Review of Contemporary Literature on Distributed-RDF Query Optimization Techniques

K.SHAILAJA[1] **DR. P.V. KUMAR**[2] **DR S.DURGA BHAVANI**[3]

[1]*Department of CSE, Methodist College of Engineering and Technology, Hyderabad, Telangana State, India*
[2]*UCE, OU, HYDERABAD, Telangana State*
[3]*School of IT, JNTUH Hyderabad, Telangana State, India*
Email: [1] shailajamtech2006@yahoo.co.in, [2] pvkumar58@gmail.com, [3] sdurga.bhavani@gmail.com

Abstract: In a distributed environment such as the Semantic Web, the knowledge exploration and discovery with RDF data poses unique challenges with the data volumes rising exponentially. The query engines of RDF data optimized in terms of the order of joining the partial query results generate an efficient query process and best fit solutions. The challenges faced in the RDF chain query optimization process are many such as the possible huge increase in the query paths generated proportionate to the increase in query size. The varying efficiency obtained with each query path is another problem. The cost of determining the query paths has to be controlled for incurring a minimal overhead and according to the allocation criteria. The optimization of query paths for RDF data has been studied widely and many approaches devised have attempted to solve this issue comprehensively. This manuscript portrayed a comprehensive review on the contemporary approaches of the RDF query optimization techniques.

Keywords: *RDF chain query optimization, ant colony optimization, genetic algorithm, iterative improvement, simulated annealing.*

1 Introduction

The Semantic Web [1] is capable of processing enormous data volumes and is capable of handling today's rapidly increasing data flow challenges and complicated problems of efficient data exploration, comprehension, and retrieval [2]. This enables the technology in making competent decisions compared to the contemporary web technology. Semantic web is now an efficient alternative to the existing web technology.

The Semantic Web achieves comparative efficiency because of its capability of storing several different sources of linked data. The interrelated and enormously huge linked data is depicted using a new and efficient data source called RDF (Resource Description Framework). The RDF data source is capable of efficient machine data interpretation of the linked data sources which enables effective data description and metadata [3] exchange. The integration of numerous sources of data possible with the technology is efficient for precise and individual information requirements.

The efficiency of querying the widely distributed and huge RDF data sources is achieved with fast RDF query engines using RDF Query Language (SPARQL) and the SPARQL Protocol [4]. A SPARQL query process has individual queries comprising of several sub-queries which query multiple sources simultaneously. In terms of the query specific complete requirements the query results are merged and the execution order of the distinct query parts determines the efficiency of the query. The query paths efficiency in optimizing the execution time is vital for real-time environments where huge user specific requirements make semantic web technology highly significant.

This manuscript comprises the comprehensive review on RDF query optimization methods observed in contemporary literature.

.

2 Review of Literature

An initial approach devised for RDF optimization is based on algorithm 2PO (two-phase optimization) [5] and applies an iterative improvement technique. The next method devised is the SA (Simulated Annealing) based approach. The contemporary methods devised for handling RDF query chain optimization found to be significant towards quality solutions are the GA (Genetic Algorithm) based method and, the ACO (Ant Colony Optimization) [2] method.

These techniques have been limited merely a single source [2], [6]. In contrast to these methods several heterogeneous sources are usually involved in an RDF query process. The Semantic Web is a wide-area network (WAN) or a distributed setting where the data sources joined are distributed far away from each other rather than being in a single hard disk memory.

The elementary models related with effective processing of SPARQL queries [7] is contributed in this literature. Here, the study is performed on (a) the intricacy analysis of entire operators in the query language of SPARQL (b) SPARQL algebra equivalences (c) algorithm aimed at optimizing "semantic SPARQL queries". Complexity analysis envisages that all SPARQL fragments come under the NP category.

Huge quantity of the RDF data often demands query optimization that combines the partial query outcomes. The work [8] presents that ACS (ant colony system) is proposed in this literature for effective query in the environment of semantic web. The development in the solution costs is compared with contemporary algorithms such as "two-phase optimization (2PO) [6]" & Genetic Algorithm (GA) [5]. It is proved that ACS method performs better than contemporary approaches.

The work [5] presents that for optimizing the SPARQL queries known as "RDF-chain queries", a novel GA is devised known as "RCQ-GA [5]". Here, chain queries are optimized by the algorithm by identifying the order which joins required to be executes. The algorithm performance is compared with 2-phase optimization & results presents the solution quality and consistency.

The work [9] presents that "cardinality estimation centric join ordering algorithm" is one that targeted for optimizing the SPARQL query join order to attain minimal performance time. This method has performance analysis, which is conducted on arbitrary & star queries. Here, the metric search optimization is deliberated in this method and also particular to scan "single RDF store".

The work [10] presents that "PSO (particle swarm optimization)" is the other evolution scheme that was utilized for optimizing distributed data queries base-context. As movement of particle is in framework of "probability distribution computational complexity" of this method is also not linear. The method is not explained as particular to the "Distributed RDF & optimality" changes as per the set of parameters.

The work [11] presents "parallel join algorithm" that is ensemble of 3 algorithms, which combines multiple queries in bounded manner. Here, cost of access is not deliberated in respect to optimize cost of join as considerable confine of the method, but this method is vigorous towards the size of data.

The work [12] presents querying graphs known as G-SPARQL, the other standard method in this literature. The G-SPARQL divides query into the subqueries & executes every subquery into the framework of optimal usage of memory. Here, computational complexity of this method is non-linear and is deficient in optimizing the search space & access cost.

The work [13] presents the “SPARQL query processing” method, which utilizes the index structure for optimizing search space to scan RDF triples is suggested. This method is depended on structure of tree for optimizing the query joins. Here, this method is vigorous, as the cost of access is not considered as factor for optimizing the join cost.

The work [14] presents “Map Reduce framework” devised in framework of reducing response time. Algorithm, which devised for assisting this context is “All-Possible-Join tree (APJ-tree)”. In respect to optimize search-space, bloom filter is utilized by the APJ-Tree algorithm. Similar to standard methods explored, this method does not consider the cost of access as the parameter for the optimization of join cost and appears as robust for query scanning on the “single RDF-store”.

The confines, which are noticed in all these contemporary methods are that optimizing queries in the environment of single source and not showing the optimal performance and its robustness in the query chains through huge amount of the joins. Further, the cost of access is not deliberated as parameter for optimizing the join cost.

Parallel query optimization can be termed as the serial optimization algorithms generating plans, which are executed in parallel [15] for query processing. Though contextually there are varied considerations in realizing the term, in the current scenario a parallel algorithm for generating query plans are discussed.

In [16] the authors have focused on problem of the load-balancing through restricting the URIs storage & literals, depending on storage capacity of local-peers. Though the process might provide the desired outcome, still the challenge is about probable loss of complete result. In [17] the study has targeted to address the issue by developing an intersection tree over DHT position of an overlay well-known triple component. Such kind of the load-balancing is delicate in instance of failure of node in the overlay tree, and could even result in loss of the complete branch of tree.

In [18], the study has worked on huge variation amid peer’s data load upon triples being indexed 3 times by fixed hash depth of 1. It envisaged that with hash depth maximum value, the better is distribution of triples amid peers. But it comes with cost of network communication in the instance of query evaluation, as many peers are queried for assessment of tripartite pattern.

In [19], for addressing the biased load balancing hotspot, the solution in respect to indexing a triple aimed at each possible combination of its 2 components such as subject + object, subject + predicate, predicate + object and such type of combination are suggested. The effect of such procedure is the triples extra storage in network, regardless of attaining fair distribution of triple load.

In [20], for enhancing the distribution of query load, study has focused on indexed triples by combination of varied triple elements, resulting in 7 replications for every triple in total. Here, it is utilized in storage overhead aimed at distributing the query processing load amid peers. However, the key element missing was about studying the overhead usage for enhancing response time for query processing.

In [21], the notion of quad is used for representing RDF data & a solution with optimized index structure aimed at assisting RDF queries assessment in “centralized RDF data stores” are considered.

The study reflects that only 6 indexes are essential for covering the entire range of 16 access patterns, but each access patterns deliberated a quad while there is an integration of predicate, context, subject or object is either variable or defines. Deliberating such opportunity for lessening the number of required indexes, and in segment IV-B only 3 of the indexes are required for new index scheme to ensure coverage of all the 8 possible triple patterns.

The proposed work connects to the earlier works that parallelize classical dynamic program designed with query optimization algorithm [22], [23], [24], [25], [26], [27], and [28]. Many of the prior algorithms were devised for shared-memory structures, which lack scalability over a certain level of parallelism [29]. In the other way, prior algorithms evaluated to consider certain volume, but the algorithm devised in [28] evinced scalability with shared-nothing architecture comprising over 250 workers. Some of the key factors that differentiate prior algorithms to proposed algorithm is the limitation and scalability.

In the earlier algorithms, the query patterns that are independent, cyclic or recursive only planned to execute in parallel as independent threads and there is presumption that all the threads might share common data structures and thus it is easier to access intermediary results generated in the other threads. This practice leads to huge communication overhead over shared-nothing architectures and moreover, these algorithms least significant towards optimizing query patterns found multiple query chains submitted in distributed environment. In addition, the earlier algorithms have relied on central coordinator for assigning rather fine-tuned optimization of tasks to the work threads. Two of the key disadvantages of such process are: i) it needs considerable communication among the master and workers ii) increased levels of time complexity in managing at central system, which leads to delay in implementing parallelism.

The primary objective of the distributed or decentralized RDF systems [30], [31], [32], [33], [34], [35] denote the data projection across the multiple data sources fall in same cluster, which is either in the form of replication or of the partitioned data tuples [36], [37]. However, each of these clusters referred as single end point in regard to RDF querying process. These systems are intended to minimize the processing time taken by the query execution on single end point. The other dimension of the RDF systems that referred as federated RDF systems allows accessing the data through SPARQL queries having independent or remote endpoints, which is since the federated RDF systems are not having control over the data.

Federated SPARQL methods such as Hi-BISCuS [38], ANAPSID [39], SPLENDID [40] are on basis of gathered statistics and information regarding the data held at every endpoint. Cost of addition of novel endpoint is comparative to the data size. The other methods like Lusail [41], [42], and FedX [43] uses the SPARQL-ASK queries in finding the related endpoint, and the cache outcomes of queries are used for future. Hence the cost of start-up and addition of novel endpoint is minor. The Federated SPARQL methods generally separate query into the selected-groups of the triple –patterns; where every group has an outcome only at 1- endpoint. The works [44], [45], [46], [47] have many efforts to concentrate on the “web-based data integration” on the “heterogeneous information sources” [48].

Generally, wrapper is allowed at every source of data to convert among the data models and assisted languages. Furthermore, methods like [49], [50], [51] are the peer-peer methods which connect the heterogeneous network of data-sources. Unlike standard methods, Lusail decays on basis of query for checking the instances of data thus shifting many of the computation intermediate outcomes towards endpoints. Furthermore, “sub-query ordering” procedure in the Lusail is conducted by connecting the outcomes of sub-queries by join-operations; and the cost of communication for executing every sub-query to find the finest ordering that manages among the cost of communication and degree of the parallelism. The work [42] however not planned to search space optimization and available cost optimization.

The contemporary methods of query optimization often achieves by reorganizing the triple pattern sequence, and the other significant parameters towards join ordering were explored here:

The work [5] presents the primary solution in context of RDF databases. They proposed hybrid algorithm called 2-phase optimization (2PO) over the query chain and applied the same contribution over sesame method [52]. The work [53] introduced a model for predicting the cardinality by utilizing the summarization based on the pattern. The model utilized 2 main functions that is pattern & their sub-pattern is almost having similar frequencies and former knowledge of the pattern’s significance. They also utilized dynamic-programming with 2 materialistic solutions. The work [54] presents a mechanism for optimizing static query for rearranging the BGP triple pattern.

The work [6] presents genetic algorithm (GA), which is an optimization algorithm and by tested queries they evinced that the performance of GA is better for the bulky chain query when it is compared with the 2PO. However, 2PO performs better for the small queries through 10 predicates. The work [55] presented RDF-3Xengine through dynamic programming in the form of an algorithm aimed at optimization query. To predict the joins cost among triple patterns, the selectivity histogram is utilized. The work [56] enhanced their former research as contributed in [55] and depicted a model for passing the information among separate joins and at the compiling time, they utilized cumulated statistics for providing triple pattern cardinality exactly. The work [57] applied their scheme in the Atlas system. Here, they utilized 3 materialistic optimization algorithms for reducing the transitional facts size that are produced by the algorithm of query processing utilizing the heuristic based on selectivity. The work [58] presented a model for predicting the cardinality on the basis of characteristic set.

The work [59] presents that a model using GA for optimizing SPARQL query and for generating the plan, bushy trees is utilized. The work [2] studied the issue of query optimization by utilizing an algorithm called ACO over chain-query. The proposed method is compared with [5] & [6] and showed the results superiority over the other algorithms. The work [9] extends the characteristic set for providing a novel “RDF statistical synopsis”, which predicts the cardinalities accurately. The work [60] presented an ACS (Adaptive cuckoo search), which is an optimization algorithm for optimizing SPARQL query. The work [61] presented a novel solution for optimization by utilizing ACO and rearranging the triple patterns & the selectivity prediction introduced by the contribution [54] with some changes. The work [62] presented novel method of join rearranging, which converts query into multiple dimensional space vector and executes distance on the basis of optimization.

The work [63] suggested “distributed RDF framework” that converts SPARQL queries into queries of Pig Latin, which are performed on the “Hadoop Map Reduce framework”. Here, this method is easy for handling the distributed RDF method, which does not need any modifications to Hadoop structure. Optimization schemes employed in the PigSPARQL lessen the quantity of data need to manage and also lessens the respective processing time of query. The suggested method processes SPARQL queries on the “Map reduce cluster” whereas the programs of Pig Latin are performed in the of “Map reduce jobs (MRJ)” series. The suggested method design is able to handle “Big RDF data” as the SPARQL queries scalability is an important problem. For the query optimization of SPARQL, they have utilized the already present variable counting model proposed in [54]. Here, the used mechanism does not considered selectivity of recurrently occurring joins.

The work [64] suggested Scalable, High Performance, Robust and Distributed Triple Store (SHARD) that is constant storage for the data of RDF and offers an end point for processing the queries of SPARQL. The triple store of SHARD is designed for storing the intermediate outcomes for the queries so that later it assists to increase processing of same SPARQL queries. In the SHARD, the map stage maps triple data towards variable bindings that satisfy query first clause. RDF triples were stored in the text files that are stored further on “HDFS (Hadoop Distributed File System)” & Map-reduce is utilized for matching query triple patterns with RDF stored data. The RDF data of SHARD groups is optimized by the subject & MRJ is formed for each TP (triple pattern) in the query of SPARQL. The triple store of SHARD limitation is that it could not execute any kind of the query restructuring or manipulation to enhance the SPARQL queries performance.

The work [65] proposed MR & distributed RDF method based on HBase. The suggested method overcomes the limitation faced generally in MR based systems because of huge quantity of the intermediate query outcomes generated at the time of query-operations. Here, the suggested method utilizes joins of map side for query processing of SPARQL to overcome decrease in the performance faced because of side joins. Besides, the algorithm of job optimization is utilized for lessening the count of jobs in MR framework. Hence, much amount of the triple patterns is assessed in the Map-job. The RDF data is divided over the group and is utilized for creating a filter that is used for further lessening the input size towards map-jobs.

The work [66] suggested new “distributed graph path query processing system” on well-known “Apache Spark framework (ASF)”. They utilize the graph data method for disseminated processing. Besides, they suggested algorithm for graph path queries (GPQ) processing on spark. The GPQ includes the task of detecting the entire matching graph sub-frameworks, which content the issued query of SPARQL. Here, the suggested method is having dependency leverage on relatively faster ASF over the contemporary MR frameworks.

The work [62] suggested a join rearranging method, which considers the relative variances between triple patterns by executing distance on the basis of optimization. This method produces deep trees in the binary left and compared with outdates heuristics & statistics-based methods. Here, triple patterns were mapped into multiple dimensional space vectors. Every triple pattern could be predicted as dimensional vector. The function of distance is implemented to compute distance matrix where the rows depict TPs and columns depict the features in multiple dimensional-space.

The work [41] proposed Lusail system aimed at effective & scalable query processing of SPARQL over “decentralized RDF graphs”. Several optimizations are implemented in system during run and compile times for attaining low response time of query and scalability. The new “locality aware decomposition” is implemented at the compile time such that huge number of TPs in query could be together sent to location of source that contents these TPs. The “selectivity-based and parallel query execution” scheme is implemented during run time for delaying the performance of such sub-queries that are estimated to return maximum result.

2.1 Heuristic methods

A chain query over an RDF data source has numerous query paths in the solution space. However not all of them are equally capable of giving the desired results as the discovery of the best query path especially with least cost associated is involved with several difficulties. The literature in this field has many techniques that have attempted to solve the optimization problem.

2.1.1 Two-Phase Optimization

The two-phase optimization approach based on the algorithm 2PO [67], determines the solutions in two stages, first with iterative improvement (II), and next with simulated annealing (SA). The process of finding the best solution for the chain query with minimal risks of local optimum first involves the stage II and the derived solutions are further improved in the SA stage.

The probing of the solution is started with the iterative improvement step where random starting points or solutions are first determined. Then based on these derived solutions several random neighbors are discovered subsequently. In case the process finds a neighbor whose cost of execution is lower than the previous established, a step is executed to the new found neighbor. A neighbor solution is used to establish the desired solution by using different joining conditions such as, the join commutativity, the join associativity, the left join exchange, and the right join exchange [68]. Here the stage of iterative improvement is applied and continued till the process is unable of finding any other solution with lesser cost or else till the completion of the maximum time set for the phase in the process.

The disadvantage with the iterative improvement technique is the problem of mostly finding the local optima. This problem is overcome with the simulated annealing technique with a strategy of declining probability usually applied only with the best local optimum obtained from the iterative improvement step with which it moves to a neighbor associated with higher cost, considering that maybe later it will discover a neighbor associated with lesser cost local optimum.

2.1.2 Genetic Algorithm

The Genetic Algorithm approach or the GA method [6], [67] is an alternative to the 2PO approach. The strategy of the approach is based on the survival rule of the fit where a number of derived solutions are in a simulation process evolved. A solution's fitness and the inverse relationship related with its cost of execution determine the selection of the best fit solution among many. In this approach a random set of solutions are chosen from the many solutions and these chosen solutions are the first-generation solutions. Then for every new generation proliferation is done with the selection of a small part of the fittest solutions. By combining these solutions with a random selection of solutions of present generation the offspring for the next generation is evolved, where over a small part of the newer generations the last mutation is realized. The probability of selecting a solution is based on the solutions fitness value. The evolution process progresses till completion of the limit set for the number of generations or when any significant value is not achieved with the number of previously set specific generations. Finally, from the last generations solution set the fittest candidate is the solution selected.

2.1.3 Ant Colony Optimization

The technique Ant Colony Optimization (ACO) [69] is devised based on the idea used by ant colonies in food exploration by using pheromone traces to mark paths to food sources. This process when repeated gives paths of shorter lengths to the food source and as these paths are frequently traversed more pheromone is deposited where the less traversed paths see fading of the pheromone’s traces. In this exploration finally the shortest path to the food source from the nest is found by the colony.

The related research study [2] has the technique of ant colony optimization implement chain queries using RDF data sources. This RDF data is applied using a graph-based encoding technique which is an approach dependent on ordinal encoding devised in the paper [67] to perform the joining operations. The technique of ordinal encoding performs joining in iterations of 2 operands, or base relations, or previous join results. Here an ordered list of operands is used to perform the joining of 2 operands, or join results. Next the join formed from 2 operands is saving to a position in the 1st appearing operand. This procedure for realizing the solution applies a process of encoding on the sequentially derived pairs of indices related to the performed join operand, which is one pair considering every join.

3 Conclusions and Future Work

This manuscript portrayed a comprehensive review on RDF query optimization techniques observed in contemporary literature. The observations of this review concluding that the majority of the contemporary query optimization methods are limited to query syntax standards such as SPARQL, and intended to optimize the queries, which are targeting single source of the RDF store. The heuristic methods are indeed optimal to achieve best at all sorts of performance metrics in regard to RDF query optimization. However, it is obvious to conclude from the observations of this comprehensive review that, the contemporary methods are still at their infant phase as such they limited to single source, and depends on RDF query syntax formats such SPARQL. The potential scope can be evinced for future research to portray the heuristic, meta-heuristic, and syntax independent query optimization techniques to deal with distributed RDF stores.

References

[1] Berners-Lee, Tim, James Hendler, and Ora Lassila. "The semantic web." Scientific american 284, no. 5 (2001): 28-37.
[2] Hogenboom, Alexander, Flavius Frasincar, and Uzay Kaymak. "Ant colony optimization for RDF chain queries for decision support." Expert Systems with Applications 40, no. 5 (2013): 1555-1563.
[3] Klyne, Graham, and Jeremy J. Carroll. "Resource description framework (RDF): Concepts and abstract syntax." (2006).
[4] E. Prud’hommeaux and A. Seaborne. SPARQL Query Language for RDF - W3C Recommendation 15 January 2008, 2008.
[5] Stuckenschmidt, Heiner, Richard Vdovjak, Jeen Broekstra, and Geert-Jan Houben. "Towards distributed processing of RDF path queries." International Journal of Web Engineering and Technology 2, no. 2-3 (2005): 207-230.
[6] Hogenboom, Alexander, Viorel Milea, Flavius Frasincar, and Uzay Kaymak. "RCQ-GA: RDF chain query optimization using genetic algorithms." In International Conference on Electronic Commerce and Web Technologies, pp. 181-192. Springer, Berlin, Heidelberg, 2009.
[7] Schmidt, Michael, Michael Meier, and Georg Lausen. "Foundations of SPARQL query optimization." In Proceedings of the 13th International Conference on Database Theory, pp. 4-33. ACM, 2010.

[8] Hogenboom, Alexander, Ewout Niewenhuijse, Frederik Hogenboom, and Flavius Frasincar. "Rcq-Acs: Rdf chain query optimization using an ant colony system." In Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01, pp. 74-81. IEEE Computer Society, 2012.
[9] Gubichev, Andrey, and Thomas Neumann. "Exploiting the query structure for efficient join ordering in SPARQL queries." In EDBT, vol. 14, pp. 439-450. 2014.
[10] Dokeroglu, Tansel, Umut Tosun, and Ahmet Cosar. "Particle Swarm Intelligence as a new heuristic for the optimization of distributed database queries." In 2012 6th International Conference on Application of Information and Communication Technologies (AICT), pp. 1-7. IEEE, 2012.
[11] Senn, Juerg. "Parallel Join Processing on Graphics Processors for the Resource Description Framework." In 23th International Conference on Architecture of Computing Systems 2010, pp. 1-8. VDE, 2010.
[12] Sakr, Sherif, Sameh Elnikety, and Yuxiong He. "Hybrid query execution engine for large attributed graphs." Information Systems 41 (2014): 45-73.
[13] Liu, Chang, Haofen Wang, Yong Yu, and Linhao Xu. "Towards efficient SPARQL query processing on RDF data." Tsinghua science and technology 15, no. 6 (2010): 613-622.
[14] Zhang, Xiaofei, Lei Chen, and Min Wang. "Towards efficient join processing over large RDF graph using mapreduce." In International Conference on Scientific and Statistical Database Management, pp. 250-259. Springer, Berlin, Heidelberg, 2012.
[15] Chekuri, Chandra, Waqar Hasan, and Rajeev Motwani. "Scheduling problems in parallel query optimization." In PODS, pp. 255-265. 1995.
[16] Cai, Min, and Martin Frank. "RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network." In Proceedings of the 13th international conference on World Wide Web, pp. 650-657. ACM, 2004.
[17] Battré, Dominic, Felix Heine, André Höing, and Odej Kao. "On triple dissemination, forward-chaining, and load balancing in DHT based RDF stores." In Databases, Information Systems, and Peer-to-Peer Computing, pp. 343-354. Springer, Berlin, Heidelberg, 2006.
[18] Mietz, Richard, Sven Groppe, Oliver Kleine, Daniel Bimschas, Stefan Fischer, Kay Römer, and Dennis Pfisterer. "A p2p semantic query framework for the internet of things." PIK-Praxis der Informationsverarbeitung und Kommunikation 36, no. 2 (2013): 73-79.
[19] Ali, Liaquat. "Efficient Management and Querying of RDF data in a P2P Framework." PhD diss., Verlag nicht ermittelbar, 2014.
[20] Liarou, Erietta, Stratos Idreos, and Manolis Koubarakis. "Evaluating conjunctive triple pattern queries over large structured overlay networks." In International Semantic Web Conference, pp. 399-413. Springer, Berlin, Heidelberg, 2006.
[21] Harth, Andreas, and Stefan Decker. "Optimized index structures for querying rdf from the web." In Third Latin American Web Congress (LA-WEB'2005), pp. 10-pp. IEEE, 2005.
[22] Han, Wook-Shin, Wooseong Kwak, Jinsoo Lee, Guy M. Lohman, and Volker Markl. "Parallelizing query optimization." Proceedings of the VLDB Endowment 1, no. 1 (2008): 188-200.
[23] Han, Wook-Shin, and Jinsoo Lee. "Dependency-aware reordering for parallelizing query optimization in multi-core CPUs." In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 45-58. ACM, 2009.

[24] Waas, Florian M., and Joseph M. Hellerstein. "Parallelizing extensible query optimizers." In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 871-878. ACM, 2009.
[25] Zuo, Wanli, Yongheng Chen, Fengling He, and Kerui Chen. "Optimization Strategy of Top-Down Join Enumeration on Modern Multi-Core CPUs." JCP 6, no. 10 (2011): 2004-2012.
[26] Chen, YongHeng, and ChunYan Yin. "Graceful Degradation for Top-Down Join Enumeration via similar sub-queries measure on Chip Multi-Processor." Applied Mathematics and Information Sciences 941, no. 3 (2012): 935-941.
[27] Soliman, Mohamed A., Lyublena Antova, Venkatesh Raghavan, Amr El-Helw, Zhongxian Gu, Entong Shen, George C. Caragea et al. "Orca: a modular query optimizer architecture for big data." In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 337-348. ACM, 2014.
[28] Trummer, Immanuel, and Christoph Koch. "Parallelizing query optimization on shared-nothing architectures." Proceedings of the VLDB Endowment 9, no. 9 (2016): 660-671.
[29] Stonebraker, Michael. "The case for shared nothing." IEEE Database Eng. Bull. 9, no. 1 (1986): 4-9.
[30] Abdelaziz, Ibrahim, Razen Harbi, Semih Salihoglu, and Panos Kalnis. "Combining vertex-centric graph processing with SPARQL for large-scale RDF data analytics." IEEE Transactions on Parallel and Distributed Systems 28, no. 12 (2017): 3374-3388.
[31] Harbi, Razen, Ibrahim Abdelaziz, Panos Kalnis, Nikos Mamoulis, Yasser Ebrahim, and Majed Sahli. "Accelerating SPARQL queries by exploiting hash-based locality and adaptive partitioning." The VLDB Journal—The International Journal on Very Large Data Bases 25, no. 3 (2016): 355-380.
[32] Galárraga, Luis, Katja Hose, and Ralf Schenkel. "Partout: a distributed engine for efficient RDF processing." In Proceedings of the 23rd International Conference on World Wide Web, pp. 267-268. ACM, 2014.
[33] Zeng, Kai, Jiacheng Yang, Haixun Wang, Bin Shao, and Zhongyuan Wang. "A Distributed Graph Engine for Web Scale RDF Data." Proceedings of the VLDB Endowment 6, no. 4 (2013).
[34] Lee, Kisung, and Ling Liu. "Scaling queries over big RDF graphs with semantic hash partitioning." Proceedings of the VLDB Endowment 6, no. 14 (2013): 1894-1905.
[35] Huang, Jiewen, Daniel J. Abadi, and Kun Ren. "Scalable SPARQL querying of large RDF graphs." Proceedings of the VLDB Endowment 4, no. 11 (2011): 1123-1134.
[36] Abdelaziz, Ibrahim, Razen Harbi, Zuhair Khayyat, and Panos Kalnis. "A survey and experimental comparison of distributed SPARQL engines for very large RDF data." Proceedings of the VLDB Endowment 10, no. 13 (2017): 2049-2060.
[37] Özsu, M. Tamer. "A survey of RDF data management systems." Frontiers of Computer Science 10, no. 3 (2016): 418-432.
[38] Saleem, Muhammad, and Axel-Cyrille Ngonga Ngomo. "Hibiscus: Hypergraph-based source selection for sparql endpoint federation." In European semantic web conference, pp. 176-191. Springer, Cham, 2014.
[39] Acosta, Maribel, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. "ANAPSID: an adaptive query processing engine for SPARQL endpoints." In International Semantic Web Conference, pp. 18-34. Springer, Berlin, Heidelberg, 2011.

[40] Görlitz, Olaf, and Steffen Staab. "Splendid: Sparql endpoint federation exploiting void descriptions." In Proceedings of the Second International Conference on Consuming Linked Data-Volume 782, pp. 13-24. CEUR-WS. org, 2011.

[41] Abdelaziz, Ibrahim, Essam Mansour, Mourad Ouzzani, Ashraf Aboulnaga, and Panos Kalnis. "Query optimizations over decentralized RDF graphs." In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 139-142. IEEE, 2017.

[42] Abdelaziz, Ibrahim, Essam Mansour, Mourad Ouzzani, Ashraf Aboulnaga, and Panos Kalnis. "Lusail: a system for querying linked data at scale." Proceedings of the VLDB Endowment 11, no. 4 (2017): 485-498.

[43] Schwarte, Andreas, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. "Fedx: Optimization techniques for federated query processing on linked data." In International semantic web conference, pp. 601-616. Springer, Berlin, Heidelberg, 2011.

[44] Ambite, José Luis, and Craig A. Knoblock. "Flexible and scalable query planning in distributed and heterogeneous environments." In Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems, pp. 3-10. AAAI Press, 1998.

[45] Duschka, Oliver M., and Michael R. Genesereth. "Query planning in infomaster." In SAC, vol. 97, pp. 109-111. 1997.

[46] Roth, Mary Tork, and Peter M. Schwarz. "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources." In VLDB, vol. 97, pp. 25-29. 1997.

[47] Tomasic, Anthony, Louiqa Raschid, and Patrick Valduriez. "Scaling heterogeneous databases and the design of DISCO." PhD diss., INRIA, 1995.

[48] Ouzzani, Mourad, and Athman Bouguettaya. "Query processing and optimization on the web." Distributed and Parallel Databases 15.3 (2004): 187-218.

[49] Halevy, Alon Y., Zachary G. Ives, Peter Mork, and Igor Tatarinov. "Piazza: data management infrastructure for semantic web applications." In Proceedings of the 12th international conference on World Wide Web, pp. 556-567. ACM, 2003.

[50] Franconi, Enrico, Gabriel Kuper, Andrei Lopatenko, and Ilya Zaihrayeu. Queries and updates in the coDB peer to peer database system. University of Trento, 2004.

[51] Bonifati, Angela, Elaine Chang, Terence Ho, Laks V. Lakshmanan, Rachel Pottinger, and Yongik Chung. "Schema mapping and query translation in heterogeneous P2P XML databases." The VLDB Journal—The International Journal on Very Large Data Bases 19, no. 2 (2010): 231-256.

[52] Broekstra, Jeen, Arjohn Kampman, and Frank Van Harmelen. "Sesame: A generic architecture for storing and querying rdf and rdf schema." In International semantic web conference, pp. 54-68. Springer, Berlin, Heidelberg, 2002.

[53] Maduko, Angela, Kemafor Anyanwu, Amit Sheth, and Paul Schliekelman. "Estimating the cardinality of RDF graph patterns." In Proceedings of the 16th international conference on World Wide Web, pp. 1233-1234. ACM, 2007.

[54] Stocker, Markus, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, and Dave Reynolds. "SPARQL basic graph pattern optimization using selectivity estimation." In Proceedings of the 17th international conference on World Wide Web, pp. 595-604. ACM, 2008.

[55] Neumann, Thomas, and Gerhard Weikum. "RDF-3X: a RISC-style engine for RDF." Proceedings of the VLDB Endowment 1, no. 1 (2008): 647-659.

[56] Neumann, Thomas, and Gerhard Weikum. "Scalable join processing on very large RDF graphs." In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 627-640. ACM, 2009.

[57] Kaoudi, Zoi, Kostis Kyzirakos, and Manolis Koubarakis. "SPARQL query optimization on top of DHTs." In International Semantic Web Conference, pp. 418-435. Springer, Berlin, Heidelberg, 2010.

[58] Neumann, Thomas, and Guido Moerkotte. "Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins." In 2011 IEEE 27th International Conference on Data Engineering, pp. 984-994. IEEE, 2011.

[59] Ouyang, Dantong, Xiaolong Wang, Yuxin Ye, and Xianji Cui. "A GA-based SPARQL BGP reordering optimization method." Advances in Information Sciences and Service Sciences 4, no. 9 (2012): 139-147.

[60] Gomathi, Ramalingam, and Dhandapani Sharmila. "A novel adaptive cuckoo search for optimal query plan generation." The Scientific World Journal 2014 (2014).

[61] Kalayci, Elem Guzel, Tahir Emre Kalayci, and Derya Birant. "An ant colony optimisation approach for optimising SPARQL queries by reordering triple patterns." Information Systems 50 (2015): 51-68.

[62] Meimaris, Marios, and George Papastefanatos. "Distance-Based Triple Reordering for SPARQL Query Optimization." In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 1559-1562. IEEE, 2017.

[63] Schätzle, Alexander, Martin Przyjaciel-Zablocki, and Georg Lausen. "PigSPARQL: Mapping SPARQL to pig latin." In Proceedings of the International Workshop on Semantic Web Information Management, p. 4. ACM, 2011.

[64] Rohloff, Kurt, and Richard E. Schantz. "High-performance, massively scalable distributed systems using the MapReduce software framework: the SHARD triple-store." In Programming support innovations for emerging distributed applications, p. 4. ACM, 2010.

[65] Oh, Hyunsuk, Sejin Chun, Sungkwang Eom, and Kyong-Ho Lee. "Job-optimized map-side join processing using mapreduce and hbase with abstract RDF data." In 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 1, pp. 425-432. IEEE, 2015.

[66] Balaji, Janani, and Rajshekhar Sunderraman. "Distributed graph path queries using spark." In 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 326-331. IEEE, 2016.

[67] Steinbrunn, Michael, Guido Moerkotte, and Alfons Kemper. "Heuristic and randomized optimization for the join ordering problem." The VLDB Journal—The International Journal on Very Large Data Bases 6, no. 3 (1997): 191-208.

[68] Ioannidis, Yannis E., and Younkyung Kang. "Randomized algorithms for optimizing large join queries." ACM Sigmod Record 19, no. 2 (1990): 312-321.

[69] Dorigo, Marco, and Mauro Birattari. "Thomas St utzle," Ant Colony Optimization," PHI and MIT press2006 (2004).