Survey Paper for Different SPARQL Query Optimization Techniques

Drashty R. Dadhaniya, Ashwin Makwana
Department of Computer Engineering
CSPIT, CHARUSAT University, Changa, Gujarat, India
drashty.dadhaniya@gmail.com

Abstract: Motivated by the ongoing success of semantic web and the increasing amount of Linked Data available on the Web, new challenges to SPARQL query optimization are emerging. According to the static vs. dynamic and top down vs. bottom up type of query optimization, the paper describes survey of different approaches for query optimization like query rewriting, query analysis, selectivity of triple patterns, mixed strategy, and evaluating graph traversal algorithms for query optimization. It also discusses each optimization technique, their characteristics, the comparisons and several challenging problems in detail.

Keywords: Linked Data, RDF (Resource Description Framework), Semantic Web, SPARQL (SPARQL Protocol and RDF Query Language)

I. INTRODUCTION
Semantic Web is like a huge single database made through structured metadata, vocabularies and linking between open data. This goal is achieved by three technical standards like RDF, SPARQL, OWL that provided by W3C. The development of semantic web is proceeds in layer one after another [1].

Foundation for representing and processing metadata is provided by RDF. Subject, predicate and object are the key concepts of RDF which is a graph based data model. RDF is domain independent but RDF schema is used to describe specific domains. RDF Schema is a primitive ontology language. Key features like class, subclass relations, property, sub property relations, and domain and range restrictions are provided by RDF Schema. OWL is built on top of RDF and RDF Schema to provide the features like scope of properties, disjointness of classes, combination of classes, cardinality restrictions, and special characteristics of properties (transitive/unique/inverse). SPARQL is the query language proposed by W3C recommendation to query RDF data [3]. A SPARQL query is made by a set of triple patterns. A triple pattern is having a variable at subject, object or predicate position. The solution of query binds the variable to appropriate URI [4] or literal in the RDF model according to query structure.

The objective of this paper is to present a short summary of different SPARQL query optimization methods. There are several approaches have been proposed for SPARQL query optimization but only few are implemented. In this paper different approaches like query rewriting [5], selectivity based triple pattern reordering [6], query analysis [7], mixed strategy [8], and other approaches of evaluating graph traversal algorithms for query optimization such as [9,10] are mentioned.

II. QUERY REWRITING BASED OPTIMIZATION
Query rewriting based optimization algorithm is presented by Olaf Hartig and Ralf Heese in [5]. The optimization process works in two phases, proposed algorithm is for the first phase which translates a SPARQL query into a SQGM (SPARQL Query Graph Model) [5] which supports all phases of query processing. Second phase rewrites the query based on generated SQGM to reduce the query execution plan.

In the first phase, the SPARQL query is given as a tuple (DS, GP, SM, R) to the input of the algorithm and returns the corresponding SQGM. Where DS is the queried RDF dataset, GP is a graph pattern, SM is a set of solution modifiers and R is the result form [3]. In the second phase to achieve a better execution strategy generated SQGM is transformed into a semantically equivalent. First define semantic equivalence of two SQGMs and than define transformation rules. Transformation rule merges the join of two graph pattern operators to a single operator. A heuristic is generated with a set of preconditions and a set of rewrite rules. The resulting SQGM reduces query execution time if all preconditions are fulfilled and rewrite rules are applied. This SPARQL query graph model can be extended to support structures like group by, sub queries and views.

III. SELECTIVITY BASED OPTIMIZATION
Selectivity based optimization algorithm is presented by authors in [6]. The given optimizer consists of three main
components. First component is BGP (Basic Graph Pattern) Abstraction. SPARQL query is abstracted as an undirected graph. Second component is core optimization algorithm which uses minimum selectivity approach to generate the query execution plan. Third component is set of heuristics that help the optimization algorithm in selectivity estimation. Heuristics without pre-computed statistics and heuristics with pre-computed statistics are presented for triple patterns selectivity and joined triple patterns selectivity. Evaluation is performed based on Lehigh University Benchmark (LUBM). Evaluation shows that the algorithm enhances the performance for simple queries and outperforms when ARQ/PF heuristic estimate same selectivity for two triple patterns. For the static query optimization this algorithm reduces the number of execution plans.

IV. MIXED STRATEGY FOR QUERY OPTIMIZATION

In [8] authors have discussed a mixed strategy for query optimization that is combination of top down [11] and bottom up strategy [12]. To take advantage of run time features of a query execution like triple pattern results, join pattern results, links to results, indexes used to compute the metrics, newly discovered information used to refine and correct previously computed metrics, corrective source ranking technique is used. To alleviate the problem of busy waiting in a loop, stream based approach with an operator symmetric hash join (SHJ) is used. This approach reduces the query execution time by early reporting of results. This early reporting is 42% faster than bottom up strategy. This approach optimize only the source ranking process for that reason it can be extended to optimize the entire evaluation process using information discovered at run time.

V. QUERY ANALYSIS BASED QUERY OPTIMIZATION

In [7] authors have proposed an approach that prevents unnecessary fetching of classes that cannot contribute to answer of the query. Approach works in two phases. Query analysis is done by first phase before the execution of query. It identifies classes that cannot contribute to the result of query and prevents those classes from being fetched. Second phase analyze the whole query execution and model it as a context graph [7]. This pattern is used by heuristic to analyze more patterns that can be only discovered at run time and it further reduces the amount data fetched from web to answer the query results. This approach reduces the query execution time up to some extent and improves query performance.

VI. FEDEX FRAMEWORK

In [13] authors have presented a practical framework for SPARQL query processing. In their framework they have used three novel techniques to minimize the number of remote requests. Modified Source Selection [11] technique is proposed which can identify relevant sources in the absence of preprocessed metadata. Exclusive Group technique is proposed to reduce the cost of query execution generated due to nested loop fashion joins. Bound Join technique is proposed to reduce the number of request by grouping a set of mappings in a single sub query. Using these techniques SPARQL query is evaluated on two different datasets Cross Domain (CD) and Life Science (LS). Evaluation shows that results of complex query are improved and it outperformance for simple query due to its expensive techniques.

VII. USING GRAPH TRAVERSAL ALGORITHMS FOR SPARQL QUERY OPTIMIZATION

Graph traversal algorithms can be used in the process of SPARQL query optimization due to its unique feature of representing the whole query as a graph [3, 9]. Basic Graph Pattern (BGP) is represented as a directed graph of subjects as node and predicate as edge. The weight of each edge is calculated as the cost of evaluating the corresponding triple pattern. Graph traversing algorithms generate the optimal query plan corresponds to the minimum spanning tree [9]. In [10] edmonds’ algorithm and prim’s algorithm are discussed over SPARQL query optimization. In [10] two optimization approaches are described. First approach generates a static query execution plan before the query execution using prim’s algorithm [14] or edmonds’ algorithm [15]. To allow each potential solution to follow an independent query plan, the execution plan generated by phase one is altered by an adaptive approach using prim’s algorithm.

Approach presented in [10] have shortcoming like the approach does not keep separate data copies for different iterations, new binding retrieval process does not produce right results. These shortcomings are overcome by the approach presented in [9]. In spite of all the above overcoming, approach of [9] suffers from new challenges like the execution time may be unexpected when executing a complex query having many triple patterns and cannot execute multiple triple patterns in parallel.

VIII. CONCLUSION

With the development of Linked Open Data techniques SPARQL query optimization has become a new research branch of semantic web. It has got more concern form the large amount of international researchers to fit those techniques in practical environment. Optimization techniques of distributed database system have also been studied to relate it with distributed SPARQL query optimization but it cannot be applied because of the difference between data representation (i.e. relations and triples). In spite of all the techniques mentioned in this paper, SPARQL query is still at the early stage for both scalability and efficiency. There is a lake of application for proposed approach like generating context graph with query analysis technique. Co-reference issue in linked data cloud is still remaining a big challenge. Current approaches only optimize some part of the query evaluation process. In future it can be extended to optimize the entire query evaluation process.
By concluding all the above approaches there is a room for graph theory based techniques because it can provide better performance and scalability over other approaches. All the current techniques focus on domain specific work so it can also be extended to make it generic for all domains.

IX. REFERENCES:
