

An Archiving System for Managing Evolution in the Data Web

Marios Meimaris^{*}, George Papastefanatos⁺ and Christos Pateritsas^{*}

Institute for the Management of Information Systems, Research Center “Athena”, Greece
{m.meimaris, gpapas, pater}@imis.athena-innovation.gr

Abstract. The rising Data Web has brought forth the requirement to treat information as dynamically evolving aggregations of data from remote and heterogeneous sources, creating the need for intelligent management of the change-driven aspects of the underlying evolving entities. Datasets change in multiple levels, such as evolving semantics, as well as structural characteristics, such as their model and format. In this paper, we present an archiving system that is driven by the need to treat evolution in a unified way, taking into account change management, provenance, temporality, and querying, and we implement the archive based on a data model and query language designed specifically for addressing preservation and evolution management in the Data Web.

Keywords: Data Evolution, Change Management, Linked Data Dynamics

1 Introduction

The rising Data Web treats information as more than a collection of linked documents, but rather as a dynamic aggregation of data collections from remote and heterogeneous resources. The Linked Data paradigm [2] advocates appropriate standards and recommendations in order to promote best practices in the creation, publication and retrieval of data with the use of standards such as RDF¹ and SPARQL². The dynamic nature of web communities and the heterogeneity that comes with the data attribute temporality to the Data Web, which in turn can introduce changes in unpredictable ways, which can in turn lead to information and quality loss, as well as trust issues [7]. In this regard, the benefits of managing evolution are placed into two categories: (i) quality/trust and (ii) maintenance/preservation. In the former lie issues such as broken links or URI changes, while in the latter, data exploitation over historic data provides new insights, and therefore added value, on dataset dynamics and their relat-

⁺ Supported by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

^{*} Supported by the EU-funded ICT project "DIACHRON" (agreement no 601043).

¹ <http://www.w3.org/RDF/>

² <http://www.w3.org/TR/rdf-sparql-query/>

ed actors. However, evolution management is a multidimensional problem that involves versioning, temporal representation, change representation and management, provenance and temporal querying [1,9]. Our contribution is an archiving system that tackles these issues in a unified manner, by employing a data model and query language specifically designed to address the diverse characteristics of evolving Linked Data. Our resulting approach is a scalable archiving system that enables multi-versioning, metadata annotations in different levels of granularity, evolution management from single records and entities to datasets as wholes, a common representation layer that offers abstraction from source model as well as time in order to represent time-agnostic entities (datasets, linked data resources and so on) from different source models (e.g. ontological, relational etc.).

Related Work. In [10,11] the authors extend HTTP with a temporal dimension for accessing past versions of web documents and LD resources. In [12] the authors address multi-versioning for XML documents by using deltas between sequential versions. [3] propose XML archiving with top-down temporality. Semantic and structural differences between versions of RDFS graphs are addressed in [13].

2 Preliminaries

We use the data model presented in [4,5] as the basis for the archive system. The data model represents datasets from different source models and formats, namely relational, ontological and multidimensional, by offering a common representation level that treats information in a dual manner that involves representation of information-rich entities along with their semantics, as well as representation of structural aspects of the data, such as the break-down of datasets into schema elements, records and record attributes, in order to enable evolution management and metadata annotations at different levels of granularity. Changes are classified as simple or complex and are stored alongside their datasets. All types of entities (datasets, changes and so on) are represented in the same domain model, thus enabling them to be directly correlated and combined in queries that span across datasets and time. Time-agnostic representations of entities are differentiated from time-aware (i.e. particular versions) entities.

Based on this model, a query language [6] has been designed and implemented for intuitive querying of evolving datasets. This enables combining semantics, structural characteristics and changes in the same queries across time and datasets. A query can be comprised of sub-queries that define particular datasets, versions, changes, temporal ranges, as well as variable bindings for particular changes, records and/or attributes. The query language has been implemented as an extension of SPARQL.

3 Architecture and Implementation

The architecture and various components of the archiving system are depicted in figure 1. The main storage component is implemented on top of a Virtuoso 7.1³ instance. The archive's web service interface is exposed via HTTP as the primary access point

³ <https://github.com/openlink/virtuoso-opensource>

through a RESTful API. This way programmatic access and integration with more complex architectures is enabled. A web based GUI is implemented in order to offer browser access to uploading and querying of the archive's data. The main components are described in the following.

Query Processor. The base mechanism for data access and query planning and execution is provided by this component. The *Query Parser and Validator* module validates query syntax as defined in [6]. The parser parses the query string and maps it to corresponding Java objects. The *Query Translator* creates execution plans for queries by translating to the native query language of the persistence store, i.e. SPARQL. The created plans are based on archive structures implemented in the persistence store and the appropriate indexes and dictionaries. Finally, the *Query Executor* module is responsible for the execution of the created plan and builds the result set of the query.

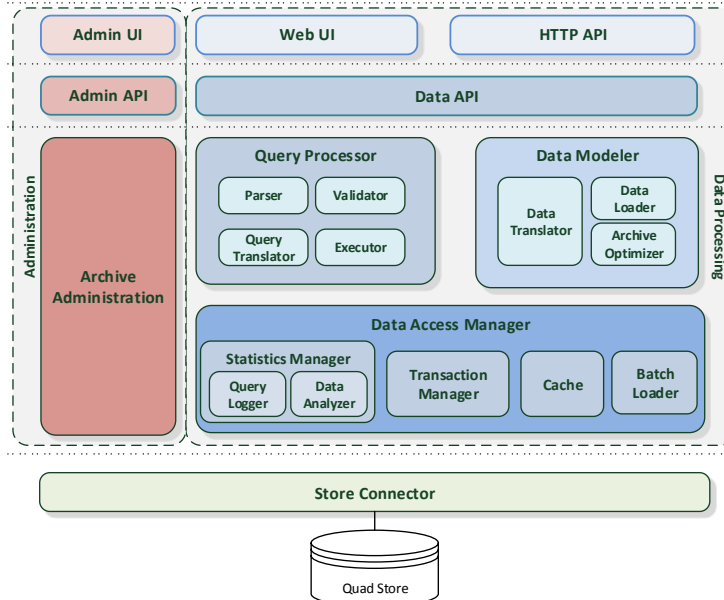


Figure 1. Architecture of the archive.

Data Modeler. This component handles creation and conversion of datasets from and to the diachronic data model defined in [4,5] to their source data models and formats/serializations. The *Data Translator* module handles dataset conversion to the archive's internal storage structures, and is also responsible for named graph management, URI minting and dictionary administration. The *Data Loader* module serves as the input interface for dataset insertion, and supports both bulk loading and single insertions of datasets. Furthermore, it links new dataset versions to their corresponding diachronic datasets. Finally, the *Archive Optimizer* module is designed to enable optimization of the storage method based on pre-defined archive strategies [8].

Data Access Manager. This component provides low level data management functionality for the archive. It decouples the archive's functionality from its underlying store. Its sub-component, the *Transaction Manager* module, provides low level trans-

action handling (commit, rollback) and also handles the store connection. The *Cache* module acts as temporary in memory storage of archive structures that are frequently accessed. The *Batch Loader* offers alternative ways of efficient loading to the store. Finally, the *Statistics Manager* module collects and analyzes datasets and querying workload so as to populate statistical information about them. This information can be used for query optimization by the *Query Translator*.

Archive Administration. This component is a set of management and configuration tools of the archive that provide functionality such as manual configuration of archiving strategies, as well as user management. Moreover this component can act as an abstraction layer for common management tasks (e.g. start/stop data store service) in order to decouple the Admin API and UI from the store technology and/or vendor.

Store Connector. The Store Connector is the software package that provides an API to other components of the archiving module for communication and data exchange with the underlying store. It is usually provided by the store's vendor and is dependent to a specific programming language, for example the Virtuoso JDBC Driver package⁴.

Data Store. This is the persistence mechanism of the archive. It employs a generic native RDF store with named graph support, where any store implementation is pluggable.

4 Implementation and Evaluation

We have implemented all components in Java; we have used the Spring⁵ framework for developing the HTTP API services and the Data API, and we extended the SPARQL 1.1 grammar parser and query planner based on Jena ARQ⁶. We evaluated the archive system in two different data contexts, drawn from the life sciences (ontological) and the government statistics (multidimensional) domains. The archive's performance was assessed in three main dimensions: (i) loading times, (ii) retrieval and conversion times (de-reification and serialization), and (iii) query execution times [6]. Our tests indicate that as far as query and load times are concerned, the system scales linearly w.r.t. the storage's dataset input size.

5 Conclusions

The increasing volume and dynamicity of datasets in the Data Web has created new issues related to the decentralized nature and evolution of Linked Data. As a consequence, this has introduced requirements for efficient solutions to such problems. In this paper, we have presented an archiving system that tackles evolution management of datasets in the Data Web. The archiving system is based on a data model and query language designed to directly and natively address the issues of data preservation, change detection, change representation, provenance and metadata annotations at

⁴ <http://docs.openlinksw.com/virtuoso/VirtuosoDriverJDBC.html>

⁵ <http://projects.spring.io/spring-framework/>

⁶ <http://jena.apache.org/documentation/query/>

different granularity levels, as well as source heterogeneity. We have implemented and tested the archive in the life sciences and government statistics domains and we have found it to scale linearly to increase in the input size as far as querying and loading are concerned. As future work, we intend to fully encompass the dynamic archiving strategies described in [8], and address longitudinal querying and data retrieval efficiency in settings of hybrid dataset storage that includes versions stored as both deltas and full materializations.

6 References

1. Auer, S. et al. "Diachronic linked data: towards long-term preservation of structured interrelated information." In *Proceedings of the First International Workshop on Open Data*, pp. 31-39. ACM (2012).
2. Bizer, C. et al. "Linked Data – The Story So Far". Special. Issue on Linked Data, In *International Journal on Semantic Web and Information Systems* (2009)
3. Buneman, P. et al. "Archiving scientific data." *ACM Transactions on Database Systems (TODS)* 29, no. 1 (2004)
4. Meimaris, M. et al. "Towards a Framework for Managing Evolving Information Resources on the Data Web." In *Proceedings of Profiles Workshop of ESWC*, vol. 14. (2014)
5. Meimaris, M. et al. "A Framework for Managing Evolving Information Resources on the Data Web", *arXiv preprint arXiv:1504.06451* (2015)
6. Meimaris, M. et al. "A Query Language for Multi-version Data Web Archives", *arXiv preprint arXiv:1504.01891* (2015)
7. Papastefanatos, G., and Stavrakas, Y.. "Diachronic Linked Data: Capturing the Evolution of Structured Interrelated Information on the Web." *ERCIM News* 2014, no. 96 (2014).
8. Stefanidis, K. et al. "On designing archiving policies for evolving RDF datasets on the Web." In *Conceptual Modeling*, pp. 43-56. Springer International Publishing (2014).
9. Umbrich, J. et al. "Dataset dynamics compendium: A comparative study." (2010).
10. Van de Sompel, H. et al. "An HTTP-based versioning mechanism for linked data." *arXiv preprint arXiv:1003.3661* (2010).
11. Van de Sompel, H. et al. "Memento: Time travel for the web." *arXiv preprint arXiv:0911.1112* (2009).
12. Raymond W. and Lam, N. "Managing and querying multi-version XML data with update logging." In *Proceedings of the 2002 ACM symposium on Document engineering*, pp. 74-81. ACM (2002).
13. Völkel, M., and Groza, T. "SemVersion: An RDF-based ontology versioning system." In *Proceedings of the IADIS international conference WWW/Internet*, vol. 2006, p. 44 (2006)