF-to-Java code generator that uses OWL and RDF Schema.

Wing Yung is a software engineer with the team. He is a member of the Semantic Web Education and Outreach Interest Group. He blogs about the Semantic Web and Web on his blog, http://tech.wingerz.com.
Boca is the cornerstone component of the IBM Semantic Layered Research Platform [5], an open-source collection of Semantic Web software infrastructure licensed under the Eclipse Public License [6]. The design principles that underlie Boca rest on three foundational pillars:

(i) **Data features.** Boca acknowledges the paramount importance of having a traditional RDF repository as the core of the system. As such, Boca supports storing, retrieving and querying RDF data.

(ii) **Enterprise features.** Boca surrounds its core features in an environment dedicated to maintaining the security, history and fidelity of its stored data. User authentication and authorization, content versioning, optimistic concurrency and transactional atomicity are all features that serve these ends.

(iii) **Application features.** Boca provides a comprehensive client programming model for developing applications backed by semantic content. This model includes server-initiated notifications of data updates, batched two-way replication for synchronizing offline changes and cached local storage of information for performance and offline use.

**COLLABORATIVE TUMOR MODELING**

The requirements and design for the Boca data store—and indeed all the components of the IBM Semantic Layered Research Platform—are driven by the needs of actual use cases. Perhaps no use case has been more influential than the problems presented by the nascent Center for the Development of a Virtual Tumor (CViT) [7], a program sponsored by the National Cancer Institute. CViT is an emerging community of *in-silico* cancer modelers whose end goal is to develop a data- and software-driven module-based toolkit for modeling and simulating multi-scale cancer-tumor growth dynamics.

CViT is an online shared virtual space which brings together multi-institutional, interdisciplinary teams of investigators to form a global collaborative community. Scientific discourse, research findings, experimental data, and, especially, a catalog of the many and varied models under separate and—in the future—joint development, can all be shared within this community. Currently hundreds of investigators representing more than 60 institutions around the world belong to CViT’s membership.

The technical requirements for a system that can support the applications required in such an effort are complex. In particular the problems of data and application integration in the face of constantly evolving scientific understanding and shifting concepts that need to be represented present a challenge for current software development technologies that are geared to creating systems that favor far less dynamic data models. Current software development practice often leads to systems that are not capable of capturing and storing the most recent developments in a scientific field. Instead, new data structures must be invented and new forms must be created to view and edit the new data. By the time the application programmers have caught up with this new information and deployed a new release to handle it, something newer will have taken its place. The ongoing expense of continually altering these systems and the time it takes to do so are both serious problems.

The model repository catalog component of CViT, to be deployed later this year, is a good example of the problem as the approaches and technologies used by all the different investigative groups to do their individual modeling, as well as the data sets employed by each, vary widely and can change or be expanded rapidly over time especially as more sophisticated models evolve. CViT relies on Boca to provide an extremely flexible approach to storage, viewing and query of the individual model information, so that the system is capable of capturing everything necessary to describe and share diverse models.

**DATA FEATURES—THE CORE OF THE SEMANTIC WEB**

Semantic Web standards provide a starting point for addressing the requirements for CViT. Familiarity with these standards and their benefits is a prerequisite to understanding some of Boca’s more advanced features. In particular, the Resource Description Framework (RDF) is the foundation Semantic Web data standard [8]. RDF is a flexible and expressive graph-based data model whose properties make it a compelling choice for modeling data with rapidly changing structures, for integrating data from multiple sources, and for accessing data in a powerful and uniform manner. One of the primary strengths of RDF is that it can be used to represent nearly any kind of information and it is therefore well suited to
become the *lingua franca* for data communication across applications and computing platforms.

RDF data forms a labeled, directed graph. An RDF statement, also known as a triple, consists of a subject, predicate and an object (much like the subject, verb and object of a simple English sentence). Each statement corresponds to a directed edge in the graph. Nodes and edges in the graph have globally unique identifiers which are often resolvable and human readable.

The technology standards supported by RDF stores, in general, and by Boca, in particular, have several appealing properties for applications constructed using them:

**Rapidly changing, highly connected data structures**

While some structure can be imposed on RDF data via the use of ontologies (specified in the Web Ontology Language (OWL) [9]), data is not invalidated if it contains properties that do not conform to an ontology. User interfaces, queries and data processors can be constructed to minimize or eliminate the effort required to maintain code when the underlying data structures change. For example, CViT contains many dynamic relationships among research institutions, people, biological models and other objects. Also, the inputs to and behavior of biological models can change quickly as new discoveries are made. For both these reasons, it is impossible to *apriori* design a comprehensive and robust data storage solution for CViT; the flexibility of RDF data structures allows the CViT application to evolve as investigators’ knowledge and hypotheses grow and change.

**Data integration**

Because RDF statements are grouped as graphs with no set order, merging RDF data from multiple sources is a trivial operation. The use of globally unique identifiers prevents accidental naming clashes between different concepts, ensuring that two occurrences of the same identifier refer unambiguously to the same concept. Additionally, these identifiers are decentralized, and OWL provides a mechanism for asserting that two independently derived identifiers refer to the same underlying concept after the fact, further facilitating the data integration process.

As noted earlier, RDF is expressive enough to make it possible to map other data models and formats to it. Today, data from multiple sources can be difficult to integrate because existing data models and formats like XML and relational databases have a rigid structure that does not easily allow their data to be taken out of context and reused. The Semantic Web standards allow the semantics or meaning of information to be stored with the information itself (not relying on outside descriptions to explain the relationship between data and its position in a file) making it easier to reuse as its applications come to rely more on the meaning of the data than its structure. In addition, RDF can be created dynamically in response to data access calls so there is generally no need to convert existing data to be stored natively in RDF. There are several projects that expose relational databases, LDAP directories, and XML data as RDF [10, 11].

**Data access**

SPARQL is the standard query language of the Semantic Web [12]. Designed from the ground up to be capable of simultaneously querying and integrating the results from multiple data sources scattered across the network, SPARQL allows clients to choose exactly the data that they want, giving them great flexibility for data access. Traditional data access APIs are limited and return proprietary structures, if they exist at all, trapping data in a single place or making it cumbersome to retrieve. Boca’s SPARQL engine, Glitter, includes a pluggable data layer that can be extended to query over both native RDF data stores and many other forms of databases in a single operation. Another means of querying the information in Boca is provided by the Sleuth component, a text search engine based on Apache Lucene [13]. Sleuth is integrated with Glitter to allow text based queries to be run against all the metadata in the Boca store as part of a SPARQL query. Results of the text query can then be filtered against semantic information and, of course, access controls. The combination of Glitter and Sleuth allows for the creation of queries that answer questions like: Find all models of brain-cancer-tumor growth that were published in the last five years in articles that mention ‘invasion’.

**ENTERPRISE FEATURES—PROTECTING YOUR INFORMATION**

Boca differentiates itself from other RDF repositories by including important enterprise features for
application development, including named graph support and access control.

Dealing with individual RDF statements can be cumbersome, and dealing with the contents of the entire store as one logical graph can be overwhelming. For this reason, Boca partitions RDF statements into graphs with their own identifiers, called named graphs [14], providing an additional layer of abstraction above RDF. This makes it possible to separate data into logical components for higher-level operations like restricting query scope, applying access controls, and tracking data changes from one version to the next.

CViT is a collaborative application, meaning that many users read, write, and share the data stored in system. Access control is a key requirement of any such shared data repository. CViT investigators must have the ability to ensure the privacy of their current experiments and hypotheses before later sharing interesting findings with selected collaborators and eventually the entire community. Boca applies role-based access controls to named graphs, allowing application architects and then end users to determine how to partition data appropriately. In a role-based system, access levels are not granted to individual users; instead they are granted to a set of roles. Users’ actions are restricted by the roles to which they belong [15]. This provides the ability to create anything from a simple role that contains a single person up to a hierarchy of roles that describes the various groups and subgroups that make up a complex organization. For each named graph in a Boca server, a particular role can specify four permissions: reading data within the graph, adding triples to the graph, deleting triples from the graph and modifying access control for the graph.

Federal regulations often require medical and pharmaceutical researchers to maintain auditable data histories. Experimental results must be accompanied by the full provenance of the data, including when additions and changes were made, who made them, and what the changes were. Boca provides version control of named graphs. When a named graph is changed, its revision number is incremented and Boca logs the user who initiated the change and the time. A user with the proper access control can retrieve any of the previous revisions of a named graph, along with any of the logged provenance metadata.

Boca maintains data integrity in a multi-user environment in several ways. First, changes to named graphs can be grouped into atomic transactions; this prevents failed operations from leaving the data store in an inconsistent state. Second, Boca provides the ability to set preconditions on transactions. Preconditions express assertions about the expected state of the data on which the transaction is operating; a transaction with a failed precondition is not executed. The most common use of preconditions is optimistic concurrency: By asserting that a named graph is at a specific revision, conflicting modifications of a specific named graph are prevented.

THE BOCA CLIENT—BRINGING COLLABORATION AND FLEXIBILITY TO THE USER EXPERIENCE

Applications such as CViT require more complex building blocks than the ones we have already covered. Research applications require local storage of working sets of data in order to improve overall application performance, provide an offline mode and publish data only when ready. For this reason, Boca includes a client component that supports local RDF storage. Local data sets might comprise one or more complete or partial named graphs. The Boca client communicates with the server via two avenues: replication and notification.

Boca transactions are composed of a set of RDF statement additions and deletions. Replication allows applications to delay sending transactions to the Boca server, allowing users in the meantime to undo and redo the operations that comprise the transactions. When the client replicates, the transactions are sent from the client to the server. Replication also retrieves updates performed by other users of the same Boca server. Depending on the nature of the application, replication may be scheduled to be performed at regular intervals or only in response to user-initiated actions, such as invoking a ‘save’ command.

While replication allows changes to be sent to the server in batch and ensures that the local view of data is periodically synchronized with the server, true collaborative applications require real-time updates. Boca provides a lightweight mechanism for receiving real-time update notifications. As with replication, applications choose specific named graphs of interest to track via notification and updates received via notification can be persisted in a client’s local data set storage.
The notification system operates atop a subscription-based messaging system which sends events to the application when statements are added or removed from tracked named graphs. In a Web application using the Boca server for persistent storage, these events might be used to invalidate entries in a distributed Web object cache. In another example application, notification can drive distributed workflow management. CViT modelers analyze microscope images and simulate tumor growth. As experiments run and generate new RDF metadata about their progress, agents listening for notifications can coordinate operations in a distributed fashion, using Boca as the communication bus.

In a collaborative application like CViT, these features are extremely important. For example, two researchers may collaborate to analyze a shared set of laboratory data. As one researcher makes modifications, the other researcher is aware of the changes and can work to minimize conflicts when they both replicate their changes back to the central data store.

The Boca client provides features to accommodate a wide variety of software including self-contained rich-client applications, data access plug-ins to existing applications such as spreadsheets and a full spectrum of Web applications. Boca application developers program against a simple named graph abstraction, needing only to conceptualize the location of data within named graphs and the appropriate transaction boundaries. This diversity is made possible by Boca’s two primary modes of operation: the embedded and Web Service (remote) paradigms.

The Boca programming model was designed to expose semantics to every layer of the application stack. It is very flexible, as Boca’s design has evolved alongside CViT, which started as a rich-client application backed by Boca. This motivated the development of the Boca client stack’s local storage, replication and notification features. The most recent version of CViT is Web 2.0-style application with rich semantics exposed through to the web client. This new style of application has motivated the need for a web application programming model for Boca, beyond the embedded client running on the web application server.

**CONCLUSION**

While the pervasive nature of digital content has transformed our lives in general and has given birth to the entire field of bioinformatics in particular, we remain constrained by the various silos that dominate our technological experiences. Applications manipulate information in proprietary formats, preventing other applications from accessing the same content. We are satisfied with visualization and analysis tools that can read data from an underlying store but are unable to write changes to that same data. And content creates silos as well: scientific journal articles imprison experimental data, results and references within an opaque stream of text, figures and tables that cannot easily be queried, expanded upon, or related to other data.

The Semantic Web is now ready to begin to change this. By focusing on the semantic relationships between information, RDF stores break down the content silo and distill data into interlinked graphs of information that express its essence, priming it for easy integration, analysis and reuse. Furthermore, by providing an extensive feature array targeted at application development, the Boca RDF server breaks down the application silo as well, enabling a wide variety of applications to gain first-class access to shared data that retains its semantic meaning throughout the application stack. As the first applications of Boca demonstrate, it gives all the applications it supports the capability to retrieve and react to collaborative semantic models in a secure, trustworthy, and robust environment. By placing the focus squarely on the new generation of semantically aware applications, Boca provides us with the tools we need to finally begin to realize the promises of the Semantic Web in bioinformatics, life sciences and beyond.

**Key Points**

- Applications constructed on Semantic Web data, ontology and query standards can flexibly integrate disparate data from multiple sources and make the data freely available for consumption by other applications.
- Boca is a multi-user semantic store with a named graph-based access control scheme.
- Boca’s automated versioning of named graphs provides a full audit trail containing the provenance of all data changes in the system.
- Boca’s notification and replication systems promote distributed, collaborative applications.

**References**


