CoPMP: Correlated Predicates Merging and Partition for SPARQL Query Optimization

Hongshen Yu[1], Tenglong Ren[1], Xiaowang Zhang[1,2,*], Lulu Yang[1] and Guopeng Zheng[1]

[1] College of Intelligence and Computing, Tianjin University, Tianjin 300350, China

[2] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China

Abstract

Characteristic sets (CS) are used for storage and indexing, as the same CS tends to have a similar schema. While many methods based on CSs can improve query performance, especially for star queries, query performance will be seriously affected when the workload is complicated and produces many CSs. In this paper, we present CoPMP, merging CSs based on the correlations between predicates within the CSs and partitioning predicates. Our method captures the predicate correlation and merges CSs to reduce the number of CSs. The merging operation is driven by the cost model considering the predicate correlation and null values. In merging CSs, each predicate belongs to only a property set, i.e., predicate partitioning. Thus, CoPMP has the advantages of both property table and vertical partitioning for query optimization. We allocate merged tables into data blocks based on subject hash partitioning, considering that CS is from the same subject. Our extensive evaluation demonstrates the efficiency and scalability of our system.

Keywords

RDF Data, Merge Characteristic Sets, Predicate Correlation, Predicate Partitioning

1. Introduction

Traditional RDF data storage schemes include triples store, vertical partitioning, and property table. But these methods are not efficient enough when workloads are complicated. Recently works show that Characteristic sets (CS) [1] can capture the implicit schema of RDF data and improve optimization. The CS for a subject is the set of predicates on the outgoing edges from that subject. For the complicated datasets, Papastefanatos et al. [4] point out that merging CSs based on the hierarchical structure reduces the number of CSs. But it only works in subset inclusion relations between property sets. In this paper, we introduce CoPMP, a distributed query engine based on Spark, which combines predicate correlation with partitioning and

ISWC'22: The 21th International Semantic Web Conference, October 23–27, 2022, Hangzhou

*Corresponding author.

hs_yu@tju.edu.cn (H. Yu); tenglongren@tju.edu.cnÍÍ (T. Ren); xiaowangzhang@tju.edu.cn (X. Zhang); luluyang@tju.edu.cn (L. Yang); guopengzheng@tju.edu.cn (G. Zheng)

allocates tables to data partitions by subject. [7] also considers predicate partition. But it is not the same as the focus of this paper. Firstly, [7] artificially sets a co-occurrence threshold and uses a similarity coefficient similar to Jaccard to represent the degree of co-occurrence between two predicates. Only when co-occurrence is greater than the threshold are two predicates considered co-occurrence. In this paper, Two predicates are considered related as long as they have the same subject. Secondly, When dividing predicates, [7] stores all predicates that co-occur with predicates, that is predicates with a degree of co-occurrence greater than a threshold, into a set. However, based on CSs, this paper selects the CS with the greatest benefit as the basic data table and tries to add all the predicates with the same subject as the predicates in the selected CS. If the benefit of the table increases, then add the predicate. Finally, To obtain a better predicate partitioning pattern, [7] enumerates different co-occurrence thresholds. In this paper, CSs are merged based on the heuristic cost function to obtain the optimal relation pattern, which is the same idea as [4], but the method is different.

Figure 1 illustrates an overview of CoPMP architecture. Our contributions are as follows:

- We propose a new heuristic algorithm for merging CSs, which considers the correlation of predicates and null values of the merged table.
- We ensure predicate partition in merging CSs, allowing CoPMP to benefit from vertical partitioning and property tables without additional storage structures.
- Extensive experiments on synthetic and real-world RDF graphs have been conducted to verify the efficiency and scalability of our method.

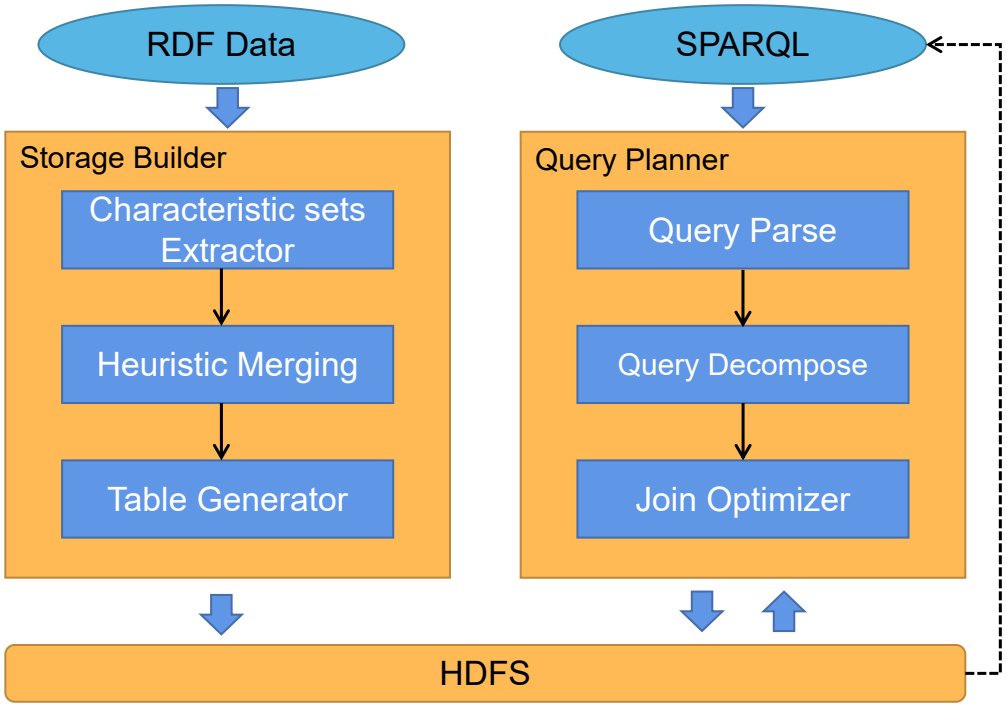

Figure 1: CoPMP Architecture

2. CoPMP

Consider two extremes, (i) a CS corresponds to a property table, and (ii) all CSs are merged into a property table. One leads to many tables and joins, and the other to excessive NULLs. Thus, we need to find a Pareto equilibrium between fewer tables and null values when we merge CSs based on predicate correlation.

Algorithm 1: CSsMerge

Input: Characteristic Sets CSs, *PC*, *PF*, a property table set $pts \leftarrow \emptyset$;
Output: schemaSet
for *each schema ∈CSs* **do**
    pts.add(schema);
    compute the cost of the schema;
end
while *!pts.isEmpty()* **do**
    schema remove predicate p, if p is marked;
    select a schema with the lowest cost $cost_{min}$;
    select a predicate pmax with max Predicate Frequency;
    pts.remove(schema);
    for *each predicate p and PC(p, pmax) = true* **do**
        let schemaTmp = schema.add(p);
        compute the $cost_{new}$ of schemaTmp;
        if $cost_{new} < cost_{min}$ **then**
            schema = shemaTmp; $cost_{min} = cost_{new}$;
            mark predicate p;
        end
    end
    schemaSet.add(schema);
end
return schemaSet;

However, enumerating the combination of all related predicates is computationally hard. For this reason, we rely on a heuristic algorithm for approximating the problem. We introduce a function to find the current minimum cost in the merging process. Predicate frequency $PF(p_i)$ is the number of sets(CSs) in which p_i appear. Predicate co-occurrence $PC(p_i, p_j)$ is the number of different sets in which predicate p_i, p_j that appear together. We design a cost model to measure the data table *T*. The column field of the table *T* is the set of predicates. And p_{max} is predicate with max *PF*, *NIL* and *RLE* are functions measuring the null values and correlation.

$$\text{cost}(T) = \frac{NIL(T)}{RLE(T)} = \frac{\sum_k^n PF(p_k) + PF(p_{\max}) - PC(p_k, p_{\max})}{\sum_i^n \sum_j^n \frac{PC(p_i,p_j)}{\min(PF(p_i),PF(p_j))}/n^2} \tag{1}$$

The main idea behind the heuristic merging in algorithm 1 is to iterate over the predicates co-occurring with the p_{max} in the lowest cost CS. If adding these predicates comes with a smaller cost, merge them, and update the cost and property set. We choose p_{max} instead of

enumerating all predicates because predicate combinations are too large.

Query Planner Query Planner generates an optimal query plan for a given SPARQL query to be further executed. Query Decompose module decomposes the query into star subqueries. Moreover, we can search candidate triples by indexing the constant predicate based on predicate partitioning. Finally, we join the subqueries to get the result of the query.

3. Evaluation

We implement the experiment on a cluster with five machines. Each machine is equipped with 24G RAM, 2TB disk, and a 6 core Intel Xeon E5-2420 processor. The cluster runs Cloudera 5.13.3 with Spark 2.4.4 on Ubuntu 16.04 LTS. We compare CoPMP with S2RDF [2] and Sempala [3]. S2RDF uses ExtVP, which is based on vertical partitioning, and Sempala is based on the property table. We choose S2RDF and Sempala as the comparison system to show that CoPMP has the advantages of property table and vertical partition at the same time. The tests were run using the synthetic dataset WatDiv [5] and real-world dataset DBpedia [6]. For the Watdiv benchmark, we generate datasets with Watdiv100M, Watdiv300M, and Watdiv500M to test the system's scalability and employ four types of queries snowflake(S), linear(L), star(S), and complex(C). Due to DBpedia's absence of query templates, we designed four queries that combine query types. Figure 2 ~ 4 show the average query time obtained on the Watdiv and Figure 1 shows the response time on the DBpedia. The results show that CoPMP proved more efficient than Sempala and S2RDF. Sempala needs to scan a big table each time since it stores triples with a unified property table, which results in a slow response. S2RDF precompute semi-join tables to reduce data shuffling. However, if the query does not appear in a semi-join table, the query can be expensive with VP tables. CoPMP merges CSs based on predicate correlation and predicate partitioning. Based on predicate partitioning, CoPMP can quickly find candidate triples since each predicate exists in only one table, similar to S2RDF. Also, CoPMP decomposes the query into star subqueries, thereby reducing joins with the benefits of property tables, but without scanning a single table containing all the data like Sempala. The results show that CoPMP benefits vertical partitioning and property tables.

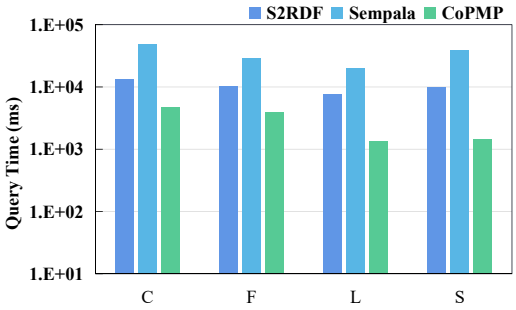

Figure 2: Watdiv100M

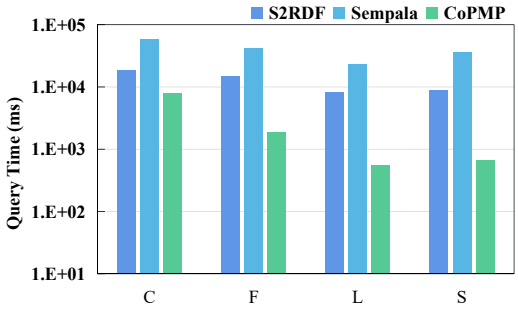

Figure 3: Watdiv300M

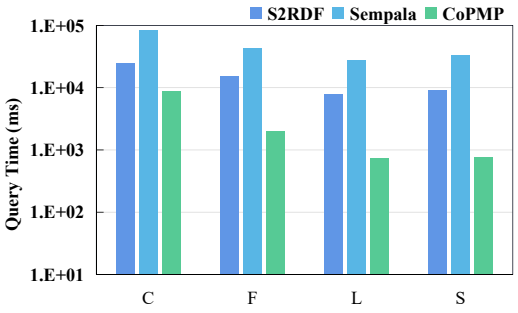

Figure 4: Watdiv500M

Table 1: Query time(ms) on different queries over DBpedia

	Q1	Q2	Q3	Q4
S2RDF	16,323	15,467	12,112	15,802
Sempala	62,843	95,854	65,618	77,429
CoPMP	8,704	8,716	7,378	4,579

4. Conclusion and Future Work

In this paper, we present a distributed query engine CoPMP that merges characteristic sets based on predicate correlation and predicate partitioning. Moreover, we allocate the merged table to the data partition based on the subject hash partition, which is the advantage of the CS with the same subject in distributed storage. Also, our extensive experimental results show the effectiveness of the CoPMP. In the future, we are planning to continue to extend the work to Well-designed SPARQL and conduct a comprehensive experiment to validate the efficiency of CoPMP.

References

[1] Neumann, T., Moerkotte, G.: Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In: ICDE. pp. 984–994 (2011)
[2] Schätzle, A., Przyjaciel-Zablocki, M., Skilevic, S., Lausen, G.: S2RDF: RDF querying with SPARQL on spark. In: VLDB. pp. 804–815 (2016)
[3] Schätzle, A., Przyjaciel-Zablocki, M., Neu, A., Lausen, G.: Sempala: Interactive sparql query processing on hadoop. In: ISWC. pp. 164–179 (2014)
[4] Papastefanatos, G., Meimaris, M., Vassiliadis, P.: Relational schema optimization for RDF-based knowledge graphs. In: Information Systems. pp. 735-754 (2022)
[5] Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: ISWC. pp. 197–212 (2014)
[6] Jens, L., Robert, I., Max, J., Anja, J., Dimitris, K., Pablo, M., Sebastian, H., Mohamed, M., Patrick, V., Sören, A., Christian, B.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. In: Semantic Web. pp. 167–195 (2015)
[7] Guangxi, J.: Optimizing Well-designed SPARQL Query Based on Constrained Pattern Tree[Master Thesis], Tianjin University, 2022.