Cluster Comput
DOI 10.1007/s10586-017-1233-5

Improved continuous query plan with cluster weighted dominant querying in synthetic datasets

M. Madhankumar[1] · **C. Suresh Gnana Dhas**[2]

Received: 17 April 2017 / Revised: 5 July 2017 / Accepted: 10 July 2017
© Springer Science+Business Media, LLC 2017

Abstract The arrival of large voluminous continuous queries sets for a given query leads an insignificant insights. The elimination of certain data tuples occurs in order to balance the system load. The streaming query removes the improper data tuples and uses proper data tuples in the form of defined tables or sets. However, major drawback occurs due to unbounded streaming and inadequate access to end data. Due to such constraints, many stream processing methods makes the processed data unavailable for any applications or to the related queries of neighborhood branches. This paper avoids such problems during the process of data tuples at the generation of queries. The study uses a streaming model that executes effective query plans in continuous data. The streaming model aims reduce the communication cost and improves the scalability of continuous aggregation queries. It sub-divides the client query and executes it over data aggregators within the incoherent limit. A weighted dominant query algorithm is formulated to provide the top dominant value in each sub-query clusters. This reduces the cost for computation in synthetic databases. The experimental results proved that the proposed model with weighted dominant query algorithm effectively improves scalability by reducing the computational cost.

✉ M. Madhankumar
manomadhanbe@rediffmail.com

C. Suresh Gnana Dhas
sureshc.me@gmail.com

[1] Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu 627012, India

[2] Department of Computer Science and Engineering, Vivekanandha College of Engineering for Women, Elayampalayam, Tiruchencode, Tamil Nadu 637210, India

Keywords Continuous data · Query plan · Dataset · Data aggregation · Weighted optimization

1 Introduction

A data stream involves a massive set of data elements that gets generated continuously at rapid rate. The query in such data streams has to be executed continuously and the tuples are registered in prior. If the total number of queries register in advance, the evaluation on multiple query set is effective in real-time environment. Here, space and time constraint is a major issue that has to be evaluated [1]. Usually, the continuous query executes increased response in applications like stock exchange, detecting telecom frauds, etc. Continuous queries monitors the variations associated with the time varying data and it further monitors the query results to obtain online decision making. Here, the stream processing tackles the high rate of data that arrives from numerous source that leaves incomplete data [2]. This problem is tackled with the aggregators that aggregates the queries based on the returned data items e.g. getting average of temperature from the set of sensors. In such queries, the client identifies the coherent requirements as a part of querying.

Data incoherency: The accuracy of data is defined as incoherency of data item. This is denoted as an absolute difference between the values of source data item and the client data item. The data item is refreshed within an incoherency limits and when the incoherency limit of the data exceeds, the refresh message is sent to the user.

Data aggregation: The refreshing of data from the source to client is done through pull or push based mechanism. This study takes into account the pull based mechanism to send the data refreshes. Once the client request is made, the refresh

Published online: 13 October 2017

message is sent to the client and this reduces the computational cost. The scalable handling of data refreshes is handled with the help of pull based mechanism, which are discussed below.

The algorithm used in finding the data tuples relevant to user query plays a major role in such huge data resources. Observing and finding top dominant relevant queries is an important process in information retrieval [3]. The object or data tuple with highest rank gets the top dominance value and that is chosen to be refreshed during each interval. The ranking is effectively evaluated using similarity search system. The similarity evaluation increases the completeness of the query results to a maximum extent within query specified limit [4].

In addition, the user query can be split into sub-queries that gets distributed from sink to source nodes. The results are processed periodically by source nodes and the partial results are sent to sink node, where the final results are computed and refreshed to users [5]. Processing each sub-query is a difficult task and increases the computational cost. Hence, the use of Data Aggregators (DA) approximates the results by combining multiple data values of sub-queries.

The data aggregation helps in achieving the required efficiency on the online data tuples [6]. There are techniques that directly transforms the queries into aggregate queries [7] rather than forming sub-queries. The aggregated data is coupled and presented to user through a concentrator. This helps in maintaining the privacy and guarantees the data user identity [8]. The aggregation data inside the dataset has to be maintained within unconstrained limit and [9] uses such objective function e.g. Gossip based DA [10], Java EE7 for telecom data [11], penalty functions [12], shifted projections [13], fuzzy rules [14], enhanced fuzzy for ordered DA [15] etc.

The aggregation queries are generally a long query or a long running query, since the data changes continuously on a time varying data. The client is further interested to listen to certain notifications, if the system holds for some processing. In such cases, the continuous querying applications tolerates the inaccuracy while clustering or aggregating the data. Hence, actual results are not presented at the querying end and the satisfactory results are not obtained. To resolve the inaccuracy in results, sub-dividing the given clustering results is needed with effective optimal technique.

Hence, this paper presents a scalable and low cost technique, which answers the aggregation queries with sub-querying data aggregators over the dynamic data sets with accurate results for the given query. Here, the sub-query data aggregators helps the data sets with specific coherency, similar to the nodes, which serves the fragments of online web pages. This technique involves specifically to decompose the given client query to sub-queries and then the sub-queries are well executed over the indicated data aggregators within the incoherency bounds of the sub-queries. To attain this, an optimal technique is provided that uses weighted dominant query algorithm, which makes the sub-query to lie within the incoherency limits and provides satisfaction as per the coherency requirement of the user query. This takes less refresh messages to be sent to the user from its data aggregators. The estimation of refresh message is built with weighted dominant query algorithm, which attains the aim of the user, who attains least refresh messages.

The main contribution of the work involves: Reduction of incomplete datasets after the translation of query into sub-query DA. Translating the incoherency limits from query into sub-query using proposed algorithm with pull based mechanism and making the given query to have a specific subset.

The conventional continuous querying techniques are discussed in Sect. 2. Section 3 presents the query cost model and elimination of incomplete datasets. Section 4 describes the translation of incoherency limits using proposed algorithm. Section 5 discusses the evaluated results of proposed model over synthetic datasets. Section 6 concludes the paper with future work to be improved in current approach.

2 Related works

Lee et al. attribute selection technique in multiple queries using run-time matching algorithm [1]. Patroumpas and Sellis used windowed patterns using monotonic semantic functions. Jeon et al. [5] processed multiple queries using on the fly mechanism. Han et al. [6] used p, ε-approximate with join aggregation procedure. Lange and Naumann used similarity search in non-static environment in real time datasets [4].

Xu and Pottinger uses query translation to solve NP-hard problems using decomposition aggregation with three-tier structure [7]. Bae et al. used DA in smart grid technology using homomorphic encryption algorithm [8]. Cui et al. used probability distribution function that finds the boundary collisions using entropy histogram threshold [9]. Panah et al. used watermarking technique in real and synthetic sensor data environment [16] for maintaining the confidentiality. Sagy et al. obtained top scoring objects using top-k algorithm with four phases in distributed database [3].

Mishra and Singh [17] proposed teaching learner parameterless optimization in unconstrained databases. Bustince et al. discussed improper quasi-penalty functions to average continuous data aggregation and non-monotonic averaging functions [12].

Mousazadeh and Ladani proposed pull-gossip algorithm in synthetic datasets [10]. Nechifor et al. proposed DA over electrical application [11]. Wu et al. proposed scalable privacy preservation over big data aggregation [18]. Duque [13] proposed shifted projections over affine spaces.

2.1 Data-dependent model

Elkano et al. proposed fuzzy aggregation function for improved effectiveness with reduced errors [14]. Garg and Garg proposed ordered weighted fuzzy aggregation function in time domain that reduces the average error rate [15]. Kao proposed relational model that decomposes the networks into weighted function with better data aggregation [19].

3 Problem formulation

The value of weighted continuous aggregation query at time t is defined by:

$$v_q(t) = \sum_{i=1}^{n_q} v_{qi} w_{qi} \tag{1}$$

V_q represents the client query (q) value that involves datasets (n_q) with with ith dataset weight (w_{qi}). If the query result of Eq. (1) is supplied to user at regular intervals within incoherency bound then,

$$\left| \sum_{i=1}^{n_q} \left(v_{qi}\,(t) - u_{qi}\,(t) \right) w_{qi} \right| \leq C_q \tag{2}$$

The incoherency bound is violated if source data value is changed, so the updated value is updated to user or refreshed at regular intervals. The incoherency bound of ith dataset, C_{qi} is satisfied using:

$$\sum_{i=1}^{n_q} (C_{qi} w_{qi}) \leq C_q \tag{3}$$

When a query is generated, the query incoherency bound is translated to individual dataset as sub-query incoherency bound. An optimal algorithm is required to translate these coherency bounds through (i) Division of client query into total number of sub-queries. (ii) Translating the incoherency bound into sub-queries. (iii) Execution of sub-queries over data aggregators (*da*) and (iv) estimation of cost by executing the total refreshes sent to client.

4 Proposed model

The major parameter that affects the performance of continuous queries is the cost incurred with communication, CPU and more specifically intra-communication. Parallel or distributed processing of these continuous queries could actually reduce the communication cost. However, large processing of distributed data reduces the scalability of the model. To solve this, an explicit model that deals with removal of incomplete abundant data. The elimination of such data helps really in parallelized or distributed processing of data tuples in an effective way. The query processing plan starts with this and then followed by this is the formation of cluster related to threshold value from the user query. A similarity search procedure is included that ranks the data tuples based on the similarity comparison with user query. The cluster region is formed depending on the ranks of the data tuples. The lesser threshold regions are further eliminated and regions with higher threshold values are refreshed to users. The selection of dominant data tuple from the synthetic dataset is carried out with weighted dominant query algorithm.

Once the similarity search and clustering procedure is done, the data value from user query is translated to higher threshold regions. The initial procedure of threshold calculation is done with ranking procedure and the incoherency limits are distributed to the data tuples. The algorithm is operated based on the incoherency limit and selection of top data tuple is based on rank and incoherency limit. The optimal solution in estimating the top data tuple improves the system for multiple data queries from the user interface. The elimination of incomplete data and lower threshold data increases the exploration space. The space required for parallel processing of next user query is achieved with this model.

4.1 Multi-objective function

Since, cost is the main parameter considered, the total cost associated with processing user queries are affinity cost, localization cost and processing cost.

Affinity cost: gives the heterogeneous degree in generation of results from each clusters. The query plan with least clusters are considered better for effective results generation. The clusters with higher concentrations are considered in relevance with different query plan, in case of multiple user queries. The affinity cost is computed as:

$$C_a = \sum_{i=1}^{M} \frac{k_i}{N} \left(1 - \frac{k_i}{N} \right) \tag{4}$$

M denotes the total clusters accessed by query plan; K_i denotes the accessing of ith site by query plan and N denotes the total relations accessible to query plan.

Localization cost: The Localization initially quantify the degree of communication between two different sites in a query plans (QP) and then it decides the control site for

query answering of respective QP. Evaluation of the communication cost for QP, Relation-Site Matrix (RSM) and Relation-Size vector (number of tuples), An QP is mapped in a query graph on which sites as nodes and edges as join selectivity (QLC values) are mapped. Computing join selectivity for a QP on dynamic and is distributed environment is biggest challenge. QLC between two sites S_1 to S_2 represents the ratio of size of relation at that site and sum of sizes of total relations in FROM clause of query. For a given query plan the minimum QLC can is computed as follows,

The localization cost between the data tuple in clusters is calculated as the ratio of size of cluster relation and sum of total relation size. The extent of communication between the data tuples in each cluster decides the central data for refreshing or answer user query. The query plan is mapped into graphs, where nodes and edges are mapped.

$$C_l = \overset{i=Nr}{min}\left[\sum_{j=1\&\&j\neq 1}^{M}\frac{size\left(Rs_j\right)}{\sum_{k=1}^{Nr} size\left(R_k\right)}\right] \tag{5}$$

C_l is the localization cost, *min* evaluates minimum value of i from 1 to Nr. Nr is total relations in clusters. $size\ (R_k)$ and $size\ (Rs_j)$ represents the size or number of tuples with R_k and S_j. R_s defines the total relations on local site.

Processing cost: considers the relations stored in cluster, selection of operators and quantifying the value of processing cost to query plan. It also considers the cost of sub and central queries and memory for sending data to user at repeated intervals.

$$C_p = \frac{N_t S_r}{\sum_{k=1}^{Nr} N_t\ (k)} \tag{6}$$

C_p is the relation processing cost and S_r represents selection of result at cluster R. The local processing cost is given as:

$$C_{QP} = \sum_{k=1}^{s_{qp}-1}\left(Cc_{QP}\ (k)\right) + \left(Cl_{QP}^{C_p(i)}\right) \tag{7}$$

where,

$$Cl_{QP} = \max_{i,j\in 1\text{to}Nr}\frac{\left[N_t\left(join\left(R_i, R_j\right)\right) S_j\left(join\left(R_i, R_j\right)\right)\right]}{\sum^{Nr} N_t\ (k)}$$

and

$$Cc_{QP} = \max_{i,j\in 1\text{to}R_s}\left[C_p\ (i)\right]$$

Cc represents the central data and *Cl* represents the remaining data in cluster.

4.2 Weighted dominant query (WDQ) algorithm

The weighted algorithm uses pull based mechanism that eliminates the incomplete data in the network. The data tuple sends a request and then obtains the user query value is translated to the sub-query. This removes the incomplete data present in the system and similar procedure is followed for other data tuple or sub-query. The sub-query value less than user query value is eliminated from the query space.

Algorithm 1: Elimination of irrelevant data tuples

Find all the data in the space
User inputs the query
input (k, M, i, N, C_i, N_r, R_k, Rs_j, N_t, S_r, C_p, C_c, C_l); // *input functions*
Calculate C_a,C_l, C_q, C_{qp} // *cost calculation for given query*
Find the overall fitness function = min(C_a,C_l, C_q, C_{qp}) // *calculation of cost function*
Generate the query plan as per user query // *refer algorithm* 1 of [1]
Clustering the Query_value (V_q) to sub-query (V_{sq}) // *cluster the query value into sub-queries*
If Fitness value of (V_{sq}) < Threshold level T // *when the value of cost function for a given query is greater than the sub-queries following operations are considered*
Eliminate data tuple with $V_{sq} \leq T$ // *discard the sub-queries limits lesser than the cost function or incoherency limits*
Else
Include $V_{sq} > T$ // *include the sub-queries range exceeds the incoherency limit*
End

The important factor that decides the total number of cluster in the space is the range of query value. The range is divided into five zones based on the value obtained. The V_{sq} is divided by giving a specific range to it based on the divided zones. V_{sq} with values specific to zones are sent to the respective zones. An inter-communication gets established between the data tuples or the sub-queries in the zones. This prepares the data to be processed for dominant one in the space.

4.3 Incoherency bound

The data tuple has to be circulated to an incoherency limit, C or new data value will be moved if value of sub-query is more than C (previous pushed value). The data tuple is considered as discrete random process and there is a correlation between the past and current steps. The data source or central data pushes the data tuple, where the value is less than T. The value is estimated using mean value of random process. The client estimates the tuple based on central data value and pushes the item, when the value of the tuple is less than T. Here the value is estimated as zero random process. The source is modelled as random average process with value known at client. Thus using Chebyshev's inequality, the data tuple gets refreshed is inversely proportional to the square of incoherency limit. This is defined as:

$$P\left(|v\left(t\right)-u\left(t\right)|>C\right)\alpha 1/C^2 \tag{8}$$

Algorithm 2 gives the pseudo code for WDQ algorithm. When a new data arrives, the central data value gets updated by multiplying T with $(1 - new_value)$. Then remove the existing element and insert the new updated data. If data expires, check if the value of T is greater than the data value. The new data tuples in a cluster is considered as an input value and the main notion is to find the dominance value of these five zones. Initially, the dominance in first zone is calculated and then for other four zones. Here, a comparison is made between each tuples and the tuple with higher value is recorded and other values are discarded. The time complexity is reduced by a factor of $O\left(|D^2|\right)$. The dominance is recorded in two loop procedure and finally the dominance in compared between all the dominated tuple in each zone/cluster.

5 Performance evaluation

A set of experiments is conducted for evaluating the effectiveness of the proposed model. The effectiveness is compared

Algorithm 2: Weighted Dominant Query (WDQ) Algorithm

While a new data arrives **do**
If data_value < T
Then insert value > T (central_data (Past and Present_value))
Else
data_expires (old_value)
Insert value > T (central_data (Past and Present_value))
End while
Given: New Dataset with values, V_{sq}
Find: Calculate Dominance(t) for each data tuple, D_{sq}
For $z = 1 : n$
For $v_n \leftarrow 1$ to |D| **do**
    Dominance(t) = 0
    For $u_n \leftarrow 1$ to |D| **do**
    if t_u dominates t_v
    Then Dominance (t_v) = Dominance (t_v) + 1
    End For
End For
$n = n + 1$
If $n = 5$
Dominance (T_v) = *Sort* (t_v (n)) /* Sort in Ascending order */
Select the row 1 with highest dominance value
Else repeat the loop
End if
End While
Refresh the dominated data value to client.

Table 1 Parameters of synthetic dataset

Parameter	Range
Total number of tuples	1000–6000
Dimension (d)	2–6
Combinatorial function	Average
Distribution	Uniform, correlated and uncorrelated
Density	10–50 in multiples of 10
Cardinality	100K, 200K, 300K, 400K, 500K

in terms of its computational time or processing time. Secondly, the precision is measured that is calculated between the original user query value and the dominated value from dataset. The synthetic datasets of varying dimensions, size and skewness is generated and client query value is regarded as cardinality (Table 1).

Effectiveness: The experiments are carried out between the total datasets and its execution time for uniform, correlated and uncorrelated datasets. The processing time is measured in logarithm scale and pre-processing avoids the unnecessary dominance over entire datasets. This results in reduced execution time and it increases as the dataset size increases. It is also observed that the higher execution time is obtained uncorrelated datasets. This is due to multiple availability of combinatorial data tuples in uncorrelated datasets. The Fig. 1a–c shows that the proposed model executes the given query in a less executed time than other methods over uniform, correlated and uncorrelated datasets. Out of this, the execution time over correlated synthetic dataset performs well than uniform and uncorrelated datasets.

Precision rate: The precision rate is defined as the ratio of value of original user query to the value generated or refreshed after dominance calculation. The precision is calculated over all the three datasets i.e. uniform, correlated and uncorrelated. It is proved from Fig. 2 that the correlated synthetic dataset has higher precision than uniform and uncorrelated datasets.

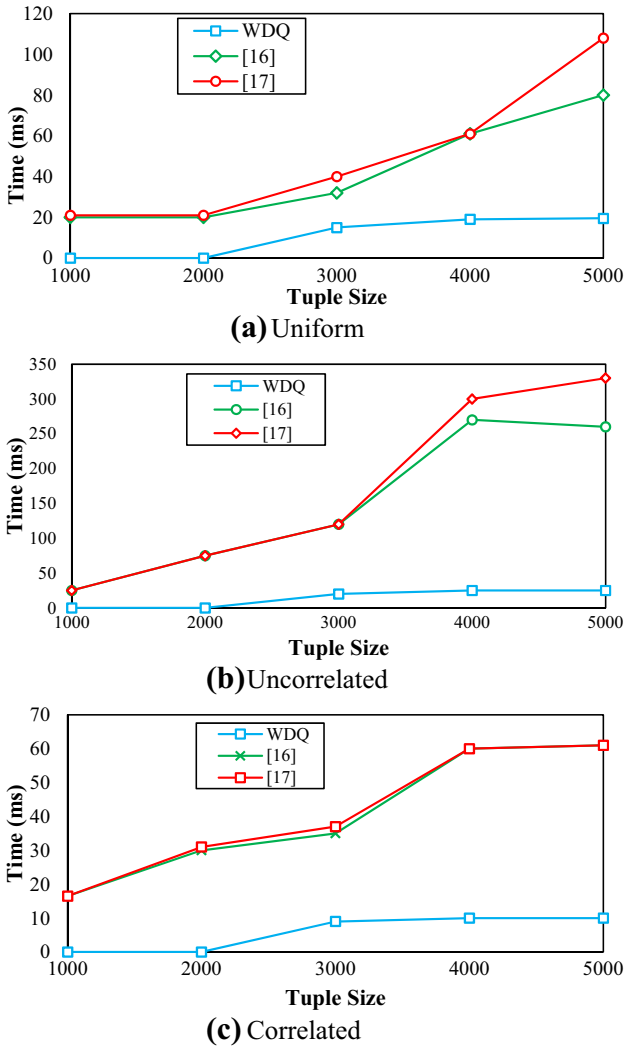


Fig. 1 Computational time of proposed model over synthetic datasets. **a** Uniform, **b** uncorrelated, **c** correlated

6 Discussions

The results of Fig. 1a shows the reduced query time with reduced number of refresh messages by the proposed method. Since, the pull mechanism is used in WDQ, the querying results based on given client query, actually matches with reduced number of time. Likewise, the uncorrelated synthetic datasets shown in Fig. 1b provides reduced refresh message w.r.t the client's query and the Fig. 1c also gives the similar output. The main reason is that the incoherency bounds of the given queries are reduced due to the use of pull based mechanism with weighted dominated query search mechanism. Also, elimination of irrelevant data queries as per the given dominated search in Algorithm 1, proved the proposed system is better than the existing techniques. Hence, the communication cost is reduced with optimization used to carry out the execution of incoherency bounds. The results of conventional techniques uses more refresh messages to execute the query plan as per the user query.

Further, precision metric provides the results related to the accuracy of the data using the proposed technique. Here, the execution of precision results are carried out with different number of tuples, which are pre-processed initially and then used to carry out the pull based mechanism to execute the query plan with reduced refresh messages. It is seen that for uniform, correlated and uncorrelated synthetic data, the results are better than the other methods. It proves that there is an increased precision rate due to reduced refresh messages.

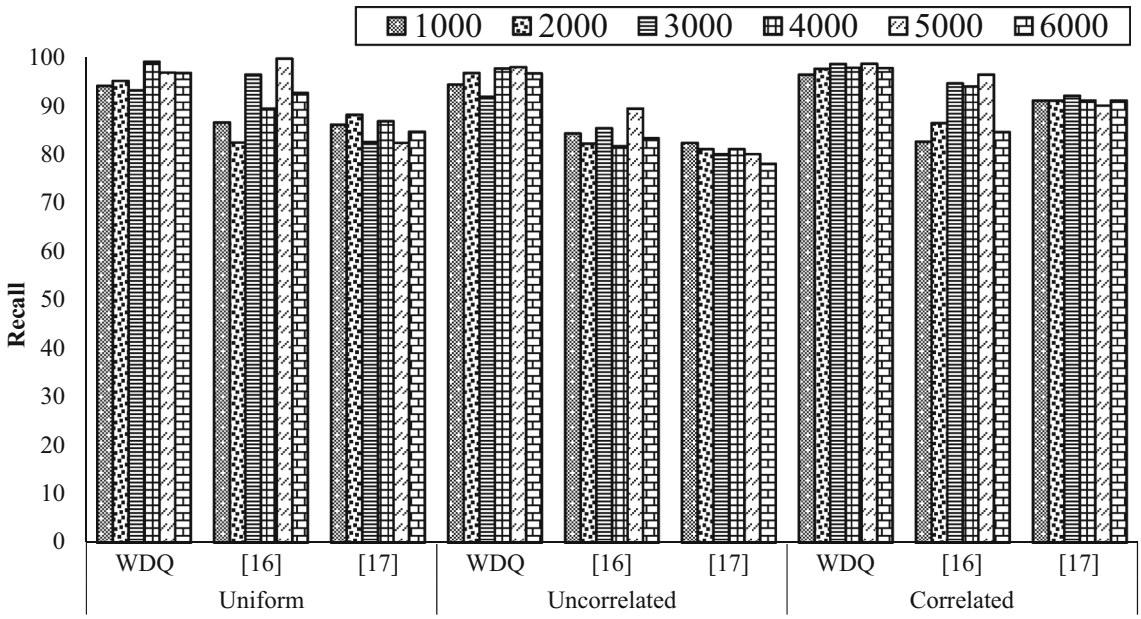


Fig. 2 Precision rate of proposed model over synthetic datasets

It could also be inferred that the conventional methods has a reduced rate of 1.5% precision than the proposed query plan.

7 Conclusion

The query execution cost is calculated in terms of execution time and precision rate. The timing parameter proves that number of refreshes sent to client is reduced considerably. This is proved in terms of lesser execution time and sending dominance value to the client. The data aggregators proved its effectiveness by disseminating the data tuples into their respective zones within incoherency limit. For optimal process, the client query is divided into sub-queries and processing on each data tuple is done, eliminating incomplete data. Performance evaluation over synthetic datasets proved that the proposed model performs well by reducing the time to one-third than previous method. The WDQ algorithm is proved to be an effective approach in the query databases. In this technique, the data aggregators satisfies the client query effectively and improved the scalability of system by reducing the processing time. Finally, it could be concluded that the proposed method executes the continuous aggregate queries for multi-data. This reduces the total refresh message and reduces the overhead due to reduced messages. Further, certain clustering techniques used to segregate the given query results to aggregate queries can be carried out with grey-cloud clustering [20] or reconstruction method with clustering [21] or local mean clustering [22].

References

1. Lee, H.H., Yun, E.W., Lee, W.S.: Attribute-based evaluation of multiple continuous queries for filtering incoming tuples of a data stream. Inf. Sci. **178**(11), 2416–2432 (2008)
2. Patroumpas, K., Sellis, T.: Maintaining consistent results of continuous queries under diverse window specifications. Inf. Syst. **36**(1), 42–61 (2011)
3. Sagy, G., Sharfman, I., Keren, D., Schuster, A.: Top-k vectorial aggregation queries in a distributed environment. J. Parallel Distrib. Comput. **71**(2), 302–315 (2011)
4. Lange, D., Naumann, F.: Cost-aware query planning for similarity search. Inf. Syst. **38**(4), 455–469 (2013)
5. Jeon, J.H., Lee, K.Y., Kim, M.H.: Communication-efficient processing of multiple continuous aggregate queries. Inf. Sci. **284**, 1–17 (2014)
6. Han, X., Li, J., Gao, H.: Efficiently processing (p, ε)-approximate join aggregation on massive data. Inf. Sci. **278**, 773–792 (2014)
7. Xu, J., Pottinger, R.: Integrating domain heterogeneous data sources using decomposition aggregation queries. Inf. Syst. **39**, 80–107 (2014)
8. Bae, M., Kim, K., Kim, H.: Preserving privacy and efficiency in data communication and aggregation for AMI network. J. Netw. Comput. Appl. **59**, 333–344 (2016)
9. Cui, G., Wang, X., Kwon, D.W.: A framework of boundary collision data aggregation into neighbourhoods. Accid. Anal. Prev. **83**, 1–17 (2015)
10. Mousazadeh, M., Ladani, B.T.: Gossip-based data aggregation in hostile environments. Comput. Commun. **62**, 1–12 (2015)
11. Nechifor, A., Albu, M., Hair, R., Terzija, V.: A flexible platform for synchronized measurements, data aggregation and information retrieval. Electr. Power Syst. Res. **120**, 20–31 (2015)
12. Bustince, H., Beliakov, G., Dimuroa, G.P., Bedregal, B., Mesiare, R.: On the definition of penalty functions in data aggregation. Fuzzy Sets Syst. **323**, 1–18 (2017)
13. Fernández-Duque, D.: Perfectly secure data aggregation via shifted projections. Inf. Sci. **354**, 153–164 (2016)

14. Elkano, M., Sanz, J.A., Galar, M., Pękala, B., Bentkowska, U., Bustince, H.: Composition of interval-valued fuzzy relations using aggregation functions. Inf. Sci. **369**, 690–703 (2016)
15. Garg, B., Garg, R.: Enhanced accuracy of fuzzy time series model using ordered weighted aggregation. Appl. Soft Comput. **48**, 265–280 (2016)
16. Panah, A.S., van Schyndel, R., Sellis, T.: Towards an asynchronous aggregation-capable watermark for end-to-end protection of big data streams. Future Gener. Comput. Syst. **72**, 288–304 (2017)
17. Mishra, V., Singh, V.: Generating optimal query plans for distributed query processing using teacher-learner based optimization. Proc. Comput. Sci. **54**, 281–290 (2015)
18. Wu, D., Yang, B., Wang, R.: Scalable privacy-preserving big data aggregation mechanism. Digit. Commun. Netw. **2**(3), 122–129 (2016)
19. Kao, C.: Efficiency decomposition and aggregation in network data envelopment analysis. Eur. J. Oper. Res. **255**, 778–786 (2016)
20. Ma, A.Z., Fan, H.M., Zhang, E.Y.: Cruise homeport location selection evaluation based on grey-cloud clustering model. Curr. Issues Tour. **7**, 1–27 (2017)
21. Wu, E., Li, B., Stathis, J.: June. Low-κ and High-κ breakdown statistics with variability: clustering model versus reconstruction methodology. In: 2015 IEEE 22nd International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA), pp. 31–34. IEEE (2015)
22. Zhang, S., Wang, H., Huang, W.: Two-stage plant species recognition by local mean clustering and Weighted sparse representation classification. Clust. Comput. **20**, 1517–1525 (2017)



M. Madhankumar completed B.E. (CSE) in the year of 2006, Thirumalai Engineering College, Kilambi, Kanchipuram, Tamilnadu. M.E. (CSE) in the year of 2009, Arulmigu Meenakshi Ammal Engineering College, Kanchipuram, Tamilnadu. Now pursuing Doctorate in CSE since 2013, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu, India. Area of Interest: Data Mining, Cluster Computing, Soft Computing, Distributed System, Artificial Intelligence, Query processing and Optimization.



C. Suresh Gnana Dhas has more than 28 years of experience in academic and research works. He is strong in the area of Artificial Intelligence, Data Storage and Retrieval, MANET, Image processing, Cryptography and Network security. Suresh has published papers with professional cycling teams and various research bodies. He has delivered speeches in many reputed institutions and Universities. He is the organizing chair of many International conferences. He has completed B.E, M.E and Ph.D. in Computer Science and Engineering.