ISSN: 2277-9655
Impact Factor: 1.852

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A New Approach for Identifying Optimal Top - K Results Using Sorted and Random Access

Pearlin Sheeja J.S
PG Student, Department of Computer Science and Engineering, Karunya University, India
pearlsheej@gmail.com

Abstract

Ranking Queries will produce results based on some computed score. It is used to order the tuples based on their score. These queries involve joins where users are interested in top –k join results. For joining the rank of different attributes, the pipeline hash rank join algorithm is used to minimize the cost of accessing the tuples and to reduce the CPU cost. 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 users 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. So in this paper we have introduced a pipeline hash rank join algorithm to reduce the CPU time by accessing few numbers of tuples.

Keywords: Ranking Queries, Top – k, CPU cost, Pipeline hash rank join, Approximate Results

Introduction

Joining ranked inputs is an essential requirement for many database applications. Because if the users are interested in finding joined results of two or more domains then the user need not to search for different domains individually. The search engine itself will join the ranked inputs based on user predicates and the results are provided to the user. Thus search engine is user friendly for all the users. Consider Hotel, House and Schools are different domains. If the user queries about finding a location where the combined cost of hotel, housing rent and school are minimum, then by using join condition we can join the ranked results of different domains. Many algorithms and join strategies were introduced for joining those ranked inputs of different domains. First Fagin et.al. has introduced an algorithm for joining ranked inputs [5]. Based on that many algorithms has introduced namely threshold algorithm [2], incremental Rank join algorithm [4], hash rank join algorithm [4] and so on.

Now-a-days the users are mostly interested in finding Top – k results. Thus after joining those different domains by any join algorithm the top best results are retrieved among them by computing global rank for each score and then the results are provided to the users. These Top – k results will change dynamically based on the user predicates. For joining these domains pipeline hash rank join operator is introduced.

The pipelined hash rank join algorithm is introduced for joining the data from various search engines with minimum cost. This pipeline hash rank join helps us to minimize the number of rows that are to be accessed and so that the access time will also decrease. In huge amount of data, instead of accessing all the tuples only few amount of data is accessed. Thus the main aim of pipeline join is to access the search engine by using sorted access at

first and then the result is given as input to another search engine in which the hash join is performed.

Here while joining these queries only return the approximate results [2]. Since the users will show interest in approximate answers based on the type of their query. Here it will return the probabilistic data. Exact Results will always return the same answer every time. It is deterministic.

The cost is also a main factor to be considered. While accessing a large table the cost of the query should be minimized. Thus, Davide Marthiengi et.al. [1] has introduced a two type of access Sorted and random access. Thus by performing these two access the CPU cost can be minimized.

Related Works

Rank join has been extensively studied in recent years. There are many type of Ranking function. The most preferable is Monotone Ranking Function [2]. At First Fagin has introduced this rank joining on a single domain [5]. And then it is slowly moved on to incremental rank join [4].

Incremental Rank Join

Here the priority queue is maintained. 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. Here post filtering method is used. After accessing all tuples the unwanted tuples are removed.

Rank Join on Aggregated Constraints

Here the pre - filtering approach is used. [4] Thus here the tuples that are satisfying the given conditions are identified and then the aggregation is done. Thus here the tuples that are necessary are aggregated together. There are many function for aggregation. For e.g. SUM, COUNT, MIN, MAX

Hash Rank Join¬¬¬¬

The Hash rank 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.

Pipeline Hash Rank Join

Pipelining is a parallelism between two techniques [3] .Pipeline is a set of data processing elements where the output of one is given as input to another. Here while giving as input to another stream a joining is made by using Hash rank join. When a tuple arrives as input it is hashed and compared to the tuples in the corresponding bucket of other operand. If a match is found, then the joining is made. Thus pipeline Hash rank join algorithm is introduced. This algorithm has a high probability of finding result tuples in an early stage of the join classes because it does not need an entire operand before it can start producing the output tuples. Therefore the pipelining hash join is expected to produce more parallelism via pipelining.

Architecture

If the user queries about the hotel, housing rent and school which are located in the same location with minimum cost then the query will be given to the search engine. The search engine will pass this query to the query processor. The query processor will decide for which table to do sorted access based on the user query. Then the sorted access is performed and then it will be passed to the processor where the result of sorted access is stored in hash table. Thus the output of hash table is given as input to the another search engine in that random access is performed. These datas will be collected through internet resources and then the result is given as response to the user via search engine

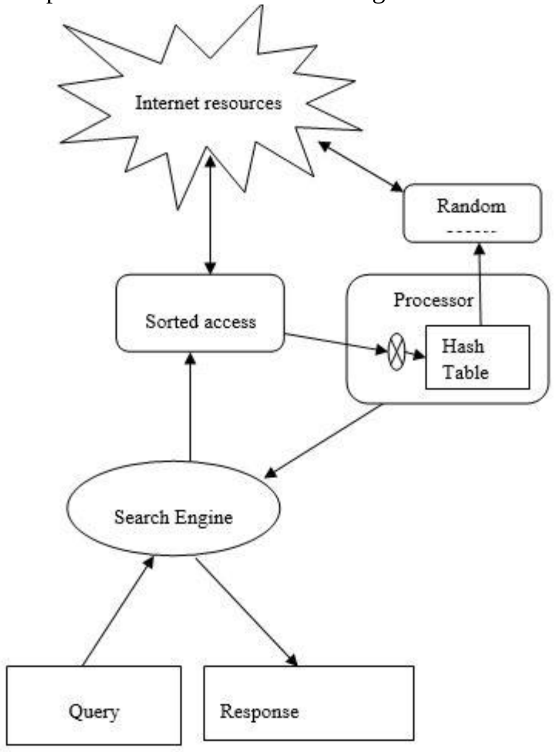

Fig 3.1 Architecture diagram for pipeline hash rank join.

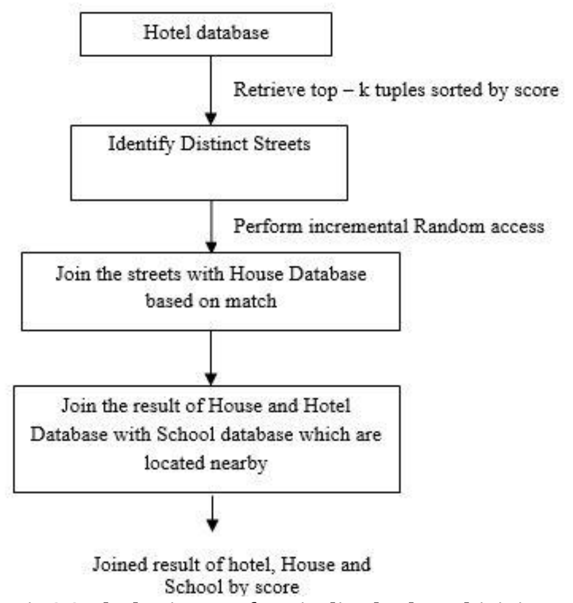
Hotel database
Retrieve top – k tuples sorted by score
Identify Distinct Streets
Perform incremental Random access
Join the streets with House Database based on match
Join the result of House and Hotel Database with School database which are located nearby
Joined result of hotel, House and School by score

Fig 3.2 Block Diagram for Pipeline hash rankjoining

If the pipe join is made between hotel, house and school database. At First, the top hotels are identified based on user query and the output is fed as input to the House database. Here this House database maintain a hash table based on location the input is hashed with House database and the matching location are identified and then the hash join is performed. This is considered as input and it is fed to another database called school. Here too hash table is maintained and the input is hashed with house and hotel database and then results are joined together. Thus, the joined results of house, hotel and school database are identified.

Methodology

Search computing

Search engine is an information retrieval system. It is designed to retrieve the information from a particular location where the results are stored. These information's are retrieved by using different kinds of access. Here the predicates that are necessary can be aggregated to form a signature [1].The Query processor will decide the kind of access that can be used to minimize the cost of the query. Here two kinds of access are used. Sorted access and Random access.

Sorted Access

Sorted access is used to return the tuples in sorted manner based on any attribute. This sorted access does not require any input. It will access all the tuples that are available and will return the sorted result.

Random Access

Random access simply means to reuse the same access pattern as sorted access but, instead of a query object, using an arbitrary object in another relation, in order to retrieve the similar objects. Random access might be provided by a different service than the one providing sorted access. Where a tuples of values for the join attributes is given as input, and tuples of values for the other attributes are returned.

Pipeline hash rank join

It is similar to Rank Join. Here the output of one search engine is given as input to the other search engine. The Hash Rank join is used while joining the data of two search engines. As a result, the combination of hotel and restaurant located in a particular street is identified.

Cost

Each request sent to a service with an expected cost of invoking it. Such cost may differ for different services, and also depends on the access pattern used for invoking the service. For each service, we shall in the following refer to its sorted access cost and its random access cost. These costs may correspond to the average service response time, which is a relevant cost indicator both in the case of data sources distributed over a network and in the case of services performing heavy computations.

Here the cost is also calculated for page size i.e, number of tuples retrieving in same page and the number of distinct join tuples that are introduced.

Result and Discussion

Relative Cost

In previous papers the cost will be high because of accessing all the tuples. By introducing pipeline rank join algorithm the number of tuples to be accessed can be minimized. In previous papers, all the tuples are accessed. Here only few numbers of tuples that are necessary are accessed. Thus CPU time will be reduced.

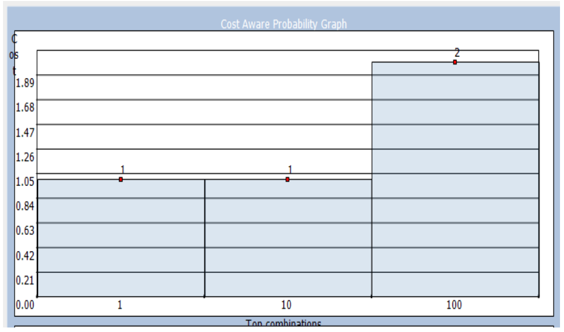

Fig 4.1. No.of tuples vs cost

Time

The cost of finding the solution is in the order of few milliseconds and thus totally negligible with respect to the typical access time required in practical cases involving remote services. This pipeline join algorithm will take few milliseconds.

Data Size

The cost of solving the problem does not depend in any way on the size of the data sets underlying the services involved in the join.

Conclusion

Pipeline Join can join the result of heterogeneous search engines. Here the domain specific search engines hotel and restaurant are used. Here the top results are retrieved for hotel database by using sorted access and the top results of hotel database is given as input to the restaurant database and then joined them to form combinations and at last the global rank is made to produce the overall Top – k result. The Monotone ranking function is used to rank the query answers. The two kinds of access are introduced i.e. sorted and random access. Here random access is used to retrieve the results randomly from unseen tuples. While accessing these pages the additive cost model is also considered.

Reference

[1] *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*

[2] *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.*

[3] *Annita N.Wilschut, Peter M. G. Apers "pipelining in query execution" PRISMA in an IEEE 1990*

[4] *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*

[5] *Wimmers, E.L. Haas, L.M. ; Roth, M.T. ; Braendli, C. "Using Fagin's algorithm for merging ranked results in multimedia middleware" IFCIS International Conference,pp. 267 – 278, 1999*

[6] *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.*

[7] *D. Braga, A. Campi, S. Ceri, and A. Rao. "Joining the results of heterogeneous search engines". Information Systems, 2008.*

[8] *R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," J. Computer and System Sciences, vol. 66, no. 4, pp. 614-656, 2003.*