ISSN No. 0976-5697

Volume 5, No. 4, April 2014 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Optimizing SPARQL queries in Linked Open Data using Heuristic approach

Gouri D. Potdar
Department of Computer Engineering
Goa College of Engineering, Farmagudi
Ponda, India
potdargouri@gmail.com

Rachel Dhanaraj
Department of Computer Engineering
Goa College of Engineering, Farmagudi
Ponda, India
racheldhanaraj@gmail.com

***Abstract*—** As today's world tends to rely more and more on search engines for quenching its thirst for information, search engines, today, are expected to be faster, more accurate, more intelligent and more powerful so as to reach a wide pool of information resources. Data sources like CKAN[1], DBpedia[2], GeoNames[3], FOAF[4] which collectively form Linked Open Data (LOD) have gained importance in this quest for better search engines. SPARQL is the w3c recommended query language which is used to extract data from LOD sets. SPARQL queries typically contain more joins than equivalent relational plans, and hence lead to a large join order search space. Consequently, query optimization in RDF Stores is a challenge. The dynamic nature of LOD prevents the application of the cost based approach which requires statistics. Moreover, the relevant correlations cannot be identified beforehand. Hence, using good heuristics for SPAQRL query optimization is an advantage.

***Keywords*—**Linked Open Data, Heuristics, Query Optimization, SPARQL, RDF

I. INTRODUCTION

We are living in an era of modern information technology. Huge data repositories are accessible on just a click away in the world of web. Organizing information on such a large scale is an important task and for structuring data Linked Open Data (LOD) is used. LOD integrates the various data sets available and forms the web of data. The major issue while dealing with the dynamic and global data space is meaningful exploitation and usage of huge amount of semantic data. Lack of efficient and effective storage and querying techniques is proving crucial at this juncture.

SPARQL has become really important query language. It provides a mechanism to express constraints and facts and the entities matching those constraints are returned to the user. However, the syntax of SPARQL requires users to specify the precise details of the structure of the graph being queried in the triple pattern. To ease querying from an infrastructural perspective, data contributors have provided public SPARQL endpoints to query the LOD cloud datasets. [3]

In this paper, we are proposing optimization of the SPARQL query which is used to fetch the response from various data sources like DBpedia and freebase. We are working on the SPARQL queries used in the search engine. The search engine will assist the user when he/she will start interacting with initial query typed in search engines input box. It will provide the autosuggestions when user will type the initial query, also the query can be replaced or refined for more precise query writing. When the query is submitted to the search engine it will fetch the information from the DBpedia and Freebase using SPARQL as a query language. The evaluation of SPARQL queries which contains filter (!regex) expressions degrades the performance and results in delay. In this paper, we are focusing on producing the execution plans with the maximum number of merge joins. Merge joins make use of the ordering of the joining attributes to achieve better execution times. [1]

II. LINKED OPEN DATA (LOD)

A. *Rationale:*

Most of the times, when we start searching for any information, we get diverted from information which we are seeking by irrelevant information which does not match our expectations accurately. Due to this reason users keep changing their initial query.

The main reason for user dissatisfaction is that the users do not know what exactly is to be typed i.e. the users do not know the precise query which will give the best answer to their need. Also the speed is important factor for search engine where optimization comes into picture.

To solve this problem the user needs some assistance from the search engine which will help the user to get the results faster. [4]

B. *Linked Open Data (LOD):*

Linked Open Data is a way of publishing data on the Web that encourages reuse, reduces redundancy, maximizes its (real and potential) inter-connectedness, enables network effects to add value to data. LOD uses the RDF (Resource Description Framework) data format for describing things and their interrelations. [6]

a. All items of interest, such as information resources, real-world objects, and vocabulary terms are identified by URI references [9].
b. URI references are dereferenceable; an application can look up a URI over the HTTP protocol and retrieve an RDF description of the identified item.
c. Descriptions are provided using the RDF/XML syntax.
d. Every RDF triple is conceived as a hyperlink that links to related information from the same or a different source and can be followed by Semantic Web agents.

These principles are sufficient to create a Web of Data in which anyone can publish information, link to existing information, follow links to related information, and consume and aggregate information without necessarily having neither to fully understand its schema nor to learn how to use a proprietary Web API. [5]

Whenever we are dealing with the large amount of data, the size as well as dirty nature of the data needs Extraction, Transformation and Loading (ETL) processes. It can easily be translated into SPARQL queries that overwhelm the current modern technique in RDF database systems. Such problems typically come down to formulating joins that produce huge results, or to RDF database systems that calculate the wrong join order such that the intermediate results get too large to be processed. [10]

Linked open data cloud is increasing day by day as more number of users is publishing their data/websites using RDF. SPARQL protocol and RDF query language, also known as SPARQL, is used to query this linked data. This protocol was developed and widely accepted after multiple attempts to make SQL the query language for RDF. But due to the dynamic structure of nodes in the cloud of connected databases this was not possible. Thus SPARQL emerged as the query language which can query RDF for data, using URIs.

Generally SPARQL end points are defined by data providers so that users do not have to know the complete graph of links and just query the data using the available information. The query execution would take place by gathering up all the resources at one point and then executing the query on that point. This may not be possible every time with the increase in density of the Linked open data cloud. Thus the need of optimization techniques became imminent. [11]

III. THE PROPOSED APPROACH

A. *RDF (Resource Description Framework):*

RDF is an essence of triple format namely subject, predicate and object. [7]

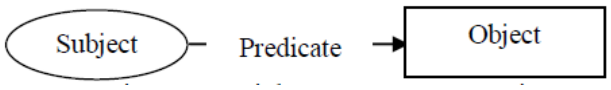


Figure 1Triple Format Representation

The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource.

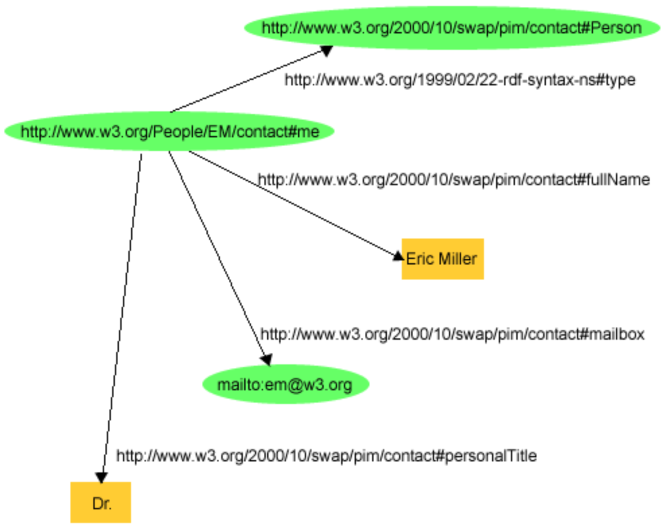
However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be *identified* on the Web, even when they cannot be directly *retrieved* on the Web.

RDF is based on the idea of identifying things using Web identifiers (called *Uniform Resource Identifiers*, or *URIs*), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a *graph* of nodes and arcs representing the resources, and their properties and values.

The group of statements "there is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr." could be represented as the RDF graph in Figure 2:

Figure 2 illustrates that RDF uses URIs to identify:

a) individuals, e.g., Eric Miller, identified by http://www.w3.org/People/EM/contact#me
b) kinds of things, e.g., Person, identified by http://www.w3.org/2000/10/swap/pim/contact#Person
c) properties of those things, e.g., mailbox, identified by http://www.w3.org/2000/10/swap/pim/contact#mailbox
d) values of those properties, e.g. mailto:em@w3.org as the value of the mailbox property (RDF also uses character strings such as "Eric Miller", and values from other data types such as integers and dates, as the values of properties)



http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

Figure 2 An RDF Graph Describing Eric Miller [8]

B. *Methodology using Heuristics:*

Due to the fine-grained nature of RDF data – where a triple is just a narrow tuple with three attributes – SPARQL queries involve a large number of joins. Such joins dominate the query execution time. In addition, RDF data does not come with schema or integrity constraints; therefore, a query optimizer cannot take advantage of such information to produce an efficient query plan. Another approach for query optimization is needed, one based on the observation that the syntactical form of a SPARQL query reveals information about the data to be accessed. We advocate the use of heuristics to determine the query execution plan, instead of maintaining costly statistics for the stored data. Due to the highly distributed, volatile, and ever-changing nature of semantic data, a cost-based optimizer is likely to under-perform more often because of outdated statistics.

A SPARQL join query consists of numerous costly joins. The first and foremost important goal is to maximize the number of merge joins in the query plan. A merge join in this context is most commonly a sort-merge join, or any other join that takes advantage of the existence of an index.

A SPARQL join query consists of numerous costly joins. The first and foremost important goal is to maximize the number of merge joins in the query plan. A merge join in this context is most commonly a sort-merge join, or any other join that takes advantage of the existence of an index.

An equally important goal is to minimize intermediate results in order to minimize the memory footprint during query execution. This is achieved by choosing the most selective triple patterns to evaluate first. Traditionally, deciding which triple patterns are more selective relies on statistics.

Here are the heuristics which we are applying for optimizing SPARQL queries. [2]

HEURISTIC 1 (Triple pattern order). Given the position and the number of variables in a triple pattern one can derive the following order, starting from the most selective, i.e., the one that is likely to produce less intermediate results, to the least selective.

$(s, p, o) \prec (s, ?, o) \prec (?, p, o) \prec (s, p, ?) \prec (?, ?, o) \prec (s, ?, ?) \prec (?, p, ?) \prec (?, ?, ?)$

The above ordering is based on the observation that given a subject and an object there are only very few, if not only one, properties that can satisfy the triple pattern. Similarly, it is very rare that a combination of a subject and property has more than one object value. In the same line of thinking we derive the rest of the orders. There can only be few subjects that have the same value for a property, while there are more many subjects with the same property no matter the object value. Finally, if a query pattern has 2 variables, then objects are more selective than subjects, and subjects more selective than properties. An exception to this rule is when the property has the value rdf:type, since that is a very common property and thus these triples should not be considered as selective.

HEURISTIC 2 (Distinct position of joins). The different positions in which the same variable appears in a set of triple patterns captures the number of different joins this variable participates in. A variable that appears always in the same position in all triple patterns, for example as subject, entails many self joins with low selectivity. On the other hand, if it appears both as object and property, chances are the join result will be smaller. The following precedence relation captures this preference:

$p \bowtie o < s \bowtie p < s \bowtie o < o \bowtie o < s \bowtie s < p \bowtie p$

Where s, p, o refer to the subject, property, and object position of the variable in the triple pattern. This ordering stems from our observations while studying RDF data graphs. RDF data graphs tend to be sparse with a small diameter, while there are *hub* nodes, usually subjects. As a result, query graph patterns that form linear paths are more selective.

HEURISTIC 3 (Triples with most literals/URIs). This heuristic is a special subcase of HEURISTIC 1 but can be used independently. Triple patterns that have the most number of literals and URIs – or symmetrically less variables – are more selective. This heuristic is similar to the bound as easier heuristic of relational query processing , according to which, the more bound components a triple pattern has, the more selective it will be.

HEURISTIC 4 (Triples with literals in the object). An object of a triple pattern may be a literal or a URI. In such case, a literal is more selective than a URI. This is true for RDF data because in many cases if a URI is used as an object, it is used by many triples.

HEURISTIC 5 (Triple patterns with less projections). This heuristic allows us to consider as late as possible the triple patterns that contain projection variables. In the case in which the compared sets of triple patterns have the same set of projection variables, we prefer the set with the maximum number of unused variables that are not projection variables.

The above heuristics can be used in combination or separately for determining the order in which triple patterns should be evaluated, and thus achieving smaller intermediate results. These heuristics are suitable for different planning approaches, such as distributed environment, or hybrid optimizers where a cost model and heuristics work together.

IV. CONCLUSION

We have studied about the SPARQL and Linked Open Data. The emerging trends in search engine demands better results in less time, to fulfill this requirement a high speed search engine with large scale data set is needed. To achieve this goal we will implement the GUI having services like query suggestion/refinement. This will naturally elevate the standard of search engine. Also the SPARQL queries will be optimized using heuristics presented above.

V. REFERENCES

[1]. Ioannis Papadakis a and Ioannis Apostolatos , LOD-based query construction/refinement service for web search engines, , Undefined 1 (2013) 1–10 IOS Press.

[2]. Petros Tsialiamanis, Lefteris Sidirourgos, Lefteris Sidirourgos, Heuristics-based Query Optimisation for SPARQL ,EDBT 2012, March 26–30, 2012, Berlin, Germany. Copyright 2012 ACM 978-1-4503-0790-1/12/03.

[3]. Prateek Jain, Kunal Vermay, Peter Z. Yehy, Pascal Hitzler and Amit P. Sheth, LOQUS: Linked Open Data SPARQL Querying System, 2010.

[4]. I. Apostolatos, I. Papadakis, GContext: Context-based search powered by Wikipedia, AI Mashup Challenge, Extended Semantic Web Conference - ESWC 2012, 5th finalist

[5]. Proceedings of the 3nd International Workshop on Scripting for the Semantic Web (SFSW 2007) Co-located with 4rd European Semantic Web Conference June 3-7, 2007, Innsbruck, Austria.

[6]. http://tomheath.com/slides/2009-02-austin-linkeddata-tutorial.pdf

[7]. Queries R.Gomathi1, C.Sathya2 , D.Sharmila , Efficient Optimization of Multiple SPARQL, e-issn: 2278-0661, p-issn: 2278-8727volume 8, issue 6 (jan. - feb. 2013), pp 97-101

[8]. http://www.w3.org/TR/rdf-primer/#figure1

[9]. Dietzold, S.: Basic vocabulary to use LDAP data in RDF. OWL ontology (2005) http://purl.org/net/ldap.

[10]. Spyros Kotoulas, Jacopo Urbani, Peter Boncz, and Peter Mika, Robust Runtime Optimization and Skew-Resistant Execution of Analytical SPARQL Queries on Pi, The Semantic Web–ISWC 2012, 2012 – Springer.

[11]. Prof. Bhaumik Nagar, Prof. Ashwin Makwana , Chirag Pandya Optimizing Query execution over Linked Data, International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 3, March 2012)