ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Unified Framework for Handling Different Types of Top-K Queries

Saxin Merona V V[1], Soumya Mathew[2]

M.Tech Student, Dept. of CSE, VJCET, Mahatma Gandhi University, Vazhakulam, Kerala, India[1]

Assistant Professor, Dept. of CSE, VJCET, Mahatma Gandhi University, Vazhakulam, Kerala, India[2]

ABSTRACT: In this paper we present a unified framework which can handle different types of queries. The function used for handle queries is called scoring function. Scoring function used to compute the score of a pair of objects, such as score is the absolute difference between the pairs. In the case of top-k queries the scoring function returns top-k objects with the smallest scores. Here we are defining two types of queries that are top-k pairs queries and top-k groups queries. We present score based method, threshold algorithm and detailed theoretical analysis that demonstrates that the expected performance of our proposed algorithms. We also demonstrate that our framework can handle multivalued top-k pairs queries, multivalued top-k group queries and exclusive top-k pairs queries. We conduct extensive experiments to demonstrate the efficiency of our proposed approach.

KEYWORDS: Exclusive objects, multivalued objects, top-k group queries, top-k pairs queries

I. INTRODUCTION

Information systems of different types use various techniques to rank query answers. In many application domains, end-users are more interested in (top-k) query answers in the potentially huge answer space. Different emerging applications want efficient support for top-k queries. To identify the top-k objects is scoring all objects based on some scoring function. An object score acts as a valuation for that object according to its characteristics. Data objects are usually evaluated by multiple scoring predicates that contribute to the total object score. A scoring function is therefore usually defined as an aggregation over partial scores. Mainly the scoring function is taken as the difference between the objects.

Our framework can handle different types top-k queries such pairs queries and group queries. The pairs queries which retrieve the results as in the form of object pairs. But in the case of group queries it retrieves the values in the form of group of objects. The top-k objects which retrieved based on the distance between objects. In our framework pairs queries based on source based method [11]. Source based method is same as internal source algorithm [8]. Our framework supports chromatic and non-chromatic pairs queries. Chromatic queries depend on the color. For example, a state consist a lot of cities. Each city in the state has same color. Chromatic queries retrieve the results based on color. Chromatic queries are two types such as homochromatic and heterochromatic. Homochromatic pairs queries retrieve the pair of objects which having the same color. In heterochromatic pairs queries, retrieve the pair of objects having different colors. But in non-chromatic queries which does not depend on the color.

Pairs queries can handle multivalued and exclusive pairs. Multi-valued pairs mean the objects which has multiple instances [12]. For example, a state consists of many cities. Here we are considering population as an attribute. Because each city has different population. The different populations are the multiple instances. So we can say that state is a multivalued object. Exclusive pairs [13] that object pair which satisfies a condition the pair consider as exclusive. The pair objects will never see in other pairs.

The group queries [14] which is another extension of top-k queries. Group queries are contains group of objects. Group queries are little different than pairs queries. It retrieves the object groups based on group count. Group count is set default as three. For example, if we need to find top-k locations closest in distance to three other locations such that the total distances is minimum. The three locations changes according to query. Group queries are also chromatic and non-chromatic. Theses are same as of pairs queries. In the group queries object groups contains the same or different colors of objects. The group queries also can handle multivalued objects.

We summarize our contributions in this paper.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

- We first propose a unified framework for a broad class of top-k pairs queries. Here we define the top-k pairs queries on multi-valued objects and top-k pairs to answer exclusive pairs objects. Here we propose an algorithm for exclusive pairs.

  This paper is an extended version of [14] and we make the following additional contributions in this version.
- We extend the unified framework for a broad class of top-k group queries including k-closest group queries, k group queries and chromatic variants.
- We define the top-k group queries on multi-valued objects and present efficient techniques based on non-trivial lower bounds.
- The extensive experimental study demonstrates a significant improvement over the pairs queries. For generic top-k group queries, a comparison with the top-k pairs queries demonstrates the improvement.

II. RELATED WORK

To increase the efficiency of unified framework we need a good method. So here we referred a lot papers to find better method and another type of top-k query. Here we get different methods such as skyband based [3] [13], view based [10] [11], graph based [4] and source based [8] [14].

Top-k monitoring and skyband monitoring algorithm [2] introduced by Kyriakos Mouratidis, Spiridon Bakiras and Dimitris Papadias. Here it defines the method to handle top-k queries. But these algorithms are very complicated and expensive. So we refer dominant graph [4] introduced Lei Zou and Lei Chen, it shows another of graph view. It is very complicated to build. If we want to find the query results as by keyword search we refer the paper [7] introduced by Jianhua Feng, Guoliang Li, and Jianyong Wang. But this also cannot handle the unified framework. In the case of spatial queries defined in [5] introduced by Joao B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Norvag. Unified framework handle top k queries using internal and external memory algorithm [8] introduced by Muhammad Aamir Cheema, Xuemin Lin, Haixun Wang, Jianmin Wang, and Wenjie Zhang. This is good paper to handle the unified framework. Source based algorithm is easy to handle.AHBA and AMSA used to top-K dominating queries [9] introduced by G. Sandhya and S. Kousalya Devi. These methods are very difficult to handle. It cannot give the efficiency for a unified framework. In the case of view based method we studied LPTA and LPTA+ used to handle top-k query answering using Cached Views[10] introduced by Min Xie, Laks V.S. Lakshmanan and Peter T. Wood. Another method for view as view based approach used to handle all top-k [11]introduced by Shen Ge, Leong Hou U, Nikos Mamoulis, and David W. Cheung. If cannot get all the answers for a query we refer Handle why-not questions on top-K queries [12] introduced by Zhian He and Eric Lo. It helps to increase the speed of the framework. To study more about the source based method use paper [10] introduced by Zhitao Shen, Muhammad Aamir Cheema, Xuemin Lin, Wenjie Zhang, and Haixun Wang. Here it defined the top-k objects and top-k pairs queries. Another is unified framework for answering closest pairs and variants [14]. This paper provides the base of the paper and gets the method. Here we use threshold algorithm to get top-k objects. To study about the extensions such as multi-valued, exclusive and group queries we refer [6] [3] [1]. In the [6] which defines the features of multi-valued objects. Multi-valued objects have multiple instances. In [3] defined about exclusive and [1] defines about the group queries based on KNN. Top-k queries and their variants are mainly depend on the source based methods and threshold algorithm. It defined in section 3.

III. PROPOSED ALGORITHM

This section focuses on the Top-k Queries. Section A, B, C describes top-k pairs queries, Top-k pairs queries on multivalued objects and Top-k pairs on exclusive objects. Section D describes top-k group queries and section E presents top-k group queries on multivalued objects.

A. *Top-K Pairs Queries:*

Top-k pairs query problems solved by applying the existing work on the top-k queries. The algorithms assume that the sources can report the elements in a sorted order. So if we need to create and maintain the sources then need to develop efficient techniques.

In top-k pairs queries which retrieve the objects in the form of pairs. Each pair is retrieved based on the source based algorithm. Here we are looking in to the scoring function such as distance between the pairs.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

The framework which shown in the Fig. 1 is the main work of the Top-k pairs queries. Here we added a technique called skyband table. Its usage is very important in the case of top-k queries. Its working is described as: Initially create skyband table in database with query and result fields. If a user wants to give a query, then first he should check whether the query is already in skyband table or not. If it is available then select the query from skyband table. The data will be retrieved very fast. Otherwise whole process will be take place and then saved into the database.

Mainly top-k pairs queries consist of three steps. Initially we need to create the skyband table then create and maintain the source and at last find the top-k pairs using threshold algorithm. These are defined in [14]. Fig. 2 defines the threshold algorithm. It is the main technique to get top-k pairs.

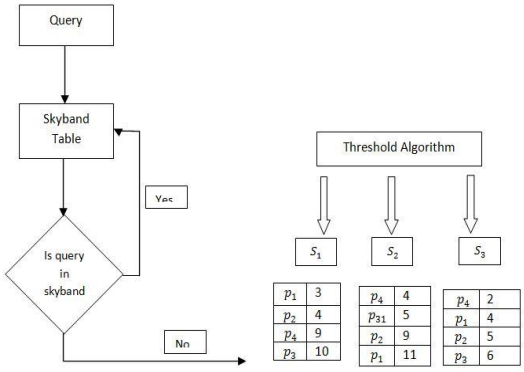

Fig.1. Framework

Algorithm 1 Top-k Algorithm

FindTop-kPair ()

1: get the top pair (o_v, o_w) from the heap
2: **if** the pair does not exists in source **then**
3: find the score of (o_v, o_w) from other sources
4: **if** K not equal to number of top-k pairs **then**
5: insert the pair (o_v, o_w) in heap with score $s(o_v, o_w)$
6: update top-k pairs and k-th best pair if required
7: t = $f(L[1].....L[d])$
8: **if** t ≥ score of k-th best pair **then**
9: **return** top-k pairs

Fig. 2. Algorithm 1

B. *Top-k Pairs on Multivalued Objects*

In this section, we formally define top-k pairs queries on multi-valued objects and propose efficient query processing techniques. Mainly top-k pairs queries on multivalued objects consist of three steps. Initially we need to create the skyband table then create and maintain the source and at last find the top-k pairs using threshold algorithm. In the case of computation of top-k pairs we need to find $SCORE_{\phi}[i](U,V)$. Creation of skyband table is already defined above. The whole technique and algorithm referred in [14].

C. *Top-k Pairs on Exclusive Objects*

Here we define exclusive pairs (ECP). A real-life application of the ECP query is the car parking assignment problem, where each car driver requests for a parking slot from the set B of available slots. The ECP searches for the one-to-one assignment of cars to parking spaces, such that the sum/average of travel distances is minimized. It is more reasonable to assign each car c to A to the parking space p to B that may not be taken by another driver c0. So the car assignment problem handled by ETP.

Mainly top-k pairs queries consist of three steps. Initially we need to create the skyband table then create and maintain the source and at last find the top-k pairs using threshold algorithm. These are already defined in [14]. Here we add algorithm 2 (Fig. 3) and 3 (Fig. 4) to know how the exclusive top-k pairs will work. The threshold algorithm is same as algorithm 1 (Fig. 2).

D. *Top-k Pairs on Exclusive Objects*

In top-k group query that contains d number of local scoring functions. If we get a query first of all we need to find out the scores of the group elements from different sources. Sources are always sorted. The maintenance of d sources is shown in Fig. 3.2 such that each source S_i incrementally returns the group with the best score according to the ith local scoring function. To retrieve the top-k groups of combining ranked or sorted inputs from sources use the top-k algorithms mainly threshold algorithm.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Algorithm 3 Exclusiveness Check

containPairObjects ()

1: get the pair (o_r, o_x) from the heap
2: **if** $o_v = o_r$ and $o_v = o_x$ and $o_w = o_r$ and $o_w = o_x$ **then**
3: **do** not add the pair (o_r, o_x) to heap
4: **else** add the pair (o_r, o_y) to heap

Fig. 3. Algorithm 3

Algorithm 2 Exclusive Pairs

initializeSource ()

1: Sort the objects in ascending order of their values
2: **for** each object o_v **do**
3: $o_w \leftarrow$ the best guest of o_v
4: **if** (o_v, o_w) satisfies the exclusiveness constraint **then**
5: insert the pair (o_v, o_w) into heap with score $s(o_v, o_w)$
6: **if** any pair with o_w and o_v **then**
7: **check** containPairObjects ()

getNextBestPair ()

1: get the top pair (o_v, o_w) from the heap
2: **if** next best guest of o_v exists **then**
3: $o_{w'} \leftarrow$ the best guest of o_v
4: insert the pair in heap with score $s(o_v, o_{w'})$
5: **return** $(o_v, o_{w'})$

Fig. 4. Algorithm 2

Top-k group query problems solved by applying the existing work on the top-k pairs queries. The algorithms assume that the sources can report the elements in a sorted order. So if we need to create and maintain the sources then need to develop efficient techniques. The technique which used here will give the source with group of objects in a sorted order. A straightforward solution to create a source S_i is to sort all possible group objects according to their local scores on the ith attribute.

In the top-k group queries they have an opportunity to select queries directly from skyband table which shown in Fig. 5. The skyband table which stores the all the top-k groups queries as well as the results. So it reduces the time of execution. We can directly retrieve the values from table. Mainly top-k group queries consist of three steps. Initially we need to create the skyband table then create and maintain the source and at last find the top-k group using threshold algorithm. The creation of skyband table is defined above.

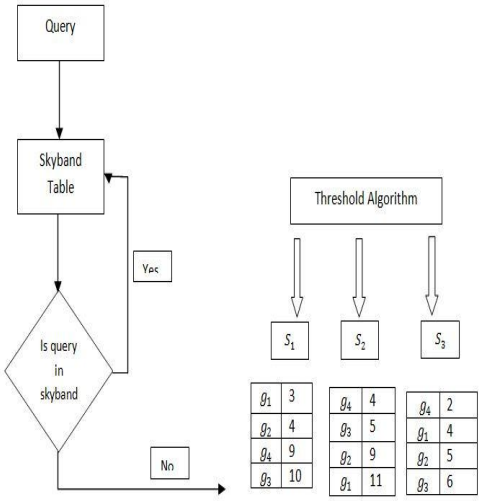

Fig. 5. Framework

Algorithm 4 Creating and Maintaining Source

InitializeSource()

1: Sort the objects in ascending order of their values
2: **for** each object o_v **do**
3: $o_w \leftarrow$ the best guest of o_v
4: $o_z \leftarrow$ the best guest of o_v
5: Check the group count as given in query
6: **if** object number in group equals to group count **then** stop
7: insert the pair (o_v, o_w, o_z) into heap with score $s(o_v, o_w, o_z)$
8: **else** continue
9: **return** (o_v, o_w, o_z)

getNextBestPair ()

1: get the top pair (o_v, o_w, o_z) from the heap
2: **if** next best guest of o_v exists **then**
3: $o_{w'} \leftarrow$ the best guest of o_v
4: $o_{z'} \leftarrow$ the best guest of o_v
5: Check the group count as given in query
6: **if** object number in group equals to group count **then** stop
7: insert the pair $(o_v, o_{w'}, o_{z'})$ in heap with score $s(o_v, o_{w'}, o_{z'})$
8: **else** continue
9: **return** $(o_v, o_{w'}, o_{z'})$

scoreCompute ()

1: Set totalscore =0
2: **for** each object group (o_v, o_w, o_z) **then**
3: find the score between o_v and o_w
4: find the score between o_v and o_z
5: totalscore = totalscore + (score of (o_v, o_w) + score of (o_v, o_z))
6: insert (object group, totalscore) into heap

Fig. 6. Algorithm 4

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Step 1: Maintaining the Sources:

Initially all the objects are sorted of their attribute values in ascending order such that $o_1 \le o_2 ... \le o_N$. For any group (o_v, o_w, o_x), the host is o_v which the first object in the group and the guests of the group (o_w, o_x) are the second and third object. A group (o_v, o_w, o_x) means the objects (o_w, o_x) are the guests to host o_v. Every object o_v can host only the objects that are on the right side of o_v in the sorted list $o_1 \le o_2 ... \le o_N$. For chromatic queries, the guest objects in the group that are on the right side of o_v that meet the color requirement. In the case of o_v which has two guests such as o_w and o_x. Here we can say (o_w, o_x) are the best guests of o_v than $(o_{w'}, o_{x'})$ if $s(o_v, o_w, o_x) < s(o_v, o_{w'}, o_{x'})$. The objects (o_w, o_x) are called the best guests of a host o_v if for every other guest (o_w, o_x) of the host o_v, $s(o_v, o_w, o_x) < s(o_v, o_{w'}, o_{x'})$. We can say that the group (o_v, o_w, o_x) has been reported as best group, if object o_v has hosted the object o_w and o_x.

Here we use the Algorithm 4 (Fig. 6) which presents the details of creating and maintaining a source. Initially, read and then sort all the objects in ascending order of their attribute values. For each object o_v, if (o_w, o_x) are the best guests of o_v then created a group (o_v, o_w, o_x) if the group number is limited to 3. At last all these groups are inserted to the heap. Whenever a request for the next best group arrives, the source retrieves the top group (o_v, o_w, o_x) from the heap and reports it to the main algorithm. The next best group is (o_v, o_x, o_y) is inserted to the heap where (o_x, o_y) are the next best guests of o_v. At any stage during the execution, the next best guest of o_v is the best guest among the guests of o_v which has not been hosted by o_v.

Example: Consider the example of Fig.7 which shows six objects o_1 to o_6 which are sorted on their attribute values. Attribute values are contains inside the circles. Here we assume the scoring function as absolute difference. A group $s(o_v, o_w, o_x)$ is shown by a directed edge from the host o_v to the guests (o_w, o_x). Initially, for each object, this inserted in the heap as a group with its best guests. If the scoring function is absolute difference, then for each object the best guests are its right adjacent objects. Fig. 3.3 shows the groups that are inserted in the heap. Here we consider the best group as (o_3, o_4, o_5) because it has smallest score. Here the score of the group is 7. After retrieving that group, the algorithm determines that the next best guest of o_3 and inserts (o_3, o_5, o_6) in the heap with score 14 (see Fig.7).

Adjacent Objects*:* The first and second objects (o_u, o_t) are on the left side of o_v in the sorted list is the left adjacent object of o_v such that the group (o_v, o_u, o_t) satisfies the color requirement. Next object of the group will be left adjacent of o_t that is o_s satisfies the color requirement. So the group is (o_v, o_t, o_s). If the group contains the homochromatic group which contains the objects with same color. If it is heterochromatic the group contains different color objects, where the host and second guest have same color.

The first and second objects (o_w, o_x) are on the right side of o_v in the sorted list $o_1 \le o_2 ... \le o_N$ is the right adjacent object of o_v such that the group (o_v, o_w, o_x) satisfies the color requirement. Next object of the group will be right adjacent of o_x that is o_y satisfies the color requirement. So the group is (o_v, o_x, o_y). In the Fig. 8 which shows some objects are (o_2, o_4, o_5) and some are white (o_1, o_3, o_6). Fig. 7(a), 7(b) and 7(c) show the adjacent objects for non-chromatic queries, heterochromatic queries and homochromatic queries, respectively. The broken lines have shown the adjacent objects. An arrow from an object o_u to o_v indicate that o_u is the adjacent object of o_v in that direction.

Step 2: Finding the Best Guest for Each Object o_v:

In the case of right increasing functions, the score is $s(o_v, o_w, o_x) < s(o_v, o_{w'}, o_{x'})$. Hence, for any object o_v their right adjacent objects are its best guest. For example, in Fig. 3.3(c), o_3 and o_6 are the best guests of o_1 if the scoring function is right increasing function.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

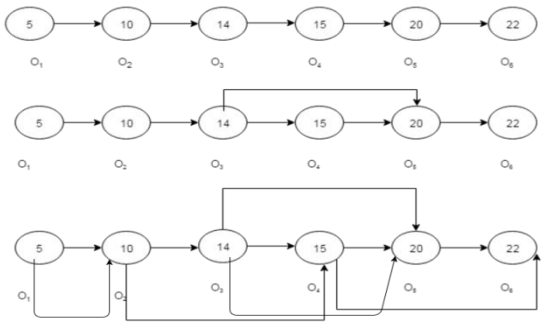

Fig. 7 Illustration of Algorithm

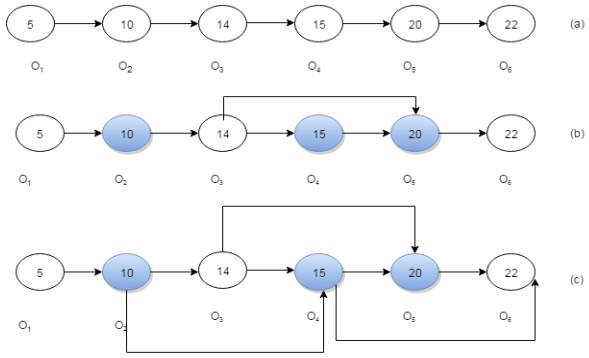

Fig. 8. (a) non-chromatic (b) heterochromatic (c) homochromatic

In the case of right decreasing functions, for any object o_v, (o_w, o_x) are the best guests because these are the right most objects such that the group (o_v, o_w, o_x) meets the color requirement. In the case of non-chromatic queries, the best guest of any object o_v is o_N. For example, in Fig. 8 (a) the best guest of every object o_6 is if the scoring function is a right decreasing function. For the heterochromatic queries, in each group contains different objects with different colors. Such as, if o_N has a color different than o_v then o_N is the best guest of o_v. In the case of left adjacent objects, o_N is the best left guest of o_v because it has the color different than o_v. In the Fig. 8(b), o_2 and o_3 are the best guests of o_1 whereas o_4 and o_6 are the best guests of o_2.

For the homochromatic queries, initially we scan the sorted list $o_1 \le o_2 ... \le o_N$ once and maintain the right most object of each color. For each object o_v, its best guest is the right most object of the same color. In the example of Fig. 8(c), o_3 and o_6 are the best guests of o_1 whereas o_4 and o_5 are the best guests of o_2.

Step 3: Finding next best guest of any object o_v:

Let $(o_w, o_{x)}$ are the current best guests of the object o_v. We need to find the next best guest of o_v. Here describe how to find the next best guests for the right increasing functions and the right decreasing functions. For non-chromatic and homochromatic queries, the next best guest of o_v are $o_{w'}$ and $o_{x'}$ which are the right adjacent object of o_w and o_x. In the example of Fig. 8(c), let $(o_3, o_{6)}$ be the current guests of o_1. The next best guests of o_1 are o_6 and o_8 which are the right adjacent objects of o_3.

For the heterochromatic queries, the next best guests are $o_{w'}$ and $o_{x'}$ of o_v if $o_{w'}$ has a color different than $o_{x'}$. Consider the example of Fig. 8(b) and assume that the current best guests of o_1 are o_2 and o_3. When (o_1, o_2, o_3) is reported, the algorithm checks o_3 to see if it is the next best guest of o_1. Since o_1 and o_3 have the same color, the next best guest of o_1 is o_4 which is the right adjacent object of o_3. The next best group is (o_1, o_4, o_6).

Step 4: Finding the adjacent objects:

The procedure of finding adjacent objects is not needed for non-chromatic queries. So, it used in the case of heterochromatic queries. So initially we need to set the right adjacent object of o_N to NULL. Then we scan the sorted list of the objects from right to left. When $o_{w'}$ and $o_{x'}$ is set as the right adjacent object of o_v if $o_{w'}$ has a different color than o_v and $o_{x'}$ has a different color than $o_{w'}$. Consider the example of Fig. 8(b), the right adjacent object of o_6 is set to NULL. The right adjacent objects of o_3 is o_4 and o_6 because they have different colors. The right adjacent object of o_3 is not o_5 because they have same color. So, the right adjacent object of o_5 is set as the right adjacent object of o_3. The left adjacent objects can be set similarly by scanning the list from left to right.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

For the homochromatic queries, here scan the list from right to left, maintain the last seen object of each color. The right adjacent object for any object o_v, is the last seen object of the same color. The left adjacent objects are set similarly by scanning the list from left to right.

Step 5: Query processing algorithm

Here we use threshold algorithm to combine the scores of a group from different sources and return the top-k groups. The threshold algorithm supports the random accesses. That is if return a group from a source S_i threshold algorithm needs to find out its score on every other attribute. All the objects are stored in memory. Initially, we return a pair (o_v, o_w, o_x) from a source, then find the objects attribute values o_v, o_w and o_x from the object table and then compute the score of the pair (o_v, o_w, o_x) on every other attribute. Algorithm 5 (Fig. 9) shows the computation of top-k groups.

Algorithm 5 Top-k Algorithm

FindTop-kGroup ()

1: get the top group (o_v, o_w, o_x) from the heap
2: **if** the group does not exists in source **then**
3: find the score of (o_v, o_w, o_x) from other sources
4: **if** K not equal to number of top-k groups **then**
5: insert the group (o_v, o_w, o_x) in heap with score $s(o_v, o_w, o_x)$
6: update top-k groups and k-th best group if required
7: t = $f(L[1].....L[d])$
8: **if** t ≥ score of k-th best group **then**
9: **return** top-k groups

Fig. 9 Algorithm 5

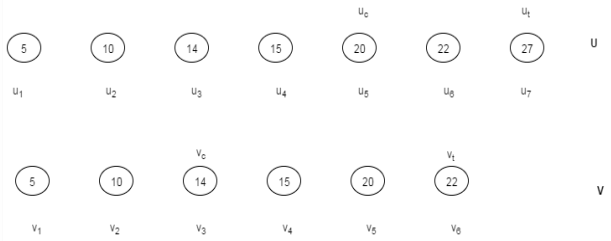

Fig. 10 Critical and terminal instances

E. *Top-k Pairs on Multivalued Objects*

In the previous section and [14] we already studied about multivalued objects. Multivalued objects have many instances. For example a state has a lot cities and each city has different population. So states multiple instances are cities. Each city is evaluated with some attributes such as coordinates, population etc. Here discussing about top-k groups on multivalued objects. The top-k groups retrieve the top k multivalued object groups. It is different than top-k pairs. Here we can get the object groups based on the group number.

In this section, we formally define top-k group queries on multi-valued objects and propose efficient query processing techniques. Mainly top-k group queries on multivalued objects consist of three steps. Initially we need to create the skyband table then create and maintain the source and at last find the top-k pairs using threshold algorithm. In the case of computation of top-k group we need to find $SCORE_{\phi}[i](U,V,W)$. Creation of skyband table is already defined above.

The local score of a group of instances is g = (u, v, w) in ith dimension which is denoted as g.score[i] that is g.score[i] = |u[i]-v[i]+ u[i]-w[i] |. Here we using g.score[i] to compute the ith local ϕ-quantile score of a group of instances of multivalued object (U,V,W) and is denoted as $S_{\phi}[i](U,V,W)$. To compute top-k groups of multivalued objects, first of all we need to divide the problem into d one-dimensional problems and then apply the algorithm. To achieve this, ensures that the global ϕ-quantile score $SCORE_{\phi}(U,V,W)$ of a pair (U, V,W) can be lower bounded by using its ϕ/d -quantile local scores $S_{\phi/d}[i](U,V,W)$ in each of d dimensions. We use $LB\_SCORE_{\phi}[i](U,V,W)$ which denote a lower bound on $SCORE_{\phi}[i](U,V,W)$ and $LB\_SCORE_{\phi}[i](U,V,W)$ to denote a lower bound on $S_{\phi}[i](U,V,W)$.

Framework*:* The basis of creation and maintenance d sources as 1) each source S_i incrementally returns the group (U,V,W) with the smallest and $LB\_SCORE_{\phi}[i](U,V,W)$ 2) $LB\_SCORE_{\phi}[i](U,V,W)$ can be computed for each given group of multivalued objects (U,V,W).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Step 1: Creating and maintaining sources

In this section, we describe how to create and maintain the sources. Initially we need to sort the instances of all multivalued objects to create the sources.

Let L denote the list of every group (u_i, v_j, w_k) where u_i in U, v_j in V and w_k in W. Let L' € L be the set of groups such that the total weight of the group of instances in L' is at least 1-ϕ/d . Let p € L' be a group with the smallest ith local score in L'.

Here we define the construction of L' and $LB\_SCORE_{\phi}[i](U,V,W)$, using these instances.

(1)Terminal instance: For a multivalued object U, its terminal instance in ith dimension is the instance with the largest value in ith dimension and is denoted as u_t .

(2)Critical instance: Let $(u_1 \ldots u_m)$ denote the instances of a multivalued object U sorted in ascending order of their values in ith dimension. The critical instance of U is an instance u_c such that $\sum_{j=c}^{m} w(u_j) \geq 1-\phi/d$ and $\sum_{j=c+1}^{m} w(u_j) <$ 1-ϕ/d .

Fig. 10 shows two multivalued objects U and V with seven and six instances, respectively. The weight of each instance u_i € U is 0.14 and the weight of each v_i € V is 0.16. u_7 and v_6 are the terminal instances of U and V which are shown as circles having thick boundaries. Assuming 1- ϕ/d =0.6, the critical instances of U and V are u_2 and v_3 respectively. Algorithm 6 presents the details of creation, maintenance and computation of group queries on multivalued objects.

Fig.7 in [14] shows the example, where five multi-valued objects U, V, W, X and Y are shown. The terminal instances of the objects are shown in that Fig.

In the sorting of multivalued objects, initially we need to sort the instances of each object in ascending order of their values in ith dimension. The terminal and critical instances of each object are identified by shown above. Here we get two sorted lists such as

1) Terminal list contains the terminal instances of the objects sorted according to their values

2) Critical list contains the critical instances sorted on their values.

Here we define how to find the best guest of U. Initially an object U serves as a host to an objects V and W only if $u_t \leq v_t$ and $u_t \leq w_t$. In Fig. 7 in [14], U will be the host of the objects V, X and Y. We consider V as best guest of U if it has minimum $LB\_SCORE_{\phi}[i](U,V)$= max $(0, v_c - u_t)$. The objects has minimum v_c that are called the eligible guests of U. Such that if consider V and W as the best guests of U which has minimum v_c and w_c. Here we consider the group (U, V, W) because it has minimum score. The score get by $LB\_SCORE_{\phi}[i](U,V,W)$=max $(0, v_c - u_t)$ + max $(0, w_c - u_t)$. Fig. 7 in [14] shows the best guest object for each object by drawing an arrow from the host object to the guest object. The source is initialized then inserts these four pairs to heap. Here V and X be the current best guests of U in a group. The next best guests of U can be finding by binary search. In the case when V, X was obtained using a binary search, then an object (V', X') are the next guests where (X', Y') is the objects that has critical instance x'_c, y'_c adjacent to v_c in the sorted list.

The algorithm 6 (Fig. 11) accesses each source in a round-robin fashion. A group (U, V, W) returned by a source S_i is possibly among the top-k groups. So we need to compute its ϕ-quantile global score and update the list of top-k groups accordingly (lines 7 and 8). $SCORE_{\phi}(U,V,W)$ is computed using Algorithm 6. Here we do not need to compute $SCORE_{\phi}(U,V,W)$ if (U, V, W) has already seen in any other source (line 4). Otherwise need to compute the lower bound $LB\_SCORE_{\phi}(U,V,W)$ by doing random access on each other source. If $LB\_SCORE_{\phi}(U,V,W)$ is larger than the score of k-th best group seen then (U, V, W) cannot be among the top-k groups. Hence, the computation $SCORE_{\phi}(U,V,W)$ is not required (line 6). The algorithm stops when the best possible score of any unseen group (i.e., threshold) cannot be smaller than the score of k-th best group. Let L_i denote the lower bound local score of the last group accessed from the source S_i (line 3). Clearly, the best possible ϕ-quantile global score of any unseen group is at least t=f (L [1]... L [d]).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Therefore, if t is not smaller than the score of k-th best group, the algorithm terminates by reporting top-k groups (lines 9 to 11).

Algorithm 6 Top-k Algorithm

1: **for** each source S_i in round-robin fashion do
2: access next group $(U,V,W) \in S_i$ with smallest $LB\_S_{\phi/d}[i]$
3: $L[i] = LB\_S_{\phi/d}[U,V,W]$
4: **if** g was never seen in any other source **then**
5: compute $LB\_S_{\phi}[U,V,W]$
6: **if** $LB\_S_{\phi}[U,V,W]$ < score of k-th best group **then**
7: compute $SCORE_{\phi}[U,V,W]$
8: update top-k groups and k-th best group if required
9: t= f(L[1].....L[d])
10: **if** ≥ score of k-th best group **then**
11: **return** top-k groups

Fig. 11 Algorithm 6

Algorithm 7 Compute $SCORE_{\phi}(U,V,W)$

Input: (U, V, W) and the value of ϕ
Output: $SCORE_{\phi}(U,V,W)$
1: set p.score=0, g.score=0, i=0
2: **for** each source S_i in round-robin fashion do
3: retrieve a group g from S_i
4: s[i] = g.score[i]
5: **if** g was never seen in any other source **then**
6: **for** each group (U, V, W) round robin fashion **do**
7: **for** each pair from (U, V) round robin fashion **do**
8: Do random access on all sources to compute score of pair (U, V)
9: Score = aggregate of all values from each source of that pair
10: p.score = p.score + score
11: increment i=i+1
12: **for** each pair from (U, W) round robin fashion **do**}
13: repeat step from 8 to 10
14: p.score = p.score / total no of pairs from group (U, V, W)
15: g.score = g.score + p.score
16: **return** g.score

Fig. 10 Algorithm 7

Step 2: Compute $SCORE_{\phi}(U,V,W)$

Here we describe how to compute the score $SCORE_{\phi}(U,V,W)$ of a group of object (U, V, W) requested at line 7 of Algorithm 6 (Fig. 11). Here we describe Algorithm 6 shows the computation of $SCORE_{\phi}(U,V,W)$. Here the algorithm maintains the sources with groups of instances according to their local scores. The sources allow sorted and random accesses on groups of multi-valued objects of a given (U, V, W). The algorithm initializes an empty list L that will maintain the accessed groups in ascending order of their global scores (i.e., g.score).

Initially set p.score=0 and g.score=0. Here p.score means that the score of each pair in the multivalued pair such as (U, V) from (U, V, W) and g.score means the total score of group. The algorithm 7 (Fig. 12) proceeds with doing sorted accesses on each source S_i in a round robin fashion (line 2). For each accessed group (line 3) from a source S_i. If g was never accessed before, its score g.score is computed by doing random accesses on each other source (line 5). First of all, need select the pair (U, V) from (U, V, W) (line 6). Then select each pair from the (U, V) and compute the score from all the sources S_i (line 7-8). Score will be the aggregate of all sources (line 9). Each time p.score will be modified as p.score+score (line 10). This will be repeated for all the pairs in a group. Example next pair will be (U, W) (line 12-13). Each time the pair number is increasing (line 11).If the group count is 4 we get p.score∈ by three times of p.score. For example (U, V, W, Z) is a group we get the p.score from (U, V), (U, W) and (U, Z). At last we find the p.score by p.score divided with total no of pairs that is number of i (line 14). Then we can find totalscore from add totalscore with p.score (line 15). Then return the g.score (line 16).

IV. SIMULATION RESULTS

We are done the experiment with some real data. Here we check whether how much time to take to get the results. It based on K values as well as no of objects. The parameters are shown in Table 1. Here we use a skyband table which helps to get the result fast. Skyband table contains the queries which we used for retrieving the results. So if user need same query, need to check the query in skyband table. If that available select it from the table. It will help to get the result quickly. In each two weeks the skyband table removes the queries that are expired. It has a big benefit that is it reduces the time.

Parameter	Range
K	1,5,10,15,20,25,30,35,40,45,50
No of objects	1000,3000,5000,7000,9000,11000,13000,15000,17000,19000

Table 1. Parameters

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

A. *Scoring Functions*

Real data: The real data set consists of location data having 30000 location points from different cities in USA. Each location corresponds to the location of a city which has four attributes: two coordinates, population and state which contains the city. The cities that are in the same town are assigned the same color. We run several heterochromatic and homochromatic queries each involving two to four preferences (i.e., attributes).

B. *Top-k Pair Queries and Top-k Group Queries*

In the top-k pair queries we get the results as pairs of cities. In the case of group queries, we will get the groups of cities based on group count. It will be 3, 4, etc. The group count will be decided by user. If the city considers the same state it will give result of homochromatic queries. If the cities in different state it will be heterochromatic query. In the case of population, homo and hetero chromatic based on the range of population. If the range changes it will be have different colors.

For example, the query used to retrieve top 4 pairs where population greater than 3000. Here population greater than 3000, those cities has same color. So we retrieve such pairs which have same color. That will be homochromatic. In the case of heterochromatic the cities are in different states. Fig. 13 shows K versus Execution Time values and Fig. 14 shows No of Objects versus Execution Time values. The time will be in milliseconds.

K	Time(ms)	
	Pairs	Group
1	345361	150270
5	360743	176145
10	373530	163293
15	392629	165052
20	398844	165470
25	371278	166396
30	375648	181414
35	354082	219323
40	364336	233435
45	344203	247792
50	354657	275625

Fig.13. K Versus Time Values

No of Objects	Time(ms)	
	Pairs	Group
1000	3049	2436
3000	16463	16803
5000	58957	61049
7000	113152	158907
9000	238348	226095
11000	410902	327251
13000	633981	517230
15000	1188228	740698
17000	1209104	749548
19000	1190245	986334

Fig. 14. No of Objects Versus Execution Time Values

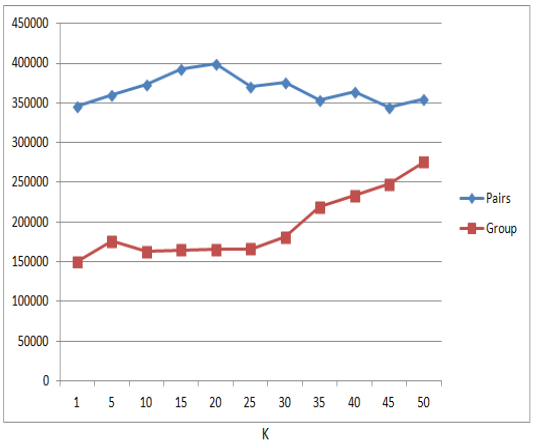

Fig. 15. K Vs Time

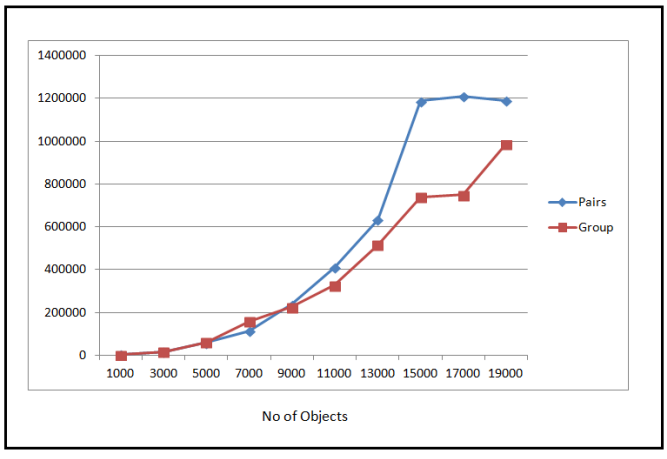

Fig. 16. No of Objects Vs Execution Time

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

According to the table we can see that execution time is reducing in the case of group queries. Fig. 15 shows the effect of increasing number of K on the queries issued on real data sets and the Fig. 16 shows the effect of increasing number of objects on the queries issued on real data sets. After the evaluation we get the result as, in pair queries which show that it will take more time than group queries. Because the group queries which has group number is low. The default no of group count is 3. Example, if we take 5 cities, it has mainly 10 pairs and 5 groups. So the execution will be reduced. So here we can say that the group queries better than pairs queries.

V. CONCLUSION AND FUTURE WORK

The unified framework is used to handle different types of queries. Mainly such a framework needs a method and algorithm. They have a lot of methods are used. So we need to find out the best method and best algorithm. Used methods are view based, sky band based, graph based and source based. So we take the best method is the combination of source and skyband based. Because it create and maintain the source based on attributes in the query. It is good method and easy to implement. These two methods will be the best method to handle the framework. All the method mainly use threshold algorithm for calculate the accurate results. So use threshold algorithm as best algorithm.

After using the method and algorithm we can identify that it is a good method to handle different types of queries. Mainly it uses in pairs and group queries. The pairs queries are single valued, multivalued and exclusive. The group queries are single valued and multivalued. The group queries are retrieve the result as group of objects depending on the group count. The pairs queries retrieve the result as pair of objects. In the experiment get the result as group queries are better than pairs queries. Because it takes less execution time compared to pairs queries.

REFERENCES

1. Dimitris Papadias,Qiongmao Shen,Yufei Taos Kyriakos Mouratidis, 'Group Nearest Neighbour Queries', Proc. IEEE 20th Intl Conf. Data Eng. (ICDE), pp.9217, April 2004.
2. K. Mouratidis, S. Bakiras, and D. Papadias, 'Continuous monitoring of top-k queries over sliding windows', Proc. ACM SIGMOD Intl Conf. Management of Data, PAGES 636-645, June 2006.
3. L.H.U.N., Mamoulis, and M.L. Yiu, 'Continuous Monitoring of Exclusive Closest Pairs', Proc. 10th Intl Symp. Advances in Spatial and Temporal Databases (SSTD), 2007.
4. L. Zou and L. Chen, 'Dominant graph: An efficient indexing structure to answer top-k queries', in Proc. IEEE ICDE, Washington, DC, USA, 2008, pp. 536545.
5. Joao B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Norv, 'efficient Processing of Top-k Spatial Preference Queries', Proc. ACM VLDB, vol. 4, no.2, November 2010.
6. W. Zhang, X. Lin, M.A. Cheema, Y. Zhang, and W. Wang, 'Quantile- Based KNN over Multi-Valued Objects', Proc. IEEE 26th Intl Conf. Data Eng. (ICDE), pp. 16-27, 2010.
7. Jianhua Feng, Guoliang Li, and Jianyong Wang, 'Finding Top-k Answers in Keyword Search over Relational Databases Using Tuple Units', IEEE Transactions on Knowledge and Data Engineering, Vol. 23,No. 112, December 2011.
8. M.A. Cheema, X. Lin, H. Wang, J. Wang, and W. Zhang, 'A Unified Approach for Computing Top-K Pairs in Multidimensional Space', Proc. IEEE 27th Intl Conf. Data Eng., pp. 1031-1042, 2011.
9. G. Sandhya and S. Kousalya Devi, "'An Adaptive Sliding Window Based Continuous Top-K Dominating Queries', in Proc. IEEE ICDE,, 2012.
10. Min Xie, Laks V.S. Lakshmanan and Peter T. Wood, 'Efficient Top-k Query Answering using Cached Views', Proc. 16th ACM Conf. Extending Database Technology, March 2013.
11. Shen Ge, Leong Hou U, Nikos Mamoulis, and David W. Cheung, 'Efficient All Top-k Computation A Unified Solution for All Top-k, Reverse Top-k and Top-m Influential Queries', IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 5, May 2013.
12. Zhian He and Eric Lo, Answering Why-Not Questions on Top-K Queries', IEEE Trans. Knowledge and Data Engineering, vol. 26, no. 6, June 2014.
13. Zhitao Shen, Muhammad Aamir Cheema, Xuemin Lin, Wenjie Zhang, and Haixun Wang, 'A Generic Framework for Top-K Pairs and Top-K Objects Queries over Sliding Windows', IEEE Trans. Knowledge and Data Engineering, preprint,vol. 26, no. 6, June 2014.
14. Muhammad Aamir Cheema, Xuemin Lin, Haixun Wang, Jianmin Wang, and Wenjie Zhang, 'A Unified Framework for Answering k Closest Pairs Queries and Variants', IEEE Trans. Knowledge and Data Engineering, vol. 26, no.11, November 2014.

BIOGRAPHY

Saxin Merona V V is a Mtech Student in the Computer Science And Engineering Department, Viswajyothi College of Engineering And Technology, Mahatma Gandhi University, Vazhakulam, Kerala, India.

Soumya Mathew is Assistant Professor in the Computer Science and Engineering Department, Viswajyothi College of Engineering And Technology, Mahatma Gandhi University, Vazhakulam, Kerala, India.