# Scientific data as RDF with Arrays: Tight integration of SciSPARQL queries into MATLAB

Andrej Andrejev[1], Xueming He[1], Tore Risch[1]
[1] Uppsala DataBase Laboratory (UDBL), Department of Information Technology,
Uppsala University, Box 337, SE-751 05 Uppsala, Sweden.
{Andrej.Andrejev, Tore.Risch}@it.uu.se, emilyhexueming@hotmail.com

**Abstract.** We present an integrated solution for storing and querying scientific data and metadata, using MATLAB environment as client front-end and our prototype DBMS on the server. We use RDF for experiment metadata, and numeric arrays for the rest. Our extension of SPARQL supports array operations and extensibility with foreign functions.

## 1 Introduction

In many branches of science and engineering, researchers accumulate large amounts of experimental data [3,4] and use widely recognized (de-facto standard) libraries of algorithms to analyze and refine that data. Tools such as MATLAB or similar serve as integrated environments that provide basic file management, extensibility with algorithmic libraries, visualization and debugging tools, and are generally oriented towards single-user scenario.

What is typically missing is the infrastructure for storing the descriptions of experiments, including parameters, terminology mappings, provenance records and other kinds of metadata. At best, this information is stored in a set of variables in the same files that contain large numeric arrays of experimental data, and thus is prone to duplication and hard to update. We have addressed this problem in our previous work [2] utilizing the Semantic Web approach for storing both data and metadata, and using Scientific SPARQL query language [1], that extends SPARQL queries with numeric array operations and external user-defined functions. The goal of SciSPARQL is to provide uniform query access to both metadata about the experiments and the massive experimental data itself, as illustrated by table 1. Section 3 gives more detailed account of SciSPARQL features.

SciSPARQL is supported by our software prototype - SSDM (Scientific SPARQL Database Manager [1,2]), a database management system (DBMS) for storing and querying data originating from scientific experiments. SSDM provides scalable storage representation of RDF and numeric multidimensional arrays.

**Table 1.** Comparison of data processing domains of MATLAB, SPARQL and SciSPARQL

| | MATLAB | SPARQL | SciSPARQL |
|---|---|---|---|
| Metadata | | √ | √ |
| Scientific data including Arrays | √ | | √ |

In this work, we demonstrate a client-server architecture featuring (i) **SSDM server**: the centralized storage for both experiment metadata (as RDF) and arrays stored in binary files linked from the RDF dataset and (ii) **MSL**: a MATLAB extension that allows to establish connections to SSDM server, run SPARQL queries and updates directly from MATLAB interpreter, and access the query result sets.

We show that the data is shipped from the server only on demand. Also, the conversion of numeric array data between native MATLAB format and internal SSDM representation only takes place if non-MATLAB function going to access the array, or, more typically, a certain range within the array.

For this demo[1] we have deployed SSDM server on a Linux machine to store RDF datasets in-memory and array data in binary .mat files [5], which is currently a de-facto standard. (This provides the same speed for reading and processing array data as it would be while using MATLAB alone) The demo script is run on the client machine inside MATLAB interpreter.

## 2 MATLAB-SciSPARQL Link

The extension to MATLAB includes two main classes: *Connection* and *Scan*, and additional classes used to represent RDF types on MATLAB client side, e.g. URIs and typed literals. An additional class MatProxy is used to represent (on the client side) an array stored in a .mat file on the server.

*Connection* encapsulates a connection to SSDM server, including methods for

- executing SciSPARQL queries and obtaining a result as a *Scan,*
- executing non-query SciSPARQL statements, e.g. updates and function definitions, apart from inserting RDF triples into the dataset on the server
- defining URI prefixes to be used both on client and server side,
- shipping MATLAB arrays from client to the server,
- managing data persistence on the server.

*Scan* encapsulates a result set of the query. The data is not physically retrieved, stored or shipped anywhere before it is explicitly accessed as a row in the scan. *Scan* includes methods for iterating through the result sets of SciSPARQL queries: the arrays and scalar numbers become represented by MATLAB arrays and numbers, other RDF values get represented by the wrapper objects defined in MSL.

As we show in the demo, the user can easily create MATLAB routines to convert (partially or entirely) the data from the *Scan* into the desired representation, e.g. for visualization.

## 3 Scientific SPARQL

We have extended SPARQL language to query and update RDF datasets extended with arrays. SciSPARQL [1] includes

- extensions for declaratively specifying element access, slicing, projection and transposition operations over numeric arrays of arbitrary dimensionality,
- a library of array aggregation functions, that are performed on the server in order to reduce the amount of data shipped to the client,
- extensibility with user-defined foreign functions, allowing to make use of existing computational libraries.

[1] The demo script is available at http://www.it.uu.se/research/group/udbl/SciSPARQL/demo3/

SciSPARQL is designed to handle both metadata (stored or viewed as RDF) and large numeric data to be accessed in the uniform way: by the same query, from the same dataset.

One important feature of SciSPARQL is ability to define *SciSPARQL functional views*, essentially, the named parameterized queries (or, similarly, updates). These can be used in other queries, or called directly from MATLAB client with parameters provided as MATLAB values. The conversion of values from MATLAB to RDF is performed automatically on the client.

## 4 SSDM Server and Array Proxy Objects

Scientific SPARQL Database Manager is designed for storing RDF data and numeric multidimensional arrays, working either as in-memory DBMS, or with a help of SQL-based [2], or any other interfaced back-end storage. In this demo SSDM server is configured to store RDF triples in-memory, and array data as managed directory of native .mat files. Reading and writing .MAT files on the server side is done via freely distributed MATLAB MCR libraries.

To save a snapshot of RDF dataset linking to the arrays stored in .mat files, `save()` SciSPARQL directive can be sent via the connection. The server can be restarted with a named image, and continue to function as in-memory DBMS.

The main purpose of SSDM server is to process SciSPARQL queries and updates. As part of an update, a `store()` function can be called from the client. A MATLAB value (e.g. numeric multidimensional array) will be shipped to the server as a binary .mat file, and saved under server-managed name in the server file system. The *Array Proxy* object pointing to the value in that .mat file will be returned to the client, and used as a replacement for the actual array e.g. as a parameter to SciSPARQL queries and updates. Once stored in RDF dataset, *Array Proxy* serves as a link from metadata RDF graph to the numeric data stored externally in a .mat file.

If the file is already on the server, and its location is known (maybe, due to some convention among the users), an alternative `link()` function can be used to obtain an equivalent *Array Proxy* object.

When SciSPARQL query involves slicing, element access, projection or array aggregate operations on an array represented by *Array Proxy*, the SSDM server reads the specified part of the array stored in file into SSDM internal array representation (thus performing slicing, projection or element access), does any further processing (e.g. applying array aggregate functions, like "sum of all columns"), and ships the resulting, typically, much smaller array to MATLAB client, where it is converted back to MATLAB representation. It is also possible to do slicing and projection operations within the native .mat array representation, when no further processing by SSDM is planned.

One of the possible workflows involving arrays is shown on Fig. 1-2. First, a MATLAB array *A* is created on the client. A call to `store()` function ships it to the server and returns an Array Proxy object. This object is used in RDF triples sent to SSDM while populating RDF graph describing the experiment.

At the query phase (Fig. 2), a subset of *A* (that is now stored on the server in a .mat file) is selected, fed to `array_sum()` aggregate function, and the result (a single number) is shipped back to the client for post-processing and visualization.

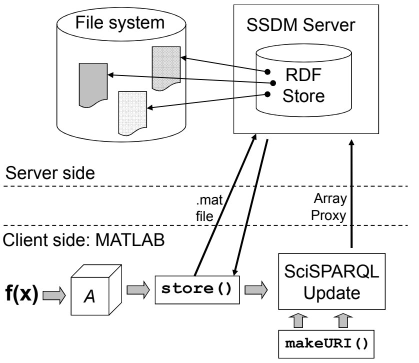

**Fig. 1.** Storing client-generated data and metadata on SSDM server.

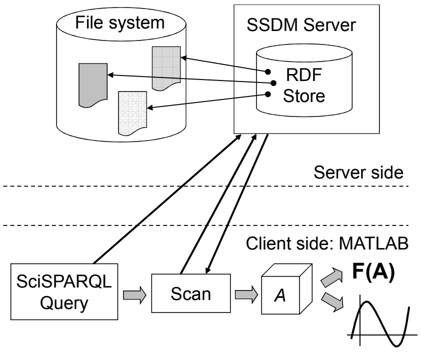

**Fig. 2.** Querying data and metadata on SSDM server from MATLAB client.

## 5. Conclusion

The use of standard query languages for bringing the remotely stored data into the computational environments is becoming increasingly popular as the data gets bigger and more distributed. MATLAB already has facility to execute SQL, and R statistical environment recently gained a simple SPARQL package [6]. We take the next step, by providing extensions to the standard query techniques, to make the database connections even more useful and efficient.

The approach with linking to the data instead of copying and storing it locally is beneficial, as the creation of the RDF graph to represent metadata takes negligibly small time compared to copying the massive data described by this RDF graph. There is a number of efficient binary storage formats around, and our approach can be easily extended to any of them, as long as it is possible to address stored data in terms of string or symbolic identifiers, and read specified parts of the arrays.

The main benefit, however, is integrating Semantic Web metadata management approach (RDF and SPARQL) into an environment that misses it so obviously. The MATLAB users can now take advantage of remote and centralized repositories for both massive numeric data and metadata, send queries that combine them both, retrieve exactly as much data as required for the task, and do any further processing the way they already do.

## References

1. A.Andrejev and T.Risch. Scientific SPARQL: Semantic web queries over scientific data. In International Workshop on Data Engineering Meets the Semantic Web, ICDE'12
2. A.Andrejev, S.Toor, A.Hellander, S.Holmgren, and T.Risch: Scientific Analysis by Queries in Extended SPARQL over a Scalable e-Science Data Store, In e-Science'13
3. M. Stonebraker, J. Becla, D. J. DeWitt, K.-T. Lim, D. Maier, O. Ratzesberger, and S. B. Zdonik. Requirements for science data bases and scidb. In CIDR '09.
4. E. Soroush, M. Balazinska, and D. L. Wang. Arraystore: a storage manager for complex parallel array processing. In SIGMOD '11
5. http://www.mathworks.se/help/pdf_doc/matlab/matfile_format.pdf
6. http://cran.r-project.org/web/packages/SPARQL/index.html