Optimal Top-K Query Processing: Sampling and Dynamic Approach

Bhavani K., Rajeesh Kumar

M.E Computer Science and Engineering¹, Faculty of Computer Science and Engineering²,
Sathyabama University¹, Sathyabama University²
Chennai-600119, India

ABSTRACT

An effective query processing plays an important role in the uncertain data streams. Specifically, multiple top-k queries processing on uncertain data streams obtained from large applications of several fields such as sensor network monitoring and internet traffic control requires periodic execution of queries and sharing results among them. The system that monitors uncertain events in such data streams manipulates the top-k queries. Here the problem is that existing systems were not designed to allow query results sharing which in turn leads to high computation cost and inaccurate response from the system. To overcome these issues (Queries results sharing), the proposed system uses a sampling algorithm for sample the top possible worlds from well-known possible worlds based on their high probability. Consequently, proposed system uses an optimal dynamic programming approach that split the multiple queries into number of groups. Then the query groups are scheduled and planned for sharing results to yield minimum computation cost. Consequently, a faster greedy algorithm is used to reduce the time and storage space of the top-k queries based on the greedy rule. Thus the proposed approach allows sharing computation among multiple top-k queries and generates best plan of query execution.

KEYWORDS:

Query processing, results sharing, split the multiple queries.

1. INTRODUCTION

There are many applications in which data naturally occur in the form of a sequence of values, such as sensor data, financial tickers, online auctions, Internet traffic, web usage logs, and telephone call records. These data can be modeled as data streams, which are unbounded data sets produced incrementally over time. Often, due to possible errors caused by limitations of monitoring equipment, human operator mistakes, and interference in data transfer, these data may be incomplete, unreliable, or noisy, with the result that uncertainty is inherent in these data stream applications. Much work has focused on uncertain data streams and the semantics of possible worlds has been widely adopted in dealing with them. A possible world is a possible instance combination of tuples, usually within a sliding window. For a given time stamp, the tuples in the sliding window result in a number of possible worlds, whose number exponentially increases with the number of tuples in the sliding window.

The applications listed above typically issue a large number of monitoring queries. These are queries that are registered to the system, and they are executed periodically. An important subclass of these queries is top-k queries. For example, traffic monitoring applications typically wish to determine the top-k speeds of cars that pass through a control point, and volcano monitoring applications monitor top-k readings from sensors that produce uncertain data streams. We discuss traffic monitoring in more detail below. Previous work has considered executing these queries one-at-a-time, but there are considerable benefits to handling them collectively by exploiting similarities. This is the well-known multiquery optimization problem, which is known to
be very hard in relational DBMSs. This problem is particularly difficult over uncertain data streams. Also, the maximum number possible world combinations can increase the processing time if the probability is not suitable to group of top-k queries.

To overcome these issues, the proposed system using a sampling algorithm for sample the top possible worlds from well-known possible worlds based on their high probability. Consequently, proposed system uses an optimal dynamic programming approach that split the multiple queries into number of groups. Then the query groups are scheduled and planned for sharing results to yield minimum computation cost. Consequently, a faster greedy algorithm is used to reduce the time and storage space of the top-k queries based on the greedy rule. Thus the proposed approach allows sharing computation among multiple top-k queries and generates best plan of query execution.

II. RELATED ARTICLES

Tingjian Ge developed a paper for Uncertain data arises in a number of domains counting data integration and sensor networks. Top-k queries that rank results based on some user-defined score are an important tool for exploring large uncertain data sets. The semantics of top-k queries on uncertain data can be indistinct due to tradeoffs between reporting high-scoring tuples and tuples with a high probability of being in the resulting data set. Here, demonstrate the need to current the score.

To current the score distribution of top-k vectors to tolerate the user to desire between results along this score-probability dimensions. One option would be to present the complete distribution of all potential top-k tuple vectors, but this set is too large to work out. Instead, we intend to supply a number of typical vectors that effectively sample this distribution. We put forward efficient algorithms to total these vectors. Here also pull out the semantics and algorithms to the scenario of score ties, which is not dealt with in the previous work in the area. Due to the vector calculations, it has taken very less time.

Mohamed A. Soliman planned a paper for Top-k processing in uncertain databases is semantically and computationally different from traditional top-k processing. The interaction between score and uncertainty makes traditional techniques inapplicable. Here commence new probabilistic formulations for top-k queries. Our formulations are based on “marriage” of traditional top-k semantics and possible worlds semantics. In the light of these formulations, we build a framework that encapsulates a state space model and efficient query processing techniques to attempt the challenges of uncertain data settings. We show that our techniques are optimal in terms of the number of accessed tuples and materialized search states. Our experiments show the efficiency of our techniques under different data distributions with orders of magnitude progress over naive materialization of possible worlds.

Ming Hua · Jian Pei planned a paper for a novel and challenging problem of continuously monitoring top-k uncertain data streams, and offer a probabilistic threshold method. Here extend four algorithms systematically: a deterministic exact algorithm, a randomized method, and their space-efficient versions using quantile summaries. A wide experiential study using real data sets and synthetic data sets is reported to verify the effectiveness and the efficiency of our methods.

we focus on probabilistic threshold top-k selection queries, where the ranking function is applied on a single instance. Another category of top-k queries is top-k aggregate queries, where the ranking function is practical on a group of instances. The top-k groups with highest scores are returned as results. It is interesting to investigate how to extend the techniques discussed in this paper to switch top-k aggregate queries on uncertain data streams.

M. Tamer ozsu planned a paper for Data stream systems process constant queries, typically posed over sliding windows and re-evaluated periodically as the windows slide onward. Due to their long-running nature, a number of similar constant queries may run in parallel at any given time, therefore multiquery optimization is particularly important. In traditional multi-query optimization, one of the primary goals is to notice common parts across multiple queries issued at the same time and perform the common task only once. Another related goal is to ensure if a new query may be answered using the answer of a related query which has been computed previously and is stored as a materialized view. Here, we argue that in the count periodically re-evaluated queries, multi-query optimization want an additional step beyond common sub-expression matching. This is because queries that have been branded as similar may be re-evaluate with different frequencies and therefore may be planned at different times.

Thus, the additional step must attempt to match the re-execution times of similar queries in order to take advantage of computation sharing. The answer
obtainable in this paper focus on periodically-evaluated aggregates over sliding windows of various lengths, which are a common class of persistent queries used for monitoring purposes. The planned solution assumes that users specify an upper bound on the interval between re-evaluations of their queries and is based upon the following insight it may be cheaper to re-execute some queries more often if their re-execution schedule can be synchronized with those of similar queries, thereby amortizing the totalizing costs. We also explain that additional list group is possible when the system is forced to lengthen the desired re-execution intervals during periods of excess. Our solutions are base upon a variant of the earliest-deadline-rst algorithm that views persistent queries are periodic tasks. Experimental results show major improvements in system throughput due to increased resource sharing.

For all but the simplest queries, it may be infeasible for a DSMS to calculate up-to-date results whenever a new stream tuple arrive or an old tuple expire from its window. Additionally, users may it easier to deal with periodic output rather than a continuous output stream.

III. SUMMARY OF EXISTING SYSTEM

For example, traffic monitoring applications typically wish to determine the top-K speeds of cars that pass through a control point, and volcano monitoring applications monitor top-k readings from sensors that produce uncertain data streams. We discuss traffic monitoring in more detail below. Previous work has based on executing these queries one-at-a-time, then it has taken more time to execute queries and it has taken more space to storage. But there are considerable benefits to handling them collectively by exploiting similarities.

An Utopk query returns the vector of k tuples that has the maximum probability in all possible worlds. A U-kRanks query returns k tuples in which the i-th tuple ranks as the largest i in all possible worlds where 1 < i < k. In PT-k query, a probabilistic threshold top-k query, which returns all the tuples whose probability of being in the top-k is greater than a threshold p. A unified approach used to ranking in probabilistic databases, which defines a unified parameterized ranking function (PRF) to compute the total ordering of all the tuples and selects the k best tuples under this ordering.

Drawback of multi top-K queries is, it takes only individual top-K query processing over uncertain data. It is more complex for uncertain top-K query processing due to its sharing among the different top-K queries.

IV. PROPOSED SYSTEM

A proposed system found an efficient solution frame work for top-k query evaluation. The applications compute queries that involve joining and aggregating multiple inputs to provide users with the top-k results. In order to reduce the possible worlds a sampling algorithm is used. In order to make the system more optimal a dynamic programming solution is been used to reduce the time, space complexity and the cost. A Faster Greedy Algorithm is used to reduce the storage space. It executes the multiple top-k queries based on the greedy rule.

The main function of multiple top-K queries is users send the multiple top-K queries to system administration, then it prepared possible worlds and it sampling the possible worlds to reduce execution time after that it breaks the multiple queries into number of groups based on frequency upper bounds of queries with same frequency upper bounds are in same group. Then sort query groups in ascending order. After combining we have intragrouping sharing and inter group sharing. The first category i.e., queries with the same frequency upper bound but different k values represents intragroup sharing, while the second and the third categories represent intergroup sharing. Combination rule applied to group queries with different frequency upper bounds. Sharing among these queries is achieved by selectively executing groups of queries with higher frequency together with groups of queries with lower frequency. In this case, the cost of executing selected groups depends on the largest k of the groups.

Advantages of proposed system is Sharing among these queries is achieved by selectively executing groups of queries with higher frequency together with groups of queries with lower frequency. In this case, the cost of executing selected groups depends on the largest k of the groups. Possible Worlds are sampled that increased query processing speed. Query executing cost is reduced using optimal dynamic programming method. Execution time reduced using the Faster Greedy Algorithm. Also Query processing storage space reduced. Multi k-Queries are sampled efficiently for future work. Accuracy increased. Here combination rule is used to processing the query groups. Dynamic programming approach its shares intermediate query result sharing, due to this it takes very less time to execute. Faster Greedy algorithm it does not share the intermediate result, rather than this
it will the time to execute the query at what time it has to execute.

V. IMPLEMENTATION DETAILS

In the implementation part explaining about phases of multiple top-k queries execution such as top-k query design and sampling, intergroup sharing, intragroup sharing, dynamic programming scheduling and faster greedy algorithm.

A. top-k query design and sampling:

This module shows the admin submit their multiple top-k queries to the server that having uncertain data streams. Once the queries are submitted into server then it executes the top-k queries. Initially, the possible worlds are identified based on generation rule. The rule is defined as the records are combined with other records, if the record is uncertain tuple then check whether the summation probability is greater or less than the high probability. If greater means combine the uncertain record individually otherwise combine the records mutually. Then the possible worlds are sampled based on user desired selection criteria i.e. based upper bound probability and lower bound probability.

B. inter group sharing:

In this module, explained the same Upper frequency bound and different k-values query group sharing process i.e. the query groups are having the same frequency upper bound and top value is less than the second query group. Initially, assign a possible worlds probability with 1 Compute every possible world’s probability from previous computed possible world’s probability. . If possible world contain same time based probability values, sum those values then check if it less than probability one. If less than one subtract the value from probability one. Otherwise calculate possible world probability individually. If less than one subtract the value from probability one. Otherwise calculate possible world probability individual tuple. If less than one subtract the value from probability one. Otherwise calculate possible world probability individual tuple. Then, the top-k probability value of a tuple will be computed as summation of that tuple presented possible worlds probability. The possible world probability can be computed by combination the tuples in the possible worlds and their probability.

C. intra group sharing:

This module explained two types of query categories combination such as different upper frequency Bound with same k-values and different upper frequency with different k value. The combination rule utilized to mutual the query groups where the query groups are combined until top-k value of any query group is less than the other top-query group.

Query combination Rule:

Q is divided into r groups G =\{G1; ...;Gr\} where i < j implies fi < fj. If k^max_i < k^max_j, then set the frequency of Gj to fi, and thus the results of Gj are returned when Gi is executed. Gi and Gj are combined into the same group with the frequency bound fi and the largest k value k^max, as defined in the previous module. This combination step will continue until there are no pair of groups (Gi,Gj) such that k^max_i > k^max_j if i<j.

First query group UFB is less than Second Query group and First query group k value is greater than second query group k value. Then Share the intermediate results of First group query to second query group. If First query group UFB is less than Second Query group and First query group k value is less than second Query group k value. Divide the query group based on Upper Frequency Bound. Ordering the query groups into ascending order based UFB. Share the intermediate results of highest k value of query group to next lowest query group k-value. The above process continues until no combination occurred between the query groups.

D. dynamic programming scheduling:

This module explored the query execution scheduling process. In existing level, all possible query plans and their unit-per cost time is measured then which query plan is having minimum cost that query plan is executed. To avoid this problem, dynamic programming approach is used. The DP cost is computed concurrently for every query group. Then which query group is having minimum DP cost that group will be executed. Compute DP-cost to every query group recursively and store DP cost table. Select initial query group from all query groups and time unit is frequency upper bound then DP-Cost=time unit/frequency upper bound x k value of query. Compute DP-cost of reaming all query groups and that query group time unit is less than initial query frequency upper bound. Last execution of query group DP cost is equal to minimum DP-cost of
initial query group at the time unit interval + maximum value of query group k value + Next DP cost of initial query group at the time interval. Then query groups are executed based on minimum DP cost.

E. Faster Greedy Algorithm:

This module explained about the Faster Greedy Algorithm that handled the no intermediate results share top-k queries. The Greedy rule is used to execute the top-queries individually. The Greedy rule state that the first time to execute initial query group is the last time when next query group is executed in zero time and upper frequency bounds. Compute GA-cost at every time unit for every group recursively and stored in GA cost table. GA cost is zero when the query group is initial group and the time unit is less than initial time unit of that query group. If a query group is remaining query group and time unit is between initial query group time and next initial query group time and then GA cost is summation of next initial query group GA cost and previous query group GA cost. If query group is less than or equal to one and the time unit is equal to the query group time unit then GA-cost is summation previous query group GA cost and k value of query. Then execute query based on minimum GA cost. The above calculated single uncertain data stream procedure converted into multiple uncertain data streams.

VI. PERFORMANCE EVALUATION

The proposed system uses sampling algorithm for sampling the top possible worlds worlds from well-known possible worlds based on their high probability. Consequently, proposed system uses an optimal dynamic programming approach that split the multiple queries into number of groups. Then the query groups are scheduled and planned for sharing results to yield minimum computation cost. Consequently, a faster greedy algorithm is used to reduce the time and storage space of the top-k queries based on the greedy rule. Thus the proposed approach allows sharing computation among multiple top-k queries and generates best plan of query execution.

Fig 1.1 performance of existing and proposed system

CONCLUSION:

Querying over uncertain data streams, in particular top-k querying is important. There are many applications that require this functionality as discussed in the introduction. These applications always involve a large number of similar top-k queries. Although there have been some studies considering top-k querying over uncertain data streams, all of them consider individual queries and cannot be directly used for sharing computation among multiple top-k queries. This sharing problem is very challenging for the uncertain top-k queries with different frequency upper bounds and different k values. In this project, we formulate the problem, and present an optimal dynamic programming solution and a GA. We show that a naive method of enumerating all possible plans is not efficient, and the dynamic programming is optimal as it satisfies the optimal substructure and overlapping sub problems. The solution computed by the dynamic programming is the best plan for executing queries. Although the GA is not optimal, it is more efficient than the dynamic programming solution in terms of time and space. However, the unwanted possible worlds may be increased because of the query groups includes limited top level values. Our proposed sampling method is overcome this issue. Finally, we obtain effective results. In future work, we can improve our proposed system through applying more selection criteria on the possible worlds.
Acknowledgment

I would like to thank Dr. B. Bharathi, Head of the Department, Department of Computer Science and Engineering, Mr. Rajeev Kumar for her encouragement and support.

REFERENCES

1. Tingjian Ge, Stan Zdonik, Samuel Madden, “Top-k Queries on Uncertain Data: On Score Distribution and Typical Answers,” 2009
3. Cheqing Jin, Ke Yi, Lei Chen, Jeffrey Xu Yu & Xuemin Lin “SlidingWindow Topk Queries on Uncertain Streams”, 2009