Advanced Science and Technology Letters
Vol.63 (NGCIT 2014), pp.125-129
http://dx.doi.org/10.14257/astl.2014.63.26

A New Continuous Query Evaluation Scheme for Multiple Data Streams

Hyun-Ho Lee[1], Hong-Kyu Park[2], Jin-Chul Park[3], Kil-Hong Joo[4]

[1] Dept. of Military Science, Yeonsung University, South Korea
[2] Samsung Electronics Co. Ltd., South Korea
[3] Dept. of Computer Science and Engineering, Hanyang University, South Korea
[4] Dept. of Computer Education, Gyeongin National University of Education, South Korea
hhlee@yeonsung.ac.kr[1], gladiator11@hanmail.net[2], jin1chul@nate.com[3],
khjoo@ginue.ac.kr[4]

Abstract. Data streams, which are a representative type of big data, are massive unbounded sequences of data elements that are continuously generated at a rapid rate. In this paper, a new scheme called a 2-phase query evaluation scheme (2PE) is proposed for a multi-way join continuous query over multiple data streams. The first phase identifies the candidate tuples that can satisfy a given query. Subsequently, the second phase judges whether each of the candidate tuples satisfies the query in order to materialize its final results.

Keywords: data stream, continous query, 2PE, Matrix-based.

1 Introduction

Data streams are characterized by a massive unbounded sequence of data elements that are continuously generated at a rapid rate [1,2]. As a representative type of big data, it emerges in many applications, such as web click monitoring, sensor data processing, network traffic analysis, telephone record analysis, and multimedia. Ongoing changes in the rapid data streams in a data stream management system (DSMS) may be monitored using a number of pre-registered queries that report results continuously as new data elements of the data streams arrive. Such queries are called *continuous queries* [2,3,4]. Due to the strict time constraints of the continuous queries in a stream environment, query optimization processes are essential.

In this paper, a new scheme called a 2-phase query evaluation (2PE) scheme is proposed for a multi-way join continuous query. Given a continuous query over multiple source streams, the first phase of the proposed scheme detects whether newly arrived stream tuples satisfy the query by identifying the maximum number of the candidate tuples that can satisfy it. It is called a pre-processing phase. In case of a detection-only query, evaluating only the preprocessing phase is enough to solve the query if there are not false detected tuples (false positive errors). The second phase identifies the stream tuples that satisfy the query among the candidate tuples specified in the previous phase. Subsequently, it materializes the final results of the query

ISSN: 2287-1233 ASTL

explicitly by concatenating the satisfying tuples that are identified from the source streams. It is called an evaluation phase.

3 reparations

Given a multi-way join continuous query for multiple data streams, its multiple join predicates can be represented by an undirected graph called as a *join graph*. All the join predicates of the query can be arranged in the specific order. It is called a *query walk*, defined in Definition 1.

Definition 1. **(Query walk)** Given a join graph $G_Q = (N_Q, E_Q)$ for an n-way join query Q with n data streams $S = \{S_1, S_2,\ldots S_n\}$, a *query walk* $\Psi_Q = <v_1\ e_1\ v_2\ e_2\ \ldots e_{m-1}\ v_m>$ ($n \le m$) is a sequence of nodes $v_i \in N_Q$ and edges $e_i \in E_Q$ ($1 \le i \le m$). The node v_i represents the stream S_i. The edge e_i represents the join attribute of the stream v_i and v_{i+1}, satisfying that every edge in E_Q appears at least once in the walk Ψ_Q. □

Like a general hash join algorithm, the main goal of the scheme is narrowing the range of the candidates for successfully joined tuples by classifying data tuples based on the hashed results for the values of their join attributes. Given an n-way join query Q and a hash function h for a join predicate $R.a=S.b$ of the query Q on the streams R and S, the tuple x of R and the tuple y of S are *buddy tuples* if the hashing results of the attributes $x.a$ and $y.b$ are the same($h(x.a)=h(y.b)$). For the tuples of the streams R and S, if their join attribute's values are the same, they are buddy tuples because the hashing results of their join attribute are also the same. In other word, if they are not buddy, their join attribute's values cannot be the same. Accordingly, a set of buddy tuples is a candidate for the matching result of the predicate $R.a=S.b$. Especially, a set of n buddy tuples each of which is a tuple of a single data stream is a candidate for the final maching result of the query Q. Such a set is called a *complete connecting sequence*, formally defined in Definition 2.

Definition 2. (Connecting sequence) Given a query walk $\Psi_Q = <v_1\ e_1\ldots e_{k-1}\ v_k\ e_k\ldots e_{m-1}\ v_m>$ of a join graph $G(N_Q,E_Q)$ for an n-way join query Q with n multiple data streams $\{S_1,S_2,\ldots,S_n\}$ ($n \le m$), a *k-partial connecting sequence* of the length-k prefix Ψ_Q^k is a sequence of k tuples $<c_1,c_2,\ldots c_k>$ where c_i ($1 \le i \le k$) is a tuple of the stream corresponding to the node $v_i \in \Psi_Q^k$, satisfying that any adjacent tuples c_{i-1} and c_i($1 < i \le k$) are buddy tuples for the join predicate that a corresponding edge $e_i \in \Psi_Q^k$ represents. A *complete connecting sequence* is an m-partial connecting sequence, which covers the whole query walk Ψ_Q. □

If a tuple of a single data stream S_j ($1 \le j \le n$) is a part of at least one complete connecting sequence, it is called a *connecting tuple* of the stream S_j. Due to hash collision, all of the connecting tuples of the stream S_j does not included in the final matching result of the query Q since join attributes' values can be different from one another although their hashing result is the same. Accordingly, the final matching result of the query Q can be obtained by justifying whether the join attributes' values of all the connecting tuples that consists of a complete connecting sequence satisfy the join predicates of the query Q where they are employed.

In the proposed approach, two different types of matrix-based synopses, called *window synopsis* (*W*) and *window index synopsis* (*WI*), are employed. They are the

same in terms of their structures. However, they are different from each other in respect of the content of their entries. The former maintains the number of tuples assigned to the matrix entry; it is employed in the first phase of 2PE (preprocessing phase) to measure the possible volume of successful query–result tuples. Meanwhile, the latter identifies each of those tuples that are assigned to the entry by maintaining a list of their addresses. This is employed in the second phase of 2PE (the evaluation phase) in order to identify successfully query-result tuples exactly. Consequently, each value of a window synopsis entry is equal to the length of the list stored in the corresponding window index synopsis entry.

4 PE: 2-Phase Query Evaluation Scheme

The proposed query evaluation scheme 2PE is based on a typical sliding-window join operation [3,4,5] over infinite multiple data streams. In general, it is composed of three tasks: *inserting*, *probing* and *invalidating*. Given a join query over two operand streams R_1 and R_2, whenever a new tuple of the stream R_1 arrives, the *inserting* task adds it to the sliding window of R_1 first. Subsequently, in the *probing* task, it is compared to each of the tuples within the current window of R_2.Whenever these two tuples meets the join condition of the given query, they are combined to produce the successful join result. Finally, the *invalidating* task removes all the old-dated tuples in R_1. These steps are also symmetrically executed when a new tuple of the stream R_2 arrives. Based on the sliding-window scheme, 2PE consists of two phases: pre-processing phase and evaluation phase. Given a multi-way join query Q, the pre-processing phase outputs the candidates of the result tuples for the query Q using the synopsis structures and operations proposed in the previous section. Subsequently, the evaluation phase finally produces the successful result tuples for the query Q by evaluating the candidate result tuples that are acquired in the previous phase. Like the sliding-window scheme, 2PE processes the query Q symmetrically by the unit of a set of the newly arrived tuples of the specific operand stream. It is called a *batch*. a series of tasks – inserting, probing and invalidating – is performed per a batch to produce the result tuples of the query Q. A batch consists of at least one tuple and its size (the number of tuples) can be adjusted depending on the current workload.

4.1 Preprocessing Phase

For each batch of multiple data streams, a pair of a *batch synopsis* (*B*) and a *batch index synopsis* (*BI*) is constructed. The newly arrived tuples of the current batch are inserted into the proposed synopsis *B* and *BI* along with the window synopses *W* and *WI*. The batch synopsis and batch index synopsis are structurally identical to a window synopsis and a window index synopsis described in Section 3., respectively. Given a query walk Ψ_Q of a join graph $G(N_Q,E_Q)$ for an n-way join query Q over n multiple data streams $\{S_1, S_{2,\ldots,} S_n\}$, in order to evaluate the synopsis product $E(\Psi_Q)$ for a new batch of a source stream S_x $(1 \le x \le n)$ denoted by ΔS_x, the window synopsis Ws_x among the window synopses formulating $E(\Psi_Q)$ is replaced with the batch

synopsis Bs_x. This modified synopsis product is called as a *batched synopsis product*, denoted by $E(\Psi_Q|\Delta S_x)$.

As described above, evaluating the batched synopsis product $E(\Psi_Q|\Delta S_x)$ consists of the following three tasks: (1) (*inserting*) Insert the batch ΔS_x into the proposed synopses *W*, *WI*, *B* and *BI*; (2) (*probing*) Evaluate the product $E(\Psi_Q|\Delta S_x)$; (3) (*Invalidating*) Remove the batch ΔS_x from the synopses *W*, *WI*, *B* and *BI*;

4.2 Evaluation Phase

The evaluation phase generates the accurate set of the final result tuples for the batched synopsis product $E(\Psi_Q|\Delta S_x)$. It is further divided into two steps: *identification* and *materialization*. The identification step refines the entries of all the intermediate result vectors of the product $E(\Psi_Q|\Delta S_x)$ in order to discard those entries that cannot lead to any complete connecting sequence. Subsequently, the materialization step produces the final result tuples for the batch ΔS_x by only concatenating the successful tuples of each data stream based on the refined result vectors.

5 Conclusions

This paper proposes a new scheme called a 2-phase query evaluation(2PE) scheme in order to evaluating a multi-way join continuous query for mutliple data streams. The proposed method can be applicable to any type of data stream because it is based on general techniques concerned with evaluating continuous queries for data streams. In the future work, it can be extended to evaluating multiple continuous queries not a single continous query.

Acknowledgments. This research was supported by Basic Science Research Programs (NRF-2012R1A1A2009170) through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning

References

1. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom: Models and issues in data stream systems, Proceedings of ACM Symposium on Principles of Database Systems, pp. 1--16, (2002)
2. S. Madden, M. Shah, J. M. Hellerstein and V. Raman: Continuously Adaptive Continuous Queries over Streams, ACM SIGMOD International Conference on Management of data, pp. 49--60, (2002)
3. R. E. Kalman: A new approach to linear filtering and prediction problems, Journal of Basic Engineering, vol. 82, pp. 35--45, (1960)

4. Lukasz Golab and M. Tamer Ozsu: Processing sliding window multi-joins in continuous queries over data streams, sources In Proc. Int. Conf. on Very large data bases, pp.500--511, (2003)
5. J. Gomes, H. Choi: Adaptive optimization of join trees for multi-join queries over sensor streams, Information Fusion, Vol.9, pp. 412--424, (2008)

