University of Bonn

GUI And Management System For RDFSlice

Author:
Emrah Ozkan

Supervisor:
Dr. James Smith

A thesis submitted in fulfilment of the requirements
for the degree of Master of Computer Science

in the

Research Group Name
Department or School Name

May 2015

Declaration of Authorship

I, Emrah Ozkan, declare that this thesis titled, 'GUI And Management System For RDFSlice' and the work presented in it are my own. I confirm that:

Signed: _____

Date: _____

UNIVERSITY NAME (IN BLOCK CAPITALS)

Abstract

Faculty Name
Department or School Name

Master of Computer Science

GUI And Management System For RDFSlice

by Emrah Ozkan

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

Dedicatory text here.....

Chapter 1

Introduction

1.1 Motivation

Finding the accurate information has been one of the main intellectual problems in human history. Different disciplines; from mathematics to philosophy, developed unique methods to obtain the information desired. In the case of computer science, the time spent for reaching the data is an important metric as the accuracy. Complexity of the search methods depend on the data structure as well as the size of the search set. In the notion O(n), n denotes the number of objects that are included the in the set. Table 1.1 shows different runtimes for various data structures.

As it is easily observed, the set size has a direct effect on the runtime. Not only search engines but many applications today use databases at great sizes, forcing developers to optimize queries so that an acceptable runtime is preserved. And as stated above, having an optimum amount of data can have a positive effect on the runtime as having a well-structured search algorithm.

Structure	**Average**	**Worst Case**
Array	$O(n)$	$O(n)$
B-Tree	$O(log(n))$	$O(log(n))$
Binary Search Tree	$O(log(n))$	$O(n)$
Hash Table	$O(1)$	$O(n)$

TABLE 1.1: Comparison of data structures in terms of search complexity.

Year	Global Internet Traffic
1992	100 GB Per Day
1997	100 GB Per Hour
2002	100 GBps
2007	2000 GBps
2013	28,875 GBps
2018	50,000 GBps

TABLE 1.2: The Cisco VNI Forecast Within Historical Context. Source: Cisco VNI, 2014

1.1.1 Linked-data and RDF

Various forms of data has been used for applications, retrieved from data sources of sorts. In parallel to the increase in internet coverage and integration of computers into daily life, many web applications have been produced to take over the processes that has been completed manually in person such as banking transactions, shopping, customer services, dispatching...etc, which required physical presence and paperwork. Keeping such records in databases instead of file cabinets seemed useful. In addition to the transfer of paperwork to digital media, use of ubiqituous computing had a huge impact on the However the exponential growth in the data size of web led to problems as well as new ideas. Table 1.2 shows the global internet traffic from 1992 to 2013 and Cisco's Visual Networking Index prediction for 2018.

One of the most noticable problem that larger physical space needed to store the data produced and provide network to transfer the data. To reduce server costs, algorithms that require less space have been demanded. Data replications were also target to such resource planning. Same data of non-controversial facts have been repeated all over the web and seperate space has been allocated for each fragment. Example for such data can be capital cities, musical album names, timestamps of historical events...etc.

Even those who owned the data was willing to share what they had, it was avaiable for one or a few applications and that would be possible if only the other applications had access to the database or the data was shared over webservices, which required additional work and converting data into types like XML or JSON thus demanding additional resources. Servers would not be able to handle numerous calls. Computers, simply could not process the information on a web page efficiently since they did not understand it. A concept was required to ease the sharing of data without having to handle server requests at enormous scale, saving space by avoiding repetations and optimize resource usage by skipping conversion into different data types, simply offering a common form that could be processed by any computer.

The term semantic-web denotes to the "meaningful part of the web", which simply means the metadata of web pages and/or applications. DuCharme describes semantic web in his book Learning Sparql; as a set of standards and best practices for sharing data and the semantics of that data over the Web for use by applications. [1] Extracting the metadata would be useful to gather it and share among the applications that are also part of it. In 2006, W3C offered the concept of linked data, which basically is data that has links on it. The links enable computers to reach the address where the metadata is stored. Usage of the data is simply done by referencing a link to it. Or can be described as : Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets. [2]

The motivation for producing such structure has been listed by Klyne and Caroll as follows :

- Web metadata: providing information about Web resources and the systems that use them (e.g. content rating, capability descriptions, privacy preferences, etc.)
- Applications that require open rather than constrained information models (e.g. scheduling activities, describing organizational processes, annotation of Web resources, etc.)
- To do for machine processable information (application data) what the World Wide Web has done for hypertext: to allow data to be processed outside the particular environment in which it was created, in a fashion that can work at Internet scale.
- Interworking among applications: combining data from several applications to arrive at new information.
- Automated processing of Web information by software agents: the Web is moving from having just human-readable information to being a world-wide network of cooperating processes. RDF provides a world-wide lingua franca for these processes.

[3]

And to come up with a feasible solution to fulfil such requirements, Tim Berners-Lee offered a set of design principles for linked-data structure.

1. Use URIs as names for things. The uniqueness of URI's make it easier to describe the entities, hence the connection between them

2. Use HTTP URIs so that people can look up those names. Different URI forms have been offered however such attempts would decrease the interoperability of linked data, which it is all about.

3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

4. Include links to other URIs. so that they can discover more things.

[4]

Linked data enables the operations mentioned by providing a form of data that can be readable and processable by machines. In order to have a better understanding of how linked data works, an introduction on RDF would be beneficial.

RDF

The notion RDF stands for Resource Description Framework, basically a data type for describing web resources. A simple RDF entity consists of three parts; object, predicate and subject hence called a "triple". Subject denotes the name of the resource and is of type URI (Uniform Resource Identifier), predicate is the property of that resource; pointing to an object. And object stores the value of that property, which belongs to the subject. An object can store different data types such as string, integer. date...etc. Or it can be another subject thus have its own URI.

Each triple can be visualised as a graph, which makes it easier to understand the RDF structure and why it is useful for linked-data movement. In such format, subjects and objects will be nodes and predicates are the edges that connects them. Suppose we would like to visualise the statement "The author of http://www.w3schools.com/rdf is Jan Egil Refsnes" [1] and store it as an RDF object.

FIGURE 1.1: Graph of the statement "The author of http://www.w3schools.com/rdf is Jan Egil Refsnes" [Generated at http://www.w3.org/RDF/Validator/]

Figure 1.1 is a graph representation of the statement mentioned. Subject is the node with the value http://www.w3schools.com/rdf, the property is shown as the edge author (which belongs to the namespace http://www.w3schools.com/rdf therefore represented as http://www.w3schools.com/rdf/author) and object value is the "author" of the subject, Jan Egil Refsnes.

RDF has an XML-like syntax, only with different classes and namespaces. The figure 1.1 can be represented as the following code :

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:si="http://www.w3schools.com/rdf/">
        <rdf:Description rdf:about="http://www.w3schools.com/rdf/">
                <si:author>Jan Egil Refsnes</si:author>
        </rdf:Description>
</rdf:RDF>
```

In 2000, Decker & Harmelen discussed the advantages of RDF structure (while XML existed long before) for semantic web as: When it comes to Semantic Interoperability, RDF has significant advantages over XML: semantic units are given naturally through is object-attribute structure: all objects are independent entities. A domain model, defining objects and relationships of a domain of interest, can be represented naturally in RDF, so translation steps, as required when using XML, are not necessary. To find mappings between two RDF descriptions, techniques from Knowledge Representation are directly applicable. [5]

1.1.2 Linked Data Applications

In this section we will have a glance over applications that are using linked-data and their database sizes.

BBCMedia

1.2 Objective

In this master thesis, we will approach the problem of handling large linked-data datasets of diffeerent types. (RDF,N-Triples, Turtle, OWL...etc). Although large linked-data sets can consist of well structured graphs, the size of triples can reach up to billions and not all this data might be required, thus not have to be queried. As in other data structures, large linked-data sets also cause the problems of storage and performance. Runtimes for different linked-data analyse tools will be presented in the state of the art section.

Today there are many projects available that are using graph-based databases as alternative to relational (SQL queried) or document-based databases. An example to such applications is DBPedia which is an important asset to Semantic Web movement providing an alternative to Wikipedia offering an open data resource in RDF format. However

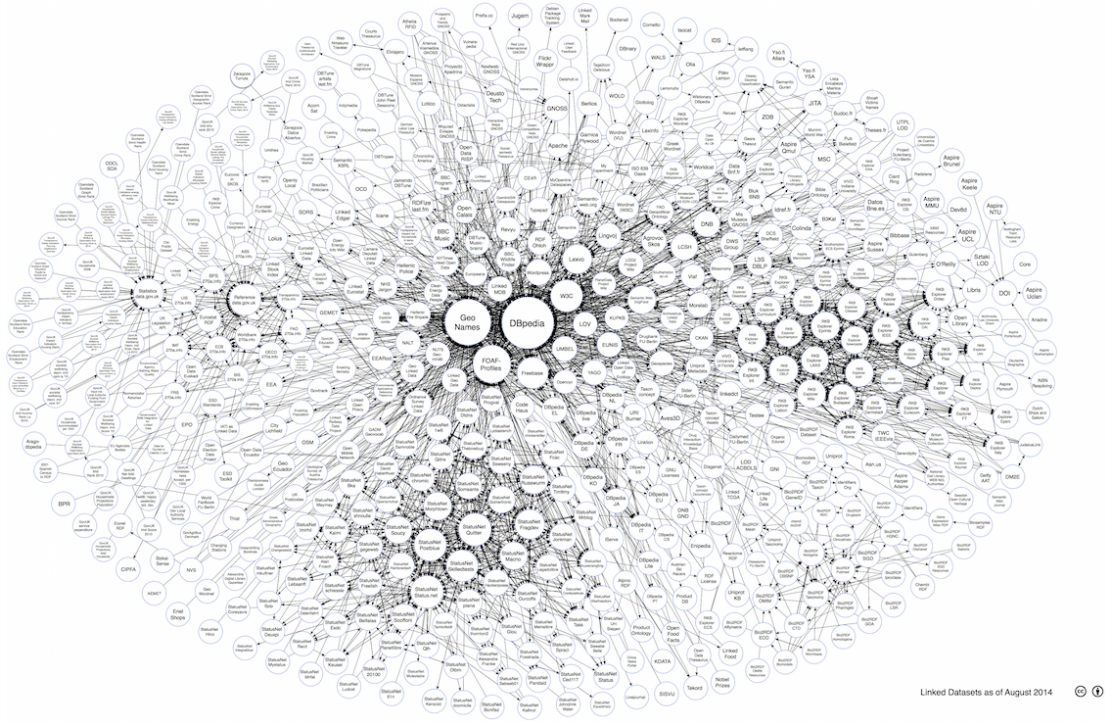

FIGURE 1.2: Latest state of Linked-data cloud [6]

using data in such size is not that easy. DBPedia into a triple store can take up to 7 hours on a computer with standard hardware. Querying such entity during application creating process also will yield to decrease in efficiency.

In order to overcome the problem of dealing with huge data dumps, AKSW Research Group has come up with a solution called RDFSlice, which basically creates slices from large datasets to form application specific knowledge. It also enables users to create slices from diferent datasources and combine them in a single file. Although RDFSlice is a platform-free tool, it currently can be executed through terminal or command prompt interpreter.

```
java -jar rdfslice.jar -source <fileList>|<path> -patterns <graphPatterns>
-out <fileDest> -order <order> -debug <debugGraphSize>
```

The parameter ¡graphPatterns¿ is either a SPARQL query or a graph pattern. Current release of RDFSlice doesn't provide a detailed feedback upon erroneous parameters. Thus users cannot identify which parameter is not accepted. Also complex patterns and class slicing requires advanced SPARQL knowledge. Plus RDFSlice doesn't store any information regarding the slice created except for the output file. This thesis will focus on a software to support RDFSlice visually with additional attributes. A graphical user interface to;

- Analyse datasets by breaking them into subjects, predicates, objects and classes without using any endpoints
- Form SPARQL queries and patterns using entities of the dataset with no syntax errors.
- Create slices by executing RDFSlice without using terminal or command prompt
- Share and update created slices over a user-account based web portal

Chapter 2

State of the Art

2.1 RDFSlice

Different applications are discussed with respect to their emphasis/focus and requirements for management systems

2.2 Related Methodologies and Applications

2.2.1 Dataset analysis

Fuseki, virtuoso...etc. Endpoints that require advanced sparql knowledge. Also sparql is used to extract classes. Slicing data (except saving manually) or sharing/storing queries is not supported.

2.2.2 Slicing tools

LODCUBE Cubes are an example of linked data slicing however useful on a very small scale.

2.2.3 Management methodology

Explaining sharing information without additional data for dataset. Requirements, examples (Spotify)

Chapter 3

Implementation

3.1 Design

Design approach with respect to the requirements (GUI explanation, HCI decisions)

3.2 Technology

Programming languages, frameworks and editors used

3.3 Implementation

Workflow, layered code structure,

3.4 Challenges

Difficulties, unexpected results, failed objectives

Chapter 4

Evaluation

4.1 Evaluation metrics

Metrics used to evaluate the software

4.2 User experience

User profiles attended, scenarios

4.3 Results

Chapter 5

Discussion

5.1 Evaluation metrics

Metrics used to evaluate the software

5.2 User experience

User profiles attended, scenarios

5.3 Results

Chapter 6

Conclusion

6.1 Summary

Objective

How much of it has been fulfilled

Future work

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] Bob DuCharme. *Learning SPARQL, Second Edition*. O'Reilly Media, Inc.,, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2013.

[2] C Bizer, T Heath, and T Berners-Lee. Linked data-the story so far. 2009.

[3] W3C. *Resource Description Framework (RDF): Concepts and Abstract Syntax*, 2004. URL http://www.w3.org/TR/2003/PR-rdf-concepts-20031215/.

[4] Tim Berners-Lee. *Linked Data-Design Issues*, 2009. URL http://www.w3.org/DesignIssues/LinkedData.html.

[5] Stefan Decker, Frank Van Harmelen, Jeen Broekstra, Michael Erdmann, Dieter Fensel, Ian Horrocks, Michel Klein, and Sergey Melnik. The semantic web - on the respective roles of xml and rdf. *IEEE Internet Computing*, 4:http://www.ontoknowl, 2000.

[6] Richard Cyganiak. *The Linking Open Data cloud diagram*, 2014. URL http://lod-cloud.net/.

[7] W3C. *DataSetRDFDumps*, 2014. URL http://www.w3.org/wiki/DataSetRDFDumps.