IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 8, Issue 6 (Jan. - Feb. 2013), PP 97-101
www.iosrjournals.org

Efficient Optimization of Multiple SPARQL Queries

R.Gomathi[1], C.Sathya[2] ,D.Sharmila[3]
[1](*Assistant Professor(Sr.G)/ComputerScience ,Bannari Amman Institute of Technology, Sathyamangalam,India)*
[2](*PG Scholar/Computer Science , Bannari Amman Institute of Technology, Sathyamangalam,India)*
[3](*Professor & Head/Electronics and Instrumentation , Bannari Amman Institute of Technology, Sathyamangalam,India)*

Abstract : *A W3C standard for processing RDF data is a SPARQL query language, a technique that is used to encode data in meaningful manner. We investigate the foundations of SPARQL query optimization by grouping into individual clusters using common substructures in the multiple SPARQL queries, propose a comprehensive set of query rewriting rules for the clustered group and finally Query execution provide the final result of optimized query. The proposed technique is efficient and scalable for multiple SPARQL query.*

Keywords - *RDF, Multiple SPARQL, Common substructures*

I. INTRODUCTION

RDF is the data format of interlinked data. RDF is a directed, labeled graph data format for representing information in the Web. RDF is an essence of triple format namely subject, predicate and object.

Subject
Predicate
Object

Figure 1.1 Triple Format Representation

SPARQL, a query language and a protocol for retrieving RDF data which has been formulated and designed by the W3C RDF Data Access Working Group. SPARQL is a query language for pattern matching for RDF graphs. SPARQL syntax is similar to SQL, but SPARQL is more powerful, which enables queries spanning multiple disparate (local or remote) data sources containing heterogeneous semistructured data.

The SPARQL query language is related to the following specifications:

- The SPARQL Protocol for RDF [SPROT] specification defines the remote protocol for issuing SPARQL queries and receiving the results.
- The SPARQL Query Results XML Format [RESULTS] specification defines an XML document format for representing the results of SPARQL SELECT and ASK queries.

SPARQL takes the description of what the application wants, in the form of a query, and returns that information, in the form of a set of bindings or an RDF graph.

Query optimization is the most critical phase in query processing. Multi Query optimization is a technique in which multiple query plans for satisfying a query are examined and a good query plan is identified. Complexity arises in MQO which leads to NP-Hard. There may be many plans to find the best strategy. Cost based query optimizers evaluate the resource of various query plans and use the basis for plan selection using algorithms. The search space can become quite large depending on the complexity of the SPARQL query.

Complex queries are becoming common, due to the advent of technological tools that help examine information from large data stores. These complex queries share a lot of common sub-expressions since i) extensive views for different query that share a common value ii) There are nested queries that are correlated where outer query and inner query variables are not common but form a common sub-expression.

Keeping the above challenges we design a framework for MQO with the following contributions:

1. Summarize the similar pattern in the SPARQL query.
2. Summarized patterns will be clustered based on the common substructures.
3. Clustered queries will be rewritten and finally query execution is performed.
4. Experiments prove that the model is very efficient and scalable.

II. RELATED WORK

Complex queries are becoming common in decision support systems. These complex queries have a lot of common sub-expressions [1], either within a single query, or multiple queries. Multiquery optimization exploits common sub-expressions to reduce evaluation cost. Three cost-based heuristic algorithms: Volcano-SH and Volcano-RU, which are based on simple modifications to the Volcano search strategy, and a greedy heuristic is used`. A performance study of comparing the algorithms, using workloads consisting of queries from the TPC-D benchmark. The study shows that algorithms provide significant benefits over traditional optimization, at a very acceptable overhead in optimization time.

The problem of *Basic Graph Pattern* (BGP) optimization for SPARQL queries and *main memory* graph implementations of RDF data is formalized. The characteristics of heuristics for selectivity based static BGP optimization is studied. Customized summary statistics for RDF data enable the selectivity estimation [2] of *joined* triple patterns and the development of efficient heuristics. Using the Lehigh University Benchmark (LUBM), the performance of the heuristics for the queries provided by the LUBM is discussed.

Efficient management of RDF data is an important factor in realizing the Semantic Web vision. Drawbacks are becoming increasingly pressing as Semantic Web technology is applied to real-world applications. Current data management solutions for RDF data does not scale properly, and explore the fundamental scalability limitations [3] of these approaches is examined. Improving performance for RDF databases using "property tables" is analysed. Vertically partitioning approach is used to study the RDF data. Further, column-oriented DBMS is used which shows an increase in performance magnitude, with query processing time is reduced.

The salient points of RDF-3X are: 1) a generic solution for storing and indexing RDF triples 2) a powerful yet simple query pro- cessor that leverages fast merge joins to the largest possible extent, and 3) Choosing optimal join orders is executed using query optimizer through which a cost model based on statistical synopses for entire join paths is identified. The performance of RDF-3X, [4] in comparison to the previously best systems, has been measured on several datasets with more than 50 million RDF triples and benchmark queries that include pattern matching and long join paths in the underlying data graphs.

BitMat introduces –(i) a compressed bit-matrix structure for storing huge RDF graphs, and (ii) a novel, light-weight SPARQL join query processing method that employs an initial pruning technique along with variable-binding-matching algorithm on BitMats [5] to produce the final results. Query processing method does not build intermediate join tables and works directly on the compressed data. Results show that the competing methods are most effective with highly selective queries. On the other hand, BitMat delivers 2-3 orders of magnitude better performance on complex queries over massive data.

Loosely-structured Exploratory queries requires only minimal user knowledge of the source network. Exploratory query evaluation usually [6] involves the evaluation of many distributed queries. The optimization problem for exploratory queries is overcome by proposing several multi-query optimization algorithms that compute a global evaluation plan which minimizes the total communication cost, a major bottleneck in distributed queries. The algorithms proposed are necessarily heuristics, as computing an optimal global evaluation plan is shown to be np-hard. Finally, an implementation of our algorithms and its illustrations shows their potential not only for the optimization of exploratory queries, but also for the multiquery optimization of large set queries is presented.

A set of novel query processing techniques, for large number of XML stream queries involving value joins over multiple XML streams [7] and documents referred to as Massively Multi-Query Join Processing techniques is proposed. These techniques enable the sharing of representations of inputs to multiple joins, and the sharing of join computation. These techniques are also applicable to relational event processing systems and publish/subscribe systems that support join queries. Experimental results to demonstrate the effectiveness of our techniques is presented. Thousands of XML messages with hundreds of thousands of join queries on real RSS feed streams is processed. Techniques gain more than two orders of magnitude speedup compared to the naive approach of evaluating such join queries.

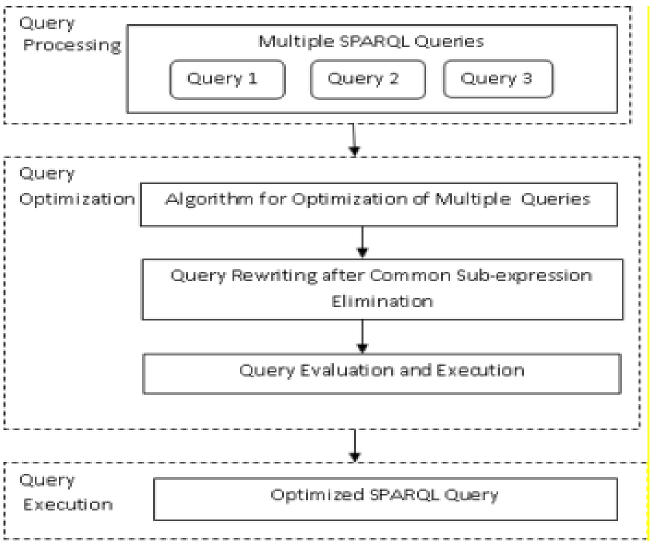
Queries with common sequences of disk accesses can make maximal [8] use of a buffer pool. We developed a middleware to promote the necessary conditions in concurrent query streams, and achieved a speedup of 2.99 in executing a workload derived from the TCP-H benchmark.

Clustering is the unsupervised classification of patterns into groups (clusters). The clustering problem [9] has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. Clustering is a difficult problem in common. An overview of pattern clustering methods is presented. A taxonomy of clustering techniques, identifying cross-cutting themes and recent advances is also proposed. It also describes some important applications of clustering algorithms such as segmentation of the image, recognition of object, and information retrieval of information .

III. PROPOSED ARCHITECTURE

MultiQuery Optimization mainly involves Query Processing, Query Rewriting and Execution as shown in the figure 3.1.

Query Processing: Query Processing converts the SPARQL query into query graph pattern which is equivalent to the query. This query graph pattern presents the query execution in sequence and optimization of the query takes place.


Query Processing
Multiple SPARQL Queries
Query 1
Query 2
Query 3
Query Optimization
Algorithm for Optimization of Multiple Queries
Query Rewriting after Common Sub-expression Elimination
Query Evaluation and Execution
Query Execution
Optimized SPARQL Query
Figure 3.1 Steps in MultiQuery Optimization

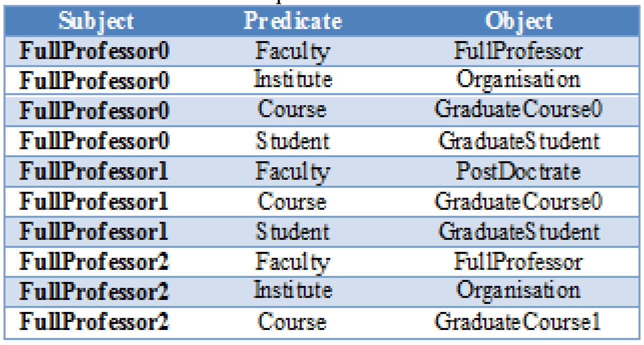
Query Optimization and Execution: Selecting the best strategy for query processing is Query Optimization. Algorithm finds the best query processing strategy. The steps involved in the algorithm are as follows:
Step 1: The input query is partitioned into clusters using K-means clustering.
Step 2: Clusters are formed based on the common sub-expression in the queries that are provided as Input.
Step 3: Formed clusters are rewritten into either of Sample 1 and Sample 2 query pattern.
Step 4: The rewritten query is distributed to the input query and the result is Optimized SPARQL Query.

IV. ILLUSTRATION

A pattern matching query recommended by W3C is SPARQL. There are two types of query variations we focus on:

Sample 1: Q := SELECT OP WHERE TP.
Sample 2: Q_{OPT} := SELECT OP WHERE TP (OPTIONAL TP_{OPT})$^{+}$

where OP is the Output Result and TP is the set of Triple Pattern. Let D be the data graph, and TP searches the triple pattern in D. The difference between the two queries is the OPTIONAL clause.



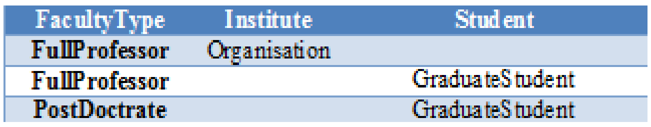
Subject	Predicate	Object
FullProfessor0	Faculty	FullProfessor
FullProfessor0	Institute	Organisation
FullProfessor0	Course	GraduateCourse0
FullProfessor0	Student	GraduateStudent
FullProfessor1	Faculty	PostDoctrate
FullProfessor1	Course	GraduateCourse0
FullProfessor1	Student	GraduateStudent
FullProfessor2	Faculty	FullProfessor
FullProfessor2	Institute	Organisation
FullProfessor2	Course	GraduateCourse1

Figure 4.1 Input Data D.

Consider the data in the table (4.1) and the SPARQL query in the figure(4.2). The query corresponds to triples Faculty and Course which has an equivalent value GraduateCourse0 as object. First OPTIONAL field in the query returns object of predicate Institute, if predicate exists. Second OPTIONAL field in the query returns object of predicate Student, if its predicate exists correspondingly. When the query is evaluated over the Input data D, it results in Q_{OPT} as shown in the figure 4.3.

```
SELECT ?FacultyType ?Institute ?Student
WHERE {?x FacultyType ?Faculty ?x Course GraduateCourse0,
          OPTIONAL{ ?x Institute ?Organisation }
          OPTIONAL { ?x Student ?GraduateStudent }}
```

Figure 4.2 Example Query Q_{OPT}



FacultyType	Institute	Student
FullProfessor	Organisation	
FullProfessor		GraduateStudent
PostDoctrate		GraduateStudent

Figure 4.3 Output Q_{OPT}(D)

Graphically the query graph pattern in the figure 4.4 will consists of four tuples : V- Vertices, E-Edges, Constants and Variables. Vertices represents the subject and object of the triple pattern, gray vertices represent the constants, white vertices represent variables. Predicates are represented in Edges, Dashed edges represent predicates with OPTIONAL graph pattern Q_{OPT} and solid edges represent required graph pattern Q.

The main problem of MultiQuery Optimization is to set query of Sample 1, compute a new set of Q_{OPT} of Sample 1 or Sample 2 queries. There are two requirements for rewriting a queries: It may produce the same result for both Q and Q_{OPT} or the the query evaluation time should be low.

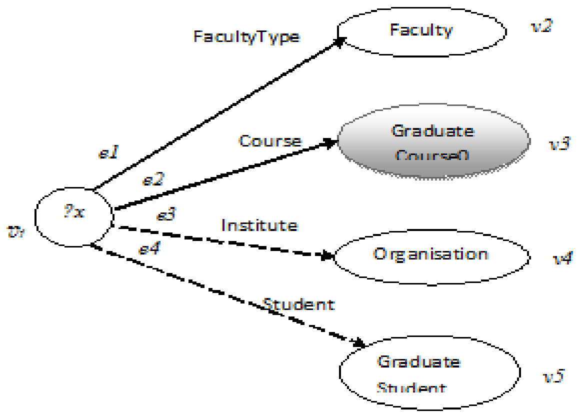


Figure 4.4 A Query Graph

Illustration of Multi Query Optimization is shown in the figures. Figure 4.5(a) to 4.5(d) shows the graph patterns of 4 queries and 4.5(e) shows the graph pattern that rewrites all the 4 queries into a single query. ?xY?p and ?qY?p is the common sub-expression in the figures (a) to (d). These common sub-expression will be rewritten in the figure 4.5(e) using OPTIONAL clause.

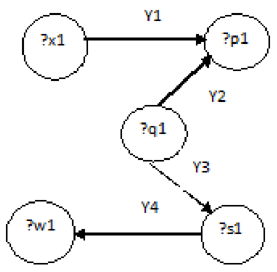


Figure 4.5(a) Input Query Q1

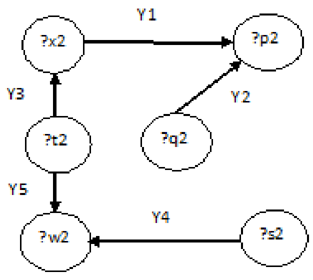


Figure 4.5(b) Input Query Q2

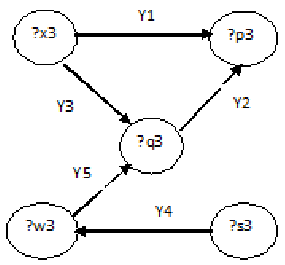


Figure 4.5(c) Input Query Q3

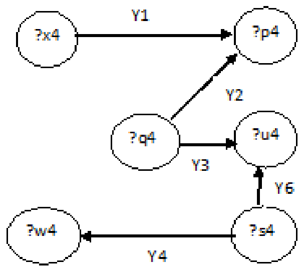


Figure 4.5(d) Input Query Q4

*SELECT **
WHERE { ?xY1?p,?qY2?p,
OPTIONAL {?qY3?s, ?sY4?w}
OPTIONAL {?tY3?x,?tY5?w,?sY4?w}
OPTIONAL {?xY1?p,?wY5?q,?sY4?w}
OPTIONAL {?qY3?u, ?sY6?u,?sY4?w}}

Figure 4.5(e) Example query for Q_{OPT}

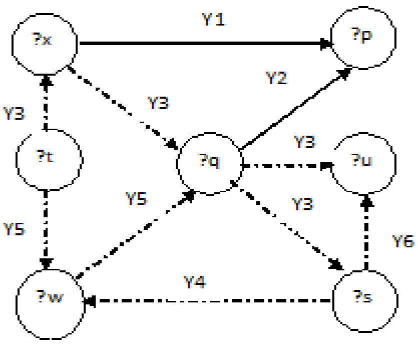


Figure 4.5(f) Query Graph Pattern for Q_{OPT}

DATASET

Lehigh University Bench Mark (LUBM) dataset is used for evaluation. This benchmark dataset describes universities with students and departments with limited predicates. LUBM dataset limits the complexity of SPARQL queries.

V. CONCLUSION

The problem of MultiQuery Optimization for SPARQL has been studied. Proposed Architecture involves an algorithm to identify the common sub-expression and partitioned the input set of queries into clusters. The clusters are rewritten to evaluate and finally optimized query is obtained. The algorithm provides an efficient, effective and scalable optimization technique. Future work can be extended to general graph databases.

Acknowledgements

Mrs R.Gomathi completed her under graduation in the year 2003 and post graduation in the year 2011. She is pursuing her Doctorate in Anna University, Chennai. At present she is working as an Assistant Professor(Sr.Grade) in Bannari Amman Institute of Technology, Sathyamangalam, Erode Dt. She has over 10 years of teaching experience. She has published her papers in 2 International conferences, 3 National Conferences and an International Journal.

Mrs C.Sathya completed her B.Tech (Information Technology) in the year 2007 from Hindusthan College of Engineering and Technology, Coimbatore. She is pursuing her M.E(Computer Science) from Bannari Amman Institute of Technology, Sathyamangalam. She has published her paper in an International Conference.

Dr D.Sharmila completed her under graduation in the year 1996 and post graduation in the year 2004. She has been awarded Doctorate in the year 2010 from Anna University, Chennai. At present she is working as Professor and Head of Electronics and Instrumentation Engineering in Bannari Amman Institute of Technology, Sathyamangalam. She has over 16 years of teaching experience. She has published her papers in 2 National and 5 International Journals. She has also presented her papers in 9 National and 8 International Conferences.

REFERENCES

[1] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and extensible algorithms for multi query optimization. In *SIGMOD*, 2000.
[2] M. Stocker, A. Seaborne, and A. Bernstein. SPARQL basic graph pattern optimization using selectivity estimation. In *WWW*, 2008.
[3] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *VLDB*, 2007.
[4] T. Neumann and G. Weikum. RDF-3X: a RISC-style engine for RDF. In *PVLDB*, 2008.
[5] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler. Matrix ”bit” loaded: A scalable lightweight join query processor for RDF data. In *WWW*,2010.
[6] A. Kementsietsidis, F. Neven, D. V. de Craen, and S. Vansummeren. Scalable multi-query optimization for exploratory queries over federated scientific databases. *PVLDB*, 2008.
[7] M. Hong, A. J. Demers, J. Gehrke, C. Koch, M. Riedewald, and W. M. White. Massively multi-query join processing in publish/subscribe systems. In *SIGMOD*, 2007.
[8] K. O'Gorman, D. Agrawal, and A. E. Abbadi. Multiple query Optimization by cache-aware middleware using query teamwork. In *ICDE*, 2002.
[9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review.*ACM Comput. Surv.*, 1999.
[10] M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL query optimization. In *ICDT*, 2010.