

A Pruning-based Approach for Supporting Top-K Join Queries

Jie Liu, Liang Feng, and Yunpeng Xing

China Knowledge Grid Research Group, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, 100080, China

{lj, feng_liang, ypxing}@kg.ict.ac.cn

ABSTRACT

An important issue arising from large scale data integration is how to efficiently select the top- K ranking answers from multiple sources while minimizing the transmission cost. This paper resolves this issue by proposing an efficient pruning-based approach to answer top- K join queries. The total amount of transmitted data can be greatly reduced by pruning tuples that can not produce the desired join results with a rank value greater than or equal to the rank value generated so far.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems – query processing;

General Terms: Algorithms, Design.

Keywords: Join Query, Prune, Top- K .

1. INTRODUCTION

An important issue arising from large scale data integration is how to select the top- K ranking answers from multiple data sources, where K is relatively small compared to the total number of tuples.

To answer a top- K query, a straightforward way is to combine or join all the data from different sources, and then to select the top- K answers ordered by a user-defined rank function. However, this approach suffers from a great consumption of bandwidth when generating the whole results.

Fagin was the first to propose efficient algorithms to answer ranking queries in middleware environments [1, 2]. In [5], an efficient algorithm was introduced to process top- K queries over web-accessible databases by maximizing source access parallelism to minimize query response time. Ripple join is a new family of join algorithms designed for minimizing the time until an acceptably precise estimate of the query result is available [3]. Based on the basic idea of ripple join, a rank-join algorithm was proposed to support top- K join queries in relational databases [4].

This paper proposes an efficient pruning-based approach for answering top- K join queries in large scale data integration.

2. BASIC IDEA

The pruning-based top- K ranking approach takes the following parameters as input: relation R , relation S , the join condition $r(a)\theta s(b)$, the monotonic increasing rank function $f(r(p), s(q))$, and the number of desired join answers K , where $r(a)$ and $s(b)$ are the

join attributes, and $r(p)$ and $s(q)$ are the rank attributes of R and S .

Assume that relation R and relation S are sorted on their rank attributes in descending order, the basic idea of the proposed approach is to produce the top- K rank value of the rank function by iteratively pruning irrelevant tuples in R and S that can not produce any join results with a rank value greater than or equal to the rank value generated till now. Finally, tuples corresponding to the top- K rank value will be joined. To help illustrate the proposed approach, the following notions are depicted in Figure 1:

- (1) T — the lower bound of the rank function f ;
- (2) α — the row number of tuple in R ;
- (3) β — the row number of tuple in S ;
- (4) Top- K join tuples in R — the top- K tuples in R that can be joined with the first tuple in S . If the number of tuples in R that can be joined is less than K , then repeat the join process with the next tuple in S until the K join tuples are generated or the end of S is reached;
- (5) Top- K join tuples in S — the top- K tuples in S that can be joined with the first tuple in R . If the number of join tuples is less than K , then repeat the join process with the next tuple in R until the K join tuples are generated or the end of R is reached;
- (6) R -Rank-Queue (α, β, f) — the queue to keep the row number of top- K join tuples in R , the corresponding row number of joined tuples in S , and the rank value of the join results;
- (7) S -Rank-Queue (α, β, f) — the queue to keep the corresponding row number of joined tuples in R , the row number of top- K join tuples in S , and the rank value of the join results;
- (8) Priority-Rank-Queue (α, β, f) — the priority queue to keep the union of R -Rank-Queue and S -Rank-Queue.

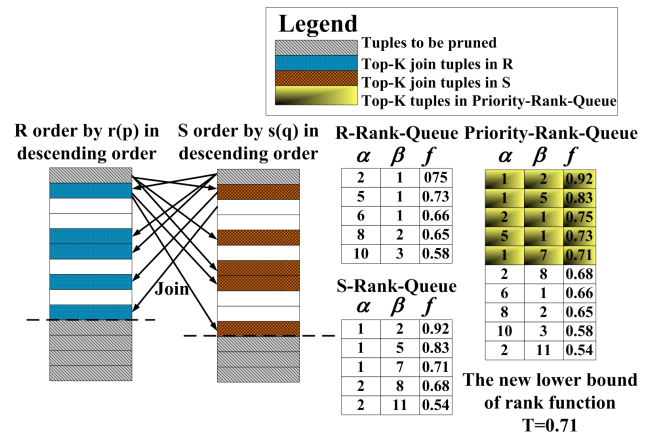


Figure 1. Illustration for pruning-based top- K ranking.

3. GENERAL ARCHITECTURE

The general architecture of the proposed approach is shown in Figure 2, which includes the following steps to answer a top- K join query:

- Step 1: (1) Initialize the lower bound of the rank function as $T=0$.
 (2) Empty R -Rank-Queue (α, β, f), S -Rank-Queue (α, β, f), and Priority-Rank-Queue (α, β, f).
- Step 2: Sort R and S on rank attributes in descending order.
- Step 3: Select top- K join tuples from R and S respectively by invoking algorithm $Top\text{-}K\text{-Join-Tuple}(R, S, f, K, T)$ and $Top\text{-}K\text{-Join-Tuple}(S, R, f, K, T)$ in Figure 3.
- Step 4: (1) Output the row number of the top- K join tuples from R , the row number of the joined tuples from S , and the rank value of the join results to R -Rank-Queue (α, β, f).
 (2) Generate S -Rank-Queue (α, β, f) in the same way.
- Step 5: Output R -Rank-Queue (α, β, f) and S -Rank-Queue (α, β, f) to Priority-Rank-Queue (α, β, f).
- Step 6: Sort Priority-Rank-Queue on the rank value in descending order.
- Step 7: If the length of R -Rank-Queue or the length of S -Rank-Queue is not equal to K , which implies no join results with greater rank value will be produced further, then the top- K ordered tuples in Priority-Rank-Queue is the top- K rank value for $R \bowtie S$, and the top- K ranking answers will be successfully returned.
- Step 8: Set the new lower bound T with the rank value of the K -th tuple in Priority-Rank-Queue.
- Step 9: (1) Prune the first tuple in R and S respectively.
 (2) Prune the tuples in R below the top- K join tuples because the rank value produced by these tuples joined with tuples in S can not be greater than or equal to the minimum rank value in R -Rank-Queue.
 (3) Prune the tuples in S below the top- K join tuples in the same way.
- Step 10: Goto Step 3.

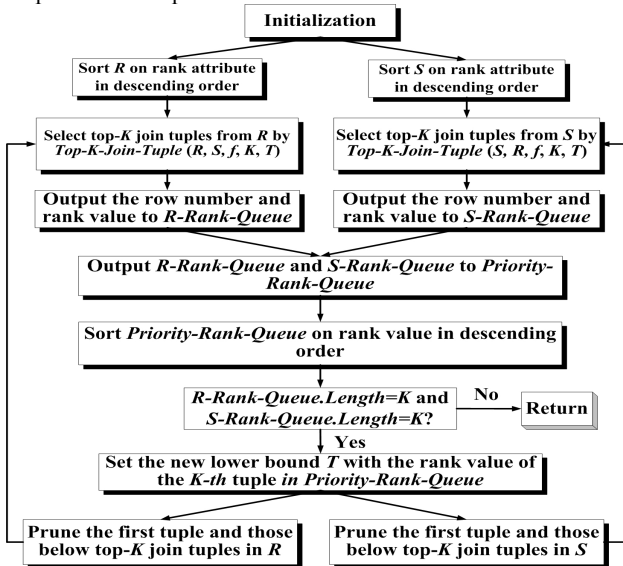


Figure 2. General architecture of the pruning-based top- K join query approach.

The algorithm for selecting top- K join tuples from relation R that can be joined with tuples from relation S is shown in Figure 3.

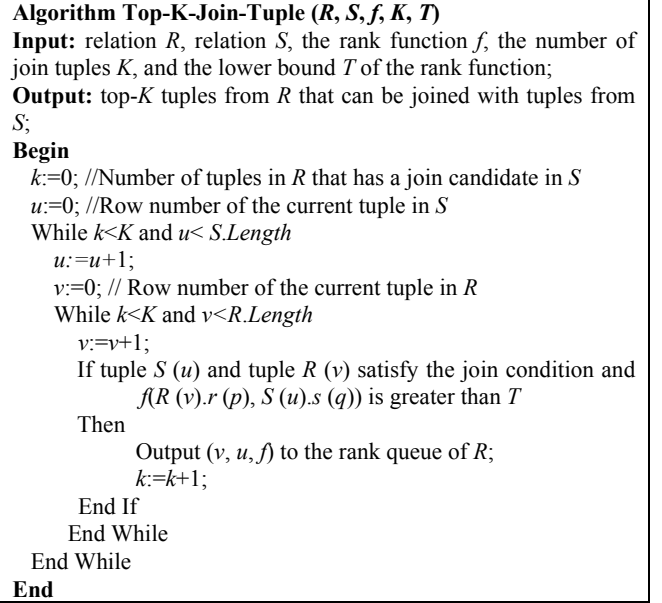


Figure 3. Algorithm for selecting top- K join tuples.

4. CONCLUSION

The contribution of this paper is to propose an efficient pruning-based approach for answering top- K join queries, which can be integrated in the semantic overlay of the Knowledge Grid and Peer-to-Peer networks to support advanced applications [6, 7].

5. ACKNOWLEDGEMENTS

The research work is supported by the National Science Foundation of China (No. 60503047) and the National Basic Research Program of China (No. 2003CB317000).

6. REFERENCES

- [1] Fagin, R. Combining Fuzzy Information from Multiple Systems. In *Proceedings of PODS 1996*, Montreal, Canada, June 1996.
- [2] Fagin, R., Lotem, A., and Naor, M. Optimal Aggregation Algorithms for Middleware. In *Proceedings of PODS 2001*, Santa Barbara, USA, May 2001.
- [3] Haas, P., and Hellerstein, J. Ripple Joins for Online Aggregation. In *Proceedings of SIGMOD 1999*, Pennsylvania, USA, June 1999.
- [4] Ilyas, I., Aref, W., and Elmagarmid, A. Supporting Top- K Join Queries in Relational Databases. In *Proceedings of VLDB 2003*, Berlin, Germany, September 2003.
- [5] Marian, A., Bruno, N., and Gravano, L. Evaluating Top- K Queries over Web-Accessible Databases. *ACM Transactions on Database Systems* 29 (2) (2004) 319-362.
- [6] Zhuge, H. The Knowledge Grid. *World Scientific Publishing Co.*, Singapore, 2004.
- [7] Zhuge, H., Liu, J., Feng, L., Sun, X., and He, C. Query Routing in a Peer-to-Peer Semantic Link Network. *Computational Intelligence* 21 (2) (2005) 197-216.