Vol.1 Issue.9, Pg: 163-169
December 2013

INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

SURVEY OF RANK JOIN ON DIFFERENT ENVIRONMENTS

Pearlin sheeja J S

PG Full Time Student, CSE Department, Karunya University, Coimbatore

pearlinsheejajs@karunya.edu.in

Abstract
Ranking is used to order the tuples based on their score. For joining the rank of different attributes, at first the attributes that are to be joined are identified and then the local ranking is made based on their individual scores. So that the ranked result of different attributes can be joined. While joining, it is not necessary to access all the tuples because the users will not expect the exact answers. Instead they will search for the approximate results that are similar to their query. So, for each query the joining will be made based on user preferences. At First, these rank joins were introduced in multimedia applications for searching the related images. Here, the techniques like incremental rank join and aggregated constraints are used and discussed. Now-a-days Rank join is used in different environments such as relational database, Heterogeneous and distributed environment where the two ranked results can also be joined. The algorithms that are used for these environments are also discussed here. The Cost is a main factor on these environments. Some of the techniques that are used for reducing the cost have also been discussed.

Keywords: Top – k, Aggregation, Heterogeneous, Distributed, Relational Database

1. Introduction

Rank join has become a vital need for many applications in multimedia including document retrieval, searching for similar data. The main benefits of rank joins are to help the users to make decisions effectively and also to make the work simpler for users. The rank join also helps in joining the ranked result from multiple atomic queries to the results of distributed environments. Rank join is used to join the ranked result of two or more reports based on their score.

1.1 Techniques

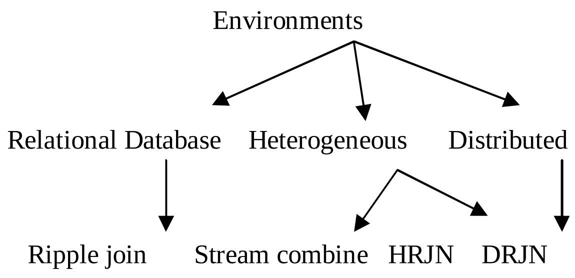
Here two different techniques are discussed one is incremental join and other one is rank join on aggregated constraints. Incremental rank join [4] is used to join the multiple atomic queries incrementally. These queries were ranked by the similarity score i.e. based on the similarity of the objects the results will be ranked. It is used in many multimedia applications and also for searching the related images. Here it joins incrementally until the top of queue is full. The Pull method is used to retrieve the next tuple if it is needed.

From the incremental rank join the aggregated rank join is evolved. Thus, the aggregated rank join also uses the concept of Incremental rank join. Here the results are aggregated at first based on the query and then by using incremental rank join the ranks are joined incrementally. Here the results are aggregated based on the similarity. For example, if the user is interested in searching the Mall and best hotels which are nearby and he imposes certain condition in which the overall cost he can spend is 1000 and the maximum cost of any purchase is less than 700. Then the query is aggregated based on the overall cost and the cost of purchase in mall and then the results are incrementally joined.

1.2 Environments

The concepts that are used above in incremental rank join and joins that are made on aggregated constraints are the base for the evolution of rank joins on different environments. These rank join can be classified into different environments as,

Fig 1 Classification on Different environments



In Relational database [1], there may be a set of relations say R_1 toR_{10}. Each tuple R_i is ranked locally. Based on the query from users, the relevant R_i are combined together to produce the total score. These total scores can be computed based on some function f that combines individual scores. I.F. Ilyas et.al. has proposed ripple join algorithm for relational database.

In Heterogeneous environments, the results are combined from different data sources. Here it does not rely on a result of single data source. Instead the results are approximated from a set of different data sources. The results are ranked by using Monotonic ranking function For example, if the user is interested in building a house in particular location where hospitals, schools, theatres are nearby with minimum cost. Then the results from multiple search engines are combined based on location and these are aggregated to find the top – k results. This ranked query model is also used in handling multimedia documents by using stream combine method which was proposed by Guntzer et.al.

In addition to heterogeneous environment, the rank join can also be made in distributed environments where the servers store fragments of data in autonomous manner [5]. Here the queries and relations may be fragmented into several parts and it can be stored in different servers. The resulting tuples can be ordered using scoring function.

1.3 Cost

While joining the rank on different environments the cost of the query may increase. This cost of the query is based on CPU performance and the time taken to complete the process. For reducing the cost instead of accessing all the tuples only the tuples which are needed are accessed. Thus it reduces the cost. For e.g. if the user is interested in retrieving schools and tuitions which are nearby and with minimum cost then it is not necessary to check all input tuples. Only the tuples, that are matching the given conditions are checked and retrieved so that the overall cost is reduced.

2. Taxonomy of processing Rank joins

2.1 Techniques

2.1.1 Incremental Rank Join

For Incremental join Query J^* [4] and J^{*PA} [4] algorithms are used. In J^* priority queue is maintained for both partial and complete join combinations ordered on upper bound [4]. At each step as the input arises an algorithm tries to complete the combination at the top of the queue which is partial. This terminates when join combination at the head of the queue is complete. Thus it joins incrementally.

J^{*PA} uses predicate access. It is better than that of J^* because it reduces the number of inputs that are to be accessed. This J^{*PA} is same as J^* with one variation. When processing an incomplete state from head of priority queue it first checks whether the state is sufficient to complete by using predicate access. In extreme case random access is also used to randomly access the tuples.

2.1.2 Rank Join on Aggregated Constraints

In naïve approach rank join is made first with all input tuples by using incremental rank join and then the filtering is used to remove the unwanted tuples which are not satisfying the given condition and at last the top – k results are determined. Thus, here post filtering method is used which increases the cost of the query.

To reduce the cost, Min xie et. al. has proposed deterministic and probabilistic algorithm. In deterministic algorithm corner bounding scheme [6] and Round robin strategy are used. Here at first the tuples which are satisfying the given conditions are aggregated based on the query and then join is made i.e. here pre filtering approach is used. Here many of the tuples were pruned off before joining. So, it reduces the cost considerably. The probabilistic algorithm will access fewer tuples when compared to deterministic algorithm.

2.2 Environments

2.2.1 Stream Combine

In Heterogeneous Environment for multimedia, stream combine method is used to join the ranked results. The streams can be combined by using an object id. For each object global and local object id are given. Guntzer et. al. says, if the query contains both keyword and image query then,

The first tuple in each attribute are accessed and has to check whether the object is present in both keyword and image. These keyword and image are sorted in ranking order locally.

For joining keyword and image the table should be formed by some variables. The attributes of the tables are A1 for keyword and A2 for image. s_1 denotes the score of keyword and s_2 denotes the score of image. Then there is a column called seen in A1 and seen in A2. If the object is seen in keyword or image then it should be updated in the column [8]. The upper bound is calculated by finding the average. In that table, incrementally check the values until any object is seen in both A1 and A2. If any object is seen in both keyword and image then it is one of the output of the given query. Do repeatedly for all seen tuples in keyword and image.

2.2.2 Relational Database

In Relational Database the new rank join algorithm was proposed by Ilyas et.al. It consist of two variants of ripple joins, hash ripple join and block ripple join. The Hash ripple join is initialized by specifying four parameters. They are two input condition, one join condition, and one combining function. Here the input conditions which are seen so far are stored in Hash table and the combining function is maintained in priority queue ordered on computed score.

Schnaitter et.al. says, this algorithm at first checks the priority queue. If any join result is found then the score is checked against threshold. If the join result has a value greater than or equal to threshold value then the method Get-Next answer is used. If the join score is below threshold then the algorithm it continues in reading tuples. For each result the combined score is generated and join is result is entered into the priority queue.

Ripple join helps to minimize the time until the query result is available. In two table ripple join unseen random tuples are retrieved from each table and these are joined with the previously seen tuples. Here they introduced two version of ripple join, square version and rectangular version.

In Square version the samples are drawn at the same rate. In Rectangular version more samples can be drawn from one relation than from the other.

2.2.3 Heterogeneous Environment

In Domain specific heterogeneous environment HRJN and HRJN* is used to join the ranked results. In relational databases HRJN is used only to join the result set of particular domain. But in Heterogeneous environment it can join the result from two or more domains. Here the inputs that are to be joined are ranked locally. Then the join condition should be identified. Join the results based on join condition and then rank them globally by using monotone ranking function.

For e.g. Consider a query, where the user needs the details of hotel and restaurants which resides in same street and with minimum cost. If the detail of hotel and restaurant resides in different domain then the domain should be ranked locally. Here in this example the hotel and restaurant are in different domains. So it must be ranked locally based on their rating.

Table 1 Hotel Domain

HID	Street	Rating
1	H	78
2	F	70
3	C	65
4	D	60

Table 2 Restaurant Domain

RID	Street	Rating
6	D	90
4	H	84
5	A	78
7	U	67

After local ranking is made, the join attribute of two domains should be identified. This join attribute can be identified using the keyword from query. In the above example, user have mentioned about hotel and restaurant which resides in same street. Thus, street is the keyword for the query. Thus, by using the street above two tables should be combined. The common street is identified from above table. Here D and H is the common street available. Based on the street, combination is made which is shown below.

Table 3 Rank join on Two Domains

HID	RID	Street	Rating
1	4	H	78
6	4	D	60

At last the global ranking is introduced to find the top results. This global ranking is made by finding the minimum number. Here for street D the rating of hotel is 60 and for restaurant the rating is 90. The minimum among them is 60. Thus, the global ranking for street D is 60.

2.2.4 Distributed Environment

HRJN* algorithm is better than that of DRJN [5] algorithm. But this DRJN algorithm is used in distributed environments where the fragments of data are stored in different servers. This DRJN framework is used for processing of highly fragmented data. In naïve approach, the fragments of different data are accessed centrally in one server. In this naïve approach communication cost and the round trip time will be high. This DRJN [5] uses statistics to determine the Score bounds and then based on this the Top – k result is found. To determine the statistics Histogram is used. In DRJN, at first the score bound is calculated for each relation. Then the corresponding score bound and the corresponding server which will store the corresponding bound is returned. Thus, this DRJN helps to minimize the above factors by utilizing histograms.

2.3 Cost

2.3.1 Sorted and Random access

Davide Martinenghi et al. say, by using both sorted and random access the cost can be minimized. The sorted access can be used to return the tuples in a sorted manner, whereas random access is used to retrieve the tuples that matching a given join attribute value. According to the authors, if the result is found in seen tuples then sorted access is used and if the result is found in unseen tuples then random access is the best choice. Here the pulling strategy is also introduced to access the next tuple.

3 Comparison of various Techniques

The merits and demerits of various techniques have been discussed below.

Table 4 Comparison of various Techniques

Techniques	Algorithm	Merits	Demerits
Incremental	J^*	It can support joins of ranked inputs based on user defined join predicates [4]. It Can handle nested join.	Only sorted access is available and the cost will be high.
	J^{*PA}	Same as J^* but also uses predicate access to reduce the cost.	It is only suitable for the multimedia applications which have similarity features. e.g. color, shape etc.
Aggregated	Deterministic	It reduces the number of tuples to be accessed by introducing a pre filtering approach.	It will return the exact answers. But not based on user predicates.
	Probabilistic	It guarantees high quality answers.	The cost of accessing next tuple may be high [6].
Relational Database	Ripple Join	It reduces the number of tuples to be accessed.	It decouples join from sort [1].
Distributed	DRJN	It can support Queries with multiple join attributes. Communication cost and latency are minimized.	Accessing the result from different servers and joining them makes it more complex.
Heterogeneous	Stream combine	It can join multiple atomic streams. Used for Multimedia applications.	It cannot join the result of two or more web searches.
	HRJN*	It can join the results of two or more search engines.	The cost for joining the result may be high.

4. Conclusion

A Survey of Rank join on different techniques and in different environments have done. For this purpose, a detailed analysis of different techniques included in three important environments like Relational Database, Heterogeneous and Distributed environments are explored. The merits and Demerits of different techniques used in these environments are also discussed.

REFERENCES

[1] I.F. Ilyas, W.G. Aref, and A.K. Elmagarmid, "Supporting Top-k Join Queries in Relational Databases," VLDB J., vol. 13, no. 3, pp. 207-221, 2004.

[2] K. Schnaitter and N. Polyzotis, "Evaluating Rank Joins with Optimal Cost," Proc. ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS), pp. 43-52, 2008.

[3] I.F. Ilyas, G. Beskales, and M.A. Soliman, "A Survey of Top-k Query Processing Techniques in Relational Database Systems," ACM Computational Survey, vol. 40, no. 4, article 11, 2008.

[4] A. Natsev, Y.-C. Chang, J.R. Smith, C.-S. Li, and J.S. Vitter, "Supporting Incremental Join Queries on Ranked Inputs," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 281-290, 2001.

[5] Christos Doulkeridis, Akrivi Vlachou1, Kjetil Nørvag1, Yannis Kotidis and Neoklis Polyzotis," Processing of Rank Joins in Highly Distributed Systems" Data Engineering (ICDE), 2012 IEEE 28th International Conference on 1-5 April 2012

[6] Min Xie, Laks V.S. Lakshmanan, and Peter T. Wood, "Efficient Rank Join with Aggregation Constraints", Proc. of the VLDB Endowment, Vol. 4, No. 11.

[7] Davide Martinenghi and Marco Tagliasacchi, "Cost-Aware Rank Join with Random and Sorted Access" IEEE transactions on knowledge and data engineering, vol. 24, no. 12, December 2012

[8] U. Gu¨ ntzer, W.-T. Balke, and W. Kießling, "Towards Efficient Multi-Feature Queries in Heterogeneous Environments," Proc. Int'l Conf. Information Technology: Coding and Computing (ITCC), pp. 622-628, 2001.